Towards Computational Efficiency of Next Generation Multimedia Systems

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der Fakultät für Informatik

des Karlsruher Institut für Technologie (KIT)

genehmigte

Dissertation

von

Muhammad Usman Karim Khan

Tag der mündlichen Prüfung: 21.Dec.2015

Referent: Prof. Dr. Jörg Henkel, Karlsruhe Institute of Technology (KIT),
Fakultät für Informatik, Chair for Embedded Systems

Korreferent: Prof. Dr. Wolfgang Karl, Karlsruhe Institute of Technology (KIT),
Fakultät für Informatik, Chair for Computer Architecture and Parallel Processing

Muhammad Usman Karim Khan
Schneidemühler Str. 12-E
76139 Karlsruhe

Hiermit erkläre ich an Eides statt, dass ich die von mir vorgelegte Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen, Internet-Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen – die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

_____

Muhammad Usman Karim Khan

Acknowledgement

For constantly motivating, supervising and inspiring me, along with his continuous guidance, I would like to present my gratefulness to my supervisor, Prof. Dr. Jörg Henkel. I consider myself greatly fortunate to be a part of Chair for Embedded Systems (CES) under the supervision of Prof. Henkel, who has successfully created an environment for excellence and competition in the lab. Without a doubt, this has provided me the impetus to test my research potential and to polish my skills, which will positively define my future course. I am also thankful to Prof. Dr. Wolfgang Karl for accepting to be my co-advisor.

I would also like to present high regards to my team-lead and technical supervisor, Dr. Shafique. He has constantly shared his vast set of expertise and knowledge (and some snacks) with me during my tenure in CES. His support, motivation, critical reviews and data-analysis has undoubtedly increased the quality of this thesis.

I would like to thank my CES colleagues and peers for extending my research horizons. The numerous research projects going on in CES during my stay in CES has helped to expand my knowledge of different aspects of computing systems. The technical discussion and informal exchanges with my colleagues at work was a great pleasure. I would give a special mention to my office roommates who have been extremely supportive, kind enough to translate German and to talk in German on my behalf: Martin Hass, Anton Ivanov, Anuj Pathania and Hongyang Zhang. Moreover, I would particularly mention the lunch time discussions and light moments shared with my colleague Fazal Hameed.

My parents, Karim Khan and Sardar Bibi, paved sound foundation of my personality and future, and words cannot describe their contributions, and the sacrifices they had made for my success. My special thanks goes to my awesome wife Rizwana Arshad, for her patience and prayers, and for looking after me and our two little boys (Affan and Hayyan) in this tough phase of our lives.

List of Publications Included in this Thesis

[1] M. Shafique, M. U. K. Khan and J. Henkel, "Content-Aware Low-Power Configurable Aging Mitigation for SRAM Memories," in *IEEE Transactions on Computers* (**TC**), 2016.

[2] M. U. K. Khan, M. Shafique and J. Henkel, "Power-Efficient Workload Balancing for Video Applications," in *IEEE Transactions on Very Large Scale Integration Systems* (**TVLSI**), 2015.

[3] M. U. K. Khan, M. Shafique, L. Bauer and J. Henkel, "Multicast FullHD H.264 Intra Video Encoder Architecture," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (**TCAD**), 2015.

[4] M. U. K. Khan, M. Shafique, A. Gupta, T. Schumann and J. Henkel, "Power-Efficient Load-Balancing on Heterogeneous Computing Platforms," in *Design, Automation and Test in Europe Conference* (**DATE**), 2016.

[5] M. U. K. Khan, M. Shafique and J. Henkel, "Hierarchical Power Budgeting for Dark Silicon Chips," in *ACM/IEEE International Symposium on Low Power Electronics and Design* (**ISLPED**), pp. 213-218, 2015.

[6] M. Shafique, M. U. K. Khan, Orcun Tuefek and J. Henkel, "EnAAM: Energy-Efficient Anti-Aging for On-Chip Video Memories," in *ACM/EDAC/IEEE Design Automation Conference* (**DAC**), pp. 1-6, 2015.

[7] J. Henkel, M. U. K. Khan and M. Shafique, "Energy-Efficient Multimedia Systems for High Efficiency Video Coding," in *IEEE International Symposium on Circuits and Systems* (**ISCAS**), pp. 613-616, 2015 (reviewed Special Session paper).

[8] M. U. K. Khan, M. Shafique and J. Henkel, "Power-Efficient Accelerator Allocation in Adaptive Dark Silicon Many-Core Systems," in *Design, Automation and Test in Europe Conference & Exhibition* (**DATE**), pp. 916-919, 2015.

[9] M. U. K. Khan, M. Shafique and J. Henkel, "Fast Hierarchical Intra Angular Mode Selection for High Efficiency Video Coding," in *IEEE International Conference on Image Processing* (**ICIP**), pp. 3681-3685, 2014.

[10] M. Shafique, M. U. K. Khan and J. Henkel, "Power Efficient and Workload Balanced Tiling for Parallelized High Efficiency Video Coding," in *IEEE International Conference on Image Processing* (**ICIP**), pp. 1253-1257, 2014.

[11] M. U. K. Khan, M. Shafique and J. Henkel, "Software Architecture of High Efficiency Video Coding for Many-Core Systems with Power-Efficient Workload Balancing," in *Design, Automation and Test in Europe Conference & Exhibition* (**DATE**), pp. 1-6, 2014.

[12] M. U. K. Khan, M. Shafique and J. Henkel, "AMBER: Adaptive Energy Management for On-Chip Hybrid Video Memories," in *IEEE/ACM International Conference on Computer-Aided Design* (**ICCAD**), pp. 405-412, 2013.

[13] M. U. K. Khan, M. Shafique and J. Henkel, "An Adaptive Complexity Reduction Scheme with Fast Prediction Unit Decision for HEVC Intra Encoding," in *IEEE International Conference on Image Processing* (**ICIP**), pp. 1578-1582, 2013.

[14] M. U. K. Khan, M. Shafique, M. Grellert and J. Henkel, "Hardware-Software Collaborative Complexity Reduction Scheme for the Emerging HEVC Intra Encoder," in *Design, Automation and Test in Europe Conference & Exhibition* (**DATE**), pp. 125-128, 2013.

[15] M. U. K. Khan, M. Borrmann, L. Bauer, M. Shafique and J. Henkel, "An H.264 Quad-FullHD Low-Latency Intra Video Encoder," in *Design, Automation and Test in Europe Conference & Exhibition* (**DATE**), pp. 115-120, 2013.

[16] M. U. K. Khan, M. Shafique and J. Henkel, "A Hierarchical Control Scheme for Energy Quota Distribution in Hybrid Distributed Video Coding," in *IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis* (**CODES+ISSS**), pp. 483-492, 2012.

List of Co-Authored Publications

[1] J. Henkel, H. Bukhari, Siddharth Garg, M. U. K. Khan, H. Khdr, F. Kriebel, U. Ogras, S. Parameswaran and M. Shafique, "Dark Silicon – From Computation to Communication," in *9th International Symposium on Networks-on-Chip* (**NOCS**), pp. 1-8, 2015.

[2] M. Grellert, M. Shafique, M. U. K. Khan, L. Agostini, J. C. B. Mattos and J. Henkel, "An Adaptive Workload Management Scheme for HEVC," in *IEEE International Conference on Image Processing* (**ICIP**), pp. 1850-1854, 2013.

[3] M. Shafique, M. U. K. Khan and J. Henkel, "Content-Driven Adaptive Computation Offloading for Energy-Aware Hybrid Distributed Video Coding," in *IEEE International Symposium on Low Power Electronics and Design* (**ISLPED**), pp. 106-113, 2013.

[4] M. Umar Karim, M. U. K. Khan, Y. M. Khawaja, "An Area Reduced, Speed Optimized Implementation of Viterbi Decoder," in *International Conference on Computer Networks and Information Technology* (**ICCNIT**), pp. 93-98, 2011.

[5] M. U. K. Khan, A. Bais, Y. M. Khawaja, G.M. Hassan, R. Arshad, "A Swift and Memory Efficient Hough Transform for Systems with Limited Fast Memory," in *International Conference on Image Analysis and Recognition* (**ICIAR**), pp. 297-306, 2009.

[6] A. Bais, M. U. K. Khan, Y. M. Khawaja, R. Sablatnig, G. M. Hassan, "Memory Efficient Vision Based Line Feature Extraction for Tiny Mobile Robots**,"** in *International Conference on Image Analysis and Recognition* (**ICIAR**)*,* pp. 287-296, 2009.

[7] Y. A. Khan, A. Ullah, H. Ali, Y. M. Khawaja, M. U. K. Khan, "Differential Based Area Efficient ROM-less Quadrature Direct Digital Frequency Synthesis," in *International Conference on Emerging Technologies* (**ICET**), pp. 81-86, 2009.

[8] S. A. A. Shah, T. J. Khattak, M. Farooq, Y. M.. Khawaja, A. Bais, A. Anees, M. U. K. Khan, "Real Time Object Tracking in a Video Sequence Using a Fixed Point DSP," in *International Symposium on Visual Computing* (**ISVC**), pp. 879-888, 2008

Posters

[1] M. U. K. Khan, M. Shafique and J. Henkel, "Hardware-Software Co-Design for Next Generation Dark Silicon Multimedia Systems", ACM/SIGDA Ph.D. Forum at the *ACM/EDAC/IEEE Design Automation Conference* (**DAC**), 2015.

[2] M. U. K. Khan, M. Shafique and J. Henkel, "Hardware-Software Co-Design for Next Generation Dark Silicon Multimedia Systems", ACM/SIGDA Ph.D. Forum at the *Design, Automation and Test in Europe Conference & Exhibition* (**DATE**), 2015. **Received the DATE 2015 Ph.D. forum best poster award**.

[3] M. U. K. Khan, M. Shafique and J. Henkel, "Application-Specific Hierarchical Power Management for Multicast High Efficiency Video Coding", in *ACM/EDAC/IEEE Design Automation Conference* (**DAC**), 2014. **Received the DAC 2015 Designer Track best poster award**.

Abstract

High throughput demands under complexity- and power-efficiency has imposed numerous design challenges for the next generation multimedia systems. Multimedia (especially video) applications impose tight throughput constraints (e.g., frame resolutions beyond 1920×1080, at more than 30 FPS), which must be met by possibly resource- and battery-constrained underlying hardware.

However, technology scaling in the nano-era has led to high transistor densities. On one hand, the technology scaling provides increased resources to the application designer, to enable high throughput multimedia processing. Contrarily, these technological advancements come with their associated challenges, like high power densities (Dark Silicon paradigm). Furthermore, high power densities lead to elevated on-chip temperatures that jeopardize the reliability of the multimedia system, like NBTI-induced aging. This suggests that a next generation multimedia system which consumes low power might not be able to fulfil the throughput constraint, while a multimedia system meeting its throughput constraints might be resource- and power-wise inefficient. These contradictions open new frontiers for exploring software and hardware level co-design and co-optimizations space. Since the complexity and power consumption of the multimedia system can be reduced at both the software and hardware level, therefore, several software and hardware factors (like varying workload characteristics, Thermal Design Power or TDP constraints, application-specific architectural optimizations and available hardware resources) play an important role for designing high complexity embedded multimedia systems. The state-of-the-art works, however, do not exploit the complete hardware-software design optimization space of advanced embedded multimedia systems under Dark Silicon constraints, to fully exploit the power-, complexity- and resource-saving, and reliability improvement potential for long-term system deployment.

The aim of this Ph.D. thesis to design efficient multimedia (specifically image/video) systems that are easily portable to programmable soft-cores, application-specific hardware platforms, and domain specific hardware accelerators, while providing power-efficiency and reliability. The key design novelty is to recognizing and mutually consider the hardware constraints and software/application-specific characteristics, and synergistically and objectively tuning software and hardware parameters. Since image/video processing workload are power hungry, therefore, this Ph.D. thesis targets to encompass multiple design aspects (complexity reduction, workload balancing, power reduction, aging optimization) in an integrated manner to improve power and reliability metrics.

Moreover, this work builds software and hardware optimizations by analyzing the applications and hardware characteristics, and then leveraging the application- and content-knowledge for design and management of next generation multimedia systems from both power and reliability perspective. The design-focus of our approaches and strategies is a multi-/many-core system, with on-chip hardware co-processors and accelerators. A brief summary of the contributions by this thesis are given below.

Power-Efficient Software Layer: For the multimedia systems, the software layer determines system parameters (number of cores used by the parallel running application(s), amount of tasks offloaded to hardware accelerators and high-end servers, voltage-frequency settings of the cores, power-gating control etc.) and adapts them by using feedback from the hardware layer. The goal is to increase the throughput-per-watt metric of the multimedia system. A synopsis of the software level approaches proposed in this thesis is given below.

- *Parallelization and Workload Balancing*: To avoid computational hotspots and utilize the underlying hardware, parallelization and workload balancing approaches presented in this thesis target power reduction while meeting the throughput demands of the video applications. At runtime, a multi-objective optimization is performed which divides the workload in either uniform or in a non-uniform manner

among the cores, and tunes the application parameters. On homogeneous cores, the proposed approaches result in up to ~19% power savings compared to the state-of-the-art approach [1], while additional ~7.8% power savings are obtained with non-uniform load distribution. Up to 64% throughput-per-watt improvement is obtained compared to [2] while using heterogeneous computing nodes.

- *Resource Budgeting*: While considering the throughput demands of multiple, multithreaded applications, the resource budgeting approach presented in this thesis divides the available cores and the TDP among these applications. The resources allocated to the applications are adapted at runtime, and this improves the throughput of the system from ~1.18× to ~1.45× compared to [3], under varying Dark Silicon scenarios.
- *Computation Offloading*: At the software side, the video content- and throughput constraints-driven offloading mechanisms are developed to offload computations to a high-end server, which achieves considerable energy savings (~20%) compared to [4].

Power-Efficient Hardware Layer: The hardware layer supports video I/O, communication among (possibly heterogeneous) compute nodes, power-efficient video memory design and aging-aware optimizations. Further, this layer exposes some of its functionality to the software layer (for approaches like software-guided frequency tuning of the cores, power gating and feedback of statistics to the software). A brief summary of the architectural contributions of this thesis are given below.

- *Video I/O and Communications*: To develop high throughput applications, video I/O architectures and custom hardware for communication among computing nodes proposed by this thesis targets communication efficiency at reduced hardware cost.
- *Hardware Accelerator Sharing/Scheduling*: To offload workload from soft-cores to the shared hardware accelerator, or, to share the hardware accelerator for processing multiple tasks in a round-robin fashion, hardware accelerator sharing and scheduling approaches are presented such that the throughput of all the soft-cores is met, hardware accelerator is fully utilized and the power consumption of the system is minimized. Similarly, efficient hardware accelerators are designed which can provide high throughput and power-efficiency (by selectively clock-gating parts of the accelerator) while meeting the computational constraints of the video system. For the H.264/AVC encoding loop, the proposed approach achieves ~4.14× hardware savings compared to [5], while the proposed edge detection mechanism (for efficient mode computations) results in ~1.9× area savings compared to [6].
- *Memory Subsystem Design*: A hybrid memory architecture, consisting of sectored non-volatile memory (MRAM) based frame buffers and SRAM FIFOs, achieves high power savings at minimal latency penalty, by adaptively turning ON the normally OFF MRAM sectors. Moreover, the on-chip SRAM aging resiliency approach presented here exploits video content-properties to reduce the aging rate of 6T SRAM cells, which store the data bits. A controller is proposed which adaptively performs aging-aware, online data adaptation at different spatial and temporal granularity.

The above mentioned software and hardware approaches have resulted in several open-source contributions, which are available for download and can be found in the free software pool of our lab's (Chair for Embedded Systems, CES) webpage: http://ces.itec.kit.edu/.

In a nutshell, software and hardware properties are synergistically evaluated to determine the degree of parallelism, task offloading and resource budgeting. Moreover, the proposed approaches result in tunable, software guided frequency and gating control of the hardware using feedback from the hardware, in order to lower the power consumption of the system. Further, the proposed video system's hardware layer consists of novel accelerator design methodology, and power- and aging-efficient memory subsystem.

Zusammenfassung

Hohe Durchsatzanforderungen unter Beachtung von Komplexitäts- und Leistungseffizienz hat zu zahlreichen Design-Herausforderungen für Multimediasysteme der nächsten Generation geführt. Multimediaanwendungen(insbesondere Video) haben starke Durchsatzanforderungen (z.B. Bildgrößen jenseits von 1920×1080 bei mehr als 30 FPS), die von Hardware erfüllt werden müssen, die möglicherweise eingeschränkt ist in ihren Ressourcen und Batterie.

Allerdings hat die Technologie-Skalierung in der Nano-Ära zu hohen Transistordichten geführt. Einerseits hat die Technologie-Skalierung mehr Ressourcen für den Anwendungsdesigner bereitgestellt und damit Multimediaverarbeitung mit hohem Durchsatz ermöglicht. Andererseits führen diese technologischen Fortschritte mit ihren damit verbundenen Herausforderungen, wie z.B. hohe Leistungsdichten (Dark Silicon Paradigma). Weiterhin führen hohe Leistungsdichten zu erhöhten On-Chip-Temperaturen, die die Zuverlässigkeit des Multimediasystems gefährden, durch z.B. NBTI-induzierte Alterung. Dies legt nahe, dass ein Multimediasystem der nächsten Generation, das eine geringe Leistungsaufnahme hat, möglicherweise nicht die Durchsatzanforderungen erfüllen kann, während ein Multimediasystem, das seine Durchsatzanforderungen erfüllt, möglicherweise ressourcen- und leistungsineffzient ist. Diese Widersprüche öffnen neue Möglichkeiten für die Erkundung von Software- und Hardware-Ebene Co-Design und Co-Optimierungsräume für Multimediasysteme. Da die Komplexität und Leistungsaufnahme eines Multimediasystems sowohl auf der Software- als auch auf der Hardware-Ebene reduziert werden kann, haben mehrere Software- und Hardware-Faktoren (wie unterschiedliche Auslastungsmerkmale, Thermal Design Power oder TDP Anforderungen, anwendungsspezifische Architekturoptimierungen und verfügbare Hardwareressourcen) eine wichtige Rolle um eingebettete Multimediasysteme mit hoher Komplexität zu designen. Die State-of-the-Art-Arbeiten nutzen aber nicht alle Möglichkeiten des Design-Raums der Hardware-Software-Optimierung von fortgeschrittenen eingebetteten Multimediasystemen unter Dark Silicon-Anforderungen, um das volle Potential für Leistung-, Komplexitäts- und Ressourcenreduzierung und Zuverlässigkeitsverbesserungen für langfristige Systembereitstellung auszunutzen.

Das Ziel diese Doktorarbeit ist es effiziente Multimediasysteme (insbesondere Bild/Video) zu erstellen, die einfach auf programmierbare Soft-Cores, anwendungsspezifische Hardwareplattformen und domänen-spezifische Hardwarebeschleuniger anpassbar ist und gleichzeitig Leistungseffizienz und Verlässlichkeit zu erreichen. Die wichtigste Designneuheit ist das Erkennen und gegenseitige Beachten von Hardware-Einschränkungen und software/anwendungsspezifischen Eigenschaften und synergistisch und objektiv Software- und Hardwareparameter abzustimmen. Da Bild/Video-Bearbeitung eine große Leistungsaufnahme hat, zielt diese Doktorarbeit darauf ab mehrere Designaspekte (Komplexitätsreduktion, Lastausgleich, Leistungsreduzierung, Alterungsoptimierung) in einer integrierten Weise zu umfassen um Leistungs- und Zuverlässigkeitsmetriken zu verbessern.

Zudem erstellt diese Arbeit Software- und Hardwareoptimierungen, indem sie die Eigenschaften der Anwendungen und Hardware analysiert und dann das Anwendungs- und Inhaltswissen für das Design und die Verwaltung von Multimediasystemen der nächsten Generation im Hinblick auf Leistung und Zuverlässigkeit ausnutzt. Das Designziel unserer Ansätze und Strategien ist ein Multi-/Many-Core-System mit On-Chip Hardware-Ko-Prozessoren und Beschleunigern. Eine kurze Zusammenfassung der Beiträge dieser Arbeit sind folgend aufgelistet.

Leistungseffiziente Softwareebene: Für Multimediasysteme bestimmt die Softwareebene die Systemparameter (Anzahl der Kerne, die von den parallel laufenden Anwendung(en) genutzt werden, Menge der Tasks, die auf Hardwarebeschleuniger und High-End-Server ausgelagert wurden, Spannungs-Frequenz-

Einstellungen der Kerne, Power-Gating etc.) und passt sie an durch Rückmeldung der Hardwareebene. Das Ziel ist es die Durchsatz-pro-Watt-Metrik des Multimediasystems zu erhöhen. Eine Übersicht über die in dieser Arbeit vorgeschlagenen Ansätze zu Softwareebene ist folgend angeführt.

- *Parallelisierung Und Lastverteilung*: Um Berechnungs-Hotspots zu vermeiden und die zugrunde liegende Hardware auszunutzen, zielen die Parallelisierungs- und Lastverteilungsansätze in dieser Arbeit auf Leistungsreduzierung unter Einhaltung der Durchsatzanforderungen der Videoanwendungen. Zur Laufzeit wird eine Optimierung mehrerer Ziele durchgeführt, die die Arbeitslast entweder in einer gleichförmigen oder nicht-gleichförmigen Weise auf die Kerne aufteilt und die Anwendungsparameter einstellt. Auf homogenen Kernen führen die vorgeschlagenen Ansätze zu bis zu ~19% Leistungsreduzierung verglichen mit dem State-Of-The-Art-Ansatz [1]. Wobei zusätzliche ~7,8% Leistungsreduzierung durch nicht-gleichförmige Lastverteilung erreicht werden. Bis zu 64% Durchsatz-pro-Watt Verbesserung wurde erreicht verglichen zu [2], wenn heterogene Berechnungsknoten benutzt wurden.
- *Ressourcenplanung*: Unter Berücksichtigung der Durchsatzanforderungen von mehreren Multithread-Anwendungen teilt der Ressourcenplanungs-Ansatz in dieser Arbeit die vorhandenen Kerne und die TDP unter diese Anwendungen auf. Die Ressourcen, welche den Anwendungen zugeteilt wurden, werden zur Laufzeit angepasst und das verbessert der Durchsatz von ~1,45× zu ~1,18 verglichen mit [3] unter Berücksichtigung von verschiedenen Dark-Silicon-Szenarien.
- *Berechnungsauslagerung:* Auf der Softwareseite werden Auslagerungsmechanismen, die von Videoinhalts- und Durchsatzanforderungen getrieben werden, entwickelt um Berechnungen auf einen High-End-Server auszulagern, was erhebliche Energieeinsparungen (~20%) verglichen mit [4] ergibt.

Leistungseffiziente Hardwareebene: Die Hardwareebene unterstützt Video-I/O, die Kommunikation zwischen den (möglicherweise heterogene) Rechenknoten, leistungseffizientes Videospeicher-Design und alterungsbewusste Optimierungen. Weiter macht diese Ebene einige seiner Funktionen der Softwareebene (für Ansätze wie software-gesteuerte Frequenzabstimmung der Kerne, Power-Gating und Rückmeldung von Statistiken zur Software) sichtbar. Eine kurze Zusammenfassung der architektonische Beiträge dieser Arbeit folgen.

- *Video-I/O und Kommunikation:* Um Anwendungen mit hohem Durchsatz zu entwickeln, wurden Video-I/O-Architekturen und benutzerdefinierte Hardware für die Kommunikation zwischen Rechenknoten vorgeschlagen, um effiziente Kommunikation bei reduzierten Hardwarekosten zu erreichen.
- *Hardwarebeschleuniger Teilen/Scheduling:* Um Arbeitslast von Soft-Cores zum geteilten Hardwarebeschleuniger auszulagern oder um den Hardwarebeschleuniger zur Berechnung von verschiedenen Aufgaben der Reihe nach zu teilen, wurden Ansätze zum Teilen und Scheduling des Hardwarebeschleunigers vorgestellt, so dass der Durchsatz aller soft-Cores erreicht wird, der Hardwarebeschleuniger voll ausgelastet ist und die Leistungsaufnahme des Systems minimiert wird. In ähnlicher Weise wurden effiziente Hardwarebeschleuniger erstellt, die hohen Durchsatz und hohe Leistungseffizienz (durch selektives Clock-Gating von Teilen des Beschleunigers) haben und dabei die Berechnungsanforderungen des Video-Systems einhalten. Für die H.264/AVC Encoding-Schleife erreicht der vorgeschlagene Ansatz ~4,14× Einsparungen von Hardware verglichen mit [5], während der vorgeschlagene Kantenerkennungsmechanismus (für effiziente Modus-Berechnung) in ~1,9× Platzersparnis verglichen mit [6] führt.
- *Speicher-Subsystem Design*: Eine Hybridspeicherarchitektur, bestehend aus Bildpuffern und SRAM FIFOs, die auf sektoriertem nichtflüchtigen Speicher (MRAM) basieren, erzielt hohe Leistungseinsparungen bei minimalen Latenzeinbußen durch adaptives EIN-schalten der normalerweise AUS MRAM Sektoren. Darüber hinaus nutzt der hier vorgestellte Ansatz zur

Alterungswiderstandsfähigkeit mit On-Chip-SRAM Video-Inhaltseigenschaften aus, um die Alterungsrate von 6T SRAM-Zellen, die die Videobilder speichern, zu reduzieren. Eine Steuereinrichtung wird vorgeschlagen, welche adaptiv alterungsbewusst Online-Datenanpassung mit unterschiedlicher räumlicher und zeitlicher Granularität durchführt.

Die oben genannten Software- und Hardwareansätze haben zu mehreren Open-Source-Beiträgen geführt, die zum Download verfügbar sind und im kostenlosen Softwarekatalog unserer Laborwebseite (Chair for Embedded Systems, CES) gefunden werden können: http://ces.itec.kit.edu/.

Kurz gesagt, werden Software- und Hardwareeigenschaften synergistisch ausgewertet um den Grad der Parallelisierung, Task-Auslagerung und Ressourcenplanung zu bestimmen. Darüber hinaus führen die vorgeschlagenen Ansätze in abstimmbare software-bestimmte Frequenzregelung und Gating-Control der Hardware unter Benutzung von Feedback der Hardware, um die Leistungsaufnahme des Systems zu verringern. Außerdem besteht die Hardwareebene des vorgeschlagenen Videosystems aus einer neuartigen Designmethodik eines Beschleunigers und einem Leistungs- und Alterungs-effizienten Speichersubsystem.

Contents

List of Figures

List of Tables

List of Abbreviations

AGU	Address Generating Unit, creates addresses to write/read data to/from the memory
AMBER	Adaptive Energy Management for On-Chip Hybrid Video Memories, an approach to reduce the energy consumption of the video memory subsystem by integrating volatile and non-volatile memories
ASIC	Application Specific Integrated Circuit, high throughput low power customized circuit for performing a specific task as opposed to a core
ASIP	Application Specific Instruction-set Processor, a processor with integrated custom hardware accelerators to increase its throughput
BD-PSNR	Bjøntegaard Delta Peak Signal to Noise Ratio, measures the difference in output image quality of two different image/video processing systems
BD-RATE	Bjøntegaard Delta bit-Rate, measures the difference in the output bit-rate of two difference video encoding systems
BTI	Bias Temperature Instability, degradation of MOSFETs due to the voltages applied at the gate of MOSFETs
CABAC	Context Adaptive Binary Arithmetic Coding, an arithmetic entropy coder (bit-stream generator) used in H.264/AVC and HEVC
CAVLC	Context Adaptive Variable Length Coding, used as an entropy coder (bit-stream generator) in H.264/AVC
CI	Custom Instruction, an instruction which is executed on custom hardware accelerator rather than the soft-core
CLK	Hardware clock
CODEC	Coder and Decoder, used to present video compressors
CTU	Coding Tree Unit, the largest block of pixels used in HEVC for encoding/decoding
CU	Coding Unit, a square block of pixels used for encoding/decoding in HEVC CODEC
DCT	Discrete Cosine Transform, a basic encoding module used in numerous video encoders
DDR	Double Data Rate memory, a synchronous, dynamic random-access memory
DFS	Dynamic Frequency Scaling, where only the frequency of a processing core is scaled, while the voltage supplied to the core is kept constant
DPM	Dynamic Power Management, for runtime power optimizations of embedded systems
DTM	Dynamic Thermal Management, for managing the temperature of the system
DVC	Distributed Video Coding, whereby constrained encoder offloads its workload to the high-end decoder by temporally downsampling at frame-level
DVFS	Dynamic Voltage-Frequency Scaling, adapting the voltage-frequency of the cores to adjust the power consumption of the cores
EC	Entropy Coder, a bit-stream generator used in video compressors like H.264/AVC and HEVC
FF	Fast-Fast, one of the many possible semiconductor etching process corner
FPGA	Field Programmable Gate Array, a programmable integrated circuit
FPS	Frames Processed per Second
GOP	Group of Pictures or video frames
GOW	Group of Wyner-Ziv frames, used in Distributed Video Coding
GPU	Graphical Processing Unit, contains programmable cores with architecture supporting efficient image processing functions
HCI	Hot Carrier Injection, an unwanted byproduct of the switching activity which increases the threshold voltage of a transistor

HDVC	Hybrid Distributed Video Coding, whereby a constrained encoder shares a part of its workload to a possibly constrained decoder
HEVC	High Efficiency Video Coding, one of the newest video encoding standard and also sometimes referred to as H.265
HT	Hadamard Transform, used to process the DC transform coefficients out of the DCT module in H.264/AVC
IDCT	Inverse Discrete Cosine Transform, generates the (approximate) input of the DCT module by processing the output of the DCT module
IF	Intra Frame, frames using only spatial information for processing the constituent blocks
IHT	Inverse Hadamard Transform, use to inverse transform the quantized DC coefficients in H.264/AVC
I/O	Input and Output
IOT	Internet of Things
IQ	Inverse Quantizer, generates the (approximate) input to the quantizer module by processing the quantizer's output
ISA	Instruction-Set Architecture, the architectural details of the processor available to the programmer/compiler
ITU	International Telecommunication Union
JCT-VC	Joint Collaborative Team on Video Coding
MB	Macroblock, basic video frame block used in H.264/AVC for video encoding
ME	Motion Estimation, the best known video compression technique
MPEG	Moving Pictures Expert Group
MPSoC	Multiprocessor System on Chip
MRAM	Magnetoresistive Random Access Memory, a non-volatile memory that incorporates magnetic storage
MRT	Memory Read Transducer, transforms the SRAM data after reading it
MUX	Multiplexer
MV	Motion Vector, defines the relative motion of a video block between two video frames
MVD	Motion Vector Difference, relative difference of a video block's motion from its surrounding blocks
MWT	Memory Write Transducer, transform the data before writing it to the SRAM
NBTI	Negative Bias Temperature Instability, BTI induced degradation on the PMOS transistor while applying a negative voltage at its gate
NVM	Non-Volatile Memory, a RAM that will retain its contents for a large time after turned OFF
PG	Power gate
PSNR	Peak Signal to Noise Ratio, usually used for quantifying the quality of an image when compared to an ideal reference image
PU	Prediction Unit, the video frame block which is used as a basic entity for Intra- and Inter-encoding in HEVC CODEC
PUMA	A PU Map (PUM) created by joining the adjacent four neighboring blocks, used for efficiently encoding a CTU in HEVC
PUM	PU Map, used for efficiently encoding a CTU in HEVC
PVC	Predictive Video Coding, employing the principles of Intra- and Inter-frame predictions
Q	Quantizer, used to quantize the transformed coefficients out of the DCT and HT modules
QoS	Quality of Service, or throughput requirement
QP	Quantization Parameter, a higher value denotes a bigger quantization step, therefore, a larger QP reduces bit-rate, at the expense of reduced reconstructed/decoded video quality

RDO	Rate-Distortion Optimization, a process to maximize the quality and minimize the output bit-rate of the video encoder
RGB	Red Green Blue, a color space format to render a raw, colored image/video frame
RO	ReOrder, a module used in H.264/AVC to reorder the quantized coefficients before pushing them to the entropy coder
ROI	Region of Interest, usually used for regions within images which are of interest to the viewer
SAD	Sum of Absolute Differences
SI	Side Information, a video frame generated at DVC/HDVC decoder using temporal interpolation of adjacent video frames
SNM	Static Noise Margin, determines the impact of noise on an SRAM cell
SOM	Self-Organizing Map, which maps a high dimensional signal to a low dimensional map
TDP	Thermal Design Power, power that can be safely handled by the thermal dissipation mechanisms of the chip
VIP	Video Input Pipeline, a hardware architecture to feed the video processing system with data from the camera(s)
WF	Wyner-Ziv Frame, a reduced workload video frame sent by the DVC/HDVC encoder
YCbCr	Luminance component with blue and red chrominance components, a color space format to present raw color images/videos, by appropriately encoding the RGB format
YUV	Same as YCbCr, but predominately for analog video

List of Variables

a	Application index
$\boldsymbol{\alpha}_k$	Workload configuration (application parameter settings) of tile/task k
$\boldsymbol{\alpha}_{k,m}$	Maximum allowable workload configuration (application parameter settings) of tile/task k
a_r	Aspect ratio of a frame, given by w/h
β	A sector of a MRAM buffer, which can be independently power gated to save leakage power
b_{frm}	Video frame size in bits
b_k	Number of bytes produced after compressing a tile k using HEVC
$b_w \times b_h$	Size of a block in pixels, in block width times height
$c_{h,i}$	Number of cycles spent on processing a subtask of task i on the shared hardware accelerator
$c_{i,k}$	Number of cycles spent on processing a subtask of task i on core k
$c_{i,max}$	Maximum number of cycles spent on processing a subtask of task i on all the cores
$c_{k,\alpha}$	Actual number of cycles consumed to process a video block/subtask in tile k, using the workload configuration $\boldsymbol{\alpha}$
$\hat{c}_{k,\boldsymbol{\alpha}}$	Estimated number of cycles used to process a video block/subtask in tile/task k, using the workload configuration $\boldsymbol{\alpha}$
$c_{s,k}$	Number of cycles of a subtask of task k, spent on a soft-core
Δ	Duty-cycle of a SRAM cell, ratio of the time the cell stores "1", to the time it stores "0"
d	Depth of testing different PU sizes
d_s	Row-downsampling rate, for computing the SAD value in H.264/AVC custom hardware architecture
E	Estimation-error covariance matrix used in the RLS filter
e_{batt}	Energy that can be supplied by the battery
$e_{dyn,w,m}$	Dynamic energy in writing a bit to the MRAM buffer
$e_{dyn,w,s}$	Dynamic energy in writing a bit to the SRAM buffer
$e_{IF,avg}$	Average energy consumption for processing an Intra-Frame
$e_{WF,avg}$	Average energy consumption for processing a Wyner-Ziv Frame
$e_{t,GOW,(enc,dec)}$	Target energy to be distributed to a complete GOW, both at the encoder and decoder side. Used in computation offloading
η	A subtask, associated with a block of pixels
$\boldsymbol{\eta}$	Task set, a set of subtasks which can be thought of as a video slice or tile
f_h	Frequency of the hardware accelerator
f_k	Frequency of the core k
$f_{k,m}$	Maximum allowable frequency of the core k
$\boldsymbol{f_m}$	A vector containing the maximum allowable frequency of all cores
f_{max}	Maximum supportable frequency of the core/node
f_{min}	Minimum supportable frequency of the core/node
f_p	Video frames processed per second
f_R	Data adaptation rate for SRAM anti-aging
$HT^n{}_{in}$	The n^{th} input of the HT lookahead module used in H.264/AVC Intra encoding
$k_{a,tot}$	Total number of cores allocated to application a
k_i	Number of cores allocated to task i
m	Ordered list of most probable Intra prediction mode in H.264/AVC Intra encoding
μ	Average value
$\boldsymbol{mv}$	2D Motion Vector
mvd	Motion Vector Difference

n_{ang}	Number of Intra angular modes available for prediction generation, used in H.264/AVC and HEVC video CODECs
n_{frm}	Number of video blocks within a frame
$n_{sec,h,i}$	Number of subtasks of task i offloaded per second to the shared hardware accelerator
$n_{sec,i}$	Total number of subtasks of task i that must be computed in one second
$n_{sec,s,i}$	Number of subtasks of task i processed per second on the soft-core
n_i	Number of subtasks within a task i
$n_{i,k}$	Number of subtasks assigned to core k
$n_{o,i}$	Difference (offset) of time consumed to process the task i and maximum allocated time
$n_{o,i,j}$	Difference (offset) of time consumed to process the subtask j (a constituent of task i) and maximum allocated time
n_v	Number of views/videos processed concurrently by the video processing platform
n_r	Number of reference frames used for motion estimation
n_{sec}	Number of blocks processed per second
o_A	Offset in the starting address of the SRAM memory, to which a video frame is written
p_{dyn}	Dynamic (or switching) power
ϕ	Efficiency index of a node, used for distributing the amount of jobs
π	A metric to represent the aging of SRAM memory based upon the actual SNM degradation experienced by the SRAM cells
p_{leak}	Leakage power
$p_{leak,d}$	Leakage power of a DRAM buffer
$p_{leak,m}$	Leakage power of a MRAM buffer
$p_{leak,s}$	Leakage power of a SRAM buffer
p_k	Power consumed by the core/node k
$p_{k,\mu}$	Average power consumed by the core/node k for all its frequencies
$p_{k,\mu,max}$	Maximum of the average powers of all the cores/nodes
ψ	Used as a temporary variable, explanation in the text
p_{sta}	Static (or leakage) power
p_{tot}	Total power consumed by the system
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^+$	Set of positive real numbers
rd	Running difference along the edges of a video block, used for early mode estimation for H.264/AVC Intra video encoder
r_f	Total number of times a pixel in the reference frame is read and then stored in the search window
r_{tot}	Total number of available compute nodes/cores
s	Samples of the original video frame
s'	Samples of the prediction of s, used for comparison with the original samples
s_r	Skip radius, used for early estimation of Intra angular modes in HEVC
$s_w \times s_h$	Search window size in pixels, in width times height
τ	Threshold value
$\tau_{b,k}$	Threshold for number of bytes produced by processing a video tile associated with core k
$\tau_{b,k,max}$	Maximum value for number of bytes used as a threshold against the number of bytes produced by processing a video tile associated with core k
τ_1, τ_2, τ_3	A set of thresholds associated with the data adaptation rate of the SRAM memory
$t_{a,max}$	Processing deadline of the application a
t_d	Total duration of processing

t_{frm}	Time to process a video frame
θ	Number of Intra Angular modes used for prediction generation in H.264/AVC and HEVC encoders
$t_{h,i}$	Time spent by the task i on the shared hardware
$t_{h,t}$	Total time spent on the shared hardware accelerator by all the compute nodes trying to access the accelerator
t_i	Time taken by a task i
$t_{i,j}$	Time taken by a subtask j of task i
$t_{i,max}$	Maximum time to process a task i
$t_{s,i}$	Time spent by the task i on a soft-core
t_t	Time of an epoch
$t_{w,m,CTU}$	Write latency of a CTU in MRAM buffer
$u_w \times u_h$	Video tile structure, with u_w tiles in width and u_h tiles in height
$w \times h$	Resolution of the video frame in pixels, in width times height
$w_m \times h_m$	Dimensions of the master video tile, used for non-uniform tiling based workload balancing approach
$\boldsymbol{\omega_k}$	A vector containing constants of the cycle estimation model of core k
X	A block of video frame in H.264/AVC and HEVC video encoders
X'	Prediction of the block of video frame X, generated by H.264/AVC and HEVC video encoders
X_i	A line of the video block X
$X^{(r)}$	Residue of a video block, used H.264/AVC and HEVC video encoders
$X^{(r)}{}_i$	Line i of the residue video block, used H.264/AVC and HEVC video encoders
$X^{(r)n}$	n^{th} 4×4 block within the residue video block, used for HT lookahed scheme in H.264/AVC custom hardware
$\boldsymbol{x}$	A vector holding the application parameters, which collectively determine the complexity, power and quality of the application
$x_{o,i}$	Accumulative offset (or misprediction) of each subtask, for all the task i in an epoch.
z	Number of tasks/video frames within two reference tasks/frames. Used for allocating energy and frequency to each task/video frame within an epoch.

Chapter 1 Introduction

Owing to the high-density fabrication technologies which allow assembling billions of transistors per processing chip, multimedia systems have universally penetrated into communication, security, education, entertainment, navigation and robotics domains. The advancement in the fabrication technology has also driven the processing capabilities and user expectations of the next generation multimedia systems. On the contrary, the evolution of next generation multimedia systems (with increasing throughput and connectivity requirements, and adaptability to application, battery etc.) requires high processing capabilities and efficient utilization of the resources, on a resource and power constraint hardware platform. Multimedia applications like latest video encoders [7] now target high quality video content compression beyond Full-HD (like 4K Ultra High Definition, 3840×2160 pixels) at high frame-rates (> 120 frames per second). Long-term deployment of such multimedia applications on small, battery driven autonomous systems is challenging, due to the high computational and power requirements which must be met to fulfill the throughput constraints. Coupled with the high throughput demands, a multimedia system must be capable of reacting to changes in the workload of the application. Further, modern nano-era fabrication technologies have their own associated challenges (like power-wall [8] and reliability [9]) which must be accounted for to forge power-efficient multimedia systems. This suggests that new design methodologies for next generation multimedia systems are needed, to address the above mentioned challenges on modern systems. This Ph.D. thesis presents some of these methodologies, both at the software and hardware layers of the multimedia system.

1.1 Next Generation Multimedia Systems

Multimedia systems holistically store, process and output/display numerous forms of content or data (like text, audio, image/video and others), for performing tasks like communication, entertainment, security, computing etc. These systems have penetrated as computers, televisions and most-recently as hand-held mobile devices. Basically, it is a union of computers, communication and signal processing domains. Typically, multimedia systems are required to process the content within a deadline, and satisfy real-time Quality of Service (QoS) constraints. This might become a considerable challenge for small, mobile, battery-driven systems (like autonomous robots) as they encounter strict QoS requirements under resource and power constraints. Interestingly, 98% of all the computing devices are embedded devices [10]. Thus, multimedia system designers try to optimize different attributes of the system, which will lead to maximize a quality metric, like power-efficiency, to maximize performance-per-unit of power that goes into the system (generally referred to as throughput-per-watt). Further, the time complexity reduction (i.e., performance optimizations) of such system will also indirectly lead to power-efficiency. This is because now, a lower clock frequency of the underlying hardware can meet the QoS requirement, and thus, lesser power is required to meet the computational demands.

Since the most compute intensive part of multimedia systems is usually image- and video-processing (taking more than 70% of the total complexity [11]), therefore, image- or video-processing is commonly targeted for optimizing the design of multimedia systems. As shown in Figure 1-1 (a), the “time spent in front of the screen” for different viewing devices is mostly consumed by battery-driven devices (e.g. smartphones), hence, power-efficiency is one of the main goals for implementing video systems. Further, from Figure 1-1 (b), more than 50% of the data on the internet and over the wireless channels is compressed video, suggesting that the majority of content processed on wireless capable devices is video. It is also predicted that by 2019, about 80% of the Internet data will be video. Moreover, many real-world critical applications embed video processing as the core algorithm. Examples are passive tracking, radar imaging, medical imaging, localization and navigation [12, 13]. Developers are even introducing real-time communication functionalities within web browsers (see WebRTC

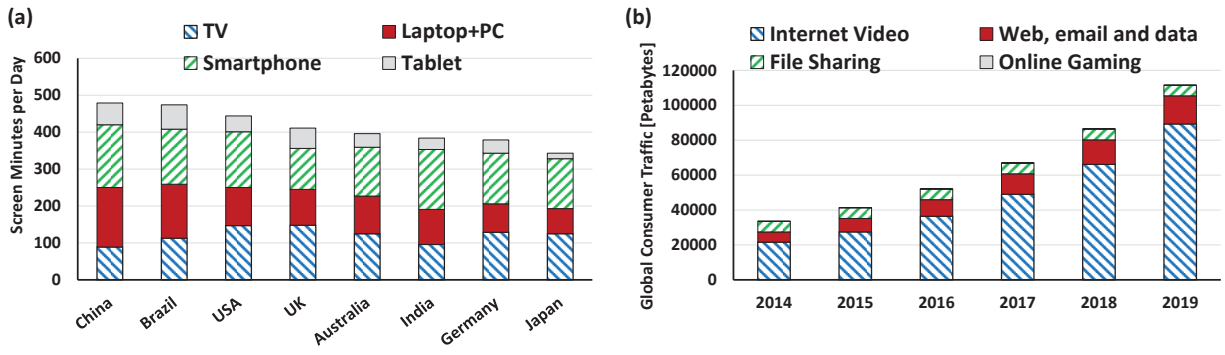


Figure 1-1: (a) Distribution of screen minutes spent in front of display mediums (via Millward Brown AdReaction 2014), (b) Global consumer internet traffic forecast (from Cisco Visual Network Index).

[14]). Furthermore, new video processing algorithms were introduced in the near past, which not only require high computational efforts by demanding a higher QoS to maximize user satisfaction, but also incur high power/energy penalty. Hence, it becomes essential to devise efficient video algorithms and system architectures, to maximize the performance/throughput-per-watt metric when video processing is performed on constraint devices.

The commonly used QoS metric in video processing is to meet the specified throughput, expressed as Frames Processed per Second (FPS) or frame rate. A video system must be able to meet the throughput demands while providing considerable output video quality (more on this in Chapter 2). This puts pressure on the underlying hardware to perform with maximum efficiency, and therefore, multiple architectural novelties and enhancements are employed by industry and research community, to maximize the throughput-per-watt for divergent set of video applications.

1.1.1 Multimedia Processing Architectures

Multimedia systems are now available in almost every compute-domain, ranging from high-end computational servers to small devices like smartphones. Processing devices have evolved over the past few decades and offer more speed, with reduced area and power consumption. The evolution of x86 processor [15] is shown in Figure 1-2. In 1978, Intel released the 8086 processor consisting of 29K transistors, capable of running at a 5MHz. However, the new Core-i7 processors with 4 cores are capable of running at 3.8GHz in the turbo boost mode. This high operating frequency of processing systems offers an opportunity to sustain high throughput requirements of multimedia applications.

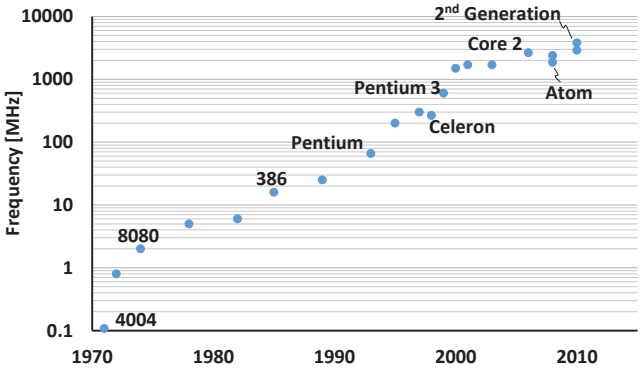


Figure 1-2: Evolution of Intel processors

Similarly, the number of compute cores embedded in the processing system are constantly increasing with every new processor release. For example, Intel has released Polaris (80-core) and Single-Chip Cloud Computer (SCC, 48 cores) [16] research chips for high-end computing servers. AMD's Opteron chips have up to 24 cores. Tilera Gx-Tile architecture has 100 cores, and Oracle's Sparc T3 houses 16 cores and can process 128 threads for web services and database applications. IBM has launched Power and System z architectures with multiple cores. Specifically for video applications, different

architectures like Larabee and Xeon Phi were also launched. Furthermore, it is now common for the new smartphones to have 4 cores and these cores are fully exploited by applications. For example, Chrome web browser (commonly used for viewing multimedia streams) tries to utilizes all available cores on an Android based smartphone. The YouTube Android application uses 4 cores on average on an 8-core processor. Moreover, Android games normally produce tens of threads and utilize most of the cores available on the system [17]. In summary, the compute power of processors is increasing with every new generation, to support the high throughput requirement of multimedia applications.

Usually, the compute power provided by software-programmable cores is not enough to meet the FPS requirement, or, to sustain the intended throughput-per-watt metric. Moreover, their power consumption is very high and thus, makes them unsuitable for implementation in small, constrained environments (like video enabled wireless sensor nodes). Therefore, video systems are also implemented using custom hardware implementations. Prominent examples are Graphical Processing Units (GPUs) [18, 19], Field Programmable Gate Arrays (FPGAs) [20, 21] and Application Specific Integrated Circuits (ASICs) [22, 23]. Such systems not only increase the performance of the video application by many folds, they also reduce the power considerably, thus increasing the throughput-per-watt ratio. Unfortunately, current Compute Aided Design (CAD) tools for custom video system hardware design and implementations do not offer low time-to-market. In addition, such designs are inflexible and require major debugging effort, compared to the software version of the video processing systems. To exploit the advantages offered by both software and hardware based designs, custom hardware accelerators are designed for high complexity functions of the software and are used in conjunction with the software [24, 25]. These accelerators can be implemented via FPGAs, and they process data in conjunction with software programmable cores. This increases the throughput of the video application while still maintaining the flexibility offered by the software solutions. Current industrial trends point towards a joint future of software and hardware implementation to offer both flexibility and performance, like Intel's acquisition of Altera and the release of a chip (Intel Atom E600C) containing Intel's x86 core with Altera's FPGA. An additional step in the same direction is to use reconfigurable fabric for implementing different types of hardware accelerators of the applications, in a time multiplexed manner [26, 27]. However, the complexity of the design increases accordingly.

1.2 Video Processing Fundamentals

In this section, we briefly discuss the video processing basics. Further analysis on video processing can be found in Section 2.1. A video consists of concatenated, temporally captured images or frames of the scene under consideration. A video frame of width w and height h consists of $w \times h$ pixels (or addresses), and each pixels stores a sample (the data used for displaying) of the frame. Usually for display, a video frame sample can be represented in 8-bits or up to 24-bits. Video algorithms perform either pixel-wise or block-based processing on samples of the frames. Pixel-wise processing considers a single pixel at a time and may process this pixel using information from the neighboring pixels (exploiting spatial correlation), or, using the same pixel in the previous frames called the collocated pixel (exploiting temporal correlation). Block-based processing considers a group of pixels at a single time and employs the same principles of spatial and temporal correlation.

Video processing can either be performed:

1. Online (or real-time), by accessing video samples directly from video cameras, or after decoding and decrypting the video packets received via wired or wireless links. These applications are also termed as streaming applications. In this situation, both the video processing algorithm and the system's architecture should be capable of handling the throughput requirement (i.e., FPS) in real-time. Hence, such systems may require high compute power.

2. Offline, by reading a video file from the disk. Here, the throughput requirement may or may not be imposed. However, a highly efficient system both in computation- and power-domains is still desirable as it will not only increase the user satisfaction, but will also lower the costs. For example, as of statistics from December 2014, YouTube receives 300 hours of videos per minute and a faster upload and process time at YouTube servers is desirable. However, about 4 billion videos are viewed per day via YouTube [28], and the end-users must be provided the data with high enough throughput to increase their viewing experience.

In both cases, the video frames are usually first stored in the external memory (DRAM) and parts of the frame are brought to the on-chip memory (generally implemented as caches or scratchpads using SRAM technology) for processing. This is because external memory can be large (GBits) while on-chip memories are usually small (few MBits). However, note that the external and on-chip memories do not need to be DRAM/SRAM, as they can also be replaced with Non-Volatile Memories (NVM) [29, 30]. The data transfer from the external to the on-chip memory depends upon the bandwidth supported by the external memory device. The on-chip computing devices (e.g., a multi/many-core system, ASIC or FPGA) reads the data from the on-chip memory, processes it and transmits it back to the external memory.

1.2.1 Video Compression

One of the major video application is video compression (encoding, or just, coding) and decompression (decoding). As shown in Figure 1-1 (b), more than 50% of data over the internet and wireless channels is compressed video. This suggests that a video encoder (at the transmission side) and video decoder (at the receiver side) are involved in video communication. For real-time, mobile implementation of video encoder/decoder (jointly called CODEC) systems, different aspects of the underlying system must be considered. For example, the throughput requirements should be reasonable, the compute power required to sustain the throughput must be available and the algorithms and architecture of the video compression system must not drain the battery at a high rate.

Historically, when the processing power of compute nodes was low (see Figure 1-2), the throughput demands were also comparatively lower. In Figure 1-3, the developmental history of video encoders is shown. The most popular video encoding standards were released by ISO-Moving Pictures Expert Group (ISO-MPEG) and International Telecommunication Unit (ITU), and their joint effort via Joint Collaborative Team on Video Coding (JCT-VC) [31]. As seen from the figure, numerous video encoding standards were developed, considering the throughput demands (i.e., frame resolution and FPS) of that era. However, as the resolution of display devices increased and the power consumed by these devices decreased, the video resolution and FPS requirements have been steadily increasing. That is, from QCIF video frames (176×144 pixels) at 30 FPS, now we are witnessing demands of 4K video frames (3840×2160) at more than 60 FPS. This is also supported by the increased processing capabilities offered by the new fabrication technologies. Therefore, every new generation of video CODECs incorporate new tools and new video processing modules, to enable better compression, in order to consume approximately the same channel bandwidth (or bit-rate) for increased throughput requirements. This has steadily led to increase in the computational complexity and power consumption of these video CODECs.

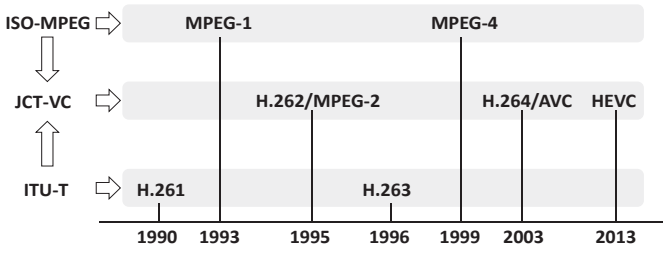


Figure 1-3: Video CODEC design history

In the past decade, the current industry standard H.264/AVC [32], and next generation CODEC tipped to overtake H.264/AVC, High Efficiency Video Coding (HEVC, also sometimes called H.265) [7] have emerged as the most prominent video CODECs. These CODECs comprehensively outperform other parallel video coding efforts (like Google's VP8 and VP9, Daala) [33]. Though H.264/AVC is almost universally supported by device vendors, both in software and hardware, HEVC is steadily finding its market. Both these CODECs use block-based frame processing and exploit sample redundancies at both spatial and temporal granularity (details in Section 2.2) for compression purpose. These video CODECs employ Predictive Video Coding (PVC) algorithms at the encoder side and the decoder is a slave of encoder, i.e., the decoder follows every command of the encoder. For minimal utilization of bandwidth, the encoder searches for the best possible way to compress the video streams. This means the encoder is a high complexity device while decoder has a relatively low complexity. This makes sense because the number of videos downloaded/played is more than videos uploaded and small decoders (like tablets and smartphones) can take advantage of a lower complexity decoder.

A new paradigm for video coding has emerged which utilizes the principles of distributed source coding and is termed as Distributed Video Coding (DVC) [34]. Here, the decoder exploits the correlation between encoded samples to reproduce missing (unprocessed and not transmitted) encoding samples. Such schemes are helpful in case of constraint video encoder, where the encoding pressure can be relaxed to save power at the encoder. To combine the advantage of PVC and DVC, Hybrid-DVC (HDVC) is also proposed [35]. More information about these paradigms will follow in Section 2.2.

1.3 Video Systems Design Complexity

Advancements in fabrication technologies are steadily increasing the number of transistors per chip. With reducing area of the cores as shown in Figure 1-4 [36], tens of cores and multitude of computational resources on chip are now available for the application designer to implement high complexity system. Each new fabrication technology is expected to reduce the minimum feature size by ~0.7×, which roughly translates to a transistor density increase of ~2× [37]. Further, the multimedia systems have advanced due to camera and ADC developments.

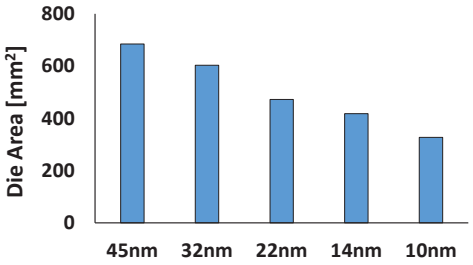


Figure 1-4: Area of a die [36]

Next generation video encoders are enhanced to meet the increasing throughput requirements with roughly the same video quality. Further, the increasing trend of 3D videos acquisition and display has also increased the complexity of video processing systems, as in this case, more than one camera is used to capture a scene. The increased resolution and FPS supportable by video cameras and displays put enormous pressure on the real-time video encoding system and introduce design complexity challenges to the designer. Examples are super-slow motion videos, captured at high resolution. Some of these challenges posed by the next generation multimedia systems are given in Figure 1-6. Similarly, new video processing algorithms are much complex than their predecessors. For example, HEVC is ~2.65× more complex than H.264/AVC in order to provide ~35% more compression compared to H.264/AVC [38]. For HEVC, the memory access requirements are more than 2× compared to H.264/AVC [39]. The power demands of these applications have also increased owing to fast processing requirements and increased memory accesses. Additionally, the fabrication technology scaling introduces numerous additional challenges for the system designer, some of which are detailed in the next sections.

1.3.1 The Dark Silicon Problem

For new fabrication technologies, the dissipated power is not decreasing at the same rate as area (as seen in Figure 1-4) due to the failure of Dennard's scaling [40]. Dennard's scaling suggested that the power density would approximately remain constant even if the size of the transistors is reduced, i.e., power and area will reduce at the same rate to keep the ratio of power to area almost constant. However, designers are facing a growing number of challenges to keep up with the voltage scaling to have constant electric field for constant power density and reliability [37]. Moreover, Dennard's scaling assumes increased channel doping to enable shorter channels (for appropriate threshold voltages). On the contrary, this causes an increase in the leakage current. Since Dennard's predictions are no longer valid, therefore, the power density (i.e., power per unit area) is increasing with new fabrication technologies, as shown in Figure 1-5 [36]. Hence, the temperature of the chip might be elevated to a level which may not be sustainable by the Silicon and the designed cooling solution. Thus, advanced cooling mechanisms maybe required to bring down the temperature within safe limits, or, the transistors of the chip must be turned ON and OFF in an adaptive manner. In other words, not all of the chip's transistors should be kept ON at maximum capacity, at all the times [8]. This underutilization of the chip's real-estate is generally referred to as Dark Silicon, which refers to the inefficient consumption of on-chip resources. Therefore, high throughput applications, like video processing, may not be able to meet their real-time throughput constraints if implications of Dark Silicon are not considered. Even a solution designed for a particular system may not be directly applicable to other systems, or, for the same system if there are parallel running workloads, or, for the applications exhibiting high workload variations.

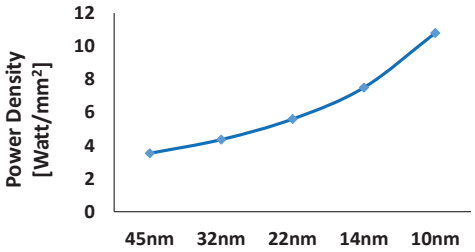


Figure 1-5: Power density for different fabrication technologies [36]

Moreover, every chip is also assigned a Thermal Design Power (TDP) [41], which is the maximum power that can be pumped into the chip. More TDP means the cores can run faster and process their assigned workloads quicker. TDP is limited by the physics of the device and the cooling mechanism. In addition, every core is also assigned a TDP. This suggest that video applications cannot arbitrarily run faster and must also consider the parallel running applications, as the TDP budget is divided among all parallel applications.

1.3.2 SRAM Aging

Some multimedia systems are employed for long durations, e.g., video processing servers like YouTube using Google File System (GFS) [42], security cameras to provide live streams, space missions. Moreover, video processing algorithms are generally memory intensive and require large on-chip buffers in case a high performance is desired. However, such systems have a service-lifetime after which they no longer functional reliably. One of the reliability concerns for modern systems is the SRAM aging, whereby the sensitivity of the SRAM cells to the noise increases, allowing for increased rate of spurious bit-flips. This is also a direct implication of elevated temperatures of the chip, where in addition to the Dark Silicon dilemma, the elevated temperature also reduces the reliability of the system. Phenomenon such as Bias Temperature Instability (BTI) become more prominent in the new fabrication technologies.

Negative Bias Temperature Instability (NBTI) is one of the foremost reliability concern for SRAM memories [43]. NBTI-induced stress on the constituent SRAM transistors results in a higher aging rate and increased read errors. It reduces the Static Noise Margin (SNM) of SRAM cells and thus, the impact of noise increases, which means that the maximum supportable frequency of the system decreases. For video processing systems, mechanisms must be devised to reduce either the impact of NBTI or eliminate conditions that increase NBTI.

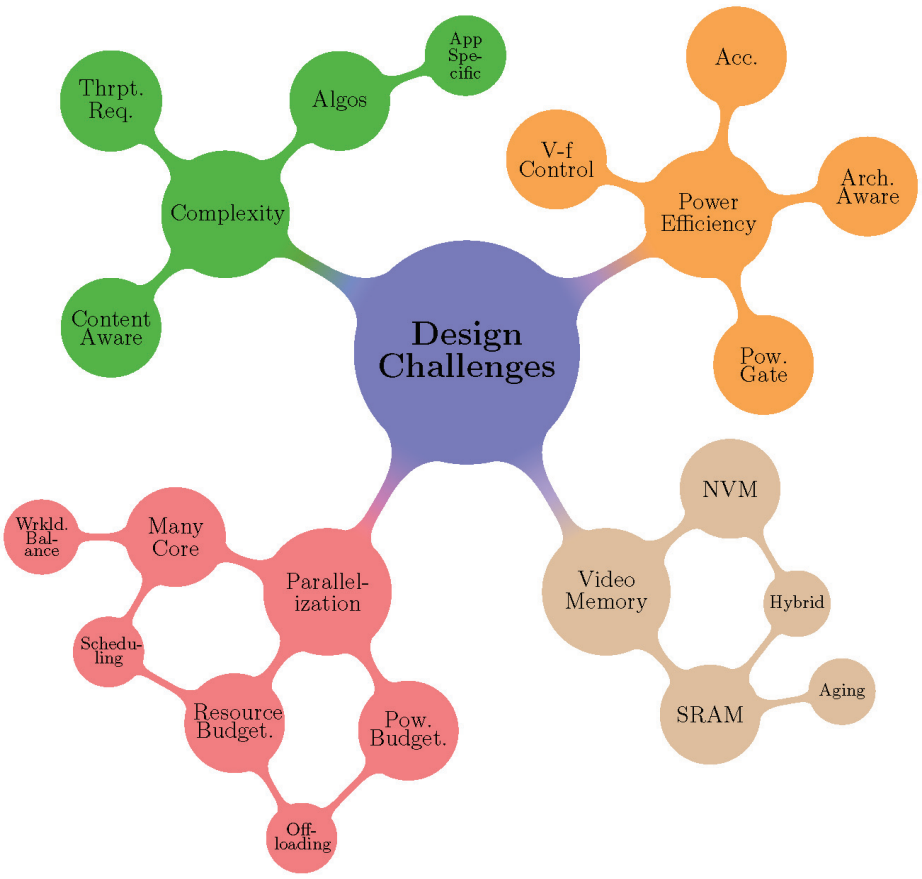


Figure 1-6: Video system design challenges addressed in this work. Here, Acc: Accelerator, Arch: Architecture, Pow: Power, Req: Requirement, Thrpt: Throughput, Wrkld: Workload

More information about NBTI and its characteristics will be detailed in Section 2.3.2.

1.4 Design Challenges for Video Systems

The video system design challenges that are addressed in this Ph.D. thesis are outlined in Figure 1-6. These challenges arise due to the throughput requirements melding with the problems initiated by new fabrication technologies. The major design challenge is to maximize the throughput-per-watt metric. For this purpose, both hardware (architectural) and software (algorithmic) layers of a video system needs to be designed while considering the above mention paradigms, like Dark Silicon and NBTI-induced aging. In short, the design of architectural and algorithmic layers of video systems must result in a video system consuming low energy/power, have high throughput and can be reliably operated. These challenges can be classified under the power-efficiency umbrella which can be summarized as:

- Either the power consumption of the system is minimized for meeting a set throughput constraint, or,
- The throughput of the system is maximized under a fixed power budget.

One of the major challenges to address for such video processing systems is to reduce their complexity such that real-time, online operations of these video systems can be carried out. This necessitates addressing the throughput requirements and designing processing algorithms with minimal overhead. To maximally achieve the complexity reduction potential, these algorithms need to leverage the application knowledge. Further, if the content properties are analyzed (either online or offline), additional complexity reduction potential can be can be achieved. The challenge remains how to determine these properties and accordingly tune the system

parameters to increase system's performance. Some of these design challenges are discussed below.

Parallelization and Power Budgeting: Almost every complex video processing algorithm allows for parallel processing, to exploit the increasing number of cores on the chip. The challenge is to design appropriate parallelization schemes, to utilize the available number of cores and their characteristics (e.g., maximum frequency) such that the throughput-per-watt metric is maximized. For systems employing many-cores, it is possible that the workload of the cores differ and (abruptly) vary at runtime. Thus, an additional challenge is to balance the workload among the cores, in order to maximally utilize the available hardware and thus, increase the throughput. Further, the processing jobs must be appropriately packed and scheduled on these cores. In addition, for multiple, multithreaded video applications (like multi-casting scenarios), the on-chip resources (i.e., the number of cores and the TDP budget) must be efficiently distributed among the competing applications. Otherwise, some applications might lose performance and miss their deadlines. In case the throughput demands cannot be met due to constrained video systems, the video processing workload can be offloaded to a high-end server. To efficiently do so, system parameters (like maximum throughput of the video processing system, energy spent in communication etc.) must be considered.

Achieving Power- and Computational-Efficiency: The goal of a video system is to be power-efficient. Thus, for a many-core system, the voltage-frequency settings of the cores must not be free variables, rather, they must be dependent upon the throughput. One way to achieve power efficiency is to utilize the dark/gray physical areas of the chip by implementing customized logic (like high throughput accelerators) to offload workload from the cores. Such a design needs careful calibration, and the accelerator must be scheduled among competing applications depending upon the processing requirements of the cores and the throughput requirements of the threads/applications running on these cores. A failure to do so may result in unfair accelerator distribution and deadline misses. In addition, parts of the systems can be adaptively power-gated to save power. However, the power-gating should not hurt the performance of the system because components of the system require a wake-up (or warm-up) time before they can be used again after gating.

Memory Subsystem Design: An intelligent memory subsystem uses the external and on-chip memory synergistically such that the accesses to the external memory are reduced and high power-efficiency is achieved. This subsystem can also employ hybrid memories (combining volatile memories like SRAM with NVMs) and efficiently exploit advantages of different memory technologies, in order to reduce the power consumption. Further, the on-chip SRAM systems will age and reduce the SNM, and the challenge is to design the SRAM subsystem in a manner to lessen the aging rate.

The above mentioned challenges can be classified into different video system layers. The data layer is responsible for handling the input to the video system. Software layer corresponds to the code maintained by the system designer, OS/Kernel, and it handles the complexity of the video system software, parallelization and its data structures. The hardware layer corresponds to the system architecture like memory, Network on Chips (NoCs) and processors. Amalgamation of software and hardware layer relates to hardware accelerators, GPUs etc., whereby the software and hardware share information and require a co-design.

Summarizing, the above mentioned challenges can be mapped to the software and hardware layers of the multimedia systems as given below.

1.4.1 Software layer Challenges

The system designer must address the following challenges at the software layer:

- Selecting the appropriate number of parallel computing nodes/cores to use on possibly heterogeneous systems, for best output video quality and maximum throughput-per-watt.

- Distributing the TDP among applications (power budgeting) for fulfilling the throughput requirements of multiple, multithreaded video applications.
- Runtime workload balancing of parallelized video applications, for maximum utilization of available resources.
- Video processing algorithmic enhancements by exploiting application-specific properties, to reduce the computational complexity of the systems.
- Adaptive computation offloading mechanisms to offload workload from the constraint devices.

1.4.2 Hardware Layer Challenges

Following challenges need to be addressed at the hardware layer:

- Appropriate voltage-frequency settings of the cores to maximize the throughput-per-watt metric at insignificant output video quality degradation.
- Power efficient application/thread scheduling, on the competing cores and the shared devices like hardware accelerators.
- Efficient hardware accelerator integration with the multi- or many-core system.
- Power efficient designs of accelerators and the memory subsystem.
- SRAM memory design with aging rate reduction, for long-term video system deployment.

1.5 Limitations of State-of-the-Art

The state-of-the-art techniques for increasing the computational/power efficiency of general and video processing applications will be discussed in detail in Chapter 2. Here, a brief synopsis of their limitations is presented to the reader for quick reference.

Most of the state-of-the-art video system design approaches do not exploit the co-design space of software and hardware. Both software and hardware layers are developed orthogonal to each other, and are optimized irrespective of the other's state. These schemes do have an advantage of high applicability and portability, because the designer is only concerned with optimization of a single layer. Moreover, such approaches may reduce the efficiency potential of the video system, and fully exploiting this co-design space might lead to much better power efficiency.

At the software layer, mostly, complexity reduction approaches are proposed which do not consider the underlying hardware properties. At the hardware layer, different architectural solutions are given. Special purpose hardware like VLIW processors, Digital Signal Processors (DSPs) and Graphics Processing Units (GPUs) are proposed, which can be programmed by the designer. Usually, these processors are power hungry. Customized hardware (ASICs and FPGAs) do present an opportunity for high performance under low power consumption. Typically, these designs ignore throughput adaptation mechanisms or software level control, and do not consider the application knowledge. The joint scheduling of cores and hardware accelerators ignores the throughput demands of the applications under consideration. Further, an important aspect of a video system's power-efficiency is runtime workload balancing among possibly heterogeneous compute nodes.

Further, state-of-the-art video systems generally do not consider the implications of Dark Silicon. Little consideration is given to application-specific optimizations, which reduces the power saving potential of the applications. Ignoring the abrupt workload variations of the application(s), or, threads of the application(s), reduces the power-efficiency potential even further. Generally, only TDP budgeting or assignment of cores is presented, and workload balancing do not consider the hardware platform properties and power awareness of

the system.

Moreover, the aging characteristics of video systems (specifically its memory subsystem) are usually not considered. General SRAM memory aging reduction involves high power overhead, multiple memory read/writes, inefficient aging reduction techniques and custom SRAM memory designs. In addition, these techniques are generally not applicable to on-chip scratchpad memories (memories in control of the programmer).

Summarizing, the complete video system design (both its software and hardware layers) is usually not entertained. The state-of-the-art limits to either one of these layers and does not address the challenges imposed by new fabrication technologies. If the software and hardware layers are tuned synergistically, a larger potential of power-efficiency is attainable, because the designer can employ the application-specific optimization.

1.6 Thesis Contributions

This Ph.D. thesis contributes to the design of a video system, by jointly accessing the co-design space of software and hardware layers. The major contributions are outlined in Table 1-1. The key-challenge addressed in this work is to synergistically optimize both the software and hardware layers to maximize the throughput-per-watt matric of the video system. In this way, the software considers the maximum throughput supported by the hardware layer, and the hardware layer feedbacks information and tunes the software layer parameters. Both layers consider the implication of Dark Silicon and SRAM aging impact. Effectively, the power-management and efficiency is performed at a higher abstraction level (i.e., at software layer) and power configuration knobs are provided by the hardware layer. These knobs are tuned by the software layer selectively and objectively, both at design-time and runtime.

In this thesis, many-core systems with hardware accelerators, and application specific custom platforms are targeted for video applications. Different software and hardware layer optimization approaches are presented, by developing algorithms and application-specific hardware accelerators, and connection between them is established. For power-efficiency, the determination of appropriate number of parallel computing threads (i.e., parallelization) and workload balancing of a video application is proposed, which considers the hardware platform properties and throughput demands of the application. This technique is extended to distribute the resources on the many-core system (i.e., the compute cores and the hardware accelerators) to the parallel threads, and also the TDP budget among the competing, multiple multithreaded video applications. In these scenarios, the voltage-frequency levels of the cores are adjusted. The workload of parallel running applications/threads on a many-core system is also offloaded adaptively to either a high-end server or a shared hardware accelerator. The offloading amount and scheduling is used for both power and throughput optimization.

At the hardware layer, efficient hardware architectures for video data I/O to the system, and communication among compute nodes is discussed. Numerous hardware accelerators are developed, especially for video coding applications (HEVC and H.264/AVC). These accelerators target high throughput at little power consumption. Moreover, a hybrid architecture for video scratchpad memories is presented, which uses NVM in conjunction with the SRAM, to reduce memory subsystem power with little throughput penalty. A power-efficient approach is also presented to reduce the aging rate of SRAM memories for long-term system deployment.

In the following, we discuss these contributions in more detail with reference to Table 1-1.

Table 1-1: A brief summary of the contributions by this thesis

Software Layer	*Runtime Video Application Parallelization (Section 4.1)* • Workload Balancing via DVFS (Section 4.1.1) • Uniform and Non-Uniform Workload Distribution (Sections 4.2.1-4.2.2.1) • Self-Regulated Frequency Modeling (Section 4.2.5) *Power-Efficient Application Configuration Selection (Section 4.3)* • Configuration Tuning and Mapping for HEVC (Sections 4.3.1-4.3.3) *Resource (Number of Cores and TDP) Budgeting* • Number of Cores and Frequency Allocation for Heterogeneous Nodes (Section 4.4) • Resource Budgeting for Mixed Multithreaded Video Applications (Section 4.5) *Hierarchical Energy Budgeting for Computation Offloading (Section 4.6)* • Multi-Granularity Energy Budgeting (Sections 4.6.1, 4.6.2, 4.6.4) • Region of Interest Extraction (Section 4.6.3)
Hardware Layer	*Video system I/O and communication (Section 5.1)* • Video Input Pipeline (VIP) (Section 5.1.2) • Custom Communication Interface (Section 5.1.4) *Accelerator Allocation and Scheduling (Section 5.2)* • Sharing Hardware Accelerator among multiple applications/threads (Section 5.2.1) • Hardware Accelerator Sharing for Multicasting (Section 5.2.2) *Accelerator Architectures for H.264/AVC and HEVC (Sections 5.3)* • Distributed Hardware Accelerator Architecture (Section 5.3.2) *Power-Efficient Memory Subsystem Design* • Hybrid Memory Architecture (DRAM + SRAM + MRAM) (Section 5.4) • SRAM Anti-Aging Circuits (Section 5.5)

1.6.1 Contributions at the Software Layer

1.6.1.1 Power Efficient Resource Budgeting/Parallelization

Power-Efficient Parallelization: Here, we consider to minimize the power consumption of a video application, while meeting the throughput constraints imposed due to the resolution of the video frame and FPS requirements [44]. The goal is to minimize the number of cores used and to reduce their frequencies as much as possible. The application's workload and hardware characteristics are used for parallelization and selecting the frequency of the cores on a many-core system. Also, the video frame is adaptively divided into tiles and a thread is associated with each tile. The proposed approach tries to balance the workload of the tiles among all the running cores. At runtime, the application-specific workload is fine-tuned using a closed-loop for further frequency (and hence power) reduction. This approach not only accounts for the hardware, but also for the system's load variation (due to parallel running applications) and content-dependent complexity. This mechanism enhances the portability of the approach, as it adjusts the number of cores and frequency of the cores at runtime. More information about this contribution can be found in Section 4.2.

Resource-Efficiency: To reduce the number of processing nodes and balance the workload, video processing jobs are packed into subtasks, structured in a manner that multiple subtasks can be packed and dispatched to a single compute node, which can process these subtasks in a time multiplexed manner [45]. This distribution process accounts for the compute capabilities of the underlying cores. A job queue is populated with subtask threads and the available cores fetch the subtasks from this queue. Additional information can be looked up in Section 4.2.2. Moreover, the same scheme is extended to distribute processing jobs and balance the workload among heterogeneous computing nodes. This is further explained in Section 4.4.

Resource- and Power-Budgeting: In addition, resource and power budgeting of multiple, multithreaded video applications (concurrently executed on a many-core platform) is considered in this work [46]. A hierarchical approach distributes the resources and power among the applications, depending upon their throughput requirement, and their resource and power utilization history. At runtime, the resources and power allocated to each application is tuned to provide fairness among the applications. For more information, reader is directed to Section 4.5.

1.6.1.2 Power Efficient Software Design

The software layer is also used for software-guided power management. Basically, application-specific algorithms are designed for resource-constrained device which run high complexity, deadline-conscious applications, like HEVC and H.264/AVC [47, 38]. Here, the number of modes (or "searches") performed for generating the best output video quality is adapted, such that the complexity is considerably reduced with minimal video quality degradation. If the complexity of the algorithm is reduced, one can reduce the frequency of the hardware and thus power consumption will also reduce. Thus, application-specific properties are leveraged to tune both complexity and power knobs, according to the design's needs. The complexity reduction approaches presented here are also utilized with workload tuning (see Section 1.6.2.1). More information about complexity management can be seen in Section 4.3.

1.6.1.3 Energy Budgeting and Computational Offloading

In case the workload cannot be sustained by the computationally constrained device, offloading mechanisms can be utilized to offload its workload to a high-end device. A typical use-case is DVC and HDVC, whereby it is assumed that the encoder is computationally and energy-wise a constrained device, and the decoder is a high-end device. For a single-encoder single-decoder scenario, mechanisms are proposed to determine the amount and locality of the workload that is offloaded from the encoder to the high-end decoder [48]. A user-requested time duration of processing the videos at the encoder, is used to determine the percentage of subtasks offloaded to the decoder and for meeting the throughput demands of the encoder. A content-dependent, video frame based, hierarchical energy budget distribution results in low video quality degradation under both throughput and energy constraints, at both the encoder and decoder sides. In Section 4.6 this approach is explained in detailed.

1.6.2 Contributions at the Hardware Layer

1.6.2.1 Power Efficient Accelerator Design

Video I/O and Communication among Heterogeneous Nodes: The hardware architecture of a low latency design for video I/O with the processing system is presented. Similarly, a communication architecture is presented which can be used for processing multiple subtasks of the same video based workload on custom multi-or many-core hardware platform. The architecture is designed to support the workload distribution and balancing decisions on this multi-/many-core paradigm. A master core assigns the workload of all the tiles to the other secondary cores using a custom interface. The approach is functionally verified on a real FPGA, using multiple, embedded soft-cores (i.e., programmable, compute cores on FPGA). More information is available in Section 5.1.

Hardware Accelerator Design: Multiple hardware accelerators are proposed in this thesis, especially for video coding applications [49, 50]. A distributed hardware accelerator design methodology is presented which can be used in conjunction with clock-gating (or power-gating) to increase the power efficiency. These hardware target the most computationally intensive part of HEVC and H.264/AVC, i.e., the evaluation of different modes to determine the right compression technique. For further information, refer to Section 5.3 in this document.

1.6.2.2 Shared hardware accelerator scheduling

Power-Efficient Shared Hardware Accelerator Allocation: The target of the approaches presented in this domain is power-efficiency of a multi-/many-core system with shared hardware accelerator [51]. The hardware layer provides support to the offloaded subtasks from the compute nodes to the custom hardware accelerator, for reducing computational complexity and power consumption of the system. In addition, they provide opportunities to exploit the Dark Silicon. Offload-scheduling mechanism are developed to minimize the system power, by adaptively determining the subtasks assigned to the compute cores and the accelerator, while fulfilling the throughput constraints. To maximize the throughput, the slack of each parallel running thread/application is minimized. The approach devises an optimization problem and proposes a heuristic to find the optimal percentage of workload offloaded to the accelerator. The goal is to maximally utilize the accelerator while meeting the throughput demands of parallel running threads/applications. The same approach can be extended to process multiple videos processing workloads. More information can be seen in Section 5.2.1.

Multicast H.264/AVC Encoder Design: In addition, a multicasting, fully custom, H.264/AVC video encoder architecture is designed using area- and power-efficient building modules [20, 49]. Hardware among different video encoders is shared (using a hardware scheduler and a re-scheduler) such that little or no penalty is incurred. The complete system integration (along with camera input, memory access and Ethernet output) is given in Section 5.2.2 and Appendix C.

1.6.2.3 Memory Subsystem Design

Hybrid Memory Architecture: For the proposed video system, the hybrid memory subsystem is designed using a NVM (specifically, Magnetoresistive Random Access Memory, MRAM) in conjunction with an SRAM [39]. The design exploits the characteristics of both NVM and SRAM such that maximum power-efficiency is achieved at minimal or no latency-penalty. Large MRAM buffers are used for storing video frames and SRAM FIFOs are used for input/output from these buffers. The MRAM frame buffer memory is sectored and these sectors are normally OFF. These sectors are powered ON only in the case they are needed to be accessed. An unsupervised learner (Self-Organizing Map, SOM) is used for predicting the next sector that needs to be powered ON. Thus, this saves leakage power and accesses to the external memory. For further information, a reader is referred to Section 5.4.

SRAM Anti-Aging Architecture: To reduce the aging of SRAM cells, deteriorated by NBTI-induced stress cycles, this thesis proposes to adapt the data written to and read from the SRAM memory [52, 53]. Memory Read Transducers (MRTs) and Memory Write Transducers (MWTs) are proposed, which adapt the video data bits on-the-fly, read from and written to the SRAM. MRT and MWT incur minimal latency and avoid extra read/write of the SRAM. More information about SRAM anti-aging is given in Section 5.5.

1.6.3 Open-source Tools

In order to contribute to the research and technical community and give them a head-start in designing power-efficient video systems, several open-source tools (pertaining to the proposed novel contributions) are made available for download. The intention is that these tools can be used for testing and extending the proposed contributions, or, for testing new design paradigms.

Parallel HEVC Encoder: For testing the parallelization, workload balancing and resource allocation approaches, an in-house, C++ based, multithreaded, open-source, HEVC video encoder in developed, called ces265 [44]. This encoder is light-weight, and highly adaptable and flexible, e.g., different parallelization settings can be introduced by the designer and different tile structures can be proposed. A thread of ces265 is about ~13× faster than the reference software (HM-9.2). Further, the NIOS-II multi-core Altera FPGA

implementation for this encoder is also provided. More information about this tool is available in Appendix B.

Multicast H.264/AVC VHDL: Similarly, for the multicasting solution for H.264/AVC, the complete VHDL of the H.264/AVC encoder (functionally verified on an Altera FPGA) is provided. The VHDL is developed using an in-house H.264/AVC encoder, which is modified to generate test vectors and simulation statistics. The implementation includes processing a real camera input and generating Ethernet output. The MATLAB scripts for testing and simulating the hardware designs are also provided to the video compression community. For more information, refer to Appendix C.

SRAM Aging Analysis GUI: For video memory aging analysis and visualization, a GUI-based tool is developed and released. Written in C#, the tool can display videos and one can implement multiple anti-aging circuits. This tool detects memory regions under high stress, and automatically plots the stress regions, histograms and other statistics. Refer to Appendix D for further information.

1.6.4 Thesis Association with Research Projects

This thesis contributes to the following major research projects currently carried out in Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany.

SFB Transregio 89 – Invasive Computing [54]**:** The basic idea of Invasive computing is to provide: (a) resource aware programming of applications by exploring their design space for possibly heterogeneous platforms, under (b) predictable performance guaranties and, (c) modern design challenges (like Dark Silicon and energy-efficiency). These goals are achieved by writing programming constructs, which allow an application to “Invade” the resources, e.g., cores and memories, and then “Infect” them by copying code and data to these resources. Once the processing tasks are finished, the application “Retreats” by releasing its occupied resources. Invasive Computing project is itself divided into multiple subprojects:

A- Basic algorithms related to scheduling/load balancing and design-time modeling of invasive patterns.
B- Design and research of application-specific architectures, and power- and energy-efficient Multiprocessor System on Chips (MPSoCs) within Invasive Computing platforms.
C- Runtime supports for Invasive systems, design-space exploration, code compilation and security.
D- Hardware architectures for robotics, and Invasive computing for high performance computing.

As seen, the scope of this thesis covers multiple subprojects of Invasive Computing. Several publications originating from this thesis are in line with Invasive Computing’s methodology. Specifically, this thesis contributes towards subprojects A (scheduling and workload balancing, modeling), B (application-specific accelerators and enabling power-efficient designs) and C (runtime power and resource budgeting).

DFG SPP 1500 - Dependable Embedded Systems [55]**:** To provide reliability, dependability and aging resilience for next generation fabrication technologies (with shrinking feature sizes), this project aims at inciting dependability techniques for both hardware and software levels. The objective of this project is to enable cost-effective features size reduction, which is achieved by employing efficient resiliency techniques to tackle soft/hard errors. Moreover, dependable software/hardware design methodologies are also developed under this project. In our lab (CES), numerous subprojects are carried out under the umbrella of SPP 1500. These projects are Get-SURE (software level reliability approaches) [56], OTERA (test strategies for dynamically reconfigurable architectures) [57] and VirTherm-3D (dependable hardware architectures) [58]. In general, one of the targets of this project is device aging-modeling, aging-reduction and aging-abstractions (from circuits to the software layer), via design- and runtime optimizations. In particular, the scope of this thesis coincides with the VirTherm-3D project, as this thesis proposes a power-efficient architecture for reducing the NBTI-induced aging of SRAM memories.

1.7 Thesis Outline

This Ph.D. thesis is structured in the following manner (more information can be found in Table 1-1).

- Chapter 2 provides detailed background and state-of-the-art on the challenges addressed in this work. Essential background knowledge of video applications is provided to get the reader accustomed with the jargon and the contributions of this work. The Dark Silicon paradigm is deliberated in conjunction with video systems implemented on a many-core and customized hardware platforms.
- Chapter 3 provides a comprehensive video system design of this work by integrating different components of the system. The communication flow between the software and hardware layers is discussed, and the architectural design and its assumptions are outlined. Several motivational analysis are performed for parallelization, workload variation, and memory subsystem's power and aging.
- Chapter 4 discusses the video system software layer in detail. Power-efficient parallelization and resource budgeting are provided for mixed multithreaded workloads. Various complexity reduction methods are discussed. Computational offloading, specifically DVC and HDVC, are also elaborated here.
- Chapter 5 relates to the video system hardware layer. It discusses the shared hardware scheduling among competing cores and for multicasting scenario. The hybrid memory architecture design is also given, along with power-efficient, SRAM anti-aging circuits.
- Chapter 6 provides experimental setup and discusses the experimental results for the proposed approaches and Chapter 7 concludes the thesis with an outlook of the future extensions to this work.
- The algorithms used in the proposed technical novelties are given in Appendix A. The inner workings of the ces265 video encoder are presented in Appendix B while Appendix C discussed the HDL of proposed multicasting H.264/AVC video encoder. Appendix D is related to the SRAM aging analysis tool.

Chapter 2 Preliminaries and State-of-the-Art

This chapter provides details regarding the basics of video processing in general, while specifically targeting the video coding applications. Fundamentals of HEVC and H.264/AVC video encoders are followed by their associated challenges while designing computationally-efficient video processing systems. Modern technological challenges that are addressed while designing such systems are also discussed. Afterwards, the state-of-the-art approaches to meet these design challenges are given, with details targeting video processing system's software and hardware layers.

2.1 Video Processing Overview

As discussed in Section 1.2, the computational complexity of video processing depends upon the type of the algorithm, along with the video frame dimensions and the FPS requirements (given by f_p). From FPS requirements, the maximum time that can be spent on processing a frame is given by $t_{frm}=1/f_p$. A general metric to present the throughput requirements is given by $w \times h \times f_p$, which denotes the number of pixels that must be processed in one second, for a frame of size $w \times h$ pixels. However, most video algorithms process a block of pixels, i.e., all the pixels within the block correspond to a particular computational mode. An example block division for a video frame is shown in Figure 2-1. Generally, the video is processed online or offline, and a video frame is divided into blocks of size $b_w \times b_h$. These video frames are stored in the external memory due to the size of the video frame. A block (or a group of blocks) is brought from the external memory to the on-chip memory for processing. Once processed, the output of the processed block is concatenated to other blocks and written to the output.

Figure 2-1 presents an example video processing system, with pre- and post-processing modules. As an example, an on-line gaming scenario can be considered. The application reads the data (possibly compressed video) from the gaming server, decodes it and then applies different image enhancement techniques before finally forwarding it to the main processing engine. The processing engine also uses data on the local disk and video content captured using the camera(s) at the user's end for processing. Afterwards, this data is displayed on the display-device, using post-processing filters like light blooming, interpolation etc. Further, the video formed at the user's end is also encoded and encrypted, and finally sent back to the gaming server. Note that pre- and post-processing modules themselves can be separate video processing systems, employing the same principles as given in this figure.

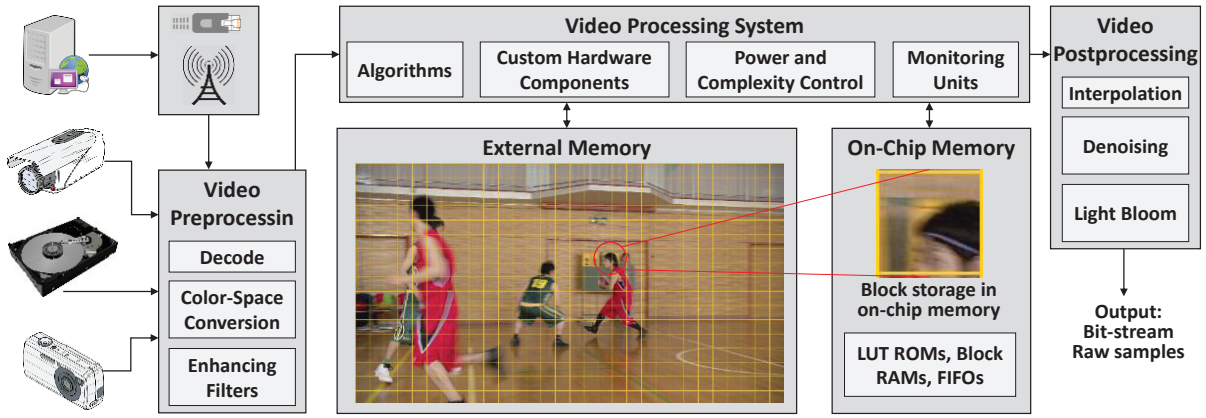


Figure 2-1: Basic video processing system. The frame is divided into blocks and logically stored in the external memory as shown in the figure. A block of size $b_w \times b_h$ is processed at a time

For block-based video processing, the throughput requirement can be written as:

$$n_{\sec} = n_{frm} \times f_p = \frac{w \times h}{b_w \times b_h} \times f_p \tag{2-1}$$

Here, n_{sec} presents the number of blocks that must be processed per second, and n_{frm} is the number of blocks within a frame. As noted, a larger n_{sec} corresponds to a more computational and power requirements.

A video sample can be stored in multiple formats. Usually, a sample is presented as an amalgamation of three colors, red (R), green (G) and blue (B). This color space format is termed as RGB format. If each color sample is presented by 8-bits, then a video sample is stored as a 24-bit value. Usually, R, G and B "planes" of the frame are stored separately. Therefore, the number of addresses (or pixels) is three times more than expected. Therefore, the size of the frame can be presented in number of bits b_{frm} by:

$$b_{frm} = w \times h \times 3 \times \text{bits\_per\_sample} \tag{2-2}$$

Usually, the RGB color format carries a lot of redundancy and burdens the storage/communication. The redundancy can be removed by transforming RGB to different color spaces while still presenting visually distinguishable features to the human user. For example, a large number of video algorithms also work with the YCbCr format (also sometimes called as YUV format), where Y presents the luminance component of the sample, and Cb and Cr collectively determine the chrominance of the sample. YCbCr 4:4:4 format means that for every luminance pixel, there are two chrominance pixels, whereas YCbCr 4:2:0 format means that the two chrominance pixels are associated with 4 luminance pixels. Hence, both the chrominance planes are downsampled by 2, in horizontal and vertical directions. This still results in high visual quality because the human eye is much more susceptible to the luminance component than the chrominance components. For YCbCr 4:2:0 case, the size of the frame can be written as:

$$b_{frm} = w \times h \times (1 + 0.25 + 0.25) \times \text{bits\_per\_sample} \tag{2-3}$$

Figure 2-2 illustrates the baseline memory architecture usually deployed in a real-time camera-based image/video processing system. The camera captures videos in real time at a certain frame acquisition rate in terms of frames per second (typical values are 30, 60 etc.). The raw data is usually preprocessed via a Video Input Pipeline (VIP). The streaming data, containing YCbCr 4:2:0 samples, is converted into words of size ≥ 8 bits using a combination of a FIFO and a shift register. Each frame is stored in a frame memory partition in the video memory. The video memory is composed of multiple frame memory partitions in order to simultaneously store multiple video frames, and support double- or triple-buffering. Frames are written to these frame memory partitions using Write Address Generating Unit (AGU) and frames are read via Read AGU. The application request a new frame once it has processed the previous frame. Frame drop mechanism (i.e., previously stored frames are overwritten, although they are not processed) is also supported, in case the video processing application has high complex, or, the underlying hardware is constrained and unable support the required frame-rate. Frame drops can also occur if the output of the video system is stalled (e.g., due to high congestion and packet loss in wired/wireless output scenarios, the output disk is full).

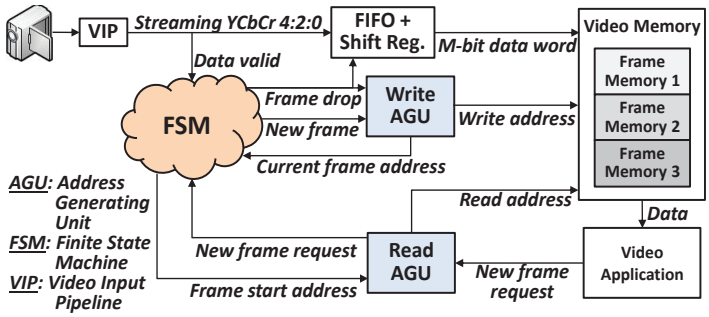


Figure 2-2: Video memory management system for storing raw video samples in the video memory

Video processing algorithms basically employ two different types of frame processing algorithms.

1- Intra frame processing, where the spatial neighborhood of the pixel (or block) under process is used for computations. That is, the same frame is used for processing the current pixel/block.
2- Inter frame processing, where the temporal neighborhood of the pixel (or block) under process is used for computations. That is, the frame(s) processed in the past are used in calculations involving the current pixels/block.

The spatial and temporal correlations are used for multiple purposes, e.g., noise filtering, motion tracking. Mostly, the spatial neighbors are not highly correlated with the current pixel or block, therefore, the video output quality is not that high for Intra frame processing, as compared to Inter frame processing. However, the computational requirement of Inter frame processing is much higher as a large amount of data transfer between external and on-chip memory will occur. It will not only increase the data processing pressure but also the power overhead. A larger complexity corresponds to searching for the best mode of computations and hence increases the output video quality. The output video quality can be compared against an ideal output using Peak Signal to Noise Ratio (PSNR), and Bjøntegaard Delta Peak Signal to Noise Ratio (BD-PSNR) or Bit-Rate (BD-RATE) [59]. This will become clear when we discuss video coding overview in Section 2.2.

Generally, small parts of the video processing algorithm's code (called kernels) take a large portion of timing complexity and power. In order to reduce the computational complexity, these kernels are optimized by exploiting past-predicts-future paradigms and by tuning the complexity knobs. For example, the number of modes searched for attaining the best mode can be reduced, which increases the throughput-per-watt metric, but may reduce the output video quality. These kernels can also be implemented as custom hardware accelerators.

2.2 Video Coding Overview

This section elaborates video coding algorithms that employ the principles mentioned in Section 2.1. More specifically, this section targets H.264/AVC [32] and HEVC [7] video encoders. The working principles of a PVC video encoder is shown in Figure 2-3. An original block (of size $b_w \times b_h$, with samples s) is brought from the external memory, and is tested via both Intra and Inter modes. Afterwards, the best mode is forwarded for processing using the mode decision module. The purpose of both Intra and Inter prediction modes at the encoder is to generate a resembling representation of the original block under test, called prediction (with samples s'), using the reconstructed video frame samples (frames generated at the decoder). The generation of prediction either uses the spatially neighboring pixels (Intra frame processing) or temporally neighboring pixels (Inter frame processing). Thus, if the quality of prediction (its resemblance with the original block) is high, the residual block, i.e., the difference between the original and the prediction, has low pixel energy (samples with low

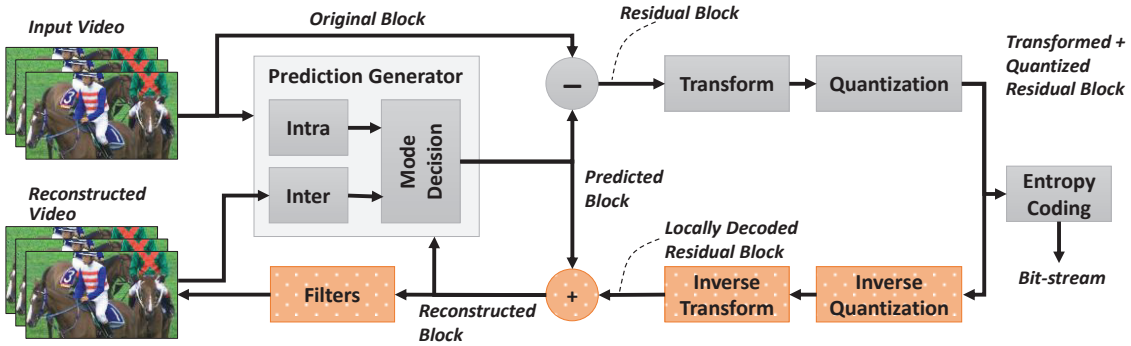


Figure 2-3: Basic modules of video encoder. The gray boxes are used for encoding purposes and the orange boxes denote the local decoder modules used inside the encoder.

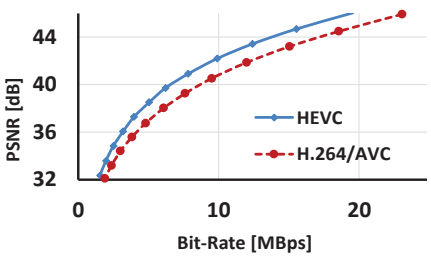


Figure 2-4: Video quality comparison for HEVC and H.264/AVC

Figure 2-5: Video sequence "RaceHorses" with overlaid PU structure after HEVC processing, with a CTU of 64×64

values). The quality of prediction is usually tested using the Sum of Absolute Differences (SAD) metric:

$$SAD = \sum_{y=0}^{bh-1} \sum_{x=0}^{bw-1} \left| s(x,y) - s'(x,y) \right| \tag{2-4}$$

Here, (x, y) is the pixel location within the original and predicted blocks. It is evident that, if the number of predictions that are searched and tested for the best residual, is high (i.e., a more complex encoder), the pixel energy is lower and a lower SAD value is achieved. Afterwards, transform of residue and its quantization takes place, before finally, the bit-stream generation (i.e., the entropy coding module) compresses the residual block and forms a bit-stream which is sent to the video decoder. The size of the bit-stream is proportional to the pixel energy in the residual block, because more energy in the residual block will increase the size of the bit-stream.

This bit-stream consists of the transformed and quantized residue, along with the prediction mode. The decoder can reverse the process, by generating the (approximated) transformed and quantized residual block from the bit-stream, by (a) inverse quantizing the block, (b) inverse transforming the block, and finally (c) adding the prediction to generate a (lossy or lossless) representation of the original block. A version of the decoder is also locally implemented at the encoder, because frame processing requires previously encoded frames which will be available at the decoder for prediction generation.

In addition to video encoding/decoding algorithms (which attracts high attention from research community), video input and output pipelines are also critical design steps. Video input pipeline involves color-space conversion, deinterlacing, downsampling etc. More on this is covered in Section 5.1.2. Video output pipeline usually incorporates noise reduction algorithms, color-space conversion and content-enhancement (like increase the sharpness, correcting brightness etc.).

2.2.1 H.264/AVC and HEVC

H.264/AVC is the current video compression industry standard developed by JCT-VC [32, 60]. Here, a video frame is divided into blocks of 16×16 pixels, called Macroblocks (MBs), and each MB is treated as a separate entity for processing. A block is tested with Intra prediction modes using angular directions, and Inter prediction via block matching (also called Motion Estimation, ME) [61, 62]. On the other hand, due to the trend of increasing video resolutions and frame rates (e.g. 8K Ultra HD, 7680×4320 pixels, at 120 fps), JCT-VC has recently developed the next generation High Efficiency Video Coding (HEVC) standard. HEVC aims at increasing the compression efficiency by 50% compared to the H.264/AVC, however, it comes at the cost of significantly high computational workload at the encoder side. Our experiments with H.264/AVC and HEVC reference software denote that HEVC is ~2.2× more complex than H.264/AVC. From Figure 2-4, we notice that the bit-rate of the video encoded via HEVC is considerably lower than H.264/AVC, for the same video quality (PSNR). These curves are also called Rate-Distortion (RD) curves. On the other hand, HEVC time consumption is substantially higher.

Compared to H.264/AVC, HEVC brings two major innovations [7, 63]. Firstly, unlike the concept of MB of its predecessor coding standard H.264/AVC, the HEVC introduces the concept of the Coding Tree Unit (CTU) as a coding structure; see Figure 2-5. A CTU is a square block of size 16×16, 32×32 or 64×64. The CTU is recursively sub-divided into multiple Coding Units (CUs) and Prediction Units (PUs). Each PU serves as an independent, basic entity for compression carrying individual header data. Secondly, each PU may be predicted using one out of many standard-defined prediction modes.

2.2.1.1 Intra-Prediction Modes

H.264/AVC and HEVC Intra compression modes generate the prediction pixels using the spatial neighboring blocks. This is because spatially, the texture of the video samples is expected to continue from the surrounding blocks into the original block. However, the proper direction of texture flow must be estimated with high probability for best representation of the original block via the prediction block and therefore resulting in a residual block with low pixel energy. As shown in Figure 2-6 (a, b), both H.264/AVC and HEVC employ different spatial directional (or angular) modes to determine the best texture flow direction. The encoders employ a brute-force search algorithm to get the best prediction mode. The choice of the best prediction mode is usually the mode which corresponds to the lowest SAD (Equation (2-4)). The SAD value is computed between the current block and the prediction block generated by the reconstructed video samples. Thus, this means that processing video blocks cannot be pipelined because processing the current block requires the output of processing the spatial neighboring blocks.

Table 2-1 provides a comparison of Intra prediction modes for both HEVC and H.264/AVC. It is noteworthy that the total number of prediction modes for the HEVC is significantly higher compared to that in the H.264/AVC. Furthermore, the selection of Rate-Distortion (RD)-wise best PU size and prediction mode is determined by a RD Optimization (RDO) process. Therefore, due to the recursive nature of best PU size selection and RDO process, the total number of mode decisions in HEVC (computed using Equation (2-5) for a given CTU of size $b_w \times b_h$=64×64) is ~2.65× more than that in H.264/AVC. n_{ang} denotes to the ordered set of number of angular modes for a particular PU size (see column 2 of Table 2-1).

Table 2-1: Comparing HEVC Intra prediction modes for 64×64 CTU with the H.264/AVC Intra modes for a 64×64 image region

Prediction Size	Total Intra Angular Modes n_{ang}	
	HEVC/ H.265 (64×64)	H.264/AVC (16×16)
64×64	4	NA
32×32	35	NA
16×16	35	4
8×8	35	9
4×4	19	9
TOTAL MODES	ψ = 7808	16×(16×9+4×9+4) = 2944

$$\psi = \sum_{i=0}^{\log_2 bw-2} \left(2^{2i} \times n_{ang}\right) \tag{2-5}$$

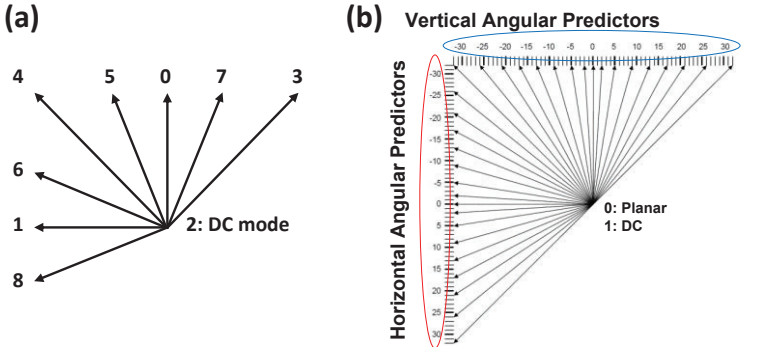


Figure 2-6: (a) H.264/AVC Intra angular modes, (b) HEVC Intra angular modes.

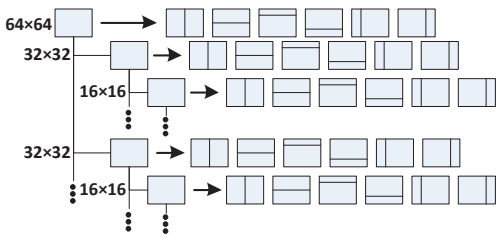


Figure 2-7: HEVC Inter modes.

2.2.1.2 HEVC Inter-Prediction Modes

Similarly, for generating the Inter predictions, both H.264/AVC and HEVC encoders search for a similar block in the previously encoded frames (called reference frames) using ME. In HEVC, ME is repeated for every PU as shown in Figure 2-7. This search is pixel-wise and uses the SAD metric. The ME process is shown in Figure 2-8 along with the hardware architecture of computing the SAD. In the Inter-encoding case, a predictor is a block of pixels of the previous frame. Usually, only parts of reference frames, called the search window, are tested for the best predictor. The search window is brought from the external memory to the on-chip memory, and predictors are searched in this search window. Search window is a rectangular region of size $s_w \times s_h$. For testing the predictors to get the best match, ideally the search window should be loaded into the on-chip memory. Additionally, note that blocks are processed in a raster-scan order, and two neighboring blocks will have most of their search windows overlapped [61], as shown in Figure 2-9. Thus, search window can be reused and "prefetching" of new reference samples can be done in parallel to the ME process. Further, a larger search window increases the probability of finding a predictor with low SAD value (and thus increases the video quality). However, this also increases the on-chip memory required to store the search window, which increases the power consumption of the system. Additionally, the external memory accesses also increase, resulting in higher power consumption [64, 65].

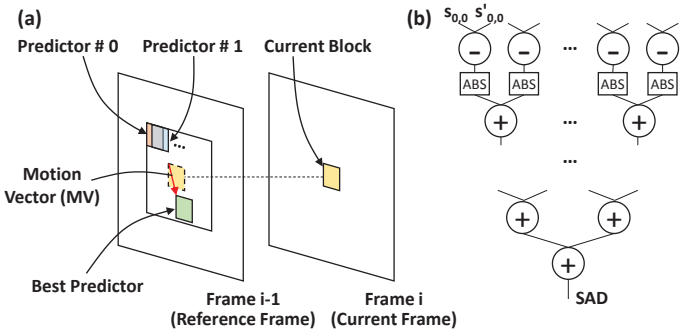


Figure 2-8: Motion estimation process and SAD computation

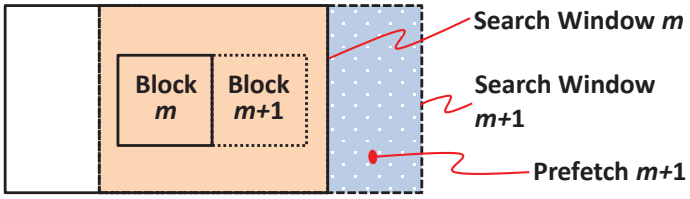


Figure 2-9: Search window structure and prefetching

Moreover, a single reference pixel is written multiple times to the search window, depending upon the height of the window. For example, a larger height (s_h) denotes that a pixel will be written more times than having a smaller search window height. A read factor r_f is defined which denotes the total number of times a pixel in the reference frame is read and then stored in the search window. Typical value of this factor is between 3 and 12. For the approach given in Figure 2-8, where the search window is shifted by b_h for every new row of blocks with size $b_w \times b_h$:

$$r_f = s_h / b_h \tag{2-6}$$

There are multiple ways to determine the best predictor. One method is to search every predictor in the search window. This scheme employs a brute-force algorithm and results in the best predictor with the lowest SAD value. However, this scheme is unreasonably complex and almost never employed in practice. Usually, fast prediction search algorithms (like EPZS, TZ [66]) are employed, which result in considerably lower power consumption and complexity with almost no video quality loss, compared to the brute-force search. In these fast algorithms, not every possible predictor of the search window is searched (i.e., not all pixels of the search window are utilized), rather the most probable predictors which will result in the lowest SAD values are iteratively tested. However, this does not mean that the number of pixels "prefetched" will reduce. The number of pixels fetched from external memory will remain the same, because the direction and pixels that will be utilized by the ME algorithm are unknown. Moreover, if a part of the search window is not fully utilized for the current block, it might be utilized for the next block.

In HEVC, each CU is split into 13 PUs (see Figure 2-7) and ME is performed for these PUs. Mathematically, the total number of block-matching iterations for a square CTU of width b_w can be given by:

$$\psi = 13 \times \sum_{i=0}^{\log_2 bw-3} 2^{2i} \quad (2\text{-}7)$$

Our simulations show that block-matching in HEVC takes around 80% of the total encoding time. It is about ~2.2× complex as compared to H.264/AVC. Also, in video encoders, generally the block-matching takes about 60% of the total energy [67]. Moreover, there can be multiple reference frames (e.g., Bi-prediction in HEVC and Multi-View Coding, MVC), and a block is searched in a window for each of these frames.

Most of the algorithms used in both H.264/AVC and HEVC can be extrapolated for other video processing algorithms, as they exercise the same principles of computations and memory access. A vast majority of video encoders use the same principles of Intra and Inter video encoding (e.g., Google's VP8 and VP9, Microsoft's SMPTE, Cisco's Thor, MJPEG). Moreover, the ME algorithm is also used in super-resolution techniques (increasing the resolution of a video frame by concatenating multiple, temporally neighboring frames), in temporal frame interpolations (used in HDVC and for reducing flicker), motion tracking, corner detection etc.

Further, almost all video processing algorithms have application configuration knobs, which can be tuned in order to leverage the computational complexity and output quality. For example, the search window for motion estimation in video coding, deniosing, frame interpolation and super-resolution algorithms can be enlarged/contracted. The number of Intra angular modes tested for HEVC can be increased/reduced. Hence, video processing applications provide opportunities for runtime adaptation of their workload. However, such adaptation might result in lower output quality, e.g., loss in PSNR or precision of tracking.

2.2.2 Parallelization

Like all compute intensive video applications, H.264/AVC and HEVC standards allow for parallel implementation. A system designer can utilize these specifications on a multi- or many-core system and exploit the parallelism to gain computational advantages. To process a video sequence, a hierarchical approach for video partitioning is employed. Video frames are divided into set of frames, called Group of Pictures (GOPs). Each GOP can be processed independently of other GOPs, thus supporting GOP-level parallelization [68]. Within a GOP, video frames can also be processed in parallel on independent compute cores [69]. A frame is divided into slices or tiles (see Figure 2-10 (a)), and each slice/tile can be processed in parallel [70, 71, 44]. As discussed before, usually the image and video processing algorithms are block-based algorithms, where a set of pixels in a rectangular region is considered as a basic processing entity. Each slice/tile is divided into blocks and a single block is processed at one time (the blocks within the tile are processed sequentially). However, Intra angular predictions and Inter predictions can be tested in parallel. An example video partition hierarchy is shown in Figure 2-10 (b), where each of frame in the GOP is divided into k_{tot} tiles. This figure also shows

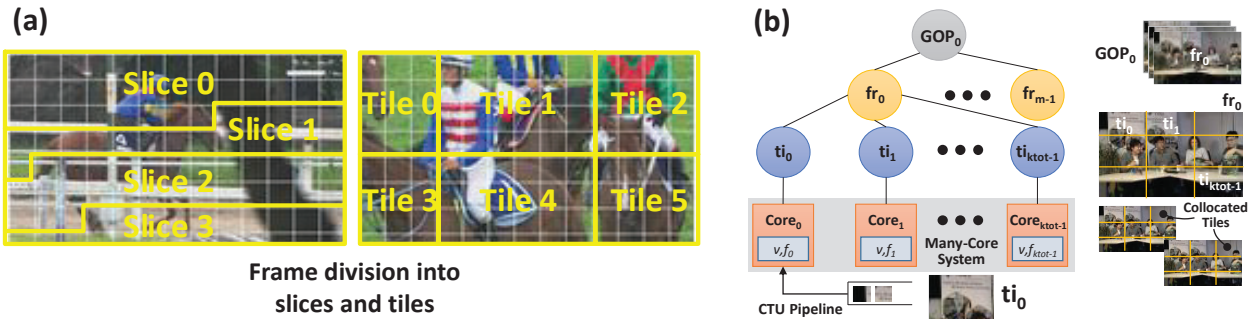


Figure 2-10: (a) Frame division into slices and tiles for parallel processing, (b) Video partitioning hierarchy. Here, fr: frame, ti: tile

collocated tiles, which are the same video tiles but in adjacent frames.

Video Workload Balancing: Via parallelization, the workload of the whole application is divided into chunks and thus, the workload corresponds to the compute complexity and energy consumption of a processing core. That is, larger the workload, more computational complexity and energy consumption of the core is expected. Generally, the workload of the complete application should be distributed to the cores in a manner that the hardware utilization is maximized. Therefore, every core should ideally consume the same amount of time in processing the jobs or subtasks assigned to that core. This will increase the throughput of the system and also increase the throughput-per-watt metric.

Load balancing can be done via centralized or distributed approaches [72, 73], which gather statistics and objectively distribute the subtasks among the compute cores. However, for fully distributed strategies, optimal scheduling decision is difficult to make due to rapidly changing environment, randomness and unpredictability. The communication delays in fully distributed strategies can also nullify a correct decision at a previous time, as the state of the system might change rapidly after the distributed load balancing decision. Further, distributed algorithms reduce the range of subtasks migration from one core to another, because physically far apart resources, with highest and lightest workloads cannot be balanced in a single iteration of load balancing. Also, the overhead of the scheme increase with the system size due to increased message passing, control logic, scheduling algorithms etc. Further, the response of distributed strategies saturate after tens of compute nodes. For the centralized scheme, the overhead of the assignment algorithm is large, and has a large communication and storage overhead. This is due to the chip-wide information gathering and storage of statistics. Further, centralized algorithms are vulnerable to faults and a complete system failure will occur if the central node (running the subtasks assignment algorithm) fails. Thus, both centralized and distributed strategies for load balancing not always produce optimal performance. This discussion is not only valid for video processing systems, but also applies to any many-core system running parallel workloads.

Multicasting: Multicasting refers to information transfer to a set of predefined destination nodes. With reference to video communication, a multicasting or multi-channel video encoder should be capable of encoding concurrent [74, 75] video streams in parallel and generating appropriate compressed bit-streams for each of these videos, as shown in Figure 2-11. Use cases of multicast video encoding include security, entertainment, video logging etc. Each video can have its own resolution, texture/motion content dependent workload and throughput requirement (frames per second). This system can be implemented using a many-core platform, or by designing custom hardware solutions. In case of a many-core system, it is a design challenge to efficiently distribute the cores and power among multiple, concurrently executing encoders, while fulfilling their throughput requirements. Moreover, the load balancing approaches discussed above need to be applicable to such paradigms. For a custom hardware solution, the multicast encoder should be area-efficient (due to multiple encoders working in parallel) and must be able to efficiently access video data.

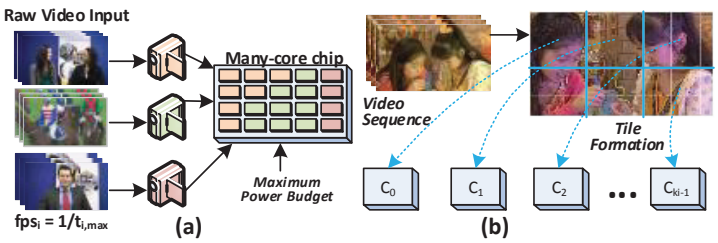


Figure 2-11: (a) Multi-video capture and encoding on a many-core chip, (b) frame division into multiple tiles

2.2.3 DVC and HDVC

As discussed, the high compression efficiency of H.264/AVC and HEVC has enabled a wide range of applications under low-bandwidth constraints. These CODECs follow the Predictive Video Coding (PVC) model, where the predictions are generated at the encoder side (using Intra angular modes or ME). PVC is

typically well-suited for scenarios where encoding devices have high computational power, while decoders are resource-/power-constrained devices, like mobile devices engaged in video streaming etc.

2.2.3.1 Distributed Video Coding

It is possible that the computational complexity of a video application is too high to be feasible on a system. For example, the significantly increased computational efforts at the encoder in HEVC prohibits its use in constrained encoded scenarios. Recently, Distributed Video Coding (DVC) has been emerged as an attractive solution for scenarios where the encoding devices are resource-constrained and must use low-complexity encoding (like in-field wireless video sensor nodes, small autonomous flying robots, mobile devices with low processing capabilities, etc.), while the decoding devices have high computational power (like high-end servers) and can execute high-complexity decoding tasks. DVC paradigm provides means to shift/offload the computational workload from the video encoder to decoder [76, 77, 34, 78, 79, 80, 81].

A DVC encoder typically consumes only 7% of the total power consumed by a PVC H.264 encoder [82, 83, 84]. In DVC paradigm, instead of the encoder, the decoder performs the ME for interpolation, extrapolation, and upsampling, while exploiting the inter-frame correlation in order to generate an estimate of the input video sequence (see Figure 2-14 (a)). At the DVC decoder side, Slepian-Wolf decoder and ME with interpolation, contribute towards 90% of the total decoding complexity. To improve the estimation quality, the DVC decoder demands auxiliary information from the encoder (i.e., parity bits generated by turbo coding), which result in a much higher transmission power compared to the PVC case (see Figure 2-12). A DVC encoder may require ~4× higher transmission energy compared to an H.264 PVC encoder for a video sensor node [82]. More are the parity bits, higher is the video quality at the decoder side and higher is the transmission energy at the encoder-side.

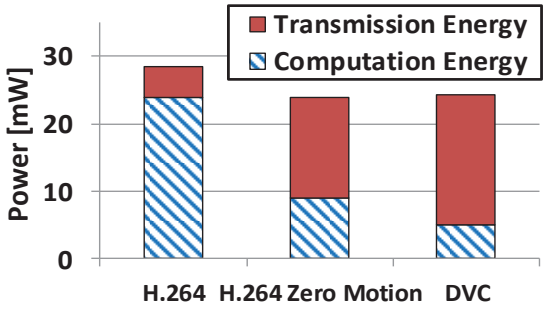


Figure 2-12: Comparing the computational and transmission power for an ASIC-based video sensor [82]

Due to resource constraints, the DVC encoder only tests spatial predictions, i.e., low complexity Intra-encoding of Key-Frames (I-frames or just IF). Other video frames between Key-Frames are called Non-Key-Frames or Wyner-Ziv frames (W-frames or WF). For W-frames, encoder only sends the auxiliary information, i.e., parity bits generated using turbo coding. The group of WFs and preceding IF is called the Group of W-frames (GOW) as shown in Figure 2-13. The number of frames in a GOW is denoted as size of GOW The decoder reconstructs the WFs using decoder-side ME and interpolation. The basic principle of decoder-side reconstruction is to exploit the correlation between two IFs. The decoder-side estimate of WFs is also called Side Information (SI), which is generated by interpolating the two received IFs. This estimate is improved by requesting more parity bits from the encoder side.

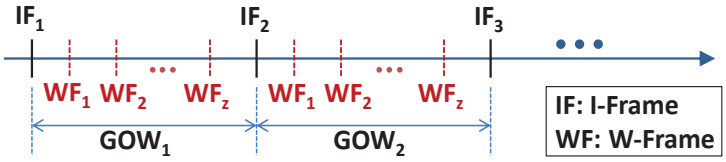


Figure 2-13: Group of WF (GOW) structure showing IFs and WFs

2.2.3.2 Hybrid Distributed Video Coding

DVC completely offloads the ME from the encoder to the decoder. The major drawback of DVC is lower video quality compared to that provided by PVC. In short, DVC is beneficial in scenarios, where the encoder-side devices are computationally constrained and have sufficient transmission power, while the decoder-side

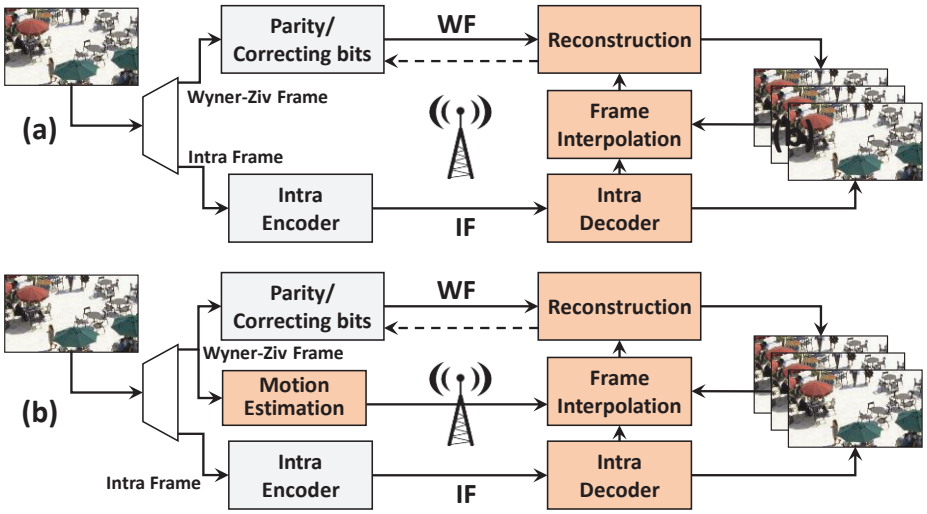


Figure 2-14: (a) Distributed Video Coding, DVC and (b) Hybrid Distributed Video Coding (HDVC)

devices have sufficient computational power. PVC and DVC become power-/energy-wise inefficient in scenarios where both encoder- and decoder-side devices are resource-constrained and/or subjected to runtime varying conditions of available energy levels and computational resources. In such scenarios, only one or neither of encoder-/decoder-side devices has sufficient computational and/or transmission power to deliver the required throughput and/or video quality. Examples of such scenarios are: (1) collaborative distributed video sensor networks for smart energy-aware surveillance; (2) mobile devices on Internet of Things (IOT) – with dynamically varying battery energy levels – communicating with each other or other power-constrained devices; (3) heterogeneous communicating devices from different vendors with distinct energy consumption properties, etc. Furthermore, DVC may not facilitate complete offloading in scenarios where multiple encoding devices concurrently offload their computational workload to a single, shared decoding device [85].

To cope with the energy-related issues for video coding in such dynamic scenarios, Hybrid Distributed Video Coding (HDVC) has emerged as an attractive solution. The architecture of HDVC system is shown in Figure 2-14 along with the DVC system for comparison. HDVC aims at combining the positive aspects of both PVC and DVC, i.e., providing high video quality close to PVC and low computational power close to DVC. In HDVC, the decoder-side ME complexity is relaxed at the cost of partial ME at the encoder side. The partial ME at the encoder side results in better reconstruction of frames at the decoder side, that corresponds to a high video quality and low energy consumption at the decoder side. However, it incurs high energy consumption at the encoder side due to ME processing. Note that ME is the most energy consuming functional module. Better ME at the HDVC encoder also reduces the number of parity bits and thus result in low transmission energy. However, this depends upon the amount of ME to be performed at the encoder side. Complex motion sequences may require more ME at the HDVC encoder side, which may not be feasible due to the unavailability of sufficient computational and battery/energy resources at the HDVC encoder side.

In a nut-shell, offloading improves the performance if [86]:

- The throughput requirement of the video application are high and is not sustainable by the constrained video processing device.
- The DVC/HDVC decoder is fast and results in computational/power benefit if the jobs are offloaded to the decoder.
- A small amount of communication of auxiliary bits takes place.
- The bandwidth of the channel between the encoder and decoder is high.

The key research challenge in the HDVC system with resource-constrained devices is to identify video frame regions (or blocks) that need to be processed at encoder sides and regions that should be offloaded to the shared decoder. The goal is to minimize the overall system energy while maintaining a high video quality. The offloading decision needs to jointly account for the computation and transmission energies, encoder devices and their compute capabilities, dynamically varying energy budgets, and throughput constraints.

2.3 Technology Challenges

Here, details about the technological challenges outlined in Section 1.3 are given.

2.3.1 Dark Silicon

Reduced transistor sizes in the modern fabrication technologies have led to new unforeseen challenges for system designers. Though the chip designers are living up to the Moore's Law challenge, it is expected that most of the transistors etched on the chip will not be fully utilized due to the power-wall problem. Specifically, the failure of Dennard's Scaling has resulted in the emergence of the Dark Silicon age, where the chip's real-estate cannot be 100% utilized continuously, at full capacity. In fact, current predictions (as shown in Figure 2-15) suggest that only 30%-50% of the chip's available resources will be bright (fully utilized) for 8nm technology, while the rest will be kept dark (unutilized) or gray (partially utilized or underutilized). This forced underutilization emerges from the fact that power per unit of area is increasing monotonously with increasing transistor density. Therefore, the temperature of the chip may reach levels which will not be contained by the available state-of-the-art coolants and result in permanent damage of the chip. Thus, power-efficient designs are of primary importance for modern systems.

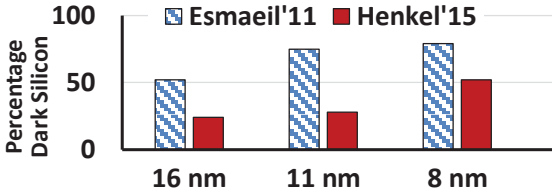


Figure 2-15: Dark Silicon prediction trends. Here, Esmaeil'11 is [8] and Henkel'15 is [299].

A digital circuit consumes two types of power, static (p_{sta}) and dynamic (p_{dyn}). The dynamic power is due to switching the transistors ON and OFF, whereas the leakage power is a result of sub-threshold current through the transistor's channel and the gate leakage, when the transistor is OFF. The static power can be reduced by lowering the supply voltage v_{dd}. For a CMOS circuit, the dynamic power consumption can be written as:

$$p_{dyn} = \alpha \cdot c_p \cdot v_{dd}^2 \cdot f \tag{2-8}$$

Here, α is the switching activity level, c_p is the capacitance of the circuit, v_{dd} is the supply voltage and f is the clock frequency of the circuit. This shows that dynamic power can be reduced by reducing the supply voltage. However, reducing v_{dd} will increase the time delay (t_d) within the device, thus, the frequency must be reduced. In fact, t_d and v_{dd} are related by the following equation:

$$t_d \propto \frac{v_{dd}}{v_{dd} - v_{th}} \tag{2-9}$$

Here, v_{th} is the threshold voltage. Therefore, we can rewrite Equation (2-8) as:

$$p_{dyn} \propto v_{dd}^3 \propto f^3 \tag{2-10}$$

Therefore, Dynamic Voltage Frequency Scaling (DVFS) can reduce the power of the circuit, owing to this relationship. However, note that reducing the frequency of the circuit also reduces the throughput, and may result in deadline misses. Moreover, as discussed above, the voltage and frequency of a processor scale together. However, this relationship does not hold for the complete frequency range, because there is a certain threshold

below which the voltage reduction results in unstable processor behavior [2]. Thus, only Dynamic Frequency Scaling (DFS) can be employed in cases where voltage is lower than this value.

For video applications, there is a strict throughput constraint that must be met. Current trends in frame resolution and FPS suggest that this throughput demand is increasing (Equation (2-1)) and hence puts pressure on the hardware to perform. However, new fabrication technologies − with about half of the cores turned OFF due to Dark Silicon constraints − require careful consideration of available TDP and its distribution among multiple, multithreaded video applications competing for systems resources. Moreover, memory may consume more than 40% of the total chip's power [64]. This power includes access to external memory, read/write energy consumption and leakage/standby power. The memory power consumption is especially of concern for video applications which are memory intensive, and their memory demands continue to grow with the growing throughput demands. Therefore, without considering the memory power, higher power-efficiency of the system might not be achieved.

The discussion above suggests that the software and hardware must be power efficient to exercise the least amount of power, while fulfilling the throughput requirements. This will not only address the Dark Silicon issue, but will provide additional power to other parallel running applications. Further, the parallelization potential of a video application must be exploited to speed up its execution. Also, hardware accelerators, running at a lower frequency/power but generating a higher throughput than its software counterpart, can be strategically placed on the die to reduce the chip's temperature. Moreover, the advantages of the new memory technologies (like MRAM) can be exploited to replace the on-chip SRAM, in order to reduce the leakage power of the memory subsystem, and limit the access to the external memory.

2.3.2 NBTI-Induced SRAM Aging

Memory intensive video applications have proliferated into various critical domains like surveillance and security, automotive, satellite imaging and video transmissions, sensor-based image/video processing over long durations, etc. For these applications, reliable operation over life-time or an extended life-time is an important system requirement. To provide fast read/write accesses, application specific architectures typically employ dedicated SRAM-based on-chip memories (like scratchpads instead of caches) for storing data, thus saving the extra power overhead of tags and other supporting circuitry. These on-chip memories are managed using specialized address generation units and/or explicitly programmed to exploit applications' attributes. However, due to continuous technology scaling resulting in small feature sizes, high power densities (Dark Silicon paradigm) and resulting temperatures, SRAM-based on-chip memories are subjected to various reliability issues like transient errors (soft errors) and permanent errors (device aging).

This work considers SRAM aging due to NBTI, which has emerged as one of the most critical reliability threats. NBTI occurs in PMOS transistors due to negative voltage at the gate (i.e., $v_{gs} = -v_{dd}$) that causes stress and breakdown of the Si-H bond at the Si-SiO_2 interface resulting in interface traps. This manifests as an increase in threshold voltage and reduction in noise margin (i.e., short-term aging) that may lead to timing errors/delay faults and/or performance degradation at runtime. To encounter this threshold voltage increase (more than 50mV [87]), the device frequency must be reduced by more than 20% over its lifetime. However, due to escalating NBTI issues and cost/power/performance constraints, the degradation of the cell stability can no longer be addressed by simply providing a design time delay margin [88]. This aging based phenomenon is partly reversed in the so-called recovery mode (the Si-H bond is reformed in a few cases) once the stress is removed from the PMOS gate, i.e., at v_{gs}=0. Figure 2-16 (a) provides an abstract view of this process for a PMOS transistor. Such a situation occurs when a "one" stored in the SRAM cell is overwritten with a "zero" and vice versa. However, 100% recovery is not possible and NBTI results in continuous degradation over years (i.e., long-term aging), such that, the total aging throughout the lifetime depends upon the stress and recovery

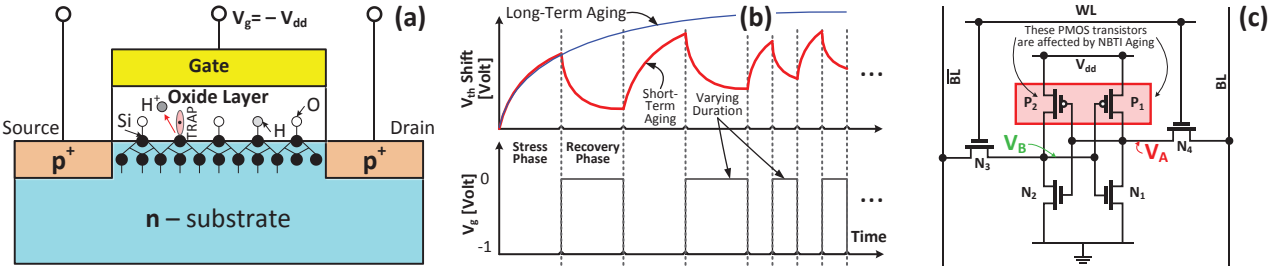


Figure 2-16: (a) PMOS transistor under NBTI aging with breaking Si-H bond at the Si-SiO_2 interface, (b) Stress and recovery phases for a PMOS transistor, and (c) Standard 6-T SRAM cell.

cycles, see Figure 2-16 (b). For ease of discussion, the duty cycle (Δ) is defined as the percentage of a cell's lifetime when the stored value is "one".

A memory composed of numerous 6T-SRAM cells is considered, where each cell is composed of two inverters to store a bit value (see Figure 2-16 (c)), and these inverters store complementary values at all times. The Word Line (WL) is enabled to write a value, while the Bit Line (BL) is used to carry data to be stored in the cell. The data is retained in the cell by turning off access transistors. To read data, WL is set high and the BL value is retrieved. In case a "zero" or "one" value is stored in an SRAM cell, one of its PMOS transistors will be under stress and the other in the recovery phase. Since the aging of an SRAM cell is determined by the worst-case aging of one of the two PMOS transistors, the overall lowest aging is achieved when both PMOS transistors are stressed by the same amount of time during the whole lifetime. That is, an SRAM cell contains "zero" value for 50% of its lifetime and "one" value for the remaining 50% of time. This corresponds to a duty cycle (Δ) of 50%. In short, SRAM aging depends upon the duty cycle of the transistors in an SRAM-cell. If the duty-cycle is balanced, the aging rate of SRAM cell will reduce.

Note that NBTI is not the only deteriorating mechanism active in SRAM cells. Hot Carrier Injection (HCI, which mainly degrades the NMOS transistor) is another aging mechanism which injects high energy (hot) carriers inside the gate oxide, causing interface traps and thus introduce threshold voltage shift [89], which can be presented by the following equation:

$$\Delta v_{th} = \psi_{HCI} s_a f e^{\frac{v_{dd} - v_{th}}{d_{ox} \psi_1}} t^{0.5} \tag{2-11}$$

In this equation, ψ_{HCI} and ψ_1 are aging rate dependent constants, s_a is the switching activity and d_{ox} is the oxide thickness. This shows that a higher switching activity will increase the HCI-induced aging rate and vice versa. On the contrary, higher switching activity will reduce NBTI. Studies like [90, 91] however, emphasize that NBTI has a greater impact and is the dominating factor in limiting the life of a circuit.

2.4 Related Work

2.4.1 Video Systems Software

This section presents state-of-the-art schemes for addressing different challenges of a video system at the software layer. These issues include parallelization, complexity reduction and power/resource budgeting.

2.4.1.1 Parallelization and Workload Balancing

As previously discussed, parallelization is a fundamental requisite of high complexity video applications, which must be exploited on many-core systems, possibly having hardware accelerators and custom logic/interfaces. The general classification of parallelization and workload mapping practices on many-core

systems is presented in the literature [92]. However, one of the objectives of this thesis is to exploit application-specific properties of the applications for workload mapping on a many-core system, and improving power efficiency of the video systems system. Numerous works have been reported to enable parallel computations of video applications. These works include parallel video coding/decoding [93, 94], tracking [95], image/face recognition [96, 97], non-negative matrix factorization [98] etc. However, these works generally do not consider resource management, hardware characteristics and workload balancing.

Workload Balancing: General load balancing of compute jobs among compute entities is discussed in [99, 72, 73, 100]. For power-efficiency, either the load of an application can be distributed on a given platform (load balancing [101, 102]), or, a platform can be synthesized for the given load (load driven synthesis [103, 104]) under throughput and/or power constraints. Most of the load balancing schemes consider homogeneous many-core systems, jobs with almost equal complexity and do not consider load variation at runtime [99, 1]. For example, [100] considers load balancing for distributed stream processing applications in wide-area environment, under dynamic resource utilization. In [105], load balancing between mirror multimedia servers is discussed for both centralized and distribution solutions. Ref. [99] deals with assigning each resource (core) with equal number of subtasks, and reach an equilibrium state if no more jobs can be migrated from one core to its physical, homogenous neighbors. However, the current architectural and physical challenges towards homogenous many-core systems are not considered. Smaller feature sizes result in physical variability of underlying transistors (also called process variation), which transforms into variable leakage power and maximum frequency achievable for the homogenous cores on the same die [106]. Thus, compute cores can have different characteristics, even though they form a homogenous many-core system. Research has focused on combining the distribution and balancing of load, and Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management (DPM) of the underlying cores [107]. Ref. [108] determines a single clock frequency for the entire chip for maximum efficiency, whereas [44, 3] independently determine the frequency of each core while distributing application load. Authors in [44] target minimizing the power consumption for a fixed deadline, while [3] tries to maximize the throughput of parallel running, multithreaded applications for a given chip's power budget.

Workload Balancing on Heterogeneous Nodes: Further, to increase the throughput-per-watt under modern system design challenges, heterogeneous multi-/many-core systems are becoming increasingly popular [2, 109]. Using architectural heterogeneity, it is now possible for the designer to schedule a processing job on a computing device which will increase the throughput-per-watt metric [110]. This way, maximum power-efficiency is achieved. For example, ARM's big.LITTLE architecture [111] incorporates high performance Cortex-A57 big cores with low power Cortex-A53 LITTLE cores, in order to achieve maximum throughput-per-watt by exploiting adaptive application mapping techniques. Thus, general load balancing methods are not applicable in heterogeneous paradigms, having cores/compute nodes with unequal compute capabilities. Parallelization and load balancing of H.264/AVC is carried out in [18], using heterogeneous CPU+GPU systems [19, 112], without considering the impact of power which is considerable when GPUs are used. In [2], authors target energy efficient workload allocation and voltage-frequency tuning of the underlying single-ISA computing nodes. The goal is to minimize energy/power of the system. However, their approach does not consider the case if the throughput of the application(s) is not met. In [113], authors propose to identify the program's and cores' characteristics and then appropriately match them for scheduling. Ref. [114] studies parallelized database on heterogeneous, single-ISA architectures. These proposed workload balancing approaches do not consider modern fabrication technological challenges like power budgeting, Dark Silicon etc. Moreover, it is not generally true that the complexity of each subtask is equal. For example, ME can have considerable different complexity for different blocks, depending upon the content properties and texture within the block [115, 67]. Further, the cache behavior of the application and the physical locality of the core (e.g., its distance from the external memory controller) also determines the complexity of a subtasks. All these challenges must be

addressed if an efficient workload mapping and balancing policy needs to be implemented. This is especially true for video systems under throughput constraints.

Parallelization of Video Systems: Multiple parallelization schemes for video systems are available in the literature. For example, a many-core based SIMD implementation of H.264/AVC is given in [116], but it does not consider workload mapping and balancing. Using partial frame-level parallelism, a 12-core system for parallel HEVC encoding is presented in [93]. Ref. [94] discusses an approach to parallelize H.264 on Cell multiprocessor. In [117], a hierarchical parallelization of H.264/AVC is presented for low cost cluster of cores, by combining multi-level parallelism. In [118], a parallel implementation on shared memory architectures for the particle filter is given. Ref. [119] uses a parallel implementation of the non-local means algorithm on a GPU for denoising 3D data. In [24], a hardware/software partitioning is targeted for a heterogeneous processor, for MPEG-2 encoder. Parallel video super-resolution methods are proposed in [120, 121]. In general, all these parallelization methods do not consider workload balancing on the many-core system, power reduction and addressing the throughput requirements. Also, these approaches might require access to multiple video frames (in the external memory) at a single time, increasing the latency of the system. Thus, they either increase the power consumption of the system by needlessly increasing the core frequency beyond requirement, or reduce the throughput and increase latency by burdening each cores with divergent workloads.

H.264/AVC parallelization and workload balancing is also discussed in the literature. Authors in [70] present a history-based approach to dynamically allocate the number of slices per frame to balance workload among the multiple cores. Here, each slice is mapped to a single compute core. A similar history-based scheme can be found in [71] where the skipped video frame blocks determine the slice boundaries for parallel encoding. A two-pass slice partitioning scheme for workload balancing of H.264/AVC is given in [122], where each frame is pre-processed, prior to being assigned into slices. However, for these approaches, no adaptation of workload and frequency of the cores (and thus, of power consumption) takes place.

2.4.1.2 Power-Efficient Video Processing Algorithms

A multitude of works in reducing the complexity of computationally heavy image/video applications also exist in the literature. Basically, by sacrificing a small amount of output quality (e.g., a reduced PSNR or accuracy of tracking, increased bit-rate), the workload of the application is curtailed to meet the throughput.

For HEVC encoding, numerous complexity reduction techniques exist in the literature [123, 124, 125, 126]. The work in [123] (basically inspired from [127, 128]) presents a gradient based fast intra mode decision for a given PU size, and results in about 20% time savings. In [124], authors have also presented a fast PU size selection algorithm for inter-frames (exploiting temporal correlations for frame compression, similar to open-loop for H.264/AVC presented in [129]). A divide and conquer strategy for selecting the best Intra angular prediction is given in [125]. First, 8 equally spaced modes (at a distance of 4 in both directions) are tested. Afterwards, 6 best modes with a distance of 2 are tested. In the end, 2 best modes are left which are tested with a distance of 1 to select the best mode. However, the number of modes selected for RD-cost determination is fixed and rather large. Similarly in [126], to reduce the total number of Intra prediction modes tested, an open-loop approach is utilized. Using the current pixels instead of reconstructed pixels, the total number of predictors is reduced from 35 to 9. These 9 modes are used for computing the Rate-Distortion (RD) cost. In [130], the CTU is downsampled and then texture complexity (via variance) is computed, leading to appropriate PU sizes. Ref. [131] exploits the high correlation of Intra predictions modes with the neighboring blocks for determining a highly probable Intra prediction mode for the current PU. An edge based Intra Prediction candidate selection scheme is given in [132] to reduce the total number of modes tested by 73% and a time reduction from ~8% to ~32%. The 4×4 pixels in each PU are treated for determining the dominant direction and a set of 9 Intra predictors is used for testing. However, the selection and truncation of Intra prediction modes is not adaptive.

Similarly, numerous works exist to reduce the complexity and energy consumption of the ME engine. Schemes that reduce the total number of operations in ME are also widely studied and employed [133, 66, 67]. For example, in [134], authors have proposed a scheme to reduce the off-chip memory accesses for ME and gain up to 56% memory access reduction. However, their scheme is tested for a very small search window size of 16, which cannot be used for large resolution sequences.

Other approaches for H.264/AVC (e.g. [135, 136]) may not be efficiently applicable to new video encoders like HEVC, due to its novel CTU structure and nature of its angular prediction modes. Moreover, these approaches usually do not jointly consider power efficiency and workload balancing, and do not exploit the speed-up achieved via parallel encoding on a many-core platform. Further, these approaches do not consider the underlying platform properties (i.e., do not exploit the opportunities provided by the hardware) and the new challenges introduced by the reduced feature sizes, while managing their workload.

In addition to video coding, there is a plethora of other video processing algorithms in diverse fields of applications, where complexity knobs are tuned at the cost of output quality. For example, see [137, 138]. Libraries like “Open-Source Computer Vision” (OpenCV [139]), and standards like OpenVX [140], provide numerous implementations of these video algorithms.

2.4.1.3 Energy Budgeting and Workload Offloading

A general overview of workload offloading methods can be found at [86, 141, 142]. The approach of [143] explores tradeoff between energy consumption of mobile device and data transmission. The history-based adaptive offloading approach in [144] aims at reducing the application processing time. However, no account for content- and application-specific properties in general offloading methods may be energy inefficient.

To process videos for a specific time interval, a constraint mobile video device can be provided with a fixed energy quota (in form of a battery). The device should spend this quota in a manner to increase the video quality while meeting other computational constraints (e.g., duration for which the encoder is expected to work).

Mostly, DVC and HDVC solutions used H.264/AVC encoders for Intra-frame encoding. However, there are works that access the feasibility of using HEVC encoders instead [145]. Prominent state-of-the-art techniques in low-power HDVC (like [4, 35, 146]) determine the number of video blocks to be processed at the constrained device (encoder) side in a raster scan order, use global motion models, send a low quality reference, or process the block completely at the encoder side based upon block’s motion intensity. The experimental analysis in Section 3.2.4 illustrate that objects in video sequences typically consist of blocks that do not occur on the raster scan order [147]. In addition, these methods usually do not consider the processing duration and/or energy budget associated with the encoder. Moreover, selection of the blocks with high motion intensity cannot cope with constant camera motion and panning. Therefore, an inexpert selection of blocks in the raster scan order, or, based on mere motion intensity may lead to inefficient energy distribution in HDVC. As a result, these state-of-the-art techniques may quickly exhaust the available energy quotas for the ME of non-important blocks which can be easily regenerated at the high-end server (also called decoder).

Summarizing, state-of-the-art approaches for HDVC and DVC may not be resource- and energy-efficient as they do not exploit:

- Video content properties (like texture, motion, etc.).
- The relationship between video content properties, and the ME effort and parity bits produced.
- Spatial and temporal correlation of blocks.

Therefore, the energy quota distribution and control scheme need to perform intelligent partitioning of ME

in HDVC by exploiting the video content properties while minimizing the overall energy.

2.4.1.4 Mitigating Dark Silicon at Software Level

The purpose of these approaches is to bind the maximum temperature or maximum power (TDP) consumed by the system at the software level. The above mentioned software level techniques (i.e., parallelization, complexity reduction, budgeting and offloading) implicitly address the Dark Silicon challenges. For realizing these techniques, mainly, two distinct approaches are employed for this purpose:

- Dynamic Thermal Management (DTM), which involves adjusting the voltage-frequency or power of the cores (DVFS [148]) and even severing the power to the compute nodes (via power gating, also called Dynamic Power Management, DPM).
- Tasks/Thread/Workload Migration, which involves migrating the workload from one core to another, in case the former core's temperature becomes critical [149].

The frame-based energy management technique for real-time systems [150] exploits workload variations and the interplay between DVFS and DPM, for a frame-based, real-time embedded application. It determines the optimal voltage-frequency setting and power levels of the devices to minimize the system's energy. In [151], a trail-and-error based centralized algorithm determines the appropriate DVFS settings for the many-core system, to enable maximum speed up under chip's power constraint. A thread mapping methodology under the constraints of Thermal Safe Power (TSP) is provided in [152]. Here, the authors argue that TDP is very conservative and power more than TDP can be provided to the chip if intelligent task-mapping decisions are made. The work in [153] aims at runtime PID-controller based mechanisms to efficiently utilize the TDP budget in order to maximize performance of micro-architecturally heterogeneous cores synthesized with different power and performance targets. However, [153] does not target power budgeting among multithreaded applications with thread-level workload variations. In [154], a two-level closed loop power control scheme is presented. Using voltage/frequency islands on a chip, the power distribution is adaptively divided to the working processing elements. However, fine-grained power distribution and configuration selection using this scheme are not possible, which are important for multithreaded applications with threads of highly varying workloads. Furthermore, the closed loop control usually responds slowly, causing performance issues for applications where the workload changes are abrupt. Single thread based power budgeting is discussed in [155], which is not applicable to multithreaded applications. PEPON [156] also presents a two-level power budgeting scheme to maximize performance within the allocated power cap. The scheme in [3] targets control-based chip level and application level power budgeting, while accounting for the critical threads of the application.

Most of these scheme do not consider assigning cores to the applications, and the varying workload of the applications at runtime, which might require readjustment of the resource allocation. Moreover, these schemes do not target power allocation to dependent-applications or subtasks of a single application, where the critical application or subtask will reduce the throughput of the system. These works also ignore the nature of the underlying applications and the opportunity it might provide for increased throughput-per-watt. Further, applications with throughput constraints must meet their deadlines (e.g., multimedia applications having soft deadlines and mission critical applications with hard deadlines), which is usually not addressed by these works, rather applications' speed-up is targeted. This also poses additional challenges if the system load (e.g., due to parallel running applications, delay in delivering Ethernet packets) may change at runtime.

In summary, most of these techniques do not exploit the opportunities provided by:

- Selecting appropriate operating modes (configuration of different variables) of the applications and Dark Silicon.

- Considering that the unadaptable application will utilize the cores powered-on at the nominal frequency/voltage setting.

Collectively, the operating modes of the application and the Dark Silicon provides opportunities of having multiple power modes [157], for instance:

- Powering-on more cores at low frequency settings to facilitate more applications or applications with high thread-level parallelism.
- Powering-on less cores at high frequency to facilitate high instruction-/data-level parallelism.
- Choosing appropriate operating modes of the applications to enable higher throughput at the same power budget.

Mostly, DTM only gathers system statistics. Since these techniques do not take the application-specific characteristics into consideration, therefore, they lack power efficiency in cases of abrupt workload variations and/or when multiple threads of different applications (especially with mixed workload characteristics) are competing for the power budget. Hence, fine-grained power distribution and configuration selection is not possible.

2.4.2 Video Systems Hardware

In this section, an introduction of the state-of-the-art approaches are provided, which targets design and implementation of video system hardware architecture to address multiple challenges.

2.4.2.1 Efficient Hardware Design and Architectures

Numerous state-of-the-art approaches exist for designing compute- and power-efficient video systems. For encoding HD videos, new methods and tools to administer the necessary data processing tasks and dependencies of video encoders are required. For example, authors in [158] discuss an H.264/AVC Intra encoder chip operating at 54MHz. However, it is only capable of handling a 720×480 4:2:0 video at 30 fps. Additionally, the authors use a parallel structure for computing the modes that increases silicon area overhead. In [6], a fast mode selection preprocessor based on spatial domain filtering is discussed. A four-stage pipeline for edge extraction increases the latency of their design and processing one video block requires 416 cycles at a maximum possible clock rate of 66MHz. The design in [159] presents a 1080p at 25 fps Intra encoder operating at 100MHz. It takes about 440 cycles to compute the Intra predictions. In [160], a 4K UHD (3840×2160) resolution at 60 fps Intra prediction architecture is proposed that replicates hardware for high throughput, and needs to execute at least at 310MHz to achieve 4K UHD while still using sub-optimal prediction selection methods. Ref. [161] presents a low-latency 1080p at 61 fps Intra encoder architecture operating at 150MHz, but it tests the predictions in parallel and takes about 300 cycles to encode one block. In [127], authors propose a fast method of selecting the best prediction, based upon the texture flow. Ref. [129] presents an open loop (OL) method to determine the most likely mode based upon the original image data rather than the reconstructed data. Similarly, a multitude of novel architectural designs for HEVC are also available. In [162], the authors proposed an HEVC Intra prediction HW for only 4×4 blocks. In addition to video coding, several other multimedia processing modules are realized using efficient architectures. For example, deblocking filters, AES, CRC [163, 164].

Motion Estimation: Many approaches to reduce the energy consumption of the video encoding process target the ME engine for optimization, because ME (or block matching) is the most time and energy consuming process of a video encoder. ME energy is reduced by reducing the supply voltage and then employing error-resiliency features in [165]. This results in an energy savings of up to 60% on 130-nm CMOS technology. But this approach results in extra control and noise-tolerance circuitry and degrades the output video quality as well.

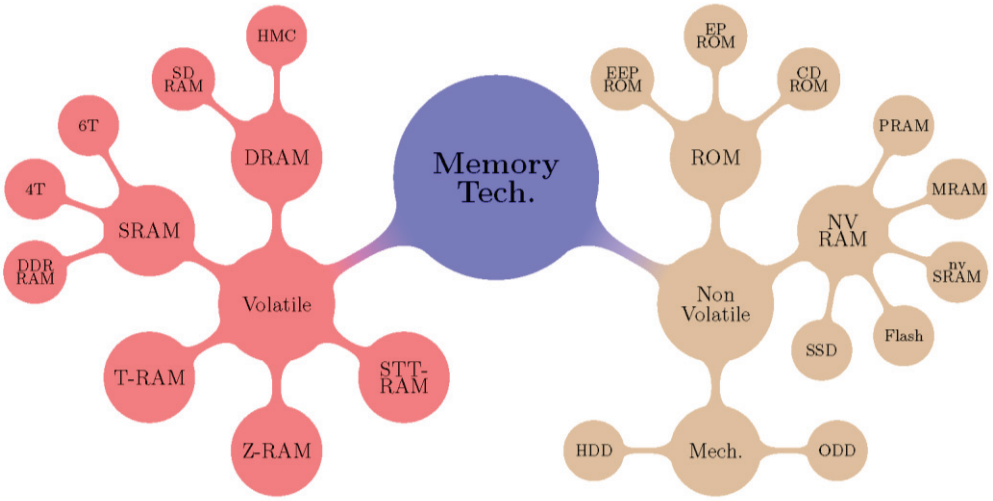


Figure 2-17: Different memory technologies

Ref. [166] reduces the search space for ME and thus avoids redundant memory accesses. However, an unnecessary brute-force full-search algorithm results in high computation effort. Furthermore, in addition to the approach's failure for sudden motion, their results are for CIF/QCIF videos which already require a small search space for ME. In [167], the on-chip memory is replaced with a cache. Further, the leakage energy reduction is not considered (which is dominant for sub-micron technology [168]). The work in [169] reduces the external memory accesses by frame-buffer compression, but requires additional computations. In [170, 171], different data reuse schemes for reducing the external memory accesses for video processing are categorized from levels A to D, with highest latency (smallest on-chip memory) to the lowest latency (largest on-chip memory). There are other extensions, like level C+ [61]. Reducing the total number of predictions (ME operations [133, 66]) also decreases the latency of execution but has little improvement for memory energy consumption and external memory access (see details in Section 2.2.1.2).

2.4.2.2 Memory Subsystem

The leakage energy of the video frames is of more importance in the sub-micron era, as it surpasses the dynamic power of the memory device [168]. Therefore, new memory technologies which address the issue of density and leakage power are evolving. A synopsis of some of the memory technologies is shown in Figure 2-17. Next-generation NVMs like MRAM [30, 172, 29] and Phase-change RAMs (PRAM) [173, 174] have shown promising results towards leakage power efficiency compared to SRAM or DRAM. Application designers are now considering exploiting these memories by analyzing their advantages and disadvantages. Table 2-2 summarizes the main differences between the NVM and VM technologies. NVMs provide high capacity and low leakage power but write latency and energy is considerably larger compared to that of SRAMs. However, their non-volatility can be sacrificed and NVMs like STT-RAM can be used as VMs.

In [175] and [176], a hybrid memory architecture is proposed comprising of PRAM and DRAM. A hybrid of ScratchPad and NVMs for on-chip memories is given in [177]. A work which utilizes hybrid memory for video decoding is given in [178], where frame-level decisions for storing H.264 frames on hybrid-memories are used. However, H.264/AVC encoder is ~10×-20× more complex than the

Table 2-2: Comparison between NVM and VM memory technologies ('H' denotes high; 'L' denotes Low; Blue color denotes ad-vantage; Red color denotes disadvantage)

Technology	Speed		Power			Density	NVM
	Read	Write	Dynamic		Leak.		
			Read	Write			
SRAM	HH	HH	L	L	HH	LL	No
DRAM	H	H	H	H	H	H	No
MRAM	H	L	L	HH	L	H	Yes
PRAM	L	LL	H	HH	L	HH	Yes

decoder [32] and presents a tougher challenge. Ref. [65] targets the HEVC application and limits the external memory access by storing the video samples (which are expected to be used later) in a hybrid combination of SRAM and STT-RAM. Other approaches utilizing MRAMs as replacements and augmentation of the traditional fast SRAMs are given in [179, 180].

Usually, most state-of-the-art hardware designs do not jointly reduce the power consumption of the system in conjunction with meeting the throughput demands. Overall system characteristics (e.g., data transfer from external to on-chip memory) are not considered.

2.4.2.3 Accelerator Allocation/Scheduling

In order to combine the advantages of both programmable and application-specific custom architectures, accelerators based many-core systems are becoming increasingly popular in the industry [181, 182]. Accelerators are usually high complexity parts of programs (called subtasks) implemented in custom hardware, and a programmable core can offload its tasks to these accelerators. For examples of accelerators, refer to Section 2.4.2.1. Accelerators naturally lend themselves to occupy the underutilized chip's area. In addition to increasing the Bright Silicon, accelerators are designed to quickly process the assigned tasks. Therefore, accelerators are fundamental to high complexity, deadline-conscious applications. Examples include video encoding and decoding [183] (also see Intel's Quick Sync Technology), software defined radios [184] etc.

For ease of discussion, we broadly classify accelerators into three categories, based upon their flexibility and access mechanisms. These categories are also shown in Figure 2-18.

- First are the in-core accelerators, which are embedded as a part of the programmable core's computation pipeline (e.g., Nios II custom instructions [185, 50], vector instructions [186], Application Specific Instruction-set Processors, ASIPs [187, 188]). However, note that these accelerators can only be accessed by the corresponding core, and they are a part of the execution stage of the computational pipeline. Therefore, these accelerators exhibit the least flexibility as these accelerators can only be accessed by their cores.
- The second category is clustered accelerators, where an accelerator can be accessed by only a specific

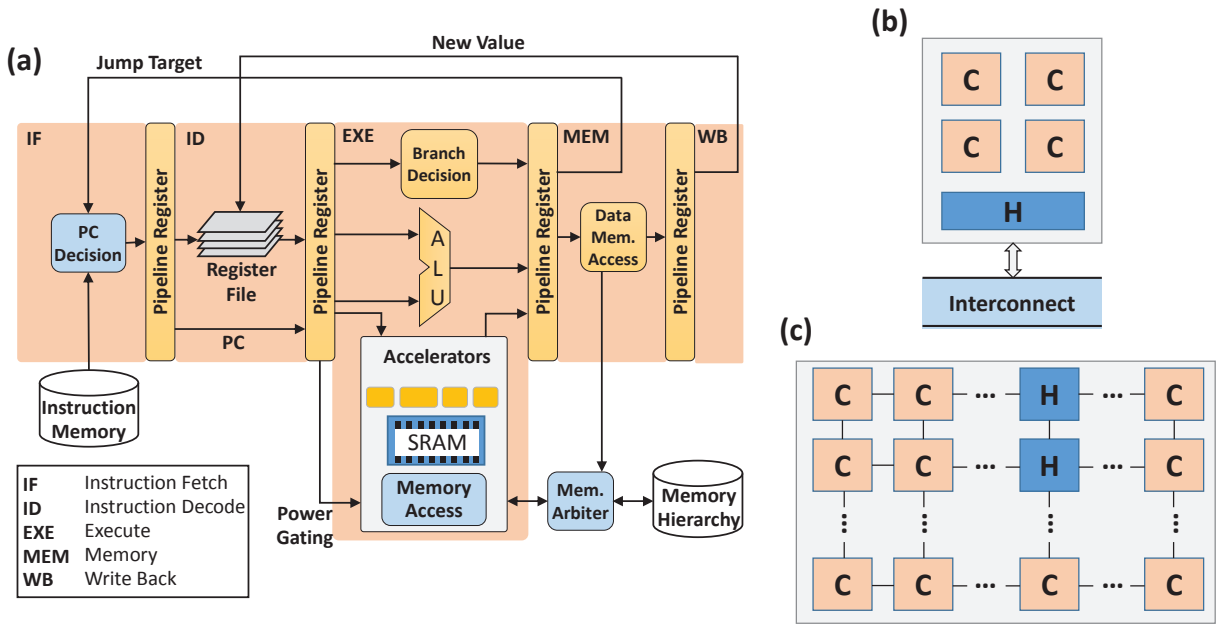


Figure 2-18: Accelerator locality, (a) In-core, (b) tightly coupled, (c) loosely or decoupled accelerators

set of cores and resides in vicinity of these cores (e.g., within a computation tile). Such accelerators are also called tightly-coupled accelerators [189, 190]. Approaches like [191, 192] are available to schedule the accelerator's sharing with the corresponding cores by offloading the soft-cores' subtasks, using past-predicts-future paradigms and dynamic programming. However, these approaches do not consider the complete power consumption of the system, and neither do they account for the deadlines of the running applications.

- The third and the most flexible category of accelerators can be accessed by all the cores (e.g., via a Network on Chip, NoC, PCIe) and are called decoupled accelerators or loosely-coupled accelerators. It is evident that the clustered and decoupled accelerators are the most versatile and offer maximum advantages. However, state-of-the-art scheduling schemes presented in the literature [193, 194, 195, 196, 197] for decoupled accelerators usually try to reduce the resources used, maximize the processing speed, or, reuse the accelerators' memory as cache or reconfigurable logic. No reference to the power consumption, frequency tuning of the cores and deadlines of the applications is made.

Since the shared accelerator can only be allotted to a single compute core at a given time, therefore, some of the applications running on these cores might miss their deadlines, or these applications might change their workload at runtime. Further, it is possible that the accelerator is not continuously utilized (i.e., accelerators are darkened) which defeats their purpose of providing power- and complexity-efficiency. In addition, it is also possible that in order to meet the deadlines, higher than required power is pumped to the cores. This will increase the power consumption of the system, and therefore, elevate the chip's temperature.

2.4.2.4 SRAM Aging Rate Reduction Methods

In general, state-of-the-art techniques for aging mitigation primarily target aging optimization for SRAM-based register files. However, these techniques do not target large-sized memories which have distinct access behavior and require different architectural support. The first category of work is based on the principle of bit rotations (i.e., moving LSB by one position) to improve duty cycle of registers [198, 199]. These techniques perform inefficient for registers with successive zeros and are only beneficial when the bits inside a registers are frequently modified, which is typically not the case for large-sized memories (see Figure 3-15). Moreover, implementing bit rotations requires barrel shifters at the read and write ports of the memory. The total number of multiplexers required to implement an n-bit barrel shifters is $n\log_2 n$, and this is in addition to the control logic which will configure the barrel shifters. Therefore, both area and power consumption overhead of such techniques is high.

Another category of work is based on bit flipping at every write to the memory [199, 200, 201]. The register value inversion techniques result in additional reads/writes and power. The recovery boosting technique [201] adds dedicated inverters in the SRAM cells to improve the recovery process. However, this incurs significant power over-head, which may be infeasible for large-sized video memories, for instance, targeting image buffers for high-definition (HD, 1920×1280 bytes) and 4K UHD (3840×2160 bytes) resolutions. Additionally, it requires an alteration to the SRAM 6T cell circuitry. In [88], a redundancy based SRAM mircoarchitecture is used for extending the SRAM lifetime. Similar to [201, 202, 203], this also requires architectural modification of the 6T SRAM cell. The work in [204] introduces algorithms for balancing the duty cycle of SRAM data caches by exploiting cache characteristics (i.e., tag bits). A similar scheme is presented in [205, 206]. These schemes depend upon the inherent properties of caches (like flushing, cache hits etc.) and are not directly applicable to general on-chip SRAM memories. Moreover, some of the mentioned balancing policies are designed for capturing the occurrence of a certain bit pattern, and thus perform inefficiently when considering different content properties and varying stress patterns. Further, many of the reported works for aging balancing require multiple read/write of the same data in the memory, rendering themselves to be power hungry.

Summarizing, state-of-the-art aging balancing approaches employ bit flipping or rotation at every bit level at every access time, and incur significant power and area overhead. These approaches do not explore the tradeoff between power consumption and aging balancing. Moreover, most of these approaches only provide elementary circuitry without exploring benefits of different aging balancing approaches and lack full architectural solution with power-aware aging control and adaptations.

2.4.2.5 Encountering Dark Silicon at Hardware Level

In this section, brief details about handling the Dark Silicon at the hardware layer of the video system will be provided. However, the above mentioned state-of-the-art approaches (for designing efficient hardware accelerators, scheduling the shared accelerator, power-efficient memories etc.) implicitly address the problem of Dark Silicon at the hardware layer.

At the hardware layer, different control knobs (e.g., for DVFS and DPM) are provided to throttle the chip's temperature within safe limits. Other approaches employ architectural heterogeneity to tradeoff performance and power. Via heterogeneity, the system supports runtime management of tasks by providing several degrees of freedom to the system designer. The different forms of heterogeneity can be classified as [207]:

- *Functional Heterogeneity*, where compute nodes exists with varying functional behaviors and architectural details. Examples are application-specific hardware accelerators, GPUs in conjunction with CPUs, super-scalar cores and RISC processors, reconfigurable architectures etc. Thus, using task migrations and using scheduling, Dark Silicon can be encountered.
- *Accelerator Heterogeneity*, same as discussed in Section 2.4.2.3, i.e., in-core, clustered and decoupled accelerators, providing different level of performance and flexibility of usage. Further, approximate accelerators [208, 209] with controllable amount of approximations can also be employed to increase the throughput-per-watt metric. This will not only increase the amount of Bright Silicon, but introduce higher performance.
- *Microarchitectural Heterogeneity*, whereby different cores on the same die have varying power and performance properties, but employ the same Instruction-Set Architecture (ISA). An example is ARM's big.LITTLE architecture [111]. For example, [210] presents a methodology to design multi-core systems while considering the Dark Silicon paradigm. The purpose is to maximize the Dark Silicon utilization. In [103], depending upon the characteristics of parallel running applications, Dark Silicon aware multi-processors are synthesized using a library of available core types. Special-purpose conservation cores (c-cores) are discussed in [211], whose goal is to reduce the energy consumption of the system, rather than the performance of the system. Device-level heterogeneous multi-cores and resource-management are exploited in [212] to speed-up the performance, as well as save energy.
- *On-chip Interconnect Heterogeneity*, where routers that connect the multiple cores of the chip are designed with heterogeneous architectures [213]. This provides diverse power and performance design points, which can be exploited by the system designer.
- *Process Heterogeneity*, where the non-ideal fabrication process results in core-to-core and chip-to-chip variations in the maximum achievable frequency and leakage power. This variation can be exploited to adaptively increase the speed-up of applications [214, 215] while meeting the TDP budgets of the chip.

2.5 Summary of Related Work

A plethora of approaches to tackle challenges imposed by video system software, hardware and new fabrication technologies are presented in state-of-the-art works. Summarizing, the state-of-the-art does not exploit the complete design space concerning both hardware-software co-design and co-optimization. This is

specifically important for multimedia systems, under Dark Silicon and reliability threats. For best throughput-per-watt ratio, the designer needs to consider the full system architecture, to fully exploit complexity-, power- and resource-savings, and reliability improvement potential for long term system deployment.

Usually, the Dark Silicon mitigation approaches proposed in state-of-the-art works do not consider the throughput constraints and they do not exploit application-specific properties. As previously discussed, multimedia systems have deadline constraints which require intelligent power budget distribution (i.e., frequency allocation) among the resources. Similarly, state-of-the-art does not consider the impact of deadlines, and resource- and power-budgeting for shared-accelerator based systems. This results in suboptimal performance of the system and also reduce the power-efficiency.

Similarly, for SRAM aging-rate reduction, state-of-the-art approaches employ fixed aging balancing techniques, with significant energy overhead. Therefore, these techniques are unable to explore the tradeoff between aging balancing and energy consumption. Moreover, due to additional power consumption, the state-of-the-art techniques might result in a higher temperature, which will increase the aging rate in a positive feedback cycle.

Further, exploring the application specific properties might result in high power-efficiency and high reliability, which is mostly ignored by the state-of-the-art.

Chapter 3 Video System Design

This chapter provides the overview of the proposed video system. Details are given about the architectural aspects, and the complexity and power control knobs of the system. Via analyzing these knobs, motivational analysis is carried out which forms the foundation of the technical approaches proposed in this thesis (in order to address the challenges outlined in Chapter 2).

3.1 System Overview

The overview diagram of our proposed video processing system is shown in Figure 3-1. This diagram categorizes the major components of the system into software and hardware layers. The software layer executes the algorithms and the hardware layer provides support to run these algorithms. Different multithreaded video applications are concurrently executed on the system, and the system generates video outputs and runtime statistics.

The hardware layer contains a many-core system, with in-core accelerators, and also coupled shared hardware accelerators. The shared accelerators is fed by a gated clock. It also provides the video I/O and communication infrastructure among the cores and the accelerators. The hardware layer also contains custom

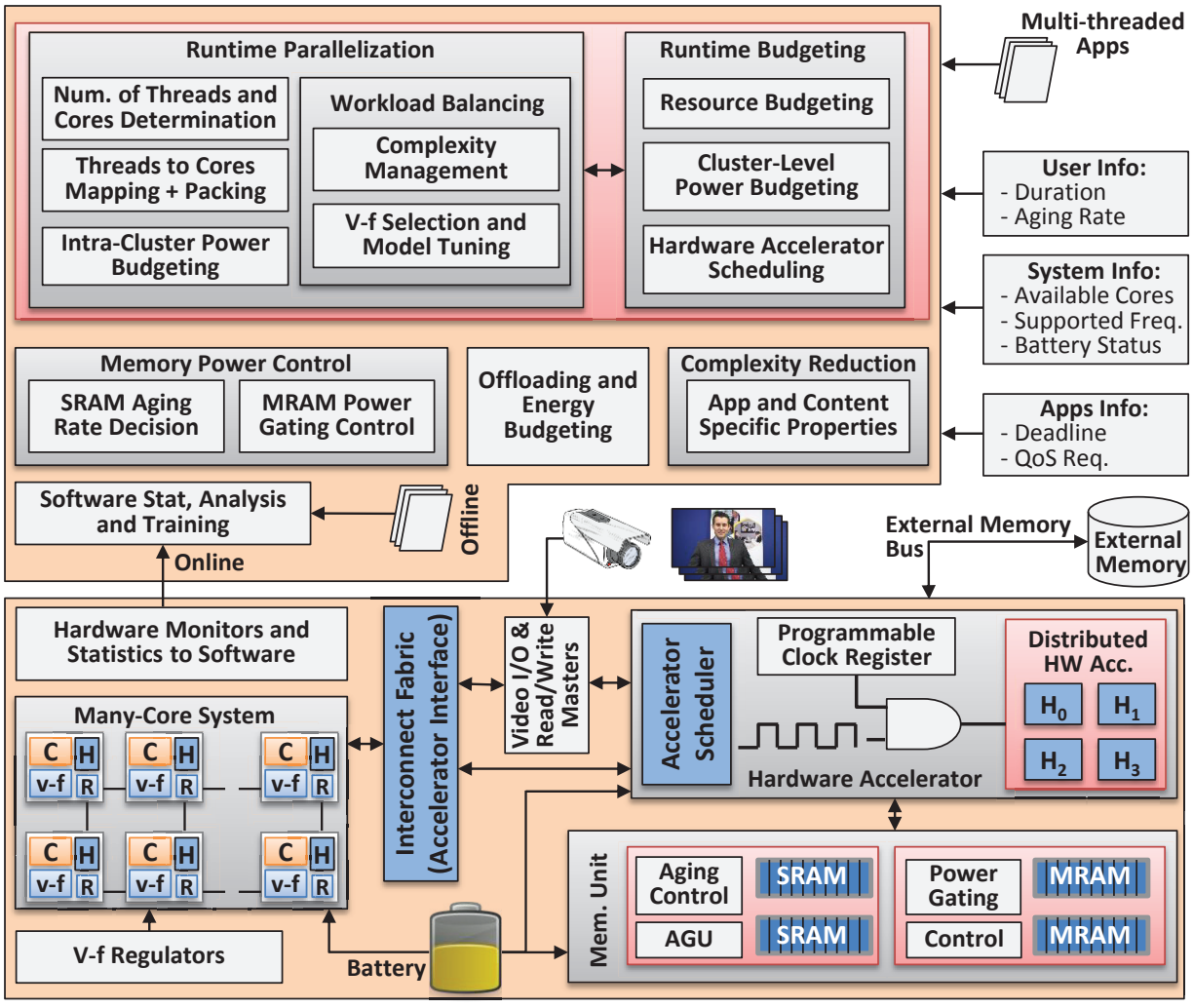


Figure 3-1: Software and hardware layers of the proposed multimedia system.

NVM (here MRAM is used) and SRAM. NVM power is controlled via power gating the memory, and SRAM aging rate is controlled adjusting the duty-cycle of each SRAM cell.

The software layer is responsible for algorithms pertaining to runtime resource- and power-budgeting, parallelization and workload balancing. It also controls the power of the system by

- Determining the most appropriate voltage-frequency levels of the underlying hardware.
- Offloading subtasks from the cores to the loosely coupled hardware accelerators and to other devices.
- Gating the clock of loosely coupled accelerators.
- Selectively turning ON sectors of the MRAM.
- Reducing the complexity (and hence the power consumption) of the application.

The software layer also intakes the user information required to run the system. For example, this information can contain the total duration for which the system must run, the number of applications that needs to be processed in parallel. The software layer also considers the underlying hardware properties for workload balancing and power management. In addition, it exploits the applications' and contents' properties to maximize the throughput-per-watt metric.

Details about design- and runtime features provided by this system will now be provided.

3.1.1 Design Time Feature Support

As discussed above, the architectural design features of the proposed video system include a many-core system, associated hardware accelerators and a hybrid memory subsystem.

The many-core system provides parallelization support, and allows multiple, multithreaded applications to execute their workloads on the cores. The cores can be homogenous or heterogeneous. A per-core DVFS to appropriately scale the voltage-frequency levels is available. Also considered in [156, 3, 216, 217, 218], the per-core DVFS is now commercially available (e.g. Opteron 12-core) and it is indispensable for fine-grain power management. Further, software libraries like "libdvfs" [219] exposes the control of cores' frequencies (or "governors" of Linux) to the programmer.

Moreover, each core has an in-core, application specific, hardware accelerator (see Figure 2-18 (a)). These in-core hardware accelerators are now available in commercial devices (e.g., Custom Instruction in Nios II processor [185], vector-processing units in Intel CPUs like SSE-SSE4 and AVX, AMD's 3DNow!). A programmer can write instructions for using the in-core hardware accelerator. Moreover, each core is connected to a global communication network using an interconnect fabric.

The subtasks from the programmable cores can be offloaded to the loosely coupled hardware accelerators, in order to increase the throughput-per-watt metric. It is assumed that the software version of the hardware accelerator is also available to the core, and the core decides whether to offload its tasks to the hardware accelerator, or do the same task via its own in software. The loosely coupled hardware accelerators communicate with the many-core system using an interconnect fabric. The accelerator scheduler receives offloading request from the programmable cores and assigns the accelerator to process subtasks of a core, one at a time, in a round-robbing fashion. Further, the accelerators are distributed hardware components, and these components are fed with gated clocks, i.e., when a particular accelerator (or part of the accelerator) is not used, its clock can be gated to save dynamic power consumption. Clock gating is available in industrial products like Open Multimedia Applications Platform (OMAP3) processor from Texas Instruments [220], Altera FPGAs etc.

The hybrid memory unit includes a high-density MRAM NVM in conjunction with a SRAM. A programmer

can use both MRAM and SRAM as a scratchpad memory. The MRAM is sectored memory and the sectors are normally OFF, i.e., under normal operation, the MRAM sectors do not consume any leakage or dynamic energy. A particular sector of the MRAM is powered ON only when it is required to read/write the data to that address. Hence, only at that instant, a little leakage power and some dynamic power is consumed by the particular MRAM sector tuned ON by demand. The SRAM memory's aging rate is controlled by using aging reduction circuits and by intelligent Address Generating Units (AGUs). The power consumed by these circuits is content dependent, and can also be controlled by the programmer.

In addition, the software/application layer must be able to exploit the different aspect of hardware layer (e.g., the maximum frequency of the cores). The software layer (application or OS, or any other middleware) should be able to adjust the voltage-frequency levels of the cores on demand of the application (see above, the discussion about "libdvfs"). Furthermore, the applications should be able to provide an ample number of independent subtasks such the proposed parallelization approach can pack these subtasks and determine the number of cores to utilize at runtime, depending upon the system events or requests. These applications can also be malleable applications [221, 222]. Note that APIs for implementing parallelism, like OpenMP [223], support this behavior. However, the condition of being malleable is not necessary, because an application can have a considerably large number of threads which can be packed and dispatched to individual cores. Examples of such applications are mobile games, where games are designed with tens to hundreds of threads. Ideally, the application also needs to have a set of operating points, whereby the complexity of the application can be traded for a drop in the output video quality (see discussion about reducing number of predictions in Section 2.2.1). Moreover, the application needs to know about the characteristics of the shared hardware accelerator (e.g., the number of cycles consumed per subtask).

The above discussion shows that to save power, different approaches can be utilized. Basically, by setting some configuration knobs, a tolerable output quality degradation can be sacrificed for gains in power savings. In summary, the following power configuration knobs are provided by the video system architecture.

- Per-core voltage-frequency knob (or per-core DVFS), to adjust the voltage-frequency, and thus the power consumption and the time consumed in processing a particular subtask of the application.
- Individual clock-gating of accelerator modules, to cut off the clock (and hence the dynamic power) to the module which is not currently utilized.
- Shared hardware accelerator scheduling, by which cores can offload their subtasks to the accelerator and can be turned OFF (or cores can run at a lower frequency to process their workloads), and hence, save system power.
- Tunable application parameters, which can result in a lower complexity and contribute to power savings of the system.

3.1.2 Runtime Features and System Dynamics

During runtime, the video processing system needs to do different kinds of application- and content-dependent resource budgeting, which includes distributing number of cores and TDP among multiple, multithreaded tasks/applications. The loosely coupled accelerators are scheduled to the processing cores and their clocks are appropriately gated. Moreover, the voltage-frequency levels should be correctly tuned, depending upon the throughput constraints and the shared accelerator. Additionally, if the hardware platform is unable to meet user constraints, a DVC or HDVC encoder is used. In this case, offloading approaches needs to be engaged with appropriate percentage of subtasks being offloaded to the decoder. In case the video system is used as a DVC or HDVC decoder, it must assign appropriate number of cores and power budgets to the competing encoding workloads. Furthermore, the hybrid memory subsystem needs to be properly accessed and the aging rate of SRAM is controlled. We will now discuss these aspects in detail.

The resource budgeting approach determines the number of compute cores that are allocated to a particular application. This resource budgeting depends upon the workload and the throughput requirements of an application. However, the resource budgeting must be fair to all applications and should not allocate too much resources to a particular application, which will starve the other parallel running applications and reduce the overall system performance. Moreover, resource budgeting needs to be adaptive, and should increase/reduce the number of cores allocated to an application (called a cluster) at runtime, taking into account all the parallel running applications and the workload variations of all applications. If possible, the threads of a single application can be mapped to a single processing core which can process the workloads in a time-multiplexed manner, thereby reducing the number of cores utilized in processing the workload. Further, the resource budgeting scheme should also allocate the loosely coupled hardware accelerator to process a part of the workload of these applications.

The power budgeting scheme distributes TDP at runtime among the applications. The TDP distribution not only considers the size of the cluster, but also the power consumption and the throughput generation history of a particular application. Specifically, an application with high power requirement in the past will require more power to be allocated to that particular application. Further, once the TDP budget is distributed among the applications, the applications also need to distribute this budget among their parallel computing cores (Intra-cluster power distribution). To balance the workload of each thread (and hence maximally utilize the hardware for maximizing the throughput), power is transferred among the threads of the same application as well. This power actually corresponds to a particular voltage-frequency level, and a higher power allocated to a particular core means that the frequency of the core can be set to a higher value (i.e., the core will take less amount of time to process the workload).

The power distribution (or frequency allocation) is workload dependent, which might vary at runtime, and may be different for threads of the same application. Therefore, it is required to derive the relationship between power/frequency required to process the workload corresponding to the application parameters (which control the complexity and output quality). Specifically, this relationship can be derived offline using regression analysis. However, such a relationship might not be accurate, and cannot handle the workload variations. Further, this relationship will only be applicable to a particular core on a particular system, i.e., the portability of the application will be an added issue. Parallel system workloads (like OS, parallel running applications on the same core) might also render this relationship inaccurate. Therefore, the proposed video system tunes this relationship online by getting statistics of the software and hardware at runtime. This also helps in balancing the workload, as a more accurate relationship can result in high accuracy power and resource distribution among the competing applications.

Runtime power control is actuated by using the hardware accelerator's power gating signals, which are control by a register written by the programmer. These signals depend upon the architectural as well as the code currently executed on the architecture, thus requiring application and architectural knowledge. Further, the sectors of MRAM that are used for read or write access are turned ON, while the others are kept OFF. This ON/OFF decision is also code dependent and the control reduces the memory wakeup latency, by tuning its predictions of which memory sector will be turned ON next.

The computational offloading mechanism analyzes the video content properties and determines the regions of the video frame (location of blocks within the video frame) that should be processed at the decoder side, in order to achieve high video quality at the expense of little energy/power consumption at the encoder side. Moreover, the SRAM aging controller also considers the content properties that are written to the SRAM, and adjusts the aging rate of the SRAM cells.

Regarding the discussion above, the following information passes between the layers. From software to

hardware, the number of parallel threads, voltage-frequency levels of the cores, hardware accelerator assignment and clock enable/disable signals are passed. On the other hand, hardware passes the time consumption, hardware accelerator information and many-core system's attributes (e.g., number of cores, minimum/maximum voltage-frequency settings) information to the software layer.

3.2 Application Analysis

In this section, analysis of applications and impact of software and hardware layers on the execution of these applications is carried out. This analysis will be later leveraged by the approaches proposed in this thesis.

3.2.1 Video Application Parallelization

Video processing applications usually consume a lot of system resources and power, in order to meet their throughput demands. An example is shown in Figure 3-2, where the timing complexity and energy consumption per-frame (with different resolutions) for encoding a video with HEVC is shown. These experiments are performed for an x86 core running at 2.16GHz. For instance, encoding one HD frame (1920×1080) consumes ~8.2 seconds and 119.02 Joules. Note that increasing the core frequency to support high frame rates under real-time constraints is not scalable and viable due to power density issues (Dark Silicon).

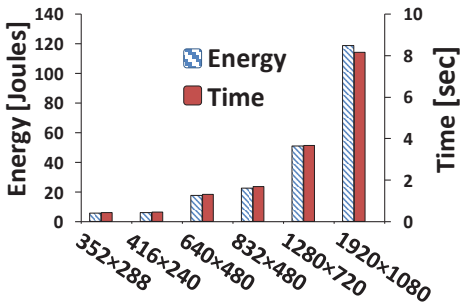


Figure 3-2: Energy and time consumption for HEVC Intra-encoding on x86 core

The total time consumption at different clock frequencies (f) and the size of the frame in blocks (n_{frm}) is shown in the color-coded plot of Figure 3-3 (a). Note that increasing f reduces the time consumption, and increasing n_{frm} increases time consumption of the core. Hence, for processing n_{frm}, a frequency of the core can be determined which will not consume more than a predefined (constant) amount of processing cycles. That is, the timing constraints can be satisfied by appropriately selecting the frequency, given n_{frm}. In fact, the surface plot is drawn by using a bilinear interpolation of the 4 neighboring (f, n_{frm}) points. The color-coded plot in Figure 3-3 (b) shows the dynamic power consumption of a single core in terms of n_{frm} and f for HEVC encoding.

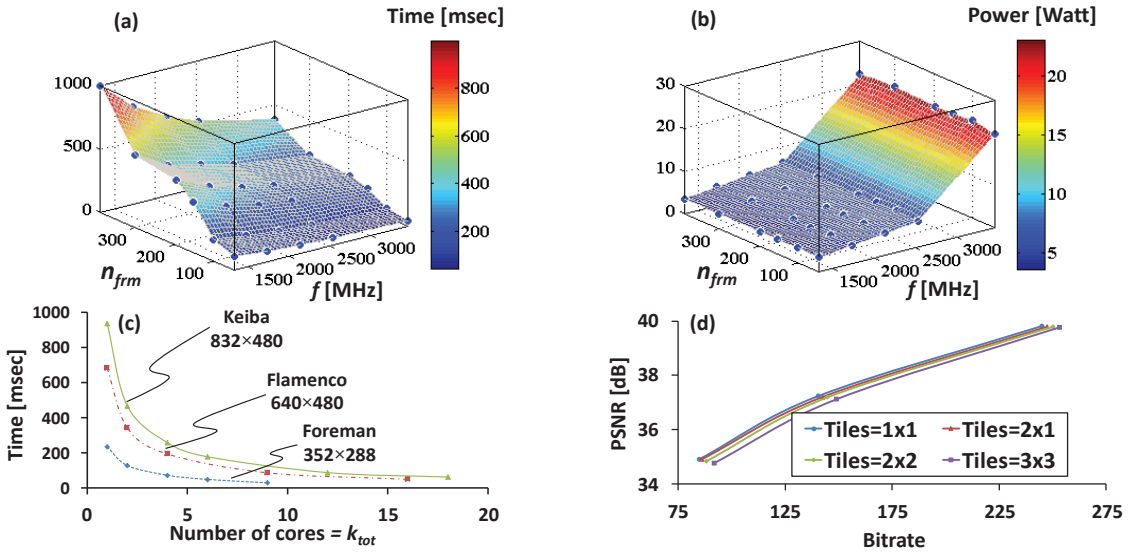


Figure 3-3: (a) Average time (msec) at different frequencies, (b) power consumption for different frequencies and frame sizes, (c) at f=2.16 GHz, average time (msec) per frame with varying number of cores and (d) PSNR Vs. Bitrate plot by using "Foreman" video sequence (352×288) for HEVC encoding

Increasing f results in high dynamic power consumption and vice versa. Power is independent of n_{frm} as the basic power formula only considers the voltage and working frequency of the core. Therefore, it is possible to select an appropriate clock frequency of the core that will limit the total power consumption below a certain threshold. In other words, the power constraint of the core requires an appropriate selection of the core's frequency. Therefore, using the allocated power to a core, its frequency could be determined.

To enable real-time encoding/decoding, the HEVC standard provides inherent support for parallelization in form of slices, tiles and Wavefront Parallel Processing (WPP) [93]. By dividing the video frame into parts such that, each part can be processed independently of other parts by breaking the sequential coding dependencies across the boundaries to enable parallelism, a potential video quality loss may occur [93]. To efficiently utilize the hardware resources and to reduce power consumption in a many-core processor, there is a need to (1) ascertain the parallel processing workload; and (2) reduce the number of active cores while fulfilling the throughput constraints. Figure 3-3 (c) denotes the total time consumed for processing a single frame for changing the number of processing cores (k_{tot}) at f=2.16 GHz. We notice that increasing k_{tot} reduces the time consumption by a second degree relation. Thus, appropriate k_{tot} can be selected that can encode the video frame within the given amount of time, i.e., for satisfying the time constraint. The output video quality for different configurations of tiles (and also different parallelism) is shown in Figure 3-3 (d). Since tiles break coding dependencies, therefore, increasing number of tiles also result in video quality loss and a single tile per frame results in the best video quality. Since parallelization is indispensable, therefore, a $u \times u$ tile structure (u columns and u rows) must be used with u^2 tiles per frame, for the best video quality. Therefore, the total number of tiles per frame must be minimized such that it fulfills the workload of HEVC encoding, not only to increase the video quality, but to reduce the power consumption of the system as well. If a perfect $u \times u$ tile structure is not possible, then effort must be made to make the total tile columns and row as similar as possible.

Usually, in video applications, the collocated tiles (same tiles of consecutive video frames, see Figure 2-10 (b)) exhibit high correlation. For HEVC, correlation of complexity and the output compressed bytes between collocated tiles is given in Figure 2-10 (b). This correlation is plotted as a histogram of percentage difference in time (Figure 3-4 (a)) and total bytes per tile (Figure 3-4 (b)). A high correlation is shown by larger crowding around zero. Therefore, time complexity and output quality (bit-rate) of the current tile can be estimated from previous collocated tile(s), and this can be used to estimate the workload of the current tile.

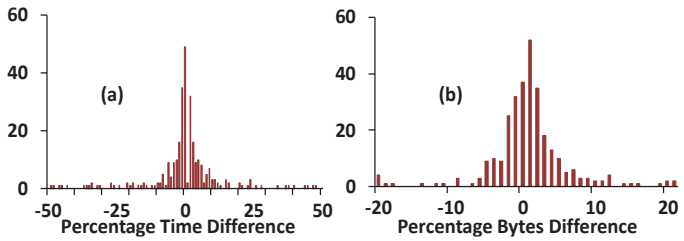


Figure 3-4: Percentage difference histogram of (a) time and (b) bytes for collocated tile number 0 of "Foreman" sequence (352×288) for 300 frames

3.2.2 Workload variations

The workload of a thread varies at runtime, whether it is the only thread of the application or one of the many threads of a multithreaded application. Figure 3-5 plots the time complexity of each tile for NLMF of the sequence "Foreman" (352×288). A tile structure of $u_w \times u_h = 2 \times 2$ (2 tile columns and 2 tile rows per frame) is specified for this experiment. As seen, the complexity of a tile is not only different than the rest, it also varies at runtime. This

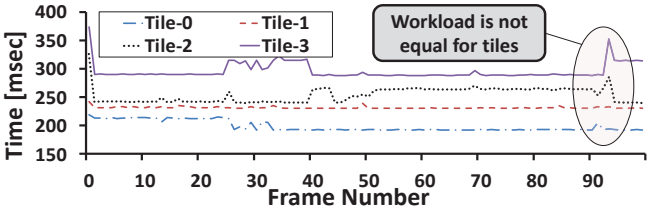


Figure 3-5: Time consumption of each tile (2×2=4 tiles per frame) of Non-Local Means Filter for "Foreman" sequence (352×288)

complexity is a direct measure of the workload. Additionally, if timing constraints are enforced, the power consumption of the cores will also vary. Therefore, it becomes imperative to control the workload of each compute core individually. This adaptation should depend upon the throughput requirement, as well as the hardware resources of the underlying many-core system.

This workload variation can be due to multiple factors. Some of these factors are described below:

- Some threads are allocated more data to process than the rest. The non-uniform data allocation can happen due to multiple reasons. Like, it may not be possible to get equal sized tiles by having odd number of blocks within a row (or column) of a video frame, e.g., 11 blocks per row of the video frame divided among 4 tiles.
- In many video applications (particularly video coding), the complexity of processing a block is also content dependent. For example, more time will be spent by ME for a tile with high motion blocks compared to a tile containing low motion blocks.
- Some video tiles may require a high quality than the rest. This is possible because Region of Interest (ROI) based processing requires more calculations while processing the ROI within a video frame. Therefore, a thread that processes the ROI may have more complexity than the rest. An example could be detection of eyes within the image, whereby the face can be the ROI. Hence, face is detected first, and then eye localization algorithm is performed on that region only. Similar examples exist for video coding, where the ROI includes faces, moving objects like cars etc. For ROI regions, compression is kept comparatively lower for a better visual effect, whereas the background regions (non-ROI) are aggressively compressed.
- Parallel system workloads (due to OS, or parallel running applications) may render a thread of an application to run slower than the others.
- Heterogeneity among the compute cores can exist which can in turn run the core slower/faster. For example, some cores might have larger caches than the rest, some cores are designed to run at a higher frequency (or some cores can run at higher frequencies due to process variations [224]).

The fundamental takeaway from the discussion above is that workload variation causes unequal time consumption of threads, which reduces the throughput of the application due to load imbalance. Hence, to achieve maximum benefit from the underlying hardware, it is necessary to balance the workload of parallel computing jobs at runtime.

3.2.3 HEVC Complexity Analysis

As previously discussed, HEVC is one of the next generation important application pertaining to video coding, and bound to take over its predecessor and current industry-standard, H.264/AVC. By dividing the CTU iteratively into respective CUs, PUs and TUs, HEVC tries to maximally exploit the redundancies within an image. This iterative and recursive behavior incurs significant complexity overhead, even for Intra-only encoders, because the RDO decision has to recursively check each possible PU and Intra mode combination (see Figure 2-6). This enormous decision significantly amplifies the computational complexity compared to H.264. Our experiments in Figure 3-6 show that the computational complexity of Intra-only HEVC has increased by a factor of ~1.4× for a compression efficiency increase of around 35% as compared to Intra-only H.264. A similar analysis can be

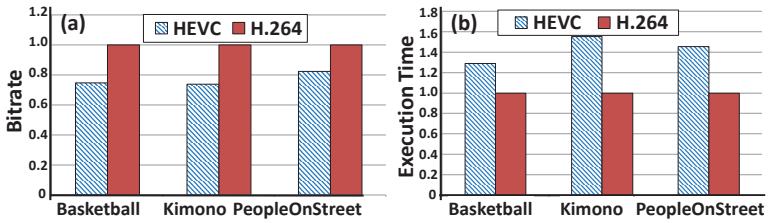


Figure 3-6: Comparing H.264-Intra to HEVC-Intra for relative (a) bit-rate and (b) execution time for different video sequences

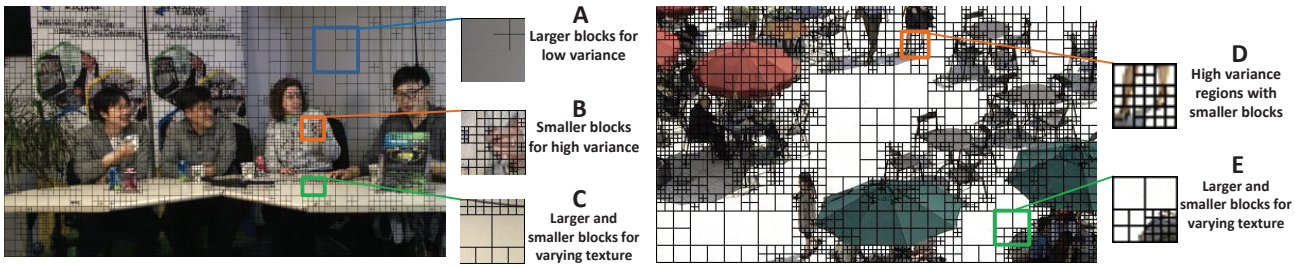


Figure 3-7: PU borders on the 4th frame of "FourPeople" and 31st frame of "BQSquare" video sequence

found in [225]. This illustrates a significant challenge towards fast HEVC encoders. Therefore, it is vital to develop fast algorithms to decrease the computational complexity of HEVC intra encoders, and to realize real-world applications.

3.2.3.1 Texture and PU Size Interdependence

In Figure 3-7, the borders of PUs (generated via RDO decision) are plotted on top of the frame of the "FourPeople" and "BQSquare" video sequence. The PU sizes and their corresponding locations illustrate that PU size decision is based upon the texture (or variance) of the video frame content. Note that image regions with high variance and texture details are usually encompassed via smaller PUs by the RDO decision. For example, in region A of Figure 3-7, a video frame region with low texture is encoded using larger PU sizes; whereas in regions B and D, a highly-detailed texture is encoded using small and dense PU partitioning. An interesting case is presented in region C where it is noticed that the wire (at the top of the region) is encoded using small-sized PUs and the uniform areas of the block are encoded using large-sized PUs. Similarly region E denotes small and large block sizes due to varying texture within the region. This analysis illustrates that a complexity reduction approach can be derived by estimating the sizes of PU using video content, instead of doing a high complexity RDO decisions. The PU size generation (involving early PU size estimation, i.e., before the RDO) needs to account for the video texture properties (i.e., variance) of frame regions in order to curtail the RDO search-space for fast mode evaluations. If such a map is available, then users can bypass the RDO process for improbable PU sizes and hence, reduce the complexity.

3.2.3.2 Edge Gradients and Intra Angular Modes

In Figure 3-8, a color-coded Intra angular direction map is overlaid on top of the frame (for reference, see Figure 2-6 (b)). This color map is shown in block D of the figure, with each Intra prediction octant (each octant has 8 angular directions) having an associated color. As seen, the angular intra mode is highly dependent upon the gradient direction (blocks A, B and C) and usually the texture flow angle (perpendicular to the gradient) is also selected as the Intra prediction. Therefore, we observe that the Intra angular direction depends upon the gradient [115, 123] of the PU. The gradient direction is perpendicular to one of the 4 octants of Intra angular direction. With high probability, the Intra prediction mode lies in that octant. We can exploit this knowledge to reduce the number of Intra angular modes tested for HEVC and achieving complexity savings. Depending upon the gradient of the video frame block, a set of highly probable Intra-prediction modes can be tested, excluding the unlikely modes. A similar case can be employed for Inter-encoding, whereby the complexity of ME can be reduced by observing the video properties.

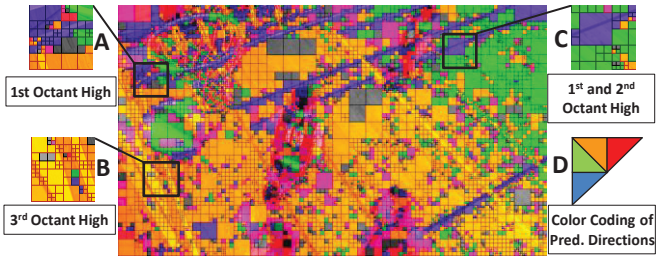


Figure 3-8: Color-coded intra angular directions per PU on 2nd frame of "BasketBallDrill" with CTU of 64×64

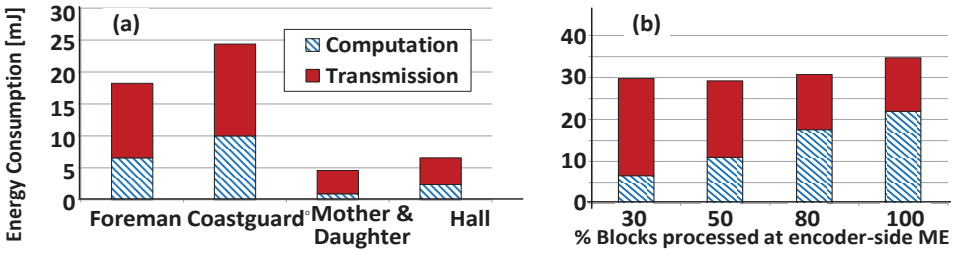


Figure 3-9: Analyzing the energy distribution for (a) different video sequences and (b) different number of MBs processed for ME

3.2.4 Computation Offloading

A resource constrained encoder can offload its job to a high-end server, using the concepts of computational offloading. However, offloading itself consumes transmission power and may increase the power consumed if no offloading is performed. Figure 3-9 shows varying distribution of computation and transmission energy for different test video sequences with diverse texture and motion properties. Note that for a given number of blocks for which ME has to be processed at the HDVC encoder side, there is an increase in the ME energy due to extensive ME search for fast moving blocks. Moreover, since for the remaining blocks decoder may not get accurate matches, it results in increased transmission energy of encoder due to more parity bits. Such an increase in the computation and transmission energy can be seen in the "Coastguard" sequence (Figure 3-9 (a)), which contains river water with ripples and moving boats that are hard to predict. Due to low texture/motion, "Mother and Daughter" and "Hall" sequences have reduced transmission and computation energy compared to the other two sequences with same PSNR = 36dB. Thus, more energy quota should be allocated for encoder side ME in case of video scenes with high texture and motion. Higher energy savings for computation and transmission at the encoder-side can be obtained for the low-texture and low-motion video scenes, as the decoder – with a high probability – generates good quality reconstructed video frames. The key is to leverage the video content properties during the energy quota distribution in order to balance the ME computation energy and the parity bit transmission energy at the encoder side, such that the video quality achieved at the decoder side is high.

3.2.4.1 Video Content Implications

Even within a video scene, due to their diverse texture and motion properties, it is important to study that which blocks have the highest impact on the computation and transmission energy. For this, we have performed experiments with different number of blocks processed for encoder-side ME. More encoder-side ME leads to high ME computation energy, but reduced transmission energy (see Figure 3-9 (b)). However, for a given number of blocks to be processed for ME, the energy consumption highly depends upon which blocks are selected for encoder-side ME. State-of-the-art techniques in HDVC (like [4]) select blocks in the raster select for ME processing. Since blocks of ROI typically do not lie on the raster scan order (see "Foreman" and "football players" in Figure 3-10 (a)), such techniques may lead to high transmission energy in case of few blocks processed for encoder-side ME; see Figure 3-9 (b). Moreover, ME of background blocks (low-texture/motion, static blocks; see Figure 3-10 (a)) may not provide effective reduction in the parity bits, thus leading to higher transmission energy. Since such blocks can be easily regenerated at the decoder side, there is no need to waste energy for encoder-side ME of such blocks. Furthermore, ME of blocks pertaining to background regions may quickly exhaust the available energy quotas. Therefore, it is beneficial to spend encoder-side energy in the ME of the complex blocks of the ROI, because this will help the decoder in estimation the true motion of complex blocks while performing frame interpolation. This shows that computation and transmission energy in HDVC highly depends upon the block types selected for the encoder-side ME. Higher overall energy savings and better video quality can be achieved if the blocks of objects in ROI are selected for encoder-side ME processing. Such knowledge needs to be incorporated during the energy budgeting of different modules. The key challenges are

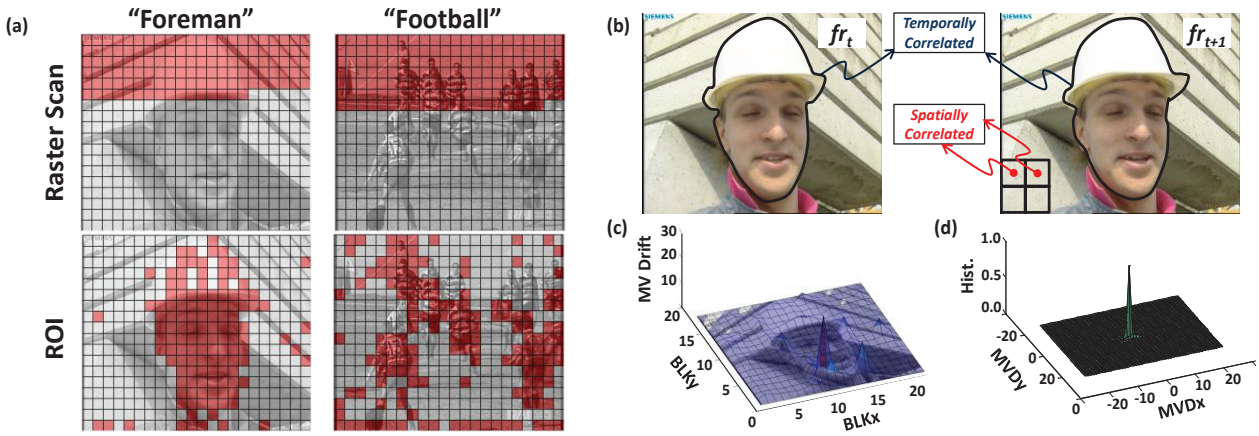


Figure 3-10: (a) Selected blocks (BLKs) at encoder for ME, in raster-scan and ROI order, (b) Spatial and temporal correlation among blocks of two frames (*fr*), (c) Motion Vector Drift (MVD) and (d) Occurrence frequency of MVD for the "Foreman"

low-overhead ROI identification, ROI-based block selection for different video scenes, and adaptive energy budgeting considering this knowledge.

Figure 3-10 (b-d) demonstrates that there is extensive spatial and temporal correlation between the MBs of the same frame and neighboring frames, respectively. Typically blocks of objects with low variance have high spatial correlation [147]. Similarly slow-moving blocks have high temporal correlation. This correlation can be used to efficiently predict the important blocks for which encoder performs the ME to reduce the prediction error at the decoder-side. However, the highly-correlated blocks are left for the decoder, as decoder has a high probability to find a good approximation for such spatially and temporally correlated MBs, thus resulting in less number of parity bits.

Figure 3-10 (c-d) shows Motion Vector Drift (MVD, motion vector difference of spatial neighbors) and MVD between the current and collocated blocks (i.e., temporal neighbors), where:

$$mvd = \left(\sum_{i=-1}^{1} \sum_{j=-1}^{1} \left\| \boldsymbol{mv}(x, y) - \boldsymbol{mv}(x+i, y+j) \right\| \right) \times SAD \qquad (3\text{-}1)$$

Here, *mvd* denotes MVD, and it equals the l^2-norm of the difference between the MV ***mv*** of the current block and its eight connected neighbors, weighted by the SAD value of the current block. This suggest that if there is a high *mvd* of the current block, then it is highly probable that it lies at the boundary of a moving object. Further, a larger SAD value denotes that the current block has high variance and texture. Thus, a block with high MVD needs to be processed at the encoder, as this block will most probably have a very low correlation with its spatial and temporal neighbors. Therefore, the blocks with high spatial and temporal correlation can be accurately predicted at the decoder-side without excessive decoder-side ME. Blocks with high spatial and temporal correlation need not to be processed for encoder-side ME, thus using the encoder-side energy budget for more important blocks. The key challenge is to account for motion vector drift for ROI identification and ranking ROI-blocks.

3.3 Hardware Platform Analysis

This section provides analysis about the architectural aspects of a video processing system.

3.3.1 Heterogeneity among Computing Nodes

For presenting the impact of heterogeneity on different characteristics of the system, we provide different

Table 3-1: Attributes of cores and benchmarks

Core	Attributes	Area [mm^2]	Average cycles per operation		
			DCT	Quant	HEVC
Tiny	L2=64 Dispatch=1	86.76	4594	3416	100×10^6
Medium	L2=256 Dispatch=2	89.99	2378	1658	59.0×10^6
Large	L2=512 Dispatch=4	95.26	1687	971	45.3×10^6

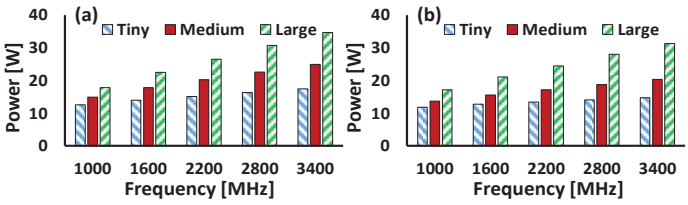


Figure 3-11: Power-frequency profiles for (a) DCT and (b) Quant

compute cores and benchmarks, as given in Table 3-1. These numbers are obtained for 45nm x86 cores, with 32 KB L1 data and instruction caches via Sniper simulator [226] and McPAT [227]. The average cycles per operation are obtained by running the benchmarks for the set of frequencies given in Figure 3-11. The variance of the cycles per benchmark is negligible, and therefore, the average numbers are reported.

Figure 3-11 also provides the power profiles of the cores used in this work. For the "Quant" benchmark, although the number of cycles consumed by the "Large" core is ~3.5× lesser than the "Tiny" core, the power consumption of the "Large" core at the lowest frequency (i.e., 1000 MHz) is greater than the power of the "Tiny" core at maximum frequency (3400 MHz). This discussion conveys that heterogeneous cores/nodes result in varying complexity (in number of cycles consumed) and power consumption for the same workload. Therefore, the workload balancing approaches must consider the complexity and power characteristics of the underlying compute nodes. Further, the power consumption is approximately independent of the type of application.

3.3.2 Memory Subsystem

Recent studies have shown that memory is the one of the main energy consuming system modules [228], especially in video processing/compression systems [32, 229]. Typically, in video coding applications, large frame dimensions and high processing rates put enormous pressure on the off-chip and on-chip memories [147]. The primary reason for this is the storage and repetitive accesses to large-sized video frame buffers [178, 177]. For instance, video coding of 4K UHD at 30fps results in a memory access rate of >356 Mega pixels-per-second, i.e., 2.78Gbps. Also, each frame will require ~12MBytes of memory.

These video frames are usually stored in the high-capacity off-chip memory. To avoid frequent accesses to the off-chip video memory or to alleviate the external memory transfers, state-of-the-art deploys on-chip video memories to store parts or full video frames [61, 230]. Therefore, data must be brought from the off-chip memory to the on-chip memory (typically SRAM) in order to reduce the access latency and external memory bus-contention [231]. Moreover, several repetitive accesses are made to the same pixel locations in advanced video coding standard due to multi-level filtering and excessive ME operations. Therefore, larger on-chip memories are essential to improve the performance and energy efficiency of video coding applications. This results in high power/energy consumption due to:

- High leakage power as a result of larger on-chip memories to store reference and current video frames. Traditional SRAM-based on-chip memories have high silicon footprint and leakage that may even surpass the dynamic energy. A reduction in the leakage may be obtained by exploiting emerging memory types.
- High dynamic power due to increased data rates as a result of bigger resolutions, high frame rates, and complex processing flow of advanced video encoders (for instance, HEVC) employs multiple filters and a complex multi-mode ME process.

Therefore, in order to significantly reduce the overall energy (considering both leakage and dynamic) of on-

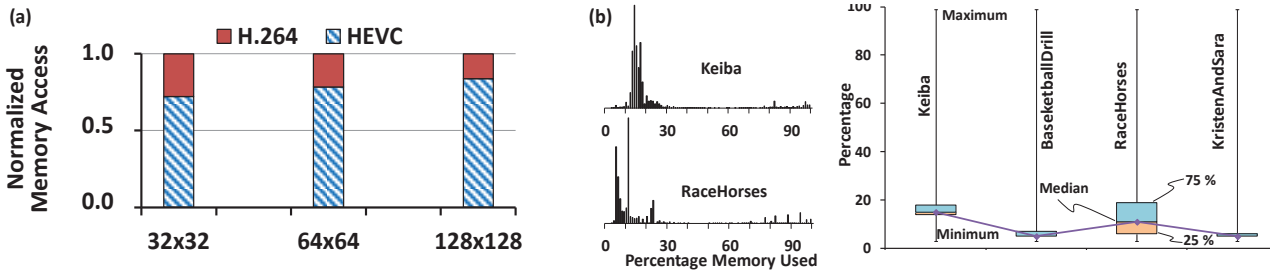


Figure 3-12: (a) ME memory access comparison between H.264/AVC and HEVC for three search window sizes, (b) ME memory access percentage statistics for TZ search in HEVC

chip video memory subsystem, application-specific video system architectural optimizations and energy-management are required.

3.3.2.1 Analysis of Motion Estimation

In Figure 3-12 (a), the memory access requirements for ME algorithm, for a single reference frame are shown. Three different search window sizes ($s_w \times s_h$) are chosen for both HEVC and H.264/AVC. HEVC memory access requirements are ~3.86× more compared to H.264/AVC. This shows that HEVC puts tremendous pressure on the memory system by making a large amount of memory read accesses. H.264/AVC based memory access reduction approaches thus do not scale properly if applied to HEVC. Furthermore, the search window method requires a frame to be read at least $r_f = s_h/b_h \geq 3$ times from the external memory (see discussion in Section 2.2.1.2). Using a single search window of size 256×256, a 4K UHD frame will therefore require 11.12Gbps fetched from the external memory. With n_r reference frames and additional read and write to the external memory for the current frame, a total of $(11.12n_r+2.78)$Gbps are read and around 2.78Gbps are written. Note that the external bus power dissipation is directly proportional to the total number of bit-toggles per transition [232]. This enormous amount of data results in high latency and energy consumption (around 40% of the total system energy is consumed by the external I/O [64, 169]). Therefore, saving external memory accesses becomes vital for reducing the energy consumption of a video encoding system. Thus, using on-chip frame buffers will reduce the total energy consumption and the external memory bus-contention.

Figure 3-12 (b) shows the histogram of percentage memory access to the search window in HEVC for different video sequences, using the state-of-the-art ME algorithm, TZ Search [66]. A search window of size 64 (or 36 Kbytes) is taken. These graphs are averaged per frame. The box-plots for these statistics are also shown and we notice that less than 20% of the search window is utilized (see discussion in Section 2.2.1.2), i.e., mainly a part of the search window is utilized. The unused search window consumes leakage power. Therefore, adapting the search window according to the needs of the block-matching algorithm can result in high energy gains.

3.3.2.2 Hybrid Memories

For general memory subsystems, in order to reduce the leakage energy consumption and to reduce the latencies and dynamic energy associated with the write operation, research has focused on combining the best of NVMs and VMs and coined the term “Hybrid Memory”, e.g., to form 2D or 3D stacked on-chip memories [30, 233]. Nevertheless, the same principles can be applied for the video memory required by ME.

However, from Table 2-2, it is evident that straight-forward replacement of the SRAM or DRAM based memories with NVMs is not possible. Special consideration to the memory characteristics must be involved in the system design. We can reduce the leakage energy of a system by introducing NVMs in place of traditional VMs (like SRAMs or DRAMs), but at the same time, NVMs can play a major role in deteriorating system response time and dynamic energy consumption. Therefore, the advantages and disadvantages of NVMs must

be carefully weighed against each other. A system designer needs to consider the application behavior, usage constraints and system dynamics before employing the correct memory subsystem, which would result in the best benefit to cost ratio. For example, using Table 2-2, we can generate a benefit to cost technology priority by adding the blues for each technology and subtracting the reds from it. Like, if the application requires processing large data sets with high data reuse (high read operations and low writes in a memory), a MRAM is a better option than SRAM etc.

The detailed design characteristics (speed, power etc.) of different memory subsystems is shown in Table 3-2. The on-chip video frame buffers can be either SRAM or MRAM based memories. From this table and Table 2-2, the SRAM and MRAM read latencies and energies are similar, but the leakage energy of SRAM is ~21× of MRAM having the same capacity. Also, MRAM capacity is about 4 times that of SRAM for the same area. Therefore, MRAM is a better candidate for on-chip video frame buffer. But the write energy and latency of MRAM is ~20× and 2.6× of SRAM respectively. Thus, feeding the MRAM from external memory is not ideal, as the external memory read latency added with the write latency of MRAM can severely degrade system performance.

Table 3-2: Comparison between different memory types for a 65nm technology [29]

Memory Type	Speed (*nsec*)		Energy/Power			Area (mm^2)
	Read	Write	Dynamic (*nJ*)		Leak. (*W*)	
			Read	Write		
SRAM (4 MB)	4.659	4.659	0.103	0.103	5.20	44
DRAM (16 MB)	5.845	5.845	0.381	0.381	0.52	49
MRAM (16 MB)	4.693	12.272	0.102	2.126	0.97	38

3.3.3 Analysis of Different Aging Balancing Circuits

In this section, we provide a detailed aging analysis, in terms of SNM degradation and duty cycle imbalance, for different test video sequences [234, 235] and highlight issues related to image regions with distinct properties. This analysis is leveraged for developing our aging resilient video memory.

Figure 3-13 shows the total percentage of bits (in form of a histogram) for a frame memory that are overwritten by a complementary bit of the new frame. As noticed, for some video sequences with low activity, this histogram is crowded towards smaller percentages, which tells us the writing new frame will only marginally release stress on some 6T cells of the SRAM (as the duty-cycle, Δ, will be highly biased towards 0.0 or 1.0). Additionally, the histograms are not dispersed, showing that there is a significant correlation (in terms of texture and motion) between temporally neighboring frames (also see Figure 3-10 (b-d)). Therefore, the properties of the subsequent frames can be estimated from the history, such that it can be leveraged for efficient aging balancing. Specifically, we can predict the aging impact of the current and future video frame by analyzing

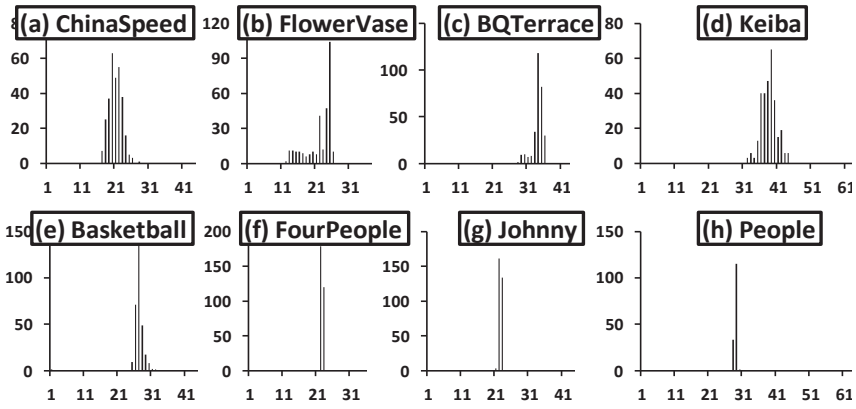


Figure 3-13: Percentage histogram of SRAM memory overwritten with new bits by video sequences. X-axis presents the percentage of bits changed in the subsequent frames, y-axis presents the number of times a certain percentage occurs for 300 continuous frames.

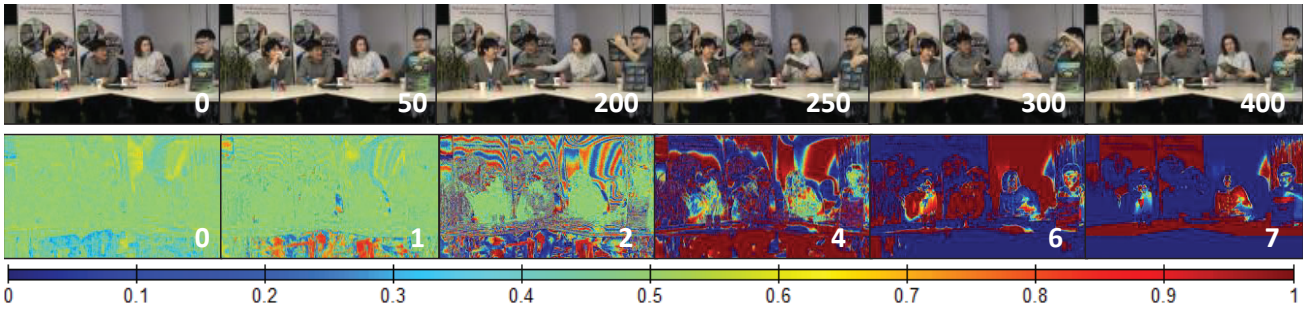


Figure 3-14: (top) Video sequence "FourPeople" (1280×720) with frame numbers written and (bottom) Δ stressmaps for specific bits of the video sequence, by plotting duty cycle (Δ) of the specific bit on spatial scale. The higher order bits have a biased Δ whereas the Δs of lower order bits are self-balancing

the aging effects of the previous video frames.

In Figure 3-14, duty cycle of a selected bits of the luminance component of "FourPeople" test video sequence is plotted on the spatial scale in form of so-called stressmaps. A balanced duty cycle (Δ=50% or 0.5) is represented with a light greenish color (see the scale below the pictures). For duty cycles heavily biased towards '0' and '1', we obtain a blue and a red colored distribution in the stressmap, respectively. It is noticed that lower order bits (LSB bits) have a balanced duty cycle, thus, the 6T SRAM cells storing these bits have a regular relaxations and an extended lifetime. However, the higher order bits (MSB bits) have a highly biased duty cycle, which causes an aging imbalance in the associated 6T SRAM cells (i.e., one out of the two PMOS transistors of the SRAM cell is under increased stress). Different critical video applications like security surveillance and space exploration missions experience such long-duration static scenes. In summary, duty cycles of different bits are not balanced and some bits age quicker than the others. In such cases, it becomes necessary to balance duty cycle of each bit individually, and to leverage the knowledge of bit location before applying an aging resiliency scheme. Therefore, it is imperative to estimate the duty cycle of each bit in addition to the balancing mechanism.

It is also important to note that less-frequently changing data will introduce the most amount of stress on the 6T-SRAM cells, for instance, low complexity texture and large, static backgrounds. This is also shown in Figure 3-14, where the large static background regions have a highly biased duty cycle. However, the moving regions in the video frame have a marginally balanced duty cycle. For example, the stressmaps for bit 7 has a relatively balanced duty cycle at the locations where people are moving. Therefore, the knowledge of less-frequently and more-frequently changing data can be exploited to distribute video samples in the on-chip memory, such that the transistors of each SRAM cell experience some relaxation.

In order to balance the duty cycle, we extend the memory architecture of Figure 2-2 with additional aging resiliency tools in form of Memory Read Transducer (MRT) and Memory Write Transducer (MWT) connected to the memory read/write ports (see Figure 3-15 (a)). These transducers can be implemented using one of the three different aging balancing circuits as shown in Figure 3-15 (b-d). As examples, we use inversion (similar to [201]), nibble-swapping (similar to [199]) and bit-rotation (similar to [198]), which correspond to the state-of-the-art approaches to tackle SRAM aging. For inversion and nibble-swapping, the bits of every second frame are inverted and swapped, respectively. In the bit-rotation circuit, the video samples bits are incrementally rotated by one with every frame, before writing to the frame memory.

In Figure 3-15 (e-h), the stressmaps of bit-7 for using no balancing (i.e., the base case) and the balancing circuits presented in Figure 3-15 (b-d) are plotted. As noticed, the duty cycle for the inverter case is much balanced compared to the other circuits. For convenience, the information about the duty cycles for all bits is presented in the form of boxplots, as shown in Figure 3-15 (i-l). In the box plot, the distribution of duty cycle

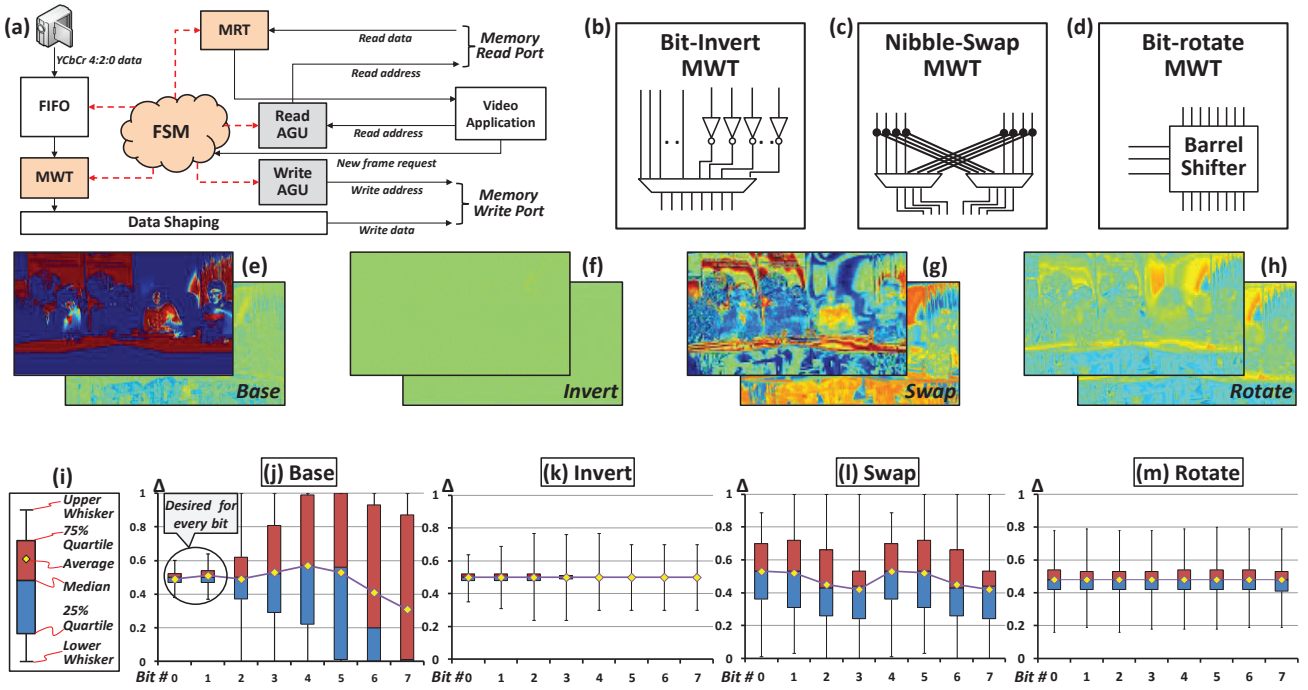


Figure 3-15: (a) Insertion of aging resiliency components (i.e. Memory Read and Write Transducers, MWT and MRT) in video memory management of Figure 2-2, (b) Bit-inversion MWT to invert all the bits of the video sample; (c) Nibble-swapping MWT to swap MSB bits with LSB bits; (d) Bit-rotation MWT to rotate bit locations with every frame; (e-h) Stressmaps for bit-7 (foreground) and bit-0 (background) for the above MWTs, with the bit-invert scheme outperforming bit-swap and bit-rotate; (i) Box plot legend; (j-m) Box plots for the above MWTs, where all the bits are best-balanced for the bit-inversion.

of each bit is mapped to quartiles. In an ideal case, the spread of whiskers in the box plot should be minimal and the median of the box plot should be at '0.5'. The spread biased towards '0' indicates a higher number of "zeros" at the bit location and vice versa. As noticed for the base case, for lower order bits (bits '0' and '1'), the spread of duty cycle is limited and the median is closer to '0.5'. However, the spread of duty cycle is large, and the median is not strictly '0.5' for higher order bits (bits 3-7). This suggests that the higher order bits in the video sequence need more care and resiliency features embedded into the system must account for these bits. Furthermore, the lower order bits experience auto-balancing and the aging resiliency feature for these bits can be turned off to save power.

By using balancing circuits of Figure 3-15 (b-d), the boxplots significantly change. For inverters, we notice that the duty cycle is nicely balanced for each bit and the spread is limited. The nibble-swapping introduce some improvement in balancing the duty cycle, but it is not comparable to that of the inverter circuit. Moreover, the nibble-swapping also adversely impacts the aging of bit '0' and '1'. Bit-rotation fits in the middle of the inversion and nibble-swapping cases. Moreover, by keeping the inverter ON for all bits, at all the time is not energy efficient. Therefore, the challenge is to design an adaptive, configurable controller to select the frame inversion rate and bits to invert at runtime.

Chapter 4 Video System Software Layer

This chapter provides details about the runtime management of the video processing system at the software layer. The main responsibilities addressed in this layer are to allocate processing nodes, realize power-efficiency and budget power to the video system. In order to parallelize the execution of a video application, resources are allocated to the application at runtime, by considering the hardware attributes of the system. Further, the workload is distributed among the parallel running threads in a way that throughput-per-watt is increased. Video application properties are also exploited at runtime and these properties are used to adjust the configuration knobs, which leverage power/complexity with the output video quality. Moreover, resource and power allocation to multiple applications running concurrently on multi-/many-core homogeneous and heterogeneous systems are also discussed. In addition, owing to the resource and energy constraints on constrained video processing systems, novel mechanisms to offload the workload from these applications to a high-end devices are described.

4.1 Power-Efficient Application Parallelization

As discussed in Section 2.1, a video applications usually processes a block of pixels at one time. Consider that a block of pixels processed by the application is treated as a job (e.g., a MB in H.264/AVC and CTU in HEVC). A set of jobs is denoted by a subtask η. And a set of subtasks constitute a task $\boldsymbol{\eta}$. To imagine the hierarchy, one can consider a task being analogous to processing the complete video frame, a subtask as processing a video slice/tile and job as processing the video frame block within the slice/tile (see Figure 2-10). Mathematically, a task i can be composed of n_i total subtask and given by:

$$\boldsymbol{\eta}_i = \left\{\eta_{i,0}, \eta_{i,1}, \cdots, \eta_{i,n_i-1}\right\} \tag{4-1}$$

An application can be designed with independent threads for each job, subtask, or a task. In case each job has its associated thread, there can be a large number of threads in the system which will result in more context switches and communication/synchronization among the threads, leading to a large overhead. For threads per subtask or task, the number of threads are reasonably low and so is their associated overhead. In case of video applications, per slice/tile (i.e., subtask) thread results in higher video quality, because the dependencies among the video blocks are exploited which can result in higher compression. A per block thread may not let a block to maximally exploit dependencies among neighboring blocks.

Let us assume an application a is required to process all its subtask (a complete task) within a deadline $t_{a,max}$, given r_{tot} number of resources (compute nodes/cores). A core can process one or more subtasks. Suppose a core j, associated with a task i, takes $t_{i,j}$ amount of time to process is allocated subtask(s). Mathematically, the workload of a task (defined in time units) is given by:

$$t_i = \max_{\forall j \in \{0,\cdots,k_{a,tot}\}} \left\{t_{i,j}\right\} \tag{4-2}$$

Here, $k_{a,tot}$ is the number of cores used for processing the subtasks. In order to maximally utilize the hardware resources and also to increase the throughput-per-watt ratio, the time taken by each core to process its allocated subtask(s) should ideally be identical, i.e., all cores start and end their allocated subtask(s) at the same time. This means that the workload among the cores processing these threads must be balanced. Generally, reducing the number of subtasks (for an equivalent amount of jobs) worsens the output of workload balancing strategies due to lesser degree of freedom, but it decreases the management overhead. Moreover, voltage-frequency levels of the associated compute nodes can be scaled (DVFS) to achieve workload balancing among the cores. On the other hand, a large number of subtasks will result in better workload balancing [236] at the expense of (a) increased management, (b) communication overhead and (c) reduced output video quality.

An additional challenge to address is that the time taken by a subtask (and thus the time taken by a task) can vary at runtime, according to the scenarios discussed in Section 3.2.2. Moreover, the system must determine the correct number of resources (cores) allocated to an application a ($k_{a,tot} \leq r_{tot}$) which will result in high power-efficiency while meeting the application's deadline. That is, the following optimization problem is solved:

$$\underset{k_{a,tot} \leq r_{tot}}{\arg\min} \left\{ \max_{\forall j \in \{0,\cdots,k_{a,tot}\}} \{t_{i,j}\} \leq t_{a,\max} \right\} \tag{4-3}$$

Basically, the number of cores utilized for processing a task is reduced as much as possible. A large number of cores will unnecessarily reduce the time consumed (considerably below $t_{a,max}$) in processing a task. However, it will also increase the resource and power consumption. This optimization problem suggests to reduce the number of cores such that they just meet the throughput requirement.

In addition, video applications need to process data under tight output quality constraints. For example, HEVC necessitates high compression with a throughput constraint [237], which generates numerous challenges. As an example, Table 4-1 shows the result of utilizing different number of cores in order to encode one frame of the "Exit" sequence under 125msec. Ideally, all the parameters in Table 4-1 should be as low as possible. Encoder-1 with the least power consumption does not meet the throughput, whereas Encoder-3 needlessly increases the number of cores and hence the power. Encoder-2 might be able to sustain the workload if its frequency (and thus its power) is increased. Encoder-4 is a special case, where the application configuration (parameters) is intelligently tuned to reduce the complexity. This results in further power reduction, at the cost of reduced compression and quality (compare with Encoder-2). Thus, it is possible to determine a **computation configuration** (the number of cores and their frequencies) and **application configuration** (allowing tolerable quality degradation), which fulfills the throughput requirement while lowering the power consumption.

Table 4-1: HEVC encoding characteristics for "Exit" (640×480) video sequence using [226, 227]. Frequency of all cores is kept constant during the execution. Here, PSNR: Peak Signal to Noise Ratio. "Time" and "Bytes" denote time and bytes to encode one frame

Encoder	*Enc-1*	*Enc-2*	*Enc-3*	*Enc-4*
Cores/threads	1	4	9	4
Freq [MHz]	2000	2000	2000	2000
Time [msec]	659	178	78	125
Power [W]	4.865	18.03	40.54	18.02
Bytes	6806	7247	7262	7328
ΔPSNR [dB]	0	0.058	0.061	0.015

4.1.1 Power-Efficient Workload Balancing

For achieving balanced application execution on a many-core system, an approach is presented to adaptively determine the computation configuration, such that throughput constraints are met at low video quality degradation, while minimizing the total power consumption. By accounting for the resources of the many-core system, the number of cores and their frequencies which must be used to manage the application's workload is determined. Further, the frequency estimation model is derived and adjusted at runtime, in order to (a) encounter the load variations of the core and (b) make the proposed scheme portable. Afterwards, the application configuration can be optionally applied which results in further power savings. Summarizing, the following presents the novel contributions of this work:

- **Selecting an appropriate computation configuration** that determines the number of cores and their frequencies (power), and the number of subtasks along with their maximum workload (output quality), depending upon the required throughput and the hardware characteristics.
- **Subtask-to-core mapping techniques** to fulfill the throughput requirement, pertaining to the structure in which task is divided in subtasks. This approach determines the optimal number of cores for sustaining the application's workload such that the number of cores and power consumption is minimized. A bin-packing heuristic to allocate subtasks to the available compute cores is employed, and hence, it is made sure that the utilization of a core is maximized, while minimizing the number of active cores.

- **Adaptive frequency generation model**, which self-regulates by adjusting model coefficients using runtime statistics, to determine the frequency of the cores and make the proposed approach portable to other many-core systems.
- **Optional application configuration**, which tunes (curtails or enlarges) the workload of each subtask by tuning their configurations independently at runtime, by utilizing a feedback mechanism to maintain the output quality within tolerable limits. Since the subtasks are associated with their respective cores, the frequency of the cores is also adapted. This results in reduced power consumption.

The outline of the proposed approach is shown in Figure 4-1. The computation configuration is an independent module which requires the throughput requirement and status signals from the application, and directly controls the frequency of the cores. This module can be implemented as a separate library, interfaced with the OS kernel via system calls and requires little effort from the application designer by exposing configurations knobs to the application. Application configuration must tune application parameters, requiring the application designer to modify the source. However, since the application designer best knows the application, therefore, application configuration can be implemented via moderate effort.

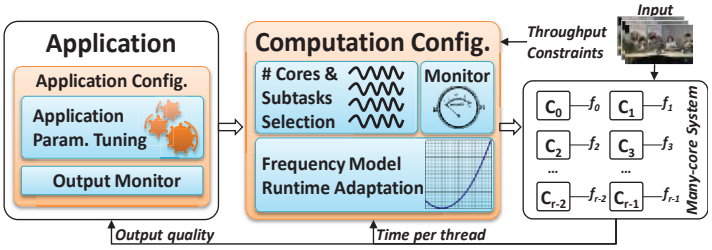


Figure 4-1: Overview of the proposed power-efficient workload balancing approach on a many-core system

The pseudo-code of selecting the proposed compute and application configuration approaches is given in Algorithm 1 and the sketch of the video system employing these approaches is presented in Figure 4-2. Since currently, the parallelization for a single application is under consideration, therefore, the "*a*" subscript for application is omitted for readability in this section. Moreover, for better understanding, the task will be replaced by a video frame and the subtasks by tiles. The index *k* is used for representing a specific core. In summary:

- At start, appropriate compute configuration is determined (line 6 of Algorithm 1) whereby the number of tiles/cores/threads is calculated ($k_{a,tot}$) to support $t_{a,max}$. The maximum required frequency to support the workload of each tile is also determined ($\boldsymbol{f_m}$, a vector having maximum frequencies $f_{k,m}$ of all cores). Further, if the number of cores is not enough, the workload of application is self-curtailed (by determining application configuration $\boldsymbol{\alpha_m}$). Throughout the execution of a core *k*, the frequency of (f_k) and workload ($\boldsymbol{\alpha_k}$) cannot exceed $f_{k,m}$ and $\boldsymbol{\alpha_{k,m}}$.
- The optional local workload tuner per tile determines the tile's application configuration $\boldsymbol{\alpha_k}$ (by changing parameters) for the epoch (a set of video frames, line 9). If reducing workload has tolerable impact on the output, the workload is reduced further.
- Based upon $\boldsymbol{\alpha}_k$, the frequency of a core *k* is predetermined for a complete epoch (group of frames, line 10), and with every new frame, a new frequency is set (line 11). Afterwards, the tile processing starts (line 12).
- Statistics are fed back to the frequency estimation model for adjusting the frequency of the next epoch (line 14). A Recursive Least Square (RLS) filter is used to adapt (or derive) the frequency estimation model constants.

The basic steps of the proposed scheme delineated above have the purpose of: (a) meeting the throughput demands by selecting the degree of parallelism (number of tiles), (b) appropriately balancing the workload and, (c) reducing the power consumption by selecting appropriate frequencies of the cores by exploiting tolerance to the output quality. In the coming text, the important blocks given in Figure 4-2 will be discussed in more detail.

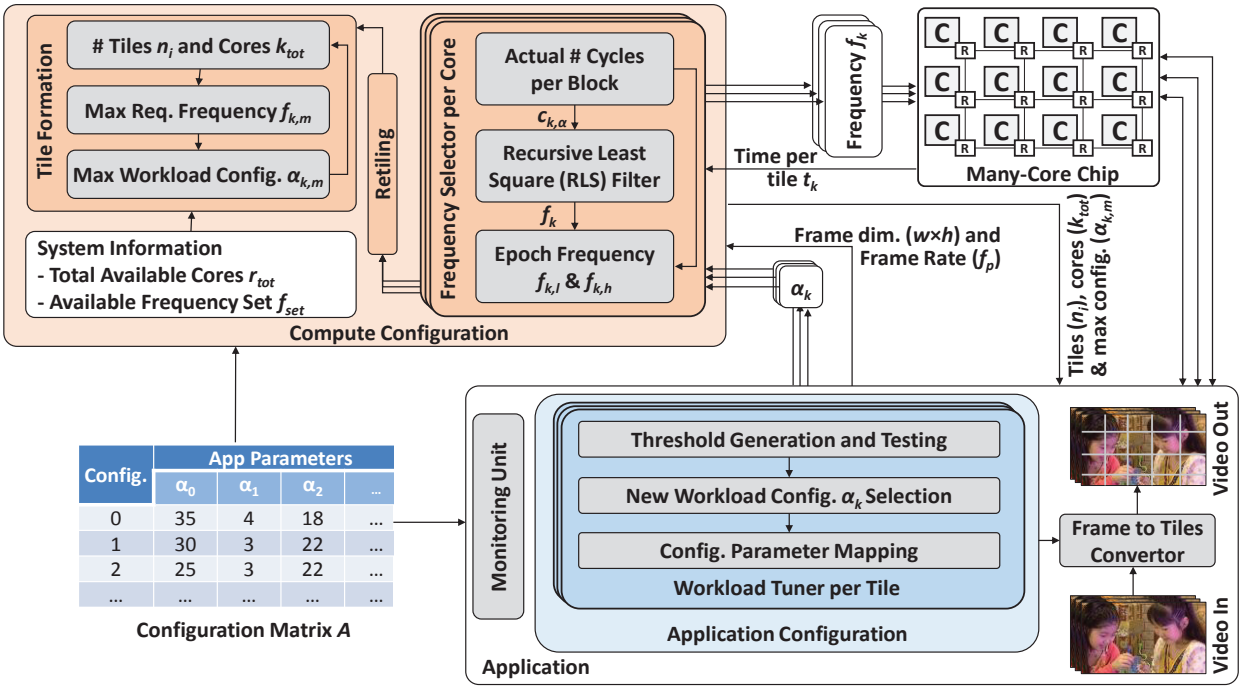


Figure 4-2: Power efficient workload balancing approach on a many-core platform.

4.2 Compute Configuration

For parallel video processing, a many-core system with per-core DVFS is used. A video frame needs to be converted into tiles for parallel processing. The tile formation block is responsible for appropriate division of the video frame into tiles (with n_i total tiles) and also selecting the number of cores (k_{tot}) to process the tiles, depending upon the allowable frequency of a core (available in a discrete set of frequencies $\boldsymbol{f}_{set}$, from f_{min} to f_{max}), the total (or allowable) number of available cores (r_{tot}) and the required frame-rate (f_p). Specifically, the tile formation scheme determines three attributes of the system. First, the number of cores (k_{tot}) is calculated which will collectively sustain application's workload according to Equation (4-3). Second, the maximally sufficient frequencies of these cores ($f_{k,m}$) is determined for the allocated workload (discussed later). And third, the maximum allowable workload configuration ($\boldsymbol{\alpha}_{k,m}$) is determined (for best output quality) which can be sustained by the hardware platform to satisfy f_p. The goal programming problem stated above can be mathematically presented as:

$$\begin{aligned} &\left\{ \min\{k_{tot}\},\ \ \min\{f_{k,m}\},\ \ \max\{\boldsymbol{v}_{k,m}\} \right\} \\ &s.t.\ \ k_{tot} \le r_{tot},\ \ t_k \le 1/f_p \\ &\qquad f_{\min} \le f_{k,m} \le f_{\max},\ \ f_{k,m} \in \boldsymbol{f}_{set} \\ &\qquad \forall\ k = 0, \cdots, k_{tot} - 1 \end{aligned} \tag{4-4}$$

Here, the highest priority goal is to reduce the total number of computing cores. However, the priority can be changed to minimize the frequency, or maximize output quality.

4.2.1 Uniform Tiling

Here, each tile (subtask η) is associated with a single thread, and every thread is dispatched to and processed by an individual core. This load distribution and balancing approach is termed as uniform tiling because it tries to keep the number of subtasks equal to the cores ($n_i = k_{tot}$) and tries to equally distribute n_{frm} jobs to every core. That is, the tile-based workload balancing attempts to equalize the number of blocks within every tile.

The program given in Equation (4-4) is solved by the algorithm given in Algorithm 2, which is diagrammatically shown in Figure 4-3. Primarily, this algorithm determines if a single tile/core/thread would potentially support the workload of video processing (line 1 in Algorithm 2). This test is carried out using the maximum output quality (or maximum possible workload, denoted by $\boldsymbol{\alpha}_{max}$=max($A$), line 6) at a minimum possible core frequency, $f_{k,m}$ (lines 7-8), while processing all blocks in a frame, n_{frm}. Such an arrangement will results in using least amount of resources, best video quality (see Figure 3-3 (d)) and minimal power consumption. If this configuration is not possible due to $f_{k,m}$ greater than the maximum supportable frequency (f_{max}), the workload (and hence, the output quality) is reduced (line 14) and the frequency is brought within f_{max} (lines 12-13). If there are other cores available, they are adaptively involved in the process (line 21). The algorithm repeats selection of the minimal frequency and maximum supportable workload for each individual core. The algorithm continues until either all the utilized cores can sustain the maximum workload (line 19) or there are no more cores available (line 15). Once the tile structure, frequencies and maximum workloads of each tile is determined, application processing starts.

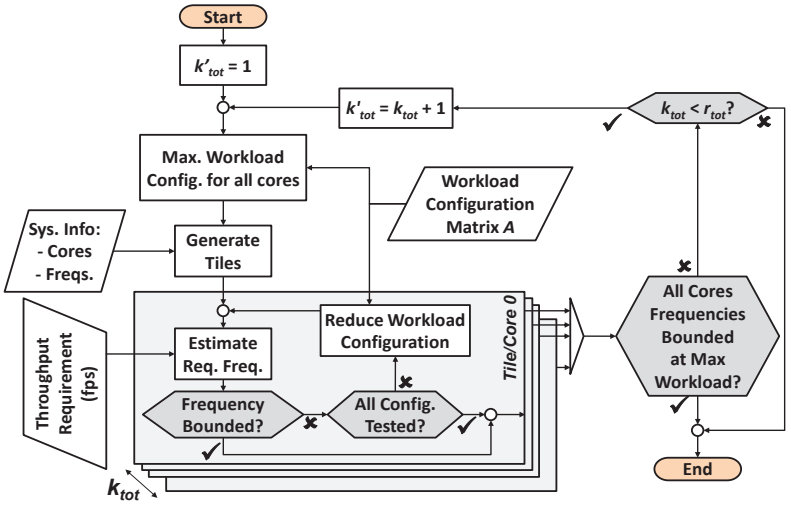


Figure 4-3: Uniform tiling, frequency and workload selection.

Tile Structure Selection ($u_w \times u_h$): For enhanced video quality, the number of tiles (and cores) and their structure $u_w \times u_h$ is adjusted by reading a lookup table (line 3 in Algorithm 2), as shown in Table 4-2. The input number of tiles/cores (k'_{tot}) is used to give the actual number of tiles/cores (k_{tot}) by reading this lookup table. For example, if k'_{tot} = 11, a higher video quality is achieved by having 4×3=12 tiles instead of 1×11=11 tiles. Thus, in this case, $n_i = k_{tot}$ = 12.

Table 4-2: Tile structure lookup table (can be adapted for an application). Here, "k'_{tot}" is the input, and k_{tot} and $u_w \times u_h$ is the output

k'_{tot}	k_{tot}	$u_w \times u_h$	k'_{tot}	k_{tot}	$u_w \times u_h$	k'_{tot}	k_{tot}	$u_w \times u_h$
0	1	1×1	**6**	6	3×2	**12**	12	4×3
1	1	1×1	**7**	8	4×2	**13**	15	5×3
2	2	2×1	**8**	8	4×2	**14**	15	5×3
3	3	3×1	**9**	9	3×3	**15**	15	5×3
4	4	2×2	**10**	12	4×3	**16**	16	4×4
5	6	3×2	**11**	12	4×3	**17**	18	6×3

4.2.2 Non-Uniform Tiling

In addition to the uniform tiling, a non-uniform tiling approach can be employed to reduce the number of cores used to process the video application's workload. This situation is useful in case of systems with no DVFS capabilities (where the frequencies cannot be tuned) to balance the workload of the cores. As the name signifies, the video tiles in this scenario are of not uniform sizes. Therefore, $n_i \geq k_{tot}$ and the n_{frm} jobs are not equally distributed among subtasks (i.e., it might occur that size(η_i) ≠ size(η_j) with $i \neq j$). Figure 4-4 (a) shows the concept overview of the proposed approach. Generating the tile structure is a two-step process where the first step consists of determining the master tile and the second step is the determination of secondary tiles. Further, depending upon the sustainable workload of a core, each tile might have different dimensions than the rest. An added advantage is that a table like Table 4-2 is not required.

For generating the video tile structure, following steps are carried out (also outlined in Figure 4-4 (b)):

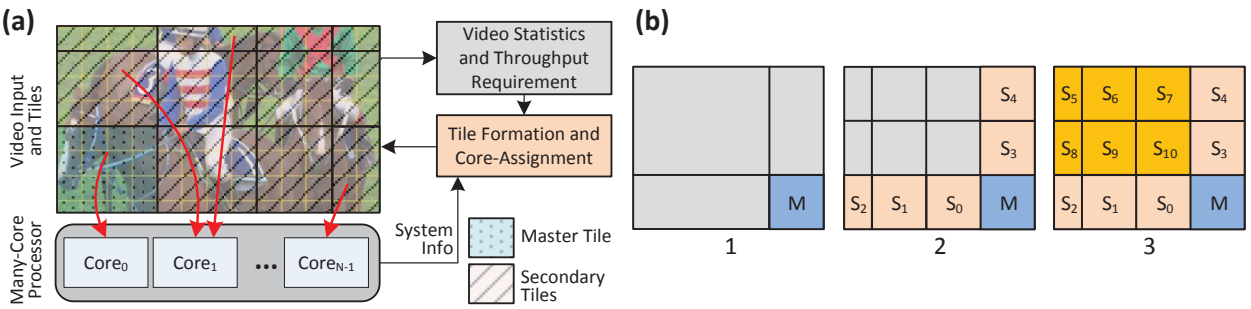


Figure 4-4: (a) Non-uniform tiling and cores assignment, (b) Master and secondary tile formation scheme

- **Determination of the master tile** (size and location) depending upon the video-content properties. The selection of the master tile location and dimensions involves computing the variance of video blocks.
- **Formation of secondary tiles** (size and location), depending upon the size and location of the master tile. The secondary tile structure is generated by extending the information about the master tile.
- **Determining the number of cores** required for video processing and assigning the master and secondary tiles to the cores. A bin-packing heuristic is used to determine the minimum number of cores.

To determine the master tile, the sustainable throughput of a core in terms of the number of blocks and frame-rate requirement are determined. Specifically, an equation can be derived via offline regression (or using online RLS filtering as given in Section 4.2.5) which can relate the time consumed in processing a certain number of blocks (n), frequency of the core (f), frame-rate requirement (f_p) and other application parameters (some of which are discussed in Section 4.3):

$$t = g_1(n, f, f_p, \boldsymbol{\alpha}) \tag{4-5}$$

Here, $\boldsymbol{\alpha}$ presents application configurations, discussed in more detail in Section 4.3. However, if a timing constraint of t_{max} is given, then one can determine n at $f=f_{max}$ using the relationship:

$$n = g_2(t_{max}, f, f_p, \boldsymbol{\alpha}) \tag{4-6}$$

Hence, the number of blocks in the master tile (n) are determined using the above equation. To find the dimensions of the master tile ($w_m \times h_m$), following formulas are employed:

$$\begin{aligned} w_m (< w) &= b_w \times \left\lfloor \sqrt{n / a_r} \right\rfloor \\ h_m (< h) &= b_w \times \left\lfloor n \times b_w / w_m \right\rfloor \end{aligned} \tag{4-7}$$

In this equation, the resolution of the video frame is $w \times h$ and the aspect ratio of the video frame is given by $a_r = w/h$. This formulation ensures that the master tile dimensions cannot exceed the dimensions of the frame. Notice that h_m is generated from w_m and thus, $h_m \leq w_m$ if $a_r<1$. However, it is possible to generate w_m with h_m. In the proposed implementation, the order depends upon the resolution of the frame. That is, if $w \geq h$, h_m is generated from w_m and vice versa.

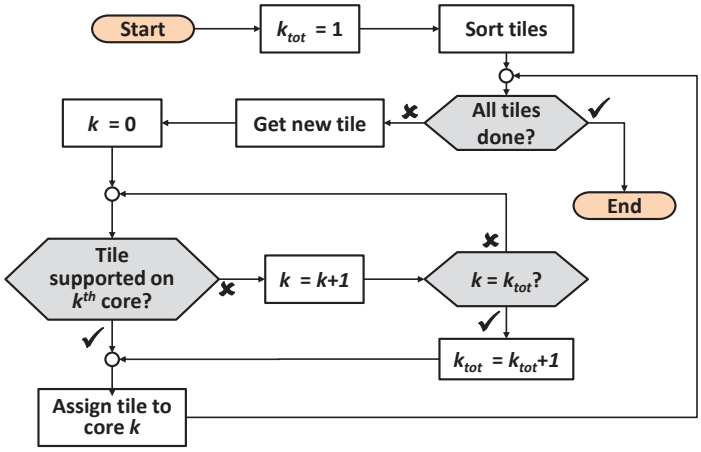


Figure 4-5: Tile-to-core assignment heuristic for non-uniform tiling.

Now, the size and dimensions of the master tile are available. The next step is to determine

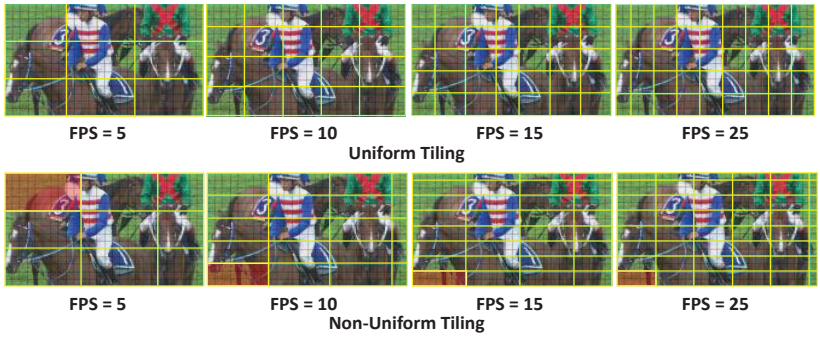


Figure 4-6: Demonstration of video tile formation in HEVC, by uniform and non-uniform tiling approaches presented in this thesis

Table 4-3: Analysis of the proposed uniform and non-uniform tiling for HEVC.

	FPS	Cores	Tiles	BD-Rate	BD-PSNR
Uniform	**5**	9	9	1.4126	-0.0897
	10	20	20	-0.1197	0.0089
	15	35	35	1.1671	-0.0667
	25	45	45	1.8266	-0.1048
Non-Uniform	**5**	9	9	1.5246	-0.0963
	10	17	20	0.0102	0.0014
	15	30	32	2.0505	-0.1182
	25	39	48	2.8470	-0.1645

the location of the master tile. For this purpose, the four corners of the video frame are searched for blocks with high variance. The corner with the highest accumulative variance has the master tile associated with it (as given in the lower right corner of the frame in step-1 of Figure 4-4 (b)). Afterwards, the determination of secondary tiles is straight-forward as given by steps 2 and 3 of Figure 4-4 (b).

Now, these tiles must be packed and dispatched to the individual processing cores. For this purpose, we use a bin-packing heuristic given in Figure 4-5. A tile is taken from the sorted list of the tiles (sorted in descending order according to the number of blocks n in the tiles). Iteratively, every core is tested whether the core can additionally sustain the workload of this tile, along with the other tiles the core is assigned. If yes, then the core will process the current tile in a time-multiplexed manner. Otherwise, a new core is introduced in to the computations. Once all the tiles are assigned to their respective cores, the frequencies of the cores are adjusted ($f \leq f_{max}$) to (a) reduce the power consumption and (b) just fulfil the workload of the tiles. The frequency adjustment formulas are given in Section 4.2.3.

4.2.2.1 Evaluation of Non-Uniform Tiling

A demonstration of tile formation for "RaceHorses" sequence using the uniform and non-uniform tiling for different FPS is shown in Figure 4-6. For the proposed non-uniform tiling, the master tile is marked for easy reference. In Table 4-3, the number of cores and tiles used for HEVC processing, BD-Rate and BD-PSNR [59] are tabulated for different frame rates. Note that uniform tiling achieves better video quality. However, the number of computation cores required to sustain the throughput constraint (frame rate) are lower for the non-uniform tiling technique.

4.2.3 Frequency Estimation ($f_{k,m}$)

In order to estimate the frequency of a core ($f_{k,m}$ line 7 in Algorithm 2), the total number of cycles required for processing a block ($\hat{c}_{k,\alpha}$) is estimated (lines 4, 11). Determination of $\hat{c}_{k,\alpha}$ can be achieved via offline or online analysis. Mathematically, the number of cycles required per second on the core k, to encode tiles/subtasks having a total of n_k blocks, at a frame rate demand of f_p frames per second can be defined by the expression:

$$f_{k,m} = n_k \times \hat{c}_{k_{tot},\boldsymbol{\alpha}_m} \times f_p \tag{4-8}$$

This equation exactly denotes the required number of cycles per second (Hertz) of the core to encode the assigned tiles. Hence, one can estimate the frequency of a core (i.e., $f_{k,m}$) if the size of the tiles in blocks and the require frame rate is provided.

4.2.4 Maximum Workload Estimation ($\alpha_{k,m}$)

As noted in line 13 of in Algorithm 2, the maximum allowable workload for a core k is given by configuration tuple ($\boldsymbol{\alpha}_{k,m}$). This means that once the tiling structure is defined and the processing starts, the application's workload must never exceed the workload defined by $\boldsymbol{\alpha}_{k,m}$. The application configuration is selected from a matrix A where:

$$A = [\cdots, \boldsymbol{\alpha}_{min}, \cdots, \boldsymbol{\alpha}_{max}, \cdots]^T \tag{4-9}$$

The tuple $\boldsymbol{\alpha}_{max}$ has configurations for best output quality and hence maximum workload. The tuple $\boldsymbol{\alpha}_{min}$ is for lowest quality and workload. Both $\boldsymbol{\alpha}_{min}$ and $\boldsymbol{\alpha}_{max}$ can be selected by the user. For the matrix A, the tuple $\boldsymbol{\alpha} = (\alpha_0,\alpha_1,\ldots,\alpha_\psi)^T$ presents the application parameters which can be configured and adapted at runtime. An example A is shown in Figure 4-7.

Config.	App Parameters			
	α_0	α_1	α_2	...
0	35	4	18	...
1	30	3	22	...
2	25	3	22	...
...	...	...	...	...

Best Configuration

Suboptimal Configurations:
- Reduced complexity
- Reduced power
- Reduced output quality

Figure 4-7: An example workload configuration matrix A

4.2.5 Self-Regulated Frequency Model

After determining compute and application configurations (k_{tot}, $\boldsymbol{f_m}$ and $\boldsymbol{\alpha_m}$), application processing can start. In order to explain the process in detail, Figure 4-8 will be referred to in this section. Basically, each tile is processed in epochs. For each epoch of size z, an appropriate frequency ($f_k \leq f_{k,m}$) is selected and adapted at runtime. The details of selecting these attributes is now given.

4.2.5.1 Frequency Estimation

The purpose of the frequency estimator (per core) is to adjust the operating frequency of the core, f_k, according to the workload assigned to the core, $\boldsymbol{\alpha_k}$ and the number of blocks assigned to the core, n_k.

The frequency of a core can be computed from Equation (4-8). However, one needs to estimate the total number of cycles for a given workload configuration, $\hat{c}_{k,\boldsymbol{\alpha}}$. The total number of cycles spent on processing a block can be estimated by using a model:

$$\hat{c}_{k,\boldsymbol{\alpha}} = g(\boldsymbol{x},\boldsymbol{\omega}) = \boldsymbol{x}^T \boldsymbol{\omega}_k \tag{4-10}$$

Here, $\boldsymbol{\omega}_k$ is the vector which holds the model constants and the vector $\boldsymbol{x}$ holds the configuration parameters. For our implementation, we chose:

$$\boldsymbol{x} = [\alpha_0, \cdots, \alpha_\psi, 1]^T \tag{4-11}$$

Note that $\boldsymbol{x}$ uses the configuration parameters from the tuple $\boldsymbol{\alpha}$ in matrix A. The value 1 is used to encounter the error in the estimation model. Equation (4-10) is both application and platform dependent. To derive this model, note that:

- High complexity applications (like HEVC) are designed for specific platforms (e.g. tablets, smart phones). Since the application designer has the best knowledge of both application and platform, it is straight-forward for the designer to derive this model via regression analysis.
- Dummy runs of the application while setup can also be used to derive this model.
- The model is derived online, at runtime, by using the scheme presented below.

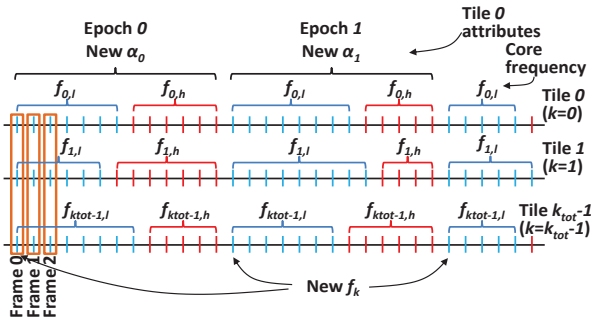


Figure 4-8: An example frequency and workload allocation to collocated tile in video application

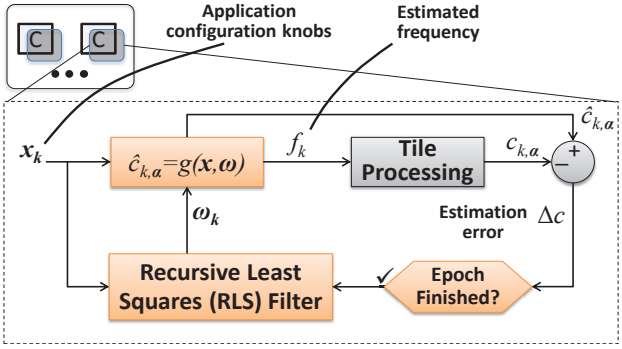


Figure 4-9: Per core frequency estimation model

4.2.5.2 Runtime Frequency Estimation Model Adjustment

The purpose of this scheme is to (a) determine the model constant $\boldsymbol{\omega}_k$ accurately or (b) fine-tune these constants at runtime. It is possible that $\boldsymbol{\omega}_k$ derived for one platform might not be accurate for other many-core systems if the system parameters (like processor type, cache size, external memory- and output-bandwidth etc.) are different. Further, scheduling other applications on the same compute cores used for the video application will also vary the total number of cycles consumed for processing. Some cores are physically located near the external memory controller and consume lesser cycles. It is also possible that no such model for estimating the number of consumed cycles is available. In all such cases mentioned above, it becomes advantageous to adjust (or derive) the model constants at runtime for an accurate estimation of the frequency. Therefore, scenarios where the system parameters are unknown or when the system load changes, adaptive model derivation or adjustment is of prime importance.

In this work, a Recursive Least Square (RLS) filter is used to adjust/derive $\boldsymbol{\omega}_k$ for each core at runtime, as shown in Figure 4-9. After the end of each epoch, the RLS filter is called to regulate $\boldsymbol{\omega}_k$. With each frame, in a feedback loop, the frequency adaptation scheme receives the time consumed for processing tile(s). Let t_k present the average time for processing on the core k. Since the frequency of the core (f_k) used for processing the tiles is known, actual number of cycles consumed per block can be calculated ($c_{k,\alpha} = f_k \times t_k / n_k$). RLS filter adjusts the model constants $\boldsymbol{\omega}_k$ at iteration m by using the formulas:

$$
\begin{aligned}
\hat{c}_{k,\alpha} &= \boldsymbol{x}_m^T \boldsymbol{\omega}_{k,m-1} \\
H &= E_{m-1} \boldsymbol{x}_m \left(1 + \boldsymbol{x}_m^T E_{m-1} \boldsymbol{x}_m\right)^{-1} \\
E_m &= E_{m-1} - E_{m-1} \boldsymbol{x}_m \boldsymbol{x}_m^T E_{m-1} \left(1 + \boldsymbol{x}_m^T E_{m-1} \boldsymbol{x}_m\right)^{-1} \\
\boldsymbol{\omega}_{k,m} &= \boldsymbol{\omega}_{k,m-1} + H \left(c_{k,\alpha} - \hat{c}_{k,\alpha}\right)
\end{aligned}
\tag{4-12}
$$

With every iteration of RLS, the model constants $\boldsymbol{\omega}_k$ and the estimation-error covariance matrix E is updated. Note that in Equation (4-12), the expression $(1+\boldsymbol{x}_i^T E_{i-1} \boldsymbol{x}_i)$ is a scalar; hence, no matrix inversion is involved. The takeaway from this discussion is that RLS algorithm can determine and regulate the frequency model constants and reduce the error in frequency estimation at runtime.

4.2.5.3 Core Frequency Allocation per Epoch

Note that the f_k might not be supported by the hardware platform due to quantized frequency levels. Usually, f_k lies in an interval bounded by $f_{k,l}$ and $f_{k,h}$ ($f_{k,l} \leq f_k \leq f_{k,h}$), where $f_{k,l}$ and $f_{k,h}$ are supported by the hardware platform. Therefore, we implement a frequency allocation heuristic as given in Algorithm 3 (and diagrammatically shown in Figure 4-8). For every core k, we solve the following integer linear program:

$$\begin{aligned} &\max\{\psi\} \\ &s.t.\ f_{k,m} \geq \sum \left(\frac{\psi\, f_{k,l} + (z-\psi) f_{k,h}}{z} \right) \geq f_k \\ &f_{k,l} \leq f_{k,h}, \quad \psi \in N^+ \end{aligned} \tag{4-13}$$

For consecutive z frames to be processed, the collocated tile k of every frame is encoded either at $f_{k,l}$ or $f_{k,h}$. In each epoch, we try to increase the number of collocated tiles associated with a lower frequency ($f_{k,l}$ in this case, by maximizing ψ). The high frequency collocated tiles (associated with $f_{k,h}$) are reduced as much as possible, while keeping the average processing time below the threshold.

4.2.6 Retiling

After a specific number of video frames (λ) are processed, if the frequency of all the cores is stuck to f_{min} or f_{max}, then retiling is performed (see Figure 4-2). Mathematically:

$$\mathrm{NT}() = \begin{cases} true & \text{if } f_k \in \{f_{\min}, f_{\max}\}, \forall k = \{0, \cdots, k_{tot}-1\} \\ false & \text{otherwise} \end{cases} \tag{4-14}$$

This is given in line 3 of Algorithm 1. It suggests that the estimation model of $\hat{c}_{k,\boldsymbol{\alpha}}$ (see Equation (4-10)) used to initially determine the compute configuration (k_{tot}, $\boldsymbol{f_m}$ and $\boldsymbol{\alpha_m}$) no longer applies. This can occur due to a non-applicable estimation model for $\hat{c}_{k,\boldsymbol{\alpha}}$. Hence, the tile structure is regenerated with the latest $\hat{c}_{k,\boldsymbol{\alpha}}$ which is adjusted by the runtime statistics of the encoder (see Section 4.2.5.2). Here, $\lambda=5z$ is taken, where, z is the number of frames in an epoch. Basically, $5z$ number of frames insure that a considerable number of times the estimation model of $\hat{c}_{k,\boldsymbol{\alpha}}$ is adjusted (according to Equation (4-12)).

4.3 Application Configuration

If required, the user's tolerance to output can be exploited for further power reduction of the system. In such a situation, any workload curtailing scheme can be added to the processing system and the frequency of the cores can be reduced. In short, the workload fluctuation and user's tolerance collectively translates to a workload-driven frequency/power adaptation that satisfies the throughput requirement. The per subtask/tile workload tuner shown in Figure 4-2 is responsible for adjusting the workload. Workload tuning is achieved by selecting a configuration tuple from A (see Figure 4-7). An update of the application configuration is performed after each epoch. And since the collocated tiles are highly correlated (see Figure 3-4), the adjusted configuration is applied to the next epoch. Since application configuration is application-specific, HEVC will be used as a case-study and the concept can be generalized to other applications.

4.3.1 HEVC Application Configurations

The tuple $\boldsymbol{\alpha}_k$ contains settings of different HEVC encoding of tile k. The following parameters can be used for adjusting the HEVC workload at runtime:

$$\boldsymbol{\alpha} = [\alpha_0, \alpha_1, \alpha_2, \alpha_3]^T = [\theta, d, QP, n_{frm} / k_{tot}]^T \tag{4-15}$$

Hence, we can rewrite Equation (4-11) as:

$$\boldsymbol{x} = [\theta, d, QP, n_{frm} / k_{tot}, 1]^T \tag{4-16}$$

Here, $\theta \in \{1,\ldots,35\}$ presents the total number of HEVC-Intra predictions that are performed per PU (see Figure 2-6 and [7, 47] for details). However, the important aspect is that increasing θ will increase the probability of selecting a prediction which results in the best compression efficiency, at the cost of additional workload and

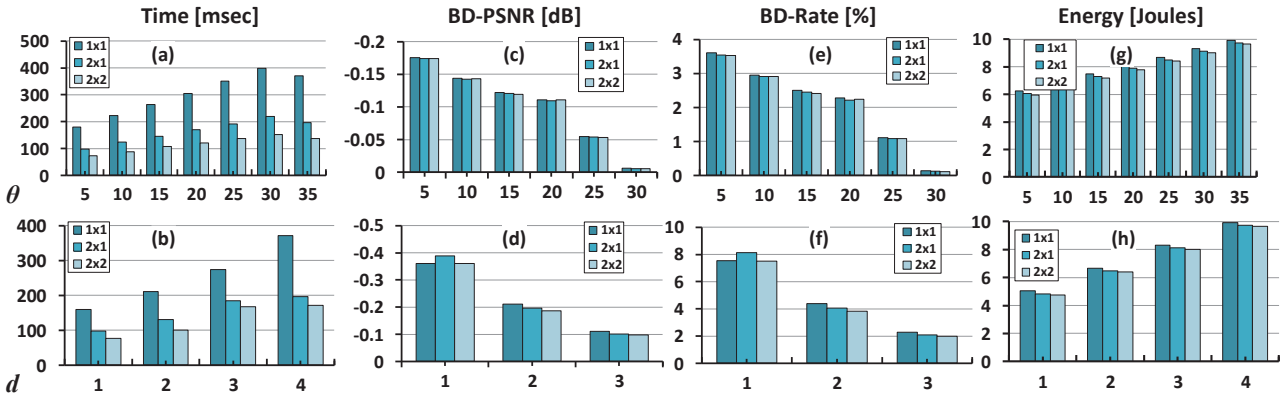


Figure 4-10: Impact of θ and d on (a, b) time per frame, (c, d) BD-PSNR, (e, f) BD-Rate [59] and (g, h) energy with different tile settings using "Keiba" sequence (832×480). The anchor encoding is done with 1 tile, maximum θ and d.

energy consumption. This situation is depicted in Figure 4-10 (a, c, e, g).

The HEVC parameter $d \in \{1,2,3,4\}$ presents the allowable depth of PU subdivision (see Table 2-1). Figure 4-10 (b, d, f, h) plots the impact of d on the time savings and bit-rate. Notice that by reducing d, the bit-rate increases, and the energy and workload decreases. This results in reduced compression efficiency. Compared to reducing θ, although the reduction of d causes a higher energy drop, it also incurs a higher quality loss (see Figure 4-10 (c-f)). Therefore, our first choice of parameter tuning is θ. Using the above mentioned definitions of θ and d (and keeping the visual quality constant by keeping Quantization Parameter QP constant), we can write the configuration matrix A (see Equation (4-9)) as:

$$A = \begin{bmatrix} 1 & 2 & \cdots & 35 & 1 & \cdots & 35 \\ 1 & 1 & \cdots & 1 & 2 & \cdots & 4 \end{bmatrix}^T \tag{4-17}$$

Further, our simulations have shown that the cycle model is highly dependent upon the parameter n_{frm}/k_{tot} and this term is also included in computation of the RLS update Equation (4-12). Hence, $\boldsymbol{\omega}_k$ and $\boldsymbol{x}$ are 5×1 vectors, and E is a 5×5 matrix due to the five workload tuning parameters (including θ and d) chosen here. A similar approach can be used for other video applications.

4.3.2 HEVC Configuration Tuning

For each epoch, the application configuration parameters (θ, d) are determined and remain fixed for all the collocated tiles in the epoch, as shown in Figure 4-8. If these parameters curtail the workload, the frequency f_k can be reduced, which will result in lower power consumption of the system. For workload tuning at runtime, the number of compressed bytes (b_k) generated after encoding video tiles associated with the core k are monitored (if b_k increases, this means that the output quality degrades and vice versa). If b_k increases beyond a certain threshold ($\tau_{b,k}$), the workload $\boldsymbol{\alpha}_k$ is increased (see Figure 4-10). This adaptive threshold is defined by:

$$\tau_{b,k} = \mu_k(b_k) + \sqrt{\sigma_k^2(b_k)} \tag{4-18}$$

Here, $\mu_k(b_k)$ is the average and $\sigma_k^2(b_k)$ is the variance of b_k, for all collocated tiles in epoch. Using Knuth's formula [238], one can update the mean and variance with every frame.

For every block within a tile, if threshold in Equation (4-18) is satisfied, θ is adjusted as:

$$\theta_k = \left\lfloor \mu_k(\theta) + \gamma_k\left(b_k - \tau_{b,k}\right) \right\rfloor_{\theta_{k,m}}$$
$$\gamma(h) = \begin{cases} +\psi/2 & \text{if } h > 0 \\ -\psi & \text{if } h \le 0 \end{cases} \tag{4-19}$$

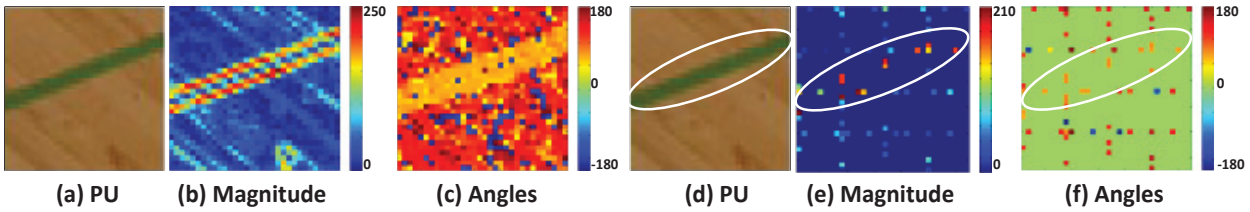


Figure 4-11: (a-c) Gradient generation for estimating the best Intra angular mode using [123], (d-f) proposed gradient generation with downsampled data

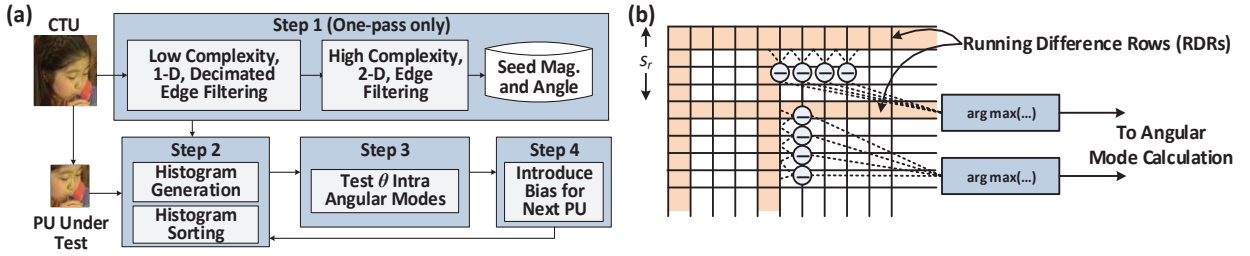


Figure 4-12: (a) Proposed fast Intra angular mode estimation, (b) Seed pixel extraction process for r_d=4

Here, ψ is a user defined parameter. Note that while employing Equation (4-19), $\theta_{k,m}$ (derived in Algorithm 2, $\boldsymbol{\alpha}_{k,m}$) also saturates θ_k as it is the maximum number of predictions tested for the tiles associated with core k. Further, it is possible that impact of reducing θ_k to increase the output bytes is not high and Equation (4-18) is always satisfied. If θ_k reaches a minimum possible value (θ_{min}=5, see Figure 4-10), the workload is reduced further by shifting d to the next lower level. If the output bytes increase and θ_k=$\theta_{k,m}$, the next higher d_k ($d_k \leq d_{k,m}$) is selected.

If a certain number of frames have been processed or b_k exceeds a threshold ($\tau_{b,k,m}$), we set (θ_k,d_k)=($\theta_{k,m}$,$d_{k,m}$) for the current epoch. This condition is mathematically given as:

$$b_k > (1+\varepsilon)\,\tau_{b,k,m} \tag{4-20}$$

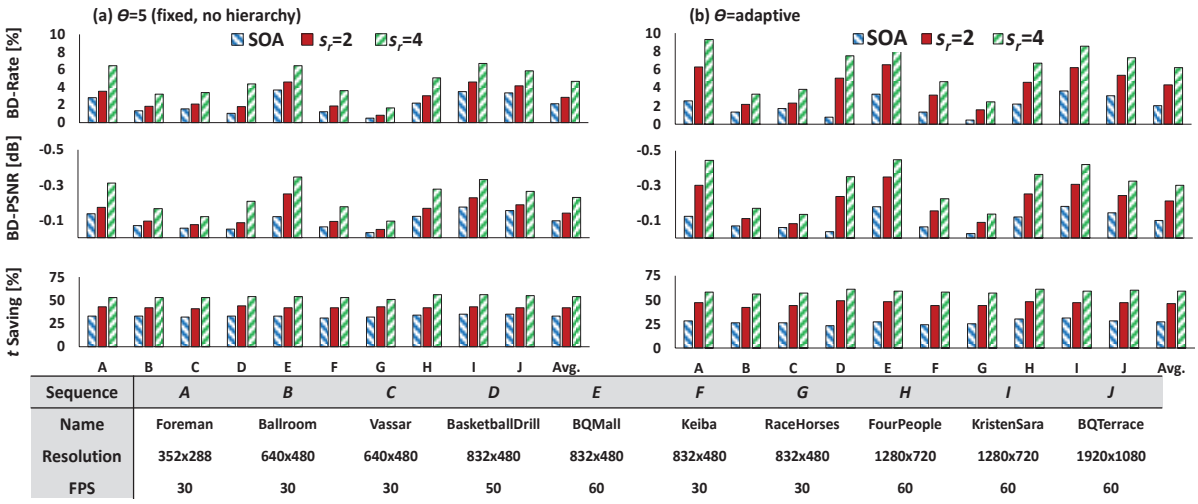
In this equation, $\tau_{b,k,m}$ equals b_k of the most recent tile with (θ_k,d_k)=($\theta_{k,m}$,$d_{k,m}$). Additionally, ε is a user-defined tolerance metric for b_k increase. Higher tolerance will result in reduced workload and vice versa. For example, if b_k of current tiles associated with core k is larger than $\tau_{k,m}$ by 5% (ε=0.05), the respective tiles in the new epoch are encoded with maximum workload and frequency (and maximum power).

4.3.3 HEVC Parameter Mapping

In the previous section, it is pointed out that the reduction in workload is achieved by curtailing $\boldsymbol{\alpha}$, i.e. reducing θ_k and d_k. However, one must intelligently include the most probable selection candidate for both θ_k and d_k while curtailing the workload by exploiting HEVC specific properties. In this way, the degradation of output quality will be smaller.

4.3.3.1 Intra Mode Estimation

As pointed out in Section 3.2.3.2, the individual pixels in a PU can provide hint to the possible texture flow direction if their gradients (generated via Sobel edge filtering kernel) are accumulated. Such a case study of a single PU is shown in Figure 4-11 (a-c) using a 32×32 sized PU actually selected by the RDO process for the "BasketballDrill" (832×480) sequence. Using gradient directions, each pixel is color-coded to suggest the Intra angular mode of the PU. Notice that computing gradients is a time consuming operation and not all pixels correctly suggest the actual direction of Intra prediction mode that is finally selected. Thus, to increase the speed



Sequence	A	B	C	D	E	F	G	H	I	J
Name	Foreman	Ballroom	Vassar	BasketballDrill	BQMall	Keiba	RaceHorses	FourPeople	KristenSara	BQTerrace
Resolution	352x288	640x480	640x480	832x480	832x480	832x480	832x480	1280x720	1280x720	1920x1080
FPS	30	30	30	50	60	30	30	60	60	60

Figure 4-13: Comparison of video quality (BD-Rate and BD-PSNR [59]) and time savings for the proposed mode estimation approach with State-of-the-Art (SOA) [123] for (a) fixed and (b) adaptive number of angular modes

of computations, it is advisable to avoid performing gradient extractions on pixels which will not contribute to estimate the Intra angular prediction with high accuracy.

In order to estimate the Intra angular prediction with high accuracy and to reduce the time complexity, the procedure is outlined in Figure 4-12 (a). As seen, a complete CTU is pre-processed before being fed to the HEVC Intra-encoder. Firstly, seed pixels and seed modes are extracted from the CTU and stored in a memory as shown in Figure 4-12 (b). Afterwards, only the contents of this memory are used to estimate the most probable Intra angular prediction that will be selected by the brute-force search process. Moreover, two complexity knobs (s_r and θ) are also available to the user through which a user can leverage the computational complexity with the compression quality.

To avoid complexity due to the Sobel operator, only two pixels with largest running difference on the boundary of $s_r \times s_r$ are used for generating gradient magnitudes and direction. These pixels are shown in Figure 4-11 (d-f) for the example PU. Afterwards, a gradient histogram is created using gradient magnitudes and angles, and this histogram is sorted. The first θ angles within the sorted list are used for Intra angular estimation. Here, the value θ can be selected as given in Equation (4-19).

Evaluation of Proposed Mode Estimation Approach: For evaluating the proposed early mode estimation scheme for HEVC Intra-encoding, our in-house ces265 video encoder [239] is used (for more information, see Appendix B). These evaluation are conducted on a Windows 7 computer, with 2.7GHz Dual-Core CPU and 8GB RAM. Only a single thread of ces265 is used for evaluations.

The resulting video quality comparison (in terms of BD-Rate and BD-PSNR), and time savings are given in Figure 4-13. In Figure 4-13 (a), θ is fixed for both the proposed and state-of-the-art [123] approaches, whereas in Figure 4-13 (b), adaptivity of both approaches is enabled. Note that the proposed approach with $s_r = 4$ provides the best time savings, but also results in the largest video quality loss. Using $s_r = 2$ provides a good balance between the time savings of the proposed approach compared to [123]. With a fixed θ=5, the additional time savings of the proposed scheme compared to [123] are 18% and 32% for $s_r = 2$ and $s_r = 4$ respectively. For adaptive θ, the additional time savings compared to [123] are 27% and 44% respectively. Note that the time savings presented here also consider the overhead of the proposed approach. The time savings are obtained using the following formula:

$$t\ \text{Savings}\ [\%] = (t_{baseline} - t) / t_{baseline} \times 100 \qquad (4\text{-}21)$$

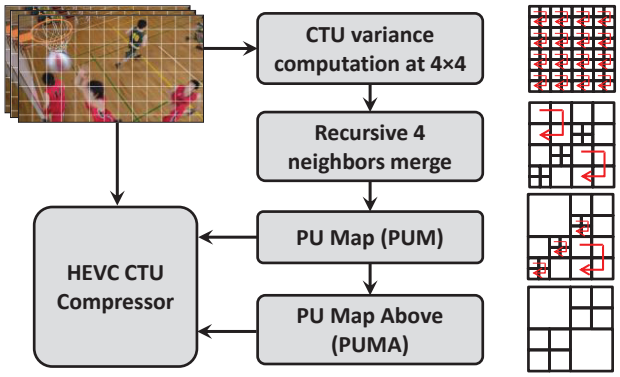


Figure 4-14: PU structure determination process

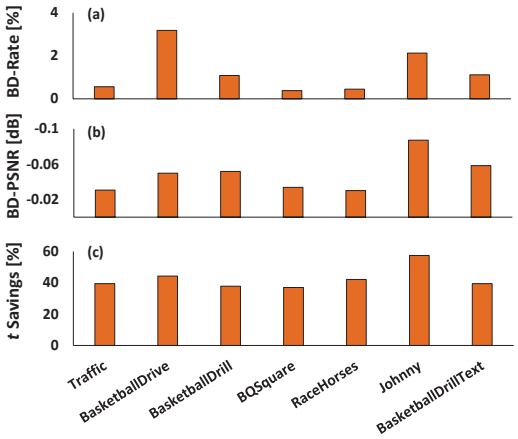


Figure 4-15: (a) BD-Rate, (b) BD-PSNR and (c) time savings for the proposed PU estimation

In this equation, $t_{baseline}$ is the time consumed by the encoder without any complexity reduction approach. Similarly, to compute BD-Rate and BD-PSNR, PSNR for a particular QP is computed via:

$$PSNR_{QP} = \frac{4 \times PSNR_{Y} + PSNR_{Cb} + PSNR_{Cr}}{6} \tag{4-22}$$

4.3.3.2 PU Depth and Size Selection

From Sections 2.2.1.1 and 3.2.3.1, we notice that HEVC tests every PU size in order to determine the PU structure which results in the best RDO. The application configuration, however, determines that a PU should only be tested for a selected number of depths. Unfortunately, testing PU sizes which are far off from the actual PU size selected by brute-force processing dents the compression performance of the encoder. For example, if allowable d = 1, then the CTU will only be tested with a PU of the same size as the CTU. For CTUs encompassing high variance regions (see Figure 3-7), this can result in compression loss. Therefore, an adaptive method is required to pre-determine the PU structure of the CTU, which accounts for the video-content (texture).

The proposed PU structure selection scheme works according the approach given in Figure 4-14 and Algorithm 4. First, the complete CTU is divided into sub-blocks of size 4×4 and variance of each sub-block is computed. Afterwards, neighboring four sub-blocks are combined/joined if they fulfill the following criteria:

- All four sub-blocks are of the same size.
- All four sub-blocks have a variance less than a threshold.
- The variance of the combined sub-block is also less than the threshold.

Once the above approach is employed for sub-blocks of size 4×4 for the complete CTU, the algorithm repeats with sub-block sizes of 8×8, until finally it cannot combine any four neighboring blocks. This PU structure is termed as PU Map (PUM). If the required depth for processing is set at 1, then PUM is used for encoding, and all other possible PU structures are discarded. If the depth is set to 2, then the adjacent/neighboring four sub-blocks of equal sizes are directly combined to form a PUM Above (PUMA), and only PUM and PUMA are given to the CTU compressor for encoding. If it is not possible to combine any sub-block from PUM to form PUMA, then the PUMA consists of all 4×4 PUs.

An important aspect of the above algorithm is to find the variance (v) of the combined sub-block. This will incur additional overhead. However, this overhead is reduced by using Chan's formula [240]:

$$(\mu, v) = \left(\frac{\mu_1 + \mu_2}{2}, \frac{2(n-2)(v_1 + v_2) + (\mu_1 - \mu_2)^2 n}{4(n-1)} \right) \quad (4\text{-}23)$$

Where two blocks, with mean and variances as (μ_1, v_1) and (μ_1, v_1) respectively, having $n/2$ entries are combined to form the resultant block of n entries with (μ, v). Additionally, the computational burden can be further reduced by simplifying the Chan's formula:

$$(\mu, v) \approx \left(\frac{\mu_1 + \mu_2}{2}, \frac{2(v_1 + v_2) + (\mu_1 - \mu_2)^2}{4} \right) \quad (4\text{-}24)$$

This formula is derived by approximating (n-1) and (n-2) as n (for $n \in \{32, 64, 128\ldots\}$).

Evaluations: The proposed complexity reduction approach is tested for HEVC and the results are reported in Figure 4-15. Figure 4-15 (a, b) provides the BD-Rate and BD-PSNR comparison of our approach to the RDO for different sequences. On average, the proposed approach incurs a BD-PSNR loss of -0.048 dB, which is insignificant compared the state-of-the-art scheme (-0.1184 dB) [126], which also uses two levels of PU size selection.

The percentage time savings of the proposed scheme for different video sequences with diverse characteristics is given in Figure 4-15 (c). The time complexity of the proposed approach is compared with the reference implementation. Note that up to 57% time savings are achieved, for sequences with diverse texture and motion properties.

4.4 Workload Balancing on Heterogeneous systems

In previous sections, the focus was mainly on distributing the jobs/subtasks to the parallel computing cores. However, it was not mentioned how the cores can be prioritized for distribution of subtasks. Mainly, a compute core with high power-efficiency must be allocated a larger quota of subtasks than a core with lower efficiency. For example, a hardware accelerator might be much more computationally efficient than a core running pure software.

Moreover, in this section, a computing element will be referred to as a node instead of a core, because the current discussion also applies to processing elements other than soft-cores.

4.4.1 System Model

Consider an application a, which processes the task i (also termed as the application's load) with n_i subtasks, that is scheduled to run on a heterogeneous system with multiple nodes. The application consists of multiple independent tasks that can be subdivided into n_i subtasks, and the subtasks can be executed in parallel. A node must process its assigned subtasks within a deadline ($t_{i,max}$).

The heterogeneous compute system consists of r_{tot} total nodes. The proposed approach selects only k_{tot} nodes ($k_{tot} \leq r_{tot}$) for sustaining the throughput of the application, and selects their voltage and frequencies (f_k for node k), which determines the power consumed by these nodes (p_k). Basically, the frequency of the node is a function of the number of subtasks allocated to the node ($n_{i,k}$) and the cycles consumed to process a job on node k ($c_{i,k}$) as will be discussed later.

The compute nodes can either be soft-cores, GPUs, accelerators etc. No assumption is made about the sustainable throughput and power consumed by these nodes. That is, each of these nodes can sustain a different throughput than the rest, and can consume different power for equivalent throughput. Further, each node has either a program written in software layer to process the subtask, or, has a custom hardware unit implemented in the hardware layer to process the given subtask.

The target of the proposed approach is to minimize the total power consumed by the heterogeneous system (p_{tot}) while meeting a certain throughput constraint. The optimization goal can, therefore, be written as:

$$\begin{aligned} &\min\left(p_{tot} = \sum_{k=0}^{rtot-1} p_k(f_k) \right) \\ &s.t. \\ &f_k \in \{(0) \cup \psi \mid (f_{\min} \le \psi \le f_{\max})\} \quad \forall k \in \{0, \cdots, r_{tot}\} \\ &\qquad t_{i,k} \le t_{i,\max} \qquad \forall k \in \{0, \cdots, k_{tot}\} \\ &\sum_{k=0}^{ktot-1} n_{i,k} = n_i \end{aligned} \tag{4-25}$$

This problem suggests that a node can either be power gated (with $f_k = 0$) or its frequency range can be between permissible limits. However, each parallel computing node must finish the tasks within $t_{i,max}$, while the cumulative number of subtasks processed by individual cores should equal the total number of subtasks.

In order to solve the proposed load distribution and balancing problem, some issues need to be addressed. First, determining k_{tot} (the actual number of nodes used for processing) is not known beforehand. Secondly, a node frequency is permissible to have values within two distinct ranges (i.e., either it can be 0 or between f_{min} and f_{max}). Moreover, the variables used in Equation (4-25) need to be derived in terms of tunable system parameters for optimization, which might be cumbersome. Thus, to efficiently solve these issues, a heuristic can be used, as explained below.

4.4.2 Load Balancing Algorithm

The resource allocation and load balancing approach is given in Figure 4-16 and Algorithm 5. Basically, the proposed load balancing approach is a two-step process:

- Gathering the compute and power profiles of all nodes before starting the distribution of subtasks among the nodes.
- Actual load distribution method, which distributes the subtasks among the nodes, depending upon the metrics gathered by the first step. Further, the compute and power profiles are also used for selecting the appropriate running frequency of the nodes.

For the first step, the power profiles (power vs. frequency of the node, $\boldsymbol{p}(\boldsymbol{f})$) of all the nodes is collected. This can be done using online power measurement (e.g., using Intel Power Gadget and reading MSR registers), or, by exploring the data-sheets of the nodes. Furthermore, the average power over the available frequency range or operation modes ($\boldsymbol{p_k}$) is also computed. Also, the maximum average power for all nodes ($p_{\mu,max}$) is determined via:

$$p_{\mu,\max} = \max_{\forall r_{tot}\ \text{nodes}} \{\boldsymbol{p}_{\boldsymbol{k},\boldsymbol{\mu}}\} \tag{4-26}$$

Note that in this equation, for simplicity, it is assumed that the power consumption of the nodes is independent of the type of the task. However, task based average power calculation can also be used here. The cycles consumed to process a job of task i on a node k (given by $c_{i,k}$) is collected for all the nodes using either offline regression analysis or online dummy runs of the application during setup. Similarly, the maximum cycles consumed by a node to process a job of task i ($c_{i,max}$) is also determined. Using these metrics, the *efficiency index* for a node k (ϕ_k) is computed using the following cases:

Case 1 $\phi_{k,1}$: The efficiency index of a node k is the ratio of the maximum cycles and power consumed to process a task i, for all the nodes, to the cycles and power consumed by the node k.

$$\phi_{k,1} = \frac{c_{i,\max}\, p_{\mu,\max}}{c_{i,k}\, p_{k,\mu}} \tag{4-27}$$

Case 2 $\phi_{k,2}$: The efficiency index of a node k is the ratio of the maximum cycles consumed in processing a task i for all the nodes to the cycles consumed by the node k.

$$\phi_{k,2} = \frac{c_{i,\max}}{c_{i,k}} \tag{4-28}$$

Case 3 $\phi_{k,3}$: The efficiency index of a node k is the ratio of the maximum average power for all the nodes to the average power of the node k.

$$\phi_{k,3} = \frac{p_{\mu,\max}}{p_{k,\mu}} \tag{4-29}$$

Index 1 shows that the node which takes the least cycles and power to process a job has higher computation efficiency (throughput-per-watt). Therefore, it must be preferred for use if a higher throughput-per-watt metric is desired. Similar argumentation can be made for Indices 2 and 3. Note that the efficiency index can also be changed to account for other metrics (like network hops among the nodes etc.).

For the second step, the actual load distribution algorithm takes place as shown in Figure 4-16 and Algorithm 5. Basically, nodes are adaptively introduced to share the load of the application, and then their frequencies are tuned such that the system consumes minimum amount of power.

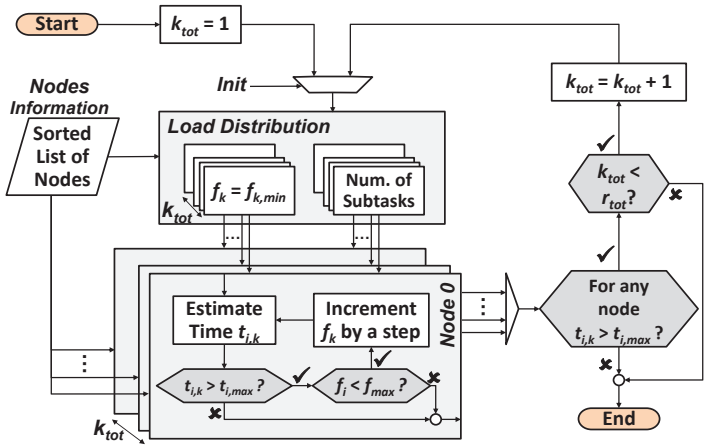


Figure 4-16: Workload distribution and balancing on heterogeneous nodes.

At start, the nodes are ordered in a list ($\boldsymbol{g}_\phi$) with descending efficiency indices (line 1). Every time a new node is required to distribute the subtasks, the next node from this list is considered (line 19). The processing starts with only one node (having the highest efficiency index max(ϕ) $\forall$ k_{tot}) and with its minimum frequency setting ($f_k = f_{k,min}$, line 5). That is, the entire load is allocated to this node ($n_{i,k} = n_{tot}$). Then, the time consumed by a node to process the workload is given by the formula:

$$t_{i,k} = \frac{c_{i,k}\, n_{i,k}}{f_{i,k}} \tag{4-30}$$

In case the timing constraints are not met ($t_{i,k} > t_{i,max}$), the frequency of this node is increased by a single step until $f_{i,k} = f_{k,max}$. If $f_{i,k} = f_{k,max}$, then the frequency cannot be increased further, and thus, more nodes are introduced ($k_{tot} = k_{tot} + 1$). The next node is fetched from the efficiency index array $\boldsymbol{g}_\phi$. Now, the algorithm starts again with minimum power configuration ($f_{i,k} = f_{k,min}$) for all k_{tot} nodes. The load is distributed to the node k using the following formula:

$$n_{i,k} = \frac{n_i \phi_k}{\sum_{j=0}^{ktot-1} \phi_j} \tag{4-31}$$

That is, the node with the highest efficiency factor gets more load. Once again, the time that will be consumed by a node while processing $n_{i,k}$ subtasks can be estimated using Equation (4-30). In case a node is unable to

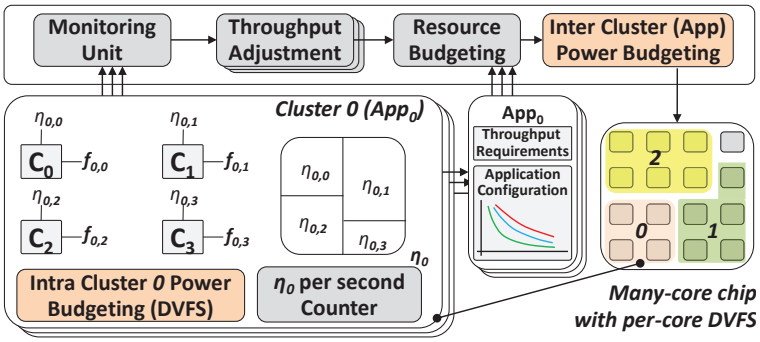


Figure 4-17: Overview of resource budgeting for mixed multithreaded loads.

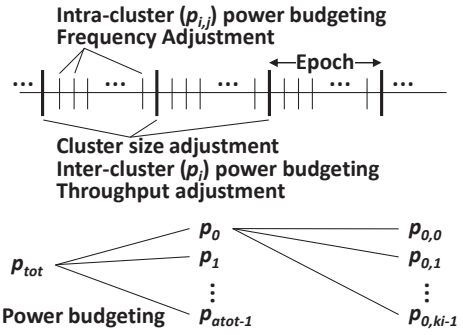


Figure 4-18: Proposed resource budgeting

process the allocated number of subtasks within $t_{i,max}$, its frequency is increased. If any node is at its maximum power capacity and the system still cannot sustain the workload, another computation node is introduced and the process is repeated. If all the nodes meet their deadlines, the algorithm terminates and the computation configuration which can sustain the application's load is found. Further, the maximum time taken by any node is defined as the time to process its assigned subtasks.

4.5 Resource Budgeting for Mixed Multithreaded Workloads

By interchanging the goals and the constraints of Equation (4-4), the target becomes to maximize the throughput of the system or an application, under a given power constraint (e.g., TDP, Dark Silicon). This optimization is particularly useful for high processing computing, and offline video processing. If, unrealistically, power constraints of the system are not specified, the techniques mentioned in Sections 4.2 and 4.4 can be extended by running each core at maximum frequency.

In such cases, it is required to determine the appropriate compute configuration (the number of cores and their frequencies) for the given power budget (p_{tot}). This resource distribution becomes more challenging if multiple tasks with associated subtasks and jobs, are running in parallel. Each of these tasks can denote a separate application, and therefore, the application index a and task index i can be used interchangeably in the coming text, i.e., $a = i$. Note that in this section, the resource of a task i denotes the number of cores k_i and power p_i associated with each task. For video applications, this resource budgeting is applicable to multicasting scenarios (see Section 2.2.2), whereby every encoder is a separate application trying to encode its own video frames (i.e., tasks), and each video frame can be divided into tiles (i.e., subtasks).

This thesis targets a multi-granularity scheme that distributes the resource budgets for multithreaded workloads at different levels (among tasks and within tasks). The complexity of the resource distribution problem is addressed via a hierarchical approach, which also provides performance isolation and fairness among applications running these tasks. The distribution of resources targets the fulfillment of the minimum throughput requirement of all tasks. Summarizing, the problem addressed in here is: How to budget resources and TDP among different tasks/applications, such that the throughput of concurrently executing, multithreaded applications is maximized? Since each multithreaded application will require multiple cores for executing its workload, we denote the set of cores of a multithreaded application as a "Cluster". The proposed scheme (Figure 4-17 and Figure 4-18) performs the following key operations:

- **Selection of an Appropriate Number of Cores (i.e., Cluster):** For a given task, the number of cores required to fulfill its throughput constraints is estimated based upon the task's workload characteristics and performance constraint.

- **Cluster-Level Power Budget Distribution:** Given a power budget per chip and considering the

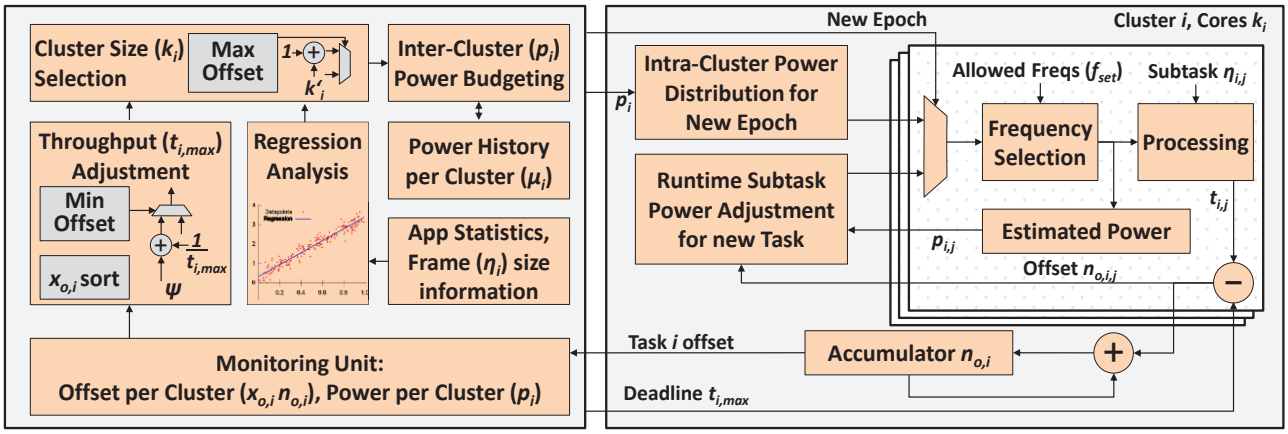


Figure 4-19: Details of the proposed multi-granularity power budgeting scheme. For simplicity, only one task is shown.

unpredictability in power demand by different tasks, a global cluster-level power distributer manages the power budget allocation to the clusters at a coarse-granularity, where the budget allocation depends upon cluster's information and the feedback error in power budgeting.

- **Intra-Cluster Power Budget Distribution:** An intra-cluster, local power distributer allocates the cluster's power budget to its individual cores at a finer-granularity. The intra-cluster power distribution depends upon the workload of the specific core. The cores can then adjust their operating frequencies to satisfy the upper limit on the power consumption of the cluster.

For this approach, a homogenous many-core chip with per-core DVFS and uniform tiling is considered. The global Inter cluster resource and power distribution takes into account (a) the execution history of the application, (b) the throughput requirement of the application and (c) the operating mode of the application. Further, this allocation is tuned at runtime after every epoch (a set of tasks like GOP, or time) based upon the tasks' requirements and the allocated budgets. Once the Inter cluster resources and powers have been distributed, the Intra-cluster budgeting commences and determines the best frequency ($f_{i,j}$) of each core under cluster power budget. Each cluster is responsible to process a task i ($\boldsymbol{\eta}_i$) within the time $t_{i,max}$. Each tasks consists of multiple subtasks ($\eta_{i,j}$). Each core in the cluster is responsible for processing a subtask. A task is deemed processed if all cores within the cluster finish processing their respective subtasks. The technical challenge is to determine the right number of cores and their frequencies within a cluster, in order to sustain the throughput requirement ($\boldsymbol{\eta}_i$ processing time < $t_{i,max}$) under the power budget (p_{tot}).

4.5.1 Hierarchical Resource Budgeting

The detailed operational flow of the proposed hierarchical resource budgeting is shown in Figure 4-19. After each epoch, the Inter-cluster resource and power distribution scheme is triggered to adjust resource allocation per cluster. The Intra-cluster power budgeting is executed to adjust the power of each core after every task ($\boldsymbol{\eta}_i$). Here, multiple tasks form a single epoch. This approach distributes power among a_{tot} concurrently executing multithreaded tasks/applications, competing for r_{tot} total cores and p_{tot} total power budget. An task i is constrained with user defined $t_{i,max}$ seconds for processing $\boldsymbol{\eta}_i$ with n_i total jobs. This corresponds to each core in the cluster i trying not to take more than $t_{i,max}$ seconds to process a subtask ($\eta_{i,j}$). The number of jobs within a subtask j of task i is denoted by $n_{i,j}$ The optimization problem solved in this work is given by:

$$\max\left\{\min_{\forall i \in \text{tasks}}\left\{1/(t_i)\right\}\right\}$$
$$s.t.$$
$$t_i = \max\{t_{i,j}\} \le t_{i,\max} \quad \forall \eta_{i,j} \in \boldsymbol{\eta}_i, \forall \text{ tasks} \tag{4-32}$$
$$\sum_{\forall \text{ tasks}} k_i \le r_{tot}, \qquad \sum_{\forall \text{ tasks}} p_i \le p_{tot}$$
$$f_{i,j} \in \{f_{\min}, \cdots, f_{\max}\} \ \forall \text{ cores}$$

The basic processing steps for allocation of compute resources and power budget are (also given in Algorithm 6):

1- Determining the size of a cluster *i* (the number of cores, k_i) that are allocated to a particular multithreaded application, in order to satisfy the service quality constraint (line 2). The cluster size is constrained by the available cores.

2- Performing cluster-level power budgeting such that the total power of the system does not exceed the TDP budget (line 3). Not all cores run at maximum frequency-voltage setting.

3- Fine-grained (Intra-cluster) power budget distribution among the cores within a cluster (lines 4-5) by only considering discrete set of frequencies of the cores.

4- Adjustment of power/frequency of individual cores within the cluster, depending upon the input data and power consumption (lines 10-12). The voltage of the cores is scaled accordingly.

Note that the overall management (selecting cluster-sizes, power distribution etc.) can be performed by OS or a specialized core within a cluster after a control period. In the following, each key component of the proposed approach will be explained with reference to Figure 4-19 and Algorithm 6. For ease of explanation, discussion will start from Intra-cluster power distribution and adjustment (step 3-4) and end with the selection of appropriate cluster size (step 1).

4.5.2 Intra-Cluster Power Distribution p_{ij}

Assume that the number of cores within the cluster *i* (k_i) and the power (p_i) allocated to the cluster (which process the task $\boldsymbol{\eta}_i$) are specified. p_i is distributed among the constituent cores of the cluster. Each core processes a single subtask *j* ($\eta_{i,j}$), i.e. $\boldsymbol{\eta}_i$ is divided into k_i subtasks. However, it is possible that the workload of each subtask differs from the rest. For example, in video applications, a video tile might have high motion content, which will result in larger workload than the rest (see Figure 3-5). This subtask can then become a critical and will hurt the throughput of the application. Therefore, the proposed Intra-cluster power budget distribution is based upon workload of the subtasks, by budgeting more power to critical subtasks.

For the first iteration of Intra-cluster power budget distribution (Fig. 4, lines 4-5), the j^{th} core of the cluster *i* is assigned power $p_{i,j}$ using the equations:

$$p_{i,j,Alloc} = \left(p_i + p_{i,d}\right) \times n_{i,j} / n_i$$
$$p_{i,j} = \begin{cases} p_{i,j,Alloc} & \text{if} \quad p_{i,j,Alloc} < p_{\max} \\ p_{\max} & \text{otherwise} \end{cases} \tag{4-33}$$
$$p_{i,d} = p_{i,j,Alloc} - p_{i,j}$$

This equation shows that each core is allocated power based upon the number of jobs within its respective subtask. Given the power of a core, frequency of the core ($f_{i,j}$) can be determined using approaches similar to

[3, 156]. The relationship between power and frequency of a core can be approximated with a linear equation ($\Delta p=\psi\Delta f$, where ψ is a design time parameter). However, since the frequency set (f_{set}) is quantized and a core cannot run at a power greater than p_{max}, therefore, only a particular frequency can be selected for the core. This particular frequency will actually consume a specific power, $p_{i,j}$. Therefore, the difference ($p_{i,d}$) between the allocated ($p_{i,j,Alloc}$) and consumed ($p_{i,j}$) powers is added for power allocated for the next core. Note that it is possible that $p_{i,j} < p_{i,j,Alloc}$ because the power consumed by a core j might be less than the actual power allocated to the core.

Once the power of each core and a particular frequency associated with it is determined, the application can start (Algorithm 6, lines 9-11). Due to varying workload of each $\eta_{i,j}$, it is possible that the time consumed of a subtask ($t_{i,j}$) differs from the rest and varies at runtime. For every subtask, a variable $n_{o,i,j}$ is used to determine the offset associated with the power/frequency of each core:

$$n_{o,i,j} = t_{i,j} - t_{i,\max} \tag{4-34}$$

Further explanation about $n_{o,i,j}$ can be found in Figure 4-20. As noted, a large positive value of $n_{o,i,j}$ denotes more power should have been allocated to the core and vice versa.

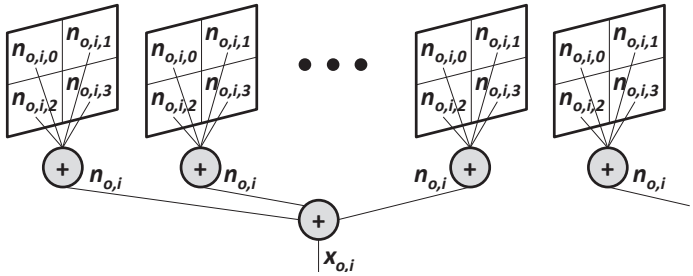


Figure 4-20: Concept of misprediction or offset used in this work. An example 4-core cluster is shown, where each core processes a single subtask and gives subtasks offset $n_{o,i,j}$.

Frequency Adaptation: If the workload characteristics of subtasks are changing or the system load varies due to parallel running applications at runtime, it becomes essential to transfer power among cores within a cluster. After a data frame is processed, $n_{o,i,j}$ for all threads of the application i are collected. The cores with larger $n_{o,i,j}$ require their frequencies to be lifted. However, since there is a power quota allocated to the application, a priority based scheme is utilized for increasing or reducing the power of the cores. This procedure is outlined in Algorithm 7. At first, the subtasks are sorted by their $n_{o,i,j}$ values (Algorithm 7, line 17). The algorithm iteratively checks if there are some subtasks with $n_{o,i,j} > 0$. This subtask is termed as T_{high} and this is the critical subtask of the application, which consumes the most time. In case such subtasks exist, the remaining subtasks are searched having $n_{o,i,j}$ lesser than the critical subtask and are called T_{low}. The algorithm tries to take power from the T_{low} and give that power to T_{high}. This corresponds to an increase the frequency of T_{high} while reducing the frequency of T_{low} by a single frequency step. By using downward frequency scaling for T_{low} and upward for T_{high}, if the power consumption of the cluster is lower than the allocated power to the cluster, the change in frequencies is accepted and the algorithm searches for other subtask pairs. This algorithm will only continue if $n_{o,i,j}$ of at least one of the remaining subtasks is greater than 0 and greater than $n_{o,i,j}$ of at least one other subtask.

4.5.3 Inter-Cluster Power Distribution p_i

If the cluster size (k_i) is already determined, the total TDP budget of the many-core system (p_{tot}, which is less than the sum of maximum power consumption of all the cores, thus determining the amount of Dark Silicon) is distributed among the a_{tot} concurrently executing tasks with every epoch (Algorithm 6, line 3). The Inter-cluster power distribution is a two-step process approach. In the first step, the power allocated to a cluster i (p_i) after every epoch is given by the following relation.

$$p_i = p_{tot} \times \frac{\left(k_i + \mu\left(p_i\right)\right)}{\left(\sum_{j=0}^{atot-1} k_j + \sum_{j=0}^{atot-1} \mu\left(p_j\right)\right)} \tag{4-35}$$

In this equation, the k_i dependent factor tries to budget more power to the cluster with larger number of cores. However, it is possible that the power of the cluster varies under workload demands. Thus, another factor $\mu(p_i)$ is used to account for the task i's power consumption history in the previous five epochs. In this equation, $\mu(p_i)$ denotes the expected (average) power of the i^{th} task for the previous epochs and acts as a feedback to the Inter-cluster power budgeter. Therefore, a task with high power consumption history is allotted more power. For the first epoch, average power of all tasks is zero, and hence each core of the many-core system gets an equal amount of power. Further, at any time instance, the summation of powers of all clusters will result in power $\leq p_{tot}$. Note that our Inter-cluster power distribution is different than the control based power adjustment approach discussed in [3], which requires a feedback loop and a tuning parameter.

Power Adjustment: In the second step of power distribution, the actual power of allocated to a cluster (p_i) is determined by using the offset information of all the task ($n_{o,i}$, see Figure 4-20). For each task i, $n_{o,i}$ is computed by:

$$n_{o,i} = \sum_{j=0}^{k_i-1} n_{o,i,j} \tag{4-36}$$

For the complete epoch, $n_{o,i}$ of each task is accumulated in $x_{o,i}$ as shown in Figure 4-20. If there is a task a_{max} with maximum $x_{o,i} > 0$ among all tasks, a part of the power from the task a_{min} with minimum $x_{o,i}$ is taken and given to this task. Mathematically:

$$\Delta p = p_{a\min} / \psi_1, \quad p_{a\max} = p_{a\max} + \Delta p, \quad p_{a\min} = p_{a\min} - \Delta p \tag{4-37}$$

Here, ψ_1 is a user defined parameter and determines the amount of power shifted from the low complexity task to the high complexity task.

Throughput Adjustment: If the maximum $x_{o,i}$ among all tasks is negative, it means that all tasks are meeting their initial throughput requirement. Therefore, it is possible to increase their throughput demand to speed up processing. In our case, $t_{i,max}$ for the next epoch is decreased using:

$$t_{i,\max} = \frac{t_{i,\max}}{\psi_2 t_{i,\max} + 1} \tag{4-38}$$

The user defined factor ψ_2 denotes the amount of change in $t_{i,max}$. A higher value of ψ_2 will reduce $t_{i,max}$ quicker and vice versa.

4.5.4 Selection of Cluster Size

Unlike [3, 154, 156], which only target the increase in average instructions executed per second of all single threaded applications, the proposed approach introduces a timing constraint (deadline, $t_{i,max}$) which the resource distribution scheme tries to meet for every task. The deadline can be specified for the whole run of the task, or, for processing individual tasks. The concept of task based deadline is a more realistic approach, especially for streaming and image, video and audio processing applications.

For selecting k_i, the proposed approach considers maximum allowable time ($t_{i,max}$) of the task i for processing a single $\boldsymbol{\eta}_i$ (Algorithm 6, line 2). In order to keep the processing time below $t_{i,max}$, only the available bright cores (k_{tot}) that can be distributed among different tasks are considered, such that all cores execute at frequency f at the startup time. Using this frequency, it is possible to estimate the total number of cores required for processing a subtask within $t_{i,max}$:

$$k_i^{'} \geq g\left(t_{i,\max}, \boldsymbol{\eta}_i, f\right) \tag{4-39}$$

This equation relates $t_{i,max}$, the characteristics of d_i and the minimum number of expected cores (k'_i) which will sustain the workload of the application i. This equation can be generated via offline statistics or regression analysis [44, 241]. Or, this equation can also be derived using the same approach as outline in Equation (4-10). For example, using a core i7 at 3.4GHz (fixed) frequency, the time consumed for HEVC encoding, for processing a video frame/task with n_{frm} jobs and k_i subtasks is given by:

$$t_{i,3.4G} = -1.844 + 0.01827 n_{frm} + 0.5441 n_{frm} \Big/ k_i \tag{4-40}$$

This equation can be scaled to account for any other frequency f via:

$$t_{i,f} = \left\lfloor \frac{3.4 \times 10^9}{f} \right\rfloor t_{i,3.4G} \tag{4-41}$$

Typically, in a performance-constrained application, the average-case required number of cores for the given deadline can be derived at the setup stage (offline), which can then be updated at runtime (online) depending upon the input data frame properties (see Section 4.2.5). Since the proposed approach adapts $t_{i,max}$ (see Equation (4-38)), therefore, k'_i also adapts at runtime.

In addition, at runtime, the workload characteristics of the tasks might change, therefore, our proposed approach also adapts the size of the cluster. Specifically, for a task i with $x_{o,i} > 0$, k'_i is incremented by one. This way, the proposed approach tries to allocate more resources to the task requiring more computations.

The number of cores (the cluster size) which is actually allocated to the task i is derived by the following formula:

$$k_i = \left\lfloor k_i^{'} \times k_{tot} \Big/ \sum_{j=0}^{atot-1} k_j^{'} \right\rfloor \tag{4-42}$$

And afterwards:

$$k_i = \begin{cases} k_i^{'} & , k_i > k_i^{'} \\ k_i & , \text{otherwise} \end{cases} \tag{4-43}$$

From Equation (4-42), even if summation of k'_i is greater than k_{tot} (see discussion above regarding $x_{i,o}$), summation of k_i will still be in bounds ($\leq k_{tot}$). This also suggests that if the collective workload of all the concurrently running tasks exceeds the global power budget and resources, the system will still continue to run at a degraded quality (time per frame > $t_{i,max}$). Equation (4-43) shows that it is possible that not all of the k_{tot} bright cores are utilized and there are unused cores which can be kept dark.

4.6 Computational Offloading

Owing to user constraints like duration of processing, it might not be possible for a battery-driven and constrained video system (with limited number of cores, cache, power/frequency etc.) to process assigned tasks while maintaining the required throughput requirement and unnoticeable video quality degradation. On the other hand, the number of tasks and the required throughput requirements might make it infeasible to implement the video processing applications, especially on small systems. For example, a platform can only process a finite set of jobs within a second (n_{sec}) and thus, the workload with higher requirements cannot be scheduled on this platform. Moreover, the required frequency might not be supportable by the hardware, and no DVFS is available, i.e., the compute configuration is fixed and unchangeable. Thus, resource budgeting proposed above in these scenarios is nontrivial and infeasible. In such cases, offloading subtasks to high-end processing nodes and utilizing energy-efficient application-configurations can enable the application to meet its throughput

demands on constraint systems.

Offloading of subtasks pertaining to multimedia, vision, graphics, gaming, text processing etc., from mobile systems (like smartphones, robots, embedded boards) to high end servers, must consider the communication layers as well as the application layer properties. The task is to reduce the system energy consumption. As pointed out in Sections 2.2.3 and 3.2.4, Hybrid Distributed Video Coding (HDVC) provides a solution to utilize low complexity and resource constrained devices in the video coding landscape. A use-case scenario of HDVC is outlined in the Figure 4-21. For HDVC, both the encoder and decoder are battery-driven, constrained devices, and, in contrast to PVC, the decoder has a higher complexity/power than the encoder. In such cases, the encoder can offload some of the subtasks (workload of video processing) to the decoder. Such a paradigm has possible applications for video transmission by wireless sensor nodes or IOT devices to a high-end server, battery-driven security cameras encoding and transmitting the compressed video for storage/ analysis to the high complexity processing node, video transmission by autonomous batter-driven robots (e.g., drones) etc. However, efficient utilization of resource and energy under dynamically varying scenarios requires an adaptive energy quota distribution at the encoder and decoder. The goal should be to determine the processing effort at both ends of the system such that the overall energy of the system (computation and transmission) is minimized, while satisfying the user constraints, like encoding duration or FPS.

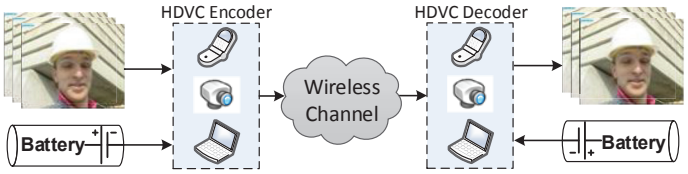


Figure 4-21: Video capture and wireless transmission scenario with resource constrained video encoder(s) and decoder(s).

To address the above-mentioned challenges, this thesis proposes hierarchical control approach for HDVC that performs video content-aware adaptive energy quota distribution and control at multiple hierarchical levels (i.e., among GOWs, frames within a GOW, and blocks within frames, see Figure 2-13) for both HDVC encoder and decoder. This approach reduces the overall energy consumption by jointly considering for the computation and transmission energy under scenarios of dynamically varying energy levels and user constraints. The proposed approach accounts for video content properties (i.e., texture and motion) for selecting blocks from image Regions of Interest (ROIs) to perform selective processing per block, while efficiently utilizing the allocated energy quotas. Figure 4-22 shows the energy quota distribution mechanism. Here, a task actually represents a video frame to process, while a group of consecutive tasks to be processed are equivalent to processing a GOW. Each video frame consist of blocks, and each block of the video frame is an independent subtask. At both the encoder and decoder, Motion Estimation (ME) is performed for each subtask. Figure 4-23 shows the proposed HDVC system with the novel contributions.

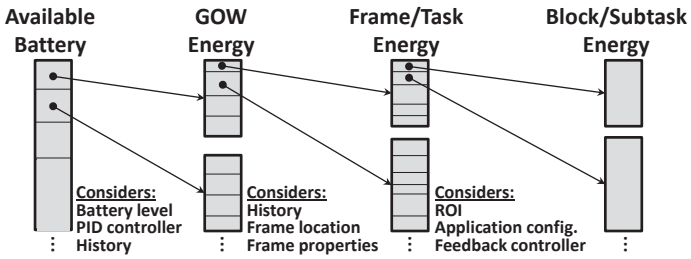


Figure 4-22: Energy quota distribution at different hierarchical levels

In summary, the proposed approach employs the following steps:

GOW-Level Energy Quota Distribution and Control: First, user defined constraints of total processing duration and available energy from the battery level are used to determine the potential processing duration. If the potential processing duration is less than the user-defined constraint, the processing rate and GOW size are re-adjusted to fulfill the user constraints in the available battery level. This processing duration along with the offline subtask energy analysis is used to derive the initial target energy quota for each GOW. Since the consumed energy of a GOW may differ from the allocated energy quota, the target of the upcoming GOW is

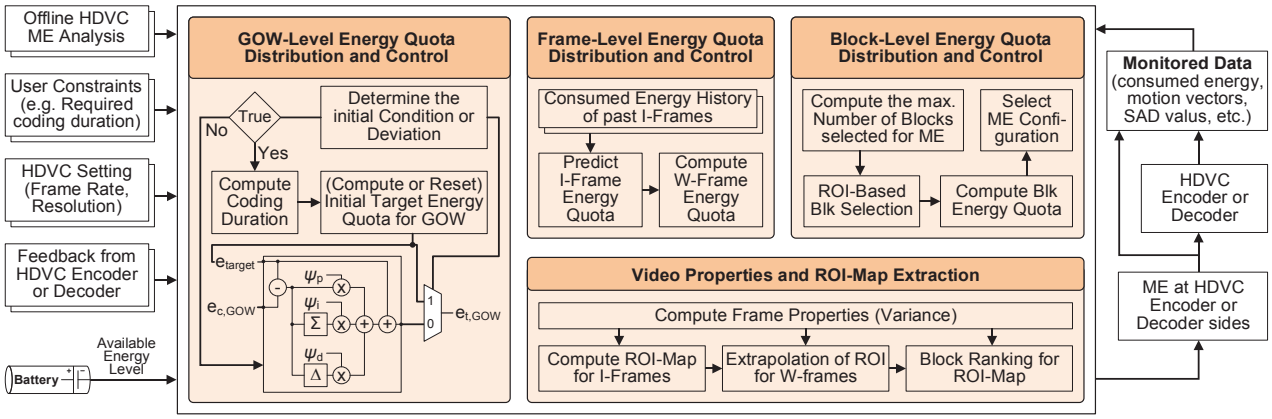


Figure 4-23: Operational flow of the proposed hierarchical energy budgeting for HDVC. Here, Blk: Block.

recomputed using a feedback control mechanism. This approach employs a PID controller to control the target of the upcoming GOWs. In case of a deviation (i.e., a significant change between two I-frames, for instance, due to a scene cut), the initial target for a GOW is reset for the controller. The GOW-level energy quota is forwarded to the frame-level energy quota distributor.

Frame-Level Energy Quota Distribution and Control: At the frame level, the energy quota is distributed among the I-frame and *z* W-frames. The energy quota of the I-frame is computed by performing a history-based prediction using the energy consumption of previous I-frames. Afterwards, the energy quota of *z* W-frames is obtained. This quota is distributed among *z* W-frames considering the temporal distance of a given W-frame from the preceding I-frame and relative variance difference of two I-frames at the borders of the current GOW. The temporal distance based energy quota distribution accommodates the changing temporal correlation inside a given GOW due to the longer temporal distance.

ROI Identification and Extrapolation: In order to intelligently distribute the frame-level energy quota among different blocks of a frame, the proposed employs a ROI-based block selection and ROI-driven energy quota distribution. The ROI is identified for I-frames considering the motion drift of a given block with respect to its neighboring blocks. Since blocks in the ROI have high spatial and temporal correlation, the proposed approach selects the ROI blocks with high rank value, which is quantified as the motion vector drift (see Equation (3-1)). It is based on the analysis of Section 3.2.4.1 that decoder can generate the estimates for W-frames for highly-correlated blocks. Typically, blocks at the object boundaries are selected by the proposed approach. An ROI map contains the ROI blocks sorted with respect to their rank values in a descending order (starting with the highest rank value). This facilitates generating a high-quality SI at the decoder side that leads to reduced energy at the encoder side. Note that the proposed approach does not waste energy for the homogeneous and slow-moving blocks at the HDVC encoder, because decoder can generate high quality estimates of these blocks with a little effort. Instead, the energy is spent on complex block with high texture and fast motion, such that the SI generation at the decoder side can be improved that leads to less transmission of bits and consequently low transmission energy.

Block-Level Energy Quota Distribution and Control: First the maximum number of selected blocks is computed based on the available frame-level energy quota and best case application configuration (i.e., ME configuration with the least energy). Since in a resource-constrained scenario, sufficient processing power may not be available to meet the throughput constraints, the number of selected blocks is readjusted. For the number of selected blocks, ROI blocks from the ROI map are extracted starting with the highest rank value. The energy quota of each block is computed and controlled in a feedback manner. Depending upon the allocated energy quota, an appropriate application configuration (i.e., ME configuration) is selected and the block is processed.

The consumed energy is monitored. This consumed energy may differ from the allocated energy quota. Therefore, the target of the subsequent block is adjusted by back-propagating the error between the consumed and target block energy in a weighted fashion. The total consumed energy is fed back for controlling the frame-level and GOW-level energy quota distribution.

At the encoder side, the block-level energy quota is used to determine the ME configuration of the selected blocks. At the decoder side, the allocated energy quota is used for determining the ME/SI configuration for the blocks that are not selected for the encoder-side block. In the following, details about the above mentioned steps are given with reference to Figure 4-23.

4.6.1 GOW-Level Energy Quota Distribution and Control

This is the first step of energy quota distribution approach and is given in the left part of Figure 4-23. Algorithm 8 illustrates the pseudo-code for computing the GOW-level target energy quota for both encoder and decoder sides. Each GOW has z W-frames and one I-frame. When HDVC is initialized, the encoder and decoder side target GOW energy quotas ($e_{t,GOW,(enc,dec)}$) are computed considering the available battery levels ($e_{batt,(enc,dec)}$) and user constraints of total encoding duration (t_d). First the potential encoding or decoding durations are computed considering the available battery levels (line 3). These durations are computed using the formula:

$$t_{d,i} = \left(e_{batt,i} \times (z+1)\right) \Big/ \left((e_{IF,avg} + z \times e_{WF,avg}) \times f_p\right) \qquad i \in \{enc, dec\} \tag{4-44}$$

Here, $e_{IF,avg}$ and $e_{WF,avg}$ are the average energy consumption of I-frames and W-frames respectively, that are obtained using an offline analysis for various test video sequences [242]. The minimum of the two durations is selected to determine the overall potential processing duration (line 4).

If the potential processing duration is less than the user-defined processing duration constraint (t_d), the required duration is set (line 5) and the frame rate (f_p) and GOW size (z+1) are readjusted to accommodate the required coding duration in the available battery levels (lines 7-8). Afterwards, the target GOW energy quotas are computed (lines 9-10). Since the consumed energy of a GOW may differ from the target GOW energy quota, our approach adapts the target GOW energy quotas in a feedback control mechanism (lines 13-16). The error between the consumed energy and target energy quota is computed after for each GOW. Our approach employs a simple PID controller to control the target energy quota of the next GOW. ψ_p, ψ_i and ψ_d are the proportional, integral, and derivative gains. ψ_p reduces the error, ψ_i eradicates the steady error effects, and ψ_d ameliorates the stability. These gains are computed using the Ziegler-Nichols Method [243] using the settings given in the following equation (K_u=*0.8,* T_u=*2*):

$$\psi_p = 0.6K_u, \quad \psi_i = 0.5T_u, \quad \psi_d = 0.125T_u \tag{4-45}$$

4.6.2 Frame-Level Energy Quota Distribution and Control

Encoder-Side Energy Quota Distribution: The GOW-level energy quota is distributed among the I-frames ($e_{t,IF}$) and the W-frames ($e_{t,WF}$) depending upon size of GOW (i.e., z+1). The energy quota at the encoder side is distributed to the I-frame for H.264/AVC Intra-encoding and ROI identification ($e_{t,IF} = e_{IF} + e_{ROI}$), and to the W-frames for selective subtask processing (ME of blocks), transmission, channel coding, and ROI extrapolation: $e_{t,WF} = e_{ME} + e_{Tx} + e_{cc} + e_{ROI,ex}$.

The energy quota for the I-frame in the GOW ($e_{t,IF}$) is allocated based on the consumed energy of the previous I-frames ($e_{c,IF}$) using:

$$e_{t,IF} = \frac{e_{c,IF(i-1)} + median(e_{c,IF(i-2)}, \cdots)}{2} \tag{4-46}$$

In order to avoid noise in the energy consumption, median operator is used for previous I-frames in the history, while the immediately previous I-frame's energy is used to compensate for the sudden error effects.

The target energy quota for z W-frames ($e_{t,WF} = e_{t,GOW} - e_{t,IF}$) is computed and distributed among individual W-frames (indexed by j) considering their temporal distance from the I-frame of the GOW:

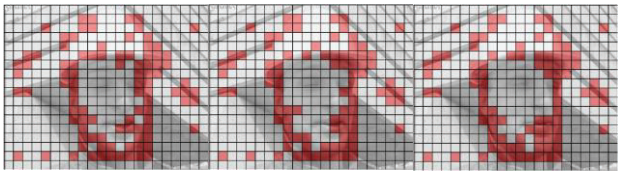
$$\begin{aligned} e_{t,WF,j} &= \left(\frac{(1-\upsilon)}{z-(j-1)} + \frac{\upsilon}{\sum_{l=1}^{z-(j-1)} l} \right) \times \left(e_{t,WF} - \sum_{n=1}^{j-1} e_{c,WF,n} \right) \\ \upsilon &= \left(1 - \min(\upsilon_{IFi}, \upsilon_{IFi-1}) \Big/ \max(\upsilon_{IFi}, \upsilon_{IFi-1})\right) \end{aligned} \tag{4-47}$$

The amount of temporal correlation for a given W-frame with respect to its preceding I-frame decreases depending upon its temporal distance. For offloading scenarios involving video coding, this means that it is difficult to efficiently encode the video frames farther away from the preceding I-frame. This is due to the fact that objects in farther video frames have more temporal distance and may not be captured by ME under a given search range and energy quota constraints. Therefore, extensive search is required for farther W-frames, and less motion search may be sufficient for the closer W-frames. To facilitate this, the proposed approach adaptively distributes the $e_{t,WF}$ energy quota among W-frames depending upon their temporal distance from the preceding I-frame and variance difference (υ) between the bordering I-frames. The variance difference υ of two consecutive I-frames hints towards the quality of the temporal correlation. Depending upon the variance difference, a part of $e_{t,WF}$ (first addend without υ in Equation (4-47)) is equally distributed to keep the fairness of allocation, in order to compensate for the energy consumption other than ME computations (like e_{Tx}, e_{cc}, and $e_{ROI,ex}$). However, the other part (second addend with υ in Equation (4-47)) is distributed based on the temporal distance to compensate for the temporal correlation changes. In summary, the W-frames farther away from the I-frame get more energy quota. Therefore, the application configuration of subtasks/blocks in farther W-frames can be set to exercise high output quality processing.

After processing each W-frame, the energy quota of the next W-frame is controlled depending upon the consumed energy of the previous W-frames ($e_{c,WF}$). For instance, for $z = 4$, the energy quota of WF_1, WF_2, WF_3, and WF_4 can be computed as follows:

$$\begin{aligned} e_{t,WF,1} &= \left((1-\upsilon)/4 + \upsilon/10\right) \times \left(e_{t,WF}\right) \\ e_{t,WF,2} &= \left((1-\upsilon)/3 + \upsilon/6\right) \times \left(e_{t,WF} - e_{c,WF,1}\right) \\ e_{t,WF,3} &= \left((1-\upsilon)/2 + \upsilon/3\right) \times \left(e_{t,WF} - \left(e_{c,WF,1} + e_{c,WF,2}\right)\right) \\ e_{t,WF,4} &= \left(e_{t,WF} - \left(e_{c,WF,1} + e_{c,WF,2} + e_{c,WF,3}\right)\right) \end{aligned} \tag{4-48}$$

Decoder-Side Energy Quota Distribution: For offloading scenarios, the decoder needs to estimate the missing information using the correlation between the missing information and the neighboring frames. For video encoding, our approach distributes the energy quota mainly among the key I-frames in order to perform the bi-directional ME in order to interpolate the frames and form SI. For i^{th} GOW, the W-frames are allocated an average energy quota based on the history considering the energy consumption for SI generation/interpolation and Slepian-Wolf decoder. The size of history is 'l' previous GOWs each having z W-frames. This is given by:



MVD	Accumulator	List
0	3	(16,16), (16,32), (48,208)
1	2	(32, 48), (96, 128)
2	4	(96,96),(352, 64),(208, 80),(128, 154)

Figure 4-24: (top) ROI identification and then extrapolation for two W-frames, (bottom) example hash-table to store ROI map

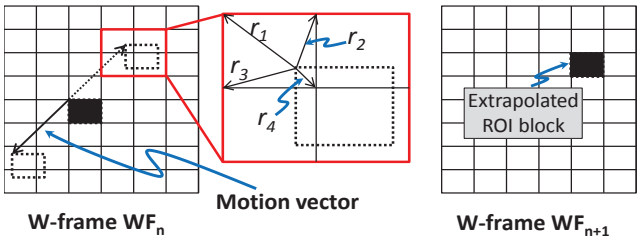


Figure 4-25: ROI blocks extrapolation between two consecutive W-frames (the dark boxes present ROI blocks)

$$e_{t,WF,i} = z \times \left(\underset{\substack{\forall j \in [1 \ldots l] \\ \forall n \in [1 \ldots z]}}{avg} \left(e_{c,WF,(i-(j,n))} \right) \right); \tag{4-49}$$

The remaining quota of the i^{th} GOW is allocated to the decoding of I-frames ($e_{IF,dec}$) and bi-directional ME (e_{ME}).

$$e_{t,IF} = e_{t,GOW} - e_{t,WF}; \quad e_{ME} = \left(e_{t,IF} - e_{IF,dec} \right) / 2 \tag{4-50}$$

4.6.3 ROI Identification and Extrapolation

Although selecting the appropriate application-configuration at the encoder-side is important for energy-efficiency, selecting the right blocks to apply this configuration is also important, because a block may differ significantly from the neighboring blocks depending upon its texture and motion. As previously discussed in Section 3.2.4.1, encoder side processing of selective subtasks (ME of blocks) may lead to significant energy improvement, due to improved SI and reduction in the channel coded parity bits. Typically, the decoder cannot perform a good estimation for the blocks with high motion and/or variance, especially at the object boundaries. Therefore, it is of key importance in HDVC to spend energy at the encoder side to perform ME of the ROI blocks. Note that if the boundary of the moving region is extracted, the HDVC decoder can exploit the correlation of blocks belonging to the same object to predict the other blocks of the moving objects/regions. Therefore, the key focus of the proposed approach is to accurately extract the ROI-blocks at the object boundaries as given in Figure 4-24. These blocks are ranked according to their *mvd* value (see Equation (3-1)) and stored in a hash table, like the one given in Figure 4-24. Thus, blocks with high *mvd* have a higher motion difference with their surrounding blocks, and hence form the boundary of moving object with high probability.

This approach exhibits an integrated motion-based algorithm for ROI identification and ROI-block selection. Further, to reduce the ROI overhead, the I-frame ROI blocks are extrapolated for W-frames. The extrapolation is considered by fitting the projected ROI blocks location to the nearest block boundary. The ROI map is generated only at I-frames and extrapolated as shown in Figure 4-25. For the current W-frame, the ME of a ROI block (shown in black) is performed and the final motion vector is achieved. The same block in the next frame is obtained by reflecting the motion vector and then selecting the nearest neighboring block as shown.

4.6.4 Block-Level Energy Quota Distribution and Control

Once the energy quota for the task/frame is specified, it can be distributed among the subtasks/blocks. The underlying concept is that energy is first distributed among the blocks of ROI and then non-ROI blocks, and the energy quota allocated to a block determines the application configuration used for processing a block. Algorithm 9 shows the ROI-driven block selection and energy distribution approach. Note that this quota

distribution is only applied to the blocks of W-frames. The input is the frame-level energy quota of the W-frame and the ROI map sorted with respect to the rank values of the blocks in a descending order. First, the maximum number of selected blocks ($n_{ME,max}$) that need to be considered for encoder-side ME is computed using the frame-level energy quota and the energy required for transmission, channel coding, and ROI extrapolation (line 1). Note that the transmission energy is a function of the transmission distance d [244]. In several cases, the underlying platform may not be able to provide the required throughput due to limited computational resources and/or limited processing power. Therefore, the maximum number of blocks per frame (n_{frm}) that can be processed by the underlying platform needs to be computed (line 2) and the maximum number of blocks which will undergo processing at the encoder-side (n_i) is determined as the minimum of $n_{ME,max}$ and n_{frm} (line 3). Afterwards, n_{ROI} blocks are extracted from the ROI map starting from the highest rank value (line 4). If n_{ROI} is greater than the number of blocks in the ROI, additional blocks are extracted from the non-ROI map (line 7-8).

The target energy quota for one block is computed depending upon the number of selected blocks (line 9). Afterwards, all the blocks in the selected block list are processed for ME starting from the highest rank blocks first (lines 11-17). An appropriate application-configuration (ME configuration) is selected considering various ME configurations that provide energy vs. quality tradeoffs (line 12). Multiple ME configurations can be formulated depending upon the search window size, search pattern types, termination rules, initial search point prediction and others as given in [67]. After the ME is done for the block, the consumed energy (e_{MB}) is monitored (line 13). The consumed energy can be different from the target energy or the average energy of the selected ME configuration. Therefore, we adjust the energy quota of the block and the average energy value of the ME configuration in order to have a better ME configuration selection for the subsequent blocks. Depending upon the error between the target and the consumed energies (line 15), the target of the next block is readjusted by back-propagating the energy error (line 16). The weighting factor ψ controls the strength of back-propagation. The total consumed energy of the W-frame is returned to control the frame-level energy quota (line 14-18).

Note that the decoder-side block-level energy distribution and ranking are very similar to that at the encoder side, except that the energy quota for the I-frame includes the energy for forward and backward ME.

4.6.5 Evaluation of the Offloading Approach

For evaluating the proposed offload approach, the proposed control approach is implemented on a 90nm technology node and the area and energy results are reported in Table 4-4. The implementation includes one divider and three multipliers, which are shared among the different calculation steps. For the GOW-, frame- and block-level calculations, 4 cycles/23 cycles (depending on s_i or s_d signals in Algorithm 8), $2+2z+4$ cycles and $6+3n_{ROI}$ cycles are required, respectively (see parameters in Algorithm 9). Table 4-4 shows that the energy overhead is insignificant compared to the savings of our approach. Moreover, the ROI extrapolation requires only 2 additions and a search for minimum magnitude vector within 4 vectors. This is a negligible overhead compared to the encoding process and our energy savings. Further, an H.264/AVC encoder is considered for encoding the I-frames.

Table 4-4: Performance, area and energy overhead of our proposed approach (z=4, n_{ROI}=120) for 90nm technology

	Latency [Cycles]	Area GE	Energy [nJ]
GOW-Level	5/23	11,103	18.4
Frame-Level	2+12	8,327	11.1
MB-Level	6+360	14,238	292.2
Total	1501	62,638	1211.2

Video Quality: Figure 4-26 illustrate the observations for video quality when using our approach. Figure 4-26 shows that our approach tends to increase the relative quality if the value of z increases, where z is the number of W-frames between two I-frames. This observation is shown for "Foreman" sequence encoded with Quantization Parameter (QP) = 18 (high video quality) and QP = 36 (low video quality) for the I-frames. The later W-frames in the GOW are usually of lower quality compared to the earlier W-frames because of the

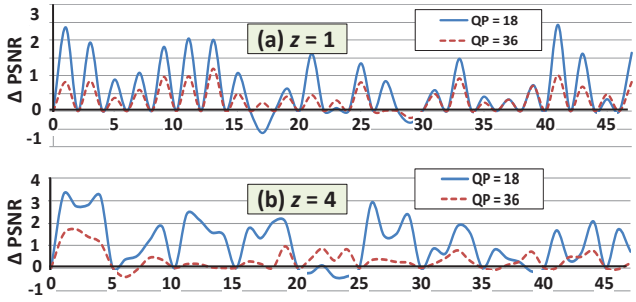


Figure 4-26: Difference in PSNR for the propose approach and that of [4] for different quantization levels

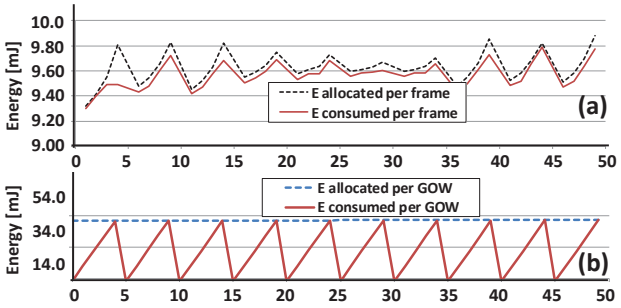


Figure 4-27: Proposed and consumed energy per (a) frame and (b) GOW

increased temporal decorrelation with the I-frames. Since our approach allocates these later W-frames a higher energy quota, a better estimate (or match) of the current block in I-frames can be found. This leads to a quality increase even for these frames, which will not be possible when allocating energy without consideration of temporal distance, i.e., equally allocating energy to all W-frames.

Energy Control: Figure 4-27 shows the energy quota adaptation at the (a) frame-level; and (b) GOW-level. It shows that the proposed approach keeps the energy quota distribution close to the target energy; thus leading to a reduced controller error (less than 1% for both frame- and GOW-level allocation). Figure 4-27 (a) shows that over the period, the controller error (i.e., difference between the target and consumed energy) is reducing. This is due to the adaptation of the controller output. Note that in Figure 4-27 (a), results for only W-frames are shown, because the variance in the energy consumption for I-frames insignificant and are adapted using a different equation (Equation (4-46)). An interesting observation in Figure 4-27 (b) is that the error between the target and consumed energy at the GOW-level is almost insignificant. This is due to the hierarchical control of the target energy quota which minimizes the risk of target energy quota violations.

Chapter 5 Video System Hardware Layer

The software layer approaches for computation- and power-efficient video processing system given in Chapter 4 do not require any custom hardware. However, custom hardware architectures for video processing systems are in wide use because they can result in high throughput, and high complexity and power reduction potential. This chapter outlines some of the novel hardware architectural enhancements and custom accelerators for highly efficient video processing systems. Efficient I/O and inter-node communications for video processing system is discussed in detail. Hardware architectures of the systems and accelerators, specifically pertaining to H.264/AVC and HEVC encoders, is given. Furthermore, the hardware accelerator allocation or workload management (whereby the accelerator provides its services to multiple nodes) is also discussed, which can be useful in shared hardware accelerator paradigms. Targeting the memory subsystem, power-efficient hybrid memory architectures and SRAM aging mitigation are also presented.

5.1 Custom Video Processing Architectures

The approaches outlined in Chapter 4 can be employed on a given multi-/many-core system with little or no modification. However, if a custom platform is implemented, with embedded soft-cores (cores that run software), and hardware coprocessors or accelerators, an architectural support is required to achieve efficient communication mechanisms among the computing nodes. A communication scheme must be adequate enough to fulfill the computation requirements, as well as it must not bloat the performance of the system.

In case of heterogeneous many-core systems, with in-core, tightly or loosely coupled hardware accelerators (see Figure 2-18), soft-cores with external memory and I/O ports, the data input/output to/from the system can become challenging. Specifically for video encoders, the data from the camera can be analog, in non-standard compliant format (e.g., instead of YCbCr 4:2:0 required by the HEVC encoder, the camera provides RGB), may require clipping/chroma resampling etc. To address these issues, this section will present an analysis of video memory, and an architecture to support reading/writing video samples to and from the video processing system is discussed. Afterwards, the discussion about data communication is generalized and extended to multi-/many-core heterogeneous systems.

5.1.1 Memory Analysis and Video Input

As seen from Figure 2-1, blocks of video needs to be fetched from the external memory or camera and fed to the video processing system. This communication must consider the bandwidth between the video generating source and the video processing system. A 16-bit DDR3 on Altera's DK DEV 2AGX260N FPGA development kit, running at 300MHz can theoretically support a maximum bandwidth of 16×300M×2 = 8.9Gbps. Taking a conservative memory efficiency of 70% for a typical DRAM [245], this bandwidth reduces to 6.23Gbps. From the discussion given in Equation (2-3), a system processing FullHD frames (1920×1080 pixels) would require ~0.58Gbps to read the video frame at frame-rate of f_p=25. For video encoders, this frame also needs to be written back to the external memory, and therefore, the total bandwidth requirement becomes ~1.16Gbps. However, if Inter encoding is used, in that case, additional reference frame(s) must also be read from the external memory. And from our discussion about Equation (2-6), a reference frame must be read at least r_f times (usually, $r_f \geq 3$) for processing a single frame. With only one reference frame and r_f=3, the bandwidth requirement swells to ~1.74Gbps. To generalize, the bandwidth requirement for 8-bit per sample, YCbCr 4:2:0 video encoder, with n_r number of reference frames, can be written as:

$$w \times h \times 8 \times f_p \times (3 + r_p n_r) \tag{5-1}$$

However, it is possible that more than one video is processed concurrently by the video processing system.

Such a paradigm is usually encountered in multicasting [74, 75] or multiview, 3D video processing [119, 241]. Assume that n_v FullHD views/videos are processed in parallel by the video application. For n_v=4, the total external memory bandwidth required for Intra encoding of is ~4.6Gbps which can be supported by the FPGA development kit discussed above. Additionally, note that external memory data transfer results in high-energy dissipation. As seen in [169, 64], ~40% of the total system power is consumed by the external I/O and the total bus power dissipation is directly proportional to the bus toggles [232]. Thus, increasing n_v not only increases the external memory bus contention but also results in a larger energy consumption.

5.1.2 Video Preprocessing

Video samples provided by the cameras may be filtered before placing these samples on the external memory. Usually, a Video Input Pipeline (VIP) is required to preprocess the samples, making them compliant to encoder's specification. For multicasting with n_v views, n_v individual VIPs and n_v base address in the external memory (to store the video frames) are required.

The streaming video data from a video camera is fed to a VIP. The preprocessing pipeline consists of a clocked video input sampler, a video frame clipper, a color plane-sequencer and an optional deinterlacer, as shown in Figure 5-1. The video input sampler is used to synchronize the byte-serial data stream from the camera by handling clock-domain crossings. Clipper adjusts the resolution of the input video according to the specifications of the encoder. Plane-sequencer converts a serial video stream (with one luminance and two chrominance components) into a parallel stream for parallel processing of luminance and chrominance components. The deinterlacer is used if the input is from an analog video source. If the chrominance sampling does not match the encoder's specifications, an additional chroma resampler (e.g., 4:2:2 to 4:2:0) is used.

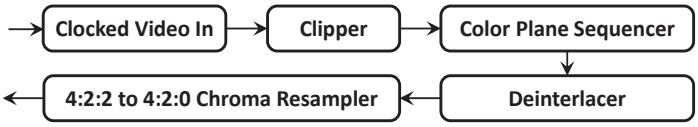


Figure 5-1: Video Input Pipeline (VIP)

5.1.3 DDR Video Write-Master

This module is shown in Figure 5-2. Streaming YCbCr 4:2:0 pixels from VIP are routed to a 4:2:0 pixel FIFO. A packet detection circuit detects the arrival of video samples and generates write enable signals to the pixel FIFO. These pixels are forwarded to shift-registers, where every luminance and chrominance component has a separate shift-register. The luminance shift-register is a 128-bits wide and chrominance shift-register is 64-bits wide. The write master controller determines when to push the data from the shift-registers to the external memory. Depending upon the size of the video, the Address Generating Unit (AGU) determines the writing address. The write master controller also directs the AGU by selecting appropriate address in luminance address space or chrominance address spaces in the DDR3 memory. The buffer controller provides the information whether a Cb or Cr address space needs to be written. The software configures the AGU to assign starting address of the frame.

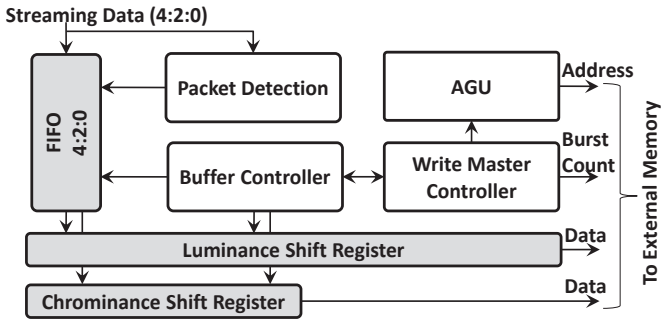


Figure 5-2: Frame writer hardware

The current frame data in the external memory is stored in a triple-buffering order (called the ring-buffer). This ordering facilitates frame droppings, because if the video system gets slower or stalls (due to DDR

bandwidth or Ethernet constraints), frames can be dropped. Only the current frame under process is retained and the last-written current frame in the buffer is updated, hence the name.

Moreover, to optionally write back the reconstructed block, an external memory write master is used. For H.264/AVC and HEVC Intra-encoding, this write-back to the external memory is not required. However, for Inter-encoding, reconstruction write-back is required. The memory write master is also configured via software.

5.1.4 Heterogeneous Computing Platform

As discussed earlier, for heterogeneous nodes, a communication mechanism must be employed whereby the master node can communicate with secondary nodes. This mechanism must be efficient and should result in minimal resource and communication overhead. For the proposed load balancing approaches presented in this thesis, one requires a synchronization and load distribution mechanism, possibly on accelerator based multi-/many-core heterogeneous system, where multiple heterogeneous cores process the input data together with hardware accelerators. The architectural diagram of the proposed system is shown in Figure 5-3. The cores can be architecturally different, and these cores can also have in-core hardware accelerators (in addition to the loosely-coupled hardware accelerators). The in-core accelerators can be accessed via Custom Instructions (CIs), e.g., as in Nios-II processor [185]. Thus, these cores consume different amount of cycles to process the same job. A master node is used to distribute the jobs among the other cores and accelerators. This node is also responsible for setting the frequency of all the other nodes in the heterogeneous system (according to the algorithm given in Algorithm 5). Processing jobs can be assigned to the loosely coupled hardware accelerator block, in which multiple accelerators pertaining to different functionalities can reside. All the nodes (including both the soft-cores and the custom hardware accelerators) are connected with the external memory via an external memory controller.

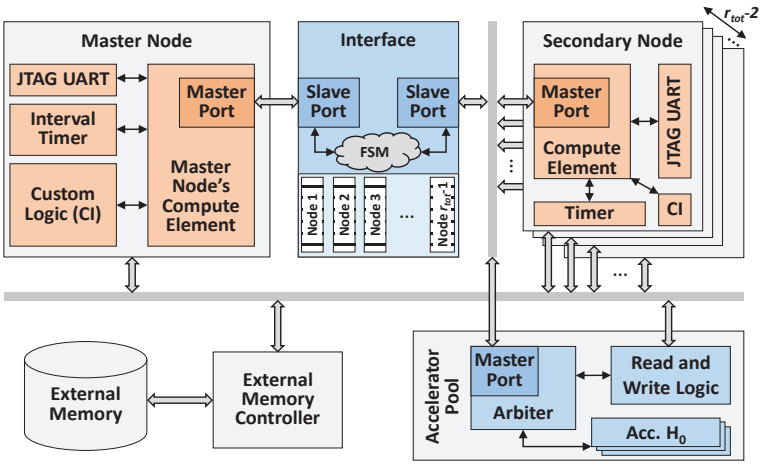


Figure 5-3: Heterogeneous system with custom hardware components

For this architecture, the data is processed as frames, which resides in the external memory. Each frame (can also be called as a task) is divided into constituent tiles (or subtasks), and the tiles can be processed in parallel by the nodes. The subtasks contain multiple jobs and each job contains a data block to be processed. Thus, the master node has information about the number of jobs within a task, and determines the number of jobs within a subtask for each secondary node. Further, the master node is also fed information about the deadlines, and the compute and power profiles of the secondary nodes.

The custom interface for communication among the nodes consists of a register file, which is filled by the nodes to exchange information. The master node writes information to specific addresses within the register file, associated with a specific secondary node. For example, the master node writes the start and end addresses of the data blocks in the external memory for a specific secondary node (corresponding to jobs associated with the data blocks processed by the secondary node). This is done by the master node for all the secondary nodes. A secondary node sends read request for its associated registers to a bridge controller (not shown), which aligns the read requests from all the secondary nodes. If the master node has allocated a valid address and number of jobs to process, the node starts fetching this data from the external memory and starts processing. Once all the

jobs have been processed, the secondary nodes write to the appropriate status registers in the custom interface. The master node, when receives this "jobs done" signal from all nodes, signals end of frame and gathers statistics. Afterwards, the master node sends new memory addresses to the secondary nodes, for the new frame.

5.2 Accelerator Allocation and Scheduling

This section provides details about sharing a hardware architecture among different compute nodes, and concurrently processing multiple video streams.

5.2.1 Accelerator Sharing on Multi-/Many-Core

One of the major technological challenges discussed in Section 2.3.1 is Dark Silicon. A method to solve this issue is to introduce highly efficient, custom hardware accelerators, which while running at lower frequencies (and hence resulting in lower spatial temperatures) can meet the throughput requirements. The hardware accelerators can be coupled to the compute nodes as shown in Figure 5-3. However, efficient allocation of the shared, loosely-coupled hardware accelerator to multiple threads/applications requires properly weighing the computational capabilities of the nodes and their available frequencies.

In this section, the following problem is addressed: How to allocate the shared hardware accelerator among the competing cores, such that the hardware is fully utilized, all application deadlines are met and the power consumption of the complete system is minimized under a specific set of running frequencies? For this purpose, an adaptive accelerator allocation scheduling approach is proposed, for using a shared hardware accelerator by multiple, concurrently running independent threads, tasks or applications. This schedule not only accounts for meeting the deadlines of the tasks/applications, it also helps to reduce the dynamic power by determining the voltages and frequencies of the soft-cores. Once a soft-core offloads its tasks to the accelerator, the core can go into sleep state which further reduces the power/temperature of the system. Summarizing, the following contributions are made by the proposed allocation and scheduling approach:

- **Shared hardware accelerator allocation schedule,** which allocates the shared hardware accelerator among the competing soft-cores such that the hardware accelerator is maximally utilized when the soft-cores offload their tasks to the accelerator.
- **Voltage-frequency tuning of the cores,** such that given throughput deadlines are met by all running tasks or applications, by distributing the workload on the programmable soft-cores and the hardware accelerator, and the power consumed by the multi-/many-core system is minimized.

The outline of the proposed heterogeneous computing architecture is shown in Figure 5-4. As seen, the "System Monitor and Control" generates the appropriate signals to determine the frequencies of the cores and the accelerator allocation by deriving and solving an optimization problem based upon the system parameters. Cores, accelerator and the external memory are connected via interconnect fabric. The accelerator consists of an arbiter to determine the core accessing the accelerator, read write control circuitries to access on-chip or off-chip memories, and internal SRAM scratchpad memory. Note that it is assumed that

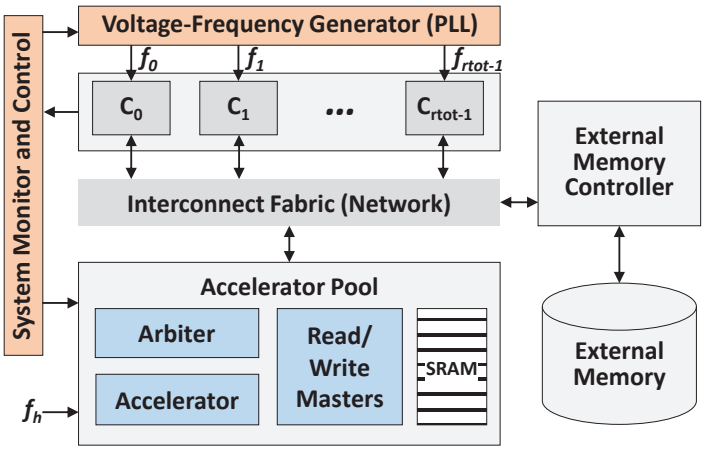


Figure 5-4: Power-efficient accelerator allocation

the subtasks which can be offloaded to the accelerator can also be done locally by the core (via software). Further, it is also assumed that the frequencies of the cores can be independently adjusted and the clock frequency of the accelerator is constant and very low (i.e., the accelerator is always bright).

For the following discussion, assume that several independent tasks are concurrently running on each compute core. In the coming text, only tasks will be used for demonstrating the proposed approach. However, the proposed approach is equally applicable to concurrently running applications, or, threads of an application. Each task has an associated set of subtasks, which must be finished within the given deadline. These subtasks can either run on software or hardware. The objective is to offload the appropriate number of subtasks to the accelerator, such that the total power is minimized and system meets the deadline(s). The discussions begin with modeling the system, and then presenting a scheduling scheme to determine the best accelerator allocation.

5.2.1.1 System Modelling and Objectives

Consider that the many-core system has r_{tot} nodes, competing to offload their subtasks to the shared hardware accelerator. The task i consumes $t_{s,i}$ seconds and $t_{h,i}$ seconds when its corresponding task is run in software and hardware respectively. An example accelerator allocation is diagrammatically shown as in Figure 5-5. This figure shows the time consumption of each application on the programmable core and the hardware accelerator. Notice that for an epoch of $t_{i,max}$ seconds, the total time for which the accelerator is engaged by the cores is given by $t_{h,t} \leq t_{i,max}$. Further, a task is run on a single core, therefore, $n_i = k_{tot}$, and the index i (for tasks) and k (for cores) can be used interchangeably (i.e., i=k).

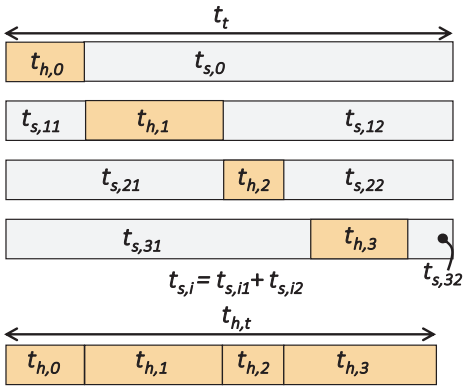


Figure 5-5: Breakdown of an example execution time on a 4-core system and a shared accelerator

The objective of the accelerator allocation approach is to minimize the power consumption of the complete system. If the power of a core k (p_k) is a function of its frequency f_k, then, mathematically, the objective is:

$$\min\left(\sum_{j=0}^{r_{tot}-1} p_j\left(f_j\right)\right) \tag{5-2}$$

At the same time, it is required to maximize the hardware utilization, i.e., the difference between the epoch time and the time for which hardware is engaged ($t_{i,max}$-$t_{h,t} \in \mathbb{R}^+$, positive real) should be as small as possible. In order to do so, we proceed by writing the total cycles processed per second on the accelerator, by all the cores to be equal to:

$$c_{h,0} n_{\text{sec},h,0} + c_{h,1} n_{\text{sec},h,1} + \ldots + c_{h,k_{tot}-1} n_{\text{sec},h,k_{tot}-1} = \sum_{k=0}^{k_{tot}-1} c_{h,k} n_{\text{sec},h,k} \tag{5-3}$$

In this equation, $c_{h,i}$ is the number of cycles per task and $n_{sec,h,i}$ is the number of subtasks per second for task i on the shared hardware accelerator. Therefore, if the difference $t_{i,max}$-$t_{h,t}$ needs to be minimized, the number of cycles processed per second on the accelerator must be matched to its clock frequency f_h. Note that the hardware is running at a fixed frequency. Mathematically, this constraint can be written as:

$$\sum_{k=0}^{k_{tot}-1} c_{h,k} n_{\text{sec},h,k} = f_h \tag{5-4}$$

Additionally, since the deadlines should be met, therefore, the additional constraint is:

$$n_{\text{sec},s,k} + n_{\text{sec},h,k} \geq n_{\text{sec},k} \qquad \forall k \in \{0, \cdots, k_{tot} - 1\} \tag{5-5}$$

This equation shows that the number of tasks per second on the hardware ($n_{sec,h,i}$) and software ($n_{sec,s,i}$) should

at least equal the number of total tasks of the task $i=k$ per second ($n_{sec,k}$).

Moreover, the clock frequencies of the cores should be bound. Thus, an additional constraint is:

$$f_{k,\min} \leq f_k \leq f_{k,\max} \tag{5-6}$$

5.2.1.2 Optimization Algorithm

In order to optimize the above function given in Equation (5-2), we are required to derive $p_k(f_k)$ in terms of the system parameters which can be tuned. If the cycles per task ($c_{h,k}$) and the number of tasks per second ($n_{sec,h,k}$) on accelerator by task $i=k$ are known, one can determine the time spent by task i on accelerator in the epoch t_t by using the formula:

$$t_{h,k} = \frac{c_{h,k} n_{\sec,h,k}}{f_h} t_t \tag{5-7}$$

Thus, the time consumed on core k ($t_{s,k}$) can be determined by using the following relation:

$$t_{s,k} = t_t - t_{h,k} = t_t \left(1 - \frac{c_{h,i} n_{\sec,h,i}}{f_h} \right) \tag{5-8}$$

Using the above equation, the frequency of the core can be determined by:

$$f_k = \frac{c_{s,k} (n_{\sec,k} - n_{\sec,h,k})}{t_{s,k}} \tag{5-9}$$

In this equation, $c_{s,k}$ is the number of cycles per task of task k on the associated soft-core. Here, the identity given in Equation (5-5) is used.

Now, by inserting Equation (5-9) in Equation (5-2), the power consumed by a core can be written as:

$$p_k(f_k) = p_k \left(\frac{c_{s,k} (n_{\sec,k} - n_{\sec,h,k})}{t_t \left(1 - \left(c_{h,k} n_{\sec,h,k} / f_h \right) \right)} \right) = p_k \left(\frac{\psi_1 - \psi_2 n_{\sec,h,k}}{1 - \psi_3 n_{\sec,h,k}} \right) \tag{5-10}$$

In this equation, ψ_1, ψ_2 and ψ_3 are constants given by:

$$\psi_1 = c_{s,k} n_{\sec,k} / t_t \qquad \psi_2 = c_{s,k} / t_t \qquad \psi_3 = c_{h,k} / f_h \tag{5-11}$$

Therefore, the complete objective function with constraints can be collectively written as:

$$\begin{aligned} &\min \left(\sum_{k=0}^{ktot-1} p_k \left(\frac{\psi_1 - \psi_2 n_{\sec,h,k}}{1 - \psi_3 n_{\sec,h,k}} \right) \right) \\ &\text{subject to:} \\ &\sum_{k=0}^{ktot-1} c_{h,k} n_{\sec,h,k} = f_h \\ &f_{k,\min} \leq \frac{\psi_1 - \psi_2 n_{\sec,h,k}}{1 - \psi_3 n_{\sec,h,k}} \leq f_{k,\max} \qquad \forall k \in \{0, \cdots, k_{tot} - 1\} \end{aligned} \tag{5-12}$$

As noted, the optimization objective given in Equation (5-12) is to find an appropriate number of tasks which are offloaded to the accelerator ($n_{sec,h,k}$), for all applications. However, the optimization algorithm presented in the above equation is non-linear, even if power and frequency approximately forms a linear relationship for the given set of frequencies.

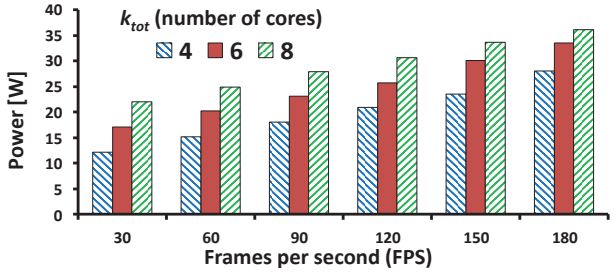


Figure 5-6: Power consumption of the programmable cores for the given FPS requirement and different number of cores

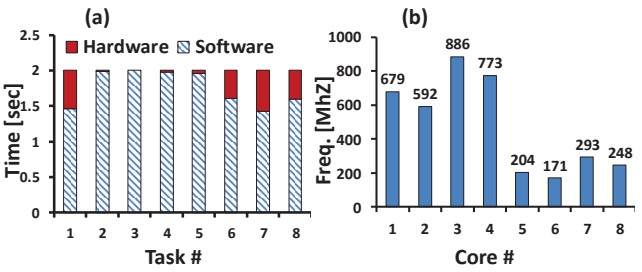


Figure 5-7: (a) Time consumed per task in hardware and software and (b) the corresponding frequency of the cores for n_{tot}=8, fps=120.

In this case, the optimization (finding the value of $n_{sec,h,k}$ for all applications) is achieved using Nelder-Mead method [246]. Nelder-Mead is a greedy heuristic that iteratively determines the value of the given objective function (v) for the given inputs and moves towards an optimum. The advantage of using Nelder-Mead method is that it does not require the derivatives of the objective function to be calculated.

In each iteration of the Nelder-Mead algorithm, the objective function is evaluated for the set of inputs ($n_{sec,h,k}$ in this case) to determine the "cost" of these inputs. That is, for the given $n_{sec,h,k}$, the value of $v=\sum p_k(f_k)$ is computed. Since in this case, constraints bound the search trajectory to find the optimum, therefore, the original constrained optimization problem is modified into an unconstrained problem. Specifically, penalty function method is used [247]. The function is evaluated as given in Algorithm 10. Here, the cost of the function increases if the difference between t_t and $t_{h,t}$ increases (maximum accelerator utilization, line 6). Since we require maximizing the accelerator utilization, therefore, we introduce an additional penalty, depending upon the difference between the number of hardware operations and the total operations (lines 8-9). Further, if the core frequencies that will support $n_{sec,s,k}$ are lesser than the minimum frequency ($f_{k,min}$), or larger than the maximum frequency ($f_{k,max}$), it will also result in increasing the cost of the function (bounded frequencies, line 10-11, Equation (5-6)). Once the cost of the function v is determined, it is compared with the previous costs and algorithm moves in the direction of the lowest cost.

Note that in the proposed approach, the accelerator populates its scratchpad by fetching the data directly from the external memory. Further, even if the size of data processed by a core and the accelerator is equal, the accelerator will still generate higher throughput, due to its custom logic implementation.

5.2.1.3 Evaluation of Accelerator Allocation

For illustrative purposes, the mean and variance computation application is considered. This application will fetch a block of 4×4 pixels from the image and generate the mean and variance of the region. This type of block-based variance computation is significant for texture classification and efficient image/video compression (e.g., see Section 4.3.3.2, where the CTU is divided into 4×4 blocks, and the mean and variance of each 4×4 is calculated).

For this evaluation, the Sniper x86 simulator is considered along with the McPAT power simulator [226, 227]. For this task, our simulations show that the number of cycles per job on the soft-core $c_{s,k}$ = 492 and $c_{h,k}$ = 70 $\forall$ k cores. The video frame is stored in the external memory. The cores process different sized parts of the same video frame and the hardware accelerator partially shares the workload of each core, by allocating its resources according to the proposed approach. A part of video frame is brought to the internal SRAM scratchpad of the accelerator. Further, if the size of the video frame a core needs to process, or, FPS requirement increases, the number of jobs per second ($n_{sec,k}$) also increases, and thus, more power and/or accelerator schedule should be allocated to that task.

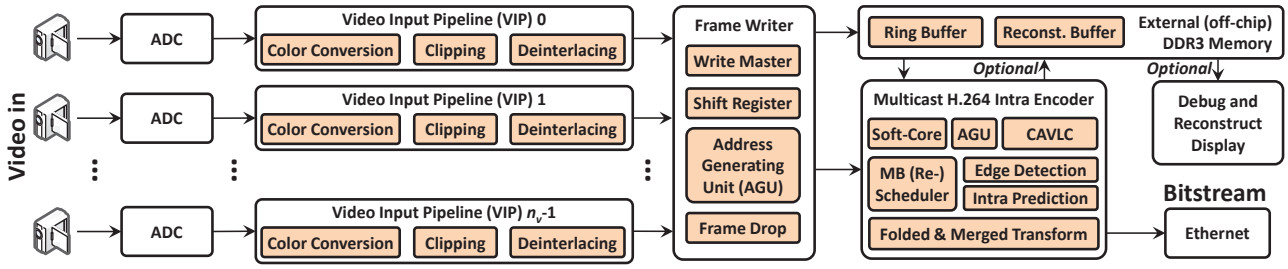


Figure 5-8: Overview diagram of the multicast H.264/AVC Intra-only encoder

Figure 5-6 shows the relationship of the power consumed by the system, by changing the FPS and the total number of cores competing for the hardware accelerator. For this evaluation, a FullHD image of size 1920×1080 pixels is considered, and regions of this image are distributed among the applications. This way, the total number of jobs per second ($n_{sec,k}$) of each task are different, and thus, the accelerator demand varies for all these tasks. Further, we consider t_t=2sec and f_h=100MHz.

Figure 5-7 (a) shows an example distribution of the hardware accelerator to the tasks. Notice that some of the tasks (e.g., 2-4) mostly run their jobs on the software. Further, summation of the time consumed by the hardware accelerator will be almost equal to t_t, which shows that the accelerator is 100% utilized. Figure 5-7 (b) shows the corresponding frequencies of the cores for the proposed offloading to the hardware accelerator. As noticed, the tasks with considerable accelerator allocation are usually running at a much lower frequency on the soft-cores. Additionally, the frequency of a core also depends upon $n_{seck,k}$ and a larger $n_{sec,k}$ either requires more offloading or a higher core frequency, determined by the optimization program given in Equation (5-12).

5.2.2 Multicast Video Processing Hardware

In Section 5.2.1, the details about sharing a hardware accelerator among different soft-cores is presented. The hardware accelerator would process the subtasks of the soft-cores in a round-robbing fashion. However, for certain types of workloads (like video encoding), processing a job or a block of video data depends upon the output of the previously processed job(s) of the same task. Therefore, the hardware needs to adapt for every application (context-switching), as would be the case in the multicasting scenario. Similarly, the data associated with different jobs can be coming from different tasks, and therefore, a mechanism is required to efficiently provide this data to the hardware accelerator.

In the same direction, an efficient hardware architecture to realize concurrent processing of multiple videos on the same device is presented in this section. The goal is multicast scheduling of the H.264/AVC encoding loop using hardware replication and reutilization, to process multiple video sources in real-time, area-efficient manner. Using a single device results in lesser cost and easier management of the system.

A high level overview of multicast H.264/AVC Intra-encoder is shown in Figure 5-8. The target of this encoder is to encode at least four FullHD frames at 25 FPS. Before encoding the individual views, the input video is preprocessed and written to the DDR3 memory. The encoder reads video frames from the DDR3 memory and compresses them. Encoding is achieved using custom hardware which will be further discussed in Section 5.3. A single soft-core is used to initialize the control registers of the VIPs and the I/O ports. It also commences the encoding, whereby the hardware co-processors start fetching video samples from the external memory. Notice that only a single H.264/AVC encoder is used for multiple video inputs.

Once the current frame buffer is ready, encoder fetches the data from the external memory, using the external memory read master. The encoder collects burst of data from the external memory and rearranges them in a

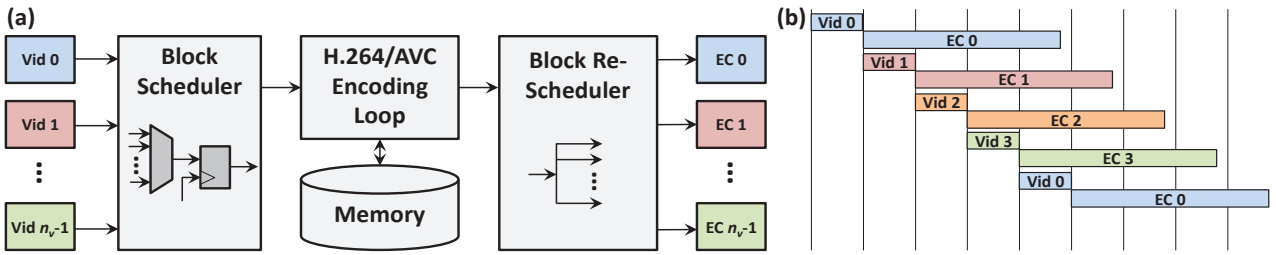


Figure 5-9: (a) Block processing scheduler and (b) example schedule with n_v=4

"MB-FIFO", where each entry in the FIFO is one MB-row width (256-bits) wide. Note that there is a separate FIFO for each of the n_v views. Every "MB-FIFO" is a set of three separate FIFOs; one for luminance and two for chrominance components (Cb and Cr). The luminance FIFO is of size 16×16 and the each chrominance component is 16×16/4. Note that these MB-FIFOs are also used for clock-domain crossing as the DDR3 and the encoder can run on separate clocks. When the data in MB-FIFO is available, it is pushed in the H.264/AVC Intra-encoding loop.

Now, the mechanisms by which video data of distinct video sources is forwarded to and collected from the video encoder are discussed.

5.2.2.1 Video Block Scheduler and Re-Scheduler

As mentioned earlier, Intra-encoding a block of video frame cannot be pipelined and a block must wait for the previous block in the encoding loop to finish (see Section 2.2.1.1). Further, blocks must be processed in raster-scan. Thus, the building modules of the encoding loop are free if not in use and only one module of the loop is active at one time, giving rise to the so-called bubbles (< 100% hardware utilization). Considering the impact of these bubbles on the multicast video encoder, large latency, area and energy overhead is incurred, as the loop is idle most of the time.

Thus, it is proposed that instead of using n_v encoding loops in parallel to process n_v independent frames, encoding loop's modules be reutilized in a time-multiplexed manner. This is accomplished by a block-level scheduler, which is used to push blocks of each view into the encoding loop in round-robin fashion. The scheduler is shown in Figure 5-9 (a) and an example schedule for n_v =4 is shown in Figure 5-9 (b). This helps in increasing the hardware utilization and reducing Silicon area. Additionally, the total energy consumption of the encoder decreases. However, note that in order to generate the bit-streams, a separate Entropy Coder (EC) unit for each view is required. In the proposed approach, a Context Adaptive Variable Length Coder (CAVLC) [32] is used as an entropy coder. The CAVLC units are fed via a re-scheduler. Using a single CAVLC unit for the multicast encoder is difficult to realize due to two factors.

- Firstly, a separate CAVLC unit per video stream is essential because the CAVLC unit requires at least 16×16+2×8×8 = 384 cycles to process a full luminance and two chrominance blocks. This is because each quantized coefficient (see Figure 2-3) must be coded using the previous quantized coefficient, and hence the name context adaptivity. Further, the bits generated by these coefficients must be pushed into a bit-buffer in serial. This is larger than the 183 cycles/block cycle budget required to process a 4K Ultra-HD frame (3840×2160, i.e., four 1920×1080 frames) at 25 FPS, if the hardware is running at 150MHz.
- Secondly, the context adaptivity required in CAVLC can only be realized by using independent data buffers for each view, which is very hard to maintain by a single CAVLC unit and will incur large latency.

Note that the output of the scheduler is registered because of a MUX in front of the I16MB loop. Contrarily,

re-scheduler's output is directly connected with all CAVLC units. Only a "valid" signal is required which determines the CAVLC unit to which data is directed.

It is possible to insert block-based CABAC units [248] seamlessly in the proposed multicast encoder, instead of the block-based CAVLC units. The proposed multicast video encoding scheme is independent of the type of entropy coder used. CABAC provides better compression efficiency compared to CAVLC (up to 15%). However, note that up to 15% bitrate savings by using CABAC instead of CAVLC will only occur in ideal scenarios. Further, the latency incurred by CABAC will be higher compared to CAVLC, as CAVLC is the low complexity entropy encoding alternative in H.264/AVC.

Although the presented multicasting scenario is explained with the help of H.264/AVC, it is also applicable to HEVC and other state-of-the-art video encoders.

5.3 Efficient Hardware Accelerator Architectures

In Section 5.2, different schemes for allocating the shared hardware accelerator to multiple threads or applications is proposed. This section deals with efficient design of some hardware accelerators, specifically for video encoding applications (H.264/AVC and HEVC).

5.3.1 Low Latency H.264/AVC Encoding Loop

A multicast solution for H.264/AVC is given in Figure 5-8. The architectural details about the H.264/AVC video encoder used in this design are further discussed here. The goal of the architecture is to provide high throughput, low latency and area-efficiency, and be capable of encoding FullHD views in real-time. H.264/AVC Intra-encoding loop has inherent sequential dependencies, which limit the throughput of the encoder. A high-level architecture of an H.264/AVC Intra-encoder with its main functional blocks for a single video input is shown in Figure 5-10. Blocks or MBs of an input frame are processed individually in raster scan order through the encoding loop (marked as dashed red arrow). The content of each MB is predicted based upon the pixel values of the left, upper, and upper-left MBs (Intra prediction generator in Figure 5-10). The residue $X^{(r)}$ (i.e., the pixel-wise difference of the predicted MB X' and the actual MB X) is sent to the transform block. There, the residue is processed by the Discrete Cosine Transformation (DCT), Hadamard Transformation (HT), Quantization (Q) and forwarded to the Entropy Coding. Additionally, the data is locally decoded, i.e., processed by Inverse Quantization (IQ), Inverse DCT (IDCT), Inverse HT (IHT), and Reconstruct. The reconstructed MBs are required as a base for the next predictions. There are various data dependencies inside this loop and those with the highest significance for encoding performance are explained in the following.

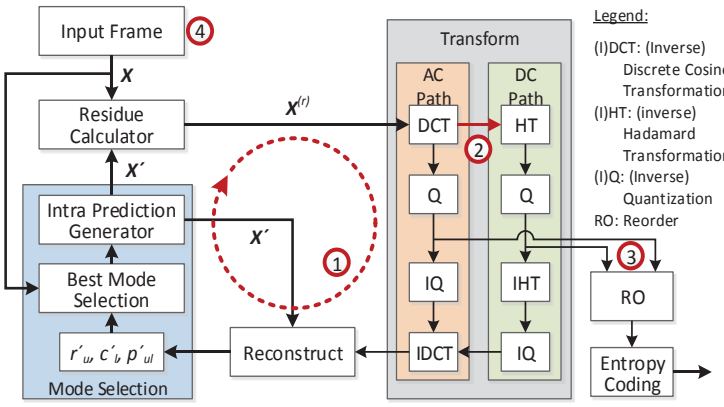


Figure 5-10: H.264/AVC Intra processing loop architecture. The current MB is labelled as X, the predicted and residue MBs are labeled X' and $X^{(r)}$; sequential data dependencies are shown by dashed arrows (① and ②)

- **Dependency 1:** The main performance degrading dependency comes from the fact that MB processing cannot be pipelined. Before entering the encoding loop, the current MB has to wait for the previous MBs in the loop to be fully encoded and then locally decoded. The dashed arrow labeled ① in Figure 5-10 depicts this dependency.

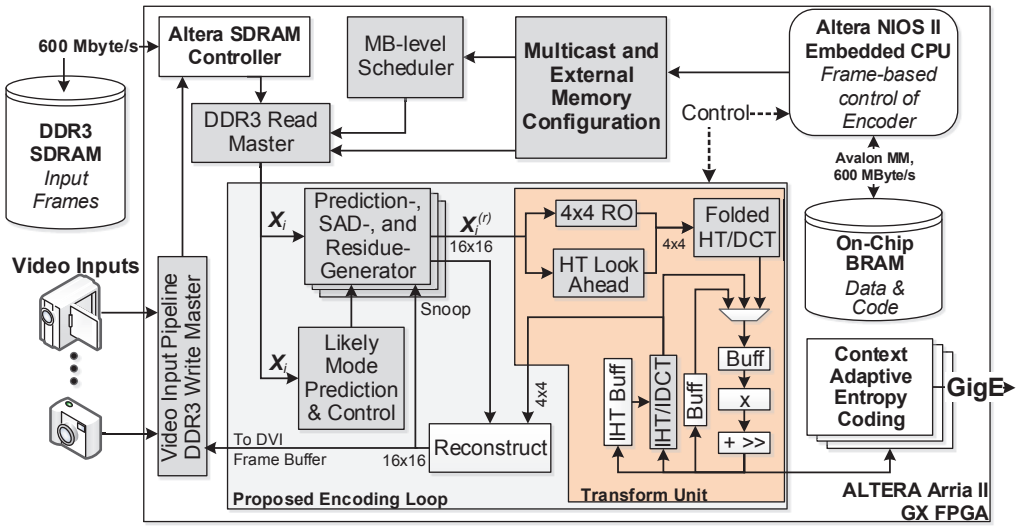


Figure 5-11: Proposed hardware accelerator architecture for multicast H.264/AVC Intra video encoder. This design also shows the connections of the multicast encoder with the I/O ports of the system

- **Dependency 2:** The transform block consists of two paths, one processing the AC part of the spatial frequencies, the other processing the DC part. The outputs from the DCT are fed both to HT in the DC path and Q in the AC path. However, HT cannot start execution until the whole MB is processed by DCT, but the Q and IQ blocks in the AC path can start processing earlier. This dependency is shown as arrow labeled ② in Figure 5-10. Additionally, to compute IDCT, data from both DC and AC paths is required. Therefore, IDCT has to stall and wait for data from the DC path.
- **Dependency 3:** The entropy-coding (CAVLC or CABAC) processes the DC output coefficients before the AC coefficients, but the DC coefficients are generated later. Label ③ denotes this dependency. In addition, the entropy coding scheme also requires to reorder the data before processing it. This adds to the latency of the output generation.
- **Dependency 4:** Video frame samples in the form of MBs are brought from camera or off-chip memory to the encoding modules (shown by Label ④). This transmission incurs high latency if the encoding loop's workload and efficient reshaping of video samples into MBs are not considered.

Low latency and high throughput for H.264/AVC encoder are obtained by addressing the dependencies presented above. The resulting architecture with hardware co-processors is shown in Figure 5-11. This figure elaborates the design of H.264/AVC hardware accelerators, and their interconnections with the video I/O (Section 5.1) and multicast enabling modules (Section 5.2.2.1). Area- and computational-efficiency is obtained by designing fast, area-efficient hardware accelerator circuits. Instead of a single module of the encoding loop, the focus of this section is on the complete H.264/AVC Intra-encoding system. In short, the contributions of this section are:

- **Adaptive H.264 Intra prediction scheme** by utilizing a small and speedy edge extractor for scheduling the calculation of different Intra prediction modes. The user can configure the number of predictions to increase the throughput of the encoding loop.
- **Area-efficient transform module design** where AC and DC path of the encoding loop are decoupled to reduce latency. Moreover, folding DCT/IDCT and utilizing the same hardware resources and interlacing of Q/IQ blocks reduces the total silicon footprint of the platform.

5.3.1.1 4×4 Reordering and HT Lookahead

To decrease the latency of the transform stage (Dependency 2 in Section 5.3.1), a method is proposed to decouple the AC path from the DC path in the transform block. The inner blocks of the transform unit work at a 4×4 granularity. Thus, the residue 16×16 block is subdivided into 16 4×4 blocks using the 4×4 ReOrder (RO) stage as given in Figure 5-11. The residue generator stage provides one line of $X^{(r)}$ (16 pixels in 1 cycle, i^{th} line of the MB given by $X^{(r)}{}_i$). Whenever four $X^{(r)}{}_i$ are accumulated in the input registers of the 4×4 RO stage, RO generates four 4×4 blocks and pushes them to the input FIFO of the 2D-DCT stage. As mentioned in Section 5.3.1, HT cannot commence until all 4×4 blocks are processed by the DCT. However, by simplifying the DCT formula, it is observed that the n^{th} output DC value (which is the n^{th} HT input value $HT^n{}_{in}$) obtained from the DCT by processing the n^{th} 4×4 residue block ($X^{(r)n}$) can be calculated by:

$$HT_{in}^{n} = \sum_{i=1}^{4}\sum_{j=1}^{4} X_{i,j}^{(r)n}, \quad n = 1,\ldots,16 \tag{5-13}$$

This shows that $HT^n{}_{in}$ can be generated by adding all the entries in the 4×4 block. Thus, the DC outputs are generated at RO stage instead of the DCT stage, effectively decoupling DCT from HT. This process is shown in Figure 5-12. A residue line can add to the current accumulators (shown as registers R) of the lookahead HT memory or it can trigger a new DC coefficient generation, controlled by the counter administering the MUXes M. Using our proposed scheme by generating HT transformed coefficients ahead of DCT transformed coefficients results in reduce latency, as the entropy coding can start earlier. Since IDCT requires IHT transformed data (Section 5.3.1), therefore, the output of AC Quantization (i.e., 16-bits per pixel, 256-bits per 4×4) must be stored in for the complete MB in an intermediate memory (256×16 = 4096 bits). However, this scheme eliminates the need of this buffer as DC path is executed first and the IDCT can fetch the data directly, without the need of an additional storage memory.

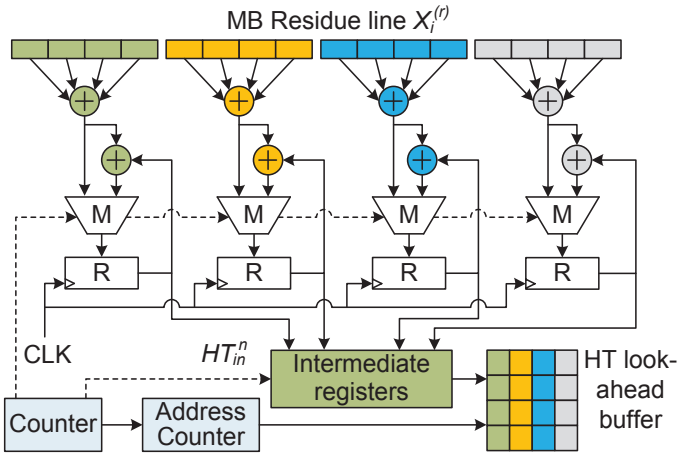


Figure 5-12: HT-Lookahead buffer filler responsible for pre-computations

The entropy coding requires the arrival of 4×4 blocks in a Z-fashion [32]. The proposed RO unit generates the 4×4 blocks for the 2D-DCT unit as required by the entropy coding on the fly, eliminating the need for an extra RO unit in front of entropy coding. This reduces the latency of the whole encoder (addressing Dependency 3 from Section 5.3.1).

5.3.1.2 Transform and Quantization Stage

In order to save area, DCT and HT are employed in the same hardware block, in a time-multiplexed manner. DCT and HT equations are similar, and these transforms can be implemented using butterflies [249, 183]. HT and DCT butterflies are composed of two stages (one for horizontal and the other for vertical transform). Each butterfly can process four inputs simultaneously, and eight butterflies are required to process a 4×4 block. In order to save area, the architectures are folded and the butterflies are reused for both horizontal and vertical transform in HT/DCT and IHT/IDCT. This arrangement is shown in Figure 5-13, where a MUX in front of the butterflies determines whether to provide new or previously computed values to the butterflies. Further, the horizontal/vertical transformation is implemented via rewiring the output. Therefore, the output is generated

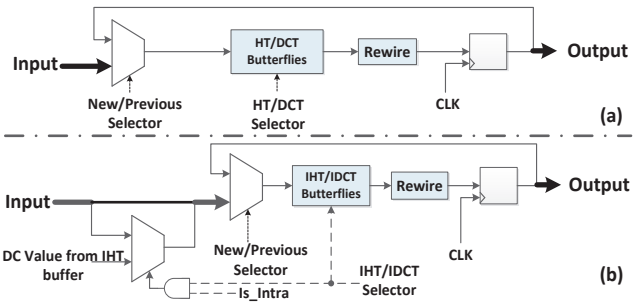


Figure 5-13: Folded design of (a) DCT and (b) IDCT

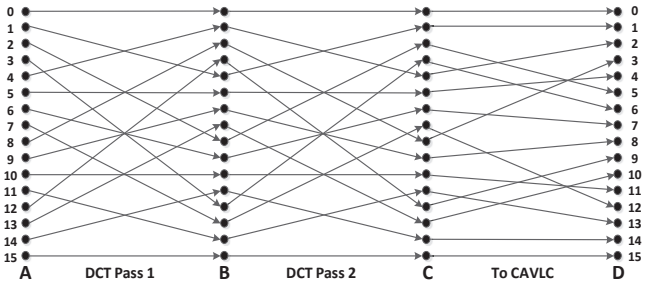


Figure 5-14: Data flow graph from the output of HT/DCT butterflies to the input of the entropy coder

with a single cycle delay. As seen in Figure 5-13 (b), the IHT/IDCT module is also Inter-ready (as it can also discard inputs from the IHT buffer using an additional MUX).

The data-flow-graph of the HT/DCT butterflies' output till the input of CAVLC is shown in Figure 5-14. From Point A to Point B, the flow graph denotes the rewiring as explained for Figure 5-13, in the first iteration of the folded HT/DCT. In the next iteration, the rewire function again transforms the data from Point B till Point C. The quantizer does not rearrange data, and hence is not shown in this figure. Since the data is jumbled, therefore, before feeding this to the CAVLC FIFO, it is rewired from Point C to Point D because the CAVLC unit requires the output data organized in the frequency-scan manner [32]. Thus, an additional latency is saved by avoiding the rewriting of the 4×4 quantized coefficients.

Similar to reusing hardware in HT/DCT and IHT/IDCT, the proposed architecture combines the Qunatizer (Q) and Inverse Quantizer (IQ) in DC and AC paths as shown in Figure 5-15 (a). The quantization and inverse quantization relationships, both for DC and AC paths, are given in Equation (5-14). Depending upon the QP value, quantization coefficients (q_i), inverse quantization coefficients (dq_i) and other values ($q_{i,const}$, $q_{i,bits}$, $q_{i,per}$) are selected. Since the goal is to reuse hardware for maximum area efficiency, the multipliers and barrel shifters are shared by interlacing Q and IQ. This scheme is shown in Figure 5-15 (a).

$$
\begin{aligned}
Q_{DC,i} &= \left(\left|DC_i\right| \times q_i + 2q_{i,const}\right) \gg \left(q_{i,bits} + 1\right) \\
Q_{AC,i} &= \left(\left|AC_i\right| \times q_i + q_{i,const}\right) \gg q_{i,bits} \\
IQ_{DC,i} &= \left(\left(\left(Q_{DC,i} \times dq_i\right) \ll q_{i,per}\right) + 2\right) \gg 2 \\
IQ_{AC,i} &= \left(Q_{DC,i} \times dq_i\right) \ll q_{i,per}
\end{aligned}
\qquad (5\text{-}14)
$$

Note that the one cycle latency in the DCT's output is exploited in Q/IQ unit as shown by the schedule in Figure 5-15 (b). This schedule represents how Q and IQ units process DCT output. Multipliers are costly in terms of area (usually implemented via DSP blocks in a FPGA and there is a limited number of DSPs available). Further, the shift operation requires barrel shifters where each shifter requires 64 multiplexers for a 16-bit input. However, in the proposed architecture, an extra output buffer (Delay Buffer) is required for assisting in scheduling. Moreover, as required by IDCT, the folded architecture of IDCT is also provided with valid inputs in alternate cycles by the IQ stage. In the proposed implementation, a single-bit register circuit that mimics the registers generates the output valid signal.

In short, using the proposed HT/DCT (IHT/IDCT) architecture given in Figure 5-13, we can effectively reduce the area by half compared to the standard H.264/AVC transform stage implementations (see Figure 5-10). First, due to the folded structure, only four butterflies are used instead of eight, per transform. Since the 4×4 output of DCT is delayed by a single cycle, we allow Q and IQ to share the multipliers and the barrel shifters. Therefore, only 16 multipliers and 16 barrel shifters are used, instead of 32 multipliers and 32 barrel

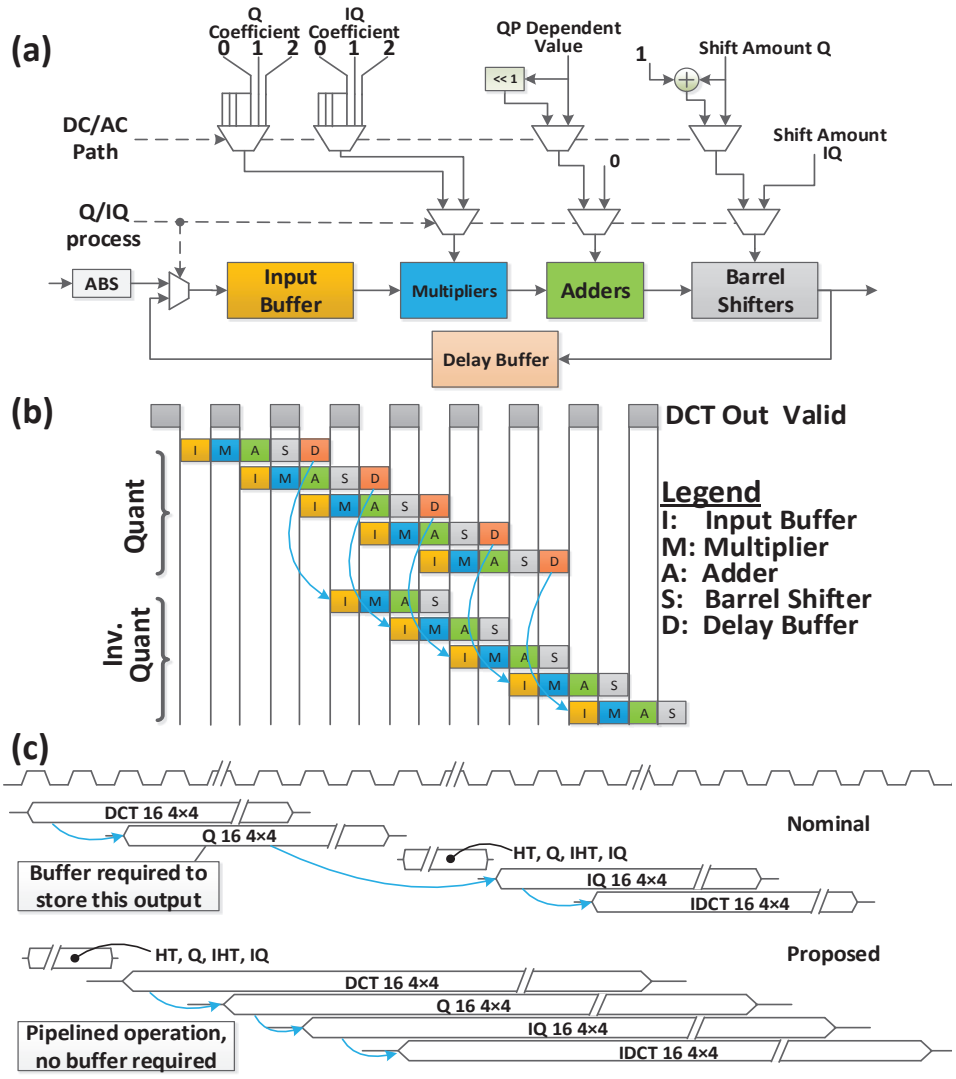


Figure 5-15: (a) Interlaced quantizer and inverse quantizer, (b) scheduler of quantizer/inverse quantizer and (c) scheduler of (inverse) transform and (inverse) quantization using nominal and proposed architectures

shifters.

Further, the proposed folding and interlacing of Q and IQ results only in a penalty of two increased computational cycles. This is shown in Figure 5-15 (c). In the nominal case, the DCT and AC quantization takes place for the complete MB (i.e., 16 DC coefficients are generated). Afterwards, the DC path can commence, which is followed by pipelined AC inverse quantization and IDCT. However, in the proposed approach, DC path is executed first due to the HT lookahead scheme (see Figure 5-12). Afterwards, the DCT, quantization, inverse quantization and IDCT are executed in a pipelined manner.

5.3.1.3 Mode Decision

The Mode Decision module has to select one mode out of the four available modes for H.264/AVC (which are usually termed as V, H, DC, and P for 16×16 MB). It can apply full search to select the best mode (i.e., the best X') using either SATD or just SAD between X and X'. The best mode decision is rather slow if the generation of X' and the cost calculation for every mode is computed sequentially using one hardware unit. On the other extreme, the parallel implementation of all four modes would demand a significant amount of hardware. Therefore, it is proposed that only 16 adders/subtractors are to be used and thus, SAD of a luminance

MB can be generated in 16 cycles. A full-reconstructed 16×16 MB is available from the reconstruction block (generating one reconstructed MB line) after 20 cycles.

Figure 5-16 shows the proposed Mode Decision module. $FIFO_{Xi}$ contains X and this data is then written to a shared on-chip memory that stores line-by-line MB. The pipelined Edge Detector (discussed in Section 5.3.1.4) unit predicts the order of the modes to test by finding the most-dominant edge for X. Using this order, predictions X' are generated by the prediction generator and the residual data $X^{(r)}$ is evaluated by calculating the SAD and stored in the same on-chip buffer as X at distinct address space. The residue calculator generates $X^{(r)}{}_i$ and passes it to both the SAD unit and the residue memory block. After all residues are stored into this memory, the residue resulting in the lowest SAD value is written to the 4×4 RO stage. Note that if the amount of cycles needed for encoding does not comply with the resolution and frame rate requirements, the proposed SAD unit can be configured to use the down-sampled version of the current MB for residual calculation, where the down-sampling factor denoted by d_s means that every d_s line of X is used for the SAD computation. When $d_s > 1$, then the number of cycles for one SAD computation decreases. For example, for one luminance MB with $d_s = 2$, it takes 8 cycles for the SAD computation rather than 16 with $d_s = 1$. However, if d_s is larger than 1, then $X^{(r)}$ is also downsampled and thus it must be updated with the full residue after final mode selection.

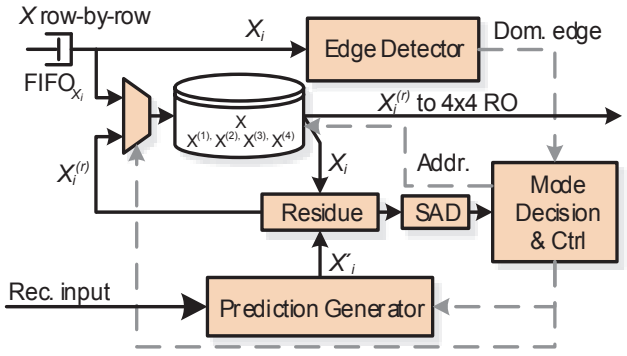


Figure 5-16: Mode decision module for Intra 16×16

5.3.1.4 Edge Based Mode Prediction

If the encoding loop cycle latency does not comply with the allotted budget, it becomes necessary to sacrifice some encoding efficiency to meet the performance goals. As the transform loop is essential to the encoding process, usually less cycles of the overall budget are available to the Mode Decision block. Unlikely predictions are not selected as candidates for residue generation by using a preprocessing stage. This procedure is not required for parallel SAD implementations but it is useful for a sequential mode decider. Various algorithms are proposed in literature for mode prediction and elimination of unlikely modes [6, 127], where texture-based edge extraction information is used to determine the probable modes. Once the reconstructed data is available, the most suitable modes are applied first and the other modes are either delayed or even skipped. However, an edge extraction procedures for a 16×16 block will require 256 iterations (requiring a tan^{-1} function and a divider) plus dominant mode search cycles. Therefore, it cannot be embedded as a stage in the encoding loop or parallel to the encoding loop for 4K-UHD sequences (encoding loop must finish within 183 cycles for 150MHz at 25 fps). It can, however, be implemented as a separate pipeline stage outside the encoding loop. This introduces latency and limits the throughput of the encoding process. In addition, the area overhead might become too large if parallel edge extractors are employed. Moreover, the edge-threshold is decided at design time but it is not applicable to every video scene. In contrast to this, the proposed approach uses a lightweight and efficient mode prediction process with a modified version of edge extraction procedures, but can still extract the dominant edge information from the input MB and does not require an edge-threshold, which would have to be adapted to the scene conditions.

For the current MB X_n, the proposed method can run in parallel with the residue calculator and transform stage and can generate the likely modes before the reconstructed previous MB X_{n-1} is available for the prediction generation stage. The crux of the method involves the estimation of edge pixels at the borders of the MBs by computing a sequential *running-difference rd*. This means that only one subtractor and one ABS (absolute)

modules is used for both the top and bottom borders, and one subtractor and one absolute module each for the left and right borders to detect the dominant edge direction. For example, at the vertical border *b* or *c* of the MB in Figure 5-17, rd_i is computed as:

$$rd_i = \left| X_{i,j} - X_{i+1,j} \right|, \ \forall i = 1,\ldots,15, \ j \in \{1, \ 16\} \tag{5-15}$$

The pixel location *i* where rd_i is maximum is declared as the point where the edge passed. Let $E_b=i$ be the location of the edge passing through *b* at *i*, generating the maximum rd_i given by rd_b. Similarly, edge *E* and running-difference *rd* at each border is found and the two borders with maximum *rd* are declared to have a passing edge. This comes up with six distinct possibilities, as shown in Figure 5-17 where only one edge out of the six can occur. Using this edge detection approach, the probabilities of modes for every line for various HD sequences are computed, and Algorithm 11 is devised for selecting the precedence order *m* of prediction generation. When the difference between all rd_i are less than some threshold (currently, it is kept constant at 5 as there were no observable dependencies on the input data), it is concluded that there is no edge and hence the algorithm detects $Line_0$, or no line. Note that the hardware works on a single line of input, i.e., X_i. In addition, in the proposed Algorithm 11, the planar mode (P) is never selected first and intermediate values for P [32] can be generated in parallel to the other modes preceding it, therefore, P mode prediction/residue computations take the same amount of cycles as the other modes. A parameter θ is defined as the number of allowable prediction (i.e., SAD) computations. The value of θ can be altered to compute SADs for the most likely modes, in order to meet the cycle budget of the encoding loop. With $\theta = 1$, there is no need of SAD computations and the most probable mode is used to generate prediction and its residue is forwarded directly to the transform stage. For $1 < \theta \leq 4$, the proposed approach start with the most probable mode in *m* and computes the SADs.

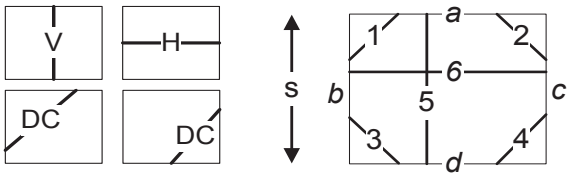


Figure 5-17: Strongest edge detection approach

5.3.1.5 Evaluating the Proposed H.264/AVC Architecture

Here, some evaluations of the proposed H.264/AVC encoding loop are presented. For comprehensive details, refer to Appendix C.

Encoding Loop: In the proposed encoding loop, the transform stages are merged to save area and power, while resulting in minimum cycle-penalties. In [5], authors have also designed a multi-transform engine, which merges all transforms (HT, IHT, DCT, IDCT) in the transform loop. Our proposed scheme only merges HT/DCT and IHT/IDCT. However, the total area consumed by our two separate transform units is lesser, because [5] implements large multiplexers. See Table 5-1. Further, the increase in the proposed merged Q/IQ unit is due to the delay buffer used by our approach for interlacing Q and IQ (see Figure 5-15). The complete transform unit proposed in [5] uses 110.29K gates, while our approach uses 106.45K gates. For a throughput of 16 pixels per cycle, this corresponds to 6.89K gates per pixel and 63.5mW per pixel via [5], and 6.65K gates per pixel and 61.77mW per pixel using our proposed approach. Note that the proposed encoding loop is merged within the multicast H.264/AVC video encoder with n_v parallel videos as given in 5.2.2. Therefore, the area and power consumption of [5] should actually be multiplied with n_v, as [5] does not provide hardware sharing for multicasting solutions. On the other hand, we use the same hardware for all the encoders. Thus, with n_v=4, our schemes reduces the area of the transform unit by ~4.14×.

Table 5-1: Area comparison of the transform unit for TSMC 65nm technology [264], power consumption via Altera [288].

	Baseline	This work	[5]
Transform	21.68	15.04K	21.39K
Q/IQ	67.32K	60.52K	51.62K
Total	129K	106.45K	110.29K
Power (mW)	1039.23	988.38	1015.93

Edge Based Mode Prediction: For the proposed edge based most probable mode estimation, Figure 5-18 reports the average hit-rate per frame. H is defined as the mode priority of the full search mode in the ordered likely candidates m generated by our Algorithm 11. $H = 4$ is the worst misprediction, which means that the Intra prediction mode finally selected after comparing the SADs of all four modes, is the least probable mode according to our prediction algorithm.

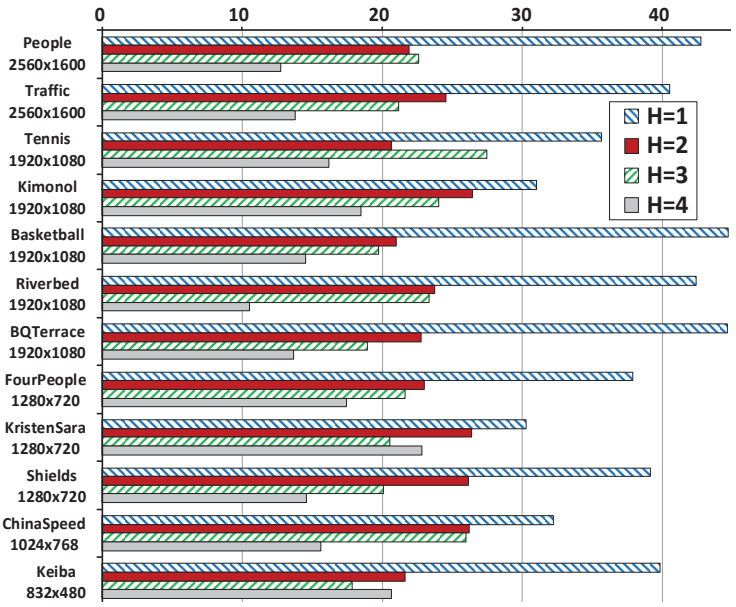


Figure 5-18: Hit rates in percent (avg. over QPs 18...32, step size=2) for various sequences; H: priority of full search within mode schedule

For evaluating the encoding quality of the proposed edge-based most probable mode selection, plots of PSNR against the bitrate at $\theta = 1$ shown in Figure 5-19. These curves suggest that the PSNR curve for likely mode predictions matches the full search Intra mode (also called Closed Loop (CL)) selection closely. As a comparison to the likely mode selection procedure presented in this paper, the PSNR curve for Open Loop (OL) algorithm [129] with $\theta = 1$ is also plotted. Notice that our scheme outperforms the OL algorithm.

Further, the edge detection approach presented in [6] uses 8.4K gates, while our proposed edge detection scheme only uses 4.4K gates for TSMC 65nm technology. Moreover, [6] requires testing at least two Intra modes ($\theta \geq 2$), whereas our scheme allows for testing a single mode as well ($\theta \geq 1$).

5.3.2 Distributed Hardware Accelerator Architecture

Although the power-efficiency (i.e., the amount of work per unit of power) of hardware accelerator is high compared to a software based solution, it is still possible to further reduce the power consumption of the hardware accelerator. As discussed in Section 2.2.1, HEVC uses PU sizes from 64×64 down to 4×4. Generating Intra prediction in hardware would thus require implementing Intra prediction circuitry for each of these PU sizes. However, this will incur large area overhead and also a large power penalty. This thesis proposes to distribute the hardware of the largest Intra prediction generating unit in HEVC, and clock- or power-gate individual components of the large hardware accelerator. This way, only a selected few constituent components

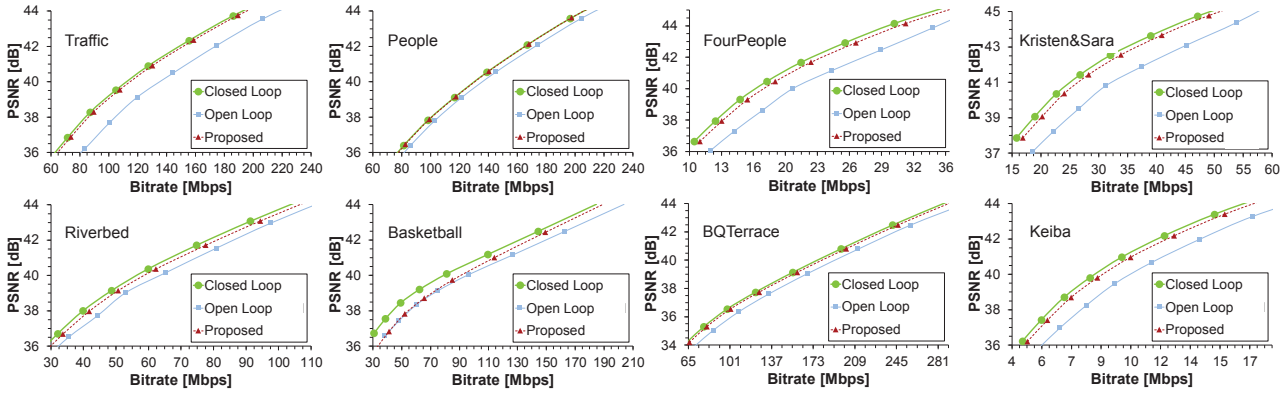


Figure 5-19: PSNR vs. Bitrate plot for proposed and open loop $\theta=1$; each value represents the average results for QP sweeps from 18 to 32 (step size = 2)

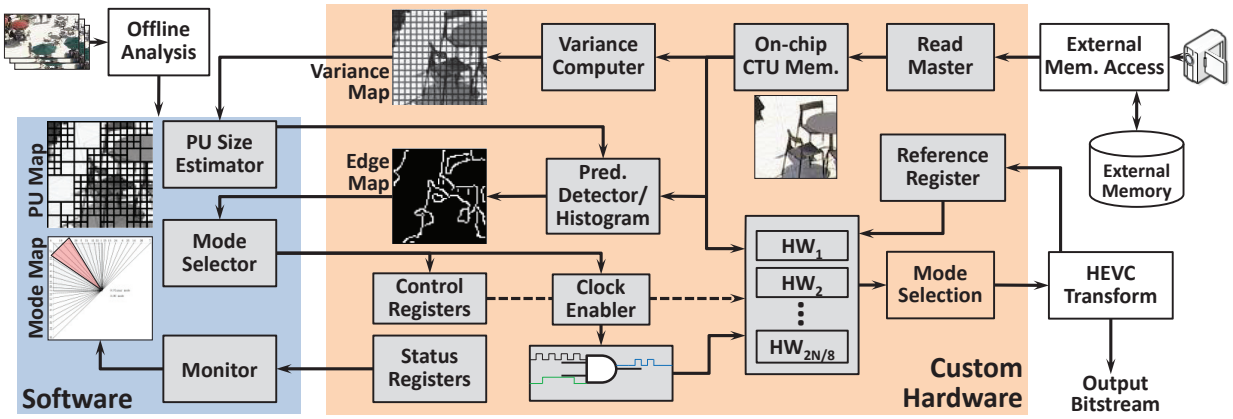


Figure 5-20: Hardware-software collaborative control for complexity and power reduction of HEVC Intra encoding

of the hardware accelerator will be consuming power while the rest would be turned off.

The proposed distributed hardware accelerator architecture is outlined in Figure 5-20. This figure shows a hardware-software collaborative complexity and power reduction approach for HEVC Intra encoder. Following the standard hardware-software partitioning trends, high complexity jobs with minimal conditional jumps are processed via hardware accelerators, whereas low complexity processing containing mainly control decisions happen at the software layer. A block of pixels (i.e., a CTU) is fetched from the external memory for processing. In order to reduce the computational complexity, edge histogram is generated at the hardware layer, whereas the software layer uses this information to determine the most probable Intra prediction mode (Section 4.3.3.1). Similarly, the variance of all 4×4s within the CTU is computed in hardware, whereas the PU maps (PUM and PUMA, Section 4.3.3.2) generation for reducing HEVC complexity are generated in the software. Based upon the size of the PU, its location and the prediction mode, appropriate video samples are generated for the prediction block.

Intra Prediction Generator: The prediction generation is carried out using a distributed hardware accelerator. The prediction generation hardware is capable of generating a prediction of the largest possible PU (i.e., 64×64) in a single cycle. However, this is achieved by concatenating the output of its eight constituent, individual 8-sample prediction generators. The value of 8-samples per individual prediction generator is chosen because by our analysis in Table 5-2 of various video sequences, having different resolutions with varying motion and texture denotes that 8×8 PU is the highest occurring PU size. Moreover, note that it is impossible to have a PU not at the 8×8 boundary because the minimum CU size in HEVC is 8×8.

Table 5-2: Percentage distribution of PU sizes

PU Size	RaceHorsesC		BQSquare		FourPeople	
	QP=22	QP=37	QP=22	QP=37	QP=22	QP=37
64	0.049	0.043	0	0	0.102	0.539
32	2.257	3.217	0.023	0.985	3.126	6.155
16	13.168	22.523	2.386	8.802	17.295	26.749
8	54.463	64.085	35.967	50.87	56.727	58.71
4	30.064	10.131	61.623	39.344	22.75	7.793

Clock Gating Logic: As discussed previously, 8 pixels of a full CTU row are associated with a single component of the Intra prediction generation hardware. Thus, when a PU is processed, it is possible that output from some of the Intra prediction generators is not required. Therefore, the software can clock-gate these generators to save energy. The clock-gating circuit is controlled by the control register, whose individual bits are the set by the software, depending upon the PU size and location of the PU within the CTU.

5.3.2.1 Energy and Resource Evaluation

For the proposed architecture given in Figure 5-20, the area and frequency results of individual blocks are

Table 5-3: HW consumption for a 64×64 CTU (1 PLL, ~205K ALUTs, ~205K Registers, 736 DSP blocks)

HW block	Freq (MHz)	ALUTs and Registers	Memory (bits)	DSP blocks	Logic
Variance	167.14	253,386	0	7 (<1%)	< 1 %
Sobel Histogram	243.72	77, 86	0	1 (< 1%)	< 1 %
Intra Pred. Block	243.61	203, 377	0	8 (1%)	< 1%
Total CI	162.79	13409, 6934	43008	72 (10%)	10 %

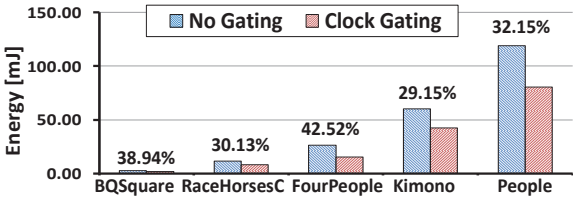


Figure 5-21: Average Energy consumption for one frame

tabulated in Table 5-3. The Altera FPGA (EP2AGX260FF3513) used for evaluations is a mid-range FPGA. Hence, by using a complete custom design, it is expected that an ASIC can improve the throughput and area savings even further.

In Figure 5-21, the energy consumption comparison between (a) no clock gating, single prediction hardware and (b) proposed approach for 1 frame is performed (with energy savings percentage on top of the bars). The stimuli data generated by the HEVC reference software and ModelSim [250] is provided to the Altera's Powerplay Power Analyzer tool (for determining signals' static probabilities and transition densities) and the energy numbers are reported.

5.4 Hybrid Video Memory Architectures

An integral architectural focus of any hardware accelerator based design is power- and compute-efficient implementation of the memory subsystem. As discussed in Sections 2.2.1.2, 3.3.2.1 and 3.3.2.2, the high access rates of external memories and high leakage power of on-chip video memories (e.g., by ME algorithm) can increase the energy consumption of the video processing system considerably. A hybrid memory architecture can therefore, be employed to exploit the advantages offered by both volatile and non-volatile memories. Basic idea is to push the video data which will be read more often (i.e., will remain valid in memory for a long duration), into memory regions within NVM, and thus save leakage power. The converse is true for\ VM, whereby the data which is overwritten considerably should be placed in low-write energy memory. Similarly, fast block-matching algorithms usually follow a fixed pattern (e.g., TZ search implemented in the HEVC reference software). This pattern can be used to determine the highly likely area of the search window where the next predictor under test should lie. Turning OFF un-accessed memory regions also saves leakage energy.

To realize the above, an Adaptive Energy Management for On-Chip Hybrid Video Memories (AMBER) is proposed. It is a novel hybrid memory architecture with an integrated energy management for video processing application. The memory architecture comprises of hardware accelerators for fetching video frames from the external memory and storing them in the on-chip high-capacity hybrid memory. The adaptive energy management approach then power gates specific hybrid memory regions/sectors to save energy. AMBER leverages the characteristics of different memory types (e.g. read/write latencies, leakage energy etc.) and the application specific knowledge to reduce the total energy consumption and processing latency of the video processing system. Moreover, AMBER can be seamlessly integrated with other orthogonal approaches that reduce dynamic power consumption of video processing systems. In a nutshell, AMBER targets:

- **Design of on-chip hierarchical video memory subsystem,** using hybrid memories (specifically, SRAM and MRAM) to exploit the advantages offered by each memory type, especially for video applications.
- **Runtime adaptive power gating of selective memory sectors** by using an adaptive energy manager, which exploits the video application- and video content-specific properties to reduce leakage energy consumption of the system.

As a proof of concept, AMBER methodology is applied for energy management of the memory subsystem,

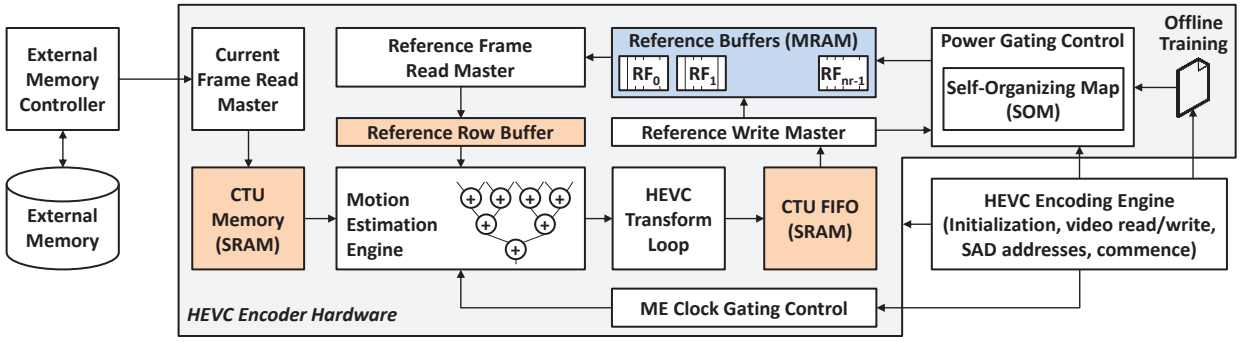


Figure 5-22: AMBER system architecture for HEVC video encoder

for the HEVC encoding system. In the following, the details about the memory subsystem architecture will be followed by the specifics of controlling the power consumption of the video processing system.

The proposed memory energy management scheme for HEVC hybrid memories is shown in Figure 5-22. The off-chip or external memory holds only the current video frame data. Using the external memory controller and hardware accelerators, this data is brought to the on-chip SRAM CTU buffer. This buffer is used for motion estimation where the motion estimation (block-matching algorithm) takes place. The motion estimation process is controlled by the HEVC controller. After block-matching, best predictor is forwarded to the HEVC encoder, which generates the reference CTU, used as a reference for the next frames. The reference CTU is pushed into the SRAM CTU FIFO. This FIFO is read by the MRAM reference frame buffers, which are power-gated by the power-gate control module. In the following, we discuss the basic components of our system in detail.

5.4.1 AMBER Memory Hierarchy

As shown in Figure 5-22, in the AMBER framework, external memory holds only the current video frame. Since high-density and low read/write latency is required for the external memory, DRAM is a suitable candidate for the external memory. It can be replaced by a PRAM but writing this memory by the video camera is both energy- and latency-wise expensive.

The current read-master accelerator reads the current CTU data from the off-chip memory and places it in the on-chip CTU memory of size $b_w \times b_h$, with b_w, $b_h \in \{16, 32, 64\}$ and each sample of size 8-bits. Due to a small amount of data (maximum 64×64 pixels) being written to this memory from the external DRAM, the CTU memory is composed of SRAMs. Thus, motion estimation can start as soon as data is available. The output of motion estimation is processed and fed to the reference frame buffers via a SRAM FIFO, also of size $b_w \times b_h$. Since SRAM has the fastest read/write characteristics, these SRAM memories hide the latencies from the external bus system and the HEVC processing engine.

The SRAM FIFO feeds the on-chip MRAM reference frame buffers. These buffers hold either n_r reference frames (in case of multi-frame prediction) or a single large frame (for example 8K UHD, 7680×4352) in n_r separate video frame buffers. The reference read master reads the predictors from these buffers and forwards it to the motion estimation engine.

5.4.2 MRAM Reference Buffers Architecture

The MRAM reference frame buffers consist of memory banks and sectors. Each bank is reserved for one reference frame of size $w \times h$. Banks are divided into sectors where individual sectors are MRAMs of size $b_w \times h$ and can be independently power-gated using the power-gate control. These memories are “normally off” and only awakened for reading or writing. The reference data generated by the HEVC encoder is fed to the SRAM

FIFO. A reference write master component, with an internal address generating unit writes this data to the appropriate bank and sector. The write master also collaborates with the power-gate control logic. Moreover, all sectors require only one address and data port that can be used for reading and writing. Moreover, the data ports of individual MRAMs can be accessed in parallel, thus, reducing the read/write latency.

The motion estimation engine requests a particular predictor from the reference frame. Its request is handled by the reference read master. Actually, motion estimation engine provides the row and column addresses of the reference frame. Read master appropriately translates these addresses to banks and sectors and cross-checks if the sector is turned ON. If not, then a turn ON request to the power-gate control unit is generated, resulting in latency. Afterwards, a row of the predictor is written to the row buffer which forwards this data to the motion estimation's SAD accelerators. Note that the concept of search window is now replaced by the full video frame buffer and there is no need to fill a memory structure repeatedly with frame data (see Figure 2-8). Thus, in the following, a search window will actually mean a region in the on-chip frame buffer.

For the AMBER architecture, the write latency of an MRAM sector should be lesser than or equal to the average motion estimation computational time. Thus, for $w \times h$ dimensions of frames and f_p frames per second, the write latency of on CTU, $t_{w,m,CTU}$ must hold the following:

$$t_{w,m,CTU} \leq (b_w \times b_h) \Big/ (w \times h \times f_p) \tag{5-16}$$

The dynamic power in the AMBER architecture is the power consumed by:

- Writing into CTU SRAM twice
- Reading from CTU SRAM twice
- Writing to MRAM buffers
- Reading from MRAM buffers

Therefore, we can write the total dynamic power p_{dyn} as:

$$p_{dyn} = w \times h \times f_p \times \left(e_{dyn,w,m} + 2e_{dyn,w,s}\right) \tag{5-17}$$

Here, $e_{dyn,w,m}$ and $e_{dyn,w,s}$ are the dynamic write energies of MRAM and SRAM respectively. The total leakage power p_{leak} is then:

$$p_{leak} = p_{leak,m} \times \left(b_w \times h + s_w \times h\right) + 2\, p_{leak,s} \times b_w \times b_h \tag{5-18}$$

Here, $p_{leak,m}$ and $p_{leak,s}$ are the leakage powers of MRAM and SRAM respectively. While the reason for SRAM leakage power term is obvious, for the MRAM, it is more involved. During write phase, only one sector is turned ON and during read phase, a set of sectors is turned ON, denoted by s_w, the width of the search window (details in Section 2.2.1.2).

The advantages of on-chip MRAM reference buffers are now compared with the usual search window based external memory fetching. With low-leakage and high capacity, these MRAM on-chip video buffers are feasible. Given n_r reference frames, for a current monochromatic/luminance frame, and search window read factor r_f, the total reads and writes directly from the external memory results in the total pixels accessed equal to:

$$w \times h \times 8 \times f_p \times r_f \times n_r + 2 \times w \times h \times 8 \times f_p \tag{5-19}$$

Thus, using the MRAM on-chip reference buffers, total number of external memory accesses is reduced by the factor $2+r_f n_r$ which is more than three times external memory access savings. Moreover, on-chip memory latency is much better as compared to the off-chip latency. Furthermore, the contention on the off-chip external memory bus is reduced.

Again, consider that n_r reference frames are stored in the external DRAM memory. The total leakage power of the can be therefore written as:

$$p_{leak} = p_{leak,d} \times w \times h \times n_r \quad (5\text{-}20)$$

Here, $p_{leak,d}$ is the leakage power of DRAM. On the other hand, the MRAM buffers can be switched OFF to dissipate no power (see the leakage of MRAM reference buffers in Equation (5-18)). In fact, AMBER will only turn ON set of sectors at a time, depending upon the predictor location. Thus, the total leakage power is reduced.

Note that power-gating is only advantageous for MRAM reference buffers and not for DRAM or on-chip SRAM. DRAM and SRAM will lose their contents as soon as the power is cutoff. Moreover, once the design is fixed (chip is fabricated), the size of the search window cannot be altered. A scheme requiring smaller search windows (e.g., low resolution, low motion video coding) employing DRAMs or SRAMs reference buffers will consume the same leakage power, regardless of the search window size. However, this is not the case with the proposed MRAM based memory subsystem. Due to the non-volatility property of the MRAM, reference frames will still be available, once the MRAM is turned back ON. Thus, AMBER adapts the leakage power consumption at runtime to save energy.

5.4.3 MRAM Reference Buffers Energy Management

In the "sliding" search window based video block-matching architecture, note that whether or not the current CTU accesses all the data in the search window, the next CTUs may use this data. Therefore, search window data cannot be discarded. Moreover, from Figure 3-12, we notice that most of the search window is wasted. Considering this knowledge in the proposed hybrid memory architecture, motion estimation algorithm accesses only a small percentage of sectors. However, since on-chip full frame buffers are employed, a search window is actually the full frame (or a full bank). Therefore, multiple writes to this search memory are not required. This saves dynamic write energy. Since the search window in AMBER case is very large, it has a high leakage power associated with it. It is denoted in the previous section that the leakage energy of AMBER can be reduced (as compared to the external memory based reference buffers). This is achieved by appropriately power-gating the unused memory sectors. For a particular bank, the power gating control unit turns on a set of sectors between β_1 and β_2 (collectively represented as (β_1, β_2)) and power-gates the rest. The values (β_1, β_2) are predicted and adapted by analyzing the search window memory access patterns. Power-gate control unit actually controls the sleep transistor associated with a memory sector.

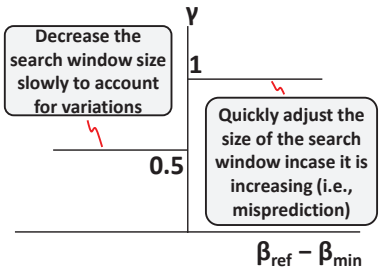
5.4.3.1 Memory Access Based Self-Organizing Map

The estimation of (β_1, β_2) is served by a small and energy-efficient Self-Organizing Map (SOM) [251], which keeps a record of the memory access pattern and updates its map whenever there is a change in the pattern search procedure.

Fast block-matching algorithms for the CTU in HEVC always follow a pattern. Therefore, some simplifications can be made for designing the SOM. SOM can be represented by a table with neurons as the keys as shown in Figure 5-23. Further, training the SOM is quite fast, as we do not need to test each input against every neuron, rather, the neurons can be updated in sequence. Each neuron of the SOM holds the minimum and maximum search window memory sector for the current CU in the form of a vector (β_{min}, β_{max}). This vector is called the weight of the neuron. Based upon the weight of the neuron, the power-gate control bits are set or reset. The power-gate unit powers ON the sectors between:

$$\beta_1 = CU_x - \beta_{min}; \qquad \beta_2 = CU_x + \beta_{max} \quad (5\text{-}21)$$

Here, CU_x is the horizontal location of the top-left pixel of CU under test. That is, only the sectors between β_1 and β_2 are turned ON. Additionally, note that there is a neuron for every CU of the CTU. From Equation (2-8), we see that a total of 85 neurons are required for a CTU of 64×64. The weight of the neuron is set during the training phase of the SOM. However, it is also updated during runtime if there is a change in the memory access pattern during runtime. That is the block-matching algorithm requests a memory sector β_{ref} to be turned ON. Then the weight setting formulas are given by:



Neuron	Weight
1	(β_{11}, β_{12})
2	(β_{21}, β_{22})
3	(β_{31}, β_{32})
⋮	⋮

Figure 5-23: Neuron weight update feedback and SOM table.

$$\begin{aligned} \beta_{min} &= \left\lfloor \beta_{min} + \gamma \left(\beta_{ref} - \beta_{min} \right) \right\rfloor \\ \beta_{max} &= \left\lceil \beta_{max} + \gamma \left(\beta_{ref} + b_w - \beta_{max} \right) \right\rceil \end{aligned} \tag{5-22}$$

The learning coefficient (γ) takes the value of 1 or 0.5 for simplicity in the implementation, as shown in Figure 5-23. If the search window size is increasing, a faster expansion of the search window is allowed, by keeping γ =1. Else, for compressing the search window, AMBER takes γ =0.5. In the formula above, b_w is the width of the CU under test.

Usually, the memory access pattern for motion estimation does not change throughout the HEVC encoding (reference software is used for experiments). Therefore, the offline-trained SOM does not change during the runtime for a given motion estimator. However, changing the memory access pattern will require the neurons to adjust first and latency will be encountered. Even with a threshold-based early termination of the motion estimation, AMBER still outperforms the search window approach because search window will always require pre-fetching new CTU column from the external memory on every CTU-iteration. Moreover, for slow-varying motion sequences with adaptive motion estimation, the number of MRAMs turned ON will reduce and hence, further reduction in the leakage power. It will be opposite for high-varying motion sequences.

5.4.4 System Computation Flow

Algorithm 12 shows the high-level algorithm of AMBER for HEVC. For the current CTU, we wait before it is completely written to the CTU SRAM and the banks are ready for reading (line 3). Afterwards, the Motion Estimation (ME) for the current CTU commences (line 5). Before the ME begins, we forecast the bits of power-gate control lines (lines 8 and 9) given by Power Gate Control Register PG. PG bits must be set to turn ON the specific MRAM memory sectors. Excess memories turned ON due to previous CU computations are also turned OFF (line 10). Once the block matching for the PUs of the current CU starts, it is checked whether correct MRAM sectors are turned ON (lines 15 and 16). In case a misprediction in the forecasting has occurred, we need to turn ON the MRAM sectors which were mistakenly kept OFF (line 17) and in parallel, we update the estimator (line 18 and Eq. 8). Afterwards, ME can start (line 19) and we repeat the procedure to the next location in the block-matching pattern for the current PU. Once the current CU is processed, it is sub-divided recursively into 4 equal CUs (line 22 and 23) and the process is repeated.

In the following, the impact of the wrong decision latency on the performance of HEVC is examined. For a frame with total n_{frm} CTUs, the total number of PUs searched is (see Equation (2-8)):

$$\xi = n_{frm} \times 1\,3 \times \sum_{i=0}^{\log_2 bw-3} 2^{2i} \tag{5-23}$$

Therefore, for a 30 FPS video, total number of PUs searched in 1 second is equal to 30ξ. Now consider we turn OFF memory forecasting and the wakeup time for a MRAM is supposedly ψ cycles, a total of $30\xi\psi$ cycles per second are wasted. For a CTU of size 64×64 of a Full-HD frame (1920×1088), it will incur an additional latency of about 16.9ψ Mega-cycles per second of video. Therefore, guessing the correct location of the predictor is important in managing the energy consumption of the prediction process.

5.5 Energy-Efficient SRAM Anti-Aging Circuits

This thesis targets aging analysis and configurable aging optimization of SRAM-based on-chip memories deployed in application-specific architectures (like camera-based video processing architectures). The proposed configurable design explores the tradeoff between aging resilience and the associated power overhead. In order to reduce the aging-rate of the SRAM video memories (Sections 2.3.2 and 3.3.3), we employ microarchitectural enhancements at the memory subsystem, which modulates or transduces the video data written to and read from the SRAM memory. Some of these circuits were also discussed in Section 3.3.3. In this section, the details of the proposed aging resilient architecture for a memory composed of 6T-SRAM based cells are given. The memory is assumed to have capacity sufficient enough to store a large chunk of data, e.g., multiple images/video frames. However, the configurable aging resiliency concepts are orthogonal to the number of memory ports and memory size. Figure 5-24 illustrates the overall architecture of the proposed aging-resilient memory. The basic operational steps are:

- Data is written to a FIFO, controlled by the FIFO Controller, which also provides appropriate valid signals (e.g. data valid) to the memory subsystem.
- The aging controller snoops the data from the data FIFO and generates appropriate control signals for the best aging resilience and power reduction tradeoff (see details in Section 5.5.3).
- The signals generated by the aging control configure the memory write transducer (MWT; see details in Section 5.5.1) to adapt input data samples before they are written to the memory. Specific bits of the data samples are selected for inversion by setting the appropriate enable signals of the Inverter Switches. In addition, the address of the data written to the memory is changed at runtime using the Write AGU (see details in Section 5.5.2), to fully utilize the memory address space and introduce stress-relaxation at the SRAM cells holding less-frequently changing data samples (e.g., pixels of static background regions in an image).
- Before the data is read by the application, it if readapted by the memory write transducer (MRT; an exact replica of MWT) and the logical address is appropriately converted to the physical address by the Read AGU.

In the following, the memory write modules are discussed in detail. The working principles of the memory read modules can be easily deduced as they perform the exact opposite operation of the writing modules.

5.5.1 Memory Write Transducer (MWT)

The MWT is used to invert specific bits of the raw data samples in order to toggle less-frequently changing data samples and release stress on the 6T-SRAM cells storing these bits. The data bits are grouped in pairs and each bit-pair can be configured separately. The higher order bit-pairs (containing bits 2-7) are passed through controlled Inverter Switches. Figure 5-24 shows that the first two bits (0 and 1) are not adapted. This is due to the fact that the first two LSB bits always have the lowest degree of stress, due to high degree of variation and hence a balanced duty cycle (as shown in the case study of Figure 3-15 (j)). Therefore, the proposed anti-aging architecture only specifies three Inverter Switches to control six MSB bits of a data sample. The input control lines (E) act as clock-gating signals to the registers R_2 and R_3 and as a "select" signal to the multiplexers M_0 and

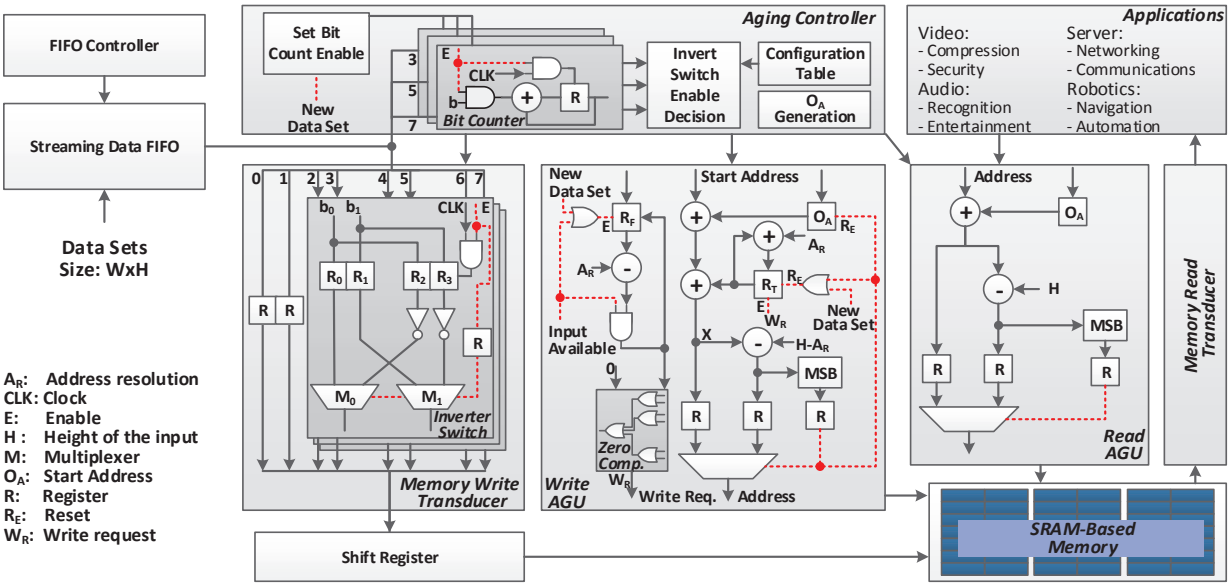


Figure 5-24: Overview of the proposed SRAM memory anti-aging architecture. Note the Memory Write Transducer, Write AGU, Read AGU and Memory Read Transducer connected to the SRAM. The aging controller configures these units by generating appropriate signals to adapt input data at read and write ports of the memory

M_1. All the registers store the original bits (b_0 and b_1). The registers R_0 and R_1 are directly connected to the multiplexers, whereas R_2 and R_3 are inverted and fed to the multiplexers. For every bit-pair, five 1-bit registers, two inverters and two 1-bit multiplexers are required. For example, for 8-bit data samples, a total of 15 1-bit registers, six inverters and six 1-bit multiplexers are required.

If the control signal E of an invert switch is high, registers R_2 and R_3 latch the bits b_0 and b_1 and thus, the inputs and outputs of the inverters are updated. Both input bits are inverted and the inverted bits are generated at the output. If the "enable" signal is low, bits b_0 and b_1 will be forwarded to the shift register unaltered. Therefore, no dynamic energy will be consumed by R_2 and R_3 and the inverters. The signal E is controlled by the aging controller.

5.5.2 Aging-Aware Address Generation Unit (AGU)

The Write AGU is responsible for selecting the appropriate memory partition (e.g., for writing an incoming video frame from a camera device) and generating addresses for writing the data words stored in the shift register. Selection of the appropriate frame memory partition for writing a complete video frame follows a round-robin approach. In addition, before overwriting a memory partition, it is required that frame memory partition is no more required by the executing application(s). An application signals this by requesting the address of a new data set (see Section 3.3.3 for details). A signal ("New data-set request" in Figure 5-24) prompts to deliver a new starting address to the Write AGU.

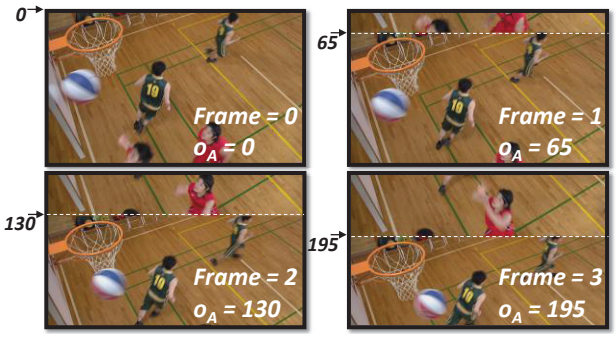


Figure 5-25: Write AGU frame writing scheme with o_A=65. The moving region (players and the basketball) are overwriting the static background (basketball court) with every frame.

As discussed earlier, less-frequently changing data will introduce the most amount of stress on the 6T-SRAM cells. If the most-frequently changed data words are identified, they can be distributed in memory in a spatial round-robin fashion. However, this requires further information from the application and additional analysis at

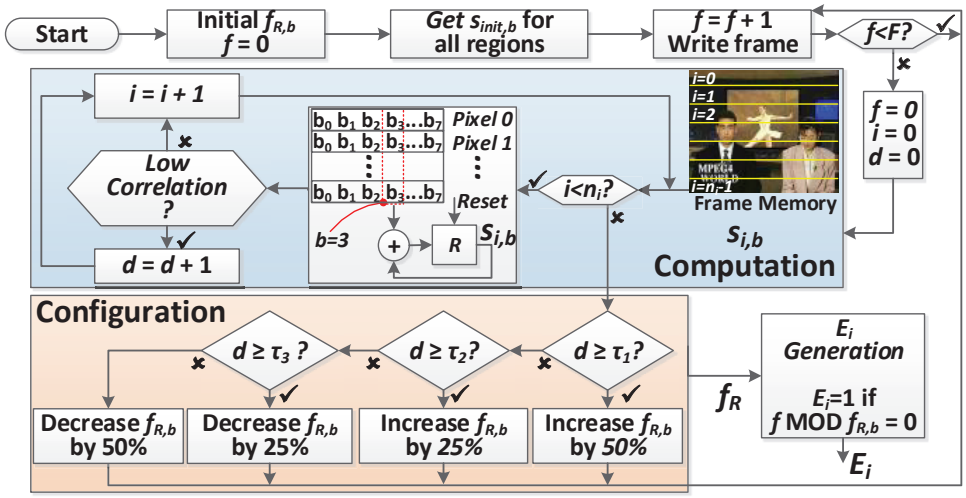


Figure 5-26: Inverter Switch enabling decision logic. In the proposed anti-aging memory, there are three such circuits, one for each Invert Switch

runtime, which is power- and area-wise inefficient. Therefore, a simple approach is suggested for introducing spatial aging resiliency. For every new data set written to the memory partition, the data set is circularly shifted. This corresponds to changing the starting address of the data set. For every new data set, an offset in the starting address (o_A) is given (by the aging controller) to the Write AGU as the starting address of the data set. With every data set, o_A is accumulated and the starting address is shifted. This will ensure that the memory partitions containing less-frequently changing data are interchanged with the more-frequently changing data at regular intervals, thus relieving stress from the SRAM cells holding bits of the less-frequently changing data. Figure 5-25 illustrates the example of video frame writing in the memory with o_A=65.

5.5.3 Aging Controller

As shown in Figure 5-24, the aging controller configures the control signals of the MWT and supplies the start addresses of the data set (e.g., the starting address of a video frame) in the Write AGU. Figure 5-26 shows the detailed flow of the proposed aging controller for enabling Inverter Switches. Note that such a controller exist for each Inverter Switch, therefore, three such controllers are used in the proposed SRAM anti-aging system. In order to generate the control signals for MWT, two decisions need to be made: (1) at what time instant the circuit should be activated, and (2) on which SRAM cells aging balancing should be applied.

Decision – 1: Activation of Aging Balancing Circuit at a Particular Time Instant: For this, the Inverter Switches are activated for a complete data set after specific time period, i.e., a certain number of video frames are written without adaptation and are processed by the application. For example, consider that a specific bit-plane of a data set *2i* is stored without inversion (E=0) and for the data set *2i+1*, it is stored with its bits inverted (E=1). This corresponds to the data adaptation rate (f_R) equal to 1, i.e., the respective bit-planes in every second video frame are inverted. Formally, f_R denotes the number of data sets stored in the memory without adaptation for every inverted data set. In the definition above, a bit-plane is defined as the collection of the bits at the same bit location, in all the samples of a data set. In case two data sets have correlation (e.g., two neighboring video frames), it is expected that the inversion of data set *2i+1* will overwrite most of the bit locations of data set *2i* with inverted bits. Thus, relieving stress on the SRAM cells and reducing the NBTI introduced aging. Note that there is a separate f_R for each Inverter Switch, and each Inverter Switch is independently activated. Additionally, f_R also plays an important role in deciding about the power consumption and aging rate of the SRAM memory. A high f_R will reduce the power consumption but will be less resilient to aging. On the other hand, a low f_R will reduce the aging rate, but will also increase the power penalty.

Decision – 2: Selecting SRAM Cells for Aging Balancing: At runtime, the MWT enable signals can be turned ON or OFF, depending upon the expected aging and the power constraint of the system, which can be generated based upon f_R. When all the enable signals are inactive (N=0), power penalty of the proposed approach is the lowest because no inversion takes place. This is because no toggling activity occurs at the inputs of the inverters as the associated registers are unchanged. However, the rate of SRAM aging is the highest because f_R number of data sets are to be written with adaptation and thus, increase the amount of stress a single transistor of a 6T cell will endure. Similarly, when all the enable signals are active (N=3), SRAM aging rate is the lowest at the cost of the highest power penalty. Basically, when only one enable signal must be active (N=1), the proposed anti-aging architecture inverts the two MSB bits (bit 6 and 7) as the SRAM cells storing these bit encounter the most stress (or aging). If the power configuration allows for two enable signals to be active (i.e. N=2), the four MSB bits (bits 4-7) are inverted.

The parameter N depends upon f_R of all Inverter Switches. If f_R of all Inverter Switches is selected such that adaptation is active for all Inverter Switches for the same data set, then N=3. Otherwise, if two Inverter Switches are active for the same data set, then N=2 etc. Selection of appropriate f_R is a control problem, which must be computed at runtime by analyzing the characteristics of the input data (i.e., computation of duty cycle for different bits). In the proposed approach, we specify a simple and efficient microarchitectural technique to estimate duty cycle online, as shown in Figure 5-26. This circuit is used to determine f_R for a single bit-plane. A bit-plane is divided into n_i parts, and the corresponding bits are accumulated for each part. For every part, if the number of 1s differs significantly than its previous stored value (depending upon the lower $s_{L,i,b}$ and upper $s_{H,i,b}$ threshold), the part has changed and a difference counter d is incremented. Afterwards, d is tested and f_R of the bit-plane is increased or decreased. The thresholds τ_1, τ_2 and τ_3 ($\tau_1 \geq \tau_2 \geq \tau_3$) determine the change in f_R and can be set at design time.

However, computation of duty cycle online incurs a power penalty. In the proposed approach, a counter logic is used to activate the duty cycle generation circuit. The set bit counter's register and the input bit to the adder are only updated if the enable signal is high. Thus, this will save dynamic power consumption. If online duty cycle computations are more frequent (finer control with smaller F), the power consumption of the proposed approach will be high and vice versa.

5.5.4 Generalization and Applicability

In general, the proposed aging-mitigation design methodology and architecture are orthogonal to the type of application and the low-level aging models. For instance, different data-parallel applications will also benefit from this. However, this may require several design optimizations to get the best power-efficient aging-mitigation design, for example:

- In addition to inverter, some rotation or swapping hardware blocks can also be integrated in the Memory Read and Write Transducers
- The values of controller parameters to adapt the aging-balancing also need to be re-evaluated.

This will require an application- and content-aware analysis, as performed for the camera-based application in this thesis. To illustrate the varying duty factor behavior of a non-frame-based application, an audio samples storage is evaluated, as discussed below.

In Figure 5-27, a 16 KHz, 16-bit Linear Pulse Code Modulated (LPCM) audio signal and the duty-cycle box-plot of the memory used for storing this signal are shown. Note that without using any aging balancing circuit, the duty cycle of all the bits will already be balanced for most of the cases, which is visible from the concentrated spread of the boxplot. This is due to the fact that audio signal swings around 0, and the number of negatives

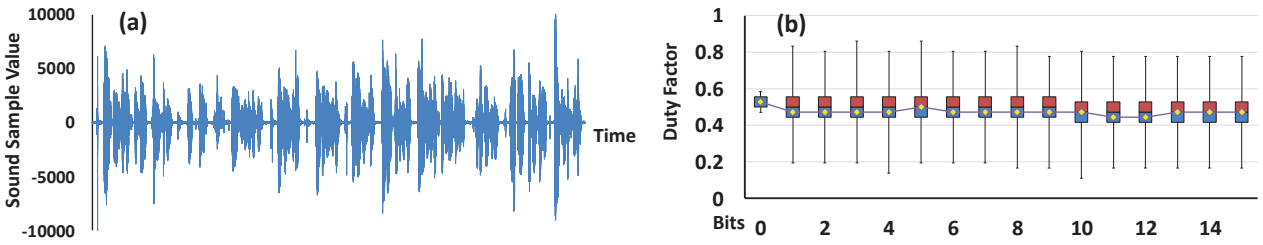


Figure 5-27: (a) A 16-bit/sample 16 KHz audio signal and (b) aging profile as box plot

(with MSB bits storing 1s) and the number of positives (with MSB bits storing 0s) is similar, unlike the video data. Moreover, the temporal correlation of audio data is low, meaning that it is highly probable that the new audio data which overwrites the previous one will have different characteristics. However, this is the not the case with video data, which will have high temporal correlation (for example, the background region, which will be static in many consequent frames), leading to biased duty cycles and therefore, higher stress on the 6T SRAM cells.

5.5.5 Sensitivity Analysis of SRAM Anti-Aging Circuits

The information on the duty cycles can be transformed to respective SNM over time using any available data mapping of duty cycle (Δ) to SNM degradation. However, the proposed approaches presented in this thesis are independent of such a mapping. In fact, any table which relates the Δ to SNM degradation can be used, because a balanced Δ will always result in lower SNM degradation. For evaluation purposes, we have defined another variable called aging map (δ) which connects the Δ ranges to respective values as shown in Table 5-4. As seen, if the duty cycle is highly balanced, ($0.4 \leq \Delta \leq 0.6$), the value of δ is low. This should be the purpose of an aging balancing approach (i.e., reduce δ as much as possible). Moreover, all the 6T SRAM cells with Δ more than 0.95 or less than 0.05 result in high SNM degradation and therefore are represented by a higher δ value. The δ for the complete video frame can therefore be represented in form of a δ histogram (e.g., see Figure 6-18). The information contained in this histogram can be concisely presented in form of a metric π given as:

Table 5-4: Duty cycle (Δ) to aging map

Duty Cycle (Δ)	Aging map (δ = |0.5-Δ|)
$\Delta \leq 0.05$ or $\Delta \geq 0.95$	0.45
$\Delta \leq 0.1$ or $\Delta \geq 0.9$	0.4
$\Delta \leq 0.2$ or $\Delta \geq 0.8$	0.3
$\Delta \leq 0.3$ or $\Delta \geq 0.7$	0.2
$\Delta \leq 0.4$ or $\Delta \geq 0.6$	0.1
$\Delta = 0.5$	0

$$\pi = \frac{\sum_{\forall bins}\left[\left(v_{bin} - v_{bin,min}\right) \times n_{bin}\right]}{\left(v_{bin,max} - v_{bin,min}\right) \times n_{samples}} \tag{5-24}$$

In this equation, v_{bin} is the value of a bin of the histogram (on x-axis, δ), n_{bin} is the number of values in the bin (on y-axis, total number of bits for the particular δ), and $v_{bin,min}$ and $v_{bin,max}$ are the minimum and maximum values of the bins, respectively. $n_{samples}$ corresponds to the total number of samples in the histogram (for SRAM storing an 8-bit per pixel video frame, this equals $w \times h \times 8$, i.e., the size of the frame memory). Thus, if the number of values in the bins are closer to $v_{bin,min}$ ($\delta = 0$ in our case), we would expect π closer to 0. Otherwise, π will have a larger value. Therefore, an aging resiliency scheme should strive to reduce π as much as possible. Further, in our calculation of π, we consider the distance between the local degradation and the least possible degradation (i.e., $v_{bin} - v_{bin,min}$).

The aging analysis of SRAM cells for different video sequences (details given in Table 6-4) is given in Table 5-5. The column "Base" denotes the amount of aging without using any MWT, i.e., without applying any aging balancing techniques. Since different video sequences lead to different amount of stress as a result of varying

distribution of "zeros" and "ones", the aging imbalance results in undesirable varying degradation of different SRAM cells. Sequences with large static structures in video frames (e.g. "Johnny") introduce the most amount of stress on the SRAM cells because of static sample values. These sequences are common for video security and communication applications. Camera panning and zooming sequences (like "BQTerrace", "FlowerVase" and "Keiba") usually have a low aging impact on SRAM cells. Largely static video sequences exhibit high aging due to less frequently changing data values (e.g. "Basketball", "ChinaSpeed", "FourPeople", and "Johnny").

In addition, the aging parameter after employing the MWTs given in Figure 3-15 are also tabulated. For a comparison with these MWTs, results for using N=0 (no Inverter Switch active) and only using the proposed Write AGU to circularly write the frames in the frame memory in the proposed architecture are also tabulated. This is achieved by having $o_A \neq 0$. Using three different values for o_A, we notice considerable aging balancing achieved by only adapting the start addresses of the video frames. Specifically, largely static sequences get the most benefit. However, an interesting observation for the "Johnny" sequence can be made where we notice low aging improvement as compared to a sequence with similar aging profile (i.e. "FourPeople"). This is due to the fact that the static regions in the "Johnny" video frames are similar throughout the height of the frame and the video samples of the new frame which overwrite the video samples of the previous frame have similar values, thus, not contributing to stress relaxation. Hence, inversion is a better option in such a case. Further, o_A adaptation results in better aging resiliency as compared to nibble-swapping, with negligible power penalty. Note that o_A is chosen as a prime number to make the cycle of offset to be as large as possible.

Table 5-5: Aging parameter (π in 10^{-2}) for different video sequences, with no inversion ($f_R=\infty$), no controller

Seq.	Base	f_R=1			N=0, $f_R=\infty$		
		Invert	Swap	Rotate	o_A=17	o_A=41	o_A=61
Basketball	37.9	3.9	32	3.7	14.5	14.4	14.4
BQTerrace	14.1	0.0	9.3	0.6	0.6	0.6	0.6
ChinaSpeed	38.9	0.0	30.1	22.1	21.3	21.3	21.3
FlowerVase	14.5	0.0	11.8	6.4	10.4	10.4	10.4
FourPeople	49.8	0.0	32.9	7.4	9.2	9.2	9.2
Johnny	52.6	0.0	34.8	5.6	25.5	25.5	25.5
Keiba	8.4	0.0	6.1	0.1	7.0	6.5	6.5
People	27.6	0.1	18.9	2.7	7.4	7.2	7.1
Traffic	42.6	0.1	29.4	8.0	8.8	8.9	8.5

Table 5-6: Aging parameter (π in 10^{-2}) for different video sequences, with o_A=0, no controller.

Seq.	Base	f_R=1			f_R=3			f_R=7		
		N=1	N=2	N=3	N=1	N=2	N=3	N=1	N=2	N=3
Keiba	8.4	1.1	0.0	0.0	3.9	2.9	2.9	5.9	5.2	5.2
Johnny	52.6	29.9	11.8	1.1	39.9	29.6	23.2	45.1	39.2	35.7
BQTerrace	14.1	4.3	0.4	0.0	8.0	5.2	4.9	11.0	9.5	9.3
Traffic	42.6	23.9	8.8	0.9	32.0	23.3	18.5	36.6	31.8	29.4

The impact of parameter f_R and N on aging for different video sequences is given in Table 5-6. Note that increasing f_R causes more frames to be inserted between two adapted frames. However, for static sequences like "Johnny", aging is accelerated due to increase in duty cycle bias. Sequences with camera panning, zooming and frequent scene changes exhibit lesser sensitivity to changing f_R, mainly because of the video memory overwritten continuously with changed video samples. For example, the sequence "Keiba" and "BQTerrace" exhibit lower sensitivity to increasing f_R. Similarly, introducing more inverters by enabling the control signals of the Inverter Switches in the MWT will largely balance the aging of video memory. For slow moving, static sequences like "Johnny" and "Traffic", N=3 results in a considerably better aging profile compared to N=2 or 1. Highly dynamic sequences can still achieve the same aging with N=2 or N=1.

Our experiments also reveal that using multiple frame memories result in almost the same aging balancing with the proposed scheme. This is because frames are highly correlated, and instead of always overwriting the frame memory with the next frame, writing the second or third frame results in nearly the same statistics.

Chapter 6 Experimental Results

The experimental evaluation of the approaches presented in Chapter 4 and Chapter 5 for the software and hardware layers of the video system are given in this chapter. The sensitivity analysis of the individual parts of the proposed algorithmic and architectural novelties is already outlined in their respective sections. Here, the major results and comparison with the state-of-the-art approaches is given. Major emphasis of the results is video encoding, specifically H.264/AVC and HEVC video encoders. It must be noted that these encoders have much higher complexity than many benchmark applications available in Parsec [252], MediaBench [253], Cosmic [254] and MiBench [255] benchmark suites. Further, since many contributions by this thesis are open-sourced, therefore, it is easy for other researchers to employ and compare their own enhancements.

The outline of the benchmarked evaluations are as follows. First, the parallelization of video application is evaluated by testing the proposed compute and application configurations. Afterwards, the resource budgeting (i.e., compute configuration) for multiple, multithreaded applications is carried out. This is followed by presenting the results of the computation offloading approaches. At the hardware layer, the hardware approaches for implementing highly efficient memory subsystem of the video system (including AMBER and SRAM anti-aging circuits) is evaluated.

6.1 Parallelization and Workload Balancing

This section provides detailed results regarding parallelization approaches and workload balancing schemes for multithreaded, video processing benchmarks.

6.1.1 Software Architecture and Simulation Setup

The simulation setup and design methodology of our proposed parallelization scheme for video applications is shown in Figure 6-1. A C++ library for generating the compute configuration is designed which generates the compute configuration (Section 4.2) and sets the application configuration (Section 4.3) given the initial configuration matrix *A* (Figure 4-7). This ensures that there is little or moderate effort from the application designer to incorporate the proposed approaches in the video application. This library also handles the frequency generation and frequency model adaptation (Section 4.2.5.2) at runtime. Similarly, another C++ library handles parallelization. A workload queue is implemented in this library, which is filled with the callback function (i.e., the tile processing function) and the associated arguments to that function. Afterwards, a start signal initiates the parallel processing where all the callback functions are executed. This library uses "pthread" API [256].

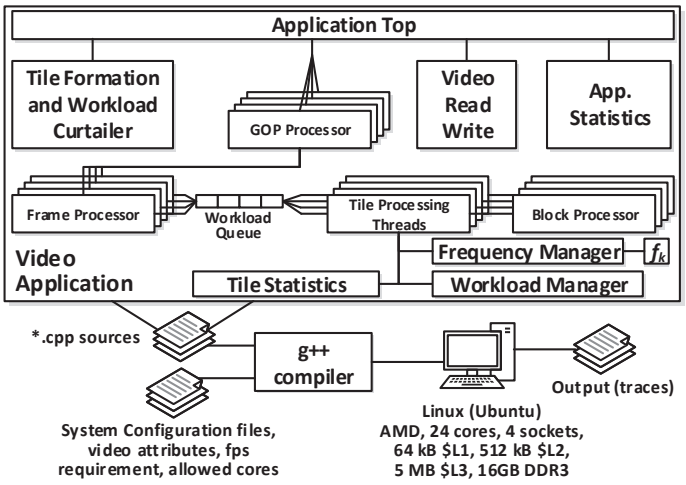


Figure 6-1: Simulation setup and multithreaded video application design

The applications can embed these libraries with minimal effort. For our application configuration case study of HEVC, the reference encoding software (named HM encoder [257]) does not provide parallelization and has a large memory footprint. Therefore, we have developed our in-house C++ based open-sourced multithreaded HEVC Intra encoder, ces265 (for Windows/Linux), where a single thread of ces265 is ~13× faster than HM

software (more information in Appendix B). Other open-source HEVC encoders (e.g. x265 [258]) are also available, however, they do not include the adaptations required and do not lend themselves to the extensions planned for this work. Further, due to a small memory footprint, no intrinsic and SIMD instructions, ces265 can be employed and tested for small embedded systems.

A 4×Six-Core AMD Opteron processor [259] (also called "Istanbul" processor) is used for experiments. For generating results, frequency scaling of all cores is disabled and using a fixed frequency, the number of cycles per block ($c_{k,\alpha}$, see Equation (4-12)) can be calculated given the computation time. Note that multiple programs are running on this computer, therefore, it is expected that the load on the computer fluctuates. This corresponds to the changing workload of the system running the application. The set of supported frequencies ($\boldsymbol{f}_{set}$) used in this work is given by:

$$\boldsymbol{f}_{set} = \{1.0, 1.2, \cdots, 3.0\}_{\text{GHz}} \tag{6-1}$$

The frequencies of the cores are used to estimate the power consumed by the application, by running the application on the Sniper many-core simulator [260] and McPAT [227]. Afterwards, an approach similar [3] [156] is used to estimate the final power consumption of the system. For our approach, we assume that no prior knowledge about the frequency estimation model constants $\boldsymbol{\omega}_k$ and estimation-error covariance matrix E is provided (see Equation (4-12)). Hence, at start of encoding, all elements of $\boldsymbol{\omega}_k$ are arbitrary chosen and E is initialized with $999I$ where I is an identity matrix.

6.1.2 Compute and Application Configuration for Uniform Tiling

In Table 6-1, different test sequences with varying dimensions, the number of given (r_{tot}), actual used cores (k_{tot}) after frequency estimation model stabilization, and average time per frame (t_{frm}) are tabulated using the HEVC application. As notice, k_{tot} becomes $\leq r_{tot}$ after model stabilization, denoting that the proposed RLS scheme will determine the correct number of parallel tasks/cores of a many-core system to sustain the workload, irrespective of initial settings.

Table 6-1: Given (r_{tot}) and used (k_{tot}) number of cores, and average time per frame (t_{avg}, in msec) for different video sequences, using f_p=5.

Sequence Number	Sequence	$w \times h$	Cores Given (r_{tot})	Cores Used (k_{tot})	t_{frm} [msec]
A	*Ballroom*	640×480	4	2	180
B	*BQMall*	832×480	8	3	197
C	*BQTerrace*	1920×1080	24	16	206
D	*ChinaSpeed*	1024×768	8	3	191
E	*FourPeople*	1280×720	12	6	193
F	*Johnny*	1280×720	12	6	193
G	*Keiba*	832×480	4	2	183
H	*RaceHorses*	832×480	16	2	177
				Average	**190**

Power Savings: Figure 6-2 denotes the power and video quality comparison of the optional runtime adaptation of the workload (application configuration in Section 4.3) using f_p=5 and ε=0.05 or 5% tolerance to the compressed output bytes. As seen in Figure 6-2 (a), significant power savings (up to 42.48%, average 39.21%) are obtained if the proposed workload tuning is active at runtime. This shows that our proposed application configuration by leveraging the application knowledge results in high power savings. For this graph, the power savings are computed by using the formula:

$$\text{PowSave}[\%] = \left(1 - Pow_{AppConfig} \,/\, Pow_{NoAppConfig}\right) \times 100\% \tag{6-2}$$

Video Quality Comparison: For the sequences shown in Table 6-1, Figure 6-2 (b,c) also present the BD-Rate and BD-PSNR [59], with and without application configuration. BD-Rate denotes the percentage increase in the average bit-rate compared to an anchor encoder, and BD-PSNR denotes the reduction in PSNR (in dB) against an anchor encoder. To compute BD-Rate and BD-PSNR, PSNR at a specific QP value, the following formula is used:

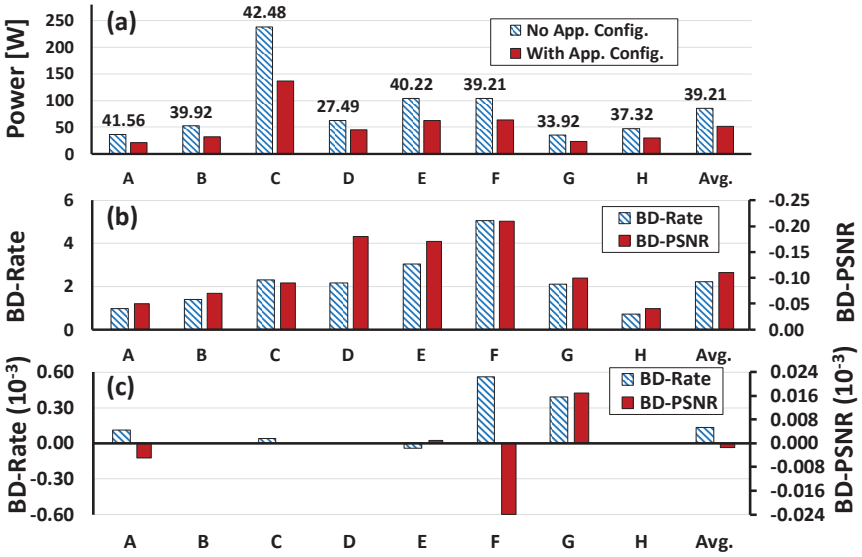


Figure 6-2: Using f_p=5, ε=0.05 or 5% for the video sequences given in Table 6-1, (a) power for application configuration, and BD-Rate and BD-PSNR for (b) without application configuration and (c) with application configuration

$$PSNR_{QP} = \left(4 \times PSNR_{Y} + PSNR_{CB} + PSNR_{CR}\right) / 6 \qquad (6\text{-}3)$$

For the case of Figure 6-2 (b) using no application configuration, the anchor encoder is the baseline encoder, using only a single tile per frame. However, this single-tile encoder is unrealistic as it is not possible to support large HEVC workload by using a single core. Therefore, the BD-Rate and BD-PSNR for the encoder of Figure 6-2 (b) only show the unavoidable overhead that must be paid while satisfying the throughput constraints (frame rate), i.e., quality degradation due to mandatory tiling. On average, BD-Rate increases by 2.22% and BD-PSNR degrades by 0.11dB. For Figure 6-2 (c), i.e., using application configuration, the anchor encoder is the multi-tile encoder used for generating the plots in Figure 6-2 (b). The degradation in bit-rate and PSNR shown here accounts for the degradation produced due to workload tuning (Section 4.3.1). However, note that the encoder give in Figure 6-2 (c) has both the BD-Rate and BD-PSNR expressed in 10^{-3}, which means that there is negligible video quality degradation produced by our proposed workload tuning scheme. On average, the BD-Rate is reduced by 0.13×10^{-3} % and BD-PSNR reduces by 1.4×10^{-6} dB.

Comparison with State-of-the-Art: Figure 6-3 compares the power-efficiency of the proposed scheme and that presented in [1]. In [1], authors maximize the throughput of a divisible workload and adapt the number of cores to process this workload, on a many-core system, whereas our approach tries to meet the application's deadline. Therefore, Figure 6-3 compares the FPS supported per unit of power (FPS per watt) for both schemes. As seen, for sequences given in Table 6-1, our proposed scheme results in significant improvements. For this analysis, the power consumption and FPS (reciprocal of average time to process a frame) are taken after the first retiling event. On average, our approach increases the FPS per watt metric by 19.20% compared to [1].

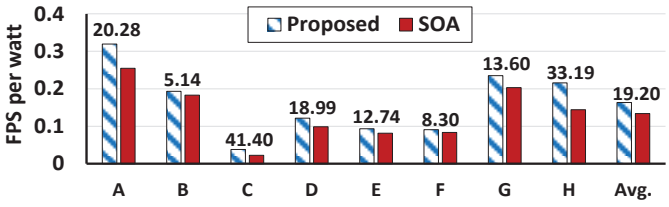


Figure 6-3: Comparing the power-efficiency (fps per watt) of the proposed workload balancing approach employing compute and application configuration compared to State-of-the-Art (SOA) [1]. The percentage improvement of our scheme is written on top of the bars.

Runtime System Dynamics: Figure 6-4 plots the power, time consumption (t_k) and frequencies (f_k) of all cores. For these experiments, $1/f_p$=1/5=200msec, ε=0.05 and z=8 (see Sections 4.2 and 4.3 for more information). Since in the first few epochs, the frequency estimation function is stabilizing (see Section 4.2.5.2),

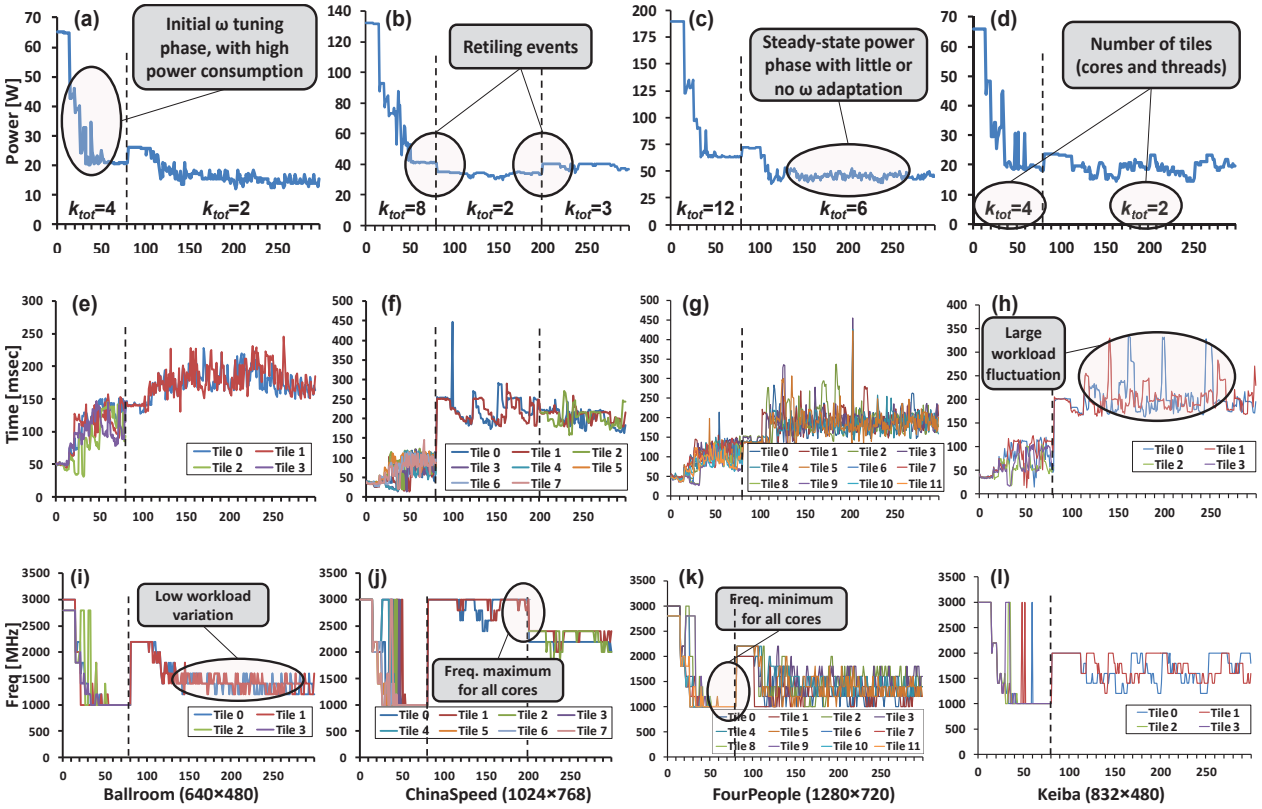


Figure 6-4: (a-d) Power [W], (e-h) time per tile [msec] and (i-l) frequency per tile [MHz] for different video sequences, using f_p=5 and ε=0.05 or 5%

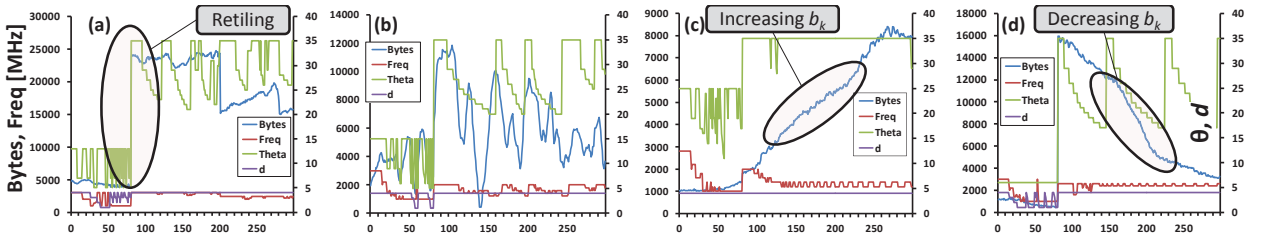


Figure 6-5: HEVC Frequency and workload tuning with the output bytes, for (a) Tile 1 of "ChinaSpeed", (b) Tile 0 of "Keiba", (c) Tile 0 and (d) Tile 7 of "BQTerrace"

therefore, the number of subtasks/cores (k_{tot}) and the initial estimated frequency is off by a large amount and t_k is unnecessarily small, resulting in high power consumption (Figure 6-4 (a-d)). The extra power consumption comes due to the frequency model's constants regulation via RLS filter at runtime. However, the advantages of determining the constants online outweigh the disadvantage of small power wastage that occur only in the initial processing stage. Our simulations show that only about 2.8µsec and 133µJ energy is spent in calling the RLS filter once after an epoch, which is negligible overhead when compared to processing a single frame ("Ballroom" sequence, single core, 858msec with 16.81J of energy). Moreover, this power wastage can be avoided by determining the constants offline and then tuning them online (via RLS). In addition, the overhead of retiling itself is negligible, because it is invoked only after 80 frames are encoded via HEVC. We note that f_k gradually moves towards a stable value, due to the adjustment of frequency estimation model constants ($\boldsymbol{\omega}_k$, Equation (4-12)). After retiling, the number of cores used for encoding (k_{tot}) and frequencies are adjusted, and thus, t_k is now considerably closer to $1/f_p$ as shown in Figure 6-4 (e-h). This results in a steady power consumption.

Note that retiling is done when frequency of all the cores are stuck at maximum or minimum (Figure 6-4 (j)

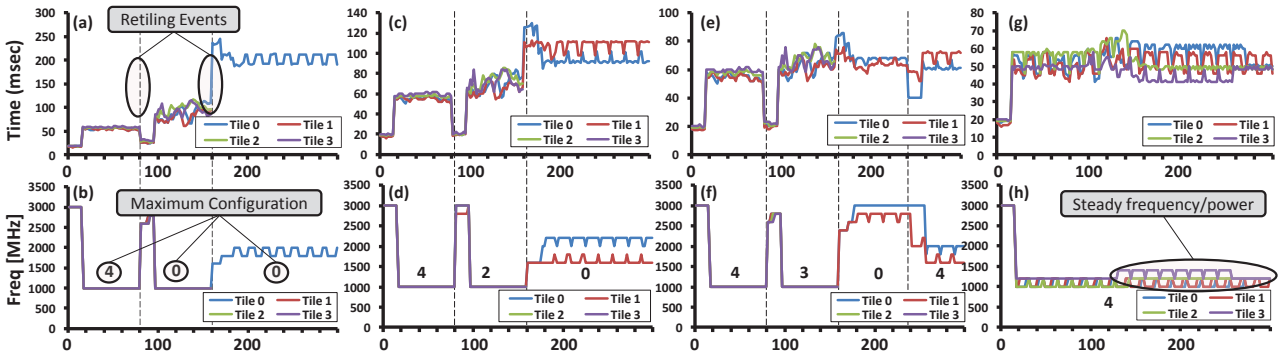


Figure 6-6: Cores' time, frequency and maximum configuration, per frame for frame interpolation with allowed number of cores (r_{tot}) = 4: (a,b) FPS = 5 (c,d) FPS = 10 (e,f) FPS = 15 (g,h) FPS = 20

and (k), see Section 4.2.6). An interesting case is demonstrated by the "ChinaSpeed" sequence, where retiling is done twice. Further, the workload of "Keiba" sequence is highly fluctuating and it results in large variations, both in frequency and power. For the "FourPeople" sequence, the average power before retiling is 99.08W and after retiling it reduces to 48.32W (an improvement of ~2.05×).

In Figure 6-5, f_k, θ_k, d_k and b_k of tiles in HEVC, for different sequences are shown. The optional tuning of workload (θ_k and d_k) due to the allowable increase in bitrate (ε=0.05) causes a change in the allocated frequencies of the cores, and in the size of compressed output b_k. Note that the large jump in b_k is due to retiling, because retiling results in a larger tile (and hence more b_k). Further, in Figure 6-5 (c), b_k *is* progressively increasing, thereby resulting in the increase of the workload (θ_k and d_k) and increasing amount of $f_{k,h}$ in the epoch (see Figure 4-8 (a)). Figure 6-5 (d) is the opposite case, and the workload is minimized, with increasing number of $f_{k,l}$ in the epoch.

In order to exhibit the impact of changing throughput requirement, Figure 6-6 demonstrates increasing FPS (f_p) of the frame interpolation application, only using compute configuration with 4 available cores (r_{tot}=4) and retiling test after every 80 frames. At start, the frequency estimation model does not apply to the current application scenario, and thus, the number of threads and the frequencies of the cores are higher while supporting a lower maximum workload ($\alpha_{k,m}$). As seen, the frequency estimation function is gradually stabilizing, and the execution time of the threads is steadily reaching to fulfill the required throughput demands. In addition, the maximum supportable workload is also increasing. Different fps requirement however do play a role, whereby we notice that Figure 6-6 (d) with highest FPS = 20 does not get its workload increased due to the limitation of resources.

6.1.3 Compute Configuration with Non-Uniform Tiling

Simulating the HEVC video encoder via Sniper many-core simulator [260] and McPAT [227], the power

Table 6-2: Core-Tile mapping, BD-Rate, BD-PSNR and power savings (ΔP) at 45nm, 2.6GHz for the uniform and non-uniform tiling scheme.

Seq.	FPS	Non-Uniform Tiling				Uniform Tiling				ΔP
		Cores	Tiles	BD-Rate	BD-PSNR	Cores	Tiles	BD-Rate	BD-PSNR	
Ballroom (640×480)	**20**	30	32	8.747	-0.4381	35	35	8.961	-0.449	+10.5
Flamenco (640×480)	**20**	30	32	12.078	-0.682	35	35	12.756	-0.719	+10.3
Vassar (640×480)	**15**	20	20	6.895	-0.238	20	20	6.904	-0.237	-0.30
Vassar (640×480)	**20**	30	32	9.862	-0.336	35	35	11.375	-0.387	+10.3
Keiba (832×480)	**15**	30	32	7.452	-0.382	35	35	9.069	-0.464	+14.0
RaceHorses (832×480)	**15**	30	32	2.050	-0.118	35	35	1.167	-0.067	+14.4
BasketballDrill (832×480)	**10**	17	20	6.120	-0.286	20	20	5.896	-0.276	+4.0
BasketballDrill (832×480)	**18**	33	40	10.524	-0.487	35	35	8.580	-0.400	-0.70
Average	**17**	**27.50**	**30**	**7.966**	**-0.371**	**31.25**	**31.25**	**8.088**	**-0.375**	**+7.81**

and resource utilization of both uniform and non-uniform tiling (Section 4.2.2) is reported in Table 6-2. This table presents BD-Rate, BD-PSNR and the total power savings in percentage for both tiling approaches. For this case, a many-core system with no DVFS and with the following model (derived via regression analysis) is used:

$$t = 29.65 - 1.28QP + 0.05n_{frm} + 3.38n \tag{6-4}$$

Here, t is the time (in msec) to process n CTUs of a video frame with n_{frm} total CTUs, encoded with the quantization parameter QP, at 2.6GHz. Using this equation and the ones given in (4-5)-(4-7), one can determine the tile structure. Afterwards, the bin-packing heuristic proposed in Figure 4-5 can be employed to determine the number of cores actually utilized.

As seen, various sequences with different frame rates are tested, which can mimic multiple encoding scenarios. Notice that non-uniform tiling, in average, saves three compute cores as compared to the uniform tiling. Further, the average video output quality is better, number of tiles is reduced and power savings are obtained when non-uniform tiling is used. Additionally, rising frame rate or increasing resolution of video sequences results in higher power savings by the non-uniform scheme. This is due to the fact that better workload balancing is obtained in case of non-uniform tiling approach.

6.1.4 Workload Balancing on Heterogeneous Platforms

Using the same heterogeneous cores and benchmarks given in Table 3-1 and Figure 3-11, a heterogeneous systems employing multiple compute nodes is used to evaluate the workload balancing approach given in Section 4.4. A multi-core heterogeneous system with four cores (i.e., r_{tot} = 4 with two "Tiny", one "Medium" and one "Large" core given in Table 3-1) is tested. A throughput constraint of $t_{i,max}$ = 30msec is set. The quality metric is defined as throughput-

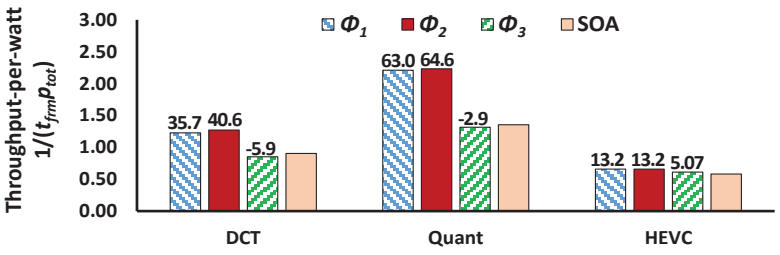


Figure 6-8: Average throughput per watt $1/(t_{frm} \times p_{tot})$ for different efficiency indices and State-of-the-Art (SOA) [2]. The percentage improvement against SOA is written on top of the bars.

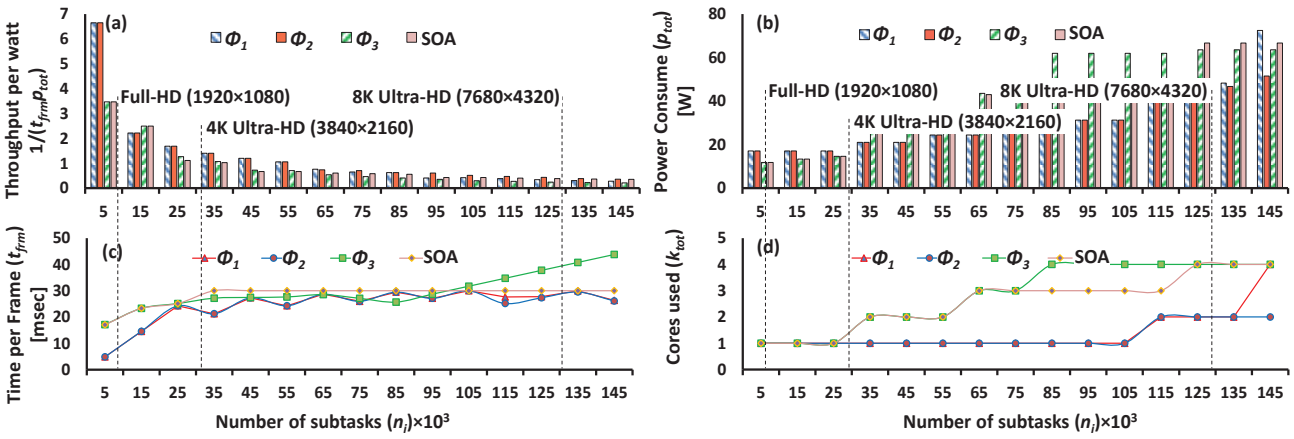


Figure 6-7: (a) Performance of the proposed and State-of-the-Art (SOA) [2] approaches for the "DCT" benchmark. The number of DCTs (or subtasks) for different image resolutions are portrayed on top of the graph. For the "Quant" benchmark, (b) power consumption, (c) time per frame and (d) number of cores used is given for different n_i, using our proposed efficiency indices and State-of-the-Art (SOA) [2] approach.

per-watt and is given by the relation $1/(t_{frm} \times p_{tot})$, where t_{frm} is the time to process a task and p_{tot} is the total power consumed by the heterogeneous multi-core system.

The average throughput-per-watt for multiple runs of the benchmarks, with varying number of subtasks (n_i) is given in Figure 6-8. As noticed, for these benchmarks, the efficiency indices $\phi_{k,1}$ and $\phi_{k,2}$ outperform the state-of-the-art load balancing scheme [2] which uses a bin-packing heuristic to distribute the load among cores. However, the efficiency index $\phi_{k,3}$ does not perform as well as the other indices. Hence, this shows that only power-aware load distribution will not result in high performance. It does performs well for the "HEVC" benchmark, because n_i for HEVC are considerably lower (from 1 to 10) as compared to the DCT and Quant benchmark (see Figure 6-7). When n_i increases considerably, the freedom to map these jobs also increases and the performance of the load balancing also improves [236]. Thus, compared to $\phi_{k,3}$, the performance of [2] improves for DCT and Quant benchmarks.

Figure 6-7 (a) breakdowns the throughput-per-watt performance for the DCT benchmark, for all efficiency indices and the approach proposed by [2], for increasing number of subtasks (i.e., n_i). The number of DCT operations performed for different image resolutions are also presented in this figure. As seen, the performance of efficiency index $\phi_{k,1}$ and $\phi_{k,2}$ is considerably better than $\phi_{k,3}$ and [2].

For the Quant benchmark, Figure 6-7 (b-d) shows the power, time (i.e., throughput) and number of cores actually used for processing (k_{tot}). As seen, for efficiency index $\phi_{k,1}$ and specially for $\phi_{k,2}$, the load is not distributed only based upon the power of the node. Index $\phi_{k,1}$ uses the combination of power and cycles consumed in processing the load, while $\phi_{k,2}$ uses only the number of cycles. Therefore, these two indices and [2] always result in throughput being satisfied ($t_{i,k} \leq t_{i,max}$ = 30 msec) as shown in Figure 6-7 (c). However, $\phi_{k,3}$ starts to miss the deadline once the load on the system increases considerably. Moreover, as shown in Figure 6-7 (d), the number of cores used for processing by $\phi_{k,1}$ and $\phi_{k,2}$ are also lesser than $\phi_{k,3}$ and [2].

6.2 Resource Budgeting

Following the discussion above, whereby power is minimized while meeting the throughput, this section provides experimental evaluation of resource (cores and power) budgeting for mixed multithreaded applications, while maximizing the throughput. For these experiments, the focus is on software level multicasting (see Section 2.2.2) of HEVC encoders. However, it should be noted that the proposed resource budgeting approach is also applicable to other parallelizable applications.

6.2.1 Experimental Setup

In our simulations, we have chosen epoch size as one second. Therefore, the Inter-cluster resource and power adaptation takes place after every fps_i frames have been processed. Following the methodology of [261], the 22nm results are scaled to 11nm using ITRS-provided technology scaling factor [262] to have representative Dark Silicon scenarios. We consider a many-core chip with number of cores k_{tot} = 24. In our implementation:

$$\boldsymbol{f}_{set,GHz} = \{1.0, 1.2, \cdots, 3.0\} \tag{6-5}$$

The power of a core at 3.0GHz is 4.94W. Hence, the chip's maximum power is 118.56W. The power exchange in Equation (4-37) uses ψ_1=8 and ψ_2=2 in Equation (4-38). These parameters are obtained using empirical analysis of HEVC video encoding. For other applications, similar application-specific numbers can be derived.

In order to evaluate our approach, we have used a trace-based simulation scheme. By disabling the frequency scaling on a real system (core-i7, 3.4 GHz, 16 GB RAM, Linux), HEVC encoding is executed and the actual time consumed for encoding each subtask (i.e., video tile) $\eta_{i,j}$ is written to a file. While simulating, this file is read to extract the time associated to process a data tile. For multi-channel encoding scenarios, online-available and widely used video sequences with diverse texture and motion characteristics utilized in our experiments are given in Table 6-3. Here, the term "set" refers to the set of video sequences which are encoded concurrently on the same chip. An encoder processes a single video sequence in the set. These experiments are performed using the open-source ces265 [44] HEVC video encoder. Equation (4-40) is used for determining the initial cluster size.

Table 6-3: Test video sequences sets. The video sequence format is: "Name (fps_{min}) (w×h)"

Set	*Video-1*	*Video-2*	*Video-3*	*Video-4*
Set-1	*Ballroom* (20) (640×480)	*Basketball* (10) (832×480)	*Exit* (20) (640×480)	*Vassar* (10) (640×480)
Set-2	*Ballroom* (10) (640×480)	*Football* (10) (352×288)	*Foreman* (20) (352×288)	*FourPeople* (5) (1280×720)
Set-3	*Bubbles* (10) (416×240)	*Coastguard* (20) (352×288)	*Keiba* (10) (832×480)	*RaceHorses* (5) (832×480)
Set-4	*BQMall* (5) (832×480)	*ChinaSpeed* (10) (1024×768)	*Flamenco* (10) (640×480)	*Vassar* (5) (640×480)

In this work, the scheme proposed in [3] is chosen as a comparison partner. Note that [3] does not consider resource allocation to the applications and assumes cluster sizes are available. Further, there is no adaptation of cluster size at runtime. In addition, the two-level, temporal power budgeting (among applications after epochs, and among threads after processing a data frame) proposed in this work is not implemented by [3]. Moreover, [3] allocates more power budget to the cluster with high relative performance to power ratio, i.e., the video encoder which generates the highest fps is allocated more power. This can result in the following case. If all the cores of the cluster are running at minimum frequency and a lower power is allocated to the cluster in the next epoch, it will not be possible to reduce the frequencies of the cores further. Therefore, the power budget of the complete chip (p_{tot}) will be exceeded. However, this is not the case with our approach.

6.2.2 Results and Discussion

Figure 6-9 shows the total achieved FPS by the proposed scheme and the scheme of [3] for different Dark Silicon (DS) configurations. Every bar shows the stacked FPS for each set of Table 6-3, for a specific power budget. When a larger power budget is available (e.g., at p_{tot}=100W or 15% DS), the frequencies of the cores have a higher degree of freedom and therefore, our proposed scheme achieves 30.86% higher FPS than [3]. However, reducing power budgets (e.g., when p_{tot}=42W or 65% DS) also reduces the degree of freedom, and therefore, the FPS improvement is not that high (still 15.6% higher than [3]).

Figure 6-10 and Figure 6-11 breakdowns cluster size k_i and Inter-cluster power p_i for all video encoders at 100W (~15% DS). As seen, encoders which process video sequences having a larger resolution and fps_{min} requirement requires more k_i and p_i allocation to the particular cluster. For example, Video-4 (Enc-4) of Set-2 is allocated higher k_i and p_i compared to the other encoders in the set. Note that the proposed algorithm will adapt the size of the cluster (i.e., include or exclude cores from a cluster). This is shown in the runtime adaptation of both k_i and p_i in these figures, where the clusters can exchange cores and power. Further, note that k_{tot} and p_{tot} of the system never exceed the thresholds ($k_{tot} \leq 24$ and $p_{tot} \leq 100$) and p_{tot} is immediately fully utilized unlike the control based scheme of [3].

The Intra-cluster power profiles of the encoders associated with videos of Set-4 at 100W (~15% DS) are shown in Figure 6-12. The impact of increasing the number threads/cores is shown in Figure 6-12 (a). Figure 6-11 (d) and Figure 6-12 (b) collectively show the result of Inter-cluster power exchange, whereby Encoder-2 transfers its power to Encoder-1. Figure 6-12 (c) shows the allocated power to the Encoder-3 (p_3) changing at runtime. Therefore, the allocated power to the individual threads ($p_{3,j}$) are also adapted as shown. The ripple in Figure 6-12 (clearly visible in Figure 6-12 (d)) is due to Intra-cluster power exchange (frequency adaptation)

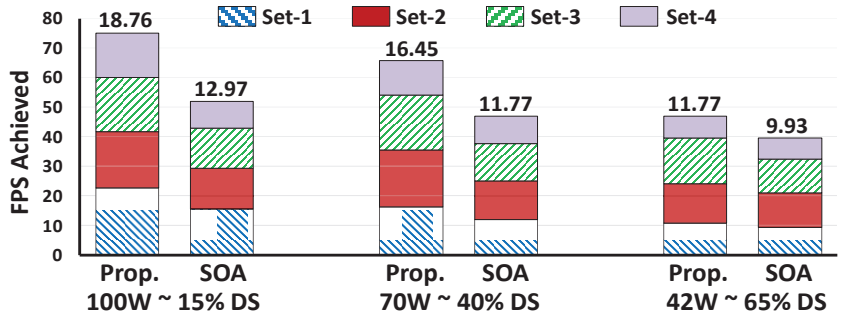


Figure 6-9: Fps achieved using proposed resource budgeting approach and State-of-the-Art (SOA) [3] for different amount power budgets (Dark Silicon, DS). The average fps per set is written on top of the bars.

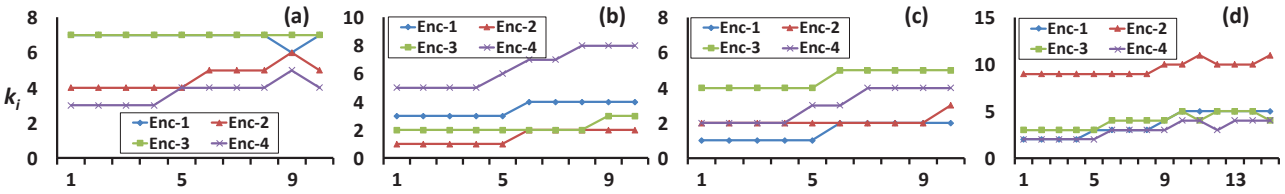


Figure 6-10: Runtime allocation (per epoch) of number of threads/cores for encoders at p_{tot}=100 Watts (~15% DS) of (a) Set-1, (b) Set-2, (c) Set-3,(d) Set-4.

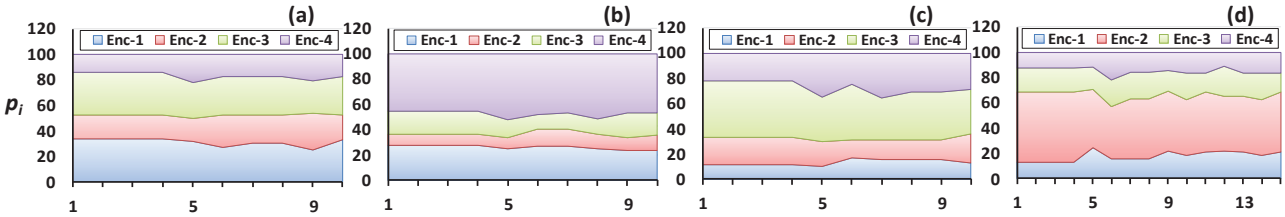


Figure 6-11: Stacked area plot for Inter-cluster, runtime power (in Watts) for encoders at p_{tot}=100 Watts, (~15% DS) of (a) Set-1, (b) Set-2, (c) Set-3, (d) Set-4

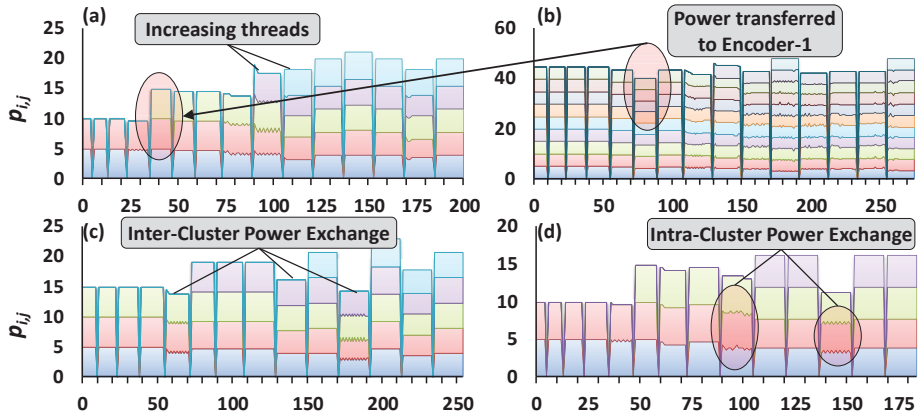


Figure 6-12: Intra-cluster power (watts) profile of (a) Enc-1, (b) Enc-2, (c) Enc-3 and (d) Enc-4 of Set-4 for each frame, shown as a stacked area plot

given in Algorithm 7.

6.3 Computation Offloading Evaluation

In case the resources (number of computation nodes and power) is not enough to sustain the workload, or the video processing device is constrained, computation offloading mechanisms (HDVC, discussed in Section

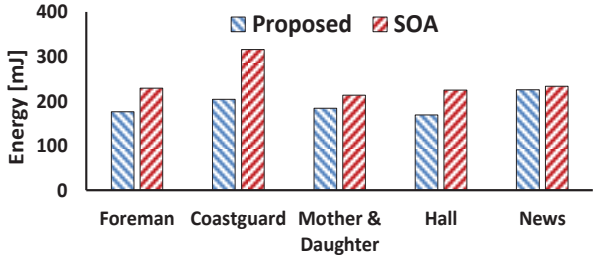


Figure 6-13: Energy consumption of the proposed scheme compared with the State-of-the-Art (SOA) [4]

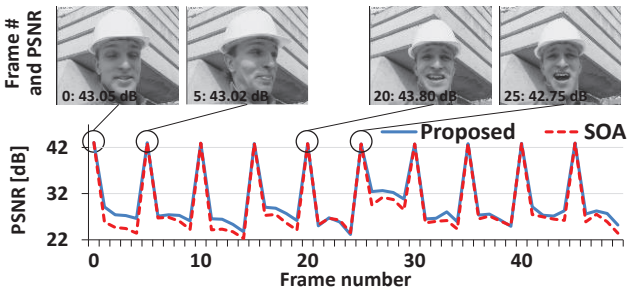


Figure 6-14: PSNR of the proposed scheme compared with the State-of-the-Art (SOA) [4], with same transmission energy and GOW size of 5 video frames (1 IF and 4 WF)

4.6) should be used. In the following, evaluation of HDVC system is presented.

6.3.1 Experimental Setup

In order to evaluate the energy efficiency of the proposed computation offloading approach, a complete HDVC system is developed with five different ME configuration classes at encoder and two ME configuration classes at decoder. The configuration of ME classes are borrowed from [67]. The Intra-frame encoder for I-frame coding is based on H.264/AVC standard. For W-frame coding and estimation, the algorithms for side information, parity-bits generation, and the quality matrix Q4 are based on [263]. For transmission energy estimation, the model of [244] is employed. For transmission energy estimation, the distance of 100 meters is assumed. Based on this model, the energy for one bit transmission is ~1µJ/bit. The computation energy results are obtained from the synthesis results using a 90nm technology as given in Table 4-4.

The proposed ROI-based energy quota distribution and control approach is compared with Raster-scan based scheme of [4]. For fairness of comparison, the same I-frame coding algorithm, same ME algorithm, and same SI generation unit is employed. The coding is performed under target PSNR constraints. The energy results illustrate the benefit of the proposed energy distribution and control scheme and ROI-based decisions. Note that the overhead of the proposed approach is included in the energy results.

6.3.2 Comparison with State-of-the-Art

Evaluations in Figure 6-13 illustrate the energy consumption comparison of our scheme with state-of-the-art for different video sequences. Figure 6-13 shows that the proposed offloading approach provides on average 20% energy reduction (maximum 25%) compared to state-of-the-art [4]. Higher savings are obtained for “Coastguard”, “Foreman” and “Hall” sequences, because they exhibit more motion. Note that the savings for “Mother & Daughter” and “News” sequences are relatively less due to limited motion content in the sequences. In such scenarios, adaptive ME already terminates earlier as it quickly determines the best match. This result also demonstrates that in case of high-motion and high-texture, the proposed approach exhibits a higher potential for energy savings compared to state-of-the-art.

Figure 6-14 depicts the quality (PSNR) comparison at frame-level between the two schemes for the cases of same transmission energy. For the frames number 0 and 5, the deviation between the two key-frames (i.e., Intra frames starting the GOW) is high, thus resulting in a higher variance between the two frames. Our proposed approach allocates more energy to the frame farther away from the Intra frame 0 and thus provides better PSNR compared to state-of-the-art. The raster-scan based scheme fails to allocate energy quota to important blocks, thus the results in PSNR loss. Our approach achieves 2 dB higher PSNR in these cases. For low variance key-frames (i.e. frames at 20 and 25 in Figure 6-14), the energy distribution among the W-frames is nearly constant and therefore, the PSNR of the W-frames between frame 20 and 25 is almost identical for both approaches.

Note that both approaches achieve the same quality at the I-frames, as we provide the same I-frame coding algorithms to both schemes and the difference only lies in the energy quota distribution and control for W-frames. Therefore, the PSNR for I-frames is same for both schemes, while PSNR difference only occurs for W-frames in this case. This is to ensure a high fairness in the comparison to highlight the benefits of only our approach.

6.4 Memory Subsystem Evaluation

This section deals with the experimental evaluation of the memory subsystem of a video processing system. Firstly, the impact of hybrid memories on the system performance will be provided. Afterwards, power-efficiency SRAM aging balancing approach is given.

6.4.1 AMBER: Hybrid Memories

This section provides evaluation reports of conjugating MRAM based NVMs with SRAM based memories. In order to test the AMBER approach of using hybrid memory structures, we augmented the HEVC reference software (HM-9.2 encoder [257]) to include the latency, energy and power numbers. The AMBER architecture is implemented via SRAMs and MRAMs for which the numbers are taken from Table 3-2for a 65nm technology [29]. Since the MRAM and SRAM read energy and latency numbers are similar and dependent upon the motion estimation algorithm, therefore, we do not include them in the results.

Figure 6-15 illustrates AMBER simulation setup. The inputs to the AMBER memory simulator are:

- Memory traces generated using the HEVC reference software,
- Memory architecture/organization, and
- Memory characteristics table that contains power, latency, and area of different memory types.

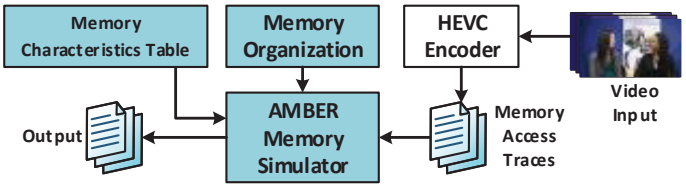


Figure 6-15: Hybrid memory simulation setup.

In Figure 6-16, the power consumption of the search window based approach and AMBER is presented for three different search window sizes, used in ME for encoding HEVC Inter frames. For evaluation, various test video sequences recommended by the JCT-VC [234] are used. Details about the video sequences are given in Table 6-1and Table 6-2. The small search window size is 129×129, medium search window size is 193×193 and for large search window, 257×257 size is chosen. Note that these search sizes produce best results according to the video frame dimensions. That is, a larger frame requires bigger search window and vice versa.

For only a single reference frame (Figure 6-16 (a)), the energy consumption of the search window approach is better. This is because only a single search window is used, and the total dynamic read and write power is small. An increase in frame dimensions causes more leakage power consumption in AMBER because the sector height increases. However, increasing size of the search window introduces more leakage and dynamic power

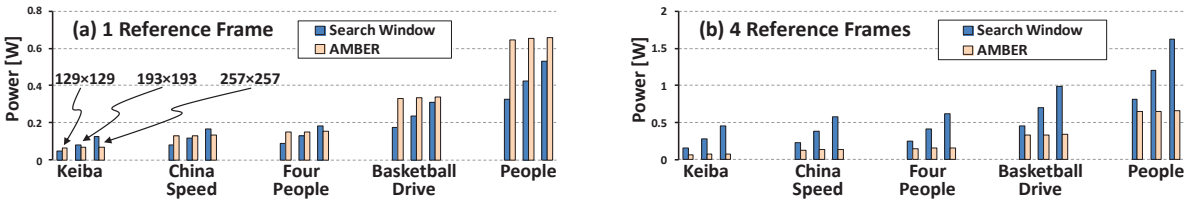


Figure 6-16: Comparing power consumption for the search window approach and AMBER, using (a) one reference and (b) four reference frames for ME of HEV Inter-encoding

consumption, and the power of search window based approach increases and surpasses that of AMBER.

On the other hand, AMBER produces better results for a bigger number of reference frames (Figure 6-16 (b)). The reason is that the power consumption of search window increase due to multiple reference frames being written and read. On average, AMBER based approach results in 43% energy savings.

6.4.2 SRAM Anti-Aging Circuits

This section provides the details results of estimating and analyzing the aging of SRAM based memories, using different state-of-the-art architectures and the proposed SRAM anti-aging architecture.

6.4.2.1 Experimental Setup

Hardware Synthesis and Aging Estimation: The experimental setup used for evaluations is given in Figure 6-17. We have implemented our aging resilient architecture in VHDL with different aging balancing circuits (inverter, bit swap, and bit rotate) as the basic building blocks and other architectural components like MWT, MRT, etc. The architecture is synthesized for a 65nm TSMC technology [264] (0.99 Volt, 25◦C junction temperature, FF corner) using Synopsys Design Compiler [265]. The gate-level simulations and functional verifications of the proposed architecture are performed using ModelSim [250], which is also used to generate the switching activity waveforms for power analysis. The aging model uses Table 5-4 presented in Section 3.3.3.

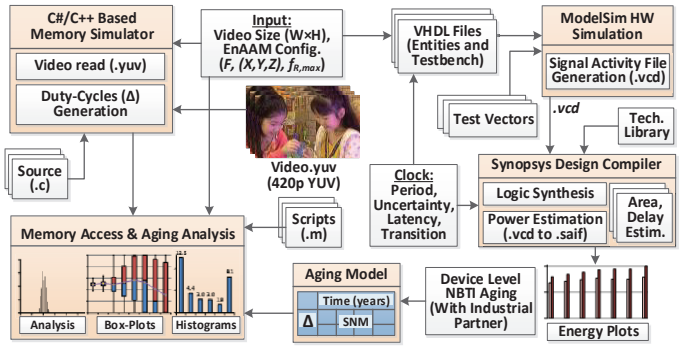
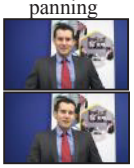


Figure 6-17: Experimental setup showing different hardware and software components for analyzing aging of SRAM circuits.

Plotting Aging Results: For fast analysis and visualization of memory aging, we have developed a GUI-based tool in C# which is made open-source [266]. This tool can accept user configurations like memory size, location of test data sets, total number of years the memory will be used etc. Using this tool, not only memory analysis and aging impacts can be visualized with ease, but also basic image and video processing applications (like filtering, color conversion, etc.) be executed for aging analysis. This tool automatically generates stressmaps, box plots, and duty cycle histograms for different input data sets (examples are shown in our motivational analysis; Section 3.3.3). For further information, see Appendix D.

Test Video Sequences: For our experiments, we employed various test video sequences recommended by the Joint Collaborative Team on Video Coding (JCT-VC) [31] and are available for download and testing [234, 267]. Some representative video sequences used in our experiments are presented in Table 6-4 along with their key attributes. These videos have diverse characteristics, and they can comprehensively represent various video

Table 6-4: Video sequences and their attributes

Name	Basketball	Flowervase	Keiba	FourPeople	Johnny	ChinaSpeed	BQTerrace	Traffic	People
Attributes *Resolution, Motion, Camera zooming/ panning, Frames*	832×480 Medium motion, no camera panning	832×480 Luminance changes, camera zooming in	832×480 Large motion, camera panning	1280×720 Very low motion, no camera panning	1280×720 Very low motion, no camera panning	1024×768 Large motion, no camera panning	1920×1080 Large static region, camera panning	2560×1600 Large static region, no camera panning	2560×1600 Medium motion, no camera panning
									

capture scenarios.

6.4.2.2 Results and Comparison with State-of-the-Art

In Figure 6-18, the percentage duty cycle histograms are plotted for different MWTs (given in Table 6-5) with a single frame memory for different test video sequences. Except for the "base" and "controller" case, these histograms are generated with $f_R = 1$, i.e., every second frame is adapted. The base case (Figure 6-18 (a)) has a high distribution of SRAM cells with a biased duty cycle. Majority of these cells are responsible for storing the higher order, low activity bits and thus exercise the largest amount of stress on the SRAM cells. For the inverter MWT (Figure 6-18 (g)), almost all SRAM cells have the best possible duty cycles. Comparing with [88, 201, 202], the usage of bit-inverter MWT in the proposed architecture will have the same aging impact. However, the aging balancing in [88, 201, 202] are achieved by employing additional hardware and architectural changes to SRAM cells. This requires the designer to only use customized SRAM memories with added enhancements. Further, the leakage energy consumed and the area-overhead by these SRAMs is much higher than proposed approach because each cell will have additional transistors associated with it.

Table 6-5: Comparison Partners

MWT/MRT	Description
Base	No MWT or MRT used
Swap	Swap Lower and upper nibbles of the complete frame [199]
Rotate	Rotate bits of every sample by 1 with every frame
***N*=1**	Proposed, with only Inverter Switch 6-7 always ON
***N*=3**	Proposed, with Inverter Switches 2-7 always ON
Controller	Proposed, with Inverter Switches controlled
Invert	Invert all bits of the complete frame [88, 201, 202]

The nibble-swap MWT (Figure 3-15 (c)) does not perform well as compared to the inverter and the rotator. For the proposed architecture without adaptive controller and Write AGU, and with only selected bits inverted (N=1 and N=3 in Figure 6-18 (d, e)), we notice that N=3 has almost the same impact on aging as the inverter. This is because bits 0 and 1 are self-balancing themselves while bits 2-7 are adapted (see the box plot in Figure 3-15 (j) for bits 0 and 1). However, N=1 will only invert bits 6 and 7, whereas from Figure 3-15 (g, h), we notice

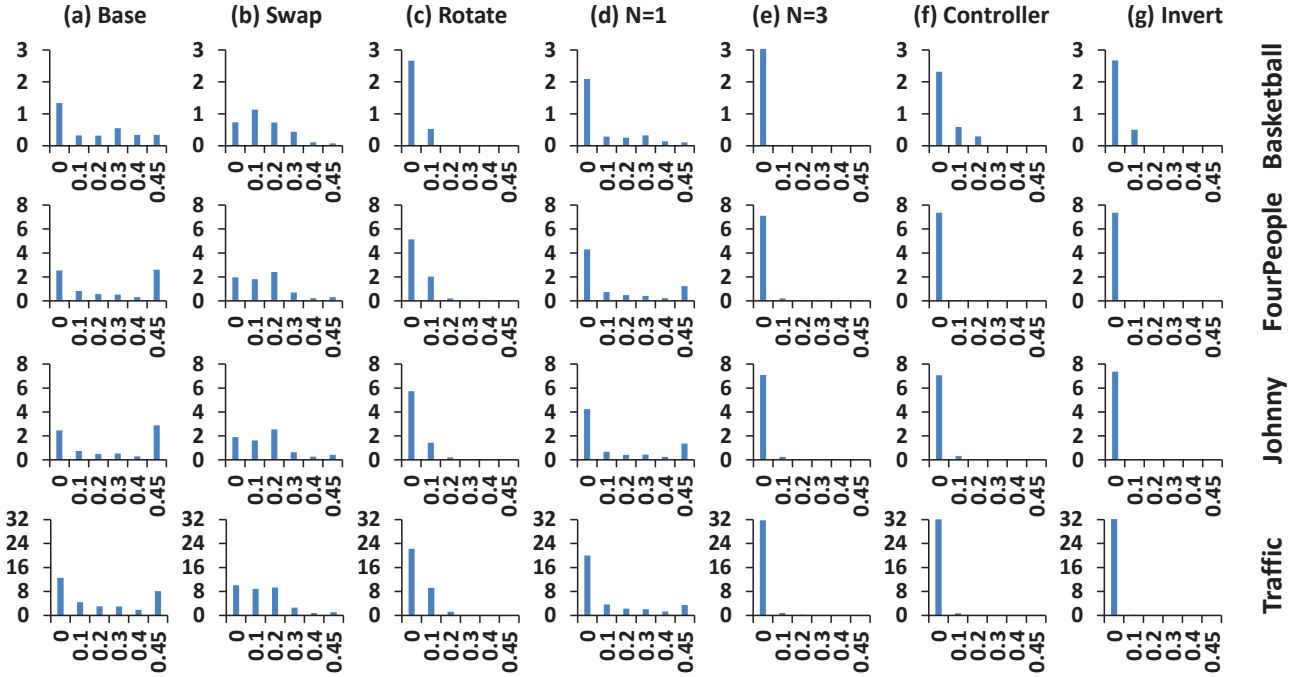


Figure 6-18: Histogram of duty cycle per bit, for a single frame memory partition with different comparison partners as given in Table 6-5. Values on x-axis denote the value δ as given in by Table 5-4. The y-axis denotes total number of bits in millions. The best aging balancing is achieved when the histogram is crowded towards 0. For all (excluding the base and adaptive controller case), every second frame is adapted ($f_R = 1$).

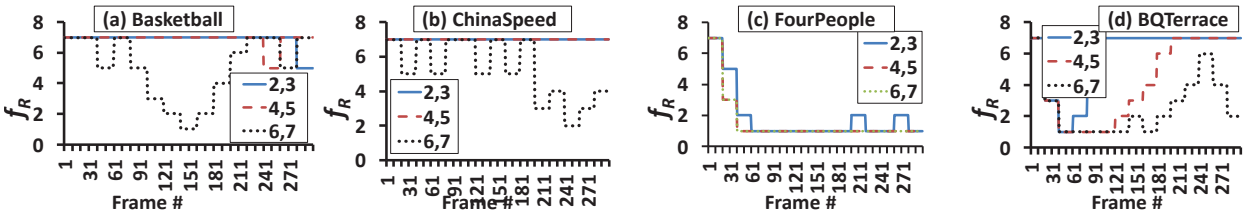
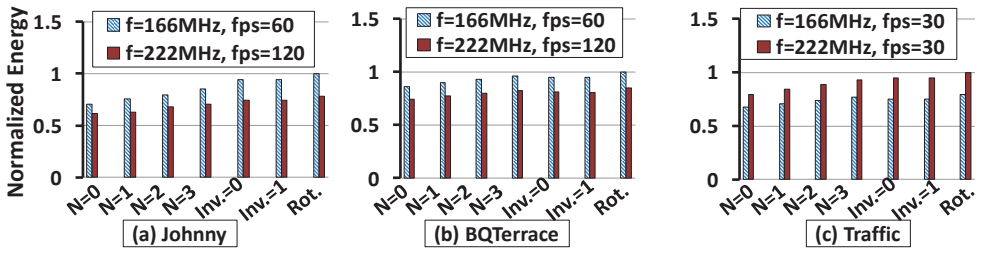


Figure 6-19: Runtime adaptation of f_R by the proposed aging controller with n_i=5, (τ_1, τ_2, τ_3) = (0.75n_i, 0.50n_i, 0.25n_i), F=20. Minimum f_R=1, maximum f_R=8. s_L=(1-ε)×s_{init} and s_H=(1+ε)×s_{init} where ε for bits (3,5,7) = (1/64, 1/32, 1/8).



MWT/MRT	Area (in Cells)
AROMA	516.23
Bit-Invert	520.91
Bit-Rotate	685.4

(d)

Figure 6-20: (a-c) Energy consumed per frame of MWTs and MRTs at different frequency and FPS configurations. (d) Total area (in cells) for different MWTs/MRTs. Inv.= 0 denotes that the Invert MWT/MRT is inactive while Inv.= 1 denotes active Invert MWT/MRT. Rot. denotes Rotate MWT/MRT

that bits 4 and 5 may also have highly biased duty cycles, which contributes to the worsening SNM degradation. Still, N=1 considerably balances duty cycle, as compared to the base and swap case.

For testing MWT with the adaptive controller (Figure 6-18 (f)), in these experiments we have chosen (τ_1, τ_2, τ_3) = (0.75n_i, 0.50n_i, 0.25n_i) where n_i =5 is the number of parts in which a bit-plane is divided (see Figure 5-26). The runtime adaptation of f_R by the proposed controller is shown in Figure 6-19 for different test sequences. For highly static sequences like "FourPeople", f_R of each Inverter Switch is lowered to the minimum possible f_R, in order to adaptively encounter the aging that such a sequence will induce. For high activity sequences or sequences with camera panning (like "BQTerrace"), f_R of each bit is increased, as it is not required to aggressively invert the frame. In addition, the aging impacts of the proposed schemes with multiple frame memories are almost identical.

The energy consumption per frame and area of different MWTs, for different running frequencies and FPS are given in Figure 6-20. This data is generated by annotating the input to MWTs using ModelSim simulation. The signal annotations are then queried by the Synopsys Design Compiler to estimate average signal activity on each pin and generate the leakage and dynamic power. As noticed, the proposed MWT with no invert switch active (N=0) consumes the smallest amount of energy, whereas the bit-rotate MWT consumes the largest amount of energy and area. From Figure 6-18, we also notice that aging balancing achieved by the bit-inverter and our proposed adaptive bit-inversion can easily surpass the performance by bit-rotate MWT. Therefore, it is reasonable to use inverters in MWTs for aging resiliency instead of bit-swapping and bit-rotation logic. Further, when the scheme presented in [199] is applied to SRAM memories, it requires testing for leading zeros, read/write of infrequently accessed memory addresses and additional information storage. On the contrary, our approach does not require such tests because it adaptively generates addresses to span the whole memory space in a circular fashion to introduce activity in low-activity cells. Compared to [200], the proposed approach does not require additional reads and writes to the SRAM memory, which itself consumes high dynamic energy.

Further, depending upon the application scenario and the allowable energy budget, our approach enables the application designer to select the best f_R and N configuration, suitable for the application. For example, from the Figure 6-20, a designer can achieve up to ~15% energy savings by turning off all the invert switches at the cost of SRAM aging. Therefore, a tradeoff between energy and SRAM aging can be established to select the best

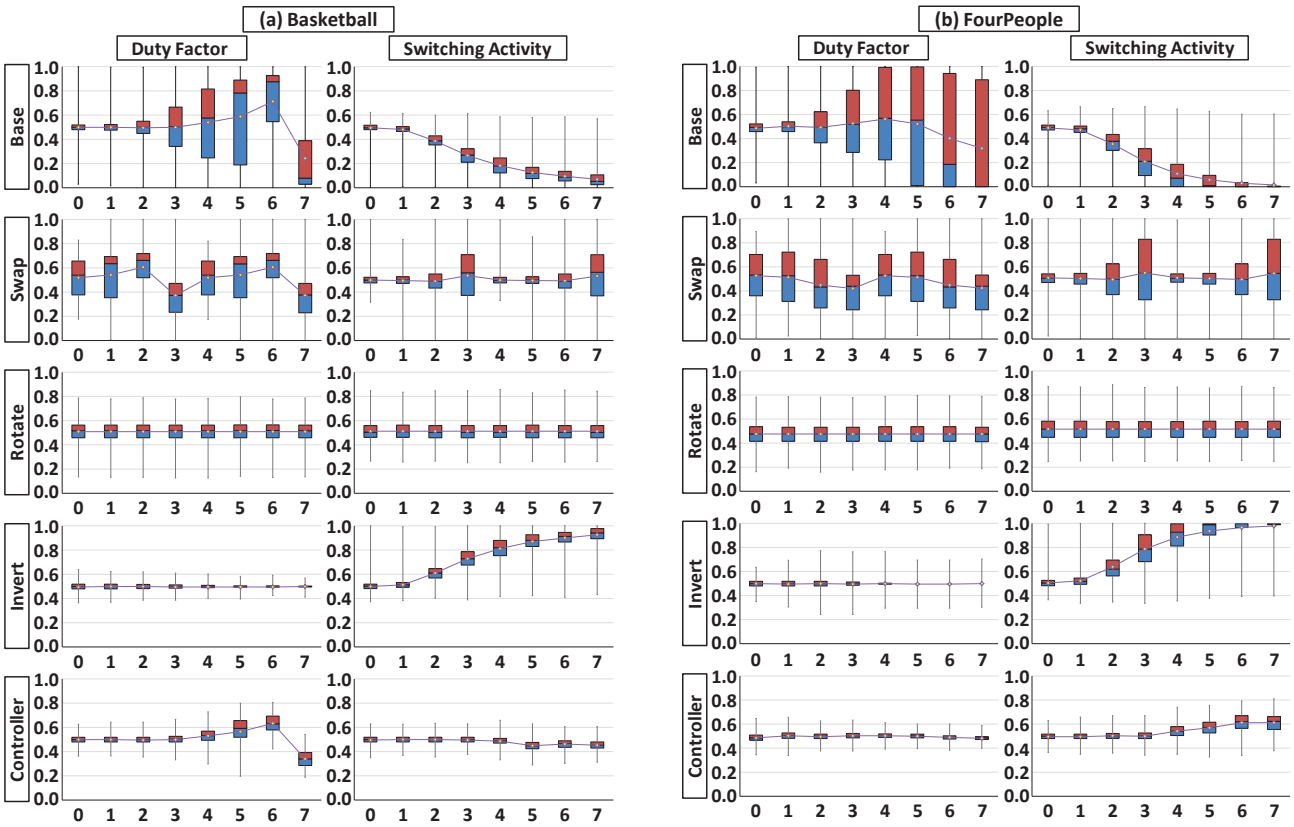


Figure 6-21: Duty-cycle and toggling statistics for (a) "Basketball" and (b) "Four People" video sequence, for different MWTs. The x-axis on all the graphs presents the bit-planes of the memory. The y-axis on the duty factor plots shows the average duty factor per bit-cell. The y-axis for the toggle graphs shows the box-plot of average toggling rate for each bit-cell, i.e., the average number of times a write to a cell results in a bit-flip. For best results, the duty factor box-plots should be crowded around 0.5, while the toggle box-plots should be crowded towards 0.

configuration.

6.4.2.3 HCI-Induced Aging

In this section, the duty-cycle (relevant for NBTI) and switching activity (relevant regarding HCI) are discussed. For this purpose, Figure 6-21 plots the duty-cycle and toggling rate for two video sequences, using different MWTs. The baseline and swap MWTs will incur the highest NBTI-induced aging, but with partial HCI-induced aging. The Rotate MWT balances the duty-cycle but with enlarged maximum duty-cycle value in the box plot: It also uplifts the toggling rate of the most significant bits (as shown by the concentration around 0.5-0.6 and a high maximum value), and hence the corresponding cells have a higher HCI-induced aging. The duty-cycle is considerably balanced by the inverter MWT thus encountering NBTI-induced aging. However, it also increases the toggling rate due to aggressive switching of every bit for every input, which will result in a higher HCI-induced aging.

In contrast to the state-of-the-art fixed techniques, the proposed anti-aging controller adapts the spatial and temporal granularity of applying the aging-balancing techniques. Therefore, it can provide improved distribution profiles for duty-cycles and switching activity across different bits.

Chapter 7 Conclusion and Future Outlook

This chapter presents a synopsis of this thesis. Afterwards, a brief prognosis of future extensions related to the complexity- and power-efficiency in relation to multimedia applications will be carried out.

7.1 Summary of the Thesis

Targeting multimedia systems under high throughput, resource and power constraints, this thesis provides efficient software/application level approaches and hardware/architectural level designs for the multimedia (video) system. The goal is to maximize the throughput-per-watt metric of the system, by addressing the modern design challenges. The challenges undertaken in this thesis include multimedia application parallelization on possibly heterogeneous systems, load balancing, resource (number of cores and power) budgeting and efficient design of the multimedia system's memory architecture. In a broad perspective, these problems can be combined and collectively represented as the Dark Silicon challenge for the next generation video processing systems.

7.1.1 Software Level Approaches

The proposed multimedia system parallelization approaches address the throughput demands of the video processing system and the attributes of the underlying hardware, while being power-efficient and providing high output video quality. The approaches presented here determine an appropriate **compute configuration**, whereby they divide a video frame into tiles, and then these tiles are packed and processed on the underlying cores, depending upon the workload characteristics and properties of the cores. For this purpose, uniform and non-uniform tiling approaches are employed [44, 45]. Uniform tiling assigns a video processing tile to a core, while non-uniform tiling may assign multiple tiles to a single core. Both these approaches are designed to balance the workload on the cores. Further, depending upon the workload of the associated tiles, the voltage-frequency levels of the computing cores are set to address the throughput demands (FPS) and provide high output video quality. The voltage-frequency levels are determined using a frequency estimation model. In case the frequency estimation model is inaccurate (or not derived), runtime adjustment/derivation of frequency model constants is carried out using a Recursive Least Square (RLS) filter.

Furthermore, appropriate **application configuration** is derived which will select the application's workload configuration (by tuning application parameters), while meeting a quality constraint set by the user. As a proof of concept, this thesis presents HEVC application configuration by selecting HEVC parameters and bargain computational complexity with the output video quality. Moreover, techniques to map these parameters appropriately are presented, which considerably reduce the degradation of output video quality. By appropriately reducing the number of Intra angular predictions used for HEVC encoding, 18% to 44% on average time savings is achieved against the state-of-the-art approach [123], for different configurations. Further, the depth of HEVC block subdivision (to determine the best PU configuration) is reduced, and hence up to ~57% time savings are obtained, with negligible video quality loss (-0.048dB compared to -0.118dB for the state-of-the-art approach [126]). Employing the proposed HEVC Intra angular and depth adaptation for application configuration on uniform tiling based compute configuration results in ~42% power savings, and ~19.2% power savings compared to a state-of-the-art approach [1]. On top of this, the non-uniform tiling results in additional ~7.8% power reduction compared to the uniform tiling.

In order to exploit the advantages of heterogeneity, **workload distribution and balancing on heterogeneous systems** is also discussed in this thesis. Here, the compute configuration selects the compute nodes and their frequencies in a manner to increase the throughput-per-watt of the system. For this, the workload distribution approach considers the power and computational efficiency of the underlying compute nodes (even

hardware accelerators), before distributing the workload of the task. In this thesis, three different efficiency indices are provided and an optimization problem is derived. The performance of these efficiency indices are tested using a heuristic. Compared to a state-of-the-art approach (which also derives the optimization problem and then proposes a heuristic) [2], the proposed approach can result in up to ~64% increase in the throughput-per-watt of the heterogeneous system. The approach in [2] allocates subtasks/video tiles to the cores in bin-packing fashion, and the core which results in minimum power is allocated the next tile.

Furthermore, considering multiple mixed multithreaded tasks/applications on the same hardware platform (e.g., as is the case with video multicasting), the **resource budgeting** problem becomes more challenging. This thesis outlines this problem and allocates compute configurations (i.e., cores and frequency/power) to individual clusters, used for processing a task. Each task gets a share of the computing cores and total system power (TDP), by considering the compute characteristics of the tasks, the available frequency levels of the underlying cores and the resource history of the tasks. Once the number of cores and power allocated to a cluster is determined, the power is divided among the cores of the cluster and adapted at runtime. The number of cores and power budgeted to the clusters is also adapted. When tested for HEVC multitasking, this approach results in up to ~18% additional throughput against a state-of-the-art resource budgeting scheme [3], for the same power utilization of the system.

In case the required workload cannot be fulfilled by the hardware platform, a part of the workload can be offloaded, which is considered for HDVC by the proposed **content-aware computation offloading** approach. To maximize the video quality, a hierarchical energy budgeting and control mechanism enables ~20% energy reduction compared to [4]. This energy reduction results from the intelligent energy distribution at group of frames, frame and block level while addressing the user constraint of processing time. A ROI within the video frame is extracted and selective/objective processing of the frame blocks at the encoder side is used to reduce the energy consumption of the system, and facilitates high quality reconstruction at the decoder side. Due to its high energy efficiency and adaptivity to provide higher quality for ROIs, the proposed approach enables HDVC in resource- and energy-constrained devices subjected to dynamically varying scenarios, involving complex motion and longer GOW lengths.

7.1.2 Hardware Level Approaches

The software level approaches are applicable to multi-/many-core systems, possibly heterogeneous and with hardware accelerators. To increase the throughput and power-efficiency of multimedia systems, the hardware level approaches discussed in this thesis target hardware accelerators' design, their allocation to tasks and the memory subsystem of the system.

In this regard, efficient architectures to support **video I/O and communication** between computing nodes on the hardware accelerators is discussed. The efficient video I/O implements Video Input Pipeline (VIP), video frame read/write to the external memory and provides the block based data to the video processing system, which can be a soft-core, co-processor or a hardware accelerator. This design is implemented and functionally verified for a real FPGA system (see Appendix C). For communication among nodes, an architecture is proposed which employs a register-file based custom interface. This interface is used to exchange data (e.g., memory pointers) and status signals (like "start", "done" etc.) among the nodes. The proposed interface is implemented on a multicore Nios-II based FPGA system, which can encode the video frames in HEVC format (see Section B.4).

In order to utilize the high power-efficiency of the hardware accelerators, an optimization problem is derived and a Nelder-Mead based heuristic is proposed to schedule the **shared hardware accelerator** for processing subtasks of individual tasks in a round-robin manner. The proposed heuristic determines the frequencies of the

computer cores, and the fraction of the subtasks of each task/application that must be offloaded to the hardware accelerator to fulfill the throughput requirement of all the tasks at reduced power consumption. Additionally, the soft-cores can be powered OFF once they offload their subtasks to the accelerator, reduce their temperature, and hence address the issues related to Dark Silicon. Moreover, a multicasting system based on H.264/AVC, employing a shared hardware accelerator is also implemented and functionally verified. The hardware accelerator is shared among different video encoders in a round-robin fashion. For this purpose, custom hardware scheduler and re-schedulers are used for utilizing the shared hardware accelerator. The functional verification of the design is carried out using an FPGA (see details in Appendix C).

Moreover, this thesis also presents novel, highly efficient **architectures for hardware accelerators**, specifically for video encoders (H.264/AVC and HEVC). For H.264/AVC, a low latency Intra-encoding loop is designed, which addresses multiple dependencies to permit high throughput of the encoding loop. Firstly, an area-efficient design of the transform module of H.264/AVC including (I)DCT and (I)Q is proposed for both AC and DC paths. This design incorporates a HT lookahead buffer that calculates the HT input coefficients at the 4×4 reorder stage instead of being generated by the DCT module in the AC path, effectively decoupling the AC and DC paths. Both (I)HT and (I)DCT architectures are merged and folded to reduce area. The one cycle latency by folding of (I)HT/(I)DCT is exploited in merging the Q/IQ stages together, thereby reducing the number of multipliers and barrel shifters. A hardware architecture of the H.264/AVC mode decision circuit is also provided, along with an efficient approach to determine the most probable Intra mode that will be selected by the mode decision module. This approach utilizes a novel, hardware based edge detection architecture, which determines the precedence order of the modes. This way, in case the cycle budget of the encoding loop is reduced (due to increased resolution or I/O stalls), the number of modes actually tested by the precedence order can be reduced. The state-of-the-art encoding loop architecture [5] utilizes 6.89K gates per pixel and 63.5mW per pixel, whereas 6.65K gates per pixel and 61.77mW per pixel are used by our proposed approach. Further, the proposed edge-based algorithm outperforms the Open Loop (OL) algorithm [129] for early estimation of the Intra mode. For HEVC Intra angular mode estimation, a hardware and software collaborative scheme is presented, which uses the software for control and hardware for computations. The hardware unit is composed of multiple sub-accelerators, and therefore, it is called a distributed hardware module. Each sub-accelerator can be turned OFF independently, and thus save power. This results in up to ~42% energy savings for HEVC Intra-encoding.

The hardware accelerators usually employ on-chip scratchpads for efficient computations. This thesis proposes an approach to use NVMs (MRAM) in conjunction with VMs (SRAM) as **hybrid scratchpad memories** for video applications, and thereby reduce the power consumption of the system. Large MRAM buffers are used for storing the video frames, while small SRAM buffer are used to (a) store the immediately fetched data from the external memory and (b) act as input and output buffers to the video processing hardware accelerators. Specifically, power consumption related to external memory access and leakage energy is reduced by adaptively turning ON the normally OFF MRAM memory sectors. The turning ON/OFF pattern is learned by an unsupervised learner (SOM). Compared to the widely utilized search window updating approach [61], the proposed hybrid scratchpad based methodology results in ~43% power savings.

Moreover, to enable reliable operation of video processing system over long deployment durations, this thesis analyzes the SRAM aging profiles for storing different video sequences. Based upon this analysis, power-efficient **SRAM anti-aging circuits** are proposed. Instead of the state-of-the-art approaches [88, 201, 202] for mitigating SRAM aging which (a) employ reading data from SRAM buffers, modifying that data and writing it back to the SRAM, or, (b) design custom SRAM cells which result in high leakage power, this thesis modifies video data on-the-fly while it is being written to the SRAM memory, using a write transducer. Further, when the application reads this data, the data is again transformed to its original state using read transducers. The microarchitecture of write and read transducers is designed such that the energy consumption is reduced while enabling high SRAM aging-mitigation. Specifically, certain bits of the video samples are adapted while others

are left unchanged. This decision in spatial and temporal locality is taken by the proposed aging controller. Moreover, the starting address for writing every new video frames is changed, and therefore, the data of the 6T SRAM cells is self-adapted.

7.2 Future Enhancements

The encouraging evaluations of proposed approaches put forth by this thesis at software and hardware abstraction levels demonstrate that high power-efficiency of a video system must jointly consider the co-design space of both the application and the hardware. Comparing with the state-of-the-art approaches, a considerable complexity, power and area improvements are achieved. However, there are some research frontiers which are not explored by this thesis but can be considered an effective means for resource and power-efficiency. Some of these future directions can be orthogonally added to proposed algorithm/architectures with minimal effort. In the following, a few of these possible future enhancements are outlined.

Approximate Computing: Approximate computing [268, 269, 270, 271, 272] exploits inherent resiliency of applications like image/video processing to gain power savings. Since for these applications, multiple outcomes are acceptable, the most power-efficient mechanism that still results in acceptable output quality is selected for processing. These mechanism can be inserted at the software level or at hardware level. At the software level, approaches using "iteration skipping" (similar to the one presented in Section 4.3) can be employed. However, the hardware level functional approximations have gathered considerable interest over the last decade. The main idea being, that by tolerating errors at the microarchitectural level, computing machines can reduce the accuracy of their computations (i.e., output is approximated) and thus save energy. Mainly, the focus of recent research about approximate computing has been to target small kernels for approximation to be implemented as hardware accelerators. Approximation schemes are employed at the basic arithmetic units like adders, multipliers etc. For example, [273, 274] explored transistor-level addition approximations. The authors of [275] proposed a runtime Accuracy Configurable Adder (ACA), while [276] implements efficient but approximate multipliers.

These microarchitectural approximations can be used to design power-efficient hardware architectures for the next generation multimedia systems. For example, the SAD unit (see Figure 2-8) can be implemented via approximate adders, the image/video processing filters can use approximate multipliers. Further, the hardware accelerator sharing approach presented in this thesis (Section 5.2) can also embed a relationship to account for the output quality/approximation level, and the hardware accelerator itself can consists of multiple approximation units. Similarly, the workload balancing and distribution approaches for heterogeneous systems can also utilize the approximation levels of these accelerators.

GPU Based Systems: GPUs are universally employed for image rendering and application benefiting from large degree of parallelism, where a small compute kernel is employed at pixel-level computations [18, 112, 119, 277]. GPUs are specifically suited for video output pipelines, like the ones used in videogames. A GPU is a heterogeneous chip-multiprocessor that can incorporate hundreds of parallel executing threads, and requires a specialized source. Another concept popular now-a-days is General Purpose Graphics Processing Unit (GPGPU), which can also perform non-specialized execution, typically conducted by a CPU.

In many cases, a GPU can be considered as a completely separate computing node (in conjunction with soft-cores) and the workload balancing and distribution schemes proposed by this thesis can be used here. Specifically, the scientific challenge to address will be division of a task into subtasks, such that the subtasks allocated to the GPU (or GPGPU) can be efficiently run on the GPU. Another important goal of workload distribution should be the power management of GPU, because GPU is one of the largest power consumer in the current desktop and laptop based systems. In addition, implementing the proposed workload balancing for

heterogeneous system on a real hardware, employing multiple GPUs, FPGAs and soft-cores, via OpenCL [278] is a scientific and engineering challenge.

Reliability and Workload Management: Adding the reliability constraints, the software and hardware layers of the multimedia system must be adapted in case the proposed approaches are to be used under soft- and/or hard-errors. For example, to process a given task, multiple computing nodes are used for Dual or Triple Modular Redundancy (DMR or TMR), the power consumption of the system will increase [279] and therefore, the resource budgeting needs to adapt. That is, the number of cores allocated to process a task and the power allocated to the task must consider the impact of reliability-aware processing. Similarly, the subtasks of the image/video processing application which needs reliability and protection needs to be identified, because usually, image/video applications are inherently resilient to errors in some parts of the computation pipeline.

Others: In addition, applications other than multimedia (databases, bio-technology, radar, signal processing for wireless communications etc.) can be tested and evaluated by the proposed parallelization, workload balancing, compute- and application-configurations, and resource budgeting. Similarly, hardware units for these applications can be designed and integrated into the heterogeneous processing system architecture proposed in Sections 5.1.4 and 5.3.2. The proposed approaches for video systems can be easily extended to 3D video processing paradigms. For example, 3D-HEVC [280] and MVC [281] used for encoding 3D video streams, Scalable Video Coding (SVC) [282] and Multiple Description Coding (MDC) [283] involving multiple layers of video content, can extend the proposed parallelization and workload balancing approaches, because the 3rd dimension of the video (i.e., the view or the layer) is added as a separate dimension to the problem. Mostly, this thesis targets the application and hardware layer for resource budgeting and optimizations, however, the kernel level optimizations and efficient utilization of available resources may result in high complexity and power savings. Moreover, it will be interesting to observe how these approaches can also be extended for reliable, hard real-time systems involving strict guarantees of fulfilling the timing constraints.

Appendix A Pseudo-Codes

A.1. Computation and Application Configuration

ParallelProcessing ( ):
Input:
Allocated number of cores r_{tot}; Allowable frequencies of the core $\boldsymbol{f}_{set}$; Image dimensions $w \times h$; Frame rate in frames per second f_p;
Output:
Processed video;

```
1.  NewTiling ← true;
2.  while(FramesRemaining){
3.      if(λ frames processed And NT( )){
4.          NewTiling ← true;}                    //New compute configuration
5.      if(NewTiling){
6.          (k_tot, f_m, α_m) ← GenerateTiles( ); NewTiling ← false;}
7.      ∀k ∈ {1 to k_tot tiles}{                  //For all tiles in parallel
8.          if(EpochFinished){
9.              α_k ← ApplicationConfig(k, α_m);
10.             f_k ← AllocEpochFreq(k, α_k);}
11.         ChangeCoreFreq(k);
12.         t_k ← ProcessTile(k, α_k);            //Tile processing
13.         if(NewEpochStart){
14.             FreqModelTuning(k, f_k, α_k);}
15. }
```

Algorithm 1: Parallel processing and workload management of video application

A.2. Computation Configuration

GenerateTiles ( ):
Input:
Allotted number of cores r_{tot}; Available frequencies $\boldsymbol{f}_{set}$; Minimum frequency of a core f_{min}; Maximum frequency of a core f_{max}; Image dimensions $W \times H$; Quantization Parameter QP; Frame rate in frames per second f_p; Workload matrix A;
Output:
Total tiles/cores/threads k_{tot}; Maximum frequency of each core $f_{k,m}$; Initial $\boldsymbol{\alpha}_{k,m}$ per tile;

```
1.  CoreAvail ← true; k'tot ← 1;          //Initial configuration
2.  while(CoreAvail){
3.    ktot ← TileMap[ktot];               //Actual number of tiles
4.    ĉk,α ← g(αmax, ktot, W × H);        //Estimate cycles per CTU
5.    ∀k, k ∈ {0,⋯, ktot}{                //For each core/tile/thread
6.      αk ← max(A);                      //Start with maximum workload
7.      f̂ ← ntot × ĉktot,α × fp;           //Estimate frequency
8.      fk,m ← Quantize(f̂, fset);         //Supported frequency
9.      if(fk,m ≥ fmax){                  //Required frequency is too high
10.       while(αk ≥ min(A)){             //Test every workload configuration
11.         ĉk,α ← g(αk, ktot, W × H);
12.         fk,m ← Quantize(nk × ĉk,α × fp, fset);
13.         if(fk,m ≤ fmax){αk,m ← αk; break;}
14.         αk ← NextLowerWorkload(αk, A);}}}
15.   if(ktot = rtot){                  //All allotted cores utilized
16.     ∀k, k ∈ {0,⋯, ktot}{
17.       if(αk,m < min(A)){WarnExit( );}} //Workload not supportable
18.     CoreAvail ← false;}
19.   elseif(αk,m = max(A) ∀k, k ∈ {0,⋯, ktot}){
20.       return;}                        //Maximum workload supportable
21.   k'tot ← ktot + 1;}                  //Retest with increased cores
```

Algorithm 2: Total tiles, frequency and workload initialization

AllocFreqToCore( ):
Input:
Suggested core frequency f_k; Allowable frequency set $\boldsymbol{f}_{set}$; Frame rate in frames per second f_p; Epoch size in frames z;
Output:
Core frequency for all tiles in one second f;

```
1.  fk,h ← GetNextHighFreq(fk, fset);
2.  fk,l ← GetNextLowFreq(fk, fset);
3.  for(i in 0 to z-1){
4.    ξ ← (fp − i) × fk,l + i × fk,h;
5.    if(ξ ≥ z × fk){break;}
6.  }
7.  ∀j∈{0 to i}f(j) ← fk,l;
8.  ∀j∈{i+1 to z-1}f(j) ← fk,h;
```

Algorithm 3: Core Frequency Allocation

A.3. PU Map (PUM) Construction

PUM_Generator ( ):
Input:
Frame block *lcu*; CTU size b_w, Variance Threshold v_{th};
Output:
PU map PUM

```
1.  ∀ i ∈ 4×4 of CTU {
2.    [sb_i.x, sb_i.y] ← GetLocationCTU(4 × 4);
3.    sb_i.PUsize ← 4;                    //Initially set the PU sizes to 4×4
4.    sb_i.m ← Mean(CTU_4×4); sb_i.v ← Variance(CTU_4×4);
5.    PUM(sb_i.x, sb_i.y) = sb_i.PUsize; }
6.  blk_size ← 4;
7.  while(blk_size < b_w) {
8.    ∀ h ∈ {sb of blk_size × blk_size} {        //Look for sub-blocks of the required size
9.      ∀ i∈{4 Neighbors of blk_size × blk_size} {  //Four sub-blocks of the same size found
10.       Merge ← true;
11.       V_i, m_i ← CombineVar(V_i, m_i, sb_i);
12.       if(V_i > v_th or sb_i.v > v_th)   Merge ← false; break; }
13.     if(Merge = true) {
14.       sb_h.m ← m_i; sb_h.v ← V_i;
15.       sb_h.PUsize ← blk_size × 2;   //Four blocks merged
16.       PUM(sb_h.x, sb_h.y) = sb_h.PUsize; }}
17.   blk_size ← blk_size × 2; }          //Start with the next sub-block size
18. return (PUM);
```

Algorithm 4: PU Map (PUM) construction algorithm

A.4. Workload Balancing on Heterogeneous Nodes

HeterogenousLoadBalancing ( ):
Input:
Task processing deadline $t_{i,max}$, Number of subtasks per task n_i; Number of available nodes r_{tot}; Power-frequency/configuration relationship of all nodes $p(f)$; Efficiency indices of all nodes ϕ; Cycles for a subtask per node c;
Output:
Total number of used nodes k_{tot}; Frequency of nodes f;

```
1.  g_φ ← SortNodesDesc(φ);                  // Sort nodes according to μ
2.  k_tot ← 1; k ← 0;
3.  LoopIf(k_tot ≤ r_tot){
4.    ∀k∈{0 to k_tot-1 nodes}{
5.      f_k ← f_k,min;                       // Start with min. frequency
6.      n_i,k ← DistLoad(n_i, φ);}           // Load distribution
7.    ThrptMet ← true;                       // Assume throughput is met
8.    ∀k∈{0 to k_tot-1 nodes}{
9.      t_i,k ← TimeOnNode(c_i,k, n_i,k, f_i,k);   // Time spend on processing
10.     NodeThrptMet ← true;                 // Assume node throughput met
11.     if(t_i,k > t_i,max){                 // Throughput not satisfied
12.       NodeThrptMet ← false;
13.       ∀f_i,j∈{f_k,min to f_k,max of node k}{
14.         t_i,j ← TimeOnNode(c_i,j, n_i,j, f_i,j);   // Time spent on processing
15.         if(t_i,j > t_i,max){NodeThrptMet ← true; break;}}
16.       if(NodeThrptMet = false){ThrptMet ← false;}}
17.   }                                      // All k_tot nodes tested with every possible frequency
18.   if(ThrptMet = false){                  // Throughput not satisfied
19.     k_tot ← IncrementNode(g_φ);          // Introduce additional node
20.   else{break;}                           // Configuration met, break
21. }
```

Algorithm 5: Workload distribution and balancing on heterogeneous cores

A.5. Resource Budgeting for Multithreaded Applications

AdaptiveResourceAndPowerAlloc ( ):
Input:
Total number of cores k_{tot}; Total applications/tasks a_{tot}; Total available power p_{tot}; Deadline per data frame $t_{i,max}$;
Output:
Power quota allocation per core of the many-core system

```
1.  do{
2.    ∀i∈{0,⋯,atot−1} ki ← EstimateCores(ti,max); // Estimate cores per cluster
3.    ∀i∈{0,⋯,atot−1} pi ← AllocatePower( );      // Allocate power to each cluster
4.    if(ConfigChanged){∀i∈{0,⋯,atot−1}{
5.      ∀j∈{0,⋯,ki−1} pi,j ← CorePowerAlloc( );}} // Allocate power to each core
6.    while(1){
7.      ∀i∈{0,⋯,atot−1}{                          // For every application
8.        GetNewDataFrame( );
9.        ∀j∈{0,⋯,ki−1} ProcessDataTile( );       // Process thread
10.       no,i ← DataFrameOffset( );              // Determine the offset from the constraint
11.       xo,i ← UpdateAppOffset(no,i);
12.       FreqAdjust( );                          // Adjust frequencies of the cores
13.       if(EpochExpired) break;                 // Start new epoch
14.     }                                         //A task finished
15.   }                                           // All tasks within the core finished
16. }while(Tasks remaining);
```

Algorithm 6: Distributing resources (cores and power) among different tasks

FreqAdjust ( ):
Input:
Total number of cores for the application k_i; Core offset $n_{o,i,j}$; Core frequency set $\boldsymbol{f_{set}}=\{f \mid f \in$ allowable core frequencies$\}$; Total power for the application p_i;
Output:
Core frequency $f_{i,j}$

```
1.  no,i,list ← SortThreadsBynO,i,j( );          //Sort in descending order
2.  zmax ← 0; zmin ← ki − 1;                     //Indices of the sorted list
3.  if(FreqAdjustAllowed){
4.    while(no,i,list(zmax) > 0 AND no,i,list(zmax) > no,i,list(zmin)){
5.      Thigh ← GetThread(no,i,list,zmax);
6.      Tlow ← GetThread(no,i,list,zmin);
7.      if(Power(fset[Tlow.f − 1],fset[Thigh.f + 1]) ≤ pi){
8.        Tlow.f ← fset[Tlow.f − 1];              //Decrease frequency
9.        Thigh.f ← fset(Thigh.f + 1);}           //Increase frequency
10.     zmax ← zmax + 1; zmin ← zmin − 1;
11. }
```

Algorithm 7: Frequency adjustment among cores after processing a task

A.6. Computation Offloading

GOWEnergyControl ( ):

Input:

Initialization Signal s_i; Deviation Signal s_d; User defined duration t_d, Battery levels $e_{batt,enc}$, $e_{batt,dec}$; Frame rate f_p; Size of GOW $z+1$; Energy consumed by GOW $e_{c,GOW,(enc,\ dec)}$);

Output:

Target energy of GOW $e_{t,GOW,(enc,\ dec)}$; Adjusted frame rate and GOW size f_p', z

1. **if(**s_i OR s_d**)** { // (Re)compute duration
2. $f_p' \leftarrow f_p$; $e_{sum,i} \leftarrow 0$;
3. $\forall_{i\in\{enc,dec\}}\ t_{d,i} \leftarrow CalcDuration(e_{batt,i}, z, f_p, e_{IF,avg,i}, e_{WF,avg,i})$;
4. $t_d' \leftarrow min(t_{d,enc}, t_{d,dec})$;
5. $t_{d,req} \leftarrow max(t_d, t_d')$;
6. **if(**$t_d > t_d'$**)** {
7. $f_p' \leftarrow \lfloor (f_p \times t_d') / t_d \rfloor$;
8. $z \leftarrow \lfloor ((z+1) \times f_p') / f_p \rfloor - 1$;}
9. $\forall_{i\in\{enc,dec\}} e_{target,i} \leftarrow \left(e_{batt,i} \times (z+1)\right) / \left(t_{d,req} \times f_p\right)$;
10. $\forall_{i\in\{enc,dec\}} e_{t,GOW,i} \leftarrow e_{target,i}$;
11. } **else**
12. $\forall_{i\in\{enc,dec\}}$ {
13. $\Delta e_i \leftarrow e_i - (e_{t,GOW,i} - e_{c,GOW,i})$;
14. $e_i \leftarrow e_{t,GOW,i} - e_{c,GOW,i}$;
15. $e_{sum,i} \leftarrow e_{sum,i} + e_i$;
16. $e_{t,GOW,i} \leftarrow e_{target,i} + \left((\psi_p \times e_i) + (\psi_i \times e_{tot,i}) + (\psi_d \times \Delta e_i)\right)$;
17. }
18. **return (**$e_{t,GOW,(Enc,Dec)}$**,**f_p'**,**z**);**

Algorithm 8: GOW level energy quota distribution for single-encoder, single-decoder model

ROI-DrivenMBEnergyDistribution( ):
***Input**:*
Hardware throughput constraint n_{sec}; Rank-wise sorted map of ROI *ROI*; List of MBs outside the ROI *nonROI*; Energy quota of the current W-frame $e_{t,WF}$; Energy for transmission for a given transmission distance e_{Tx}; Energy of channel coding e_{cc}; Energy of extrapolating ROI $e_{ROI,ex}$;
***Output**:*
Consumed Energy of W-frame, $e_{c,WF}$

1. $n_{ME,max} \leftarrow \frac{(e_{t,WF} - e_{Tx} - e_{cc} - e_{ROI,ex})}{AvgMBEnergy(LowQualityME)}$; // MEs per WF
2. $n_{frm} \leftarrow n_{sec} / f_p$; // Sustainable MEs per WF
3. $n_i \leftarrow min(n_{ME,max}, n_{frm})$;
4. $n_{ROI} \leftarrow NumMBs(ROI)$;
5. **if**$(n_i = n_{ROI})$ $List_{MBs} \leftarrow MBs(ROI, all)$;
6. **if**$(n_i < n_{ROI})$ $List_{MBs} \leftarrow MBs(ROI, n_i)$;
7. **if**$(n_i > n_{ROI})${
8. $List_{MBs} \leftarrow MBs(ROI, n_{ROI}) \cup MBs(nonROI, (n_i - n_{ROI}))$;}
9. $e_t \leftarrow (e_{WF} - e_{Res} - e_{Tx} - e_{cc} - e_{ROI}) / n_i$;
10. $e_{c,WF} \leftarrow 0$; $e_{t,MB} \leftarrow e_t$;
11. $\forall_{mbi \in ListMBs}$ { // loop over all MBs in the list of selected MBs
12. $CurrMEConfig \leftarrow MEconfig(e_{t,MB})$;
13. $e_{MB} \leftarrow PerformME(CurrMEConfig, mb_i)$;
14. $e_{c,WF} \leftarrow e_{c,WF} + e_{MB}$;
15. $\Delta e \leftarrow e_t - e_{MB}$;
16. $e_{t,MB} \leftarrow e_t + \psi \times \Delta e$;
17. }
18. **return**$(e_{c,WF})$;

Algorithm 9: Block-level energy quota distribution in W-frames

A.7. Hardware Offloading Cost Function

NelderMeadFunctionCost ( ):
***Input**:*
Epoch time t_t; $n_{sec,h,k}$, $n_{sec,k}$, $c_{s,k}$, $c_{h,k}$, for all tasks; Accelerator frequency f_h; Minimum core frequency $f_{k,min}$; Maximum core frequency $f_{k,max}$;
***Output**:*
Cost of the objective function for the given inputs v;

1. $v \leftarrow 0$;
2. $\forall_{k \in \{0 \text{ to } k_{tot}\text{-1 soft-cores}\}}${
3. $t_{h,k} \leftarrow ComputeTimeOnAcc(t_t, n_{sec,h,k}, c_{h,k}, f_h)$;
4. $t_{s,k} \leftarrow t_t - t_{h,k}$;
5. $f_k \leftarrow ComputeCoreFreq(n_{sec,k}, n_{sec,h,k}, c_{s,k}, t_{s,k})$;}
6. $v \leftarrow v + exp(|t_t - \sum t_{h,k}|)$;
7. $\forall_{k \in \{0 \text{ to } k_{tot}\text{-1 soft-cores}\}}${
8. $v \leftarrow v + |n_{sec,k} - n_{sec,h,k}|$;
9. if$(n_{sec,h,k} < 0)\{v \leftarrow v + |n_{sec,h,k}|;\}$
10. if$(f_k > 0)\{v \leftarrow v + f_k;\}$
11. if$(f_k > f_{k,max})\{v \leftarrow v + (f_k - f_{k,max});\}$
12. if$(f_k < f_{k,min}$ AND $t_{s,k} > 0)\{v \leftarrow v + (f_{k,min} - f_k);\}$
13. }

Algorithm 10: Function used in the Nelder-Mead optimization for shared hardware allocation

A.8. Edge Detection in 16×16 MB

DominantMode ( ):
Input:
MB X;
Output:
Dominant prediction mode m;

1. rd_w, $E_w \leftarrow$ EdgeFeature(X), $\forall$ $w = a, b, c, d$
2. $Line_{out} \leftarrow$ DominantLine($R_{d,w}$, E_w)
3. **switch** $Line_{out}$ **do** // there is an implicit 'break' after each case
4. **case** $Line_0$
5. $m \leftarrow$ (DC, P, H, V) // No edges found
6. **case** $Line_1$
7. **If** E_a-$E_b > s/2$ **then** $m \leftarrow$ (DC, H, P, V)
8. **elseif** $E_a > E_b$ **then** $m \leftarrow$ (DC, P, H, V)
9. **elseif** E_b-$E_a < s/2$ **then** $m \leftarrow$ (DC, P, V, H)
10. **else** $m \leftarrow$ (DC, V, P, H)
11. **case** $Line_2$
12. **If** E_a-$E_c > s/2$ **then** $m \leftarrow$ (H, DC, P, V)
13. **elseif** $E_a > E_c$ **then** $m \leftarrow$ (DC, H, P, V)
14. **elseif** E_c-$E_a < s/2$ **then** $m \leftarrow$ (DC, P, H, V)
15. **else** $m \leftarrow$ (DC, P, V, H)
16. **case** $Line_3$
17. **If** E_b-$E_d > s/2$ **then** $m \leftarrow$ (DC, V, P, H)
18. **elseif** $E_b > E_d$ **then** $m \leftarrow$ (V, DC, P, H)
19. **elseif** E_d-$E_b < s/4$ **then** $m \leftarrow$ (DC, V, P, H)
20. **else** $m \leftarrow$ (DC, P, H, V)
21. **case** $Line_4$
22. **If** E_c-$E_d > 3s/4$ **then** $m \leftarrow$ (DC, V, P, H)
23. **elseif** $E_c > E_d$ **then** $m \leftarrow$ (DC, P, V, H)
24. **elseif** E_d-$E_c < 3s/4$ **then** $m \leftarrow$ (DC, P, V, H)
25. **else** $m \leftarrow$ (DC, H, P, V)
26. **case** $Line_5$
27. $m \leftarrow$ (V, DC, P, H)
28. **case** $Line_6$
29. $m \leftarrow$ (H, DC, P, V)
30.

Algorithm 11: Edge based likely mode selection for H.264/AVC Intra 16×16

A.9. Motion Estimation via Hybrid Memory Subsystem

PowerGatedME( ):
Input:
Video frame *F*; Reference frame Search Window *sw*; Memory access pattern for ME Engine ME_{patt};
Output:
Search window MRAM power gate control register PG_{SW};

```
1.  ∀ i | i ∈ {CTU in F} {
2.    wait(ReadBankReady);                    //Block-matching is ready for CTU
3.    CU ← CTU_i;
4.    PerformME(CU);
5.  }
6.  Function PerformME(CU) {
7.    [β1 β2] ← EstmMRAMAddr(ME_patt, CU);
8.    PG ← TurnONMem(β1, β2);                 //Preemptive turn ON
9.    TurnOFFExcessMem(ME_patt, CU, PG);
10.   ∀ j | j ∈ {PU in CU} {
11.     ∀ n | n ∈ {Search number in ME_Patt} {
12.       [b_w b_h] ← GetDim(PU_j);
13.       WaitMEReady( );                     //ME not ready yet
14.       [x_ref, y_ref] ← GetRefAddr(n, ME_patt);
15.       if (β1 > x_ref or β2 < x_ref + b_w) {
16.         PG ← TurnONMem(x_ref, x_ref + b_w);   // latency
17.         UpdateEstimator(x_ref, CU);  }
18.       MotionEstimation(n, ME_patt, x_ref, y_ref, PU);  }
19.   }
20.   if (CU = SmallestCU)  return;           //8×8 CU, don't divide further
21.   ∀ k | k ∈ {0,1,2,3} {                   //Split the CU in 4 equal sub-CUs
22.     CU_k.C_W ← CU.C_W / 2; CU_k.C_H ← CU.C_H / 2; CU_k ← CU;
23.     PerformME(CU_k);      }               //Recursive function call
24. }
```

Algorithm 12: AMBER pseudocode for motion estimation search in HEVC video encoder

Appendix B ces265 Video Encoder

Due to the high workload of video compression and the advent of the power-wall, video encoding standards now allow for multithreaded (parallel) encoding possibilities, which can be exploited on multi- and many-core systems. This work proposes an open-source software architecture for parallel video encoding, and demonstrates the proposed concepts using a case study on High Efficiency Video Coding (HEVC). By simultaneously encoding video tiles on independent cores, time complexity and power savings are obtained, especially for large workload (like encoding of video with high frame-resolution). The proposed architecture is flexible and allows for easy integration, runtime adaptation and extension.

B.1. Introduction and Motivation

New video standards, like High Efficiency Video Coding (HEVC) [229] and VP9 [284], permit parallel encoding of video frames by breaking coding dependencies across boundaries of independent coding regions. This allowance gains more significance because of the increasing trend of encoding videos with high resolutions, at high frame rates. Although current industry standards like H.264/AVC [32] and VP8 also allow for parallel encoding, their compression efficiency is not high compared to their successors HEVC and VP9. On the other hand, the large compression efficiency offered by HEVC and VP9 is accompanied by high computational complexity and workload. For example, HEVC consumes about 40-70% more time as compared to H.264 [285]. Therefore, it is vital to parallelize the newest video encoding standards if they are to replace the current industry video encoding standards.

In addition, current trend of designing programmable processing chips is to have multiple low-frequency, low-power cores instead of a single, fast and power-hungry core. This is because as already explained, the dynamic power (p_{dyn}) of the compute core is related to its frequency (f) with the following relation:

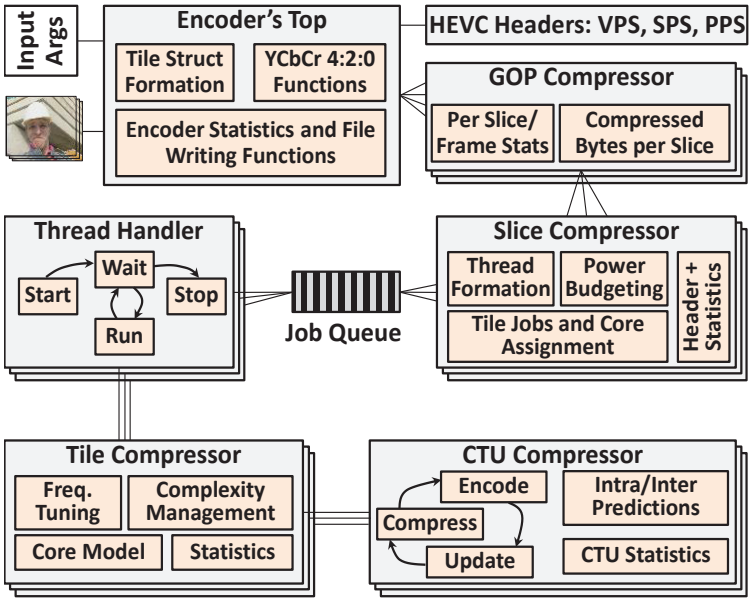
$$p_{dyn} \propto c v_{dd}^2 f \tag{A-1}$$

In this equation, c is the capacitance and v_{dd} is the device voltage. Further, increasing the frequency of a core requires a linear increase in v_{dd}, which concludes that dynamic power is roughly proportional to the cube of the frequency. Further, for every chip, there is a power cap, which must not be violated, otherwise, we risk damaging the chip. Hence, a compute core's frequency cannot be arbitrarily increased. This constraint is usually termed as the power wall which suggests that increasing the frequency of the compute cores in order to sustain the quality of service demand (i.e., encoding frame rate) is not advisable. Hence, chip designers have steadily moved towards having multiple compute cores, which has resulted in software designers to develop parallelization mechanisms, in order to fully exploit the available potential of multiple compute cores.

The HEVC software can divide a video frame into rectangular regions, termed as tiles. These tiles can be independently encoded (in parallel). However, to satisfy the encoding frame rate demand, the software needs to distribute tiling workload by considering (a) the total number of available compute cores and (b) the frequency of the core. In addition, it is possible that HEVC video encoding takes place on a heterogeneous multi- or many-core system. In such a scenario, some compute cores have lower resources/capacity (e.g., lower frequency, smaller caches etc.) then the rest. On the contrary, workload of video tiles may not be similar due to non-uniform tile structure or due to change in the video content at runtime. Thus, tiles may consume different amount of compute cycles and the workload of cores is unbalanced, which may result in computational hotspots and thus decrease core utilization. The proposed software architecture of HEVC, termed as ces265 [44], targets the above mentioned challenges by providing adaptability and flexibility to HEVC video encoding, and adept integration and extension possibilities.

B.2. Technical Description of ces265

The proposed software architecture of ces265 is shown in App-Figure B-1. There are multiple classes used in ces265, each performing a specific purpose. The Encoder's Top class is used to access YCbCr 4:2:0 planar video data, create tile structure for encoding video frames and write encoder statistics. The main compression class is the Group of Pictures (GOP) Compressor class. A group of video frames is provided to this class, and it outputs compressed bit-streams of each video frame to the Encoder's Top class. The GOP Compressor calls the Slice Compressor class to compress video frames. The Slice Compressor class calls functions of the Tile Compressor class, and each Tile Compressor class has an associated Coding Tree Unit (CTU) Compressor class. The CTU Compressor class is the main class where the Intra and Inter encoding functions of HEVC are executed and bit-stream is generated.



App-Figure B-1: Software architecture of ces265. The Encoder's Top object can have multiple GOP Compressor objects. Each GOP Compressor object can have access to multiple Slice Compressor objects. With each Slice Compressor object, a Workload Queue object is associated, with can be accessed by multiple Thread Handler objects.

With reference to the challenges mentioned in Section B.1, some scientific and technical aspects of ces265 will now be briefly discussed.

Tile Formation: In ces265, the user can assign a tile structure of a video frame via three options. In the first option, if the multi- or many-core system is homogeneous (i.e., all the computing cores have the same resources e.g., frequency, cache size), the user can opt for a uniform tile structure. This requires providing the software with width and height on the frame in tiles. For the second option, the user can declare the exact width and height of each tile in CTUs. This scheme is suitable for workload balancing on heterogeneous systems with diverse set of compute cores, or, when it is required to reduce the number of compute cores [45]. If this option is used, the user is required to provide the exact width and height of tiles in CTUs. For the third option, the software itself determines the number and location of uniform tiles, depending upon the number of available/allotted compute cores and frequency of the cores [44]. These parameters (number of cores and frequency) can be directly provided to ces265 using command line parameters.

Multithreading: Multithreading is achieved by three entities; a processing thread, a job queue (or FIFO) and a manager to fill the job queue. Manager fills the job queue from one side and the independent threads pulls the jobs from the other side. To encode multiple tiles simultaneously in ces265, the Slice Compressor class creates a pool of parallel threads before start of encoding. These threads (defined in Thread Handler class) wait for the arrival of jobs allocated to the threads. Tile compression jobs are also created by the Slice Compressor class. These jobs are pushed to a Workload Job Queue (also a class). A processing thread which is currently idle and waiting for a processing job pops a tile compression jobs from the queue. A video frame is deemed encoded when all the processing threads are idle and there are no more jobs left in the Workload Queue.

Consider the case of uniform tiles encoded via a homogeneous many-core system. In such case, all the

threads will approximately take the same amount of time and the number of threads must equal the number of tiles. In case of non-uniform tiling (with some tiles smaller than the others), the number of parallel threads created can be less than the total number of tiles. This is because a single thread can process multiple smaller tiles in the same amount of time as a thread processing a larger tile. However, for both uniform and non-uniform tiles, the software does not require any changes. Larger and smaller tile jobs will enter the same workload queue and will be treated by the threads all the same. An idle thread will start processing the tile as soon as there is a job available in the queue.

Note that the Workload Queue and the Thread Handler classes are independent of ces265 tile encoding. These classes can be reused in ces265 (or other programs) as they only need to know which function to call as a parallel computing entity and a structure containing input arguments to this function.

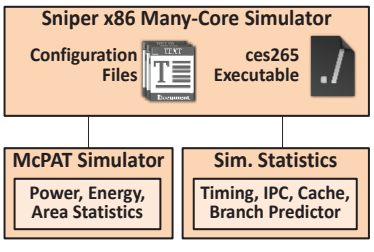
Power Management: As discussed in Section B.1, power consumption of a device will be a major concern in the future systems. The proposed ces265 is designed with the idea of application-level power management. If the platform allows the application to change the working frequency of the compute core, then the additional functionality of core's frequency tuning can be embedded into the software. By changing the frequency, the application can reduce the power consumption of the system by not running the core too fast, or satisfy the quality of service demand (frame rate) by not running the core too slow. The ces265 allocates such frequency control function in the Tile Compressor class, whereby just after the core starts executing the Tile Compressor class, the frequency of the core is adapted to compress the corresponding tile. The frequency of a core depends upon the total power budget allocated to the many-core system.

B.3. Implementation and Uses of ces265

ces265 is written in C++, without using any architecture specific Intrinsics [286] or SIMD [287] instructions. Multithreading is achieved by using pthread API. Using the same source, ces265's binary can be compiled on both Windows and Unix/Linux based operating systems. In summary, the aim of writing ces265 is to be:

- Easily understandable and readable, for getting started with HEVC and attaining video encoding concepts.
- Portable to multiple platforms, by using only standard C++ libraries and function calls.
- Easily extendible to include additional functionalities, such as parallelizing GOP and Slice compression.
- Portable to small systems with limited fast memories, by having a small memory footprint.
- Runtime configuration possibility, by presenting the user multiple encoding options and configuration knobs.
- Extensive analysis of encoder's workload and parameters, by dumping encoder's statistics at multiple levels of hierarchy (e.g. at CTU- and Tile-level).

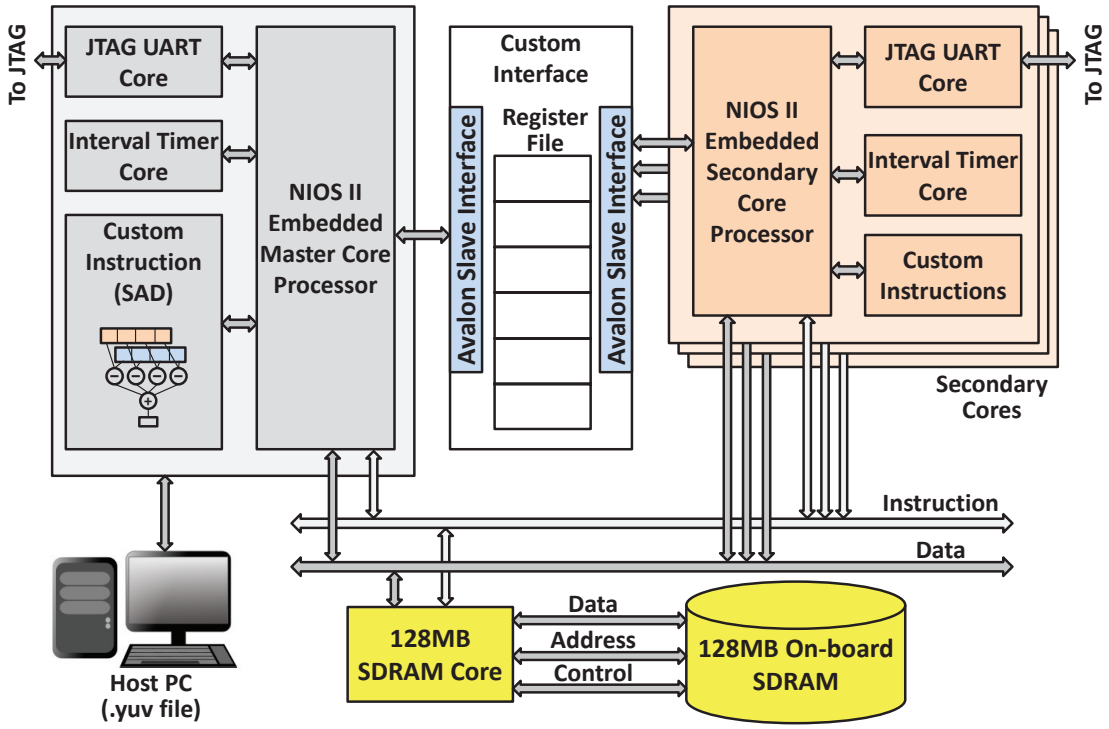
The ces265 is open-source software (available for download, http://ces.itec.kit.edu/ces265/) under GNU General Public license (http://www.gnu.org/licenses/). To test tiling, multithreading and power management (by frequency scaling) of ces265 on a many-core system, Sniper x86 many-core simulator [217] is used, as shown in App-Figure B 2. This simulator allows for frequency scaling within the running application, which is exploited by ces265 to reduce the power consumption of the system.



App-Figure B-2: Sniper simulator setup to run ces265

B.4. FPGA Implementation of Multi-Core ces265

The proposed ces265 is also ported to an Altera's FPGA based soft- core system as shown in, which



App-Figure B-3: Multi-core implementation of ces265 on an Altea's FPGA

illustrates the multi-core hardware architecture (an extension of the single core ces265 encoder design). For this purpose, the soft embedded cores called NIOS-II (provided by Altera) are used to process HEVC encoding. Each core is used to process a single tile of the video frame. One of the core called Master core prepares the data to be processed by the other cores, called Daughter cores.

The multi-core ces265 implementation on FPGA involves efficient arbitration and sharing of peripherals. In the present system, all the cores share:

- SDRAM Controller Core for storage of software and processed data.
- A Custom peripheral for efficient communication among the cores.
- System ID peripheral for successful debugging the software in multi-core environment.

Sharing the peripherals is another challenge of the multi-core system. There is no guarantee to which core peripheral will be assigned for accessing. Sharing of external memory via the SDRAM controller is one of the toughest challenges of a multi-core system. Since the memory contains the program and data for each core, it is important to use separate area for code execution of each processor. It should be noted that cores sharing the same program memory space will result in erroneous results as one core might overwrite other cores' memory. Each processor is given its own unique .text, .rodata, .rwdata, .heap and .stack sections. These are usually the five default linker sections that are defined as:

- .text contains the executable code.
- .rodata is the read only data used for execution of the software.
- .rwdata is used for storage of read and write variables, along with pointers that have been used in the software.
- .heap is used as a storage for dynamic memory allocation (using "malloc" in C or "new" in C++).
- .stack is used for storing function call parameters and other temporary data.

Another valuable inclusion to the multi-core systems is the implementation of Custom Instructions (CI) or

in-core hardware accelerators. These accelerators are user defined macros that are designed and implemented in hardware. Its sole purpose is to accelerate time critical software algorithms. With the help of in-core accelerators, software algorithms which might take a large number of clock cycles to execute can be replace with custom macros which usually take far lesser number of cycles (minimum of just one cycle), thus further decreasing the total computation time of the ces265 encoder. In this implementation, the SAD function (which usually takes about 80% of the processing time) is implemented as a hardware accelerator.

Further, note that the proposed architecture is not limited to ces265. Any parallelizable application can be implemented on custom hardware like FPGAs, using the above architecture.

Working: The Master core governs the processing flow of the proposed architecture. It knows about the addresses of the data frames written to the external memory, the tile structure which must be generated for Daughter cores to properly work, and the details of the custom interface used for communication.

Depending upon the number of Daughter cores, the Master core first determines the exact dimension of the video tiles, as well as the starting and ending pixel locations of these tiles. For this purpose, it can employ uniform or non-uniform tiling approaches. The Master core also allocated memory for bit-stream handlers and other associated data structures, for all the tiles (which will be processed by the Daughter cores). Afterwards, it starts writing the tile addresses and other memory pointers to the custom interface (which is basically a set of registers associated with each daughter core). Afterwards, the Master core sets a “go” signal which commences the tile processing. In this case, the Master core also processes a single tile. Once it finishes processing its tile, it starts collecting status signals of all other Daughter cores (which fill the appropriate registers within the custom interface to inform the Master core about the compute status). If all cores have finished processing their workload, the Master core initiates processing the next frame.

B.5. Conclusion and Future Direction

Here, ces265, a software architecture for multithreaded HEVC encoding is presented. Using the proposed multithreading support, a user can select different tiling configurations depending upon the characteristics of the many-core system, suitable for the needs of the user.

In future development, it is planned to extend ces265 by including vector processing instructions (or SIMD instructions). Parallelizing the GOP and Slice compression (in addition to parallel tile compression) is a logical step towards hierarchical resource and power management of ces265. This can be achieved with minimal effort, because as previously pointed out in Section B.2, the Job Queue and Thread Handlers can be replicated and inserted before GOP Compression and Slice Compression classes. In addition, it is planned to use ces265 for real-time 3D-HEVC for efficient compression of 3D videos.

Appendix C H.264/AVC Prototype

To functionally verify the proposed hardware accelerators for H.264/AVC and multicasting, the complete H.264/AVC encoder is prototyped on an Altera's FPGA. This encoder includes:

- Parallel processing of up to four parallel FullHD (1920×1080) video frames at 25 fps.
- Capturing raw video samples from real-world analog camera inputs, and converting them into appropriate video format, i.e., YCbCr 4:2:0.
- Writing and reading video frame samples to/from the external DDR3 memory of the FPGA development board.
- Processing the video frame samples on the FPGA and compressing them to H.264/AVC standard-compliant bit-stream.
- Displaying the locally decoded bit-stream via DVI for quick debugging.

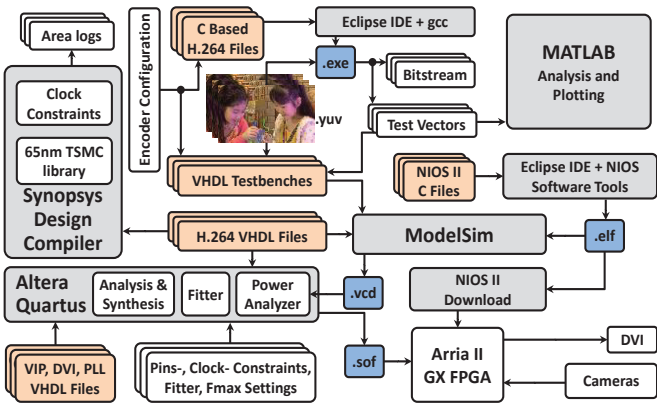
The design challenges include for implementing the prototype include:

- Debugging the hardware board.
- Integration of video cameras to the FPGA board via the daughter board.
- Converting analog streams into digital YCbCr 4:2:0 interpretation.
- Interfacing the DDR3 memory with the FPGA, by appropriately tuning the timing parameters.
- Setting the clock frequencies of the DDR3 and the video encoder.
- Integrating Gigabit Ethernet with the video encoder.
- Setting up the DVI output.

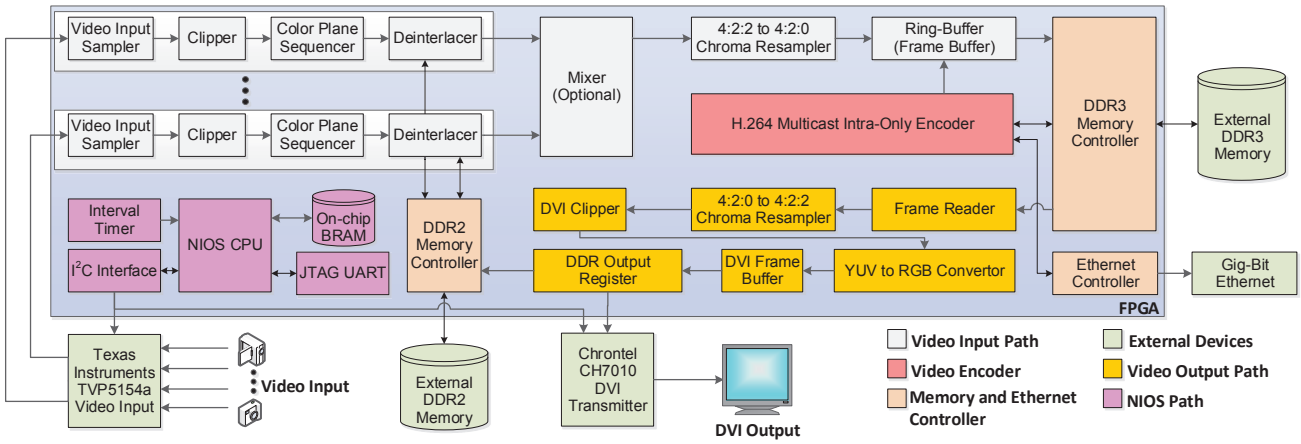
In the following, brief details about the working principles of the prototype will be provided.

C.1. Simulation and Design Workflow

The simulation and design workflow of the proposed multicast encoder is shown in App-Figure C-1. The multicast H.264 Intra encoder is developed using an in-house C-based H.264/AVC Intra encoder, MATLAB and ModelSim co-simulation framework. Altera's Quartus together with Synopsys Design Compiler are used for design analysis and implementation. Encoding configurations are provided to the C-based H.264/AVC encoder, which is used to generate bit-streams and test vectors to be used in ModelSim simulation. These test vectors are also used to visualize the impact of the proposed hardware accelerator via MATLAB's plotting and analysis tools. The VHDL files (for H.264/AVC entities and related test-benches), test vectors and the Executable and Linkable (elf) file of the embedded NIOS-II CPU [288] are used for RTL-level ModelSim simulations and debugging. These VHDL files are also forwarded to the Altera Quartus 12.1 software to generate the hardware bit-stream, which is burned on the FPGA. The Synopsys Design Compiler is used to generate area and power logs for the H.264/AVC VHDL entities using a 65nm TSMC library.



App-Figure C-1: Simulation and design methodology of the proposed multicast H.264/AVC video encoder.



App-Figure C-2: Prototype architecture on Altera's Arria II GX FPGA. For complete system implementation of H.264 Intra-only

C.2. FPGA Prototype

For functional verification of the architecture, the H.264/AVC encoder and the multicast video gather and display separately is implemented on a real-world FPGA. These are prototyped on Altera's DK DEV 2AGX260N FPGA Development Kit containing a cost-effective Arria II GX EP2AGX260 FPGA. The architecture for the proposed encoder is shown in App-Figure C-2. The development kit used in this work features a 64-bits wide DDR2 dual inline memory and a 16-bits wide single chip DDR3. For functional verification, the video data is fed from an SD camera to the proposed FullHD encoder pipeline and the output bit-stream (generated by CAVLC [289]) is passed to the output node as AVB packets [290] via Gigabit Ethernet. The camera is connected to the FPGA prototyping board as shown in App-Figure C-3. The whole encoder uses a single clock domain of 150MHz. DDR3 is clocked at 300MHz. NIOS II embedded CPU is used for frame-by-frame control signaling and for future extensions of the prototype. The NIOS II is also used to configuring the video input, DVI and Ethernet module at startup (not shown in the figure). Note that the video output path is not required in the actual implementation, it is used to view the reconstructed video frames. Similarly, the mixer component is optional as it is used only to stack the frames together. This is also helpful for background generation.

App-Figure C-3: Prototype implementation on Altera's Arria II GX FPGA.

As a service to the research community, the VHDL of the proposed H.264/AVC encoder is open-sourced, which can be downloaded at [291]. Writing and testing custom hardware for a large and complex project (like H.264/AVC) is a highly time-consuming process. Therefore, with this release, the research community will gain a head start in developing video compression schemes due to fast evaluation and simulation. Moreover, it will help the VLSI and video-system architecture community to implement and test their novel algorithms and architectures in an efficient manner.

C.3. Prototype Evaluation

On the FPGA prototype, the total time taken by the trans-form module is 56 cycles. Each prediction (X') generation requires 17 cycles, including the P prediction mode. Compared to the state-of-the-art approaches, the different attributes of our encoding scheme are tabulated in App-Table C-1. The state-of-the-art schemes here do not consider multicasting. Therefore, we have given the resolution supported by a single encoder for

the pro-posed architecture. In contrast to the others approaches, our proposed encoder can be adjusted to balance the workload and encoding methods. Moreover, hardware is saved by: (a) reusing the same structure for SAD computations of all the four modes, (b) by realizing only one DCT folded butter-fly [249] and (c) reusing same hardware for quantization and inverse quantization. Further, compared to [160], our approach only uses one prediction unit (instead of two parallel units) at 150MHz instead of 310MHz. However, our approach can also be extended in a similar fashion and is more flexible be-cause the parameters d_s and θ are controllable. Area usage and maximum frequency of important modules of the encoding loop implementation are given in App-Table C-2. The M9K embedded SRAM block memories are used as FIFOs to connect the encoding stages and the total area of the encoding loop also includes these FIFOs.

App-Table C-1: Comparison of the proposed encoding system with State-of-the-Art Schemes (NI: Not Implemented, Tf: Transform)

	[158]	[161]	[300]	[5]	**This work.**
MHz	54	150	140	114	150
Size	720×480 4:2:0	1920×1080 4:2:0	1920×1080 4:2:0	1920×1080 4:2:0	4068×2288 4:2:0
FPS	31	61	30	30	25
Mode	I4MB, I16MB Parallel	I4MB, I16MB Parallel	I4MB, I16MB Serial	I4MB, I16MB Parallel	I16MB Serial
Design	TSMC 250 nm	TSMC 180 nm	TSMC 130 nm	TSMC 130 nm	Altera FPGA 40nm
Area	89K Gates	201.8K Gates	94.7K Gates	265.3K Gates	11K ALUS, 8K Regs, 562 Kbit BRAM
Entropy Coding	CAVLC	NI	CAVLC	CABAC	CAVLC
Multicast Support	No	No	No	No	Yes
Cam. input	No	No	No	No	Yes
Tf Folding	No	No	No	No	Yes
Tf Merging	Yes	Yes*	Yes	Yes	Yes
Q Merging	No	No	No	Yes	Yes

In App-Table C-3, the results for compiling the VHDL of the proposed encoder on a TSMC 65nm technology [264] are shown. Synthesis is carried out using Synopsys [265], with medium effort mapping and automatic hold time violation correction. The total number of gates present the 2-input NAND equivalent of the 65 nm TSMC technology. Note that we target hardware frequency of 310MHz.

App-Table C-2: Synthesis results for our I16MB encoding loop. The total area (last row) exceed the sum of the components, because there is further glue logic between the components (e.g. FIFOs)

Module	**MHz**	**Aluts**	**Regs**	**Memory [Kbit]**
4 × 4 RO+HT	321.34	438	1,604	0
Transform	167.87	7,901	3,958	5.34
Mode Pred.	171.14	1,426	747	256
Edge Detector	385.8	283	525	0
Reconstruct	475.74	460	969	0
Total	–	10,583	8,088	562

App-Table C-3: Synthesis results for TSMC 65 nm at 310MHz. The total gates denote the total logic equivalent of a 2-input NAND gate

Module	4 × 4 RO+HT	Luma Tf	Choma Tf	Edge Detector	Reconstruct	Scheduler (n_v=4)
Gates	5.83K	106	217.2	4.38	1.14	1.56
SRAM	0	5.34K	10.69K	0	0	0

In order to determine the total number of MBs n_{frm} that can be processed by the encoder, given the total cycles allocated per MB c in the loop clocked at f_k MHz with a target FPS of f_p, we calculate n_{frm} as shown in Equation (A-2). Using n_{frm}, we can calculate the maximum image dimensions for given dimension ratios. In App-Table C-4, we show the maximum sustainable frame dimensions at 16:9 for different encoder configurations. We use f_k of 150 MHz at 25fps. As seen, the maximum dimensions supporting all 16×16 modes at 25 fps with no row down-sampling is 4068×2288.

App-Table C-4: 16:9 resolution supported by the proposed encoder for 25 fps operating at 150MHz

	$\theta = 1$	$\theta = 2$	$\theta = 3$	$\theta = 4$
$ds = 1$	4894×2752	4564×2568	4294×2416	4068×2288
$ds = 2$	4894×2752	4564×2568	4416×2484	4280×2408

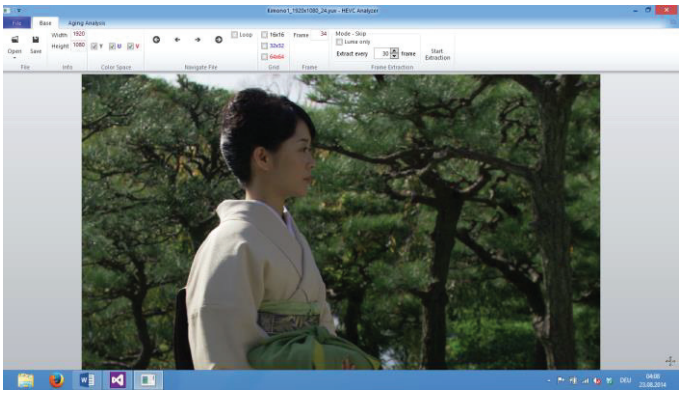
$$n_{frm} = \frac{f_k}{f_p \times c} \tag{A-2}$$

Appendix D Memory Aging Analysis Tool

This appendix details a GUI-based tool (YUV Aging-Simulator, Analyzer and Viewer, YASAV), to simulate aging and analyze the impact of video content on different regions of the SRAM memory. While the main focus is to develop SRAM aging resilient schemes for video memories, the tool can also be used to visually present the impact of aging. This tool can display YCbCr 4:2:0 videos, perform and present SRAM aging analysis if the same video is written to that memory, analyze the impact of different aging balancing circuits and provide opportunity to the viewer to access the impact of video-content on the aging of SRAM. This is an open-source tool and available for download at [266], and the major contributor is Mr. Orcun Tufek.

D.1. The YASAV Tool

The screenshot of the proposed YUV Aging-Simulator, Analyzer and Viewer (YASAV) tool is given in App-Figure D-1. This screenshot shows the tool after opening the "Kimono1" YCbCr 4:2:0 sequence. As seen, there are three tabs ("File", "Base" and "Aging Analysis") in the ribbon at top of the screen. The user can open video files, provide video resolutions and get help documents by using the "File" tab in the ribbon. The "Base" tab is used to visualize the file, as well as put block-based grids on the images (for understanding block-based video processing). "Aging Analysis" tab provides different configuration settings for aging balancing and generates plots (boxplots, histograms, stressmaps etc.) for the user. These plots can also be saved to files for later analysis.



App-Figure D-1: A screenshot of YASAV tool.

File Handling: The tool is capable of opening raw YCbCr 4:2:0 videos, images with .bmp, .jpeg and .png extensions, and all other files types. The opened images/videos can be zoomed or panned using keyboard shortcuts of the mouse control. It can directly guess the resolution of the video sequence using the name of the sequence (because usually for raw video streams, the resolution of the video is a part of the name of the video). The tool is capable of displaying all combinations of the luminance or the chrominance components of the YCbCr 4:2:0 video and overlay grids of different square sizes (in accordance with the HEVC standard, like 64×64, 32×32 etc.). YASAV can also be used to extract different frames of the sequence and save them in a bitmap file. Further, it can also be used to extract only the luminance component of the video and store it separately.

Aging Analysis: For aging analysis of SRAM memory to which the image/video/data-content is written can be configured by the user. Specifically, a user can define the size of the SRAM and the width of a word in bits. However, if no file size is defined, the video resolution is used to define the size of SRAM such that the size corresponds to the number of bits which will be used to present the video frame. And it is assumed that successive video frames overwrite these SRAM locations, causing different duty-cycles for each 6T SRAM cell. In addition, the number of years for which the SRAM will be employed under the given content characteristics and the reference Static Noise Margin (SNM) can be set by the user. Once these values are set, the user can select the aging balancing strategy. Currently, four different aging balancing strategies are embedded in the YASAV tool. One is the base, which means that no aging balancing should be employed. The other three are Circular Swap, Circular Shift and Inversion as given in Figure 3-15. Moreover, the balancing frequency of these circuits is also set by the user. Further, the granularity of the aging histograms can also be set by the user and afterwards, the processing can start.

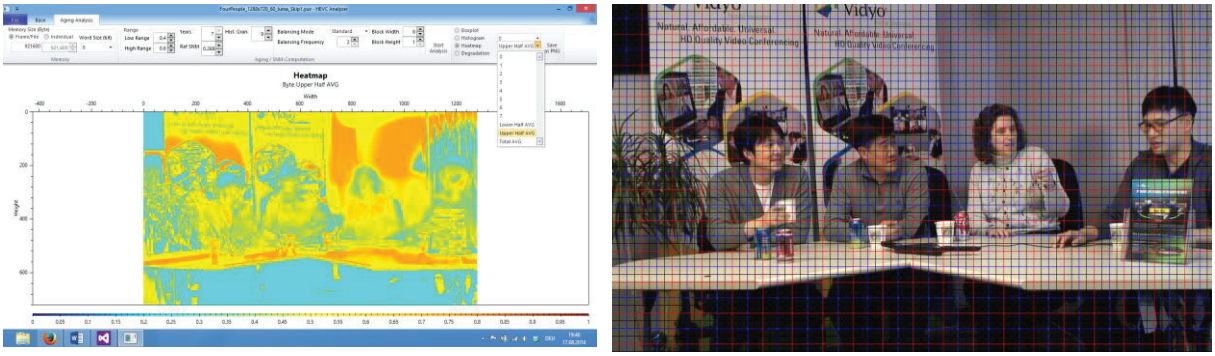
Tool Output: Once the processing is done, different files are generated and are available for the user to view. A file which contains information about the distribution of duty-cycles, for each bit of the data is generated. This file is used to draw the box-plots within the tool for visualization of the impact of the aging balancing circuits. Moreover, another file which contains the duty cycle of each 6T SRAM cell is generated, and this file is used to create the stressmaps within the YASAV tool. Moreover, the SNM degradations of all the cells are also saved and used for creating SNM degradation histograms.

D.2. Tool Development

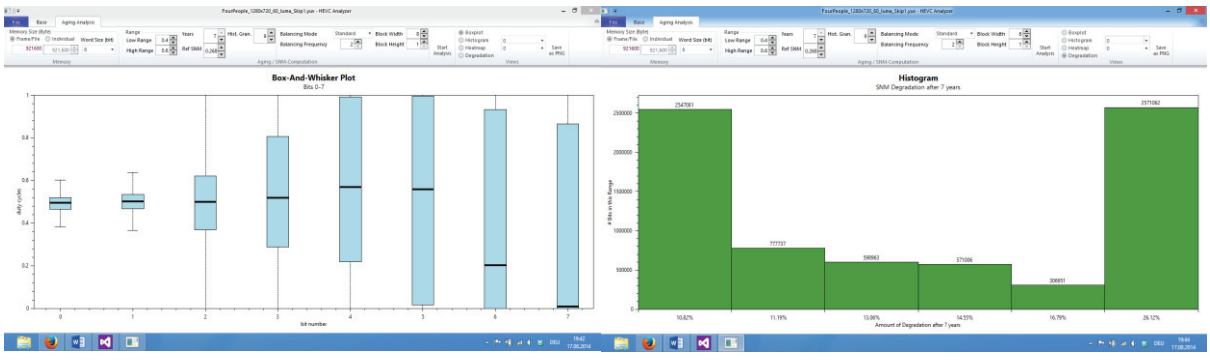
The YASAV tool is developed on a Windows system, using C# and Windows Presentation Foundation (WPF) graphical rendering system [292]. To make the tool interface easy to navigate and access, the "Ribbon" interface is added, which is a graphical representation of multiple tabs to replace menus of a GUI, for increased management and reduced complexity. The "Ribbon" library specifically used in this tool is the "Fluent Ribbon Control Suite" [293]. To create plots and histograms, the "OxyPlot" [294] library is used.

D.3. Screenshots

Some screenshots of the YASAV tool are given below.



App-Figure D-2: (a) Visualizing the stressmap of "FourPeople" sequency and (b) 16×16, 32×32 and 64×64 grid overlaid on the video.



App-Figure D-3: (a) Box-plot of duty-cycles for all bits and (b) SNM degradation histogram.

Bibliography

[1] C. Rosas, A. Morajko, J. Jorba and E. Cesar, "Workload balancing methodology for data-intensive applications with divisible load," in *Symposium on Computer Architecture and High Performance Computing*, 2011.

[2] A. Colin, A. Kandhalu and R. Rajkumar, "Energy-Efficient Allocation of Real-Time Applications onto Single-ISA Heterogeneous Multi-Core Processors," *Journal of Signal Processing Systems,* pp. 1-20, 2015.

[3] K. Ma, X. Li, M. Chen and X. Wang, "Scalable power control for many-core architectures running multi-threaded applications," in *Internation Symposium on Computer Architecture*, 2011.

[4] D. Forte and A. Srivastava, "Energy and Thermal-Aware Video Coding via Encoder/Decoder Workload Balancing," in *International Symposium on Low Power Electronic Devices*, 2010.

[5] H. C. Kuo, L. C. Wu, H. T. Huang, S. T. Hsu and Y. L. Lin, "A Low-Power High-Performance H.264/AVC Intra-Frame Encoder for 1080pHD Video," *IEEE Transactions on Very Large Scale Integrated Systems (TVLSI),* vol. 19, no. 6, pp. 925-938, 2011.

[6] J.-C. Wang, J.-F. Wang, J.-F. Yang and J.-T. Chen, "A fast mode decision algorithm and its VLSI design for H.264/AVC intra-prediction," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 17, no. 10, pp. 1414-1422, 2007.

[7] G. J. Sullivan, J. Ohm, W. Han and T. Wiegand, "Overview of High Efficiency Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 22, no. 12, pp. 1649-1668, 2012.

[8] H. Esmaeilzadeh, E. Blem, R. Amant, K. Sankaralingam and D. Burger, "Dark silicon and the end of multicore scaling," in *International Symposium on Computer Architecture (ISCA)*, 2011.

[9] J. Henkel, L. Bauer, N. Dutt, P. Gupta, S. Nassif, M. Shafique, M. Tahoori and N. Wehn, "Reliable On-Chip Systems in the Nano-Era: Lessons Learnt and Future Trends," in *Design Automation Conference (DAC)*, 2013.

[10] ARTEMIS, [Online]. Available: http://www.artemis-ju.eu/embedded_systems. [Accessed 20 October 2015].

[11] M. Shafique, "Architectures for Adaptive Low-Power Embedded Multimedia Systems," Karlsruhe Institute of Technology (KIT), 2011.

[12] D. Nister, "Automatic passive recovery of 3D from images and video," in *International Symposium on 3D Data Processing, Visualization and Transmission*, 2004.

[13] L. Cutrona, E. Leith, L. Porcello and W. Vivian, "On the application of coherent optical processing techniques to synthetic-aperture radar," *Proceedings of the IEEE,* vol. 54, no. 8, pp. 1026-1032, 1966.

[14] "WebRTC," [Online]. Available: http://www.webrtc.org/. [Accessed 26 Aug 2015].

[15] "Intel chips through the years," 12 September 2015. [Online]. Available: http://interestingengineering.com/intel-chips-timeline/. [Accessed 5 October 2015].

[16] J. Held, "Introducing the Single-chip Cloud Computer: Exploring the Future of Many-core Processors," in *Intel White Paper*, 2010.

[17] A. Pathania, S. Pagani, M. Shafique and J. Henkel, "Power management for mobile games on asymmetric multi-cores," in *Low Power Electronics and Design (ISLPED)*, 2015.

[18] S. Momcilovic, A. Ilic, N. Roma and L. Sousa, "Dynamic Load Balancing for Real-Time Video Encoding on Heterogeneous CPU+GPU Systems," *IEEE Transactions on Multimedia,* vol. 16, no. 1, pp. 108-121, 2014.

[19] W. Xiao, B. Li, J. Xu, G. Shi and F. Wu, "HEVC Encoding Optimization Using Multi-core CPUs and GPUs," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT),* vol. 9, no. 99, pp. 1-14, 2015.

[20] M. Khan, M. Shafique, L. Bauer and J. Henkel, "Multicast FullHD H.264 Intra Video Encoder Architecture," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* pp. 1-5, 2015.

[21] K. Benkrid, D. Crookes, J. Smith and A. Benkrid, "High level programming for real time FPGA based video processing," in *Acoustics, Speech, and Signal Processing (ICASSP)*, 2000.

[22] K. Iwata, S. Mochizuki, M. Kimura, T. Shibayama, F. Izuhara, H. Ueda, K. Hosogi, H. Nakata, M. Ehama, T. Kengaku, T. Nakazawa and H. Watanabe, "A 256 mW 40 Mbps Full-HD H.264 High-Profile Codec Featuring a Dual-Macroblock Pipeline Architecture in 65 nm CMOS," *IEEE Journal of Solid-State Circuits,* vol. 44, no. 4, pp. 1184-1191, 2009.

[23] J. Zhou, D. Zhou, W. Fei and S. Goto, "A high-performance CABAC encoder architecture for HEVC and H.264/AVC," in *International Conference on Image Processing (ICIP)*, 2013.

[24] J. Henkel and L. Yanbing, "Energy-conscious HW/SW-partitioning of embedded systems: a case study on an MPEG-2 encoder," in *International Workshop on Hardware/Software Codesign*, 1998.

[25] M. Zuluaga and N. Topham, "Design-Space Exploration of Resource-Sharing Solutions for Custom Instruction Set Extensions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD),* vol. 28, no. 12, pp. 1788-1801, 2009.

[26] A. Grudnitsky, L. Bauer and J. Henkel, "COREFAB: Concurrent Reconfigurable Fabric Utilization in Heterogeneous Multi-Core Systems," in *International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, 2014.

[27] M. Majer, J. Teich, A. Ahmadinia and C. Bobda, "The Erlangen Slot Machine: A Dynamically Reconfigurable FPGA-based Computer," *VLSI Signal Processing Systems for Signal, Image, and Video Technology,* vol. 47, no. 1, pp. 15-31, 2007.

Bibliography

[28] C. Smith, "120 Amazing YouTube Statistics," [Online]. Available: http://expandedramblings.com/index.php/youtube-statistics/. [Accessed 5 October 2015].

[29] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li and Y. Chen, "Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement," in *Design Automation Conference (DAC)*, 2008.

[30] X. Wu, J. Li, L. Zhang, E. Speight, R. Rajamony and Y. Xie, "Hybrid cache architecture with disparate memory technologies," in *International Symposium on Computer Architecture (ISCA)*, 2009.

[31] "Joint Collaborative Team on Video Coding (JCT-VC)," ITU, [Online]. Available: http://www.itu.int/en/ITU-T/studygroups/2013-2016/16/Pages/video/jctvc.aspx. [Accessed 7 October 2015].

[32] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer and T. Wedi, "Video coding with H.264/AVC: Tools, Performance, and Complexity," *IEEE Circuits and Systems Magazine,* vol. 4, no. 1, pp. 7-28, 2004.

[33] D. Grois, D. Marpe, A. Mulayoff, B. Itzhaky and O. Hadar, "Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders," in *Picture Coding Symposium (PCS)*, 2013.

[34] B. Girod, A. M. Aaron, S. Rane and D. Rebollo-Monedero, "Distributed Video Coding," *Proceedings of the IEEE,* vol. 93, no. 1, pp. 71-83, 2005.

[35] F. Dufaux and T. Ebrahimi, "Encoder and decoder side global and local motion estimation for Distributed Video Coding," in *International Workshop on Multimedia Signal Pro-cessing*, 2010.

[36] W. Huang, K. Rajamani, M. Stan and K. Skadron, "Scaling with Design Constraints: Predicting the Future of Big Chips," *IEEE Micro,* vol. 31, no. 4, pp. 16-29, 2011.

[37] M. Bohr, "A 30 Year Retrospective on Dennard's MOSFET Scaling Paper," *IEEE Solid-State Circuits Society Newsletter,* vol. 12, no. 1, pp. 11-13, 2007.

[38] M. Khan, M. Shafique and J. Henkel, "An adaptive complexity reduction scheme with fast prediction unit decision for HEVC intra encoding," in *International Conference on Image Processing (ICIP)*, 2013.

[39] M. Khan, M. Shafique and J. Henkel, "AMBER: Adaptive energy management for on-chip hybrid video memories," in *International Conference on Computer-Aided Design (ICCAD)*, 2013.

[40] R. Dennard, V. Rideout, E. Bassous and A. LeBlanc, "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE Journal of Solid-State Circuits,* vol. 9, no. 5, pp. 256-268, 1974.

[41] S. Huck, "Measuring Processor Power, TDP vs. ACP," Intel, 2011.

[42] S. Ghemawat, H. Gobioff and S.-T. Leung, "The Google File System," in *ACM Symposium on Operating Systems Principles*, 2003.

[43] S. Kumar, C. Kim and S. Sapatnekar, "Impact of NBTI on SRAM read stability and design for reliability," in *International Symposium on Quality Electronic Design (ISQED)*, 2006.

[44] M. U. K. Khan, M. Shafique and J. Henkel, "Software architecture of High Efficiency Video Coding for many-core systems with power-efficient workload balancing," in *Design, Automation and Test in Europe*, 2014.

[45] M. Shafique, M. U. K. Khan and J. Henkel, "Power Efficient and Workload Balanced Tiling for Parallelized High Efficiency Video Coding," in *International Conference on Image Processing*, 2014.

[46] M. U. K. Khan, M. Shafique and J. Henkel, "Hierarchical Power Budgeting for Dark Silicon Chips," in *International Symposium on Low Power Electronics and Design*, 2015.

[47] M. U. K. Khan, M. Shafique and J. Henkel, "Fast Hierarchical Intra Angular Mode Selection for High Efficiency Video Coding," in *International Conference on Image Processing*, 2014.

[48] M. U. K. Khan, M. Shafique and J. Henkel, "A Hierarchical Control Scheme for Energy Quota Distribution in Hybrid Distributed Video Coding," in *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2012.

[49] M. U. K. Khan, J. M. Borrmann, L. Bauer, M. Shafique and J. Henkel, "An H.264 Quad-FullHD low-latency intra video encoder," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2013.

[50] M. U. K. Khan, M. Shafique and J. Henkel, "Hardware-Software Collaborative Complexity Reduction Scheme for the Emerging HEVC Intra Encoder," in *Design, Automation and Test in Europe (DATE)*, 2013.

[51] M. U. K. Khan, M. Shafique and J. Henkel, "Power-efficient accelerator allocation in adaptive dark silicon many-core systems," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015.

[52] M. Shafique, M. U. K. Khan, O. Tüfek and J. Henkel, "EnAAM: energy-efficient anti-aging for on-chip video memories," in *Design Automation Conference (DAC)*, 2015.

[53] M. Shafique, M. U. K. Khan and J. Henkel, "Content-Aware Low-Power Configurable Aging Mitigation for SRAM Memories," *IEEE Transactions on Computers (TC),* 2016.

[54] J. Teich, J. Henkel, A. Herkersdorf, D. Schmitt-Landsiedel, W. Schröder-Preikschat and G. Snelting, "Invasive Computing: An Overview," in *Multiprocessor System-on-Chip*, Springer, 2010, pp. 241-268.

[55] "Dependable Embedded Systems," Karlsruhe Institute of Technology , [Online]. Available: http://spp1500.itec.kit.edu/index.php. [Accessed 9 October 2015].

Bibliography

[56] "SPP1500-Porjects - Phase II - Get-Sure," Karlsruhe Institute of Technology, [Online]. Available: http://spp1500.itec.kit.edu/286_300.php. [Accessed 9 October 2015].

[57] "SPP1500-Projects - Phase II -OTERA," Karlsruhe Institute of Technology, [Online]. Available: http://spp1500.itec.kit.edu/286_292.php. [Accessed 9 October 2015].

[58] "SPP1500-Projects - Phase II - VirTherm-3D," Karlsruhe Institute of Technology, [Online]. Available: http://spp1500.itec.kit.edu/286_291.php. [Accessed 9 October 2015].

[59] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," VCEG Contribution VCEG-M33, 2001.

[60] T. Wiegand, G. Sullivan, G. Bjontegaard and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 13, no. 7, pp. 560-576, 2003.

[61] C. Chen, C. Huang, Y. Chen and L. Chen, "Level C+ data reuse scheme for motion estimation with corresponding coding orders," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 16, no. 4, pp. 553-558, 2006.

[62] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Transactions on Image Processing,* vol. 9, no. 2, pp. 287-290, 2000.

[63] F. Bossen, B. Bross, K. Suhring and D. Flynn, "HEVC Complexity and Implementation Analysis," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 22, no. 12, pp. 1685-1696, 2012.

[64] V. Sze, D. F. Finchelstein, M. E. Sinangil and A. P. Chandraksan, "A 0.7-V 1.8-mW H.264/AVC 720p video decoder," *IEEE Journal of Solid-Sate Circuits,* vol. 44, no. 11, pp. 2943-2956, 2009.

[65] F. Sampaio, M. Shafique, B. Zatt, S. Bampi and J. Henkel, "Energy-efficient Architecture for Advanced Video Memory," in *International Conference on Computer-Aided Design*, 2014.

[66] N. Purnachand, L. N. Alves and A. Navarro, "Improvements to TZ search motion estimation algorithm for multiview video cod-ing," 2012.

[67] M. Shafique, L. Bauer and J. Henkel, "enBudget: A Run-Time Adaptive Predictive Energy-Budgeting Scheme for Energy-Aware Motion Estimation in H.264/MPEG-4 AVC Video Encoder," in *Design, Automation and Test in Europe*, 2010.

[68] A. Gurhanli, C.-P. Chen and S.-H. Hung, "GOP-level parallelization of the H.264 decoder without a start-code scanner," in *International Conference on Signal Processing Systems (ICSPS)*, 2010.

[69] "VideoLAN - x264," [Online]. Available: http://www.videolan.org/developers/x264.html. [Accessed 5 October 2015].

[70] L. Zhao, J. Xu, Y. Zhou and M. Ai, "A dynamic slice control scheme for slice-parallel video encoding," in *International Conference on Image Processing*, 2012.

[71] K. Ba, X. Jin and S. Goto, "A Dynamic Slice-resize Algorithm for Fast H.264/AVC Parallel Encoder," in *International Symposium on Intelligent Signal Processing and Communication Systems*, 2010.

[72] I. Ahmad and A. Ghafoor, "Semi-distributed load balancing for massively parallel multicomputer systems," *IEEE Transactions on Software Engineering,* vol. 17, no. 10, pp. 987-1004, 1991.

[73] R. Williams, "Performance of dynamic load balancing algorithms for unstructured mesh calculations," *Concurrency: Practice and experience,* vol. 3, no. 5, pp. 457-481, 1991.

[74] "Juice Encoder– 4 in 1 MPEG-4 AVC/H.264 HD encoder," Antik Technology, [Online]. Available: http://www.antiktech.com/iptv-products/juice-encoder-EN-5004-5008/.

[75] "Marvell 88DE3100 High-Definition Secure Media Processor System-on-Chip (SoC)," [Online]. Available: http://www.marvell.com/digital-entertainment/armada-1500/assets/Marvell-ARMADA-1500-Product-Brief.pdf/.

[76] "Distributed Coding for Video Services (DISCOVER). Application scenarios and functionalities for DVC".

[77] A. Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder," *IEEE Transaction on Information Theory,* vol. 22, pp. 1-10, 1976.

[78] R. Puri and K. Ramchandran, "PRISM: an uplink-friendly multimedia coding paradigm," in *International Conference on Acoustics, Speech, and Signal Processing*, 2003.

[79] J. Chen, A. Khisti, D. Malioutov and J. Yedidia, "Distributed source coding using serially-concatenated-accumulate codes," in *Information Theory Workshop*, 2004.

[80] H.-y. Tseng, Y.-c. Shen and J.-l. Wu, "Distributed Video Coding with Compressive Measurements," in *International conference on Multimedia* , 2011.

[81] D. Sejdinovic, R. J. Piechocki and A. Doufexi, "Rateless distributed source code design," in *Mobile Multimedia Communica-tions Conference*, 2009.

[82] S.-y. Chien, T.-Y. Cheng, C.-C. Chiu, P.-K. Tsung, C.-H. Lee, V. Somayazulu and Y.-K. Chen, "Power Optimization of Wireless Video Sensor Nodes in M2M Networks," in *Asia and South Pacific Design Automation Conference*, 2012.

[83] Y.-W. Huang, T.-C. Chen, C.-H. Tsai, C.-Y. Chen, T.-W. Chen, C.-S. Chen, C.-F. Shen, S.-Y. Ma, T.-C. Wang, B.-Y. Hsieh, H.-C. Fang and L.-G. Chen, "A 1.3TOPS H.264/AVC single-chip encoder for HDTV applications," in *International Solid-State Circuits Conference*, 2005.

[84] C.-C. Chiu, S.-Y. Chien, C.-h. Lee, V. Somayazulu and Y.-K. Chen, "Distributed video coding: a promising solution for distributed wireless video sensors or not?," in *Visual Communications and Image Processing*, 2011.

[85] M. Shafique, M. U. K. Khan and J. Henkel, "Content-Driven Adaptive Computation Offloading for Energy-Aware Hybrid Distributed Video Coding," in *International Symposium on Low Power*

Electronics and Design (ISLPED), 2013.

[86] K. Kumar, J. Liu, Y.-H. Lu and B. Bhargava, "A Survey of Computation Offloading for Mobile Systems," *Mobile Networks and Applications,* vol. 18, no. 1, pp. 129-140, 2012.

[87] D. K. Schroder and J. A. Babcock, "Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing," *Journal of Applied Physics,* vol. 94, no. 1, pp. 1-18, 2003.

[88] J. Shin, V. Zyuban, P. Bose and T. Pinkston, "A proactive wearout recovery approach for exploiting microarchi-tectural redundancy to extend cache SRAM lifetime," in *International Symposium on Computer Architecture (ISCA)*, 2008.

[89] A. Tiwari and J. Torrellas, "Facelift: Hiding and slowing down aging in multicores," in *International Symposium on Microarchitecture (MICRO)*, 2008.

[90] R. Vattikonda, W. Wang and Y. Cao, "Modeling and minimization of PMOS NBTI effect for robust nanometer design," in *Design Automation Conference (DAC)*, 2006.

[91] J. B. Velamala, K. Sutaria, T. Sato and Y. Cao, "Physics matters: statistical aging prediction under trapping/detrapping," in *Design Automation Conference (DAC)*, 2012.

[92] A. Singh, M. Shafique, A. Kumar and J. Henkel, "Mapping on Multi/Many-Core Systems: Survey of Current and Emerging Trends," in *Design Automation Conference (DAC)*, 2013.

[93] C. C. Chi, M. Alvarez-Mesa, B. Juurlink, G. Clare, F. Henry, S. Pateux and T. Schierl, "Parallel scalability and efficiency of HEVC parallelization approaches," *IEEE Transactions on Circuits and Systems on Video Technology,* vol. 22, no. 12, pp. 1827-1838, 2012.

[94] M. Alvanos, G. Tzenakis, D. S. Nikolopoulos and A. Bilas, "Task-based parallel H. 264 video encoding for explicit communication architectures," in *International Conference on Embedded Computer Systems*, 2011.

[95] O. Brun, V. Teuliere and J. M. Garcia, "Parallel Particle Filtering," *Journal of Parallel and Distributed Computing,* vol. 62, no. 7, p. 1186–1202, 2002.

[96] K. Rujirakul, C. So-In and B. Arnonkijpanich, "PEM-PCA: A Parallel Expectation-Maximization PCA Face Recognition Architecture," *The Scientific World Journal,* 2014.

[97] X.-Y. Jing, S. Li, D. Zhang, J. Yang and J.-Y. Yang, "Supervised and Unsupervised Parallel Subspace Learning for Large-Scale Image Recognition," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 22, no. 10, pp. 1497-1511, 2012.

[98] C. Dong, H. Zhao and W. Wang, "Parallel Nonnegative Matrix Factorization Algorithm on the Distributed Memory Platform," *International Journal of Parallel Programming,* vol. 38, no. 2, pp. 117-137, 2010.

[99] S. Shah and R. Kothari, "Convergence of the dynamic load balancing problem to Nash equilibrium

using distributed local interactions," *Information Sciences,* vol. 221, pp. 297-305, 2013.

[100] Y. Drougas, T. Repantis and V. Kalogeraki, "Load balancing techniques for distributed stream processing applications in overlay environments," in *IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, 2006.

[101] T. G. Robertazzi, "Ten Reasons to Use Divisible Load Theory," *Computer,* vol. 36, no. 5, pp. 63-68, 2003.

[102] K. Wang, X. Zhou, T. Li, D. Zhao, M. Lang and I. Raicu, "Optimizing load balancing and data-locality with data-aware scheduling," in *International Conference on Big Data*, 2014.

[103] Y. Turakhia, B. Raghunathan, S. Garg and D. Marculescu, "HaDeS: Architectural synthesis for heterogeneous dark silicon chip multi-processors," in *Design Automation Conference (DAC)*, 2013.

[104] M. Buss, T. Givargis and N. Dutt, "Exploring efficient operating points for voltage scaled embedded processor cores," in *Real-Time Systems Symposium (RTSS)*, 2003.

[105] A. Matthur and P. Mundur, "Dynamic load balancing across mirrored multimedia servers," in *International Conference on Multimedia and Expo*, 2003.

[106] K. Bowman, S. Duvall and J. Meindl, "Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration," *IEEE Journal of Solid-State Circuits,* vol. 37, no. 2, pp. 183-190, 2002.

[107] J. Kim, S. Yoo and C.-M. Kyung, "Program Phase-Aware Dynamic Voltage Scaling Under Variable Computational Workload and Memory Stall Environment," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 30, no. 1, pp. 110-123, 2011.

[108] V. Devadas and H. Aydin, "DFR-EDF: A Unified Energy Management Framework for Real-Time Systems," in *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2010.

[109] M. Dehyadegari, A. Marongiu, M. Kakoee, S. mohammadi, N. Yazdani and L. Benini, "Architecture Support for Tightly-Coupled Multi-Core Clusters with Shared-Memory HW Accelerators," *IEEE Transactions on Computer,* vol. 64, no. 8, pp. 2132-2144, 2015.

[110] S. Sarma, T. Muck, L. Bathen, N. Dutt and A. Nicolau, "SmartBalance: A sensing-driven linux load balancer for energy efficiency of heterogeneous MPSoCs," in *Design Automation Conference (DAC)*, 2015.

[111] "ARM big.LITTLE Architecture," ARM, [Online]. Available: http://www.arm.com/products/processors/technologies/biglittleprocessing.php. [Accessed 07 08 2015].

[112] S. Momcilovic, N. Roma and L. Sousa, "Exploiting task and data parallelism for advanced video coding on hybrid CPU + GPU platforms," *Journal of Real-Time Image Processing,* pp. 1-17, 2013.

[113] C. Jian and L. John, "Efficient program scheduling for heterogeneous multi-core processors," in

Design Automation Conference (DAC), 2009.

[114] T. Mühlbauer, W. Rödiger, R. Seilbeck, A. Kemper and T. Neumann, "Heterogeneity-conscious Parallel Query Execution: Getting a Better Mileage While Driving Faster!," in *International Workshop on Data Management on New Hardware*, 2014.

[115] M. Shafique, B. Molkenthin and J. Henkel, "An HVS-based Adaptive Computational Complexity Reduction Scheme for H.264/AVC Video Encoder using Prognostic Early Mode Exclusion," in *Design, Automation and Test in Europe Conference (DATE)*, 2010.

[116] M. Bariani, P. Lambruschini and M. Raggio, "An Efficient Multi-Core SIMD Implementation for H.264/AVC Encoder," in *VLSI Design*, 2012.

[117] A. Rodríguez, A. González and M. P. Malumbres, "Hierarchical Parallelization of an H.264/AVC Video Encoder," in *International Symposium on Parallel Computing in Electrical Engineering*, 2006.

[118] P. Gong, Y. Basciftci and F. Ozguner, "A Parallel Resampling Algorithm for Particle Filtering on Shared-Memory Architectures," in *Parallel and Distributed Processing Symposium Workshops*, 2012.

[119] S. Cuomo, P. D. Michele and F. Piccialli, "3D Data Denoising via Nonlocal Means Filter by Using Parallel GPU Strategies," in *Computational and Mathematical Methods in Medicine*, 2014.

[120] M. Moustafa, H. M. Ebied, A. Helmy, T. M. Nazamy and M. F. Tolba, "Satellite Super Resolution Image Reconstruction Based on Parallel Support Vector Regression," in *Advanced Machine Learning Technologies and Applications*, Springer, 2014, pp. 223-235.

[121] P. Garcia Freitas, M. Farias and A. De Araujo, "A Parallel Framework for Video Super-Resolution," in *Graphics, Patterns and Images (SIBGRAPI)*, 2014.

[122] B. Jung and B. Jeon, "Adaptive slice-level parallelism for H.264/AVC encoding using pre macroblock mode selection," *Journal of Visual Communication Image Representation,* vol. 19, no. 8, pp. 558-572, 2008.

[123] W. Jiang, H. Mal and Y. Chen, "Gradient based fast mode decision algorithm for intra prediction in HEVC," in *International Conference on Consumer Electronics, Communications and Networks*, 2012.

[124] M. B. Cassa, M. Naccari and F. Pereira, "Fast Rate Distortion Optimization for the Emerging HEVC Standard," in *Picture Coding Symposium*, 2012.

[125] H. Zhang and Z. Ma, "Fast Intra Prediction for High Efficiency Video Coding," in *Advances in Multimedia Information Processing*, 2012.

[126] H. Sun, D. Zhou and S. Goto, "A low-complexity HEVC Intra prediction algorithm based on level and mode filtering,," in *International Conference on Multimedia and Expo (ICME)*, 2012.

[127] F. Pan, X. Lin, S. Rahardja, K. P. Lim, Z. G. Li, D. Wu and S. Wu, "Fast mode decision algorithm for intraprediction in H.264/AVC video coding," *IEEE Transactions on Circuits and Systems for*

Video Technology, vol. 15, no. 7, pp. 813-822, 2005.

[128] A. C. Tsai, A. Paul, J. C. Wang and J. F. Wang, "Intensity gradient technique for efficient intra-prediction in H.264/AVC," *IEEE Transactions on Circuits and Systems on Video Technology,* vol. 18, no. 5, pp. 694-698, 2008.

[129] T. A. Fonseca, Y. Liu and R. L. D. Queiroz, "Open-Loop Prediction in H.264 / AVC for High Definition Sequences," in *SBrT*, 2007.

[130] G. Tian and S. Goto, "Content adaptive prediction unit size decision algorithm for HEVC intra coding," in *Picture Coding Symposium*, 2012.

[131] L. Zhao, L. Zhang, S. Ma and D. Zhao, "Fast mode decision algorithm for Intra prediction in HEVC," in *Visual Communications and Image Processing (VCIP)*, 2011.

[132] T. d. Silva, L. V. Agostini and .. L. A. D. S. C. Cruz, "Fast HEVC Intra prediction mode decision based on EDGE direction information," in *European Signal Processing Conference (Eusipco)*, 2012.

[133] G. D. Haan and P. Biezen, "An Efficient True-Motion Estimator Using Candidate Vectors from a Parametric Motion Model," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 8, no. 9, pp. 86-91, 1998.

[134] H. Shim and C.-M. Kyung, "Selective search area reuse algorithm for low external memory access motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 19, no. 7, pp. 1044-1050, 2009.

[135] Z. kun, Y. chun, L. Qiang and Z. Yuzhuo, "A Fast Block Type Decision Method for H.264/AVC Intra Prediction," in *International Conference on Advanced Communication Technology*, 2007.

[136] Y.-k. Lin and T. Chang, "Fast Block Type Decision Algorithm for Intra Prediction in H.264/FRex," in *Internatianal conference on Image Processing (ICIP)*, 2005.

[137] M. Kivanc Mihcak, I. Kozintsev, K. Ramchandran and P. Moulin, "Low-complexity image denoising based on statistical modeling of wavelet coefficients," *IEEE Signal Processing Letters,* vol. 6, no. 12, pp. 300-303, 1999.

[138] M. U. K. Khan, A. Bais, M. Khawaja, G. M. Hassan and R. Arshad, "A Swift and Memory Efficient Hough Transform for Systems with Limited Fast Memory," in *International Conference on Image Analysis and Recognition (ICIAR)*, 2009.

[139] "OpenCV," [Online]. Available: http://opencv.org/. [Accessed 08 08 2015].

[140] "OpenVX," [Online]. Available: https://www.khronos.org/openvx/. [Accessed 08 08 2015].

[141] G. Chen, B.-T. Kang, M. Kandemir, N. Vijaykrishnan, M. Irwin and R. Chandramouli, "Studying Energy Trade Offs in Offloading Computation/Compilation in Java-Enabled Mobile Devices," *IEEE Transactions on Parallel Distributed Systems,* vol. 15, no. 9, pp. 795-809, 2004.

[142] Z. Li, C. Wang and R. Xu, "Computation offloading to save energy on handheld devices: a partition scheme," in *International Conference on Compilers, Architectures and Synthesis of Embedded Systems*, 2001.

[143] Z. Li, C. Wang and R. Xu, "Task Allocation for Distributed Multi-media Processing on Wirelessly Networked Handheld Devices," in *Vehicle Navigation and Information Systems Conference*, 1993.

[144] G. H.-Canepa and D. Lee, "An Adaptable Application Offloading Scheme Based on Application Behavior," in *Conference on Advanced Information Networking and Applications*, 2008.

[145] C. Brites and F. Pereira, "Distributed video coding: Assessing the HEVC upgrade," *Signal Processing: Image Communication,* vol. 32, pp. 81-105, 2015.

[146] E. Martinian, A. Vetro, J. S. Yedidia, J. Ascenso, A. Khisti and D. Malioutov, "Hybrid distributed video coding using SCA codes," in *Workshop on Multimedia Signal Processing*, 2006.

[147] M. Shafique, B. Zatt, F. L. Walter, S. Bampi and J. Henkel, "Adaptive Power Management of On-Chip Video Memory for Mul-tiview Video Coding," in *Design Automation Conference*, 2012.

[148] T. S. Muthukaruppan, M. Pricopi, V. Venkataramani, T. Mitra and S. Vishin, "Hierarchical Power Management for Asymmetric Multi-Core in Dark Silicon Era," in *Design Automation Conference (DAC)*, 2013.

[149] H. Khdr, T. Ebi, M. Shafique, H. Amrouch and J. Henkel, "mDTM: Multi-objective dynamic thermal management for on-chip systems," in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2014.

[150] V. Devadas and H. Aydin, "On the Interplay of Voltage/Frequency Scaling and Device Power Management for Frame-Based Real-Time Embedded Applications," *IEEE Transactions on Computers,* vol. 61, no. 1, pp. 31-44, 2012.

[151] C. Isci, A. Buyuktosunoglu, C. Cher, P. Bose and M. Martonosi, "An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget," in *Microarchitecture* , 2006.

[152] S. Pagani, H. Khdr, W. Munawar, J.-J. Chen, M. Shafique, M. Li and J. Henkel, "TSP: Thermal Safe Power: Efficient Power Budgeting for Many-core Systems in Dark Silicon," in *International Conference on Hardware/Software Codesign and System Synthesis*, 2014.

[153] T. Muthukaruppan, M. Pricopi, V. Venkataramani, T. Mitra and S. Vishin, "Hierarchical power management for asymmetric multi-core in dark silicon era," in *Design Automation Conference*, 2013.

[154] A. Mishra, S. Srikantaiah, M. Kandemir and C. R. Das, "CPM in CMPs: Coordinated power management in chip-multiprocessors," in *International Conference on High Performance Computing, Networking, Storage and Analysis*, 2010.

[155] J. a. Winter, D. H. Albonesi and C. a. Shoemaker, "Scalable thread scheduling and global power management for heterogeneous many-core architectures," in *Parallel Architectures and Compilation*,

2010.

[156] A. Sharifi, A. Mishra, S. Srikantaiah, M. Kandemir and C. R. Das, "PEPON: performance-aware hierarchical power budgeting for NoC based multicores," in *Parallel Architectures and Compilation Techniques* , 2012.

[157] M. Shafique, S. Garg, J. Henkel and D. Marculescu, "The EDA Challenges in the Dark Silicon Era," in *Design Automation Conference*, 2014.

[158] Y.-W. Huang, B.-Y. Hsieh, T.-C. Chen and L.-G. Chen, "Analysis, fast algorithm, and VLSI architec-ture design for H.264/AVC intra frame coder," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 15, no. 3, pp. 378-401, 2005.

[159] M. Roszkowski and G. Pastuszak, "Intra prediction hardware module for high-profile H.264/AVC encoder," in *Signal Processing Algorithms, Architectures, Arrangements and Applications Conference*, 2010.

[160] G. He, D. Zhou, J. Zhou and S. Goto, "Intra prediction architecture for H.264/AVC QFHD encoder," in *Picture Coding Symposium*, 2010.

[161] C. Diniz, B. Zatt, C. Thiele, A. Susin, S. Bampi, F. Sampaio, D. Palomino and L. Agostini, "A high throughput H.264/AVC intra-frame encod-ing loop architecture for HD1080p," in *International Symposium on Circuits and Systems*, 2011.

[162] F. Li, G. Shi and F. Wu, "An efficient VLSI architecture for 4×4 intra prediction in the High Efficiency Video Coding (HEVC) standard," in *International Conference on Image Processing*, 2011.

[163] T. Cervero, A. Otero, S. López, E. de la Torre, G. Callicó, T. Riesgo and R. Sarmiento, "A scalable H.264/AVC deblocking filter architecture," *Journal of Real-Time Image Processing,* pp. 1-25, 2013.

[164] S. Mangard, M. Aigner and S. Dominikus, "A Highly Regular and Scalable AES Hardware Architecture," *IEEE Transactions on Computers,* vol. 52, no. 4, pp. 483-491, 2003.

[165] G. V. Varatkar and N. R. Shanbhag, "Error-Resilient Motion Estimation Architecture," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 16, no. 10, pp. 1399-1412, 2008.

[166] S. Saponara and L. Fanucci, "Data-adaptive motion estimation algorithm and VLSI architecture design for low-power video systems," *IEE Proceedings-Computers and Digital Techniques,* vol. 151, no. 1, pp. 51-59, 2004.

[167] C.-Y. Tsai, C. Chung, Y.-H. Chen, T.-C. Chen and L.-G. Chen, "Low power cache algorithm and architecture design for fast motion estimation in H. 264/AVC encoder system," in *In-ternational Conference on Acoustics, Speech and Signal Processing*, 2007.

[168] N. S. Kim, T. Austin, D. Blaauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir and V. Narayanan, "Leakage current: Moore's law meets static pow-e," *Computers,* vol. 36, no. 12, pp. 68-75, 2003.

Bibliography

[169] Z. Ma and A. Segall, "Frame Buffer Compression for Low-Power Video Coding," in *International Conference on Image Processing*, 2011.

[170] M.-Y. Hsu, "Scalable Module-Based Architecture for MPEG-4 BMA Motion Estimation," 2000.

[171] J.-C.Tuan, T.-S. Chang and C.-W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 12, no. 1, pp. 61-72, 2002.

[172] Z. Diao, Z. Li, S. Wang, Y. Ding, A. Panchula, E. Chen, L.-C. Wang and Y. Huai, "Spin-transfer torque switching in magnetic tunnel junctions and spin-transfer torque random access memory," *Journal of Physics: Condensed Matter,* vol. 19, no. 16, pp. 1-13, 2007.

[173] M. K. Qureshi, V. Srinivasan and J. a. Rivers, "Scalable high perfor-mance main memory system using phase-change memory technology," in *International Symposium on Computer Architec-ture (ISCA)*, 2009.

[174] S. Hanzawa, N. Kitai, K. Osada, A. Kotabe, Y. Matsui, N. Matsuzaki, N. Takaura, M. Moniwa and T. Kawahara, "A 512KB Embed-ded Phase Change Memory with 416kB/s Write Throughput at 100uA Cell Write Current," in *International Solid-State Circuits Conference (ISSCC)*, 2007.

[175] S. Yang and Y. Ryu, "A Memory Management Scheme for Hybrid Memory Architecture in Mission Critical Computers," in *International Conference on Software Technology*, 2012.

[176] G. Dhiman, R. Ayoub and T. Rosing, "PDRAM: a hybrid PRAM and DRAM main memory system," in *Design Automation Conference*, 2009.

[177] L. Bathen and N. Dutt, "HaVOC: A hybrid memory-aware virtualization layer for on-chip distributed ScratchPad and Non-Volatile Memories," in *Design Automation Conference*, 2012.

[178] L. C. Stancu, L. A. D. Bathen, N. Dutt and A. Nicolau, "AVid : Annotation Driven Video Decoding for Hybrid Memories," in *Embedded Systems for Real-Time Multimedia*, 2012.

[179] R. Desikan, C. Lefurgy, S. Keckler and D. Burger, "On-chip MRAM as a high-bandwidth, low-latency replacement for DRAM physical memories," University of Texas at Austin, 2002.

[180] K. Nomura, K. Abe, H. Yoda and S. Fujita, "Ultra low power processor using perpendicular-STT-MRAM/SRAM based hy-brid cache toward next generation normally-off computers," *Journal of Applied Physics,* vol. 111, no. 7, 2012.

[181] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan and P. Hanrahan, "Larrabee: A Many-Core x86 Architecture for Visual Computing," *ACM Transactions on Graphics,* vol. 27, no. 3, 2008.

[182] M. Mattina, *Architecture and Performance of the Tile-GX Processor Family,* White Paper, 2014.

[183] M. Shafique, L. Bauer and J. Henkel, "Optimizing the H.264/AVC Video Encoder Application Structure for Reconfigurable and Application-Specific Platforms," *Journal of Signal Processing*

Systems (JSPS), vol. 60, no. 2, pp. 183-210, 2010.

[184] C. Liu, O. Granados, R. Duarte and J. Andrian, "Energy Efficient Architecture Using Hardware Acceleration for Software Defined Radio Components," *Journal of Information Processing Systems,* vol. 8, no. 1, pp. 133-144, 2012.

[185] "Nios II Custom Instruction User Guide," Altera, 2011.

[186] H. Shojania and L. Baochun, "Parallelized Progressive Network Coding with Hardware Acceleration," in *International Workshop on Quality of Service*, 2007.

[187] H. C. Doan, H. Javaid and S. Parameswaran, "Flexible and scalable implementation of H.264/AVC encoder for multiple resolutions using ASIPs," in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2014.

[188] S. D. Kim, J. H. Lee, C. J. Hyun and M. H. Sunwoo, "ASIP approach for implementation of H.264/AVC," in *Asia and South Pacific Conference on Design Automation (ASP-DAC)*, 2006.

[189] S. Swanson and M. B. Taylor, "GreenDroid: Exploring the Next Evolution in Smartphone Application Processors," *IEEE Communications Magazine,* pp. 112-119, April 2011.

[190] J. Teich, J. Henkel, A. Herkersdorf, D. Schmitt-Landsiedel, W. Schröder-Preikschat and G. Snelting, "Invasive Computing: An Overview," in *Multiprocessor System-on-Chip*, Springer New York, 2011, pp. 241-268.

[191] D. Sheldon and A. Forin, "An Online Scheduler for Hardware Accelerators on General Purpose Operating Systems," Microsoft Research, 2010.

[192] C. Huang, D. Sheldon and F. Vahid, "Dynamic Tuning of Configurable Architectures: the AWW Online Algorithm," in *International Conference on Hardware/Software Codesign and System Synthesis*, 2008.

[193] T. Majumder, P. P. Pande and A. Kalyanaraman, "High-throughput, Energy-Efficient Network-on-Chip-Based Hardware Accelerators," *Journal of Sustainable Computing: Informatics and Systems,* vol. 3, no. 1, pp. 36-46, 2013.

[194] J. Cong, M. A. Ghodrat, M. Gill, B. Grigorian and G. Reinman, "Architecture support for accelerator-rich CMPs," in *Design Automation Conference*, 2012.

[195] E. Cota, P. Mantovani, M. Petracca, M. Casu and L. Carloni, "Accelerator Memory Reuse in the Dark Silicon Era," *Computer Architecture Letters,* pp. 1-4, 2012.

[196] J. A. Clemente, I. V. Beretta, V. Rana, D. Atienza and D. Sciuto, "A Mapping-Scheduling Algorithm for Hardware Acceleration on Reconfigurable Platform," *Transactions on Reconfigurable Technology and Systems,* vol. 7, no. 2, 2014.

[197] S. Paul, R. Karam, S. Bhunia and R. Puri, "Energy-Efficient Hardware Acceleration through Computing in the Memory," in *Design, Automation and Test in Europe (DATE)*, 2014.

Bibliography

[198] S. Kothawade, K. Chakraborty and S. Roy, "Analysis and mitigation of NBTI aging in register file: An end-to-end approach," in *International Symposium on Quality Electronic Design (ISQED)*, 2011.

[199] H. Amrouch, T. Ebi and J. Henkel, "Stress Balancing to Mitigate NBTI Effects in Register Files," in *Dependable Systems and Networks (DSN)*, 2013.

[200] S. Wang, T. Jin, C. Zheng and G. Duan, "Low power aging-aware register file design by duty cycle balancing," in *Design, Automation and Test in Europe (DATE)*, 2012.

[201] T. Siddiqua and S. Gurumurthi, "Recovery boosting: A technique to enhance NBTI recovery in SRAM arrays," in *Annual Symposium on VLSI*, 2010.

[202] A. Sil, S. Ghosh, N. Gogineni and M. Bayoumi, "A Novel High Write Speed, Low Power, Read-SNM-Free 6T SRAM Cell," in *Midwest Symposium on Circuits and Systems*, 2008.

[203] J. Abella, X. Vera, O. Unsal and A. Gonzalez, "NBTI-resilient memory cells with NAND gates". US Patent US20080084732 A1, 2008.

[204] S. Wang, G. Duan, C. Zheng and T. Jin, "Combating NBTI-induced aging in data caches," in *Great lakes symposium on VLSI*, 2013.

[205] E. Gunadi, A. A. Sinkar, N. S. Kim and M. H. Lipasti, "Combating Aging with the Colt Duty Cycle Equalizer," in *International Symposium on Microarchitecture*, 2010.

[206] A. Calimera, M. Loghi, E. Macii and M. Poncino, "Partitioned cache architectures for reduced NBTI-induced aging," in *Design, Automation and Test in Europe (DATE)*, 2011.

[207] J. Henkel, H. Bukhari, S. Garg, M. U. K. Khan, H. Khdr, F. Kriebel, U. Ogras, S. Parameswaran and M. Shafique, "Dark Silicon – From Computation to Communication," in *International Symposium on Networks-on-Chip (NOCs)*, 2015.

[208] H. Esmaeilzadeh, A. Sampson, L. Ceze and D. Burger, "Neural Acceleration for General-Purpose Approximate Programs," in *International Symposium on Microarchitecture*, 2012.

[209] D. Mahajan, A. Yazdanbakhsh, J. Park, B. Thwaites and H. Esmaeilzadeh, "Prediction-Based Quality Control for Approximate Accelerators," in *Workshop on Approximate Computing Across the System Stack*, 2015.

[210] J. Allred, S. Roy and K. Chakraborty, "Designing for Dark Silicon: A Methodological Perspective on Energy Efficient Systems," in *International Symposium on Low Power Electronics and Design (ISLPED)*, 2012.

[211] G. Venkatesh, J. Sampson, N. Goulding, S. Garcia, V. Bryksin, J. Lugo-Martinez, S. Swanson and M. B. Taylor, "Conservation Cores: Reducing the Energy of Mature Computations," in *Architectural Support for Programming Languages and Operating Systems*, 2010.

[212] K. Swaminathan, E. Kultursay, V. Saripalli, V. Narayanan, M. Kandemir and S. Datta, "Steep-

Slope Devices: From Dark to Dim Silicon," *IEEE Micro,* vol. 33, no. 5, pp. 50-59, 2013.

[213] H. Bokhari, H. Javaid, M. Shafique, J. Henkel and S. Parameswaran, "darkNoC: Designing energy-efficient network-on-chip with multi-Vt cells for dark silicon," in *Design Automation Conference (DAC)*, 2014.

[214] B. Raghunathan, Y. Turakhia, S. Garg and D. Marculescu, "Cherry-picking: Exploiting process variations in dark-silicon homogeneous chip multi-processors," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2013.

[215] M. Shafique, D. Gnad, S. Garg and J. Henkel, "Variability-aware Dark Silicon Management in On-chip Many-core Systems," in *Design, Automation and Test in Europe Conference and Exhibition*, 2015.

[216] R. Jevtic, H.-P. Le, M. Blagojevic, S. Bailey, K. Asanovic, E. Alon and B. Nikolic, "Per-Core DVFS With Switched-Capacitor Converters for Energy Efficiency in Manycore Processors," *IEEE Transactions on Very Large Sclae Integration Systems (TVLSI),* vol. 23, no. 4, pp. 723-730, 2015.

[217] W. Kim, M. Gupta, G.-Y. Wei and D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," in *International Symposium on High Performance Computer Architecture (HPCA)*, 2008.

[218] W. Lee, Y. Wang and M. Pedram, "VRCon: dynamic reconfiguration of voltage regulators in a multicore platform," in *Design, Automation & Test in Europe and Exhibition (DATE)*, 2014.

[219] "libdvfs," [Online]. Available: https://github.com/LittleWhite-tb/libdvfs. [Accessed 12 08 2015].

[220] "OMAP 3 Processor," Texas Instruments, [Online]. Available: http://www.ti.com/product/omap3530. [Accessed 13 08 2015].

[221] D. G. Feitelson and L. Rudolph, "Towards convergence in job schedulers for parallel supercomputers," in *Workshop on Job Scheduling Strategies for Parallel Processing*, 1996.

[222] T. Desell, K. E. Maghraoui and C. A. Varela, "Malleable applications for scalable high performance computing," *Cluster Computing,* vol. 10, no. 3, pp. 323-337, 2007.

[223] "OpenMP," [Online]. Available: http://openmp.org/wp/. [Accessed 12 08 2015].

[224] K. Kuhn, C. Kenyon, A. Kornfeld, M. Liu, A. Maheshwari, W.-k. Shih, S. Sivakumar, G. Taylor, P. VanDerVoorn and K. Zawadzki, "Managing Process Variation in Intel's 45nm CMOS Technology," *Intel Technology Journal,* vol. 12, no. 2, pp. 93-109, 2008.

[225] T. Nguyen and D. Marpe, "Performance analysis of HEVC-based intra coding for still image compression," in *Picture Coding Symposium (PCS)*, 2012.

[226] T. Carlson, W. Heirman and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *High Performance Computing, Networking, Storage and Analysis*, 2011.

Bibliography

[227] S. Li, J. H. Ahn, R. Strong, J. Brockman, D. Tullsen and N. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Microarchitecture*, 2009.

[228] L. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer,* vol. 40, no. 12, pp. 33-37, 2007.

[229] G. J. Sullivan, J. Ohm, W. Han and T. Wiegand, "Overview of the High Efficiency Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 22, no. 12, pp. 1649-1668, 2012.

[230] D. Gangadharan, L. Phan, S. Chakraborty, R. Zimmer-mann and L. Insup, "Video Quality Driven Buffer Sizing via Frame Drops," in *Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2011.

[231] P.-K. Tsung, W.-Y. Chen, L.-F. Ding, S.-Y. Chien and L.-G. Chen, "Cache-based integer motion / disparity estimation for Quad-HD H.264/AVC and HD multiview video coding," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2009.

[232] K. Ning and D. Kaeli, "Power Aware External Bus Arbitration for System-on-a-Chip Embedded Systems," *High Performance Embedded Architectures and Compilers,* vol. 3793, pp. 87-101, 2005.

[233] G. H. Loh, "Extending the effectiveness of 3D-stacked DRAM caches with an adaptive multi-queue policy," in *International Symposium on Microarchitecture (MICRO)*, 2009.

[234] F. Bossen, "Common test conditions," Joint Collaborative Team on Video Coding (JCT-VC) Doc. I1100, 2012.

[235] "Common YUV 4:2:0 test sequences," [Online]. Available: https://media.xiph.org/video/derf. [Accessed 16 Aug 2015].

[236] E. Cesar, A. Moreno, J. Sorribes and E. Luque, "Modeling Master/Worker applications for automatic performance tuning," *Parallel Computing,* vol. 32, no. 7, pp. 568-589, 2006.

[237] C. J. Lian, S. Y. Chien, C. P. Lin, P. C. Tseng and L. G. Chen, "Power-aware multimedia: concepts and design perspectives," *IEEE Circuits and Systems Magazine,* p. 26–34, 2007.

[238] D. E. Knuth, The Art of Computer Programming, Addison-Wesley, 1998, p. 232.

[239] "ces265: Open-source C++ based multithreaded HEVC software," Chair for Embedded Systems, [Online]. Available: http://ces.itec.kit.edu/ces265/. [Accessed 06 08 2014].

[240] T. F. Chan, G. H. Golub and R. J. LeVequ, "Updating Formulae and a Pairwise Algorithm for Computing Sample Variances," Stanford University, Stanford, CA, 1979.

[241] M. Shafique, B. Zatt, F. L. Walter, S. Bampi and J. Henkel, "Adaptive power management of on-chip video memory for multiview video coding," in *Design Automation Conference*, 2012.

[242] ITU, [Online]. Available: http://wftp3.itu.int/av-arch/video-site/sequences. [Accessed 15 October 2015].

[243] J. Ziegler and N. B. Nichols, "Optimum Settings for Automatic Controllers," *Journal of Dynamic Systems, Measurement, and Control,* 1993.

[244] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *International Conference on*, 2000.

[245] Altera, "External Memory Interface Handbook," June 2011. [Online]. Available: http://www.altera.com/literature/hb/external-memory/emi.pdf. [Accessed 29 September 2015].

[246] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal,* vol. 7, no. 4, 1965.

[247] J. Snyman, N. Stander and W. Roux, "A dynamic penalty function method for the solution of structural optimization problems," *Applied Mathematical Modelling,* vol. 18, no. 8, pp. 453-460, 1994.

[248] C. H. Tsai, C. S. Tang and L. G. Chen, "A flexible, fully hardwired CABAC encoder for UHDTV H.264/AVC high profile video," *IEEE Transactions on Consumer Electronics,* vol. 58, no. 4, pp. 1329-1337, 2012.

[249] H. Malvar, A. Hallapuro, M. Karczewicz and L. Kerofsky, "Low-complexity transform and quantization in H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 13, no. 7, pp. 598-603, 2003.

[250] "ModelSim - Leading Simulation and Debugging," Mentor Graphics, [Online]. Available: http://www.mentor.com/products/fpga/model/. [Accessed 7 October 2015].

[251] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE,* vol. 78, no. 9, pp. 1464-1480, 1990.

[252] C. Bienia, "Benchmarking Modern Multiprocessors," Princeton University, 2011.

[253] J. Fritts, "MediaBench II," [Online]. Available: http://euler.slu.edu/~fritts/mediabench/. [Accessed 6 October 2015].

[254] Z. Wang, W. Liu, J. Xu, B. Li, R. Iyer, R. Illikkal, X. Wu, W. H. Mow and W. Ye, "A Case Study on the Communication and Computation Behaviors of Real Applications in NoC-based MPSoCs," in *Annual Symposium on VLSI*, 2014.

[255] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge and R. B. Brown, "MiBench: A Free, Commercially Representative Embedded Benchmark Suite," in *International Workshop on Workload Characterization (WWC)*, 2001.

[256] "POSIX Threads for Windows – REFERENCE - Pthreads-w32," sourceware.org, [Online]. Available: https://sourceware.org/pthreads-win32/manual/. [Accessed 6 October 2015].

[257] "HEVC reference software," Fraunhofer Institute, [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/. [Accessed 29 08 2013].

[258] "HEVC x265 encoder," Google Code, [Online]. Available: https://code.google.com/p/x265. [Accessed 29 08 2013].

[259] "4×Six-Core AMD Opteron processor," [Online]. Available: http://www.amd.com/en-us/products/server/benchmarks/sap-sd-two-tier-four-socket. [Accessed 08 09 2014].

[260] T. Carlson, W. Heirman and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *SC*, 2011.

[261] H. Esmaeilzadeh, E. Blem, R. Amant, K. Sankaralingam and D. Burger, "Dark silicon and the end of multicore scaling," in *International Symposium on Computer Architecture*, 2011.

[262] ITRS, "International technology roadmap for semiconductors, 2010 update," 2011.

[263] C. Brites, J. Ascenso and F. Pereira, "Improving Transform Domain Wyner-Ziv Video Coding Performance," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2006.

[264] "Taiwan Semiconductor Manufacturing Company Limited," TSMC, [Online]. Available: http://www.tsmc.com/. [Accessed 7 October 2015].

[265] "Design Compiler," Synopsys, [Online]. Available: http://www.synopsys.com/Tools/Implementation/RTLSynthesis/DesignCompiler/. [Accessed 7 October 2015].

[266] M. U. K. Khan, M. Shafique and J. Henkel, "CES Free Software - EnAAM," Chair for Embedded Systems (CES), KIT, [Online]. Available: ces.itec.kit.edu/EnAAM/. [Accessed 5 October 2015].

[267] "Video Library and Tools - NSL," Network Systems Lab, [Online]. Available: https://cs-nsl-wiki.cs.surrey.sfu.ca/wiki/Video_Library_and_Tools. [Accessed 7 October 2015].

[268] D. Shin and S. Gupta, "Approximate logic synthesis for error tolerant applications," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2010.

[269] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy and A. Raghunathan, "Salsa: Systematic logic synthesis of approximate circuits," in *Design Automation Conference (DAC)*, 2012.

[270] S. Venkataramani, K. Roy and A. Raghunathan, "Substitute-and-simplify: A unified design paradigm for approximate and quality configurable circuits," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2013.

[271] A. Ranjan, A. Raha, S. Venkataramani, K. Roy and A. Raghunathan, "ASLAN: synthesis of approximate sequential circuits," in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2014.

[272] L. N. Chakrapani, K. K. Muntimadugu, A. Lingamneni, J. George and K. V. Palem, "Highly energy and performance efficient embedded computing through approximately correct arithmetic: a mathematical foundation and preliminary experimental validation," in *international conference on Compilers, architectures and synthesis for embedded systems*, 2008.

[273] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan and K. Roy, "IMPrecise adders for low-power approximate computing," in *International Symposium on Low Power Electronics and Design (ISLPED)*, 2011.

[274] V. Gupta, D. Mohapatra, A. Raghunathan and K. Roy, "Low-Power Digital Signal Processing Using Approximate Adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD),* vol. 23, no. 1, pp. 124-127, 2012.

[275] A. B. Khang and S. Kang, "Accuracy-Configurable Adder for Approximate Arithmetic Designs," in *Design Automation Conference (DAC)*, 2012.

[276] P. Kulkarni, P. Gupta and M. Ercegovac, "Trading Accuracy for Power with an Under-designed Multiplier Architecture," in *International Conference on VLSI design*, 2011.

[277] S. Mittal and J. S. Vetter, "A Survey of CPU-GPU Heterogeneous Computing Techniques," *ACM Computing Surveys,* vol. 47, no. 4, pp. 69:1-69:35, 2015.

[278] "OpenCL - The open standard for parallel programming of heterogeneous systems," Khronos, [Online]. Available: https://www.khronos.org/opencl/. [Accessed 12 October 2015].

[279] M. Salehi, M. K. Tavana, S. Rehman, M. S. Florian Kriebel, A. Ejlali and J. Henkel, "DRVS: Power-efficient reliability management through Dynamic Redundancy and Voltage Scaling under variations," in *International Symposium on Low Power Electronics and Design (ISLPED)*, 2015.

[280] K. Muller, H. Schwarz, D. Marpe, C. Bartnik, S. Bosse, H. Brust, T. Hinz, H. Lakshman, P. Merkle, F. Rhee, G. Tech, M. Winken and T. Wiegand, "3D High-Efficiency Video Coding for Multi-View Video and Depth Data," *EEE Transactions on Image Processing,* vol. 22, no. 9, pp. 3366-3378, 2013.

[281] A. Vetro, T. Wiegand and G. Sullivan, "Overview of the Stereo and Multiview Video Coding Extensions of the H.264/MPEG-4 AVC Standard," *Proceedings of the IEEE,* vol. 99, no. 4, pp. 626-642, 2011.

[282] H. Schwarz, D. Marpe and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 17, no. 9, pp. 1103-1120, 2007.

[283] V. Goyal, "Multiple description coding: compression meets the network," *IEEE Signal Processing Magazine,* vol. 18, no. 5, pp. 74-93, 2001.

[284] "The WebM Project | VP9 Video Codec Summary," WebM, [Online]. Available: http://www.webmproject.org/vp9/. [Accessed 03 October 2015].

[285] B. M. T. Pourazad, C. Doutre, M. Azimi and P. Nasiopoulos, "HEVC: The New Gold Standard for Video Compression: How Does HEVC Compare with H.264/AVC?," *IEEE Cosnumer Electornics Magazine,* pp. 36-46, 2012.

[286] "Compiler Intrinsics," Microsoft, [Online]. Available: http://msdn.microsoft.com/en-us/library/26td21ds.aspx. [Accessed 03 October 2015].

[287] S. Siewart, "Using Intel® Streaming SIMD extension and Intel® integrated performance primitives to accelerate algorithms," White paper, Intel, 2009.

[288] Altera, "Nios II Process Reference Handbook," [Online]. Available: https://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf. [Accessed 08 06 2015].

[289] "Advanced video coding for generic audiovisual services," ITU-T Rec. H.264 and ISO/IEC 14496-10:2005 (E) (MPEG-4 AVC), 2005.

[290] "IEEE 802.1 AV Bridging Task Group," IEEE, 20 March 2013. [Online]. Available: http://www.ieee802.org/1/pages/avbridges.html. [Accessed 04 October 2015].

[291] M. U. K. Khan, "H.264 VHDL," Chair for Embedded Systems (CES), KIT, June 2015. [Online]. Available: http://ces.itec.kit.edu/h264hdl/. [Accessed 04 October 2015].

[292] "Extended WPF Toolkit™ Community Edition," [Online]. Available: https://wpftoolkit.codeplex.com/. [Accessed 4 October 2015].

[293] "Fluent Ribbon Control Suite," [Online]. Available: https://github.com/fluentribbon/Fluent.Ribbon. [Accessed 4 October 2015].

[294] "OxyPlot," [Online]. Available: https://github.com/oxyplot. [Accessed 4 October 2015].

[295] A. Rodriguez, A. González and M. Malumbres, "Performance evaluation of parallel MPEG-4 video coding algorithms on clusters of workstations," in *International Conference on Parallel Computing in Electrical Engineering*, 2004.

[296] x264, "H.264/MPEG-4 AVC video encoder," [Online]. Available: http://www.videolan.org/developers/x264.html/. [Accessed 15 01 2014].

[297] C. C. Chi, M. Alvarez-Mesa, B. Juurlink, V. George and T. Schierl, "Improving the parallelization efficiency of HEVC decoding," in *International Conference on Image Processing*, 2012.

[298] D. N. Truong, W. H. Cheng, T. Mohsenin and e. al., "A 167-Processor Computational Platform in 65 nm CMOS," *IEEE Journal of Solid-State Circuits,* vol. 44, no. 4, pp. 1-15, 2009.

[299] J. Henkel, H. Khdr, S. Pagani and M. Shafique, "New Trends in Dark Silicon," in *Design Automation Conference (DAC)*, 2015.

[300] Y.-K. Lin, C.-W. Ku, D.-W. Li and T.-S. Chang, "A 140-MHz 94 K Gates HD1080p 30-Frames/s Intra-Only Profile H.264 Encoder," *IEEE Transactions on Circuits and Systems for Video*

Technology, vol. 19, no. 3, pp. 432-436, 2009.