Stochastic Performance Throttling for Multicore Architectures under Spatial and Temporal Dependencies

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der Fakultät für Informatik
der Karlsruher Institut für Technologie (KIT)

genehmigte

Dissertation

von

Nabeel Iqbal

Tag der mündlichen Prüfung: 03, Juli 2013

Referent:	Prof. Dr.-Ing. Jörg Henkel, Karlsruher Institut für Technologie (KIT), Fakultät für Informatik, Lehrstuhl für Eingebettete Systeme (CES)
Korreferent:	Juniorprof. Dr. Jian-Jia Chen, Karlsruher Institut für Technologie (KIT), Fakultät für Informatik, Chair of Micro Hardware Technologies for Automation

Declaration

I hereby declare under penalty of perjury that I have written and presented my own work in this thesis. Other sources of information used in the work have been acknowledged in the bibliography.

Nabeel Iqbal
Meissener Str. 14
76139, Karlsruhe
Germany

[illegible]

N a b e e l I q b a l

Dedication

To my parents, siblings and wife Maimoona Iqbal, their unconditional love and support has made this work possible.

Acknowledgements

I am most grateful to my advisor, Professor Jörg Henkel for providing the guidance, inspiration, motivation and pushing me to make progress without breaking spirit. His multi disciplinary approach and global vision of research problems are the instrumental factors behind this work. This thesis could not have been as productive as it was without his countless insightful comments and suggestions that saved me a lot of effort to complete it. I sincerely thank him for all I have achieved during the course of PhD especially the freedom to follow my own research ideas. Because of it, I owe a substantial gratitude to Prof. Henkel for his technical guidance and patient discussions.

I would like to thank my family for their support, patience, encouragement, and teaching the difference between being smart and wise. I will never be able to match the love and care of my parents, wife and siblings. Pursuing studies in a foreign country has many fold impacts on life and I wish that as a son and a brother I could always have been there when they needed me. I am on the cross roads of life and about to start a new page and will try my level best to return as much as possible to my family.

Achieving the title of Dr. is impossible without the support of colleagues and friends. I specially thank to my colleagues Waheed Ahmed, Mohammad Abdullah Al Faruque, Lars Bauer, Talal Bonny, Martin Buchty, Thomas Ebi, Fazal Hameed, Dominic Hillenbrand, Chih-Ming Hsieh, Sebastian Kobbe, and Renate Murr-Grobe.

In the end, I would like to thank everyone in the Chair for Embedded System (CES) at Karlsruhe Institute of Technology (KIT) for providing the excellent learning and research environment. Finally, this work cannot be completed without the financial support of different funding agencies and their role is pivotal in enabling the research presented in this thesis.

“Every soul shall taste death”

List of Own Publications Included in This Thesis

Conferences (double-blind peer reviewed)

[C.1] N. Iqbal, M.A Siddique, J. Henkel, "SEAL: Soft Error Aware Low Power Scheduling by Monte Carlo State Space Under the Influence of Stochastic Spatial and Temporal Dependencies", 48th ACM/EDA/IEEE Design Automation Conference (DAC´11), San Diego, CA, USA, 2011. (**Received "European Network of Excellence on High Performance and Embedded Architecture and Compilation" (HiPEAC) Paper Award)**

[C.2] N. Iqbal, J. Henkel ,"SETS: Stochastic Execution Time Scheduling for Multicore Systems by Joint State Space and Monte Carlo", IEEE/ACM International Conference on Computer-Aided Design (ICCAD´10), San Jose, CA, USA,2010.
(Best paper nomination at ICCAD 2010 for IEEE/ACM William J. McCalla award)

[C.3] N.Iqbal, M.A Siddique, J.Henkel, "DAGS: Distribution Agnostic Sequential Monte Carlo Scheme for Task Execution Time Estimation", IEEE/ACM 13th Design Automation and Test in Europe Conference (DATE´10), Dresden, Germany, 2010.

[C.4] N.Iqbal, M.A Siddique, J.Henkel, "RMOT: Recursion in Model Order for Task Execution Time Estimation in a Software Pipeline", IEEE/ACM 13th Design Automation and Test in Europe Conference (DATE´10), Dresden, Germany, 2010

[C.5] N.Iqbal, J.Henkel, "Efficient Constant-time Entropy Decoding for H.264", IEEE/ACM Design Automation and Test in Europe Conference (DATE´09), Nice, France, 2009.

Abstract

Power consumption and thermal constraints have lead to the evolvement of multicore architecture, which has changed computing in numerous aspects. Not only the underlying architecture has undergone drastic changes but the associated overlying software has also had to evolve accordingly. The goal of high performance with low power for multicore is non-trivial. The demand for extracting a high performance-to-power ratio leads to new power management strategies. One such mechanism is Dynamic Voltage Frequency Scaling (DVFS): The processor’s clock frequency and supply voltage are altered in tandem by software on the fly. The observer monitors the workload in specified time intervals and forecast the workload conditions. The regulator selects the appropriate performance (voltage and frequency) level taking into account the forecasted workload. Compared to single core systems, the parallelism exposed by multicore architectures pose new challenges as well as open up new design opportunities. The constraints of timeliness of tasks, necessitates efficient DVFS mechanism with an accurate knowledge of the workload characteristics. Most state-of-the-art work in the field of DVFS assumes that the workload characteristics are known. Since the workload is a run-time property and exhibit stochastic characteristics and solution must be found considering the real world scenarios. The mathematical models proposed under this research work obviate the need for an a priori knowledge. Naturally, run-time estimation accompanies an error and hence the primary focus of this work is the DVFS system design for soft real-time systems.

This thesis develops two frameworks for DVFS management (1) Periodic performance throttling and (2) Low power scheduling. In the periodic performance throttling, to capture the source of error, observer is modeled in a joint non-linear state space for simultaneous estimation of the workload and model parameters. The state space model is solved recursively by an Extended Kalman Filter (EKF), a well suited framework for nonlinear and non-stationary conditions. The regulator selects the appropriate performance level according to the forcasted workload and the user selectable QoS. The periodic invocation of the DVFS manager abstracts the execution time of multiple tasks in a single workload interval. This abstraction may hamper the quality of the forecast in a case of fine-grained tasks; as multiple sub-processes are considered as a single process. Moreover, it does not take into account task deadlines, hence not suited for real time systems. The solution of low power scheduling is developed for soft real time systems in which observer is built upon a non-linear, non-Gaussian state-space model to cater the un-modeled parameters and unknown stochastic characteristics. The solution of the state space model is found independent of stochastic characteristics by an online Monte Carlo method. Another aspect of modern multi-core processing platforms is the monotonic decrease in the logic gate size which increases the probability of the occurrence of ‘soft-errors’. The probability of the occurrence of these errors is linked to the processor’s operating voltage. Hence, to keep the system reliability constant the developed low power scheduling incorporates the ways to cater the soft errors.

Zusammenfassung

Energieverbrauch und thermische Randbedingungen haben zur Entwicklung von Multicore-Systemen geführt, was die Art des Rechnerbetriebs in zahlreichen Aspekten verändert hat. Nicht nur die zugrunde liegende Architektur hat drastische Veränderungen erfahren, auch die Software muss sich entsprechend weiterentwickeln. Das Ziel, hohe Rechenleistung bei geringer Energieaufnahme zu erreichen ist nicht trivial und erfordert neue Power-Management-Verfahren wie Dynamic Voltage Frequency Scaling (DVFS). Die Prozessor-Taktfrequenz und Versorgungsspannung werden hierbei von der Software zur Laufzeit angepasst. Hauptbestandteile des DVFS sind ein Beobachter und ein Regler: Der Beobachter überwacht die Auslastung des Systems in bestimmten Zeitabständen und schätzt die zukünftige Auslastung des Systems. Der Regler wählt aufgrund dieser geschätzten Auslastung die entsprechende Kombination aus Spannung und Frequenz um die benötigte Leistung zu erzielen. Im Vergleich zu Ein-Prozessor-Systemen stellt die Parallelität von Multi-Core-Architekturen neue Herausforderungen und eröffnet gleichzeitig neue Möglichkeiten. Von Anwendungen vorgegebene Zeitschranken erfordern effiziente DVFS Verfahren mit einer genauen Kenntnis der Systemauslastung. Die vorherigen Arbeiten auf dem Gebiet des DVFS gehen davon aus, dass die Systemauslastung im Voraus bekannt ist. Da die Systemauslastung jedoch eine Merkmal ist welches sich zur Laufzeit ändert und stochastischen Eigenschaften hat, muss eine Lösung für reale Szenarien gefunden werden. Eine Schätzung der zukünftigen Systemauslastung zur Laufzeit ist jedoch mit einem Fehler verbunden. Der primäre Fokus dieser Arbeit ist das Design eines DVFS Systems für weiche Echtzeitsysteme. Die mathematischen Modelle, welcher in dieser Arbeit vorgeschlagen werden, ermöglichen eine genaue Schätzung der zukünftigen Systemauslastung ohne Vorabwissen zu benötigen.

In dieser Arbeit wird der Beobachter als eine Funktion des "zeitlichen und räumlichen Zusammenhangs" modelliert. Wir schlagen vor, die DVFS Regelung periodisch unter Berücksichtigung der Systemauslastung in dem jeweils vergangenen Intervall anzupassen. Um die Ursache des Fehlers zu erfassen, ist Beobachter in einem nicht-linearen Zustandsraum für die gleichzeitige Abschätzung der Systemauslastung als auch der Modellparameter modelliert. Das Modell an sich wird rekursiv durch einen erweiterten Kalman-Filter (EKF) gelöst. Die periodischen Aufrufe des DVFS Managers abstrahieren die Ausführungszeiten von mehreren Anwendungen in einem einzigen Intervall. Um den unmodellierten Parameteren und den unbekannten stochastischen Eigenschaften gerecht zu werden, basiert der vorgeschlagene Beobachter auf einem nicht-linearen, nichtgaußförmigen Zustandsraum-Modell. Die Lösung des Zustandsraummodells wird unabhängig von seinen stochastischen Eigenschaften durch eine Online-Monte-Carlo-Methode gefunden. Ein weiterer Aspekt moderner Multi-Core Systeme ist die stetige Abnahme der Strukturbreite der Logikgatter, was die Wahrscheinlichkeit des Auftretens von sog. „Soft Errors“ erhöht. Die Wahrscheinlichkeit des Auftretens dieser Fehler ist mit der Betriebsspannung des Prozessors gekoppelt - die DVFS Regelung muss daher in der Lage sein, solche Fehler zu berücksichtigen.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

ACPI	Advance Configuration and Power Interface
ASIC	Application Specific Integrated Circuit
CAVLC	Context Adaptive Variable Length Coding
CPU	Central Processing Unit
DCT	Discrete Cosine Transformation: a computational kernel that is used in H-264 video encoder
DPM	Dynamic Power Management
DVFS	Dynamic Voltage and Frequency Scaling
EDF	Earliest Deadline First
EEPROM	Electrically Erasable Programmable Read Only Memory
EKF	Extended Kalman Filter
FIFO	First In First Out buffer
FPS	Frames Per Second
FSM	Finite State Machine
HFM	High Frequency Mode
IP	Intellectual Property
KF	Kalman Filter
LFM	Low Frequency Mode
LMS	Least Mean Square
MB	Mega Byte
MBs	Macro Blocks
MC	Monte Carlo
MIMO	Multiple Input and Multiple Output
MMSE	Minimum Mean Square Error

MPSoC	Multi Processor System on Chip
NAL	Network Abstraction Layer
OS	Operating System
PDA	Personal Digital Assistant
PDF	Probability Density Function
PID	Proportional Integral Derivative
PM	Power Manager
POD	Partially Observable Domain
QoS	Quality of Service
QP	Quantization Parameter
RAM	Random Access Memory
RDRAM	Rambus Dynamic Random Access Memory
SDLC	Software Development Life Cycle
SEAL	Soft Error Aware Scheduling
SER	Soft Error Rate
SET	Stochastic Execution Time Scheduling
SISO	Single Input and Single Output
SMC	Sequential Monte Carlo
SP	Service Provider
VLC	Variable Length Coding
WCET	Worst Case Execution Time

Chapter 1 Introduction

The significant paradigm shift in computing platforms has ushered in the form of multicore computers which brings forth numerous challenges, opening up new areas of research, or necessitating the revision of old techniques from new perspectives. The problem of optimizing the processor's performance-to-power ratio serves as a good example. A processor's performance is directly linked to its power consumption; higher performance requires more power, and the jobs that need mediocre performance suffice with reduced power. Hence, it is a performance-to-power trade-off and achieving an optimum is a challenge. Particularly in a case of multicore architectures that have multiple processing elements (i.e. cores) executing the parallelized tasks. In this scenario, an optimum may only be achieved by selecting the best suitable performance level that consumes minimum energy to complete a job. Such as, accurate workload forecasting would allow finding the suitable performance level without compromising performance. The multicore/multithreaded systems give rise to additional scenarios such as the interdependence of threads mapped on different cores (Software pipelining). In the scenario of real-time systems where timelines are essential, the advances in multicore system become part of the problem rather than the solution. Futuristic computer architectures and software have made it extremely difficult or impossible to estimate the workload [L2005]. In the realm of multicore architectures, workloads of the current applications exhibit stochastic characteristics that limit the benefits towards conventional techniques [MEP2002]. Despite the enormous design efforts required for runtime power management techniques, it is worthwhile because of the improvements in energy efficiency. Beside power management, ever reducing gate size of logic circuits means it will store less charge that results in more vulnerability to soft errors. The occurrence of soft error is also tied to the operating voltage of the processors. Hence, a power management solution must cater the problems of soft error while employing the low power mechanism. To develop the inspiring novel and practical solution for power management in multicore processors, this chapter discusses the background, fundamental concepts, novel contribution and organization of this thesis.

1.1 Background

The semiconductor industry's perseverance to follow the Moore's law resulted in a perpetual growth in transistor integration capacity, performance and complexity. The transistor integration capacity followed the exponential growth and surpass the Moore's Law [BJS2007, M1965, OH2005] because of decreasing feature size that leads to more than doubling of transistors on the chip with each technology node [SYM$^+$2007, FMJ$^+$2007, GAD$^+$2006]. In addition to integration capacity, ever decreasing feature size yields reduced gate delay and hence faster switching transistors, which makes it possible to operate at higher frequencies. In conjunction to fabrication technology, novel architectures, compiler technology, speculative processing, instruction level parallelism, hyper- and multithreaded processing and multicore systems significantly increased the computation per unit time [SA2005, HP2003, PLM$^+$2012].

The giant leap forward in increased computations per time raised the issues of device dependability and power dissipation [C2006, FMJ$^+$2007, RML$^+$2001]. As power density and hence power dissipation increases exponentially, which makes power dissipation and related issues as primary constraints in computing system design. The persistent increase in the power density over the years enhanced the energy footprint of computing devices, which inflated the environmental and financial impacts. Considering the cloud computing and data centers, the annual expenditures of electricity and cooling has already reached millions of dollars. According to studies, the running cost of large computing facilities has already surpassed the facility instantiation budget [B2005]. The regulatory bodies had started to introduce the regulations to address the ever-increasing share of power budget of computing devices. To address the issue specifications are enforced for computing devices power efficiency requirements [U2006]. It is estimated that by following the specifications multibillion dollar in energy savings can be achieved in a decade.

Beside the environmental impact and its implications, power dissipation is a critical factor for the processor's operations as well as its operating life. The heat dissipation and limitation of thermal management solutions give rise to the problems of temporal and spatial heat gradients on a chip [GSJ$^+$2004]. Temporal heat gradient is the source of thermal shock and accelerate the aging process thus reducing the effective lifetime of the processor. Beside other effects, spatial gradients also increase the rate of the electro migration causing the spatial stress which may result in the permanent break down of a processor. To mitigate the power densities computing systems has already seen the paradigm shift in silicon technology in early 1990s. At that time to eliminate the power dissipation barriers, silicon industry switched from bipolar devices to CMOS devices, hence enabling the order of magnitude improvements in power densities [S2005]. In the absence of any practically viable cooling system and industry proven solution of novel fabrication technology, silicon industry has circumvented the problem by introducing the multicore processors.

The advances in CMOS technology encountered the performance wall enacted by power consumption and thermal management. Increasing the clock speed does not yield the comparable increase in performance because the pragmatic space of computational performance is already exhausted [BJS2007]. Beside the change of design direction to multicore processors, industry still faces the problem of power dissipation and thermal issues. The reduction in energy consumption has become an outcry and the problem is complicated by the fact that traditional power management techniques are not applicable to multicore processors because of new technologies and different design requirements. As these techniques are developed for single core systems and when applied to multicore systems may result in significant overhead, complicated task scheduling, unstable system states, back and forth performance state oscillation etc. [LM2001]. The aforementioned issues become more critical with the ever-increasing number of cores on a single chip. It is predicted that the number of cores on a chip will double with each technology node [S2007] which makes it mandatory to reinvestigate the power management techniques. This pose the challenge of scalability of power management techniques and is a critical non trivial requirement [JP2008]. The power management problem spans the complete spectrum of computing domain, ranging from large data centers to small embedded systems. The power management for networked cluster computing is discussed in [HSK$^+$2006, MRH$^+$2007] while the multiple clock domain processors are addressed in [IM2002, WJM$^+$2005]. Another aspect of modern multicore processing platforms is the high scale of integration employed. The reduced size of the gate (in the processor's logic circuits) means it can store lesser charge, and hence the susceptibility to 'soft errors' is increased. The occurrence of a soft error at run-time requires re-computations to obtain the correct result. Since this error is likely to increase manifold for the next technology nodes [BJS2007], due to an even higher scale of integration, it is imperative to devise techniques to compensate for it. It has to be appreciated that the occurrence of soft error is also linked to the processor's operating voltage i.e. power consumption. Hence, the solutions employed to improve the performance-to-energy tradeoff must incorporate ways to cater for the soft errors as well. These requisites renew the interest in devising improved ways for power management of multicore systems to obtain optimal tradeoff.

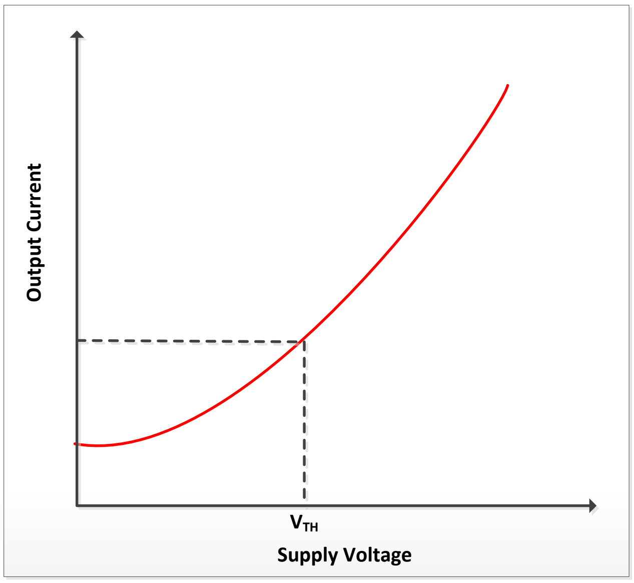


Figure 1.1 Transistor gate voltage-current curve for sub threshold leakage

1.2 Power Management

In the last two decades, significant research efforts and resources are allocated to reduce the power consumption of semiconductor devices covering all levels of design hierarchy. To achieve this goal, research efforts are directed to reduce the power consumption according to type. In general, based on transistor physics and operation power consumption of the digital semiconductor devices is differentiated in two types [HP2003]:

a) Static power consumption
b) Dynamic power consumption

1.2.1 Static Power Consumption

The power consumed due to the leakage of the current in transistor sub-threshold voltage state and gate leakage current is termed as static power consumption. Sub-threshold leakage power represents the power dissipated by a transistor whose gate is intended to be off. Ideally, transistors operate as a switch, but in reality, the relationship between current and voltage is analog. As shown in Figure1.1, V_{TH} is the threshold voltage and ideally current should be zero when the gate voltage falls below V_{TH}. Due to nonlinear relation between voltage and current, there is always a non-zero current even for voltages lower than the V_{TH} . This current is called leakage current and power dissipation due to this current is called sub-threshold leakage power. Hence, threshold leakage current happens due to unwanted charge leakage to ground. While the Gate leakage occurs due to direct tunneling of electrons through the gate insulator (e.g. silicon dioxide). The gate insulator separates the gate terminal from the transistor channel and should ideally become a perfect insulator. In practice, there is no perfect insulator, so there are always some electrons passing through the insulator. With the advances in technology, static power consumption is becoming an influential factor for the total power consumption [KM2008]. It is noteworthy to mention that the power management techniques to reduce static power are generally applied in the design phase [HP2003] and comprises of circuit level transistor reordering and dual-threshold circuits [KGS2004, LWO2004, SS2003].

1.2.2 Dynamic Power Consumption

Dynamic power or switching power refers to the power dissipation attributed to the switching activity of the transistor (discharging of capacitors). In the current designs, it accounts for the major portion of total power dissipation. To quantify dynamic power consumption, the first order approximation is given by the equation (1.1).

$$P_{dyn} = C_L V_{dd}^2 f \tag{1.1}$$

Where P_{dyn} represents the dynamic power dissipation, C_L is the aggregate load capacitance which is dependent upon the drain and gate capacitance of transistors as well as on the wire lengths of on-chip structures. V_{dd} is the operating voltage, the relationship in eq (1.1) shows that voltage has quadratic impact on dynamic power. Where f is the operating frequency and has two fold impacts on power dissipation. Operating frequency not only effects the power dissipation directly, but also indirectly influences due to the required threshold voltages for the given frequency. Generally, operating at higher clock frequency required higher operating voltage. Thus, the combined effect of $(V_{dd}^2 f)$ factor of the dynamic power equation has a cubic impact on power dissipation [HP2003].

To curtail the power dissipation wide array of different power management strategies are proposed which can be categorized as either static or dynamic approaches. Static management approaches (known as offline techniques) are finalized at the design time of the System Development Life Cycle (SDLC) and are primarily based on profile based optimizations or compiler driven management. These approaches are employed at various design stages and abstraction layers. At architectural layer, adaptation and modifications are carried out based on application profiles and main execution kernels [HRT2003, AIC$^+$2002]. Low power favored task partitioning, deadline based scheduling and compiler based power management is carried out at the system and application levels [ACM$^+$2003, DMZ2002, HPH2004, HK2003, KHR2000, LS2000, MSS$^+$2003, SKL2001, TRJ2003, XMM2003, XTS$^+$2008, KBL$^+$2008].

Contrary to the static approaches, the dynamic power management techniques engage in the runtime control mechanisms of the hardware or software and also endeavor to tune the configurable computing resources. Similarly to the static techniques, multitude of dynamic techniques are proposed and thwarts the continuum of computing systems hierarchy. Adaptive body biasing and power gating are well thought-out to be dynamic management at the circuit level [AAE2003, KR2002]. Whole surfeit of architectural level power management techniques are proposed and comprises of pipeline reconfigurations schemes [ABD$^+$2003, BSB$^+$2001, IM2001, PKG2001, SSC2003]. The discussion of adaptive cache scaling with supply voltage tuning and speculative control mechanisms are discussed in [DS2002, FKM$^+$2002, KHM2001, ENP$^+$2003, CCV$^+$2008]. The domain of multiple clocks for multicore processors can be found in [SMB$^+$2002, WJM$^+$2005, JWP$^+$2005, KFJ$^+$2003, LM2006]. Beside circuit and architecture level there exists countless system level dynamic power management techniques addressing system operation and underlying platform. The most effective and well researched technique at the system level is the work load aware Dynamic Voltage and Frequency Scaling (DVFS) or processor performance throttling [M2005, WB2002, HBA2003, CSP2004, RGA$^+$2006, JP2008].

Taking into consideration the design philosophy, static approaches result in to an absolute control methodology because of the wide-ranging view of the entire application and known ‘a priori’. However, static approaches cannot incorporate dynamic application knowledge and do not adapt to changing scenarios. Thus, these approaches only work well when all the design time metrics and assumptions are full filled during the course of operation. Contrary to the application profiling knowledge, dynamic techniques must have to respond to the dynamic execution properties at run time and take the suitable management decisions. The major problem in the design of the run time adaptive technique is the unavailability of ‘a priori’ knowledge. Things become more complex when underlying platform is not known or design should be independent of it. Most of the online techniques assume that the application and its execution time bounds are known quantities whereas; the hardware platform is fixed at design time. Because of the several factors affecting workload, it is rather unfair to make any assumptions on the workload especially when hardware platform cannot be known. The workload is unknown before the task(s) is mapped on the architecture, but even afterwards, the workload remains stochastic due to application-dependent, platform-dependent and environment-dependent factors. The variations in the amount and type of data input to the application, micro-architecture of the processing units (which influences caching, queuing etc.), interaction with the environment, database accesses, communication links etc. introduce uncertainty in the workload. In the realm of uncertainties, dynamically adaptable performance throttling or DVFS is the most practical mechanism to implement the power management scheme [I2007].

1.3 Dynamic Voltage and Frequency Scaling (DVFS)

The major energy-consuming component of computing devices is microprocessor and forms the main chunk in power budget. DVFS scheduling is aggressively employed to diminish the power consumption [KL2003] of processors. In DVFS, the operating frequency and applied voltage of a processor are increased or decreased at run time in tandem to the workload requirements. The increase or decrease of performance level must have to satisfy the stringent condition that there should be no or minimal performance degradation.

Although relationship in equation (1.1) shows that if we can reduce operating voltage by a small factor, power can be reduced by the quadratic factor. But the performance of a system is directly related to the operating voltage. The reduction of the operating voltage results in slowing down the transistors. The reduction of voltage increases the delay of gates, which results in increased access time and latency of the micro-architecture and cause the reduction in operational clock frequency. The reduction of clock frequency increases the execution time of tasks (reduction in performance). So within the given system, reducing the operating voltage offers the potential of a cubic reduction in power dissipation with the performance degradation [DMZ2002].

To elaborate further consider Figure 1.2, assuming that a single task is running on a processor from time (t=0) and to time (t= T). When processor runs at the frequency (f) and voltage (V_{dd}) the task finishes at time interval (t=T/2). Though the processor is active during the time interval (T/2 to T) but not executing any instruction, this interval is known as idle interval. Now consider lowering the clock frequency to (f/2), depicted in Figure 1.2b, in that case, processor's power consumption is reduced to half and task meets its deadline. The reduced operating frequency allows the reduction in applied voltage if we reduce the frequency to (f/2) as well as operating voltage ($V_{dd}/2$) it results in a further reduction of the power consumption, Figure 1.2c. In this scenario task will be able to meet the deadline but energy consumption is one fourth compared to processor running at full frequency. Thus by lowering the operating frequency and applied voltage, the quadratic reduction in the energy consumption can be achieved.

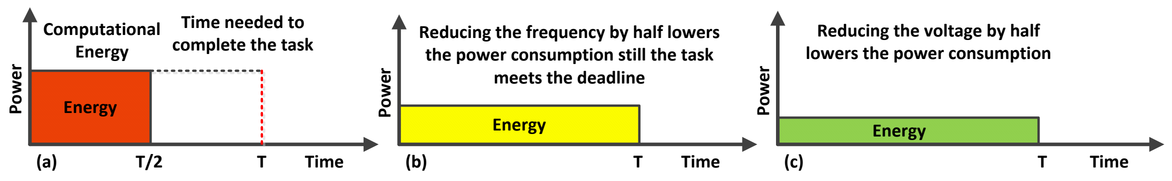

Computational Energy
Time needed to complete the task
Power
Energy
(a)
T/2
T
Time
Reducing the frequency by half lowers the power consumption still the task meets the deadline
Power
Energy
(b)
T
Time
Reducing the voltage by half lowers the power consumption
Power
Energy
(c)
T
Time

Figure 1.2 Power reduction and energy savings by DVFS

It is necessary to mention that the modern day processors support several performance states. An industrial standard for the power management of computational components is an Advanced Configuration and Power Interface (ACPI) [ACPI2010]. According to ACPI, the processor can have 4 different classes of states to save the energy consumption. The description of these states as follows

- C0- Processor is active that can execute instruction without any delay and is known as the DVFS state or P-State. This is the only state in which processor can execute the instructions. In ACPI, this is defined as the main operating state of the processor and usually comprises of several sub operating states. The sub operating states represent the multitude of possible operating frequencies and voltages. The performance throttling or DVFS is only applicable in the operating state of the processor.
- C1-This state represents that the process is in halt state and can start executing instruction instantly.
- C2- In this state execution context is preserved and transition to operating state consume time and energy.
- C3- It refers to the sleep state of the processor and no execution context or cache context is maintained. This state may comprise of multiple sleep sub states.

Different processor states with interstate transitions in ACPI are shown in Figure 1.3, the research presented in this thesis only consider the operating state (C0) of the processor. The DVFS mechanism is not applicable to non operating states, (C1,C2 and C3) and power management of these states is carried out by Dynamic Power management (DPM) scheme [BPB^{+}1999]. There are three key requirements for the implementation of the DVFS on a processor:

- A microprocessor that can operate over a wide voltage range and at least supports two operating sub states. More states are better for improved performance-to-energy efficiency.
- A hardware voltage-regulation-loop that can generate the minimum voltage required for the desired frequency.
- A software controller observes the workload and estimates the future workload conditions. After obtaining the workload estimate, it issues the appropriate frequency and voltage commands.

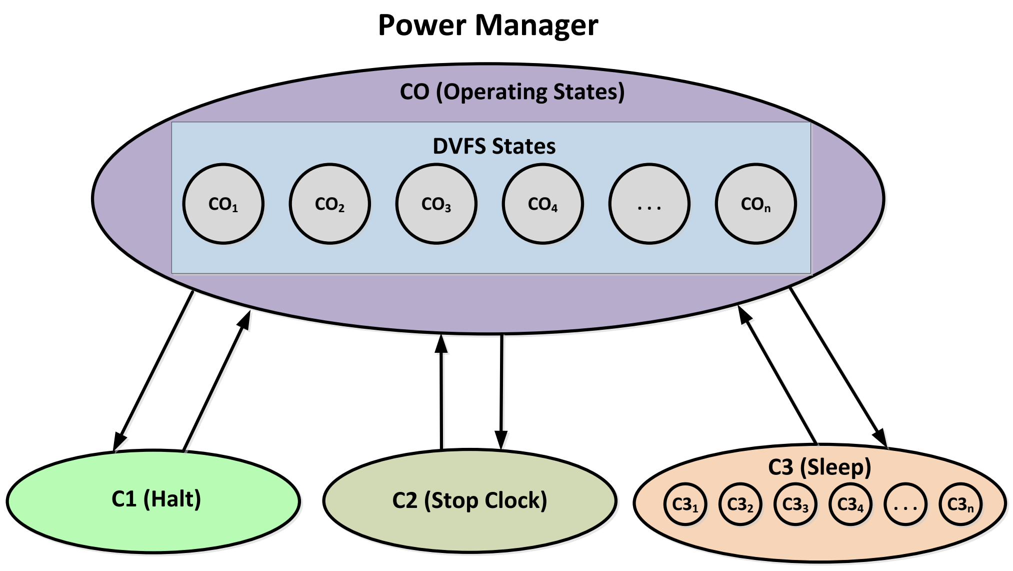


Figure 1.3 ACPI power management states of a processor

1.3.1 DVFS Implementation

Fundamentally, DVFS is termed as a greedy approach that exploits the slack in execution. This slack can appear at different abstraction levels while various DVFS approaches have been used for each level. DFVS schemes operating at the identical level share a similar set of resources, limitation, and available information. In the design hierarchy, there are three abstraction levels where DVFS decisions can be made [KM2008]. (a) At the hardware level, where hardware performance monitors observe the activity of computational unit and adjusts the performance level accordingly. (b) The program level DVFS mechanism exploits the slack due to memory or I/O operations. (c) At the system level DVFS, the idleness of the whole system is monitored whereas, the decision to scale the frequency/voltage is made according to the system workload conditions. The workload estimate always accompany with uncertainty therefore, exact forecast is not possible. Hence, runtime adaptive system level DVFS is not suitable for hard real time systems. The focus of this thesis is soft real time systems and proposed schemes are system level DVFS.

System level DVFS is implemented as a software component. The DFVS manager, which implements the power management scheme, comprises of two components [GCW1995]. (a) Observer and (b) Regulator

a) *Observer*

The design objective of the observer is to observe the dynamics of the workload and intelligently forecast the future workload conditions. In DVFS mechanism observer is the key component because the efficiency of DVFS scheme totally depends on the accuracy of the observer. It must track the workload conditions accurately and obtain the workload estimates with minimal error.

b) *Regulator*

The regulator makes the decision about the assignment of voltage and frequency level to processor by taking into account the forecasted workload and processor’s operating frequency and voltage table. Another goal of the regulator is to sort the voltage frequency command which generate minimal back and forth oscillations. The user selectable Quality of Service (QoS) mechanism may also be the part of it.

As mentioned earlier that the modern processors are designed according to the specification of ACPI and support multiple operating voltages and frequencies in order to deliver desired performance while conserving energy. For example, the Intel® architecture supports 128 voltage levels between low frequency mode (LFM) and high frequency mode (HFM) with extremely small voltage steps. Even for the primitive microcontrollers the designer are striving to provide more and more refined frequency levels to increase the performance-to-energy efficiency. M-core is a typical example of fine grained frequency stepping processors and supports 36 different operating states ranging from 2 to 24 MHz [VGS+2003]. In the scenario of fine-grained frequency steps, the role of DVFS manager becomes daunting. The wrong decision will incur either energy penalty or performance penalty. Figure 1.4 shows both penalties occurring due to wrong frequency step selection. In case of over estimation, there will be energy penalty, and in case of under estimation, there will be a performance penalty. That is why the task of DVFS is not trivial and necessitates finding the suitable power performance tradeoff between opposite extremes. After the availability of essential components for DVFS implementation, the DVFS scheme can be implemented to obtain optimal performance while conserving the energy. Figure1.5 shows the typical implementation details of a system level DVFS. It depicts the power management in a system, which can be classified into two levels; high level and low level. The high level belongs to the user space modules while the low level management comprises of CPU specific drivers. The CPU drivers are provided by the manufacturer and are specific to the particular processor. The processor drivers expose the ACPI interface of the processor to the user space modules and serve as a bridge between DVFS manager and underlying hardware.

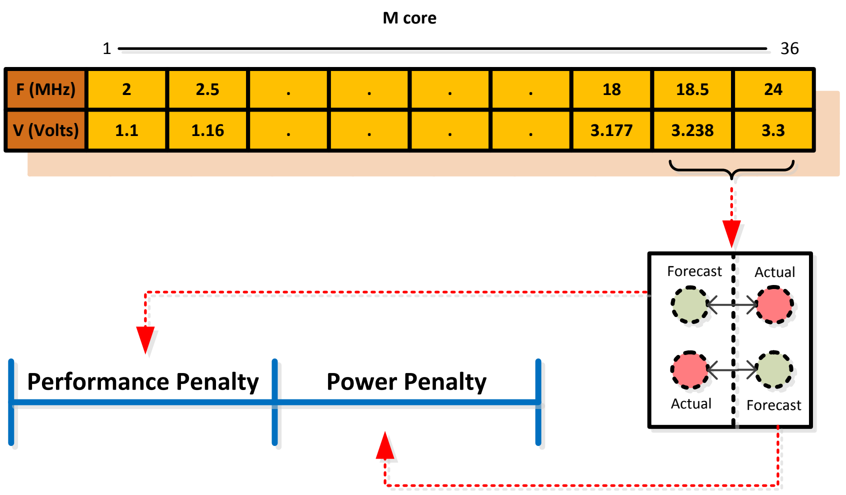


Figure 1.4 Power performance tradeoff

The role of governors at each level is predefined in the reference design, and the reader is referred to [ACPI2010] for in depth technical details. The performance governor monitors the processor for workload and does not consider power consumption and keep the processor at the highest frequency within a user-specified rang. The role of power save governor is to minimize

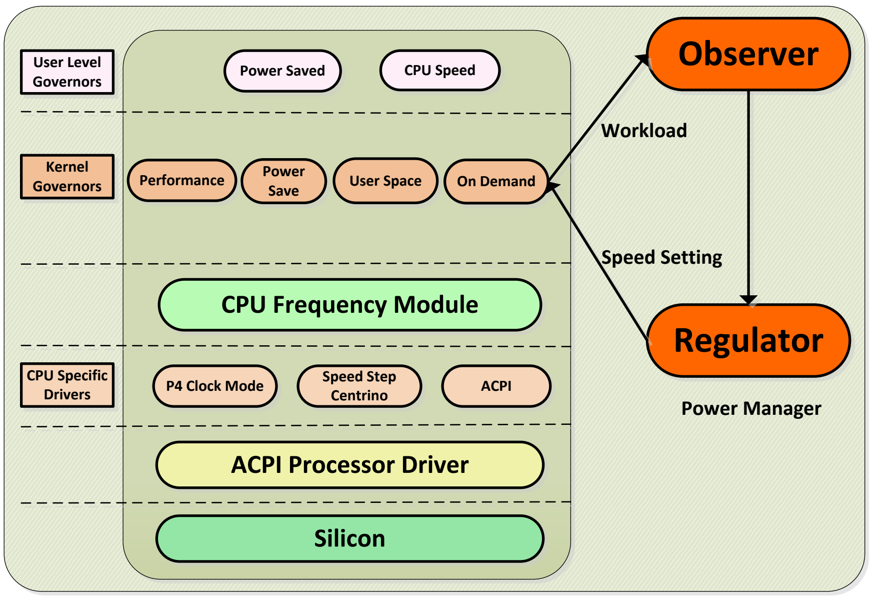


Figure 1.5 Realization of DVFS in a system

the power consumption subjected to workload conditions and govern the processor to lowest possible frequency within the range specified by the user. The task of user space governor is to export the available frequency information to the user level and sanction its control to user. On demand, governor facilitates the user to develop its own power management policy and makes an interface with low-level CPU specific drivers via OS modules. DVFS schemes are implemented by using the interfaces exposed by the on demand governor. As shown in the Figure 1.5, DVFS implementation is managed by power manager, where the Observer collects the workload information from on demand governor and estimates the future workload by using workload information. The estimated workload information is passed to the regulator, which takes the decision about suitable voltage/frequency assignment and passes it back to the on demand governor. Then commands are passed to the OS low-level modules in order to set the desired voltage and frequency level of the processor.

1.4 DVFS in Multicore Processors

The surge in multicore system is driven by the easy availability of inexpensive, high-performance processors while at the same time offering a prospective solution for overcoming the power constraints and the thermal issues. In the DVFS perspective, multicore processors pose new challenges as well open up new design opportunities. In the realm of real-time systems, the timeliness requirement of task execution presents rather a formidable problem for DVFS manager. The most important attribute of real-time systems is that the correctness of such systems depends not only on the computed results but also on time at which the results are produced. These stringent constraints make it essential that the problem of performance throttling must be investigated thoroughly to exploit the new modeling opportunities made available by multicore processors.

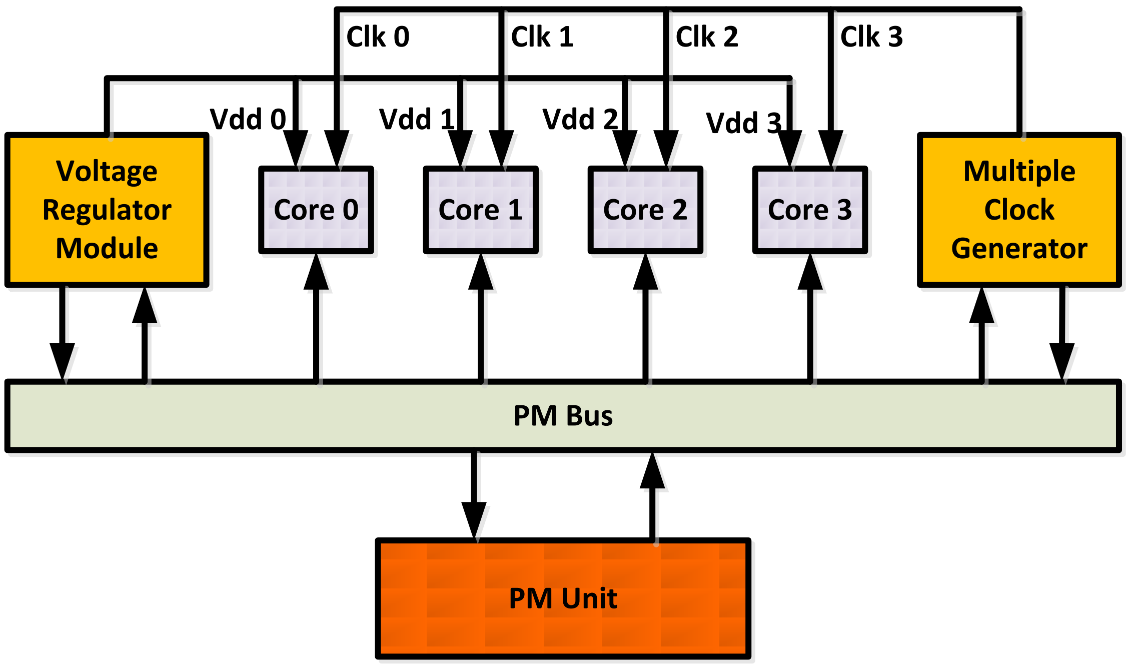


Figure 1.6 DVFS management in multicore processors

The essential components required for DVFS management of multicore processors are similar to single core systems. Figure 1.6 elaborates the interaction of cores on a chip, DVFS manager, multiple clock generator and voltage regulator module. As evident from the name that the multiple clock generators are responsible for the clocks of different frequencies and usually are integrated on a chip. Voltage regulator is the key component and is responsible to deliver the power at fixed or time varying voltage. The voltage regulator can be part of processor chip itself or it may be off chip. The on chip voltage regulators have the advantage of fast voltage switching compared to off chip regulators [KGW$^+$2008].

In multicore systems, DVFS management can be performed at per chip level or per core level. In per chip level management, each core follow the same DVFS schedule and only one voltage regulator is required. The down side of this scheme is the coarse grain control of performance, which leads to inefficient DVFS management. Moreover, performance and energy penalties occur more frequently due to DVFS policy implementation on the average workload conditions of all cores. The per core DVFS management allow the fine grain performance control of each core which allows the individual frequency settings according to the workload conditions. Because of individual treatment of each core, better control is possible and hence it incurs far less performance and energy penalties compared to per chip DVFS management. The down side of per core power management is that it requires individual voltage regulator for each core. The on chip incorporation of voltage regulators for each core comes with the significant cost of area, power and complexity overheads [HM2007, LM2006]. Researcher have strive to figure out the middle way of two extremes of per chip and per core DVFS by introducing the concept of cluster based DVFS management. In such a scheme, the cores are grouped on the basis of similar workload conditions and then DVFS scheme is applied on these clusters. Especially cluster based DVFS management is more attractive in the case of heterogeneous cores [KTR$^+$2003, KTJ$^+$2005, MWK$^+$2006].

1.4.1Temporal and Spatial Dependencies

The foundation of adaptive DVFS management lies on the principle that the future workload conditions can be estimated from the history of workloads. The proposed schemes apply different mechanisms to exploit the information available from workload history to deduce the future workload. The modeling and strength of the scheme defines that how well the information is exploited to obtain the desired results with minimal uncertainty [GCW1995]. Figure 1.7 highlights this principle and is called temporal dependence of workload. As mentioned earlier that the multicore processors give rise to additional modeling opportunities to improve the workload estimation quality. To explore the new modeling opportunities consider the shared resources among different cores such as caches, communication infrastructure etc. These shared resources affect the instruction execution ability of resource-shared cores. Hence, the workload conditions of an individual core also depend on the usage of shared resources. Another aspect of the multicore processor is that only massively parallelized applications can make use of immense computing resource of these systems. The most practically viable solution to exploit the parallel execution is software streaming (software pipeline). Suppose that two tasks (T1 and T2) are running in a software pipeline such that the data output of T1 drives the input of T2. Also, let us assume that T1 and T2 are mapped to different cores, Figure 1.7b. Both tasks are working on the same data set such that the execution time of T1 will influence the execution time of T2. The shared resources and software streaming give rise to the workload dependencies in space. The usage of spatial dependencies in workload modeling can significantly improve the estimate quality and are depicted in Figure 1.7. The modeling of spatial dependencies constitutes the fundamental motivation of this thesis and in depth, analysis and discussion can be found in the motivation section of Chapter 2.

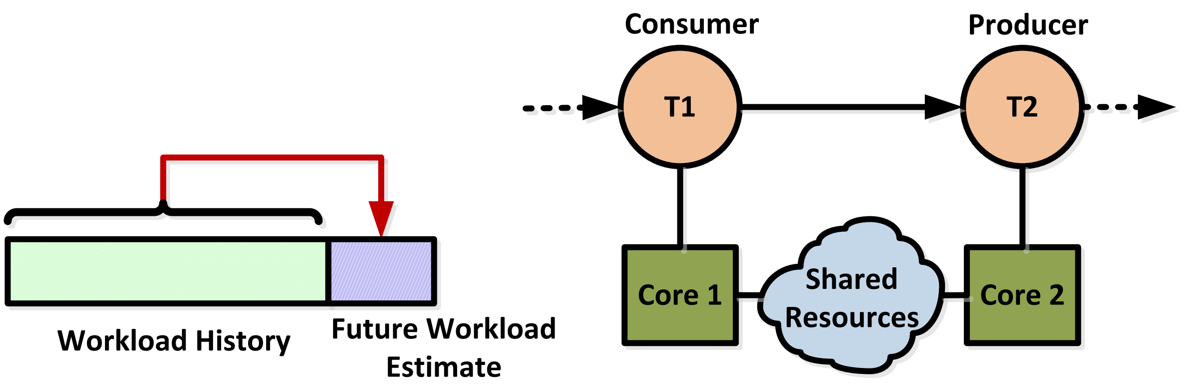


Figure 1.7 Temporal and Spatial workload dependencies

1.5 DVFS and Soft Errors

The computing systems suffer from faults during the operations and these are categorized as permanent and transient. The permanent faults require repair to the device to resume its operation. On the other hand, transient faults do not stop the system operation but eventually lead to incorrect results. In the context of thesis, research can only focus on the transient faults and their relationship with DVFS. The transient faults are predicted to increase many folds with the technology scaling and will pose the greater challenge to the correct operations of the computing systems [IRH1986, HS2000]. The fundamental reason to the increase in transient faults or soft error is the availability of less and less charge on transistor gate as technology scales down. To cater the transient faults applications incorporate the checkpoints and roll back mechanism for error detection and correction. In check pointing and roll back mechanism, a given task is divided into a number of execution segments. A checkpoint joins the two consecutive execution intervals. During the execution of task when a checkpoint is reached, the results and context from processors are gathered for comparison. The difference in results and contexts reveals the occurrence of error and the task is rolled back to the previously known healthy checkpoint for re-computations [RML+2001].

A number of DVFS schemes consider the incorporation of fault tolerance but do not address the negative implication of the DVFS mechanism on the overall system reliability [EMM2002, MME2004]. It is important to consider that the soft error increases with the decrease of voltage and DVFS scheme may reduce the overall system reliability. Lowering the voltage has a twofold impact on system reliability, consider Figure 1.8, the left side shows processor is executing the task at full throttle, and as a result of which it will finish well before its deadline. When a DVFS mechanism is applied to conserve energy, it is required that task must finish just on its deadline. Now consider the right side of the Figure 1.8, which shows that task finishes on deadline but has two note worthy factors. (1) Task has longer execution time which will increase the probability of soft error during the execution (2) Lowering the voltage will increase the soft error rate compared to maximum voltage. Taking into account the futuristic systems it is imperative for DVFS schemes to conserve the overall system reliability.

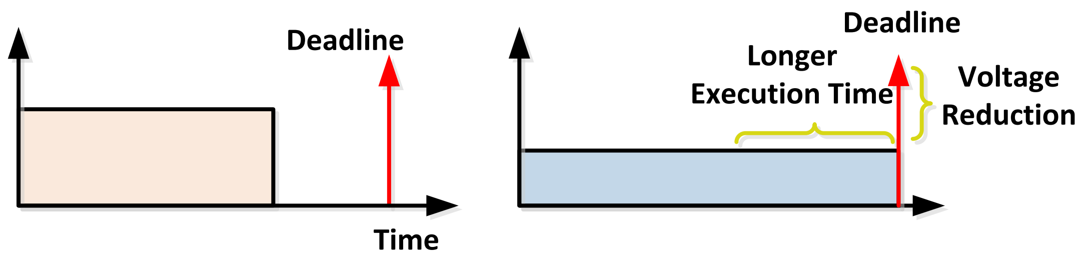


Figure 1.8 Relation of soft error and DVFS

1.6 Thesis Contribution

The endeavor of this thesis is to develop the performance throttling mechanism for the futuristic multicore systems. The goal of achieving high performance with low energy consumption is non-trivial, and this thesis devise the DVFS schemes to extract a high performance-to-power ratio. This thesis raises the concern that the futuristic multicore systems will be nearly impossible to determine the workload conditions at design time. Moreover, it is already established that workload or task execution times exhibit purely stochastic behavior and necessitates that the problem must be tackled in a stochastic framework. Because of this fact previously proposed schemes that rely on the design time or profiling information becomes less effective in a practical scenario. This thesis assume that no a priori information about workload/task execution time is available neither its functional or statistical characteristics are known. The schemes developed in this thesis adapt at run time by the mean of statistical learning which eliminates the need of a priori knowledge. Moreover, contrary to the current state of the art methods, work done in this thesis eliminates the constraints associated with the runtime adaptation methods. This is achieved by developing the joint state space models for simultaneous estimation of workload/task execution time with model parameters. In meticulous, the novel contributions of this thesis are

- In addition to the temporal dependence of workload, this thesis exploits the spatial dependencies of the workload in the modeling process to obtain quality estimates.
- The modeling of the workload as non linear non Gaussian multivariate self organizing state space.
- The estimation of workload in conjunction to model parameters eliminate the constraints associated with the run time adaptation.
- The recursive solution of the state space models are found by the stochastic frame works of EKF and SMC. The recursive solutions eliminate the need to maintain workload history.
- A novel regulator design, which takes the reference processor utilization, compensates for the estimation error and maximizes the performance as well as power savings. Moreover, the selection of reference utilization makes it possible for the user to choose the suitable tradeoff between performance and energy savings

The novel contribution of thesis helps in obtaining the better performance-to-energy efficiency compared to state of the art.

1.7 Thesis Outline

After introduction and background discussion of the topic in Chapter 1 thesis follows the emblematic structure. Chapter 2 discusses in detail the previous literature and highlights the state of the art research work relevant to this thesis. The summary concludes the literature review and identifies the short falls and limitations in state of the art. Chapter 2 is concluded with the point-by-point motivation discussion to establish the reason for this research work. It is noteworthy that multicore architectures give rise to additional modeling opportunities that constitute the core motivation.

Chapter 3 begins with the introduction of the state-space modeling paradigm and then proceed to the modeling of the DVFS observer in state space. The performance-throttling scheme discussed there is a periodically invoked DVFC manager. To estimate the model parameters in conjunction with the workload, the derivation is carried out for multivariate self-organizing state space model by taking into account the temporal and spatial dependencies. The joint estimation eliminates the fundamental constraints associated with the optimal model parameter computations. After the development of workload model, Extended Kalman Filter (EKF) framework is discussed for the recursive solution of the developed model. The chapter concludes with the QoS aware DVFS regulator design.

Chapter 4 discusses two task level schemes, which are modeled in non-linear self-organizing state space. The assertion of the modeling is that the task execution times are purely stochastic, and a priori assumption cannot be made on the execution times or its stochastic properties. The recursive solution of the developed models is found by employing the Sequential Monte Carlo (SMC) framework. As a result, the task execution time estimates are found agnostic to the stochastic or functional properties and the solution is orthogonal and scalable to any number of linear acyclic task graphs. Furthermore, Chapter 4 conclude with the soft error aware low power scheduling scheme appropriate to the application which support check point and roll back mechanism for error detection/correction.

Chapter 5 gathers the performance throttling schemes of previous chapters and extends these to a complete dynamically adaptable power management framework. This chapter detailed the experimental setup and benchmarks and proceeds to the evaluation of the work and comparison with state of the art. Finally, Chapter 5 establishes the effectiveness of research work carried out in this thesis by showing the superiority of developed DVFS schemes.

In due course, Chapter 6 concludes this thesis and gives the synopsis of the research. The shortcomings and weaknesses found in the research work are spelled out in black and white and aid in formulating the future direction of the research.

Chapter 2 State of the Art and Motivation

The potential of energy savings by performance throttling sparks huge research effort and this chapter give the detailed overview of previously proposed techniques. In addition to the general overview significant attention is paid to the current state of the art DVFS techniques for single core as well as multicore processors. The overview is categorised according to the design level, workload characterization principles and techniques employed to adapt the dynamic workload conditions. The last section of related work covers the advances and current state of the art in DVS schemes for multicore systems with the perspective of reliability. The related work and state of the art is concluded with limitations and drawbacks of the existing work which establish the motivation of this work. The motivation section discusses in detail the real world scenarios to highlight the shortfalls in the state of the art. It also emphasizes on the fact that this research work is inevitable for the practically viable solution for the domain of workload adaptive DVFS.

2.1 Related Work

2.1.1 Communication and Memory

The communication devices such as I/O ports, network cards, mass storage devices (BlueRay, DVD, CD, Hard drives) and data input devices (keyboard, mouse and barcode reader etc) are the integral part of computing systems. The operation of these devices is subject to jobs, and multiple power modes are available to save energy. In general, nearly all communication devices support at least two power states, namely "active" and "sleep" states. As the names suggest, in active state device is fully operational and in sleep state device must be awakened first to service the job. The transition of modes from active to sleep and vice versa incur overheads in terms of energy and time consumed in the state transition. The literature proposed multiple energy saving schemes considering the two state operating modes. The energy estimation methods and tool for peripheral devices are discussed in [CRM2004 to CKE2000]. Energy consumption by communication devices in mobile platform running virtual machine is discussed in [FFB$^+$2000]. Modeling of energy consumption by components of the operating system with a focus on communication peripherals is discussed in [LJ2003]. The energy consumption of hard disk by assuming two state power modes is discussed in [RW1994, ZSG+2003]. The rigorous experimentation showed that two state power models may not be able to achieve optimal goal. The fundamental reason is the lack of enough idle time so that the state transitions can be compensated [GSK+2003, PB2004]. To overcome the problem researchers in [GSK+2003] suggested the use of multiple performances level peripheral devices rather than two state model. In this configuration, device is throttled to appropriate performance level to meet the job deadline. One such example is dynamic revolution disks, in which the rotational speed of the disk is varied according to the service requirements [PL2007, H2007].

Similar to the communication devices different types of main memories incorporate service levels to reduce the energy consumption during their operations. A typical Rambus Dynamic Random Access Memory (RDRAM) has multiple active and sleep states. When there is no read or write request, the RDRAM is in attention state and can transit to read or write state with no or minimal latency [R2000]. The power management of main memory poses additional problems of bus-errors because of latency involved in state transition and use of synchronous buses. Therefore, additional overhead must be considered while designing the power management scheme for the main memory [DMV+2001]. The cache decay scheme is proposed in [HKM2002] where a cache line is put in to sleep state after a predefined time out. The experiments show that the cache miss rate, due to sleeping cache lines, is well below the acceptable rate mainly due to the burst access patterns of cache lines.

The performance throttling framework developed in this thesis is directly applicable to the communication and require the development of state space model for a specific device. Like processors, modern communication devices support multiple performance level. The framework of this thesis may select the optimal performance level by accurate forecast of a workload for a specific device.

2.1.2 Instruction and Architectural Level Power Management

The earlier work on the power management involves primitive table based methods targeting inter instruction level energy consumption. These approaches consider simple processors without power gating or other sophisticated power management scheme. In the absence of power management at an architectural level, power consumption largely depends on the applied voltage and operating frequency. The static table based approaches yield reasonable power estimates and prove to be effective in practical scenarios [TMW1994, LEM$^+$2001, CS2001]. The concept of instruction level power modeling is further scaled to estimate the energy consumption of program structures [RJ1998]. Significant research effort is focused on the power management of processors at architectural and function module levels. In such studies, power models are derived by extensive profiling, module complexity and involved switching activity [L1996]. The proposed methods encompass from basic average case estimates to sophisticated lookup table power models incorporating the inter module power modeling [SNT1994, LR1994, LR1995, MOI1996].

The ever-increasing complexity, new fabrication technologies, power management schemes has limited the effectiveness of look up table and profiling based approaches. The drawback lies in to the fact that these are purely design time schemes and cannot adapt to dynamic runtime scenarios. Initial studies showed that different processors events directly reveal the processor power dissipation behavior and are effective to estimate the power dissipation of functional modules [B2000, JM2001, WB2002]. The incorporation of more functionality in performance monitoring units gives refined insight to the power consumption. The fixed static analytical models incorporating the run time information from performance counters are proposed in [BLV$^+$2004, BVL$^+$2005, CM2005, LB2006]. To encounter the limitations of static models adaptive feedback loop based models are becoming attractive [GK2006, ERK$^+$2006, HCG$^+$2006].

2.1.3 Application Level Power Management

Application layer in the design hierarchy is a lucrative option to devise power management scheme because of simplicity and design time decisions. The programmer/designer of the application knows the structure and resource requirement in advance hence strong assumptions can be made. Contrary to the other power management schemes in the design hierarchy, application level schemes are highly custom designed and are specific to the particular application. The availability of idleness is the key to the operation of low power scheme and the inside knowledge of the application plus information of the workload allow the programmer to induce idle phases in the application. In [WBB2002] authors exploit the application calls to I/O library to induce idleness by servicing the low priority calls after all high priority calls are serviced. Pre-fetching of the data and service delay is exploited in [XL2006] at the application level to have frequent idle intervals to improve the energy savings. The programming environment is proposed by [XLL2007] to automate/facilitate the incorporation of these policies in the application. In this approach, a computational offloading is proposed by two competitive policy and statistically optimal time out. To save energy to prolong the operation of battery operated devices an innovative workload reduction is proposed in [FS1999]. In this work, authors have proposed to reduce the video resolution in a video decoding application on a mobile platform. The reduced resolution decoding requires less energy and prolongs the battery life.

Multitude of literature proposes the program phase analysis (periods of high and low work load conditions) as a key to classify the computational requirements to implement the power management scheme. The framework for program phase by the wavelet analysis for multi resolution representation is proposed in [CL2006]. In this work, dynamic representation of multi resolution program phase allows the phase analysis at different scales. The detailed evaluation infrastructure for phase based power and energy optimization is proposed in [HJK2005]. Divide and concur rule are used in [SPC2001] to build the periodic workload characteristics of the program by analysing the basic blocks in the program. The work is extended by automating the process for a large program structures [SPH$^+$2002]. A study in [COJ2001] evaluates the performance requirements in conjunction to the workload conditions. The statistical approach is used in [ESY+2005] to generate the workload compositions and phase periods. Some studies evaluate the dynamic flow of the program and use statistical clustering approaches to detect different phases of workload conditions [LSP+2005, NKH$^+$2006]. The comparison of different program phase analysis techniques is discussed in [DS2003].

Multiple scientists extended the research area by working on dynamically adaptable program phase schemes. A plethora of workload features are employed to achieve the goal of energy savings. Runtime phase shift detection mechanism is proposed to adapt the changing workload conditions and hence to adjust the performance level [HRS2003]. Other used program feature for dynamic workload adaptation are branch counts, number of calls to subroutines, identification of executed program components [DS2002, IM2001, HRT2003, LSC2005, SSC2003]. The fundamental principle of these approaches is the identification of patterns in the execution flow and figure out the best possible mode to operate.

The fundamental to the application level power management is the in depth knowledge of the application and workload it has to process. Application power management is not practically feasible on multicore platforms because of the fundamental constraints of the multitasking environment. The incorporation of power management policies at the application level essentially require interleaving of I/O requests, process utilization and detection of idle periods. In the presence of other processes, required information will incorporate the uncertainties, and it is near to impossible to have the correct information as each process run according to its computational requirements. The problem can be circumvented by commanding and modifying the resource requirements of other applications, but it violates the key principle of process isolation in multi tasking environments.

2.1.4 System Level Power Management

The system level processor throttling started with simple techniques, and most of the research was focused on low power periodic interval algorithms [PBB1998]. The fundamental principle behind these algorithms is to observe the global system state and to set the operating frequency of the processor accordingly, irrespective of the deadlines of individual tasks. The reduced CPU scheduling based on fixed periodic interval observation and speed setting for the next interval is proposed in [WWD+1994], namely the PAST algorithm. The assumption of the algorithm is that the future utilization of the CPU will be similar to the past utilization and set the new operating frequency for the next interval. The major drawback of PAST algorithm is the rapid frequency oscillations that incur energy penalty or performance penalty. The comparison and the summary of the experiments of earlier periodic interval based algorithms is discussed in detail in [GCW1995]. The study in [PBB1998] implemented different periodic interval DVFS {PAST, COPT, AVGN} algorithm on a Personal Digital Assistant (PDA) benchmarks and results showed that the performance of each algorithm depends on the weighting coefficients values. It was observed that the performance for different benchmarks with fixed coefficients was different which means algorithms performance is highly sensitive to the coefficient or model parameters. Taking into account the limitation of earlier work, the heuristic based schemes are proposed which fundamentally compute the weighted averages of past behavior to match the utilization pattern of the processor [GCW1995]. In this, no pattern of the processor utilization is found therefore, it is concluded that simple algorithms based on rational smoothing works better that smart predicting.

The resource management framework architecture for cluster based hosting center is proposed in [CAT$^+$2001]. The study considers the efficient allocation of resources with minimum performance degradation while considering energy as a prime resource. The proposed scheme continuously monitors the performance requirements and tries to work out the best possible resource allocation. The study concludes that over 29% energy savings were achieved for web services workload. ECOSystem [ZEL$^+$2002] consider the energy as a prime resource and strive to allocate all system resource requests directed by the objective of energy minimization. The problem is formulated as a energy accounting model and is carried out for all system resources. The novel concept of DVFS implementation for multimedia applications in client server domain is proposed in [CMB2002]. The authors argued that the multimedia applications exhibit periodic workload conditions and intra period workload varies significantly. It is proposed that the server can communicate to its clients the execution time of the multimedia tasks. This allows the simplicity in DVFS scheme implementation on the client side. In addition, the major computations are carried out at the server end for many thousand clients, which yield additional energy savings. The proposed scheme is of a significant importance to modern mobile computing platforms. An EScheduler energy efficient CPU scheduling algorithm designed specifically for multimedia platforms is proposed in [YN2006]. Considerable energy savings are achieved for various multimedia codecs on Linux based mobile computing platform.

A DVFS scheme for distributed systems is discussed in [LJ2002] and it considers the scheduling of multi-rate periodic as well as aperiodic tasks. The targeted platform is an embedded system and solely focuses on the voltage scheduling of the general-purpose embedded processors. A two phase scheme is presented which comprises of heuristic based static scheduler and dynamic scheduler. The dynamic scheduler adjusts the static schedule at run time according to new scenarios and task dependencies. Application level dynamic energy efficient scheme is proposed in [LSC2005], the authors proposed that while only one application executes on the processor then there is no need to implement power management at OS level. They proposed that all the power management decision must be carried out by the application as it can have more information about its resource requirement and execution times.

Most of the work related to power management of processors is based on thread schedulers. These schemes take into account the task parameters such as arrival time, deadline and execution time. Research related to real time embedded system assumes that task execution times and respective deadlines are available. The required information is extracted through extensive profiling. The popular mechanism to implement the DVFS scheme with known parameters is to employ the weighted averages of the elapsed interval to estimate the dynamics of future interval [PBB2000, SYK2001, LTC2001]. The combination of static real time scheduling and look ahead scheduling is proposed in [PS2001]. The assumption of the study is that the exact task deadlines and execution times are known and allows limited variation in execution time, which is due to resources and communication delays. The experimentation on the desktop system and mobile computers conclude that the energy savings of up to 40% can be achieved by employing the proposed scheme.

Another line of scheduling based power management is to estimate the execution times of the tasks at run time. The researchers argued that multiple factors affect the execution time of the task and hence it cannot be determined at design time or through profiling. The Probability Density Function (PDF) is one of the fundamental characteristics employed by the researcher to obtain the execution time estimates. The task execution time PDF is employed by [XL2006] to estimate the run time execution time considering the task dependencies. An energy efficient heuristic based algorithm for real time scheduling is proposed in [XMM2005]. The proposed algorithm is exact but exponential and determines the upper bound on the energy savings. It is shown that linear time heuristic can approximate these bounds in a very short time. To cater the dynamic workload conditions and execution times, authors in [ZM2004] propose the feedback driven Earliest Dead Line (EDF) DVFS scheme. In this scheme, each task is divided into two parts; the first part represents the average execution time of the task and the second part caters the dynamic variations in the execution time. The second part of the task execution time may be extended to accommodate the Worst Case Execution Time (WCET) of the task.

The requirement of robustness and uncertainty in the workload and execution times give rise to the control theoretic power management schemes. Because of close loop requirements and formal modeling of the problems, these schemes show promising results. A short form of Proportional-Integral-Derivative controller is proposed in [VGJ2003], named as “not quite PID (nqPID)”. It was theories that the proportional component of the controls reacts quickly to the change in workload, the integral part of the controller compute the average utilization and derivative part looks for variations in the workload conditions. Finally, after obtaining the utilization of the processor for the next interval, the appropriate frequency and voltages are selected by the method proposed in [WWD$^+$1994]. A control theoretic scheme for embedded controller is proposed in [XTS$^+$2008]. The authors proposed a feedback control approach for tracking the utilization of system. This approach compares the utilization with the reference utilization. Reference utilization is the function of the deadlines of the tasks; reference utilization may ensure that all deadlines will be respected while achieving the goal of energy savings. The system is modelled analytically and PI controller is applied to control the operating frequency of the processor. The pole placement method is employed to find the coefficients of PI controller.

This algorithm works perfectly when worse case execution time of the tasks is known, but does not guarantee optimal results when there is uncertainty in execution time of the tasks. A stochastic approach based on Markov chains is proposed in [BPB⁺1999] considering the system performance and system history. The authors divide the power management system into two sections, service provider (SP) and Power manager (PM). PM is the controller which observes the history of SP and forms the request queue and then issues the command to switch between system states. The author modelled SP as discrete time controllable Markov chain with finite states, service rate and power consumption is associated with each state and command. Each state is assigned a service rate which is approximately proportional to the average number of requests serviced in unit time. The author optimizes the performance cost metrics subject to the constraint of power. Another stochastic control theoretic approach utilizing Kalman Filter framework is proposed in [BBY⁺2009]. In this study authors argue that workload is a run time property and its characteristics must be adapted at run time. For run time adaptive workload estimation authors propose the feedback enabled mechanism for error minimization. The workload is modelled as a realization of moving average process in a linear state space model. The recursive solution of the model is found by the derived Kalman filter framework. The experiments show significant energy reduction for MPEG-2 video decoder benchmark. However, because of moving average model and linear estimation limits applicability as a general solution.

The power management techniques discussed earlier are proposed for the single core processor. Power management for multicore processors is a highly active area of research. Next section discusses proposed DVFS schemes for multicore processors.

2.1.5 Power Management in Multicore Processors

The diminishing gains in performance pushed forward for the multicore processors. The domain of chip multiprocessors is well established in embedded systems and researchers have put many fold efforts to solve the related problems. Therefore, earlier work in the domain of multiprocessors scheduling revolves around Multiprocessors System on Chip (MPSoC). Because of complexity of the problem researchers have started to explore the stochastic scheduling schemes for general purpose multicore systems as well [MA2010]. The static schedule was generated by offline profiling in [BBT⁺2001] by applying period graph search. The simulated annealing is used to avoid local minima and globally optimal performance solution is found. The optimal number of processing units and respective operating voltage is explored in [RAB⁺2005] for low power operation under the constraints of QoS for a specific class of applications. The experiments are conducted on cycle accurate simulator for streaming applications exhibiting regular workload conditions. The task allocation, scheduling and optimal voltage selection for MPSoC are proposed in [RGA+2006] and special attention is paid to minimize the back and forth frequency switching. The solution comprises of heuristic base optimal voltage search and intended area is the multimedia applications exhibiting periodic workload. A hierarchical decomposition approach considering the power management overhead is proposed in [BBG+2005]. In [ASE+2004, LTK2004], the applied voltages are assumed to be continuous and convex nonlinear programming with polynomial time complexity and mixed integer programming is employed to find the optimal operating voltages. The timeliness of task completion requirement in conjunction to DVFS is discussed in [HM2004]. This research considers the issue of inter task communication and real time scheduling for low energy consumption. Assuming the mapping of the tasks on the processors is known, the authors in [GK2001] present a scheduling technique that maximizes the available slack, which is then used to reduce the energy via voltage scaling. A heuristic based approach to solve simultaneously the problem of task mapping, scheduling and voltage scaling is proposed in [GJ2005]. With technology scaling leakage current is also a dominating factor and leakage aware and adaptive body biasing approach combined with DVFS scheme is proposed in [MFM+2002].

The extensive profiling information is used to find the optimal DVFS configuration for multicore based clusters in [HSK+2006]. The power efficiency of voltage scaling in the domain of multiple clocks driven by different voltages for multicore processors is explored in [IM2002]. The study in [JP2008] proposes the dynamic workload prediction using regression and argues in favour of hardware implementation of power manager. In this study, authors explore the workload characteristics of the parallel applications and adjust the operating system configuration for low energy consumption. The operating system’s observations of running tasks combined with the knowledge of the underlying architecture platform are utilized in [KBL+2008] to reduce overall energy consumption. The technique proposed there basically identifies the best suitable architectural module to execute a particular task, e.g. using the vector processing units for matrix operation. The search for optimal DVFS points for the multicore processors is proposed in [BJ2008] and authors explore the effects of idle core frequency and its effects on the performance and power of the active cores. The authors proposed optimized hardware and operating system configurations to adjust the idle core frequency to minimize its effects on the active cores. The benchmarks used are SPEC2000, SPEC2006 and Sysmark2007 to analyze the runtime power consumption and management on the AMD Quad-Core Opteron™ and Phenom™

processors. Another study [LBK$^+$2007] explore different task scheduling configurations to achieve a low energy consumption. The prime motivation behind this work is the usage of heterogeneous computing platform so that task can be executed efficiently on most suitable processing core, which can yield a considerable energy savings. Hence, the primary goal in this study is to find the best suited processing unit to save energy.

A machine learning based approach to reduce the energy consumption in multicore mobile platforms is proposed in [T2006]. The author strives to develop the scheme based on different user pattern and most commonly used applications to learn the usage behaviour. The exploration framework for power performance analysis for multicore processors is given in [HM2007]. The frame work proposed in this study is configurable and can adapt to different scenarios. An efficient DVFS scheme considering that on chip voltage regulators are available is proposed in [KGW$^+$2008]. The [IBM2005] worked out an optimal DVFS operating point on a multicore processor for a number of active cores running parallel applications. They assume dynamic configurations under limited performance and power constraints and then develop the analytical models for attainable speedups guided by heuristic to reach an optimal operating point. A novel scheme for parallel environments to achieve optimal DVFS management is proposed in [RWB2009]. In this study, authors propose mapping and remapping of thread to balance the workload on cores and then utilize DFVS scheme to obtain the optimal performance levels. It is argued that it is better to run as many cores as possible on low performance level in comparison to some cores running at higher performance levels. The heuristic based approach to achieve the better tradeoff between performance, energy efficiency and fair resource management is discussed in [KSN2007]. A statistical approach to energy efficiency for parallel systems using hardware counters is discussed in [SIK$^+$2007].

The authors in [RLS$^+$2009] explore the performance and power tradeoff for multicore system in the domain of high performance computing. A domain of supercomputing with bounded energy consumption is explored in [RLF2007] while preserving the performance. In this study, the linear programming solution is developed to define the bound for maximum energy savings by an application profiling. The dynamically configurable voltage and frequencies are found in [JWP$^+$2005]. The authors proposed a method to adjust individual core execution times in order to improve performance-to-power trade-offs by balancing inter-thread producer-consumer rates. In [KST2004] authors proposed the technique for trading between power and performance using an integrated Power Management (PM) unit. This PM unit monitors the performance and power of each core and then dynamically adjusts the individual voltages and frequencies. It maximizes the system performance under a given power budget. The multidimensional regression based workload estimation is proposed in [MSB$^+$2008]. The authors first develop the prediction models for a single core and then combined these models to formulate the multi dimensional model. Multiple models proposed in the study show significant energy savings. Another study [MJD$^+$2008] applies the multi dimensional optimization by developing the analytical method to achieve the goal of energy efficiency in multicore processors. The development of analytical approach reduces the likely hood of wrong frequency decision and hence better efficiency is achieved. The schedule of multitasking workload for energy efficiency is discussed in [GKS$^+$2008]. The authors exploit the idea of flexible cores, in which small processing units when combined form the large logical processing entity. The performance level of small processing unit is adjusted according to the workload conditions and optimal operating voltages are applied. The energy aware scheduling for real time multicore systems in multi tasking environment is proposed in [XLL2007]. The probabilistic distributions of the tasks are employed to compute the execution times. A scalable low power thread scheduling algorithm for many core systems is proposed in [WAS2010]. Hierarchical Hungarian Scheduling Algorithm is used for low over head thread scheduling without the loss of accuracy. The study renders linear programming or integer programming techniques infeasible due to their exponential computational cost and propose steepest drop algorithm to achieve performance-to-power efficiency with minimal computational overhead.

The offline methods or assumptions of a priori information has limited applicability in practical scenarios and are confined to small systems. The offline methods cannot adapt to changing workload conditions, whereas the violation of a priori assumption degenerates the efficiency of DVFS schemes. The developed dynamic power management schemes adapt the workload conditions under certain assumption and constraints. The violation of these assumptions and constraints may lead to unstable system state. In a result, there would be more energy and performance penalties due to DVFS manager compared to no power management scheme. It is required that the DVFS manager must adapt the workload conditions at run time without making any assumption about the workload, applications or underlying hardware. This thesis focuses on the development of the general solutions for power management scheme without making any assumption or knowledge of a priori information.

2.1.6 Soft Error Aware Power Management

As mentioned in Chapter 1 that the presence of DVFS scheme has an impact on transient error or soft error rate. The practical soft error rates may increase in the presence of DVFS which in result reduce the system reliability. It is predicted that the soft error will be a major concern to the reliable system operation and will increase manifold [BJS2007, ZC2003, ZMM2004] because of reducing logic size with each technology node. The concern of reliability during the operation of DVFS was mainly revolved around thermal induced stability issues [J2009]. The study in [ZC2008, H2009] considers the probabilistic execution time information of tasks and leave minimum slack to guarantee the required reliability. The solution is found by the heuristic based algorithm. It is reported that achieved energy savings are comparable to the reliability ignorant power management scheme. The proportional feedback controlled based method is proposed in [SGM2008]. The control theoretic scheme aims to reduce the energy consumption while preserving the system reliability. Soft error aware DVFS scheme for application specific processors is discussed in [SAK$^+$2009, SAC2010]. It considers the scaling of voltages while preserving the correctness of the system by employing the linear programming solution. The soft error hardening by selective voltage scaling is proposed in [W2008] and makes use of dual supply voltage. An algorithm is proposed in [SM2010] for multicore system for reliability aware power management that enhances the primary backup model. The better slack management for reliability is carried out by exploiting the uncertainties in the task execution time. This thesis discusses the design of soft error aware low power scheduling and assumes that error detection and correction mechanism of check points in the application is available. The work done in this thesis sought to keep the reliability of the processor constant regardless of changes in operating voltages.

2.2 Motivation

Emergence of multicore systems and persistent prevailing in future computing platforms makes it essential to device the exclusive performance throttling techniques for these systems. The development of new technologies, related to multicore systems, spans from fabrication to programming models opens up new design opportunities. It is essential that new avenues of performance throttling schemes must be explored to make the best use of vast parallelism made available by these systems. The work done in this thesis uses the concepts and tools from the theory of stochastic processes. It is also assumed that the reader has moderate to advanced knowledge of the stochastic system theory. It is not possible to discuss the concepts of probability and stochastic processes here as it is a vast subject in its own and with this limitation the reader is referred to the classical texts covering random variables, probability theory and stochastic process [H1997, IHS2010, P1991, P2001, FF2010, A2006].

2.2.1 Stochastic Workloads

The workload of a core is a function of the execution times of the tasks running on it. It is imperative to dissect the task execution time (the building block of workload) to explore the general characteristics of the workload. In general, a task can be characterized by an integer tuple (S_t, C_t, D_t), where S_t is the start-time of the task, C_t is the execution time and D_t is the deadline. The most non-trivial entity in the context of workload and scheduling is C_t as remaining variables can be known exactly for the given interval. C_t, is the property of the run-time and depends on a large number of factors, ranging from the task program structure to the run-time state of the underlying hardware platform. Most of the mapping and scheduling algorithms assume that task execution times or their bounds are known quantities. Because of the several factors affecting it, it is rather unfair to consider it a known quantity. The execution time is unknown before the task is mapped on the architecture, but even afterwards, the execution time remains stochastic due to application-dependent, platform-dependent, and environment-dependent factors. The variation in

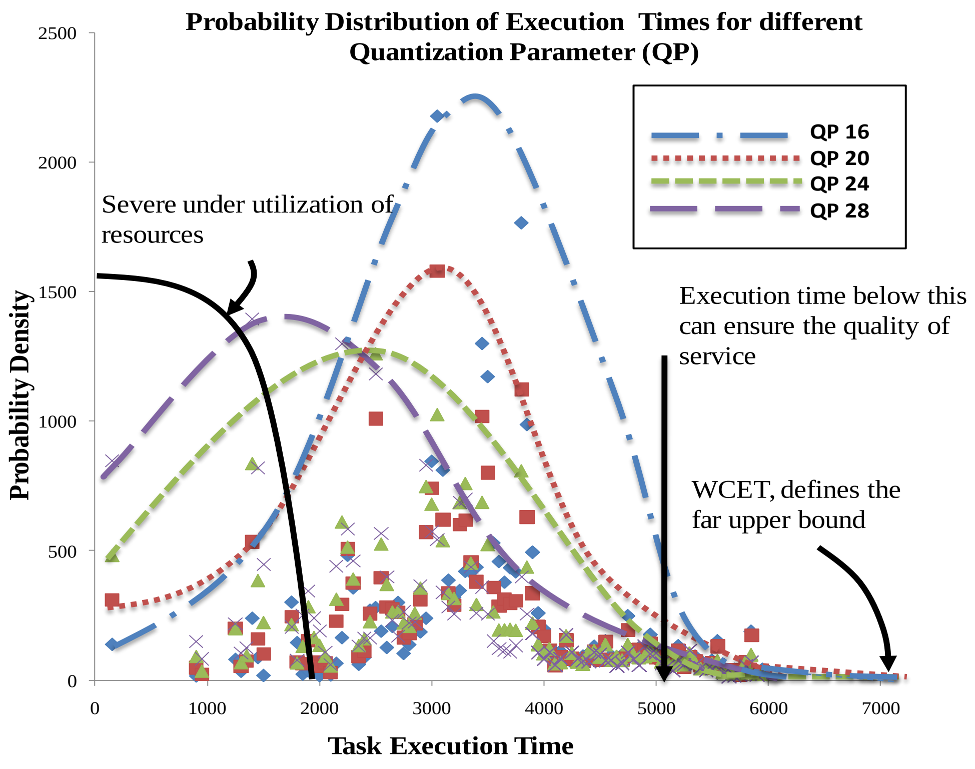


Figure 2.1: WCET and stochastic task execution time

the amount and type of data input to the application, the micro-architecture of the processing units (which influences caching, queuing etc.), interaction with the environment, database accesses, communication links etc. introduce uncertainty in the execution time of the task and hence the workload of the core. Because of the influence of a large number of factors at run-time, it is essential to treat the task execution time or workload as a stochastic entity [MEP2002].

Historically, the Worst Case Execution Time (WCET) estimates are used to exploit the slack for offline DVFS policy designs and are investigated in depth by researchers. The system utilization defined by WCET defines the far upper bound on the processor usage and leads to severe underutilization of the system resources. Moreover, in the realm of multicore, where a large number of external factors influence the execution, guaranteed calculation of WCET may not be viable. Figure 2.1 shows the different probability distribution functions of the task and shows that WCET lies at the far upper bound. An expensive system would be required to meet the WCET imposed deadline because it only considers the worst-case scenario. However, during the lifespan of the system, the occurrence of WCET situation has a small probability. Depending on the required quality of service and nature of application, and if some deadline misses are tolerable, then DVFS scheme based on variable execution time of tasks can be considered to offer a cheaper solution.

As discussed earlier, the stochastic behavior of execution time of a task stems from several factors and is the function of run-time parameters. To make is clear, consider the execution time of a Macro Block (MB, 16x16 pixel block), decoding task in H.264 video decoding process. Figure 2.1 shows the distributions of execution times of MB for 4 different Quantization Parameters (QP). A change in QP causes a change in the amount of data to be decoded. For each case, all sets of conditions were kept same except that the QP was changed at the encoding time. Figure 2.1 makes it clear that MB decoding time is purely stochastic in nature. It is very important to note that MB decoding task has different distributions because of different QP. This makes it esential that no a priori knowledge can be assumed for the workload and its functional or stochastic properties.

2.2.2 Spatial Dependencies

As software pipelining will prevail in the future for the programming of multicore systems, it gives rise to a very important and new design opportunity to improve the workload forecasting. Consider two tasks (T1 and T2) running in a software pipeline such that the data output of T1

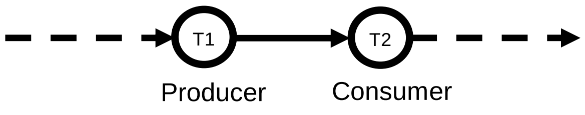


Figure 2.2 : Producer consumer in a software pipeline

drives the input of T2 and both are mapped on different cores, see Figure 2.2. Both tasks are working on the same data set such and it is logical to deduce that the execution time of T1 will influence the execution time of T2.

A theoretical study of such dependencies and derivation of bounds can be found in [DF1996]. The detailed analysis of such dependencies in the scenario of power consumption is outlined in [YDT2008]. This generates the motivation to investigate the workload forecasting/execution time in space (spatial dependence). Figure 2.3 shows two tasks CAVLC (T1) and ISQ (T2) that belong to H.264 video decoding process. CAVLC is the entropy decoder and in software pipeline acts as the predecessor to ISQ (Inverse Scaling and Quantization process). It is observed that the execution time of T2 follows T1. A quantitative term to describe this relation is the cross correlation coefficient of the task execution times. Figure 2.3 shows the coefficient (right axis) drawn over the scale of -1 (negative correlation) to +1 (positive correlation). In case of positive correlation, increase in the execution time of T1 shows the increase in the execution time of T2 and in negative correlation increase in execution time of T1 cause the decrease in the execution time of T2. Both positive and negative correlations are measure of dependence and can

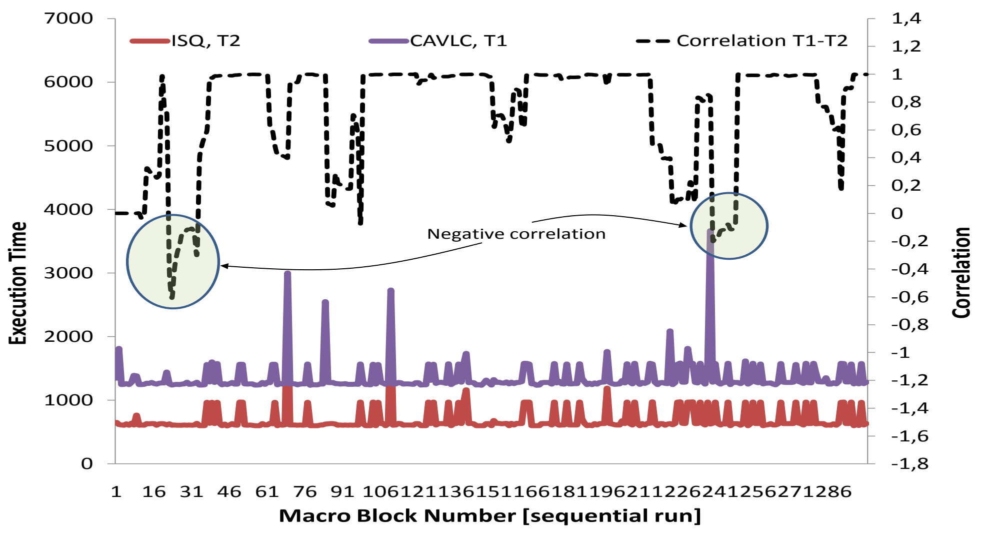


Figure 2.3: Spatial correlation between task execution times

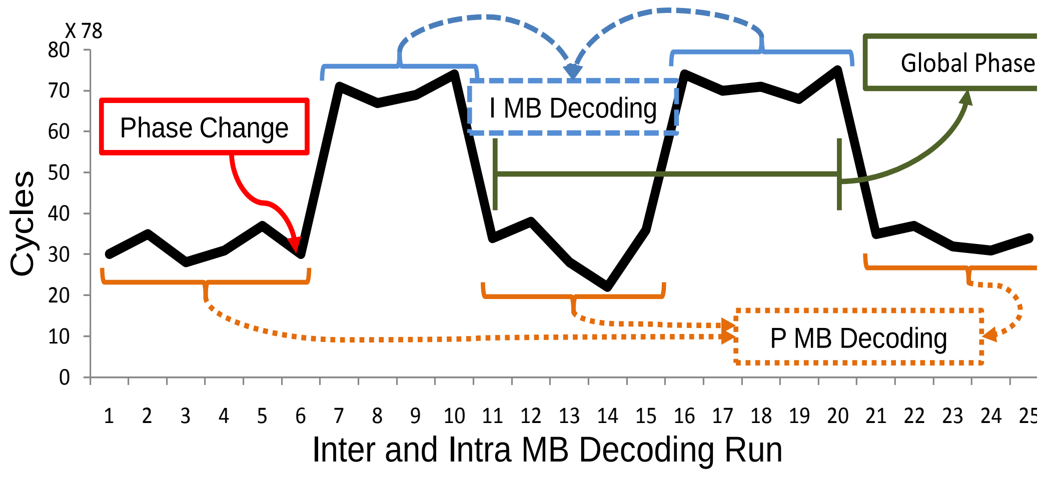


Figure 2.4: Local and Global variations in MB decoding of H.264 video

contribute to improved forecast. When correlation hover around zero then it means there is least dependence between two execution times. The Figure 2.3 clearly shows that most of the time correlation is either positive or negative. This implies that the execution times of the latter and former tasks in the pipeline indeed show strong dependence, which is a time dependent dynamic function. Hence, just like the temporal dependence the extent of spatial dependence cannot be known a priori or at design time and must be adapted/computed at run-time because at some instance in time there could be a strong spatial dependence while at another instance there could be least dependence.

2.2.3 Dynamic Workload Phase

Moreover, the tasks experience phases of high computational loads followed by low computational loads and vice versa. The estimation of execution time suffers from a large error in case of an abrupt phase change. At design time, it is not known when such a phase shift will occur, due to the dynamic nature of the data to be processed. In this context, an accurate forecast of workload/execution time necessitates the modeling of local as well as the global phase. Figure 2.4 attempts to elaborate this concern, for the particular case of H.264 video decoding task for 16 x 16 pixels Macro Blocks (MBs). The Figure 2.4 clearly shows that the execution time of the MBs changes abruptly for some MBs (i.e. high and low computational load for Intra (I) MB and Inter (P) MB respectively).

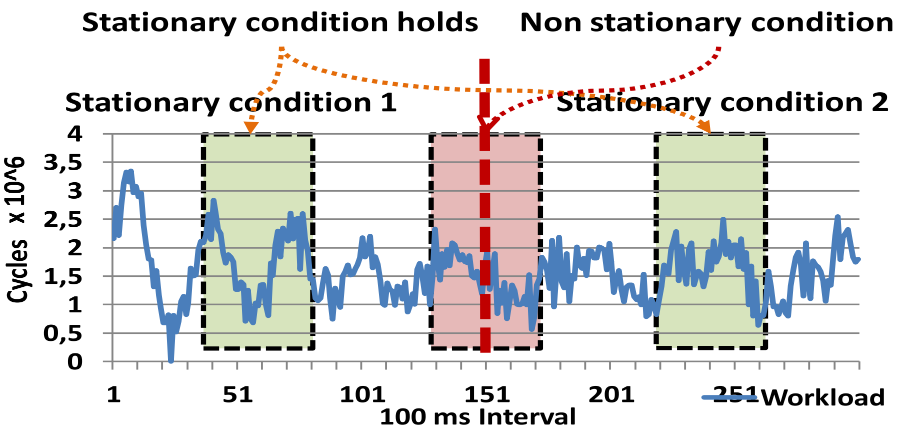


Figure 2.5 : The conventional parameter estimation

2.2.4 Runtime Adaptation

The essential characteristic of the efficient DVFS scheme is the accuracy of future workloads estimates and is vital to its operation. Workload estimation depends on the accuracy of modeling and accurate knowledge of its parameters. These model parameters cannot be known a priori as workload and its characteristics change at runtime. Runtime adaptive DVFS schemes compute model parameters at runtime, by maintaining the history of workloads or sliding window. The sliding window implies that there is a requirement to store values of previous workload intervals to estimate the future workload. At the arrival of new workload information, history window moves one-step forward while keeping the length fix. The computation of model parameters using a sliding window in history imposes the constraint of linearity and stationary conditions within the window [H1996]. Figure 2.5 depicts this limitation of window based techniques by assuming the time varying characteristics of workloads. We can elaborate it as these methods are only valid when history window is in stationary condition1 or stationary condition2, as highlighted in Figure 2.5. If the window is at the boundary of stationary condition1 and stationary condition2 then it violates stationary condition. On the boundary, one side represents the different characteristics of data than the other side. So non-stationary condition is introduced, sliding window based techniques are developed to give optimal results for the stationary data. Hence, for non stationary workload conditions optimal solution for model parameters cannot be obtained.

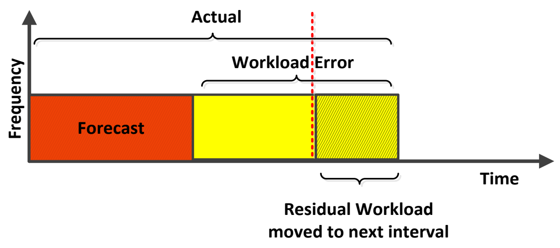


Figure 2.6 : Residual workload due to forecast error

Another non-trivial problem associated is the sufficient length of the window to grasp the underlying dynamics of workload, which are needed to be determined on the fly. The fundamental problem transforms to the determination of suitable window length, but there is no suitable way to find it. Secondly, when the workloads are non stationary, which is the real world scenario; these methods do not provide the optimal solution for the model parameters estimation. During the course of this thesis, work has been carried out to automatically determine the optimal history window length by exploiting the recursion in model order [ISH2010]. It was determined that model order recursion yields long history windows than essential window length found by the hit and trial. Another aspect of the automatic model order recursion is the computational cost, which prohibits it as a run time solution.

The forecast error results in performance or power penalties, elaborated in Chapter 1. The most of the research intended for run time adaptive DVFS manager design consider it as a open loop system which cannot cater intrinsic or extrinsic dynamics of the workload. To achieve the objective of minimum power and performance penalties, the DVFS observer must have the knowledge of forecast error to adapt according to the unknown dynamics. This essentially means that observer must be designed as a feedback loop system, adjusting itself according to error. Moreover, consider the case where forecasted workload happens to be less than the actual workload and will cause the performance penalty. To explore it further, proceed with an example elaborated in Figure 2.6; the estimator forecasted that 100 Kilo cycles will be required to meet the computational requirements and regulator selects the appropriate voltage frequency level. Contrary to the forecast, actual workload was 110 Kilo cycles and at end of the interval 10 Kilo cycles will not be executed. At this point DVFS manager has no information of the remaining workload and cannot cater it. This residual workload must be added in the next interval, which requires close loop simulation [OW1997].

From the above discussion, it is apparent that there is a need to model the workloads of cores as multivariate, mutually interacting and interdependent systems. As the characteristics of workload are unknown, it is required to find the solution independent of history window. It is also necessary to drive a new solution which should be independent of the underlying architecture, workload characteristics and equally applicable to periodic/non periodic, stationary/non stationary workloads.

Chapter 3 Periodic Performance Throttling

As elaborated in Chapter 1, the primary components of the Dynamic Voltage Frequency Scaling (DVFS) are the *Observer* and the *Regulator*. This chapter discusses the design of observer and regulator for periodically invoked DVFS manager. In this scheme observe monitor the workload conditions, irrespective of the running task(s), in a predefined fixed time interval and try to estimate the workload for the future interval. In lieu of the estimated workload, the decision is carried out for the performance level of processor by the regulator. The periodic DVFS manager is applicable only in a case of a non real time system because of the fact that no consideration is paid to the running task(s) and averaging of the workload. This chapter first review the state space modeling, an extensively used framework in feedback control systems, machine learning, financial forecasting and analysis; however, it is a general time-domain method for modeling and analyzing the dynamics of a system. The time-domain is a mathematical domain that expounds the system description and response in terms of time. The generality of the state-space allows its implementation for multivariate, non-linear and time-variant systems. It followed by the development of the state space model for the observer incorporating temporal and spatial dependencies, which constitutes a multivariate, non-stationary process. Moreover, it extends the model to the simultaneous workload forecasting and model parameter estimation at runtime that yields joint state space model. The recursive solution of the nonlinear joint state space model is found by the Extended Kalman Filter (EKF) which eliminates the need to maintain workload history. In the end, the design of regulator assimilating the user defined Quality of Service (QoS) conclude the chapter.

3.1 Introduction to State Space Modeling

The concept of state space can be used to analyze and synthesize dynamic systems in time-domain directly and efficiently. As opposed to the methods that require an accurate computation of the system impulse response to compute the desired system behavior, state space modeling is more generic as it can work with approximate system model and can be extended to nonlinear and time variant systems as well. Systems with multiple inputs and outputs like multivariate problems can also be efficiently represented and modeled in state space. It is not possible to cover the full aspects of the state space modeling paradigm in a limited space and hence basic introduction is covered. The reader is advised to consult the texts [F2005, B2001, CK2007, DK2001, B1990, CK2007, B1994, C2001, C2002, H1989], which spans from classical control theory to financial modeling as well as social phenomena modeling. This shows the wide applications of the state space modeling paradigm in nearly all areas requiring estimation and forecasting.

The state of a system may be defined as “the set of variables which at some initial time t_0, together with the system inputs and the equations describing the dynamics, completely determine the future state and output of the system”. These variables are referred to as the *state variables*. They act like inner system variables that allow analyzing the inner structure of a system. However, for a particular dynamic system, the state variables may not be uniquely defined because it is possible to determine new state variables by carrying out a Bijective state transformation. Such a state transformation sometimes leads to a simplified description of the system. Moreover, the state variables are the smallest number of states that are required to describe the dynamic nature of the system, and it is not mandatory that the states need to be measurable. The state of a system may or may not represent a physical parameter associated with the system. The state of a system is described by the set of n first-order differential equations written in terms of the state variables $x_1, x_2, x_3 \dots x_n$. The manner in which the state variables change as a function of time may be thought of as a trajectory in n-dimensional space. The state variables and n-dimensional space constitute the state space of a system.

Particularly, for the case of linear, time invariant systems, the state space representation of the systems has the following form (as shown in Figure 3.1 as well)

$$\dot{\boldsymbol{x}}(t) = \mathbf{A}\boldsymbol{x}(t) + \mathbf{B}\boldsymbol{z}(t) \qquad (3.1.a)$$
$$\boldsymbol{y}(t) = \mathbf{C}\boldsymbol{x}(t) + \mathbf{D}\boldsymbol{z}(t) \qquad (3.1.b)$$

where, $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{D}$ are the system matrix, input matrix, output matrix and the direct coupling matrix respectively. The (3.1.a) is called the state equation (time update) and describes the system dynamics. The (3.1.b) is called the output equation (measurement update) and describes the relationship between the output and the state variables. The state space model is equally applicable to the modeling of systems with Single Input and Single Output (SISO) and Multiple Input and Multiple Output (MIMO) systems. For n states, p inputs and q outputs to the system, we have the following:

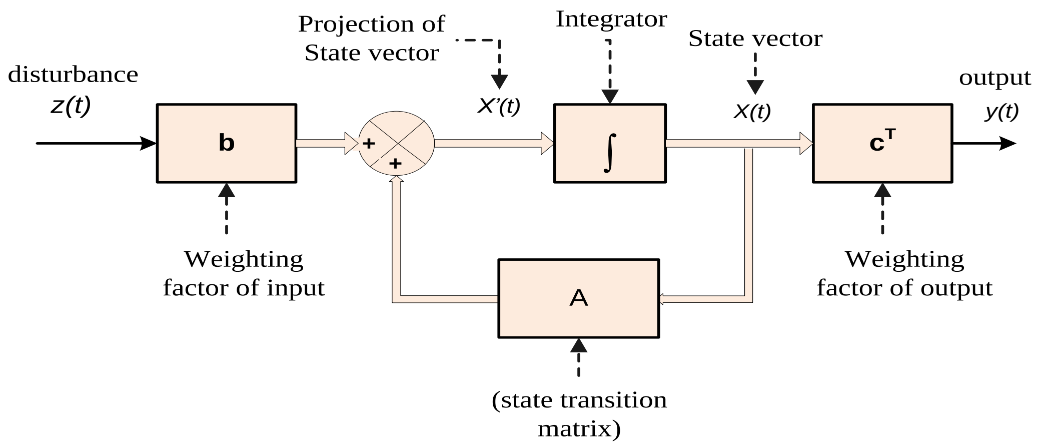

Projection of State vector
Integrator
State vector
disturbance
z(t)
X'(t)
X(t)
output
y(t)
b
+
+
∫
c^T
A
Weighting factor of input
Weighting factor of output
(state transition matrix)

Figure 3.1 : Signal flow diagram for state space representation

state vector: $\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix}$

input vector: $\boldsymbol{z}(t) = \begin{bmatrix} z_1(t) \\ z_2(t) \\ \vdots \\ z_p(t) \end{bmatrix}$ output vector: $\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_q(t) \end{bmatrix}$

system matrix: $\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$ input matrix: $\mathbf{B} = \begin{bmatrix} b_{11} & \cdots & a_{1p} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & a_{np} \end{bmatrix}$

output matrix: $\mathbf{C} = \begin{bmatrix} c_{11} & \cdots & c_{1p} \\ \vdots & \ddots & \vdots \\ c_{q1} & \cdots & c_{qn} \end{bmatrix}$ direct decoupling matrix: $\mathbf{D} = \begin{bmatrix} d_{11} & \cdots & d_{1p} \\ \vdots & \ddots & \vdots \\ d_{q1} & \cdots & d_{qp} \end{bmatrix}$

The matrix **D** describes the direct influence of the input variables on the output variables. **D** is normally zero, which implies that the inputs are not immediately/directly transferred to the outputs. As obvious from the given equations, state space representations is a system of equations, which allows transforming a differential equation of order n into a system of n first order differential equations. The matrix **A** is also referred to as the state transition or system dynamic matrix as it contains information about the Eigen behavior of the system/process i.e. the coefficients of this matrix define the characteristics of the process.

If system to be modeled by state space is time varying process, the condition of time invariance would not hold. In this case, the elements of the $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{D}$ matrices, also called the model parameters, are time-dependent. In a general scenario, where the system cannot be considered linear either as shown in Figure 3.2, we have the following pair of equations:

$$\dot{\boldsymbol{x}}(\mathrm{t}) = f(\boldsymbol{x}(t), \boldsymbol{z}(t), t) \qquad (3.2.a)$$
$$\boldsymbol{y}(\mathrm{t}) = g(\boldsymbol{x}(t), \boldsymbol{z}(t), t) \qquad (3.2.b)$$

The state-equation is now a function of time, which means that the state-space representation varies with time.

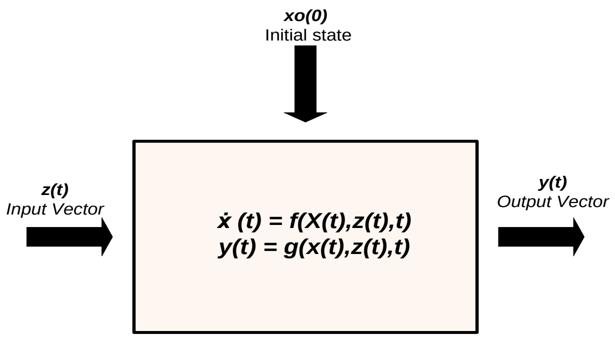


Figure 3.2 Signal flow graph of state representation

3.1.1 Advantages of State Space Modeling

The sate space modelling paradigm has several advantages over conventional methods and some but not the least are as follows:

- The availability of recursive solutions to the state space model makes it suitable for the analysis of the higher order systems by computer aided techniques. The higher order system modelled by classical methods requires large computational resources and renders these infeasible due to multi level integrals.
- More information about the system can be available by modelling the non measured system parameters as unobservable states. The accessibility of unobservable state(s) is of paramount importance as it significantly improves the final state estimates. Because of this fact it is well research area and known as Partially Observable Domains (POD) [BD1996, CKL1994].
- Modelling of time invariant, time variant and non-linear systems can be done within a modelling paradigm.

- The analysis of multivariate processes can be carried out by the same procedures as for SISO system.

Another advantage of the state space modelling is that it has the flexibility to represent the system in number of standard forms. Depending on the problem, one can use any of the following;

- Observable canonical form
- Controllable canonical form
- Diagonal canonical form
- Jordan canonical form

3.2 State Space Modeling for Workload

The literature related to DVFS proposes several schemes for hard and soft real time systems. In real time systems as discussed in detail in Chapter 2., it is assumed that task periods and to some extent Worst Case Execution Times (WCET) are known as a priori. In such a case, the problem boils down to a design of a regulator with limited uncertainty and can be solved at design time. However, in the case of run time adaptive DVFS scheme, only a regulator does not suffice and requires an efficient and accurate observer.

The observer monitors the workload in specified time intervals and forcast the future workload conditions. Therefore, the role of the observer is the ‘prediction’ of the workload in terms of cycles required to complete the job. The underlying assumption of the observer is that the future workloads can be predicted from past workload observations (i.e. temporal dependence of workloads). The regulator selects the appropriate performance level based on the forecasted workload and aims to compensate for the uncertainty of the estimation process. Since the uncertainty in forecast gives rise to power-performance tradeoffs and designing a quality observer is of paramount importance. The reasons discussed in Chapter 2, it is essential to derive a general solution, independent of the underlying architecture, workload characteristics and orthogonally applicable to linear/nonlinear and stationary/non-stationary workload conditions.

In real world scenarios, it is extremely difficult to determine and model all the parameters that influence the workload variations. Hence, only a limited parameter vector can be used to model it. That is why the net effect of temporal and spatial dependencies is important rather than the extent of individual factor influencing the workload. This approach essentially requires the modeling of a workload as a POMD process. To begin the formulation of the model considers that the process innovation and un-modeled factors will cause forecast error. The workload of a core is a stochastic entity and hence forecast error is also stochastic. These stochastic entities can be modeled as a random variable, given as:

$$y = f(z) + \varepsilon$$

where $f(z)$ is computable random variable from model and ε is purely stochastic and would always be present because of the process innovation and un-modeled factors affecting the workload. The factors which affect ε are unknown but show their dependence on the modeled parameters because both modeled and un-modeled parameters belong to the same process [IZP1999]. A state-space model is proposed here as the required general modeling solution to minimize the error. The state space model offers an additional degree of freedom to minimize the forecast error by introducing state variables. In a carefully designed model, state variables may correspond to unobservable factors (un-modeled), which are prime contributors in forecast error. Hence, state variables model the source of error which in result may improve the quality of forecast. The workload is a discrete time entity and in the development of DVFS solution digital computations are carried out therefore, rest of the thesis will stick to the discrete time domain unless continuous time derivations are necessary. The discrete time equivalent of the state and output equations are as following:

$$\mathrm{X}[k+1] = \mathrm{AX}[k] + \mathrm{B}Z[k] \qquad (3.3.a)$$
$$\hat{Y}[k] = \mathrm{CX}[k] \qquad (3.3.b)$$

The components of (3.3) are discrete time equivalent to the components of model of (3.1), refer to [DB2008] for derivation. The notable difference in the notation is of time variable k and absence of direct coupling matrix. In (3.3) it is considered that the inputs are not directly transferred to the output and therefore, D is zero. This essentially means that system is a linear function of control signals. The error between forecast and the actual workload is called innovation (convention of state space modeling), the aim of the estimation process is to minimize the innovation given by

$$\varepsilon[k] = \hat{y}[k] - y[k] \qquad (3.4)$$

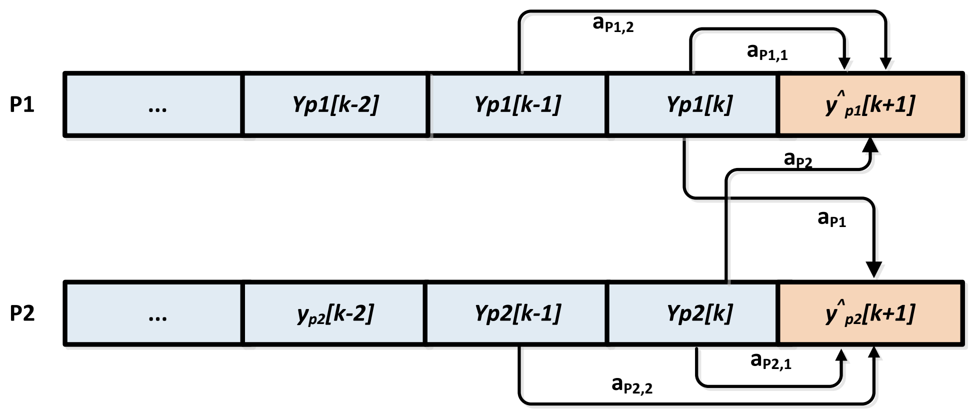


Figure 3.3 : Temporal and Spatial workload dependencies in two core processor

3.3 Modeling the Temporal and Spatial Dependencies

To model the workload for multiple cores in a state space, it is posed as an estimation problem in a multiple time series. For simplicity and ease of derivation, let us first consider two cores, P_1 and P_2, Figure 3.3. The $y_{P_1}[k]$ and $y_{P_1}[k-1]$ are the observed (measured) workloads of the two recently past intervals for the core P_1. On the end of the interval k, it is required to forecast the workload for next interval $(k+1)$ given by $\hat{y}_{P_1}[k+1]$. Now define the $\alpha_{P_{1,1}}$ and $\alpha_{P_{1,2}}$as the temporal dependence coefficients associated with previous two workloads $y_{P_1}[k]$ and $y_{P_1}[k-1]$ respectively. The future workloads has less dependence on the workload distant in history and the last two intervals represent the most relevant process innovation [VGS+2003]. Moreover, the recursive solution learns the systems dynamics on the fly by improving the statistics on the availability of the new workload observation. Hence, workload forecast is the combination of all previous workloads, this fact is proven mathematically in the derivation of the EKF. Now consider the spatial dependence between workloads which stems from shared resources as discussed in motivation section of Chapter 2. Only the current interval of P_2 is relevant for spatial dependence modeling as the impact of the previous intervals of P_2 has already been propagated to constitute the temporal dependence. Let $y_{P_2}[k]$ be the workload of P_2 in the current interval k, and α_{P_2} be the associated spatial dependence coefficient. Combining the temporal and spatial dependence coefficient yields a 2nd order difference equation for the one step forward forecast of the workload, given by

$$\hat{y}_{P_1}[k+1] = \alpha_{P_{1,1}} y_{P_1}[k] + \alpha_{P_{1,2}} y_{P_1}[k-1] + \alpha_{P_2} y_{P_2}[k]$$

Substituting above equation in (3.4) yoelds

$$y_{P_1}[k+1] - (\alpha_{P_{1,1}} y_{P_1}[k] + \alpha_{P_{1,2}} y_{P_1}[k-1] + \alpha_{P_2} y_{P_2}[k]) = \varepsilon_{P_1}[k+1] \qquad (3.5)$$

Similarly, the one step forward forecast equation for P$_2$ core can be given as

$$y_{P_2}[k+1] - (\alpha_{P_{2,1}} y_{P_2}[k] + \alpha_{P_{2,2}} y_{P_2}[k-1] + \alpha_{P_1} y_{P_1}[k]) = \varepsilon_{P_2}[k+1] \qquad (3.6)$$

If temporal and spatial coefficients in (3.5) and (3.6) are known exactly then forecasting is a trivial computation. The thorough discussion in the Chapter 2 highlight that these dependencies are the function of run time and cannot be found at design time. Run time adaptive methods use workload history to compute these dependencies and have short falls, which are also elaborated in the motivation of this thesis. To step forward, to achieve the goal of forecast independent of workload history, lets derive the state space model for (3.5) and (3.6). To proceed forward, consider workload forecasting problem of two cores as a mutually interacting and dependent systems, as evident from the spatial dependencies. The first order decomposition of (3.5) and (3.6) yields the following set of equations,

$$x_1[k+1] = x_2[k] + x_3[k]$$

$$x_2[k] = -\alpha_{P_{1,1}} y_{P_1}[k] - \alpha_{P_{1,2}} y_{P_1}[k-1]$$

$$x_3[k] = -\alpha_{P_2} y_{P_2}[k]$$

$$x_4[k+1] = x_5[k] + x_6[k]$$

$$x_5[k] = -\alpha_{P_{2,1}} y_{P_2}[k] - \alpha_{P_{2,2}} y_{P_2}[k-1]$$

$$x_6[k] = -\alpha_{P_1} y_{P_1}[k]$$

Let $X = [x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6]^T$ forming the state vector from above given first order equations. Then it implies that the state space can be defined from the first order difference equations. The introduction of vector X comprises of observable and non-observable states as evident from the first order difference equations. The ‘unmodeled’ and ‘unobserved’ factors affecting the system are the source of error, but they show statistical dependence on modeled parameters as both belong to the same process. In a multicore system, many factors contribute to the workload, but we are only interested in their net effect and not the individual contributions.

This introduction of state variables gives rise to an additional degree of freedom in the model and hence it significantly reduces the estimation error. From the first order equations and state vector, the transformed state space model is obtained as follows [CK2007],

$$\begin{bmatrix} x_1[k+1] \\ x_2[k+1] \\ x_3[k+1] \\ x_4[k+1] \\ x_5[k+1] \\ x_6[k+1] \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ -\alpha_{P_{1,1}} & -\alpha_{P_{1,2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\alpha_{P_2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & -\alpha_{P_{2,1}} & -\alpha_{P_{2,2}} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\alpha_{P_1} \end{bmatrix} \begin{bmatrix} x_1[k] \\ x_2[k] \\ x_3[k] \\ x_4[k] \\ x_5[k] \\ x_6[k] \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \varepsilon_{P_1}[k] \\ \varepsilon_{P_2}[k] \end{bmatrix} \qquad (3.7.a)$$

$$\hat{Y}[k] = \begin{bmatrix} \hat{y}_{P_1}[k] \\ \hat{y}_{P_2}[k] \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1[k] \\ x_2[k] \\ x_3[k] \\ x_4[k] \\ x_5[k] \\ x_6[k] \end{bmatrix} \qquad (3.7.b)$$

$$Forecast, \qquad x_1[k+1], x_4[k+1] \qquad (3.7.c)$$

While deriving the model of (3.7), only forward paths are considered to derive the inherently stable model. Let's assume that the model parameters,

$$W = \left[\alpha_{P_{1,1}}, \alpha_{P_{1,2}}, \alpha_{P_2}, \alpha_{P_{2,1}}, \alpha_{P_{2,2}}, \alpha_{P_1}\right]^T \qquad (3.8)$$

are known. In that case, on the beginning of each interval, the workloads of P1 and P2 for the recently elapsed interval are measured. Then the error, $\varepsilon_{P_1}[k] = y_{P_1}[k] - \hat{y}_{P_1}[k]$, and $\varepsilon_{P_2}[k] = y_{P_2}[k] - \hat{y}_{P_2}[k]$ are the innovation of the process and for core P_1 and P_2 respectively. At the beginning of each interval the solution of developed state space yields the forecast workloads for the next interval $[k+1]$. Several frameworks are available to solve the state space model and categorized as exact and recursive solutions. Exact solution sought to solve the state space model by the numerical integration and require complete history data from time=0 to time=k [T2001]. These kinds of solution require vast resources in terms of computation and memory and are infeasible in DVFS scheme where computational cost must be negligible for efficiency. The recursive solutions employ stochastic integration, learning the statistics of the process at run time, and hence do not require history. It is worthwhile to mention that all these solutions are the applications of recursive Bayesian estimation theory and hence are stochastic solutions. The most popular frame work to solve linear time invariant state space model is a Kalman Filter and is discussed in the next section.

3.4 Kalman Filter

The Kalman algorithm was proposed by RE Kalman in 1960 [K1960] for the recursive solution of data estimation problems and its concepts are defined using the state space modelling paradigm. Because of its general approach to recursive solution Kalman filter attracted unprecedented number of researchers for further exploration and the reader is referred to [GA2008, CC2008, H2001, ZBS2005, B1998, WB2001] for the in depth treatment of the Kalman filter and its variants. Kalman filter provides recursive solution to estimate the future system states by exploiting the current system states, statistics of past states (learned system statistics) and the currently available system observations. Kalman filter sought to minimize the Minimum Mean Square Error (MMSE) by applying the stochastic difference equations of the state space model in a two step procedure comprises of estimate and correct. Because of recursive scheme the statistics/dynamics of the systems are updated/corrected on the arrival of new data and corresponds to the learning of system statistics. This property makes it suitable for the forecast of online data as a forward n step estimator. Its versatility and general frame work have vast usage in nearly every field of science and technology [GA2008]. The typical configurations of Kalman filter are:

- Estimator:
  The Kalman filter as an estimator try to estimate the system states from noisy or corrupt observations. The estimate of the measurements includes the time at which state is to be estimated i.e. $t_{obs} \le t_{est}$. The most common use of it as an estimator is in state feedback controllers.
- Smoother:
  In a smoother configuration, its role is to extract actual measurements from the noisy/faulty measurements. The batch processing of data is carried out by employing back and forth passes. All measurements are available beyond the estimation interval and can be represented as $t_{obs} > t_{est}$.
- Predictor:
  If Kalman filter is applied as a predictor, measurements are available only before the time of estimation i.e. $t_{obs} < t_{est}$. The recursion cycle of Kalman algorithm forecast the system state on the arrival of new observation.

3.4.1 Kalman Filter Properties

Kalman filter is an application of recursive Bayesian estimation theory so most of its properties are inherent to its theoretical framework. Some genial properties are listed below that make Kalman filter unique compared to non recursive estimation algorithms which makes it suitable for the design of DVFS observer:

- Kalman algorithm gives optimal MMSE estimates when noise or innovation of the process exhibit Gaussian distribution.
- The recursive solution eliminates the need to store previous data and systems dynamics are learned/updated on the arrival of new measurement. This makes it suitable for online data processing.
- The online statistical learning mechanism makes it equally suitable for stationary as well as non-stationary systems.
- Rapid convergence to an optimum solution in case of Gaussian distributions and required limited iteration for convergence in case of changed statistics.
- Orthogonally applicable to scalar and vector data processing; hence SISO and MIMO systems states are estimated in a seamless way by Kalman Filter.

3.4.2 Kalman Filter Algorithm

The primary goal of Kalman filtering is to estimate the system states optimally that belongs to the state space model. The scope of this thesis does not permit the detailed derivation of the Kalman filter and the reader is referred to [H2001, WB2001] for further study. From here we proceed directly to the description of Kalman filter and its parameters with respect to (3.7). Lets assume that the entries of the system matrix, the model parameters, are known and do not change with the passage of time. In workload forecasting problem, model parameters cannot be extracted using physical properties. In that case grey box system identification techniques can be employed to find model parameters at design time. The profiling information can be used to extract the model parameters using system identification methods such as Least Mean Square (LMS), Levinson Durbin recursion, expectation maximization etc. In offline system identification, these parameters are computed once and then used in the operation of the estimator. The offline design cannot cater the run time workload dynamics which require run time adaptation. The run time parameters can be computed by using a system identification process working on limited data history (window). Regardless of which method is applied to obtain model parameters the obtained information completes the model for run time operation. The state space model developed in 3.7(a,b,c) forms the steady state which means that the system matrix, input matrix and observation (output) matrix of the state space model remains constant. It means that only the state variables of the process are changed/updated with each time step. The (3.7) is a linear difference equation and does not account for process noise and measurement noise, considering the presence of these two entities, the parameters of Kalman filter can be given as

$x[k+1]$: A priori or future state estimate (forecast).

Z: The innovation vector

A : The system matrix of the process.

B : The input matrix.

C : The output matrix .

$\hat{y}[k]$: The estimated measurement.

$\hat{P}[k]$: The a priori state error covariance matrix.

$x[k]$: The a posteriori estimate of the state.

$P[k]$: The a posteriori state error covariance.

Q : The process noise covariance and assumed to be constant.

R : Measurement noise covariance, which depends on measuring device and environment.

M : The Kalman gain or blending factor.

The Kalman filter recursion comprises of two steps (1) Correct and (2) Predict [BH1992, MS1979].

Correction: The correction step comprises of three equations (3.9.a,b,c). The first equation computes the Kalman gain using a priori covariance and measurement noise. The Kalman gain acts as a blending factor and minimize the error covariance to obtain the optimal estimate. In second step state estimate is updated after adjusting to error. In this step Kalman gain works as a weighting factor and sort to decreases the residual (innovation). In third step error covariance is (e.g. a posterior error covariance) is computed. Then correction step finishes and passes the values of updated state estimate and error covariance to prediction step.

$$M = \hat{P}[k+1]C^T(C\hat{P}[k+1]C^T + R)^{-1} \tag{3.9.a}$$

$$X[k] = \hat{X}[k+1] + MZ[k] - C^T\hat{X}[k+1] \tag{3.9.b}$$

$$P[k] = (I - MC^T)\hat{P}[k+1] \tag{3.9.c}$$

Prediction: Similar to correction, prediction step also comprises of three equations given in (3.10.a,b,c). At first state prediction is carried out according to new observation. In the second step, observation estimate is computed, which completes the step involving state space model. In third step, a priori error covariance is computed using the values of system matrix A and process noise covariance. Then the values of predicted state variables, predicted data and a priori error covariance are passed to correction step.

$$\hat{\mathrm{X}}[k+1] = \mathrm{AX}[k] + \mathrm{Bz}[k] \tag{3.10.a}$$

$$\hat{y}[k] = \mathrm{CX}[k] \tag{3.10.b}$$

$$\hat{P}[k+1] = AP[k]A^T + Q \tag{3.10.c}$$

Because of two-step recursion it is also known as predictor-corrector framework, Figure 3.4 summarize the recursive cycle and related steps. To predict the workload of a core, Kalman filter is derived as a one-step forward estimator. The state space model developed in Eq 3.7(a,b,c) can be solved by the Kalman filter given in (3.9) and (3.10) which forms the steady state solution. Figure 3.5 shows that Kalman Filter predicts the future state variables at each time step. These

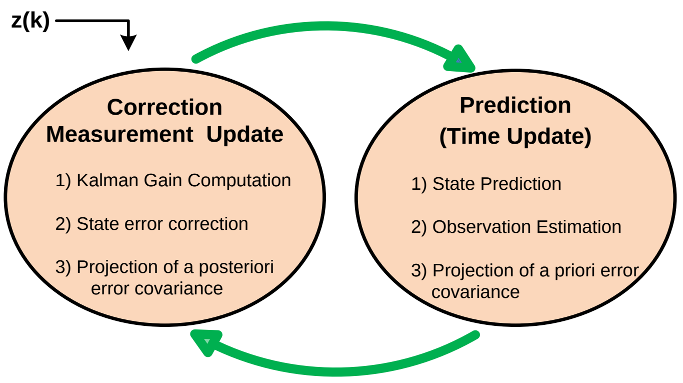

z(k)
Correction
Measurement Update
1) Kalman Gain Computation
2) State error correction
3) Projection of a posteriori error covariance
Prediction
(Time Update)
1) State Prediction
2) Observation Estimation
3) Projection of a priori error covariance

Figure 3.4 : Prediction correction recursion of Kalman filter

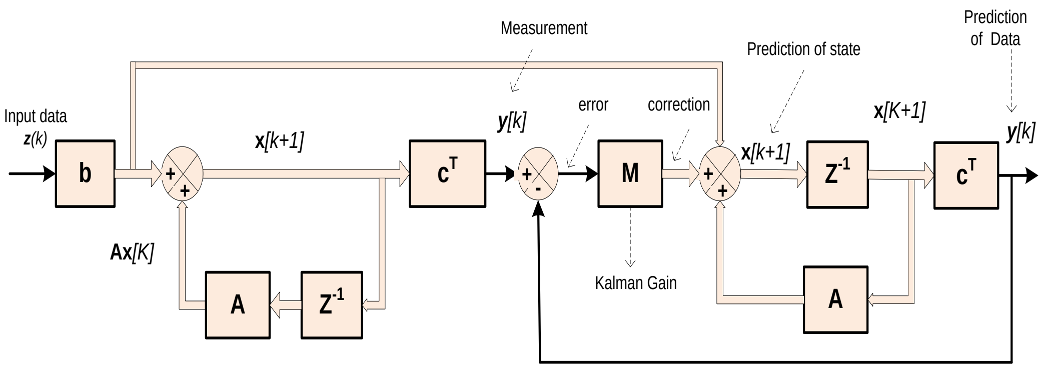


Figure 3.5: Signal flow diagram of Kalman filter as one step forward workload predictor

state variables are used to compute future workload conditions. On the availability of actual workload condition, the error between forecasted and actual workload is computed and then scaled with the Kalman gain to minimise the error in workload forecasts.

The forecast by Kalman filter requires complete information of the model and suffers the limitations raised in motivation section of Chapter 2. To overcome the shortfalls of conventional parameter estimation techniques, next section develops the joint state space model for simultaneous estimation of model parameters and workload.

3.5 Joint State Space Model

In general, for a given state space model, the parameter vector *W*, (3.8), is assumed to be known or can be obtained using the system identification process. In case of time varying dynamics, the model parameters are computed at run time using a sliding window by employing a system identification algorithm. However, the sliding windows tend to be suboptimal and are always valid under certain assumptions, as motivated in Chapter 2. No assumption can be made on workload or its characteristics and hence it is essential to derive the solution independent from the sliding window. To obtain such a solution, the key idea is to estimate the model parameters just like the states of the state space model. The estimation of model parameters in conjunction with the states of the model is achieved by augmenting the state vector with model parameters. The new augmented state vector from (3.7.a) and (3.8) for the joint state space model is defined as

$$X_j[k] = [X[k] \quad W[k]]^T \tag{3.11}$$

The (3.11) define the new state vector in which system parameters are also time varying and it is required that these parameters must also be updated with the beginning of each DVFS interval. The adaptation of model parameters can proceed like the evolution of states with time in the problem space defined by the model. The multiplication of time varying model parameters (system matrix) and state vector makes it non-linear model. As a result, the state space becomes nonlinear and time varying. The time varying state space jointly forecast the workload and updates the model parameters and is called joint state space. The general form of joint state space from (3.3) can be formulated as

$$X_j[k+1] = A_j[W[k], k]X_j[k] + B_jZ[k] \tag{3.12.a}$$

$$\hat{Y}[k] = C_jX_j[k] \tag{3.12.b}$$

The components of (3.12) are exactly similar to the components of (3.3) and subscript j point to the fact that these components forms joint state space model. It is worthwhile to make it clear that even if, system matrix A is a linear function of parameters, nonlinearity is introduced due to the product operation between $W[k]$ and $X[k]$ which makes workload forecasting a nonlinear problem. Kalman filter offers optimal estimate of states in case of linear process but it is not applicable for the state estimation of nonlinear models. It is possible to derive the linearized version of the nonlinear model and recursive estimates can be found by deriving the Extended Kalman Filter (EKF). The goal is to design the near to optimal yet computationally viable DVFS manager, and to make this goal feasible more modeling effort at design time will keep the computational cost negligible. To achieve this goal, it is essential to derive a linear model of a nonlinear state space at design time suited for an estimate by EKF framework. The design time linearization has the following advantages.

- Linearization of the model is guaranteed
- Each new estimate uses the better reference trajectory
- Large initial estimation errors do not propagate

The linearization can be accomplished by the Taylor series expansion of the state space model around a current instant in time [D2010]. To linearize the state space using Taylor series expansion, consider the general time varying (non linear) state space model of (3.12). The state update equation after obtaining the state estimate is given by

$$\hat{X}[k+1] = \mathrm{A}(\mathrm{k}, \mathrm{X}[k]) + \mathrm{B}(\mathrm{k}, \mathrm{Z}[k]) \tag{3.13.a}$$

$$\hat{Y}[k] = \mathrm{C}(\mathrm{k}, x[k+1]) \tag{3.13.b}$$

Consider the current interval k as the steady state point in the nonlinear state space then the deviation of estimated states and innovation can be written as:

$$\Delta X[k] = X[k] - \hat{X}[k] \tag{3.14.a}$$

$$\Delta Z[k] = Z[k] - \hat{Z}[k] \tag{3.14.b}$$

Replacing (3.14.a) and (3.14.b) in (3.13) yields

$$\Delta X[k+1] = A(k, X[k]) + B\big(k, Z(k)\big) - A\big(\hat{X}[k]\big) - B\big(\hat{Z}[k]\big) \tag{3.15}$$

Expansion of (3.15) by Taylor series on steady state point k taking into account nonlinear functions $[A(.), B(.)]$ yields

$$A(k, X[k]) = A\big(k, \hat{X}[k]\big) + \frac{\partial A\big(\hat{X}[k]\big)}{\partial(X[k])} \Delta X[k] + \cdots \tag{3.16.a}$$

$$B(k, X[k]) = B\big(\hat{Z}[k]\big) + \frac{\partial B(\hat{z}[k])}{\partial(Z[k])} \Delta Z[k] + \cdots \tag{3.16.b}$$

In steady state at point k the fist term of Taylor series will be

$$A\big(k, \hat{X}[k]\big) = 0\,,\ \ B\big(\hat{Z}[k]\big) = 0$$

Considering only the first order partial derivatives of Taylor series and truncating the remaining terms yields

$$A\big(\hat{X}[k]\big) \approx \frac{\partial A(\hat{X}[k])}{\partial(X[k])} \Delta X[k] \quad \text{and} \quad B(\hat{z}[k]) \approx \frac{\partial B(\hat{z}[k])}{\partial(Z[k])} \Delta Z[k]$$

Similarly, non linear observation equation can be formulated as

$$C\big(\hat{X}[k]\big) \approx \frac{\partial C\big(\hat{X}[k]\big)}{\partial(X[k])} \Delta X[k]$$

Because of Taylor series expansion terms $A, B,$ and C are Jacobian matrices, and the linearized form of (3.15) can be obtained as

$$\Delta X[k+1] = \mathrm{A}\big(k, \hat{X}[k]\big)\Delta(X[k]) + B\big(\hat{Z}[k]\big)\Delta(Z[k]) \tag{3.17.a}$$

$$\Delta y[k] = C\big(\hat{X}[k]\big)\Delta(X[k]) \tag{3.17.b}$$

It required that the Jacobean matrices must be computed in every iteration of recursive solution. Combining (3.7, 3.11, 3.13) and (3.17) defines the nonlinear joint state space model for the workload forecasting

$$X_j[k+1] = \begin{bmatrix} X[k+1] \\ W[k+1] \end{bmatrix} = \begin{bmatrix} f_1(X[k], W[k], Z[k], k) \\ f_2(W[k], k) \end{bmatrix} \tag{3.18.a}$$

$$Y[k] = [C \quad 0]\begin{bmatrix} X[k] \\ W[k] \end{bmatrix} = C_j \;\; X_j[k] \tag{3.18.b}$$

To linearize the time update equation, linearization of the system matrix and input matrix is required which can be obtained by taking the partial derivatives. The output equation is not directly coupled to inputs and will have null partial derivatives. The following equation shows the system and input matrices and the corresponding partial derivates required to derive the linearized version of both matrices.

$$X_j[k+1] = A_j\big(\hat{X}_j[k]\big)\Delta\big(X_j[k]\big) + B_j\big(\hat{Z}[k]\big)\Delta(Z[k]) \tag{3.19.a}$$

$$\Delta y[k] = \mathrm{C}_j\big(\hat{X}[k]\big)\Delta\big(X_j[k]\big) \tag{3.19.b}$$

where

$$A_j = \begin{bmatrix} {\partial f_1}/{\partial X} & {\partial f_1}/{\partial W} \\ {\partial f_2}/{\partial X} & {\partial f_1}/{\partial W} \end{bmatrix}_{\substack{X=X[k] \\ W=W[k]}} \qquad B_j = \begin{bmatrix} {\partial f_1}/{\partial Z} \\ {\partial f_2}/{\partial X} \end{bmatrix}_{Z=Z[k]} \qquad \mathbf{C}_j{}^T = [C \;\; 0] \tag{3.20}$$

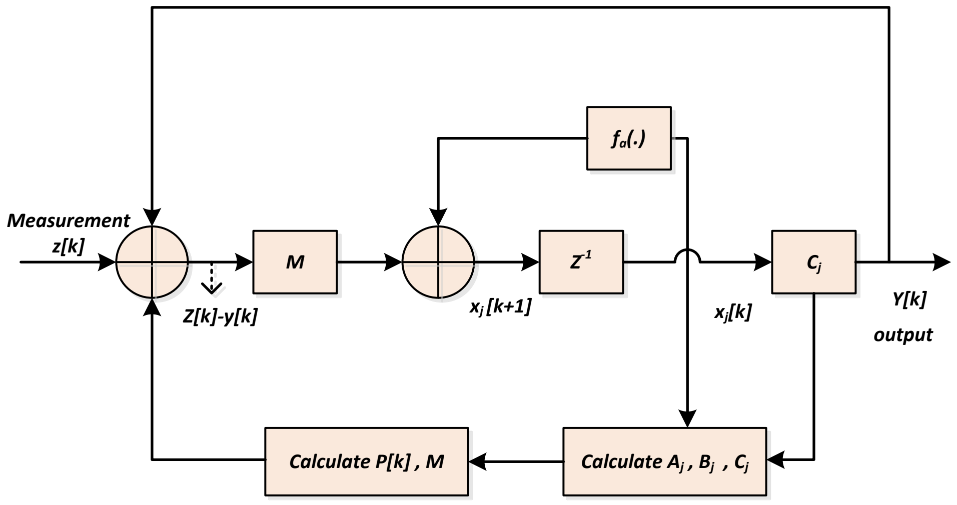


Figure 3.6: Signal flow diagram of Joint State and Parameter Estimation

Figure 3.6 shows the signal flow in joint state space model, and it is essential to note that the Jacobean matrices $\mathbf{A}_j, \mathbf{B}_j, \mathbf{C}_j$ are computed at run time. The components of (3.7) according to (3.20) are

$$f_1(.) = \begin{bmatrix} 0 & x_2[k] & x_3[k] & 0 & 0 & 0 \\ -\alpha_{P_{1,1}}x_1[k] & -\alpha_{P_{1,2}}x_2[k] & 0 & 0 & 0 & 0 \\ 0 & 0 & -\alpha_{P_2}x_3[k] & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_5[k] & x_6[k] \\ 0 & 0 & 0 & -\alpha_{P_{2,1}}x_4[k] & -\alpha_{P_{2,2}}x_5[k] & 0 \\ 0 & 0 & 0 & 0 & 0 & -\alpha_{P_1}x_6[k] \end{bmatrix}$$

$$f_2(.) = \left[\alpha_{P_{1,1}}, \alpha_{P_{1,2}}, \alpha_{P_2}, \alpha_{P_{2,1}}, \alpha_{P_{2,2}}, \alpha_{P_1}\right]^T$$

After solving the partial derivatives of $f_1(.)$ and $f_2(.)$ following are obtained

$$({}^{\partial f_1}/_{\partial X})[k] = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ -\alpha_{P_{1,1}}' & -\alpha_{P_{1,2}}' & 0 & 0 & 0 & 0 \\ 0 & 0 & -\alpha_{P_2}' & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & -\alpha_{P_{2,1}}' & -\alpha_{P_{2,2}}' & 0 \\ 0 & 0 & 0 & 0 & 0 & -\alpha_{P_1}' \end{bmatrix}$$

$$({}^{\partial f_2}/_{\partial W})[k] = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ -x_1' & -x_2' & 0 & 0 & 0 & 0 \\ 0 & 0 & -x_3' & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & -x_4' & -x_5' & 0 \\ 0 & 0 & 0 & 0 & 0 & -x_6' \end{bmatrix}$$

$$\left({}^{\partial f_2}/_{\partial X}\right)[k] = 0_{6,6}\,,\ \left({}^{\partial f_2}/_{\partial W}\right)[k] = I_{6,6}\,,\ \left({}^{\partial f_1}/_{\partial Z}\right)[k] = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}\,,\ \left({}^{\partial f_2}/_{\partial Z}\right)[k] = 0_{6,2}$$

where, **I** and **0** are the identity and null matrices respectively. α_i', x_i' are the partial fractions of the respective variables and are required to be computed in each iteration. Substituting derived components in (3.20) yields the joint state space model for workload forecasting of two core processor and are given as

$$A_j[k] = \left[\begin{array}{cccccccccccc} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ -\alpha_{P_{1,1}}' & -\alpha_{P_{1,2}}' & 0 & 0 & 0 & 0 & -x_1' & -x_2' & 0 & 0 & 0 & 0 \\ 0 & 0 & -\alpha_{P_2}' & 0 & 0 & 0 & 0 & 0 & -x_3' & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & -\alpha_{P_{2,1}}' & -\alpha_{P_{2,2}}' & 0 & 0 & 0 & 0 & -x_4' & -x_5' & 0 \\ 0 & 0 & 0 & 0 & 0 & -\alpha_{P_1}' & 0 & 0 & 0 & 0 & 0 & -x_6' \\ \multicolumn{6}{c}{\mathbf{0}_{6,6}} & \multicolumn{6}{c}{\boldsymbol{I}_{6,6}} \end{array}\right] \qquad (3.21.a)$$

$$B_j[k] = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & \mathbf{0}_{2,6} \\ 0 & 0 & 1 & 0 & 1 & 0 & \end{bmatrix}^T \ ,\ C_j = \begin{bmatrix} 1 & 0 & 0 & 0 & \mathbf{0}_{2,8} \\ 0 & 0 & 0 & 1 & \end{bmatrix} \qquad (3.21.b)$$

The above matrices/equations define the time varying joint state space. Similarly, the joint state space for any number of cores can be derived. The regularity in matrices enable the principle of mathematical induction and the joint state space model for *n* cores can be obtained without derivation from scratch. The procedure given in Algorithm 3.1 can be employed to obtain the joint state space model for n cores. The algorithmic derivation is carried out using the symbolic math representation.

Algorithm 3.1: Joint state space derivation

```
$Event : Derive\ joint\ state\ space\ model\ for\ n\ number\ of\ cores$
$Definitions :$
$A : System\ matrix, B : Input\ matrix, C : Output\ matrix, X : State\ vector, Z : Input\ vector$
$A_j, B_j, C_j, X_j and\ Z_j\ are\ the\ joint\ state\ space\ counterparts\ of\ A, B, C, X\ and\ Z\ respectively$
$Procedure :$
1 :  $Label\ the\ cores\ as\ p_i, i = \{1, 2, \dots n\}, no\ prder\ prefernce\ is\ required$
2 :  $A = null_{(n+1)n,(n+1)2n}, B = null_{(n+1)n,n}, C = null_{n,(n+1)n}$
3 :  $Z[k] = [\varepsilon_{P_1}[k], \varepsilon_{P_2}[k], \dots . \varepsilon_{P_n}[k]]^T, X_j = [x_1, \quad x_2, \dots x_{(n+1)2n}]^T$
4 :  $i = 0;\ j = 0;\ k = 0;\ l = 1;\ m = 1;$
5 :  $for\ \big(i = 1; i \le ((n+1)*n); i = (i+n+1)\big) //Derive\ matrix\ A$
6 :      $for\ (j = 1; j \le (i+n+1); i++) //\ rows$
7 :          $if\ (j\ ==\ i+1)$
8 :              $A(j,i) = a'_{l,1};$
9 :              $A(j,i+1) = a'_{l,2};$
10 :             $A(j, i+(n+1)*n) = x'_l$
11 :             $A(j, i+(n+1)*n+1) = x'_{l+1}$
12 :         $end$
13 :         $for(k = i; k \le i+n; k++) //columns$
14 :             $if\ (i\ ==\ j)\ \&\&\ (k\ != i)$
15 :                 $A(j,k) = 1;$
16 :             $end$
17 :             $if\ (k\ ==\ j)\ \&\&\ (j \ge i+2)$
18 :                 $A(j,k) = a'_m;$
19 :                 $A(j, k + (n+1)*n) = x'_m;$
20 :             $end$
21 :         $end$
22 :         $m = m + 1;$
23 :     $end$
24 :     $l = l + 1;$
25 :  $end$
26 :  $j = 1;$
27 :  $for\ (i = 1 : i \le n; i++) //Derive\ C\ matrix$
28 :      $C(i,j) = 1;$
29 :      $j = j + (n+1);$
30 :  $end$
31 :  $l = 1;$
32 :  $for\ (i = 1; i \le ((n+1)*n; i = (i+n+1))$
33 :      $flag = 1;$
34 :      $for(j = i : i \le (i+n); i++)$
35 :          $if(j\ != i)$
36 :             $if(flag)$
37 :                 $B(j,l) = 1; flag = 0;\ else$
38 :                 $for\ (k = 1; k \le n; k++)\ B(j,k) = 1;$
39 :  $end\ \ end\ \ end\ end$
40 :  $l = l + 1; end$
41 :  $A_j = \begin{bmatrix} & A \\ null_{(n+1)n,(n+1)n} & iden_{(n+1)n,(n+1)n} \end{bmatrix}, B_j = [B \quad null_{n,(n+1)n}]^T,$
42 :  $C_j = [C \quad null_{n,(n+1)n}]$
```

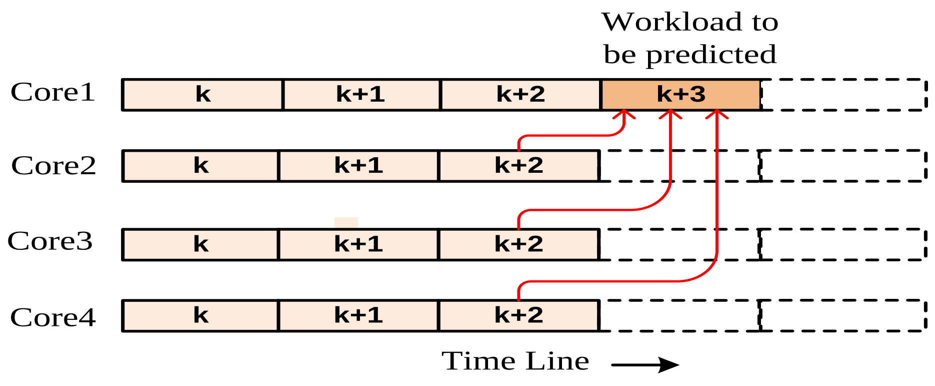


Figure 3.7: Temporal and spatial dependencies in the 4 core processor

Figure 3.7 depicts the workload forecast for quad core processor, an experimental platform of this thesis. For simplicity, Figure 3.7 shows only the spatial dependencies for one core, though all four cores have spatial dependencies with each other. At the end of the DVFS interval the workloads observations are available and considering the spatial and temporal dependencies, it forecasts the workload for next interval. It is worthwhile to mention that resultant matrices of the model are sparse and regular. The full form is shown for compliance with theory and ease of understanding. The sparse property and regularity of the model significantly reduce the computation cost at run time. Consider the measurement update equation, at first it appears that it is the multiplication of two vectors but actually there is no multiplication involved. Similar patterns can also be observed in the time update equation e.g. the multiplication of observations and input matrix.

3.6 Extended Kalman Filter Derivation for Workload

A Kalman filter that linearize about the current mean and covariance (the selected steady state point) is referred to as an Extended Kalman filter (EKF). To proceed for the derivation of EKF for workload forecasting, consider two zero mean random variables y to denote workload and x as an estimate of workload. The optimal least mean square estimate of x given y is $\hat{x} = Fy$ [H2001]. Where $F$is a coefficient matrix that can be found by solution of the normal equation, considering the R_y as covariance of y and R_{xy} the cross covariance between x and y. Assuming that R_y is positive definite then

$$F = R_{xy}R_y^{-1} \tag{3.23}$$

The workload information is becomes available after each DVFS interval, in that case y will be a vector and given by $= [y_0\ y_1 \ldots .\ y_k]^T$, where each y_i can itself be a vector which is the case of multicore workload forecasting. For ease of derivation, let's assume that y_i is a scalar and assume that Y can be replaced by another vector ε of same dimension for some lower triangular and invertible matrix A such that

$$\mathcal{E} = AY \tag{3.24}$$

Assume further that the transformation A can be chosen in such a way that the entries of $\mathcal{E} = [\varepsilon_0, \varepsilon_2 \ldots\ \varepsilon_k]^T$are uncorrelated with each other. In that case the covariance matrix of $\mathcal{E}$ will be block diagonal, given by

$$R_{\mathcal{E}} = diag\left[R_{\varepsilon,0}\ R_{\varepsilon,1} \ldots\ R_{\varepsilon,k}\right] \tag{3.24}$$

In (3.24) $\mathcal{E}$ and Y define each other uniquely, then the problem of estimating $\hat{x}$ from y would be equal to estimating the $\hat{x}$ from $\mathcal{E}$, then the least mean square estimator of x can be given as

$$\hat{x} = R_{x\mathcal{E}}R_{\mathcal{E}}^{-1}\mathcal{E} \tag{3.25}$$

According to least mean square estimation theory [H2001] the estimate of $\hat{x}$ in (3.25) is equivalent to

$$\hat{x} = R_{xY}R_Y^{-1}Y$$

The key advantage of working with $\mathcal{E}$ instead of Y is that the covariance matrix of $R_{\mathcal{E}}^{-1}$ is block diagonal. Expanding (3.25) yields

$$\hat{x} = \sum_{i=0}^{k}(E(x\varepsilon_i^*))\,R_{\mathcal{E}}^{-1}\mathcal{E} = \sum_{i=0}^{k}\hat{x}_i \tag{3.26}$$

Which means that estimator $\hat{x}$ can be evaluated as the combined sum of individual estimators and constitute the basic principle of recursive estimation. Now introduce the time index to indicate directly that estimation is based on all previous observations,

$$\hat{x}[k] = \sum_{i=0}^{k} \hat{x}[k] = \sum_{i=0}^{k-1} \hat{x}[k] + \hat{x}[k] \tag{3.27}$$

It follows that

$$\hat{x}[k+1] = \hat{x}[k] + E(x\varepsilon^{*}[k])R_{\varepsilon}^{-1}[k]\varepsilon[k] \tag{3.28}$$

The (3.28) shows that how the estimate can be obtained by adding the recent measurement. The variable ε can be defined by following the Gram-Schmidt procedure. The uniquely defined ε from Y and vice versa leads to following

$$\varepsilon[k] = y[k] - \hat{y}[k] \tag{3.29}$$

The definition of ε is the same defined earlier in (3.4) and is known as innovation. The ε and y are orthogonal to each other and hence are uncorrelated. By this virtue it deduced that all ε are orthogonal and uncorrelated with each other. Now consider the joint state space model of (3.19) in the presence of process and observation noise

$$\hat{X}_j[k+1] = \mathrm{A}_j\big(\hat{X}_j[k]\big)\Delta\big(X_j[k]\big) + \mathrm{B_j}\big(\hat{Z}[k]\big)\Delta(Z[k]) + N[k] \tag{3.31.a}$$

$$\hat{Y}[k] = \mathrm{C}_j\big(\hat{X}[k]\big)\Delta\big(X_j[k]\big) + V[k] \tag{3.31.b}$$

Whereas $N[k]$ and $V[k]$ are the process noise and observation noise respectively at time instant$k$. The experimental setup employ hardware sampling units to gather the cycles elapsed in a given interval. This means that in the case of workload forecasting there will be no observation noise. This render that the derivation of EKF has to consider only process noise and let's define the covariance of process noise to be

$$Q = E(N[k](N[k])^{T}) \tag{3.32}$$

And consequently from (3.29)

$$\varepsilon[k] = Y[k] - \hat{Y}[k] = Y[k] - C_j X_j[k] \tag{3.33}$$

Therefore, problem of finding the $\varepsilon[k]$ innovations reduces to finding $X_j[k]$. Then using (3.28) and (3.33) yields

$$X_j[k+1] = \hat{X}_j[k+1] + E(X[k+1]\varepsilon^{*}[k])R_{\varepsilon}^{-1}[k](Y[k] - C_j X_j[k]) \tag{3.34}$$

Since $X_j[k+1]$ obeys the state equation then

$$X_j[k+1] = \mathrm{A}_j\big(\hat{X}_j[k]\big)\Delta\big(X_j[k]\big) + 0 \tag{3.35}$$

Combining the equations (3.32) to (3.35) yields

$$\varepsilon[k] = Y[k] - C_j X_j[k] \tag{3.36.a}$$

$$X_j[k+1] = \mathrm{A}_j\big(\hat{X}_j[k]\big)\Delta\big(X_j[k]\big) + F[k]\varepsilon[k] \tag{3.36.b}$$

Whereas the gain matrix is defined as

$$F[k] = \varepsilon[k]\, E(X[k+1]\varepsilon^*[k])R_\varepsilon^{-1}[k] \tag{3.37}$$

The (3.36) defines the complete recursion for state estimation. The gain matrix is defined but requires a recursive solution to update it on the arrival of new observation. To find the recursion of the gain matrix define the state error covariance as

$$P[k] = E\big((\tilde{X}_j[k])(\tilde{X}_j[k])^T\big) \tag{3.38}$$

In workload forecasting problem there is no observation noise and states are the linear combinations of observations therefore

$$\hat{R}[k+1] = R[k] + C_j P[k] C_j^T \tag{3.39}$$

Likewise

$$E(X[k+1]\varepsilon^*[k]) = \; P[k]C_j^T \;\;,\;\; E(N[k]\varepsilon^*[k]) = S[k] \tag{3.40}$$

Following is obtained from (3.36) to (3.40)

$$F[k+1] = \big(A_j[k]\; F[k]C_j^T\big) + (B_j[k]S[k]\;)\varepsilon^*[k])R_\varepsilon^{-1}[k] \tag{3.41}$$

Likewise after obtaining the recursive solution for the gain the state error covariance in terms of it can be found as

$$P[k+1] = \left(A_j[k]\,P[k]A_j^T\right) + \left(B_j[k]Q[k]B_j^T[k]\right) - F[k]R_\varepsilon(F[k])^T \qquad (3.42)$$

The (3.36), (3.38) (3.39),(3.41) and (3.42) formulates the recursive solution for the workload forecast.

3.7 Regulator

The regulator works in conjunction with the observer and selects the appropriate voltage frequency pair in accordance to forecasted workload. The qualms in forecasted workload may incur penalties and it has to take care of these issues to maintain certain QoS. Furthermore, it has to minimize the back and forth frequency oscillations to reduce energy consumption in state transitions. To accomplish all these requirements, a methodology incorporating the confidence level of forecast is developed. The state vector error-covariance in Kalman forecast is a measure of the confidence level. The error covariance does not define any bounds, which makes it very difficult to interpret the exact confidence. In this case, normalized covariance can be used and is known as Pearson correlation coefficient. Then the absolute Pearson correlation of state error is given

$$X_{error}[k] = X_j[k] - \hat{X}_j[k]$$

$$\eta[k] = abs[cov(X_{error}[k])/\sigma(X_{error}[k]) \qquad (3.43)$$

The absolute value bound the normalized covariance in an interval [0 1], which can be interpreted as 0 for no confidence and 1 for full confidence. The lack of confidence in forecast results in either performance or power penalty. In the presence of confidence level user can select the tradeoff between power or performance penalty, lets define the parameter $U_r = [0\ 1]$ as the user selectable reference to ensure the QoS. Then the target frequency adjusted according to forecast confidence and user define reference is given by

$$y_{adj}[k+1] = \begin{cases} \hat{y}[k+1] + \hat{y}[k+1](1-U_r) & \eta \geq 0.75 \\ \hat{y}[k+1] + \hat{y}[k+1](1-U_r\eta) & \eta < 0.75 \end{cases} \qquad (3.44)$$

The (3.44) add safeguard workload (cycles) to minimize the risk of performance penalty. In case of high confidence estimate ($\eta \geq 0.75$), (3.44) does not take into account η and safeguard workload is solely due to the user defined QoS parameter. In the second case, when confidence level is low the product of U_r and η becomes the defining parameter for a safeguard workload. After obtaining the adjusted workload regulator selects the appropriate voltage frequency pair to service the workload.

Chapter 4 Stochastic Low Power Scheduling

The previous chapter develops the periodic performance throttling for multicore processors by employing the joint state space model and recursive estimator. The periodically invoked DVFS manager does not take in to account the number of tasks running in an interval and respective deadlines. This constraint limits the practical use of such a scheme in a real time system, where tasks must be serviced according to desired quality of service. Figure 4.1 shows three different execution scenarios in a periodic DVFS interval, in the first scenario only one task is executing in an interval and deadline issue can be catered. In case of multicore systems, it is evident that more and more fine grained tasks will execute in the system to make use of available parallel execution platform. It means that the number of task in a DVFS interval will not be fixed, and many tasks start and finish their execution in a single interval. In that scenario, the timeliness constraints of individual task cannot be respected that requires the development of DVFS scheme at task scheduling level.

Moreover, the assumption about the distribution may not hold well in case of execution time forecast of a lone task. Each task might have its own execution time distribution therefore, central limit theorem may not hold in that scenario. Because of this, it is necessary to obtain the execution times orthogonal to task execution time distribution. This chapter discusses in detail the framework for the distribution independent task execution time estimation. In this chapter, the estimation of execution time is carried out by developing the framework of self-organizing state space modeling and Sequential Monte Carlo (SMC) method. In addition to the distribution

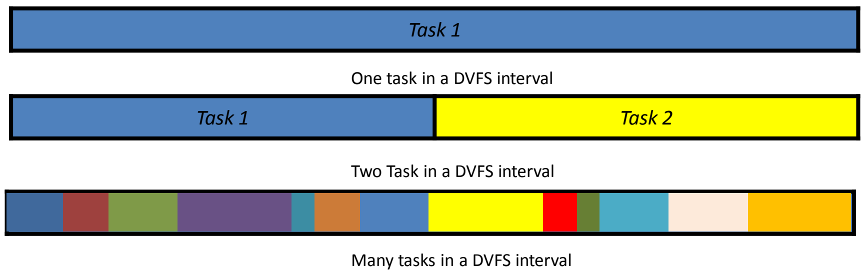

Task 1
One task in a DVFS interval
Task 1
Task 2
Two Task in a DVFS interval
Many tasks in a DVFS interval

Figure 4.1: Periodic DVFS interval has no information about individual task

independent recursive solution of the state space, SMC does not require linearization of the model. The linearization requirement may lead to the dead end as certain state space models may not be linearizeable, for example, if a state space model has particular cyclic variables then differentiation will not be able to eliminate the cycle and as a result of which the state space model will remain non linear even after many iterations of differentiation. The inherently nonlinear and non stationary state space modeling, independent of statistical characteristics, and SMC recursive solution make it ideal for the performance throttling of processors. Moreover, the ever-diminishing feature size increases the susceptibility of future systems to soft errors. It is mandatory to cater the soft error issues in low power scheduling to keep the system failure probability constant. This chapter discusses step by step, the design and development of the low power scheduling mechanisms and framework for the soft error aware scheduling.

4.1 Low Power Scheduling

The workloads on the multicore processor can also be viewed from the perspective of different tasks being mapped on the processor. Due to numerous reasons, the tasks running in a software pipeline may be inter-related and thus influencing each other's execution time, as discussed in Chapter 2. Real time low power scheduling requires an accurate estimate of the execution time, which in turn affects the mapping stage and ultimately the processor's performance in terms of the energy-performance trade-off. In particular, time-critical tasks must be serviced within certain pre-determined deadlines, dictated by the required quality of service. The most important attribute of real-time systems is that the correctness of such systems depends not only on the computed results but also on the time at which the results are produced (timing guarantee). The multicore systems pose new challenges while opening up new design opportunities for low power scheduling. Therefore, it is essential to examine the problem of low power scheduling in the context of multicore environment to devise efficient methods and tools.

General Scheduling of a task by any policy can be characterized by an integer tuple (S_t, E_t, D_t) , where S_t is the arrival of a task in the ready queue, E_t is the execution time and D_t defines the dead line of a task i.e. it must be completed before D_t to ensure the timing correctness. The S_t and D_t can be known exactly in a scheduling interval, but E_t is a non-trivial entity in the context of scheduling. As motivated in Chapter 2, the advances in the multicore systems become a part of the problem rather than the solution to determine the execution time for real time systems. Futuristic computer architectures and software have made it difficult or impossible to estimate the execution time of the task at design time [L2005]. WCET results in sever under utilization of resources and leave remarkably little space for the processor performance throttling. In this scenario, it is essential for low power scheduling that the estimation of E_t must be carried out at run time without 'a priori' assumptions. Obtaining the E_t at run time with acceptable accuracy completes the task scheduling tuple (S_t, E_t, D_t). After having full information about the scheduling parameters and the Soft Error Rate (SER), the low power scheduling becomes a trivial case by taking the decision using scheduling algorithms such as Earliest Deadline First (EDF). Since the estimation will accompany with some degree of uncertainty, the deadlines cannot be guaranteed, and the solutions devised in this chapter are restricted to soft real-time systems.

4.2 State Space Modeling for Task Execution Time

To proceed forward, consider the task graph of Figure 4.2 with four tasks. Task T1 is the predecessor (Producer) and the first task in the software pipeline. In this task graph, T1 processes the data and hands it over to tasks T2 and T3 (consumers) for further processing. The software pipelining of T2 and T3 as consumer of T1 constitute the parallel processing in time (forking). The task T4 has two predecessors namely T2 and T3, and act as the joining task for both, parallel processing in space (Joining). In the task graph of Figure 4.2, no task is consumer and producer at the same time for any task. It means that in this thesis only acyclic linear task graphs are considered. In theory, there is no restriction to model and formulate the solution for cyclic graphs, but practical limitations prohibit it. This limitation stems from the formation of ill-posed first order difference equations in the model, which seriously hampers the forecast quality. The task graph of Figure 4.2 is comprehensive in its nature as it contains both forms of the parallelism (parallelism in time and space). This scenario is considered to aid in understanding the derivation of the state-space model; a general solution for an arbitrary number of tasks in a task graph is developed subsequently.

To model the execution time of the tasks of Figure 4.2 in a state space, let's pose it as a multivariate problem in a time series [IH2010, IAH2010]. Figure 4.3 depicts the execution time estimation process in pictorial form. In Figure 4.3, $y_{T1}[\mathrm{k}], y_{T2}[\mathrm{k}], y_{T3}[\mathrm{k}]$ and $y_{T4}[\mathrm{k}]$ are the execution times in kth scheduling interval, of task T1, T2, T3 and T4 respectively. The aim is to estimate the execution times in the next interval, $k+1$. To proceed forward consider task T1, it is the first task in the task graph and has no predecessor task. In the absence of predecessor task, the forecast of execution time of T1 can only be carried out on the basis of temporal dependence. If

α_{T1} is the temporal dependence coefficient associated with the interval k then the estimated execution time for the interval $k+1$ considering the first order difference defined as

$$\hat{y}_{T1}[k+1] = \alpha_{T1} y_{T1}[k] \tag{4.1}$$

whereas $\hat{y}_{T1}(k+1)$ is the estimated execution time in the next interval. Now consider task T2 with predecessor T1, having temporal and spatial dependence with itself and with T1 respectively.

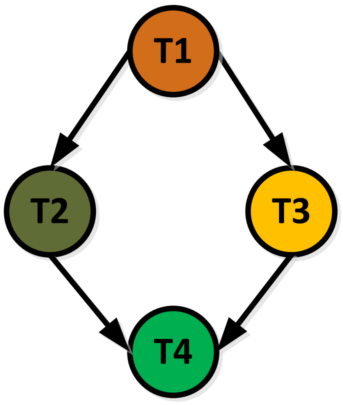


Figure 4.2 Application task graph for state space modeling

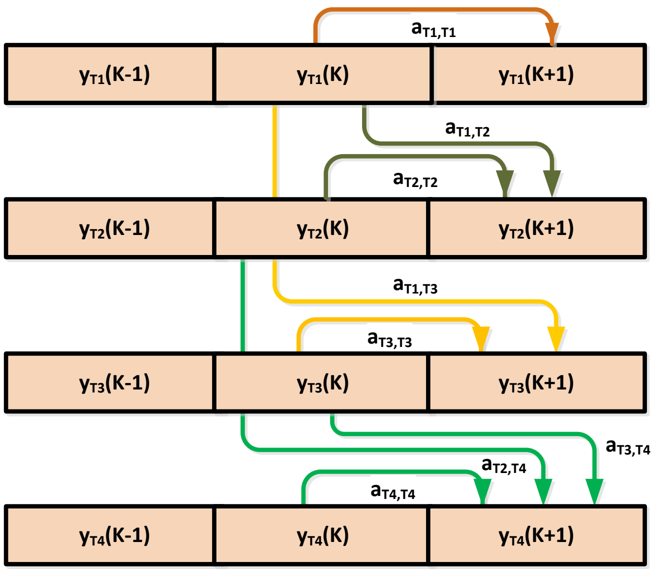


Figure 4.3 Execution time forecast with temporal and spatial dependencies

In order to form multivariate second order difference equation for T1, lets define the temporal dependence coefficient as α_{T2} and spatial dependence coefficient as $\alpha_{T1,T2}$. $\alpha_{T1,T2}$ defines the extent of spatial correlation between T1 (the predecessor) and T2 (the successor). Then, the second order multivariate difference equation for T2 is given by

$$\hat{y}_{T2}[k+1] = \alpha_{T1,T2} y_{T1}[k] + \alpha_{T2,} y_{T2}[k] \tag{4.2}$$

Similarly, multivariate regression equations for T3 and T4 respectively can be obtained as shown below, (for elaboration refer to Figure 4.3).

$$\hat{y}_{T3}[k+1] = \alpha_{T1,T3} y_{T1}[k] + \alpha_{T3} y_{T3}[k] \tag{4.3}$$

$$\hat{y}_{T4}(k+1) = \alpha_{T2,T4} y_{T2}[k] + \alpha_{T3,T4} y_{T3}[k] + \alpha_{T4} y_{T4}[k] \qquad 4.4)$$

Arranging (4.1, 4.2, 4.3, 4.4) as the system of linear equations and rewriting in matrix form yields

$$\begin{bmatrix} \hat{y}_{T1}[k+1] \\ \hat{y}_{T2}[k+1] \\ \hat{y}_{T3}[k+1] \\ \hat{y}_{T4}[k+1] \end{bmatrix} = \begin{bmatrix} \alpha_{T1} & 0 & 0 & 0 \\ \alpha_{T1,T2} & \alpha_{T2} & 0 & 0 \\ \alpha_{T1,T3} & 0 & \alpha_{T3} & 0 \\ 0 & \alpha_{T2,T4} & \alpha_{T3,T4} & \alpha_{T4} \end{bmatrix} \begin{bmatrix} y_{T1}[k] \\ y_{T2}[k] \\ y_{T3}[k] \\ y_{T4}[k] \end{bmatrix}$$

Naturally, the estimation will accompany with an error between measured and forecasted execution time; it is referred to as ‘innovation’, and for any task Ti is defined as

$$\varepsilon_{Ti}[k] = y_{Ti}[k] - \hat{y}_{Ti}[k] \tag{4.5}$$

The above system of linear equations and the definition of error imply the direct formulation of state space model. The reader is referred to Chapter 3 for the basic understanding of state space modeling and general form is given by [CK2007].

$$\hat{\boldsymbol{x}}[k+1] = \mathbf{A}\boldsymbol{x}[k] + \mathbf{B}\boldsymbol{z}[k] \qquad (4.6.a)$$
$$\hat{y}[k] = \mathbf{C}\boldsymbol{x}[k] \qquad (4.6.b)$$

The components of the state space model for the problem of Figure 4.3 can be obtained by rearranging the equations and the resultant state space model is

$$\begin{bmatrix} x_1[k+1] \\ x_2[k+1] \\ x_3[k+1] \\ x_4[k+1] \end{bmatrix} = \begin{bmatrix} \alpha_{T1} & 0 & 0 & 0 \\ \alpha_{T1,T2} & \alpha_{T2} & 0 & 0 \\ \alpha_{T1,T3} & 0 & \alpha_{T3} & 0 \\ 0 & \alpha_{T2,T4} & \alpha_{T3,T4} & \alpha_{T4} \end{bmatrix} \begin{bmatrix} x_1[k] \\ x_2[k] \\ x_3[k] \\ x_4[k] \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \varepsilon_{T1}[k] \\ \varepsilon_{T2}[k] \\ \varepsilon_{T3}[k] \\ \varepsilon_{T4}[k] \end{bmatrix} \qquad (4.7.a)$$

$$\begin{bmatrix} \hat{y}_{T1}[k+1] \\ \hat{y}_{T2}[k+1] \\ \hat{y}_{T3}[k+1] \\ \hat{y}_{T4}[k+1] \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1[k+1] \\ x_2[k+1] \\ x_3[k+1] \\ x_4[k+1 \end{bmatrix} \qquad (4.7.b)$$

The system matrix A defines the model of parallel tasks and encapsulates all dependencies and associated coefficients. In the derivation of state space model, we define the single coefficients, $\alpha_{i,j}$ for each dependency. The primary goal of the solution of the state space model is to minimize the forecast error and hence the objective function, which can be defined as

$$J_{min} = Min(\varepsilon_{Ti}[k]) \qquad (4.8)$$

Let’s define the model parameter vector as

$$W = \left[\alpha_{T1}\ \alpha_{T1,T2}\ \alpha_{T2}\ \alpha_{T1,T3}\ \alpha_{T3}\ \alpha_{T2,T4}\ \alpha_{T3,T4}\ \alpha_{T4,T4}\right]^T \qquad (4.9)$$

Similarly, the development of the state space model for periodic performance throttling as discussed in Chapter 3, the state space model parameter vector of (4.9) is not known. In general, to find the model parameters for task execution time estimation, system identification processes are employed at design time and recalibration runs are performed from time to time to adapt the parameters according to the changing scenarios [IZP1999]. Model parameters obtained by calibration runs cannot adapt rapid changes in the scenarios at run-time and are always valid under certain assumptions as argued in the motivation part of Chapter 2. No assumption can be made on execution time or its characteristics and hence it is essential to derive the solution independent of the prior knowledge of system dynamics. The Chapter 3 develops the joint state space model for periodic performance throttling and workload estimates are obtained by EKF. The estimates of EKF are piece wise linear approximation to the non linear workload conditions that requires the linearization of the state space model at design time and computation of differentials at run time as well. The dynamics of workload in a period change slowly compared to the execution time dynamics of a single task. This stems from the fact that the averaging effect of execution times of numerous tasks makes workload conditions less dynamic and hence piece wise linear estimate error falls in the tolerable limits. Because of inherently non linear characteristics the solution of task execution time is obtained by employing the framework of SMC, an inherently non linear estimation method. The idea of estimating the state space model parameters in conjunction with the system states remains the same, but now it is required to derive the self-organizing state space model suitable for the estimation by SMC. To develop the self-organizing state space model for SMC estimation, lets define the new augmented vector by augmenting (4.9) in the state vector of (4.7.a), given by

$$X_J[k] = \big[X[k] \;\; W[k]\big]^T \tag{4.10}$$

It is important to note that the augmented state vector model parameters are also time-varying and updated at the beginning of each interval; like states. As a result, the state space of (1.a,b) with parameters in the state vector becomes time-varying and the generalized form of time update and measurement update equations (4.6.a,b) of self-organizing state space model for SMC estimation will take the following form

$$X_S[k+1] = A_S(W(k),k)X_S[k] + B_S Z_S[k] \tag{4.11.a}$$
$$\hat{Y}(k) = C_S X_S[k+1] \tag{4.11.b}$$

Because of the non linear estimation properties of SMC, the derivation of the self-organizing state space model is straight forward whereas, the components of general state space models are augmented with identity ($I_{i,i}$) and null ($O_{i,i}$) matrices to satisfy the system of equations (number of equations must be equal to the number of unknowns). The state vector of (4.9) and the augmented components of (4.7.a,b) define the self-organizing state space components and are given by

$$\left\{\begin{array}{cc} A_S = \begin{bmatrix} A & O_{4,8} \\ O_{8,4} & I_{8,8} \end{bmatrix} & X_S(k) = \begin{bmatrix} X[k] \\ W[k] \end{bmatrix} \\ X_S[k+1] = \begin{bmatrix} X[k+1] \\ W[k+1] \end{bmatrix} & Z_S[k] = \begin{bmatrix} Z[k] \\ O_{8,1} \end{bmatrix} \\ B_S = \begin{bmatrix} B & O_{4,8} \\ O_{8,4} & O_{8,8} \end{bmatrix} & C_S = \begin{bmatrix} C & O_{4,8} \\ O_{8,4} & O_{8,8} \end{bmatrix} \end{array}\right\} \tag{4.12}$$

By putting the components from (4.12) into (4.11.a,b), yields a complete self-organizing state space model for estimation by SMC

$$\begin{bmatrix} X[k+1] \\ W[k+1] \end{bmatrix} = \begin{bmatrix} A & O_{4,8} \\ O_{8,4} & I_{8,8} \end{bmatrix} \begin{bmatrix} X[k] \\ W[k] \end{bmatrix} + \begin{bmatrix} B & O_{4,8} \\ O_{8,4} & O_{8,8} \end{bmatrix} \begin{bmatrix} Z[k] \\ O_{8,1} \end{bmatrix} \qquad (4.13.a)$$

$$\hat{Y}[k] = \begin{bmatrix} C & O_{4,8} \\ O_{8,4} & O_{8,8} \end{bmatrix} \begin{bmatrix} X[k+1] \\ W[k+1] \end{bmatrix} \qquad (4.13.b)$$

The state space model developed in (4.13.a,b), is suitable for the recursive solution (estimation) by SMC. Moreover, the execution times of 4 tasks are estimated simultaneously and the number of tasks required to be modeled in a single state space depends on the choice of low power scheduling scheme designer. It is worthwhile to mention that the resultant matrices of the self-organizing state space model are sparse and shown in their full form for the sake of understanding and compliance with the theory. The sparse property and regularity of the model significantly reduces the computation cost at run time. Consider the measurement update equation (4.13.b); at first it appears that it involves the multiplication of matrix of dimension 12×12 with vector of dimension 12×1. But in actual implementation, there is no multiplication involved and the first four states of vector $X_S[k]$ are copied to $\hat{Y}[k]$. Similar patterns can also be observed in the time update equation e.g. the multiplication of observations and input matrix. Therefore, in actual implementation, the solution of the state space requires only a handful of computations.

The regularity in the derived state space model of (4.13.a,b) allows the direct derivation of the model by algorithm for an arbitrary number of tasks in a task graph. The task graph may contain the isolated task(s) and comprise of collection of arbitrary number of sub graphs. Algorithm 4.1 outlines a systematic procedure for the derivation of the self-organizing state space model for estimation by SMC. It is important to note that the labeling of tasks should be carried out in such a way that the predecessor task must be labeled before all of its successors. However, defined labeling rule is not a constraint but has the implications for the implementation. If tasks are labeled according to the rule then the system matrix will always be a diagonal matrix. The diagonal matrix means, half of the operations of major computational part are not required at all (multiplication by zeros).

4.3 Execution Time Estimation Incorporating Phase

In the previous section, the self-organizing state space model is developed by considering the temporal and spatial dependencies among tasks to obtain the quality forecast of execution times. The motivation section in Chapter 2 highlights the fact that the task execution times observe the high computational loads followed by low computational load and vice versa; the change of computational load and consequently the task execution time are known as execution time phases. The state space model developed in previous section has the tendency to suffer from the large error with the abrupt change of execution time phase. The DVFS literature sort to predict the phase and duration for DVFS schemes [IBM2005], but all these methods suffer from limitations and always assume that some kind of application knowledge is available. As highlighted in Chapter 2, the execution times or their characteristics are the function of run time and cannot be determined exactly at design time. At design time, it is not known when such a

Algorithm 4.1: Self-organizing state space derivation

```
$Event : Derive\ self-organizing\ state\ space\ model\ for\ \Omega\ number\ of\ tasks$
$Definitions :$
$A, B, C, X and\ Z\ are\ system\ matrix, input\ matrix, output\ matrix, state\ vector\ and$
$observation\ vector\ respectively.$
$A_s, B_s, C_s, X_s and\ Z_s\ are\ the\ state\ space\ counterparts\ of\ A, B, C, X, and\ Z\ respectively$
1 : $Label\ the\ tasks\ in\ ascending\ order\ in\ such\ a\ way\ that\ predecessor\ should\ be$
2 : $labeled\ before\ its\ successor\ tasks$
3 :  $A = null_{\Omega\times\Omega}, B = C = identity_{\Omega\times\Omega}, cnt = 0$
4 :  $X = [x_1, x_2, \dots x_\Omega]^T, Z = [\varepsilon_{T1}, \varepsilon_{T2}, \dots \varepsilon_{T\Omega}]^T$
5 :  $for(i = 1; i \le \Omega; i++)$
6 :      $for(k = 1; k \le \Omega; k++)$
7 :          $if(i == k)$
8 :              $A(i,i) = a_{Ti,Ti};$
9 :              $X = [X\ a_{Ti,Ti}]^T;$
10:                  $cnt = cnt + 1;$
11:         $endif(k < i)$
12:          $if(T_i\ is\ predecessor\ of\ T_k)$
13:              $A(i,k) = \beta_{Ti,Tk};$
14:              $X = [X\ a_{Ti,Tk}]^T$
15:                  $cnt = cnt + 1;$
16:         $end$
17:     $end$
18: $end$
19: $X_s = X\ , Z_s = Z\ , A_s = \begin{bmatrix} A & O_{\Omega,\mathrm{cnt}} \\ O_{\mathrm{cnt},\Omega} & I_{\mathrm{cnt,cnt}} \end{bmatrix}$
20: $B_s = \begin{bmatrix} B & O_{\Omega,\mathrm{cnt}} \\ O_{\mathrm{cnt},\Omega} & O_{\mathrm{cnt,cnt}} \end{bmatrix}, C_s = \begin{bmatrix} C & O_{\Omega,\mathrm{cnt}} \\ O_{\mathrm{cnt},\Omega} & O_{\mathrm{cnt,cnt}} \end{bmatrix}$
```

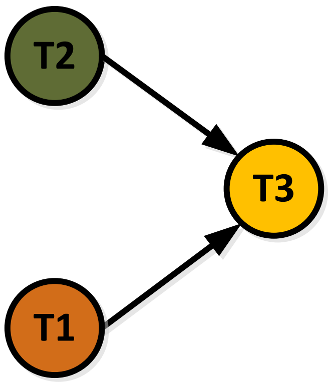


Figure 4.4: Task graph for the state space model incorporating phase

phase shift will occur, due to the dynamic nature of the data to be processed. In this context, an accurate forecast of the execution time necessitates the modeling of local as well as the global phase. The incorporation of the phase information can significantly improve the forecast quality because estimator can respond instantly to the change of phase. To model the execution time estimation problem in the state space incorporating the phase information, considers the task graph of Figure 4.4, which depicts a software pipeline of three tasks: T1, T2 and T3. T3 is the consumer of T1 and T2. This task graph is chosen to highlight the execution time phase modeling in state space as T1 and T2 are independent tasks and has no spatial dependencies.

At the kth scheduling interval, the objective is to estimate the execution time of the tasks for the $(k+1)$th scheduling interval. Considering T1, it is the first task in the task graph and has no predecessor. In the absence of a predecessor task, the execution time forecast of T1 can be carried out only on the basis of temporal dependencies and phase information. Lets define the states for T1. If α_{T1} is the temporal coefficient associated with the interval k and y_{T1} is the actual execution time, then the estimated execution time for the interval $k+1$ as a state of T1 can be defined as a function given below [ISH2011]:

$$x_1[k+1] = \hat{y}_{T1}[k+1] = f(\alpha_{T1}, y_{T1}[k], y_{T1}[k-1], y_{T1}[k-1], \hat{y}_{T1}[k]) \qquad (4.14)$$

The phase change of the execution time of a task can be detected by taking its first-order derivative over successive intervals. Figure 4.5 depicts the first-order derivative of an example execution time. When the execution time is constant, the derivative is zero; no phase shift occurs. In case of an increasing execution time, the derivative yields a positive value whereas, a negative value occurs when the task undergoes a transition from a high to low computational load. It is inferred that the first-order derivative is suited to capture the phase changes in the execution time. Considering this, lets define the first-order derivative as the second state for T1, as following:

$$x_2[k+1] = d\, y_{T1}[k]/dt = y_{T1}[k] - y_{T1}[k-1] \tag{4.15}$$

Similarly, states for T2 can be defined as

$$x_3[k+1] = \hat{y}_{T2}[k+1]$$

$$\hat{y}_{T2}[k+1] = g(\alpha_{T2}, y_{T2}[k]\,, y_{T2}[k-1], y_{T2}[k-2]) \tag{4.16.a}$$

$$x_4[k+1] = d\, y_{T2}[k]/dt = y_{T2}[k] - y_{T2}[k-1] \tag{4.16.b}$$

To model the execution time of T3, we can benefit from temporal as well as spatial factors. Let $\alpha_{\mathrm{T1,T3}}$ and $\alpha_{\mathrm{T2,T3}}$ be the spatial coefficients for dependence of execution time of T3 on execution times of T1 and T2 respectively. Then the states of T3 can be defined as

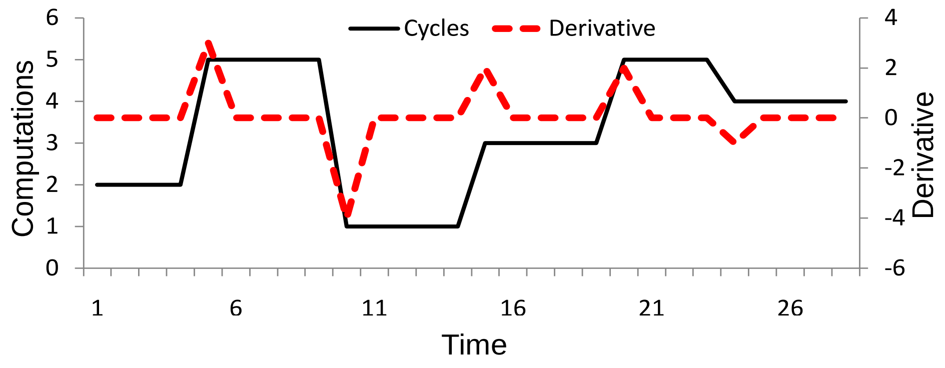
$$x_5[k+1] = \hat{y}_{T3}[k+1]$$

$$\hat{y}_{T3}[k+1] = h(\alpha_{T1,T3}, \alpha_{T2,T3}, \alpha_{T3}, x_1[k], x_3[k]) \tag{4.17.a}$$

$$x_6[k+1] = d\, y_{T3}[k]/dt = y_{T3}[k] - y_{T3}[k-1] \tag{4.17.b}$$

The error between the predicted and the actual execution time for any task is defined as

$$\varepsilon_{Ti}[k] = y_{Ti}[k] - \hat{y}_{Ti}[k] \tag{4.18}$$



Fisgure 4.5: Detection of phase by first order derivative

And we define $\varepsilon_{\delta \mathrm{T}i}[\mathrm{k}]$ for any task i as the difference between the actual execution times (separated by an interval in time), as given below:

$$\varepsilon_{\delta Ti}[k] = y_{Ti}[k] \ - y_{Ti}[k-2] \tag{4.19}$$

By forming a system of equations from (4.15, 4.16. 4.17) with states in ascending order and with the above-mentioned definition in (4.19) and (4.6), we are implicitly arriving at the derivation of state-space components,

$$\begin{bmatrix} x_1[k+1] \\ x_2[k+1 \\ x_3[k+1] \\ x_4[k+1 \\ x_5[k+1] \\ x_6[k+1 \end{bmatrix} = \begin{bmatrix} \alpha_{T1} & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha_{T2} & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \alpha_{T1,T3} & 0 & \alpha_{T2,T3} & 0 & \alpha_{T3} & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1[k] \\ x_2[k] \\ x_3[k] \\ x_4[k] \\ x_5[k] \\ x_6[k] \end{bmatrix} + I_{6\times 6} \begin{bmatrix} \varepsilon_{T1}[k] \\ \varepsilon_{\delta \mathrm{T}1}[\mathrm{k}] \\ \varepsilon_{T2}[k] \\ \varepsilon_{\delta \mathrm{T}2}[\mathrm{k}] \\ \varepsilon_{T3}[k] \\ \varepsilon_{\delta \mathrm{T}3}[\mathrm{k}] \end{bmatrix} \tag{4.20.a}$$

$$\begin{bmatrix} \hat{y}_{T1}[k+1] \\ \hat{y}_{\delta T1}[k+1 \\ \hat{y}_{T2}[k+1] \\ \hat{y}_{\delta T2}[k+1 \\ \hat{y}_{T3}[k+1] \\ \hat{y}_{\delta T3}[k+1 \end{bmatrix} = I_{6\times 6} \begin{bmatrix} x_1[k+1] \\ x_2[k+1 \\ x_3[k+1] \\ x_4[k+1 \\ x_5[k+1] \\ x_6[k+1 \end{bmatrix} \tag{4.20.b}$$

The mathematical expressions of the functions are implied by the state-space. From the resultant state space model, it is evident that the states x_2, x_4 and x_6 represent the differential of the task execution time phase. No system coefficient is associated with the phase differential because the phase duration is always equal or longer than the interval because of which it will be estimated under the influence of the random walk. The input vector will comprise of the error between the forecasted and the actual execution time, and the errors between the phase differentials. The Algorithm 4.2 facilitates the automatic derivation of state space model for an arbitrary number of tasks incorporating phase differential. To minimize the error in the state estimation the objective function will remain the same and given by

$$J_{min} = Min(\varepsilon_{Ti}[k]) \tag{4.21}$$

In the above developed state space model the extent of the temporal dependencies, spatial dependencies and phase differential of the task cannot be known a priori, and must be adapted at run-time. To adapt the coefficients of the system matrix as it encapsulates all the dependencies

Algorithm 4.2: Self-organizing State space model derivation incorporating phase

```
$Event : Derive\ self-organizing\ state\ space\ model\ for\ \Omega\ number\ of\ tasks$
$Definitions :$
$A, B, C, X and\ Z\ are\ system\ matrix, input\ matrix, output\ matrix, state\ vector\ and$
$observation\ vector\ respectively.$
$A_s, B_s, C_s, X_s and\ Z_s\ are\ the\ state\ space\ counterparts\ of\ A, B, C, X, and\ Z\ respectively$
1 : $Label\ the\ tasks\ in\ ascending\ order\ in\ such\ a\ way\ that\ predecessor\ should\ be$
2 : $labeled\ before\ its\ successor\ tasks$
3 :  $A = null_{2\Omega\times2\Omega}, B = C = identity_{2\Omega\times2\Omega}, cnt = 0$
4 :  $X = [x_1, x_2, \dots x_{2\Omega}]^T, Z = [\varepsilon_{T1}, \varepsilon_{\delta T1}, \varepsilon_{T2}, \varepsilon_{\delta T2}, \dots, \varepsilon_{T\Omega}, \varepsilon_{\delta T\Omega}]^T$
5 :  $for(i = 1; i \le 2\Omega; i++)$
6 :      $for(k = 1; k \le 2\Omega; k++)$
7 :          $if(i == k\ \&\&\ isodd(i))$
8 :              $A(i, k) = a_{Ti};$
9 :              $A(i, k+1) = 1;$
10:              $cnt = cnt + 1;$
11:          $else$
12:              $A(i, k) = 1;$
13:              $cnt = cnt + 1;$
14:          $end$
15:          $if(k < i)$
16:            $if((T_i\ is\ predecessor\ of\ T_k)\ \&\&\ isodd(i))$
17:              $A(i, k) = \beta_{Ti,Tk};$
18:              $cnt = cnt + 1;$
19:            $end$
20:          $end$
21:      $end$
22: $end$
23:  $X_s = X, Z_s = Z, A_s = \begin{bmatrix} A & O_{2\Omega,2\Omega} \\ O_{2\Omega,\Omega} & I_{2\Omega,2\Omega} \end{bmatrix}$
24:  $B_s = \begin{bmatrix} B & O_{2\Omega,2\Omega} \\ O_{2\Omega,\Omega} & O_{2\Omega,2\Omega} \end{bmatrix}, C_s = \begin{bmatrix} C & O_{2\Omega,2\Omega} \\ O_{2\Omega,2\Omega} & O_{2\Omega,2\Omega} \end{bmatrix}$
```

and the system dynamics, lets define the vector of unknown coefficients (subsequently called the parameter vector) as

$$W = [\alpha_{T1} \quad \alpha_{T2} \quad \alpha_{T3} \quad \beta_{T1,T3} \quad \beta_{T2,T3}]^T \tag{4.22}$$

The self-organizing state space model suitable for SMC estimation can be derived directly by considering the joint parameter vector. The steps involved in this are similar to the derivation in the previous section. In the model, the system matrix and the output and the input matrices are augmented with identity ($I_{i,i}$) and/or null ($O_{i,i}$) matrices to satisfy the system of equations. After obtaining the state space model for p number of tasks from Algorithm 4.2 and by considering the joint state vector of (4.22), the state vector has the dimension of $2p$ (i.e. two states for each task). First consider p unknown system matrix coefficients, then the self-organizing state space model for any task graph can be defined by replacing the components of a general state space model in

(4.11). The self-organizing state space model for p tasks with q unknown coefficients in the system matrix is given by

$$\begin{bmatrix} X[k+1] \\ W[k+1] \end{bmatrix}_{(2p+q),1} = \begin{bmatrix} A_{2p,2p} & O_{2p,q} \\ O_{q,2p} & I_{q,q} \end{bmatrix} \begin{bmatrix} X[k] \\ W[k] \end{bmatrix}_{(2p+q),1} + \begin{bmatrix} B_{2p,2p} & O_{2p,q} \\ O_{q,2p} & O_{q,q} \end{bmatrix} Z_s(k) \quad (4.23.a)$$

$$\hat{Y}[k] = \begin{bmatrix} C_{2p,2p} & O_{2p,q} \\ O_{q,2p} & O_{q,q} \end{bmatrix} \begin{bmatrix} X[k+1] \\ W[k+1] \end{bmatrix} \quad (4.23.b)$$

Again, the obtained self-organizing state space model is sparse in nature and in implementation phase only necessary operations are carried out. The recursive solution of self-organizing state space model is found by employing the SMC method. The next section discusses the SMC frame work developed to find the recursive solution of model to forecast the task execution times.

4.4 Sequential Monte Carlo Method

The recursive solution based on online Sequential Monte Carlo sampling technique is formulated independent of the distribution to find a solution for developed self-organizing state space models. Contrary to conventional solutions, the developed solution builds the underlying distributions at run-time by learning the execution time statistics at the beginning of each interval. The reader is referred to [DFG+2001] for the understanding of the formulation of online Monte Carlo method and its variants. The online or recursive solution means that only the previous states and the current measured innovation are needed to compute the forecast, just like the EKF solution developed in Chapter 3. Contrary to batch estimation techniques, no history of previous forecast and execution time measurements is required. The recursive solution by Monte Carlo comprises of two steps: (i) Correct (measurement update) and (ii) Predict (time update). The correction step is carried out on the arrival of the new observation vector, as observation vector comprises of errors between previously forecasted and currently measured execution times. After the correction step, the new statistics and the dynamics of execution times, till the current interval, are known and the estimation step is performed to compute the forecasts for the next interval. Figure 4.6 shows the correction/prediction recursion steps computed at the beginning of each scheduling interval.

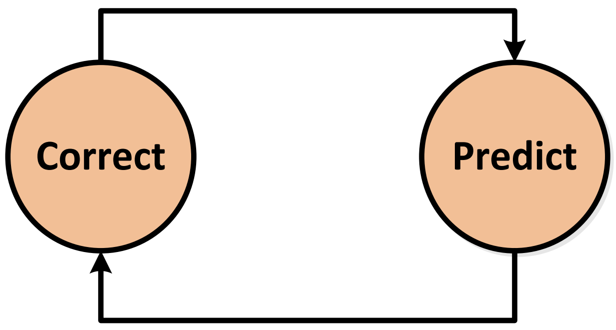


Figure 4.6: Recursion of correction and prediction steps

To derive the solution for the state estimation of self-organizing state space model consider the progression of individual state given by

$$x[k+1] = f[k](x[k], \varepsilon_x[k]) \tag{4.24}$$

Whereas $f[k]$ is a nonlinear function at an interval k (due to multiplication of time varying system matrix A_s and state vector X_s) in developed models. As mentioned in the last two sections that $\varepsilon_x[k]$ is the innovation or error between the forecasted execution time and the measured execution time. In the context of estimation, it will be referred to as a process noise. It is noteworthy to mention that the recursive solution by SMC solves the states in a seamless way so that the solution developed for the single state is directly applicable to an arbitrary number of states. Keeping this fact in mind, the derivation of SMC is carried out by considering a single state, which makes the solution relatively simple to understand and eliminate the need of complex multidimensional time and state indexes. Now for the state estimation, the objective is to minimize the process noise on the arrival of new observation and estimate $x[k+1]$ is given by

$$z[k+1] = h[k+1](x[k+1], n[k+1]) \tag{4.25}$$

Whereas $h[k+1]$ is a nonlinear function and $n[k+1]$is the observation noise. As mentioned in Chapter 3, during the derivation of EKF that the workload is measured in terms of actual consumed cycles in computations and obtained by hardware sampling unit so that there will be no observation noise; measurements are discrete and have no error in terms of censor and floating point accuracy. So, in the derivation of SMC recursive solution, the process noise is considered the only source of error. The objective is to find the estimate of $x[k+1]$ by considering the set of all previously available execution time observations, in summary

$$z[1:k] = \{z[1], z[2], \dots, z[k]\} \tag{4.26}$$

According to Bayesian estimation theory, the estimation of execution time recursively calculate the progression (evolution) of the state $x[k+1]$ at time k by considering all available measurements, $z[1:k]$, from first interval up to interval at time k. In that case, it is required that all the previous observations must be available to obtain the future estimate, a batch processing. The Bayesian estimation approach proposes the construction of posteriori probability density function of the state that utilizes all information including the complete set of previous measurements. The constructed pdf encompasses statistical information of the state, which in principle gives an optimal estimate of the state. To utilize this principle, it is required to construct the pdf $p(x[k]|z[1:k])$. At the beginning of the estimation process, measurements will not be available that means the measurement vector will be $z[0]$; without any measurement. Therefore, for a state vector the initial pdf at time 0 can be given by $p\big(x[0]|z[0] \equiv p(x[0])\big)$. Then, by the principle of probability induction, the recursive process of prediction and update can yield the required pdf $p(x[k+1]|z[1:k])$. Let's assume that this pdf is available then the task execution time forecast process can be categorized as estimation by utilizing all previous execution time measurements without actual requirement to store the history of the execution times. Figure 4.7 highlights this fact that the forecast of the future interval is the function of previous intervals, starting from the first interval to the current interval. Till this point the pdf $p(x[k+1]|z[1:k])$ is not available and it is required to construct it online on the arrival of new execution time measurement. The online step-by-step construction of the pdf with correction on the availability of a new execution time (measurement) can be viewed as on line learning. This learning process consumes first few observations in the construction of statistical structure of the process because of which forecasts obtained in the beginning are not meaningful and it is known as convergence time.

To construct the pdf of the state from the measurements, consider the solution by the principle of mathematical induction. Suppose that the pdf $p(x[k]|z[1:k])$ at time k is available. In the prediction stage Chapman Kolmogorov equation can be used to obtain the prior pdf by considering the (4.24) and can be given as

$$p(x[k+1]|z[1:k]) = \int p(x[k+1]|x[k])p(x[k]|z[1:k])\mathrm{dx[k]} \qquad (4.27$$

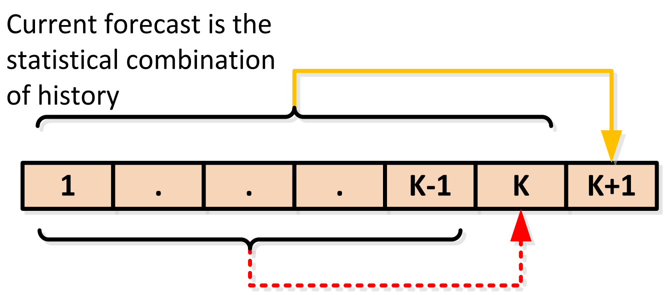


Figure 4.7: Learning at each interval and forecast as a combination of history

The (4.27) assumes that the process under consideration is first order Markov process and $p(x[k+1]|x[k], z[1:k]) = p(x[k+1]|x[k])$. On the arrival of new measurement $z[k]$ at time k, using the bayes rule the prior pdf can be updated by

$$p(x[k+1]|z[1:k]) = \frac{p(z[k]|x[k])p(x[k]|z[1:k])}{p(z[k]|z[1:k-1])} \tag{4.28}$$

where as the normalizing factor can be obtained by

$$p(z[k]|z[1:k-1]) = \int p(z[k]|x[k])p(x[k]|z[1:k-1])dx[k] \tag{4.29}$$

The (4.29) is defined by the measurement model. In the update equation (4.28) the posteriori density is obtained by the prior density after the availability of new measured execution time. The developed posteriori and priori pdf solutions for the state statistics define the Bayesian estimation and lay down the foundation of framework.

To construct the pdf for all the state variables at run time, consider posteriori pdf $p(x[0:k]|z[0:k])$ which is characterized by the points in the space $x[0:k]$, which represents the complete set of states associated with state variable x. Now define the weight factors $l_i$associated with each state x_i where $i = 1 \dots N$(the total number of samples to emulate pdf). If associated weights satisfy the condition of normality $\sum_i l_i = 1$, then the posteriori density can be approximated by the following equality and is a discrete weighted approximation to the true posteriori pdf

$$p(x[0:k]|z[0:k]) \approx \sum_{i=1}^{N} l_i[k]\delta(x[0:k] - x_i[0:k]) \tag{4.30}$$

In the above equation δ is the dirac delta function. The true approximation of the pdf is subject to the selection of appropriate weights and can be done by calculating the importance sampling [B1999, D1998]. The fundamental principle is that for a given probability density (x) , if it is difficult to draw samples from it but it is possible to compute the proportional probability density $\pi(x)$ then it holds that $p(x) \propto \pi(x)$. In such case $\pi(x)$ is called instrumental pdf, it is not the true pdf of the process but an approximation generated by the principle of sampling. In addition to it let $x_i \sim q(x), i = 1, \dots, N$ are the sample drawn from the proposal $q(.)$ then the weighted approximation is given by

$$p(x) \approx \sum_{i=1}^{N} l_i \delta(x - x_i) \tag{4.31}$$

In (4.31) the normalized weights are the ratio of instrumental pdf to proposal pdf, given by

$$l_i \propto \frac{\pi(x_i)}{q(x_i)} \tag{4.32}$$

In the problem at hand the samples of the distribution $x_i[0{:}k]$ can only be drawn from the importance or proposal density then the weights of the (4.32) can be defined by the

$$l_i \propto \frac{p(x_i[0{:}k]|z[1{:}k])}{q(x_i[0{:}k]|z[1{:}k])} \tag{4.33}$$

On the arrival of each new execution time measurement the importance density must be updated, to achieve the seamless update of importance density factorize it as

$$q(x[0{:}k]|z[1{:}k]) = q(x[k]|x[0{:}k-1], z[1{:}k])q(x[0{:}k-1]|z[1{:}k-1] \tag{4.34}$$

In the factored form (4.34) the importance density can be updated by augmenting each of the existing samples with newly obtained states. To derive the weight update equation we have,

$$p(x[0{:}k]|z[1{:}k]) = \frac{p(z[0{:}k]|x[0{:}k]|z[1{:}k])\, p(x[0{:}k]|z[1{:}k-1])}{p(z[k]|z[1{:}k-1])} \tag{4.35}$$

$$= \frac{p(z[k]|x[k])\, p(x[k]|x[k-1])}{p(z[k]|z[1{:}k-1])} \times p(x[0{:}k-1]|z[1{:}k-1])$$

$$\propto p(z[k]|x[k])\, p(x[k]|x[k-1])p(x[0{:}k-1]|z[1{:}k-1]) \tag{4.36}$$

Finally by substituting (4.34) and (4.36) into (4.33), finally the weight update equation takes the form

$$l_i[k] \propto \frac{p(z[k]|x_i[k])p(x_i[k]|x_i[k-1])}{q(x_i[k]|x_i[0:k], z[1:k])} \tag{4.37}$$

$$l_i[k] = l_i[k-1]\frac{p(z[k]|x_i[k])p(x_i[k]|x_i[k-1])}{q(x_i[k]|x_i[0:k], z[1:k])} \tag{4.38}$$

It is evident that the importance density is only dependent on previous state and current observation. This fact eliminates the need of previous measurements and the record of evolution of the state. Hence, the (4.38) will take the form

$$l_i[k] = l_i[k-1]\frac{p(z[k]|x_i[k])p(x_i[k]|x_i[k-1])}{q(x_i[k]|x_i[k-1], z[k])} \tag{4.39}$$

and the posterior density will be approximated as

$$p(x[k]|z[1:k]) \approx \sum_{i=1}^{N} l_i[k]\delta(x[k] - x_i[k]) \tag{4.40}$$

From the defined weights and posteriori probability it is clear that as the number of samples (N) approaches infinity the posteriori density approximation approaches the true posteriori density.

4.4.1 Degeneracy

The developed recursive solution has a tendency of degeneracy of the samples. After a certain number of iterations, almost all the weights$l_i[k]$ tend to 0 and the instrumental pdf will be dominated by the few non zero weights. The work done in [D1998] shows that the variance of the weights can only be increased monotonically while degeneracy will always occur as the iterations proceed in time. This means that beside the degradation of estimation quality, weights with no or negligible contribution to the pdf, will be computed. The problem of degeneracy can be avoided by defining the suitable number of weights, required to approximate the system statistics, and is described in [B1999, LC1998]. Lets define the suitable number of weights required to approximate the system dynamics to be N_e , which can be found by

$$N_e = \frac{N_s}{1 + Var(l_i[k])} \tag{4.41}$$

The solution of (4.41) requires the true weights of the statistics, which are not available in case of approximate pdf. The solution in [B1999], proposes the use of the estimate given by

$$N_e = \frac{1}{\sum_{i=1}^{N_s}(l_i[k])^2} \tag{4.42}$$

The (4.42) implies that $N_e \leq N_s$and indicates the degeneracy of the solution, which is undesirable effect. The use of extremely large sample set can mitigate the degeneracy, but it is not practical due to computational costs and memory requirements. The solution to circumvent the degeneracy, it is proposed that [D1998] by selecting the good importance samples and replacing the samples having weights near to zero. This requires the construction of suboptimal approximation to the optimal importance density which approximate the Gaussian by choosing the importance density to be prior density

$$q(x[k]|x_i[k-1], z[k]) = p(x[k]|x_i[k-1]) \tag{4.43}$$

Substitution of (4.43) into (4.37) yields

$$l_i[k] \propto l_i[k-1]p(z[k]|x_i[k]) \tag{4.44}$$

The (4.44) is the most fundamental choice for the importance density and is practical choice because of its simplicity and ease of use. After the availability of approximate density, samples can be replaced whenever N_e fall below the predetermined threshold. The idea is to replace the samples having negligible weights with the samples having weights nearly equal to the large weight samples. This is essentially replenishing the samples by generating a new sample set N_s from an approximate discrete representation of $p(x[k]|z[1\colon k])$ and is given by

$$p(x[k]|z[1\colon k]) = \sum_{i=1}^{N_s} l_i[k]\delta(x[k] - x_i[k]) \tag{4.45}$$

The newly obtained samples are independent and identically distributed, and are drawn from (4.45), which after normalization $l_i = \frac{1}{N_s}$ approximate the prior density. The resembling procedure can be programmed by order statistics and require $O(N_s)$ operations to solve N_s samples [R1987]. The Figure 4.8 shows the flow chart of the solution for developed self-organizing state space model by SMC method.

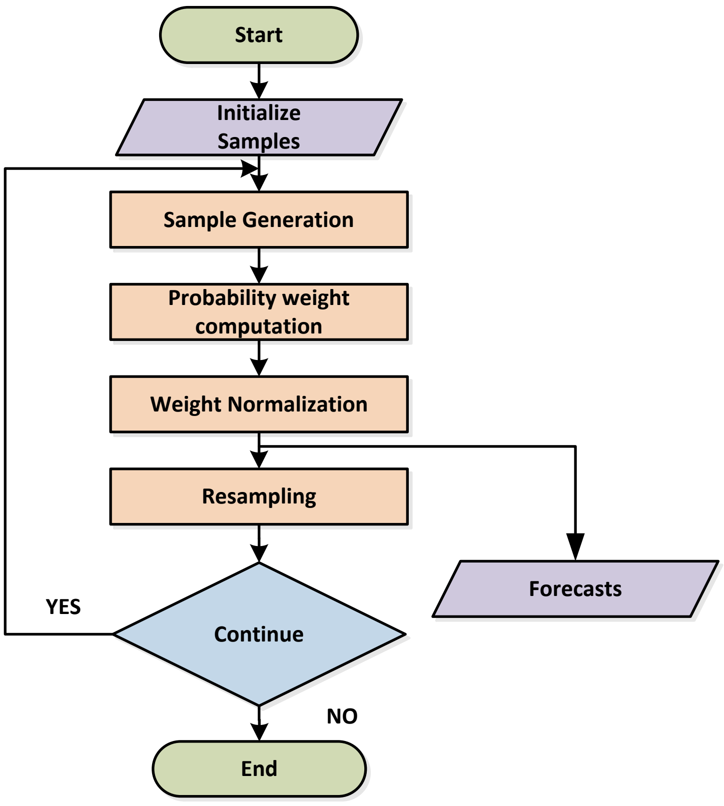


Figure 4.8: Flow chart of SMC recursive solution

4.5 Stochastic Execution Time Scheduling (SET)

The estimation process yields the task execution times due to which the stochastic behavior will vary from one scheduling interval to another scheduling interval. To schedule the task with stochastic execution time, it is required to incorporate it in the scheduling decisions. The obtained execution time forecast completes the scheduling tuple (S_t E_t D_t) and tasks can be scheduled by any scheduling scheme. Earliest Deadline First (EDF) is an optimal scheduler for uni-processor real time systems and is widely used for multicore systems. EDF scheduler primarily works on laxity, the measure of the spare time permitted for the task before it misses its deadline. In case of stochastic execution times, the laxity will be the function of deadline and forecasted execution time. The variable laxity at the beginning of each scheduling interval can be computed by laxity = (D_t - S_t). Because of variable laxity and multicore platform any variable laxity EDF scheduler [KS2009] can be employed to obtain the task schedule.

4.6 Soft Error Aware Low Power scheduling (SEAL)

The monotonic increase in Soft Error Rate with evolving fabrication technology makes it a matter of concern for the reliable operation of the computing system. To detect the soft errors at run time applications incorporate checkpoint and rollback recovery mechanism [NIS+2011, NIS+2010, AVS2008]. In check pointing and roll back mechanism, a task is divided into a number of execution segments and checkpoints are inserted on the boundaries of consecutive execution segments. To implement checkpoint and rollback mechanism at least dual redundancy is required. To achieve the redundancy either task is executed on two processors in parallel or it is executed two times sequentially on the same processor. During the execution of a task when a checkpoint is reached, the results and context from processors are gathered for comparison. In case of no difference in obtained results and context at checkpoint, it is deduced that no error has occurred and computations proceed further to next execution segment. The difference in results and context at checkpoint reveals that error has occurred in the current executing segment (error detection). In case of error at checkpoint, the application recovery procedure is invoked. The recovery procedure roll back the execution to previously known healthy checkpoint and erroneous execution segmented is executed again. There is a tradeoff between the total number of checkpoints and execution time overhead. Coarsely placed checkpoints will reduce the checkpoint overhead but increase the time spent in re-execution due to delayed detection of error. On the other hand, fine-grained checkpoints will increase the task completion time due to checkpoint evaluation overhead. The checkpoint and roll back mechanism is a vast subject and many practical implementations are available. The work done in this thesis assumes that the checkpoint and the rollback mechanism are available in tasks, and checkpoints are uniformly distributed. The work done in this thesis assumes that profiling analysis has already yielded the optimal checkpoint interval time and no further analysis is required for fine tuning. Hence, the work presented here is concerned only with the impact of power management scheme on system reliability.

In the design of real time systems with check pointing and roll back mechanism, the SER is assumed to be constant. In general, the SER rate at the highest processor frequency is used as a design metric. This assumption of constant SER does not hold in the presence of DVFS scheme. Beside other factors, the occurrence of soft error is linked to DVFS when processor operates at reduced speed to save energy. The two factors contribute to increase in the effective soft rate at low operating speed(s) which are (a) reduced voltage means less charge on gate which results in

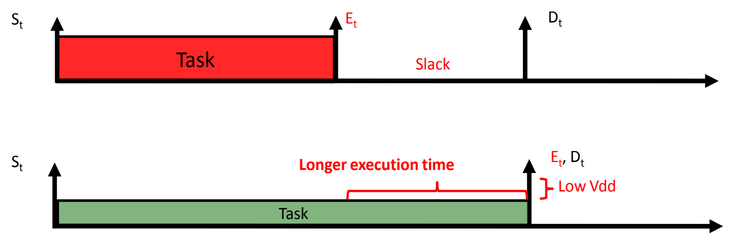


Figure 4.9 : Impact of DVFS on soft error

increased soft error (b) The increased execution time due to low speed increase the task pruning to soft error.

Figure 4.9 shows the influence of these two factors when processor's frequency is set lower than the highest available frequency. Hence, the solutions employed to improve the performance-to-power tradeoff must incorporate ways to cater for the soft errors as well. To maintain the system reliability constant regardless of operating frequency, it is required to derive the effective SER for lower operating frequencies. To proceed further consider the relation of SER at the highest frequency to the target frequency levels, given by[STA+2002],

$$SER_{tf} = SER\ 10^{d(1-f)/1-f_{min}} \tag{4.46}$$

where SER_{tf} is the error rate of the targeted frequency level, f is the highest frequency, $f_{min}$the minimum frequency and d is a constant to represent the sensitivity of fault rates to frequency scaling. The probability of an error for a given task Ti is the function of its execution time and SER_{tf} [6]; it is given by

$$p_{tf} = 1 - e^{SER_{tf}\, y_{Ti}} \tag{4.47}$$

where y_{T1} is the estimated execution time and the target frequency can be the highest frequency level of the processor. Then the additional time required by a task to maintain the constant system failure probability is given by

$$\Delta y_{Ti} = (p_{lf}/p_f)\, y_{Ti} \tag{4.48}$$

where, $p_f$is the probability of failure at the highest frequency level. Assuming the uniform distribution of the checkpoints at intervals of ΔT (corresponding to the highest frequency), then the extra time, for rollback(s) is given by

$$TC_{tf} = \lceil \Delta y_{Ti}/\Delta T \rceil \Delta T \tag{4.49}$$

TC_{tf} is the extra time which must be added to the estimated time to have the constant probability of system failure. Hence, from (4.49) the actual execution time of the task for the targeted frequency is given by

$$\mathrm{E}_{tf,Ti} = y_{Ti} + TC_{tf} \tag{4.50}$$

As described in the previous section EDF is common scheduling scheme for multiocre systems and is also used for low power scheduling. The function of EDF requires slack time to obtain the task for execution. In the case of SER aware low power scheduling, from (4.50) on arrival of the task slack time can be computed by: slack= $\mathrm{D} - \mathrm{E}_{tf,Ti}$, where D is the task's deadline. To schedule the task for minimum energy while respecting the task deadline, the scheduler attempts to offer an operating voltage as low as possible such that the task finishes just on its deadline. Given the slack, the voltage selection becomes a trivial case [ZM2004]. The selection of voltage frequency pair is carried out by satisfying the following condition

$$\sum_{k=1}^{n} 2(\hat{y}_{Ti})/T_i \leq 1 \tag{4.51}$$

where $\hat{y}_{Ti}$ and T_i are the forecasted execution and inter arrival time of a task respectively. The (4.46) is the modified schedulability test for EDF, in whichWCET is replaced by the forecasted execution time and considering double execution of each task. Figure 4.10 shows the two instances of a task with check point and error recovery mechanism. It is shown that processor must have to operate at higher frequency to cater the extra time to keep the system reliability constant.

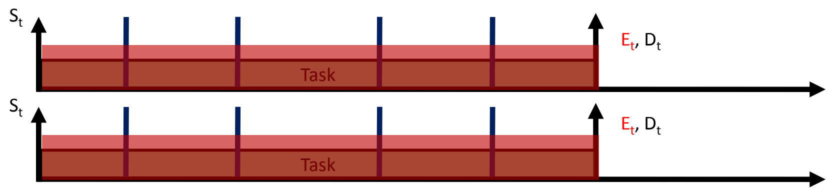


Figure 4.10 : Impact of DVFS on soft error

Chapter 5 Experimental Setup and Results

The previous chapters highlight the fact that the performance throttling is an essential component of modern day systems and discuss the proposed novel schemes in detail. The main focus of the prior research for performance throttling is to utilize the application information or prior knowledge to improve the quality of workload forecast. In recent years, researchers developed dynamically adaptable run time mechanisms for power management. In most of the cases, such schemes made up assumptions about the application, workload and run time conditions of the system. In case of violation of underlying assumptions, the performance of dynamically adaptable schemes degrades drastically and may lead to the undefined system state, which in result cause more energy and performance penalties. The fundamental principle behind all dynamically adaptable power management schemes is the utilization of temporal dependencies, i.e. the future workloads conditions depend on the recently observed workloads. However, the advent of multicore systems and massively parallel applications give rise to the strong spatial dependencies. The workload estimation under the influence of temporal and spatial dependencies is the main concern of this thesis. The power management schemes discussed in the previous chapters take into account the temporal and spatial dependencies and do not assume any a priori knowledge of the application, system or workload conditions. Hence, the proposed schemes are truly dynamically adaptable power management schemes and are most suited for the practical scenarios. The stochastic modelling and development of the problem tries to find the best suitable solution in probabilistic terms. Moreover, the soft error aware power management scheme proposed in the previous chapter caters for the futuristic problems.

This chapter fetches the performance throttling schemes of previous chapters and extends these schemes to a complete dynamically adaptable power management framework. The proposed schemes in the previous chapter fully utilize the temporal and spatial dependencies among the cores and software pipelines tasks. The periodic performance throttling scheme discussed in Chapter 3 is the foundation scheme. The modelling of the workload of the individual core as a multivariate joint state space model not only cater for the temporal dependencies but also consider the spatial (inter core) workload dependencies. The further derivation of the state space model to joint state space modelling allows the run time estimation of the model parameters in conjunction to the workload forecast. This alleviates many constraints associated with the model estimation. The recursive solution by EKF allows the non linear estimation of the workload and model parameters. The scheme proposed in Chapter 3 is remarkably general and do not take into account the deadlines of the individual tasks. The low power scheduling schemes for soft real time systems are discussed in Chapter 4. In addition to the temporal and spatial dependencies, the developed scheme also considers the phase change of the task and the recursive solution is found by the Sequential Monte Carlo method. Furthermore, the futuristic concerns of soft errors in the nanometre technologies also catered, while taking the scheduling decisions. The soft error aware low power scheduling scheme is developed assuming that the check pointing and roll back mechanism is available. This chapter establishes the effectiveness of the developed performance throttling schemes by detailing the benchmarks, experimental setup, and comparison of obtained results with state of the art approaches.

5.1 Experimental Setup

The evaluation platform is based on Intel® Quad core (Q9100) processor having 4GB of memory and Front Side Bus (FSB) speed of 1.33 GHz. The Operating System (OS) is Windows® XP which support Intel Enhanced Speed Step technology. Table 5.1 highlights the general specification of Q9100 with multitude of power management states [INT2008]. According to ACPI specifications, the processor states are broadly classified as powered states and sleep states. The Figure 5.1 shows the power state (C state) transition diagram of an individual core. The sleep and deep sleep states are not the concern of this thesis and the reader is referred to [INT2008] for further details. In the context of this thesis, we are concerned with the operating states of the processor encapsulated by the global operating state C0. The sub states of C0 are called low overhead active power management states termed as P or DVFS states.

In Intel architecture, P states are implemented by Intel SpeedStep® technology which provides the liberty to set the multiple voltages and frequency operating points and legitimate values can be selected from the voltage frequency tables. The Q9100 have 7 voltage identification pins (VID pins) for the automatic selection of power supply voltage. As a result, dozens of legitimate permutations of supply voltages and frequency are available. The voltage and frequency selection is software controlled and can be changed by configuring the MSR registers of the processor. The processor adjusts itself automatically to the newly applied voltages or the frequency by the following principles. In case of higher requested frequency than the current operating frequency, the applied voltages are increased in steps by placing new values in the VID pins and then PLL locks to the new frequency. In the other case, if requested frequency

Table 5.1: Intel® Quad Core Processor Q9100 specifications

Available cores	4	Core Voltage	0.9V – 1.25 V
Clock speed	2.26 GHz	Operating Temperature	0°C - 100°C
L1 Cache	4 x 32 KB Instruction 4 x 32 KB Data	ACPI States	Arbitrary P States 4 C States Stop Grant mode Sleep mode Deep Sleep mode Deeper Sleep mode
L2 Cache	12 MB (Shared)		
Bus to core ratio	8.6		
FSB Speed	1333 MHz		
FSB Parity	Not available	Floating Point Unit	Integrated
Instruction Set	64 Bit	Minimum Power	16.54 Watt
Embedded Options	Not available	Maximum Power	74.35 Watt
Fabrication Technology	45 nm	Technologies	SSE1 - SSE4 Supplemental SSE3 Trusted Execution Enhanced SpeedStep
Thermal Design Power (TDP)	45 W (Maximum)		

Table 5.2: P state details

Voltage (V)	Power (W)	Frequency (GHz)
1.25	74	2.26
1.13	62	1.9
1.0	51	1.7
0.9	39	1.6

is lower than the current operating frequency then PLL locks to the new frequency and then applied voltages are changed through VDD pins.

There are no constraints in applying the frequency transition and voltages as these are controlled internally for glitch free transition. Though many permutations of frequency and voltage pairs are possible, but for fair evaluation only coarsely distributed discrete frequency and voltage pairs are considered in the simulation. Table 5.2 shows the 4 discrete frequency and voltage pairs used in the evaluation. During the experiments, it is assumed that on chip regulators are available because of which state transition has negligible latency.

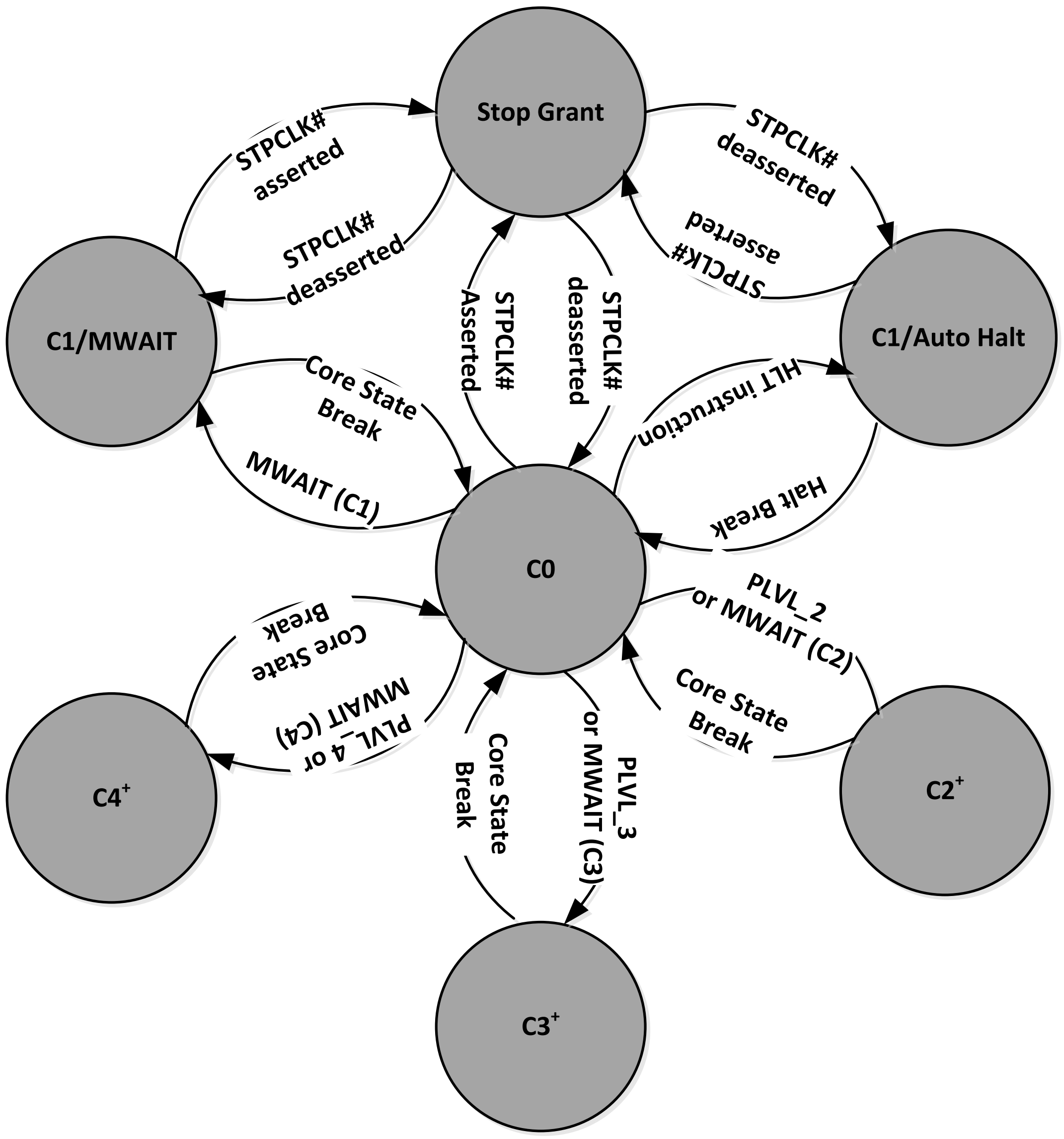


Figure 5.1 Power management states of Q9100 processor

The source codes of the benchmarks (discussed in next section) are compiled using Microsoft Visual C++ compiler configured using Visual C++ environment (IDE). The benchmarks were compiled with the optimization enabled by complier switches Og, Ot, Oy, Gs and Gy. These switches performs all the general code optimizations and include copy propagation, dead store elimination, common sub-expression elimination, register allocation, function in-lining, loop optimizations, flow graph optimizations, peephole optimizations, instruction scheduling, and stack packing. It does not do loop unrolling, although only for few small special cases such as block memory moves. It means that compiler is configured to produce a code for maximizing the speed of the application. The individual selection of compiler optimisation switches makes it possible to keep some source code information in the binaries e.g. function names, source code to instruction mapping etc. Beside standard optimization no other fine tuning is carried out to affect the performance of the applications. To avoid the undesired function inlining, Pragmas are added in the code to instruct the compiler to disable the function embedding for specified functions.

Intel's VTune® performance analyzer [V2009] is used to collect the execution data of the benchmarks. VTune supports simultaneous sampling of multiple events by employing hardware counter monitors. It can be configured to collect the execution data of required events. Though VTune supports both time based and event based sampling, it was configured to collect performance data using event and time based sampling. Since VTune has a low intrusion, the collected data provide a clear representation of an application's actual execution metrics. VTune was configured to collect the following events: clockticks, instructions retired, loads retired, stores retired and branches retired. The precise hardware sampling of the performance unit and Vtune left no room of error in collected data and can be considered as an exact measurement. The continuous epochs of benchmarks are executed for longer data collection. Thread migration is disabled during data collection by defining the thread affinity so that threads remained mapped on a single core.

The performance throttling schemes developed in Chapter 3 and 4 are implemented in Matlab [MA2008] development platform, which is an excellent programming environment for rapid prototyping and implementation of complex systems. The motivation in Chapter 2 establishes that performance throttling constitute a close loop system which requires the close loop simulation to replicate the physical system. The open loop simulations are widely implemented to simulate the DVFS manager but are essentially inaccurate and unreliable. Open loop simulation cannot consider the changes in input workload due to performance penalty. Moreover, it cannot sense environmental or internal system changes due to DVFS manager decisions. Because of which open loop simulation may produce unreliable results. The close loop simulation of developed methods requires that the residual workload left after performance penalty must be added to the input workload. The DVFS manager has no knowledge of the residual workload and simulation framework has to take care of it. The simulation of developed scheme is carried considering the close loop system which makes obtained results equivalent to the results of actual system.

5.2 Bench Marks

Two different benchmarks with different characteristics are used to evaluate the performance throttling schemes developed in this thesis. The SPEC CPU2000 and H.264 video decoder are employed to evaluate the periodic performance throttling and low power scheduling respectively. The next two sections discuss the details of each benchmark.

5.2.1 SPEC CPU2000 Benchmark

To evaluate the periodic performance throttling on different voltage and frequency settings, SPEC CPU2000 benchmark is used [SPE2000]. Though the latest version of performance benchmark suit is SPEC CPU2006 but this benchmark is tailored for the performance evaluation of CPU and memory and leaves little room for performance throttling; its most applications are CPU bound. On the other side, SPEC CPU2000 comprises of the applications, which are CPU as well as I/O bound and is suitable for evaluations of the periodic performance-throttling scheme developed in this thesis. In total 10 different applications included in SPEC CPU2000 are used for the evaluation purposes. Table 5.3 lists the application by name with their workload characteristics and are classified in three categories according to workload behavior. The applications exhibiting varying workload conditions are most suitable candidates for the evaluation of performance throttling scheme. The applications with low and high workload conditions are included to have the rigorous and fair evaluation comparable the real world scenario.

Table 5.3 : Workload characteristics of SPEC CPU2000 applications

Workload	**Applications**			
Low	Galgel	eon.cook	applu	
High	bzip2.graphic	eon.kajiya	equake	
Low+High	Parser	eon.rushmeier	bzip2.program	mesa

5.2.2 H.264 Video Decoder

To evaluate the low power scheduling schemes the H.264 video decoder is carefully selected against generic task graphs or basic applications. The control dominant and computation-intensive properties of H.264 decoder fit well for the benchmarking and evaluation of all points raised in the developed performance throttling scheme of Chapter 4. In multimedia, video encoding/decoding is the most compute intensive process. Many video compression standards are in use like MPEG-2, H.263. MJPEG-4, H.264 etc. The latest video codec from Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T Video Coding Expert Group (VCEG) is H.264 or MPEG-4 AVC part 10, which offers better compression as compared to other standards under the envelop of the same video fidelity[ITU2012]. The primary goals of H.264 are better compression, suitability for network transmission and error resilience. To achieve these goals, a wide variety of tools are included in the standard, which makes it computationally complex and predominantly control-flow oriented [MBS2004]. H.264 is widely accepted by industry and would likely be a leading video codec for mobile multimedia systems in the coming years, mainly due to its versatility and coding efficiency. The reader is referred to [WS2003, ITU2012] for a general overview or thorough understanding of the processing and tools involved in H.264.

The main advantage of H.264 standard is its increased performance in bit rate that owes to the inclusion of additional tools, which leads to an increased computational complexity in encoding as well as decoding. This increased complexity is a significant problem when it comes to devise a cost effective H.264/AVC-based video solutions [HJK+2003]. To meet the computational requirement of the H.264 decoder, many parallel approaches have been proposed for Multicore systems [CSC+2007, LVK2005, SSF2000, MAA+2008, VJG2003] utilizing from 2 cores to 60 cores. The common design metric in all these approaches is to map the entropy decoder on a single core, because entropy decoding is an inherently sequential process and cannot be parallelized. The proposed approaches offer time parallelized or space parallelized H.264 decoder but not both forms of parallelisms. In order to evaluate the modeling impact of spatial and temporal factors, it was imperative to develop the new parallelized H.264 decoder simultaneously incorporating time and space parallelism. This is achieved by parallelizing the JM13 [HHI2008] open source reference decoder. In the parallelized decoder, the space parallelization is achieved by enabling multiple slice decoding within a single frame and time parallelization is achieved by software pipelined MB decoding process. Figure 5.2 shows the task graph of decoder and Table 5.4 defines the functionality of each task.

It is evident from Figure 5.2 that output of entropy decoder (T1) is the input of remaining tasks. The performance of entropy decoding is fundamental to the throughput of the decoder and becomes the bottleneck in parallel decoder design [MAA+2008, CSC+2007]. This constraint

Table 5.4: Tasks and their functionality

Task	Functionality
T1	Network abstraction layer decoder
T2	Slice 1: Entropy decoder, Inverse Scaling/D-Quantizatiion and Inverse discrete cosine transform
T3	Slice 2: Entropy decoder, Inverse Scaling/D-Quantizatiion and Inverse discrete cosine transform
T4	Slice 1: Motion compensation and inloop filter
T5	Slice 2: Motion compensation and inloop filter
T6	Slice to frame merger (logical task)

requires further investigation for the efficient parallel implementation and is discussed in [NH2009]. In H.264, there are two kinds of entropy encoding schemes: the conventional Variable Length Coding (VLC) and Context Adaptive Variable Length Coding (CAVLC). VLC is based on Exp-Golumb codes; its decoding process is straightforward, and it is only being used to transmit the header data. On the other hand, CAVLC is used to encode actual video data and is a combination of entropy and ‘run length’ coding with a context-adaptive mechanism. The whole entropy encoding/decoding process is elaborated in [R2003]. The H.264 standard document defines the symbol tables for the CAVLC process and is fixed for all profiles which include coeff_token, total_zero, and run_before syntax elements for “luma” and “chroma” but do not define the decoding method. Sighting the bottleneck researchers proposed multitude of methods to reduce the entropy decoding time. For the purpose of self-sufficiency, a brief over view of efficient CAVLC decoding schemes is given.

The basic method is to parse the stream bit-by-bit to decode the symbol information by employing the Table Lookup Sequential Search (TLSS) and has been implemented by H.264 reference software [HHI2008]. The sophisticated method may take a chunk of bits e.g. 8 bits at a time to speed-up the decoding process. To mitigate the computation cost, Table Lookup by Binary Search (TLBS) is proposed in [KYS+2006]. In TLBS method, table accesses involve random memory accesses and its speed-up gain may diminish with the increase in length of incoming symbol. To reduce the memory access of CAVLC decoding process, software

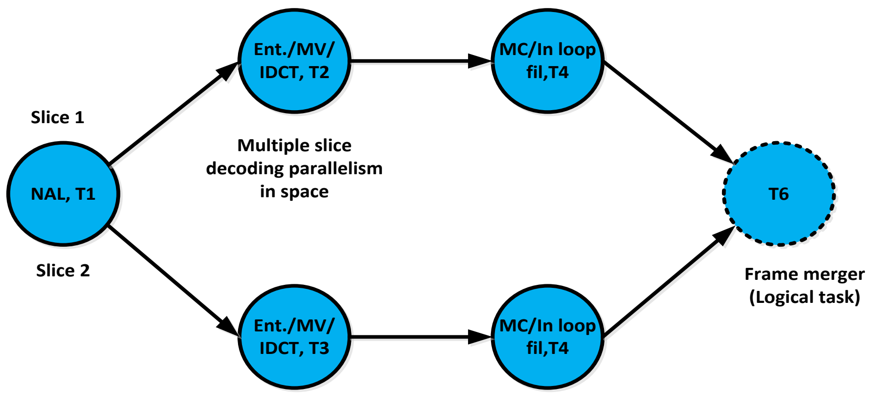


Figure 5.2: Slice parallel and pipelined parallel H.264 decoder

decoding for some components is proposed in [ML2008] and is known as Moon's method. The main idea is to use arithmetic operations for highly probable short VLC codes to reduce memory access, and use conventional TLSS method to decode symbols with low probability of occurrence. It assumes that 95% of the symbols will have length shorter than 9 bits, which is not valid in case of high fidelity video. Because of the limitation of state of the art approaches it was essential to device a new CAVLC decoding method for parallel implementation reflecting the real world scenarios.

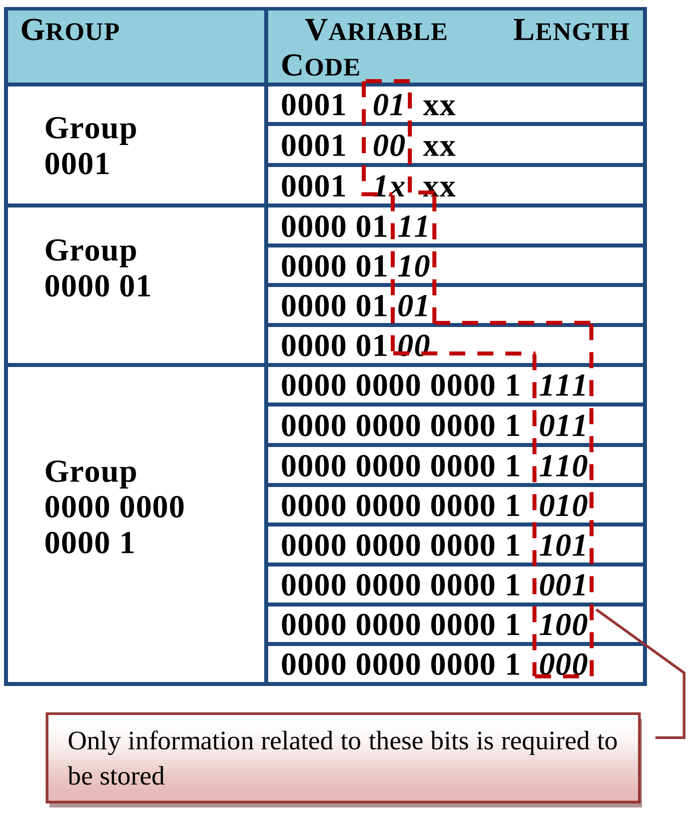
(a) Efficient Constant-time Entropy Decoding for H.264

In the course of this thesis to have a suitable benchmark for evaluation, a new way to construct CAVLC tables by exploiting the regularity and similarity among codeword symbols is developed [IH2009]. This regularity and similarity are not evident at first, and each entry should be evaluated manually. It gives many fold benefits, first decoding tables are small, secondly decoding algorithm is fairly straightforward and lastly the decoding time is independent of incoming symbol length (constant time). In this approach, the decoding table entries arranged according to the occurrence of regular pattern. The self grouping of code words is carried out on the basis number of leading zeros; the self grouping exists in H.264 tables with exceptions in total_zero table. After grouping, entries are sorted in ascending order using the occurrence of first '1' as sentinel. Based on this grouping decoding tables are derived and decoding algorithm is developed. Moreover, the grouping of the symbols yields very small decoding tables and straightforward decoding algorithm. The table building process comprises of three steps

- Arrange the decoding symbols in ascending order for leading zeros count.
- Arrange symbols, which have the same number of leading zeros in a same group, first column of Figure 5.3.

- After leading zero(s), the first occurrence of ‘1’ is a signal, which acts as a classification of group. The occurrence of first 1 after leading zero also acts as sentinel and information bits follows it. Here, afterwards leading zeros and first ‘1’ will be called prefix and remaining information bits as suffix. The sorting of the suffix bits and merging the information associated with the symbol yields the decoding tables with minimum memory requirements.

After grouping, the information that needs to be stored (for decoding the incoming symbol) comprises of a few bits and is shown in Figure 5.3 as a vertical group. Considering the example of Figure 5.3 full table require storage of 114 bits, but after grouping only the information related to 38 bits is required. The information bits occurring after the first 1 are few as compared to the total length of code, which significantly reduces the permutations, and hence memory requirements of tables.



GROUP	VARIABLE CODE	LENGTH
Group 0001	0001 *01* xx	
	0001 *00* xx	
	0001 *1x* xx	
Group 0000 01	0000 01 *11*	
	0000 01 *10*	
	0000 01 *01*	
	0000 01 *00*	
Group 0000 0000 0000 1	0000 0000 0000 1 *111*	
	0000 0000 0000 1 *011*	
	0000 0000 0000 1 *110*	
	0000 0000 0000 1 *010*	
	0000 0000 0000 1 *101*	
	0000 0000 0000 1 *001*	
	0000 0000 0000 1 *100*	
	0000 0000 0000 1 *000*	

Only information related to these bits is required to be stored

Figure 5.3: Table grouping process and information required

Table 5.5: The grouping for part of coeff_token table 0.

Code	Prefix	Suffix	[TC, T1]
0000 0100	0000 01	00	[6, 3]
0000 0101		01	[4, 2]
0000 0110		10	[3, 1]
0000 0111		11	[2, 0]
0000 0010 0	0000 001	00	[7, 3]
0000 0010 1		01	[5, 2]
0000 0011 0		10	[4, 1]
0000 0011 1		11	[3, 0]
0000 0000 0100 0	0000 0000 01	000	[8, 0]
0000 0000 0100 1		001	[9, 2]
0000 0000 0101 0		010	[8, 1]
0000 0000 0101 1		011	[7, 0]
0000 0000 0110 0		100	[10, 3]
0000 0000 0110 1		101	[8, 2]
0000 0000 0111 0		110	[7, 1]
0000 0000 0111 1		111	[6, 0]
0000 0000 0010 00	0000 0000 001	000	[12, 3]
0000 0000 0010 01		001	[11, 2]
0000 0000 0010 10		010	[10, 1]
0000 0000 0010 11		011	[10, 0]
0000 0000 0011 00		100	[11, 3]
0000 0000 0011 01		101	[10, 2]
0000 0000 0011 10		110	[9, 1]
0000 0000 0011 11		111	[9, 1]

Table 5.6: The grouping for part of total_zero table.

Code	Prefix	Suffix	[TC, T0]
100	1	00	[3, 1]
101		01	[3, 2]
110		10	[3, 3]
111		11	[3, 6]
0000 10	0000 1	0	[1, 7]
0000 11		1	[1, 8]
0000 010	0000 01	0	[1, 9]
0000 011		1	[1, 10]

The CAVLC tables are generated by grouping and sorting as described previously and the procedure is similar for each CAVLC table. In H.264, coeff_token syntax has four tables and selection of appropriate table depends on the context. The context table selection is based on the average number of non-zero transform coefficient levels of top and left 4x4 luma blocks, and it is -1 for chroma blocks. VLC table of the current block is adaptively selected according to neighboring blocks' non-zero transform coefficient levels.

To elaborate the method further, Table 5.5 and 5.6 show the grouped table with associated information. As CAVLC tables are large and it is not possible to show complete tables here, only small portions are shown for elaboration. Table 5.5 shows the part of coeff_token table 0, TC and T1 represent the total number of coefficient and the total number of ones respectively associated with the symbol. The table is divided into 14 groups and similarly coeff_token table 5.6 is divided into 10 groups, in this way method is seamlessly applicable to nearly all tables. There is an exception in total_zero table that a few entries cannot be merged and grouped because of irregularities. These entries are grouped separately, which in result require all permutations to keep the decoding algorithm simple. Similarly, all coeff_token, run_before, zero_left tables for luma and chroma tables are generated by the following the elaborated method. The total size of all CAVLC suffix tables is less than 8 KB, and it is highly likely that tables will be readily available in the cache due to frequent access and small size. The simplicity and regularity in the derived tables result in a simple decoding algorithm.

Algorithm 5.1: Pseudo code of coeff_token decoding routine

```
1:  code ← show_bits(16)  // read 16 bits from bitstream
2:   idx_one ← find_index_one(code) // using only one instruction
3:  code ← code >> 16 – (idx_one + 3)
4:  suffix ← code && 0x0007 //extract last three bits
5:  if(vlcnum == 0) { // first vlc table
6:       info ← coeff_token_suff_0[idx_one>>3 + suffix]  // access 32 bits info from table
7:       num_coeff ← info && 0xff
8:       num_trailing_one ← ( info >> 8) && 0xff
9:     code_length ← ( info >> 16) && 0xff
10:  }else if(vlcnum == 2) { // second vlc table
11:     info ← coeff_token_suff_1[idx_one>>3 + suffix]   // access 32 bits info from table
12:     num_coeff ← info && 0xff
13:     num_trailing_one ← ( info >> 8) && 0xff
14:     code_ length ← ( info >> 16) && 0xff
15:  }else { // Third vlc table
16:     info ← coeff_token_suff_2[idx_one>>3 + suffix]  // access 32 bits info from table
17:    num_coeff ← info && 0xff
18:    num_trailing_one ← ( info >> 8) && 0xff
19:    code_ length ← ( info >> 16) && 0xff
20:  }
21: frame_bitoffset ← frame_bitoffset + length
```

The Algorithm 5.1 shows the decoding of coeff_token syntax elements for luma and can be adapted for remaining syntax elements by changing the suffix table. The conditional statements in the algorithm are for context adaptation means selecting the different decoding table according to context. The decoding algorithm comprises of three steps, which include reading of bit-stream, classification of incoming symbol to its group and then decoding the suffix using suffix tables. In the first step, bits equal to the maximum size of the symbol in a given table are read from a bit stream, line 1 in the Algorithm 5.1. Hence, bit stream reading is carried out once for the decoding of single syntax element. In step 2 (line 2 of Algorithm 5.1), the codeword is classified into its group by counting the number of leading zeros. The counting of leading zeros is fairly fast as most of the modern processors include assembly instruction for this purpose, assembly instructions MSU and NSAU are examples in Intel x86 and Xtensa processors respectively. In the absence of leading zero count instruction, the number of leading zeros can be obtained in constant time using bit-twiddling algorithms [A2004]. Line 4 and 5 of Algorithm 5.1 extracts the information bit or suffix bits by bit shifting. Lastly, the suffix bits are used to extract the decoded information from decoding table, line 7 to 11. The remaining algorithm works in the similar way for different suffix table. It is evident that the decoding method is independent of codeword length and without any conditional statement, which results in constant decoding time for a coded symbol. Due to constant decoding time, the performance of the algorithm is scalable for high fidelity videos and parallel implementation of H.264 decoder.

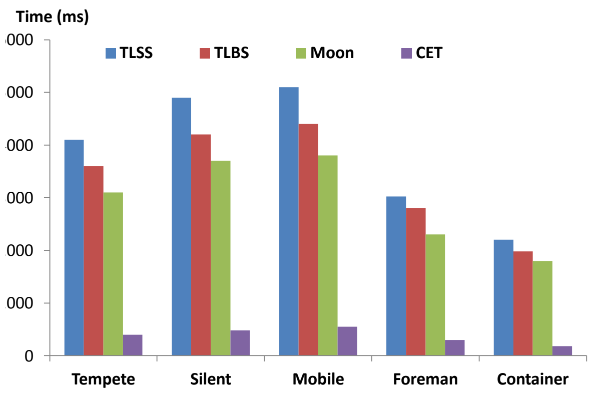
Table 5.7: Parameters of test sequences.

Sequence	Resolution	Frame rate	Frames
Tempete	CIF	30	60
Silent	CIF	25	60
Mobile	CIF	20	60
Foreman	CIF	35	60
Container	CIF	15	60

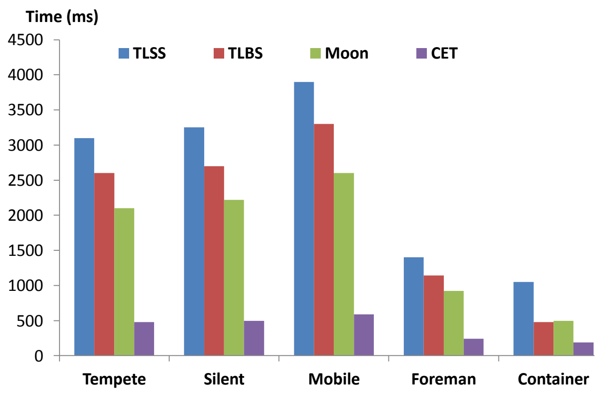
The evaluation is carried out with multiple test sequences given in Table 5.7. The diversity in test sequences aid in avoiding the influence of the system and data properties. The sequences are encoded with JM 13 and Table 5.8 shows the encoding parameters. Each sequence is encoded with four different QPs to highlight the scalability of the proposed method. Though baseline profile is used for encoding but proposed method is not limited to baseline, and conformance testing for Main profile is carried out successfully. Application binaries were generated with instrumentation code as required to obtain different performance metrics at runtime on an experimental platform.

Table 5.8: Encoding parameters of test sequences.

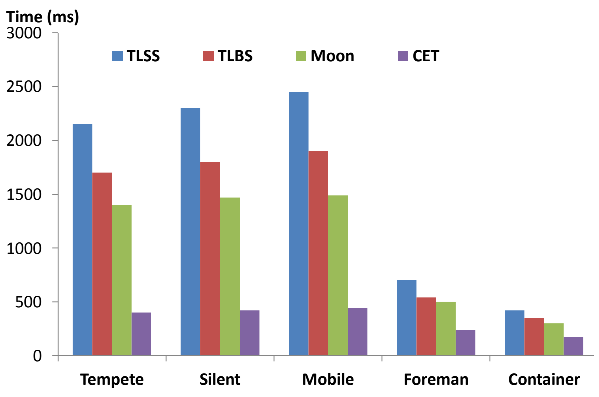
Parameter	Selected option
Profile	Baseline
RDO	On(fast algorithm)
MV search range	+/- 32 pixels
Reference frame	4
QP	16, 20, 24, 28
Motion search	Fast search
Intra interval	0
Encoder	JM 13



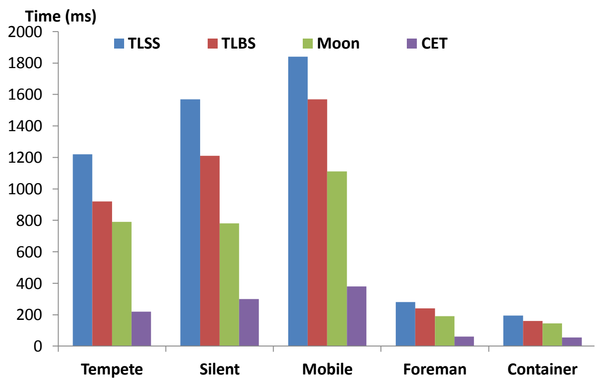
(a)



(b)



(c)



(d)

Figure 5.4 : CAVLC decoding execution time comparison

The comparison of developed CAVLC decoding method is carried out with TLSS, TLBS and Moon’s algorithms in term of execution time and memory access count. For ease of discussion term CET will refer to the developed method. Figure 5.4 shows the execution time comparison for different values of QP, and Figure 5.4 (a), (b), (c) and (d) corresponds to QP values 16, 20, 24 and 28 respectively. The comparison clearly shows that the CET is 7 times faster than TLSS and around 5 times faster than TLBS and Moon’s method. The better speed up is available for lower QP’s because of frequent longer length symbols and speeds up gains diminish with increase in QP. It can be inferred that the computational requirement of the CET increases smoothly with the decrease of QP, which makes it suitable for parallel implementation of H.264 decoder. In comparison to the memory access, the CET algorithm performs better compared to other algorithms, Figure 5.5. The memory accesses of CET are only 13% compared to TLSS and savings are around 20% more than TLBS and Moon’s method (QP =16). The memory access saving gain decreases monotonically and reaches Moon’s method, but CET algorithm uses small tables compared to Moon’s method.

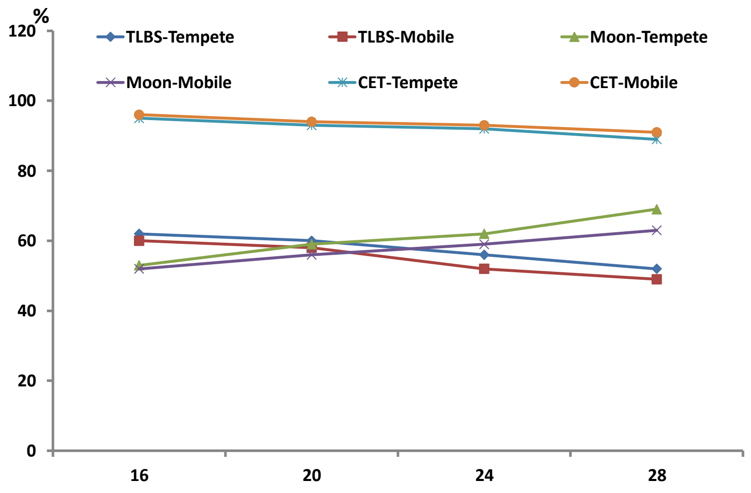


Figure 5.5: Memory access comparison

5.3 Results

5.3.1 Periodic DVFS

This section analyzes and compares the results of periodic performance throttling developed in Chapter 3. Before evaluating the performance using actual workload the evaluation of the state space model developed in Section 3.3 is carried out on a computer generated simulated workload. The simulated workload is generated by considering a time series representing the second order under damped linear time invariant system with constant model parameters. The additive zero-mean Gaussian noise is mixed in the generated time series to simulate the innovation process of real workload. The solution of joint state space is found by derived EKF for joint workload forecast and model parameter estimation. The Figure 5.6 shows the results, the upper part of the figure shows that most of the time forecast is closer to the actual workload. At the beginning of the process, forecasts have large error and are not meaningful because observer is in learning phase; EKF convergence. The lower part of the Figure 5.6 shows the comparison of actual parameters and jointly estimated parameters. The estimated parameters values fluctuate and deviate most of the time from actual parameter values because of uncertainties in the model and process noise. It is known that, there can be an infinite representation for a given state space model provided models are the linear combination of the basic models. Hence, parameter deviation is not important as far as forecast results are of acceptable quality. The results for the estimation of parameters also show that at the beginning there is more deviation in the model parameter values because EKF requires some iteration for the convergence.

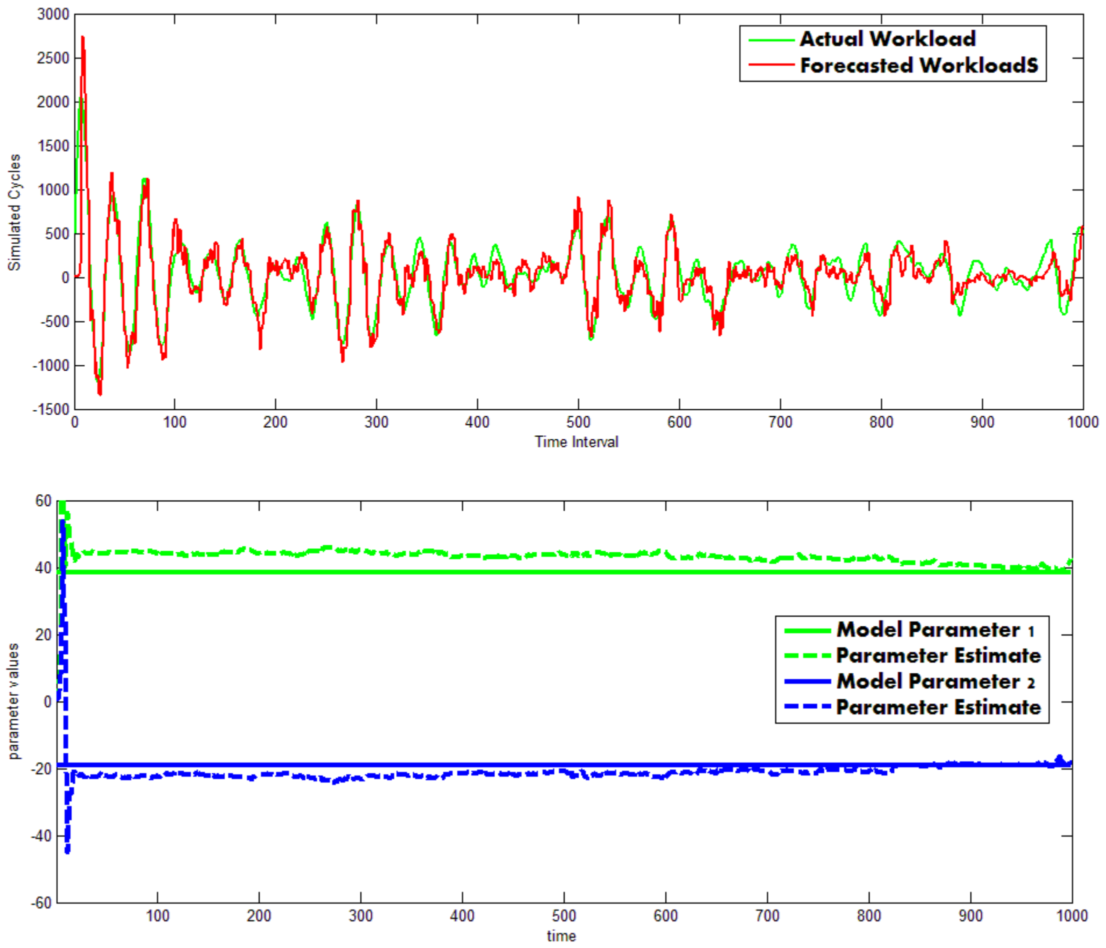


Figure 5.6: Joint workload forecast and parameter estimation

After establishing the correctness of joint workload forecast and parameter estimation, it is evaluated for real workloads of SPEC CPU2000 applications. The workload is generated by running the continuous epochs of SPEC CPU2000 applications at least for five minutes. Threads are locked to the cores throughout their span by disabling thread migration. The hardware sampling unit is configured to gather the cycle utilization and other sampling data at the continuous uniform interval of 100 ms. The pair of observer and regulator for each core are instantiated in the simulation and variables of the state space are initialized by random values drawn from the zero mean Gaussian process. The workload of first 50 intervals is used for the convergence of the EKF because of which forecast and regulator decisions of these intervals are not included in the results. Table 5.9 shows the summarized results of benchmarks for observer and regulator by setting $(U_r = 90\%)$. The second column of the table shows the absolute observer error and highlights the fact that most of the time it is bounded within 20%. It is noted that the percentage error is relatively large because of the fact that SPEC CPU2000 contains relatively small sized applications. Because of which the new instantiation of an application on a core cause abrupt change of phase due to the external dependencies such as I/O. The remaining two columns of Table 5.9 show the decision taken by the regulator. It appears that regulator is not able to compensate most of the forecast errors as these scenarios occurr at the frequency step boundary due to new instantiations.

Table 5.9: Forecast error and regulator decisions for Ur = 95%

Core	**Abs. Forecast error < 20% / %**	**Compensated by regulator / %**	**Wrong decisions / %**
SPEC CPU2000			
1	1972 / 66	485 / 16	306 / 9.8
2	2137 / 71	398 / 13.4	261 / 8
3	2495 / 83	427 / 10.7	173 / 5.8
4	1864 / 62	572 / 19	416 / 14

The forecast error cause either performance penalty or energy penalty. The performance penalty is incurred by under forecast and failure of the regulator to compensate the uncertainty. To overcome the performance penalty the reference utilization can be lowered. Surely, it will cause a power penalty and is a tradeoff between energy and performance. Figure 5.7 shows the

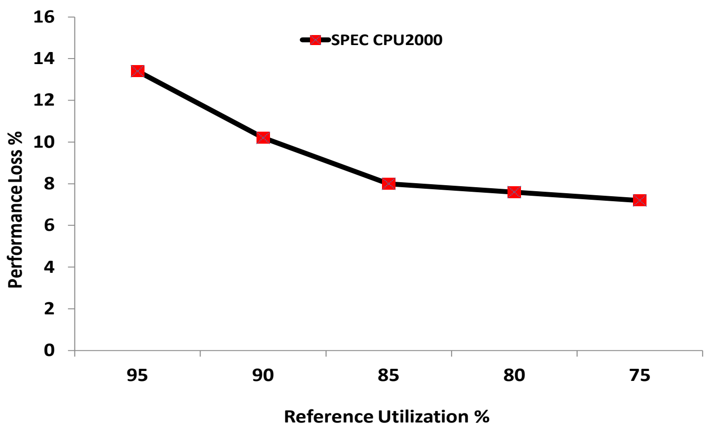


Figure 5.7 : Performance loss for different reference utilizations

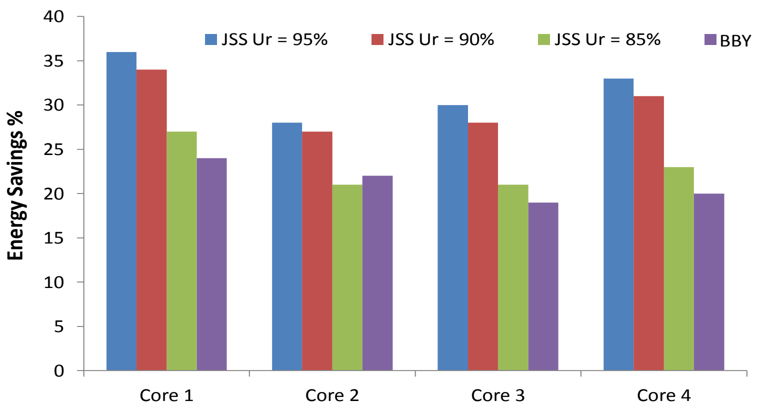


Figure 5.8 : Energy savings per core

average power performance tradeoff for the different U_r. Figure 5.7 highlights the fact that decrease in U_r cause monotonic decrease in the performance loss. The performance loss tends to slow down beyond 85% of U_r. In case of 85% of U_r the tolerance for forecast error is sufficient to avoid the performance penalty. The performance penalty persist even U_r decreases beyond 80%, because of the fact that the workload is going through an abrupt phase changes and observer is not able to fully grasp the dynamics.

The comparison of the periodic performance throttling is carried out with closely resembled state of the art schemes proposed in [BBY$^+$2009]. In [BBY$^+$2009] workload is modeled as a linear regression process which is formulated in a linear state space model. The parameters of state space model are initialized with equal and normalized weights, this generalized model may hamper the forecast quality. Moreover, the linear state space model is solved by employing the Kalman filter because of which only linear estimation is possible. Moreover, in [BBY⁺2009] workload forecast is considered as a temporal process and do not consider spatial dependencies. This formulates the SISO process, which means that each core requires an independent observer.

To compare the results and ease of analysis, let's name the periodic performance throttling as "JSS" and and approach proposed in [BBY⁺2009] as "BBY". The aggregated energy savings for both approaches and individual cores are shown in Figure 5.8. The JSS achieves 36% energy savings on Core1 compared to no power management scheme. The energy savings vary among cores due to the fact that each core execute the different set of applications. The average total energy savings of all four cores by JSS are more than 31 % , which shows the significance of power management scheme. The results clearly show that JSS outperforms state of the art in terms of energy, even when the maximum utilization is 85%. The energy savings by JSS slightly degrades in comparison to BBY for core 2 when utilization is set to 85%. The energy savings on Core4 are best case scenario while achieving 43% energy savings $(U_r = 95\%)$ compared to BBY. While considering the highest processor utilization of 95%, on average JSS save 38% more energy compared to BBY. Only energy savings are not sufficient to declare the superiority of power management scheme. The power management scheme has to choose the suitable tradeoff between energy savings and performance loss. Hence, it is necessary that performance loss must also be analyzed in conjunction to energy savings. The major advantage of JSS is clear from Figure 5.9, which shows the performance loss per core. The JSS performance loss is negligible

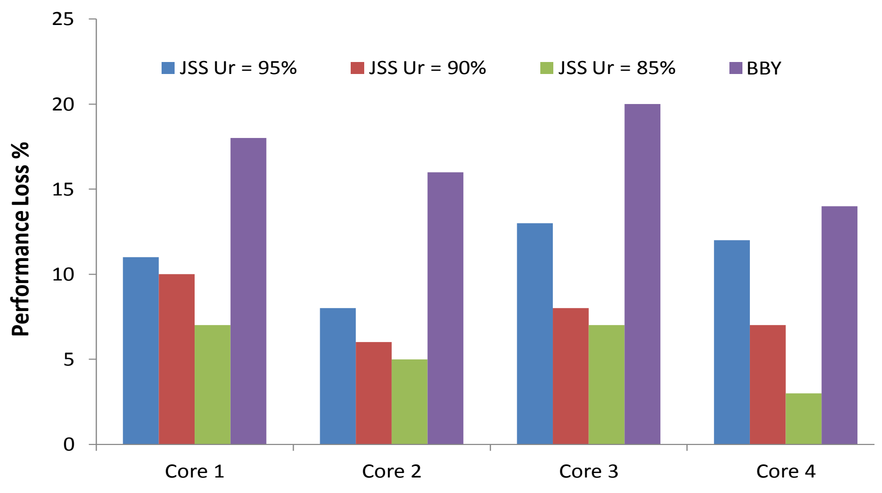


Figure 5.9 : Performance loss per core

compared to BBY, primarily because of quality workload forecast. It can be clearly observed that the performance loss by JSS is 54% lower than BBY. The energy savings and performance loss results clearly show the superiority of JSS because it achieves better tradeoffs between energy and performance. The fundamental reasons for achieving better tradeoff is the multivariate state space base modeling considering temporal and spatial dependencies. Another factor in quality workload estimation is the recursive solution by EKF, which allows non stationary/non-linear workload forecasts. Moreover, the EKF solution try to capture the source of error, the un modeled factors, by indirectly estimating their influence on the workload conditions.

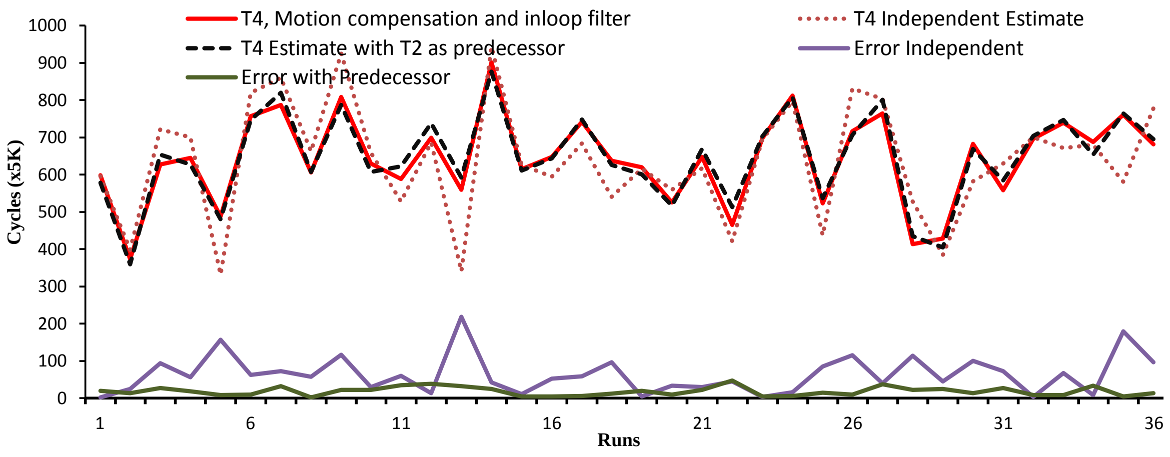


Figure 5.10 : Execution time forecast with and without predecessor task

5.3.2 Stochastic Low Power Scheduling

To evaluate the low power scheduling scheme developed in Chapter 4, parallelized H.264 video decoder developed in Section 5.2.2 is used. First, consider the inclusion of spatial dependencies on execution time model and its impact on forecast (Section 4.2 and 4.5). Figure 5.10 shows the execution time forecast of T4 (motion compensation and in-loop filtering for slice 1) in two different scenarios. (a) Forecast as an independent task and (b) forecast as the consumer of T2. Results in Figure 5.10 clearly show that the forecasted execution times lie closer to actual when execution time of the predecessor task, T2, is included in the estimation process. The absolute error plots reveal that the inclusion of spatial dependencies not only reduces the error but it always stays less compared to the estimation error as an independent task. It is apparent that the inclusion of spatial dependencies in the execution time model significantly improves the quality of the forecast. The state vector variance, an indirect measure of convergence, of estimator is shown in Figure 5.11 from the beginning for task T4. The rapid monotonic decrease in state vector variance shown in Figure 5.11 further strengthen the significance of spatial dependency. Convergence curves make it clear that the observer model incorporating spatial dependencies converges rapidly than independent estimate. Moreover, state covariance of the pipelined estimate remains lower after the convergence and yields stable and quality forecast.

The Figure 5.12 and 5.13 show the forecast error probability distribution for T1 and T4 respectively. In case of T1, the error distribution is widely distributed while for T4, error distribution has limited range. Properties of distributions also suggest that significant improvement in forecast has been achieved by incorporating the spatial dependency among tasks.

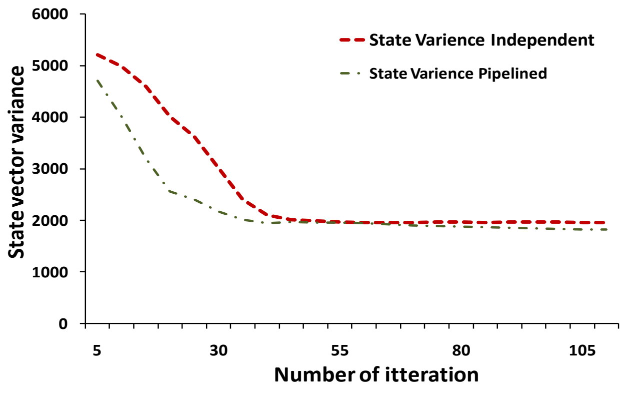
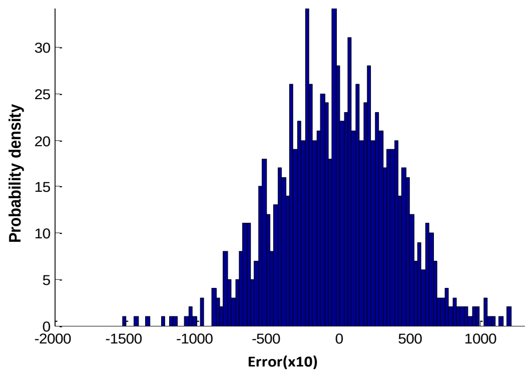


Figure 5.11: Convergence



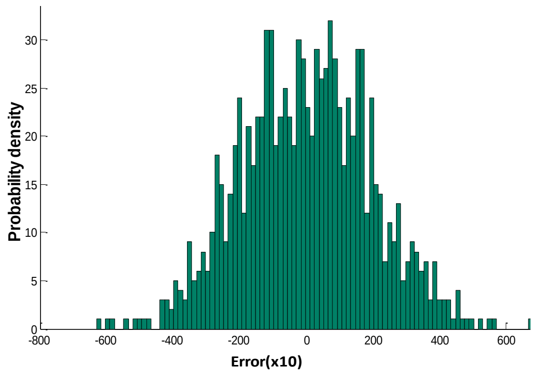


Figure 5.12: Probability density function of error in the forecast of T1

Figure 5.13: Probability density function of error in the forecast of T4

To evaluate the forecast quality, consider Figure 5.14 which shows the estimation error for five tasks of the benchmark. In Figure 5.14, percentage residual error is computed for the comparison purposes and the forecasts are clustered into the discrete bins having of 20% of absolute error width. The forecast error greater than 100% is combined into a single bin instead of showing the lengthy decaying tail of error bins. The analysis of forecast and task execution time characteristics reveals that the abrupt change in the task phase leads to the large errors, sometime greater than the 100%. Results clearly show that the forecast error by SETS largely falls within the error range of 20% which is the quality forecast considering the stochastic nature of the execution time. The computational overhead of the SETS for the experimental setup is 7.8% (on average). This overhead is relatively very small compared to the benefits obtained in term of quality forecast. Furthermore, computational overhead of SETS is also the function of task granularity. In the experimental benchmark, the tasks are of fine granularity which results in bigger share of computation cost in percentage terms. Consider the case of coarse grained tasks the percentage overhead will diminish monotonically with the increase in granularity. Without a priori assumption/knowledge about the tasks or computing platform the quality of forecast is sufficient for further applications.

To analyze the scheduling, the actual and forecasted execution time traces are used to simulate the scheduling process on Cheddar. On Cheddar, 4 cores are simulated and are scheduled by variable laxity EDF scheduling scheme. The experiments are performed with non-preemptive uniform priority attributes. The results are measured in terms of task utilization (execution time/deadline), task schedulability and deadline misses.

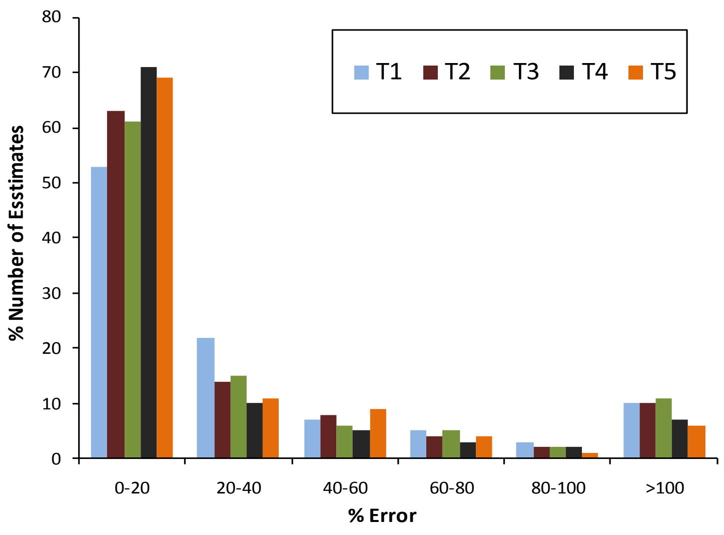


Figure 5.14: SETS forecast error

To establish the effectiveness of SEAL, comparison of results is carried out with state of the art proposed in [MA2010], will be called “MILLS”. MILLS also consider the stochastic execution time behavior of the tasks and argue that the WCET seriously underutilizes the resources and an average case can be considered. Moreover, for scheduling it also considers the global EDF policy which establishes the fairness of the comparison. The prime difference in MILLS lies in the derivation of the prior tardiness bound for global EDF. However, it only considers the upper bounded deviation from the mean execution time of the task. The absence of lower bound results in the under-utilization of task when execution time falls below the mean execution time. The simulation on Cheddar shows that the SETS improves the task utilization to the factor of 0.37, an improvement of 87%.

Figure 5.15 summarizes the results of the task utilization, task schedulability and the deadline misses. The results clearly show the advantage of SETS over MILLS. The improvement in task utilization achieved by SETS is 76% better than the MILLS. The huge improvement in the task utilization is due to the lack of consideration of the execution time below the mean value. The SETS does not define upper or lower bounds on execution times, computes it at run time, and hence gives the better utilization. The same trend can be observed for the task schedulability. However, the percentage improvement of the SETS over the MILLS is 68% because of some large over forecast of execution times by the SETS which benefits in terms of the deadline misses. The task schedulability and deadline misses can be considered as a tradeoff. Lastly, the task scheduled by the SETS are less susceptible to deadline misses. The Figure 5.15 shows that average deadline missed by the SETS are 53% less than MILLS.

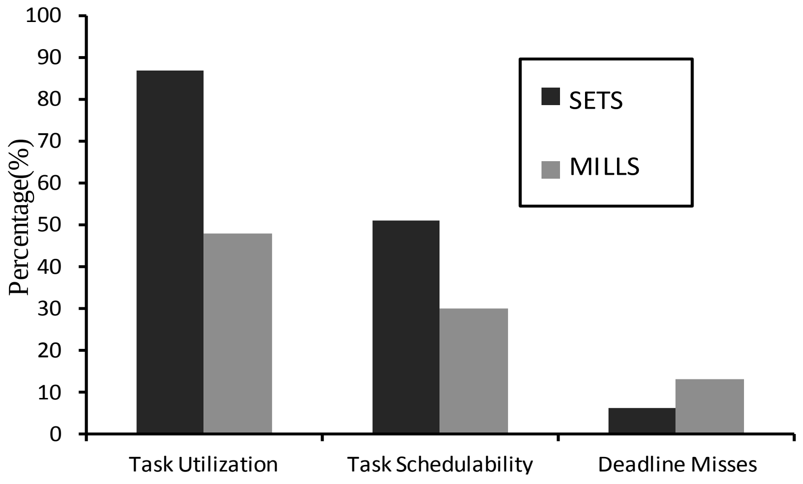
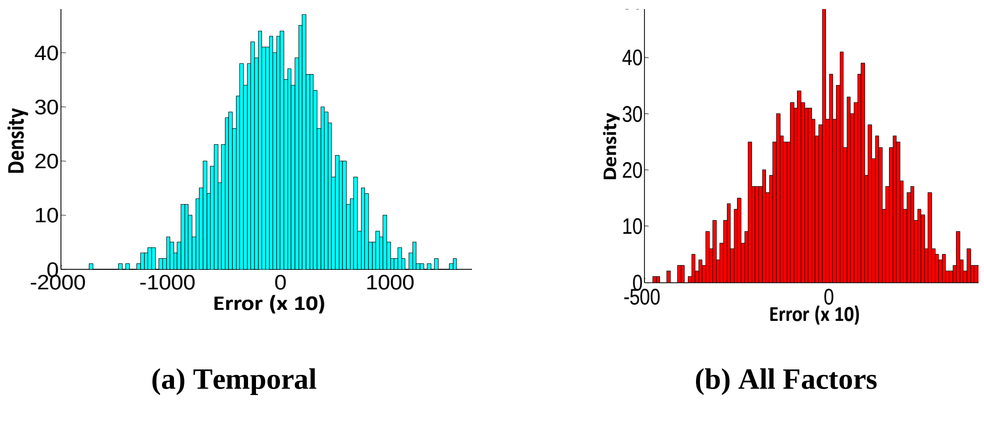


Figure 5.15: Comparison of SETS with MILLS for different metrics

The results and their analysis make it clear that the spatial correlation has profound positive impact on forecast quality. Moreover, run time estimate of execution time alleviates the constraints of fixed time scenario and thus significantly improves the effective utilization of system resources. The results reveal the large forecast errors by SETS in case of a abrupt execution phase change. To eliminate this limitation Section 4.3 developed the execution time model which incorporates the phase information.

The soft error aware low power scheduling scheme, SEAL, is developed in Section 4.3 and 4.6, which addresses the short coming of the SETS as well as extend the solution to the futuristic problem of dependability. To evaluate SEAL, first consider the impact of the inclusion of the derivative (phase information) and spatial dependencies in the model. The forecast error distribution is plotted in Figure 5.16.(a,b) for task T4. Figure 5.16.a shows the error distribution when the forecast is carried out as an independent task and without phase information (setting the spatial and derivative coefficients to zero in the system matrix). Figure 5.16.b shows the error distribution while considering all the factors discussed in the development of SEAL. The cross comparison of Figure 5.16(a,b) clearly shows that the distribution assumes a very limited range when the derivative and spatial dependency are taken in to account. It is evident that a considerable improvement has been achieved to obtain a better execution time forecast.

The Figure 5.17 shows the forecast error for five tasks of the benchmark application and error is clustered into bins (like Figure 5.14) . It is evident that most of the tome forecast is bounded to the first error bin (20% error). It is clear that the forecast by SEAL has less large errors compared to SETS, which is due to the inclusion of phase information in the model. Overall, the forecast quality of SEAL is superior to SETS; a quality forecast is the essential requirement for the desing



(a) Temporal **(b) All Factors**

Figure 5.16: Forecast error distribution

of low power scheduling. Results clearly show that SEAL achieves better forecast (even in case of the H.264 which is a compute-intensive control-dominant application), primarily because of an innovative modeling providing a generic solution independent of the stochastic characteristics of the systems dynamics and data to be processed.

To obtain the results of SEAL for soft error aware low power scheduling, each task is scheduled twice for the check point and roll back redundancy. As mentioned earlier, the video sequences are encoded with different QPs. Low valued QP results in the high bit-rate encoding, hence more data to be decoded by the decoder and vice versa. A change in QP induces a significant change in the execution time, and results in varying task utilizations, which is necessary for an appropriate evaluation [SM2010]. Figure 5.18 shows the average energy savings for 3 different SERs by keeping the system failure probability constant. The results reveal that energy savings are maximum (73%) when tasks have very low utilization (QP=28) and SER=1.00E-8. The gain in energy savings reduces significantly (27%) as the task utilization increases (QP=16); increased utilization leads to a longer execution time and significant amount of extra checkpoint time is required to achieve a constant system failure probability. Hence, increase in the task utilization incurs a two-fold impact on energy savings. Increase in SER also causes a negative impact on energy savings. This impact is relatively moderate for low task utilization (QP=28), but is significant for high task utilization (QP=16). The gain in energy savings is reduced by 58% for high task utilization while a reduction of only 12% is observed for low task utilization.

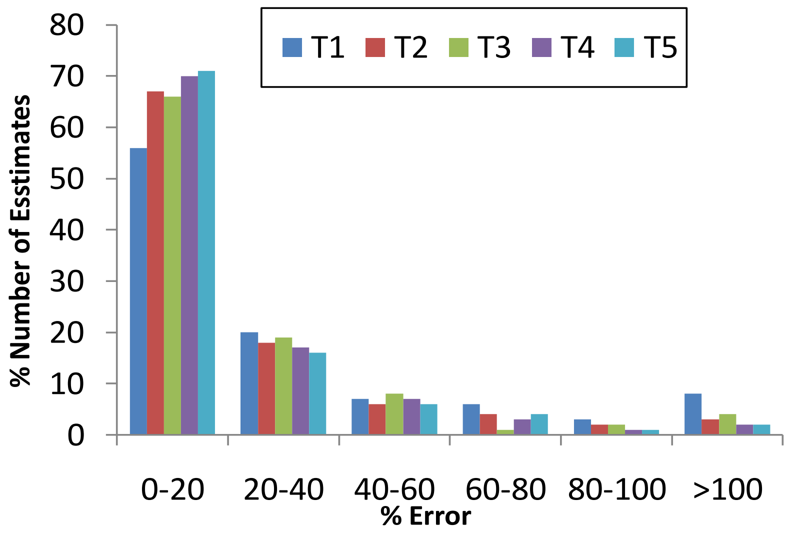


Figure 5.17: Forecast error (all runs)

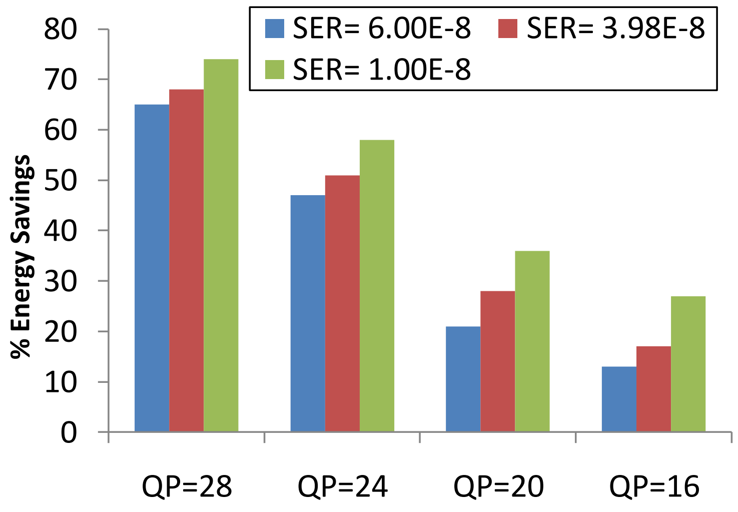


Figure 5.18: Energy savings by SEAL

The SEAL is compared with state of the art Reliability Aware Power Management scheme (RAPM) proposed in [SM2010]. The RAPM considers probabilistic execution times with variable task utilization and tasks are categorized in terms of low and high utilization. These characteristics make RAPM suitable candidate for comparison with SEAL. For the fair comparison of SEAL with RAPM, the tasks are categorized with QP 24 and 28 (average utilization= 19%) as low utilization tasks and tasks with QP 16 and 20 (average utilization= 52%) as high utilization tasks. Figure 5.19 gives the comparison of energy savings for SEAL and RAPM considering the SER of 3.98E-8; a pragmatic real world SER. Results clearly show that SEAL outperforms RAPM for both categories of task utilization. SEAL achieves 29% and 21%

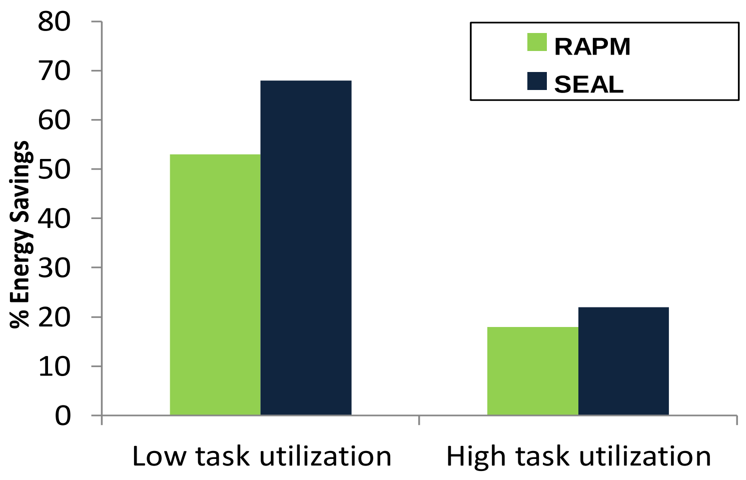


Figure 5.19:Comparison of SEAL with RAPM

more energy savings compared to RAPM for low and high task utilization respectively. The better energy savings for high utilization tasks is due to the pessimistic heuristic used in RAPM. RAPM does not report the deadline misses, though it assumes the uncertainty in execution times, and we found it to be 11.8% for our data set while SEAL deadline misses are under 7%. It is clear that SEAL is superior to state-of-the-art with the fundamental advantage of no a priori requirements of execution time characteristics.

Chapter 6 Summary

The power and thermal constraints bring forward the dawn of multicore processing platforms that have revolutionized computing in numerous aspects. Not only the underlying architecture has undergone drastic changes in switching from single to multiple cores on the same die, but the associated overlying software has had to evolve itself accordingly. It is required to meet the challenges of the large computational parallelism being ushered forth by the multiplicity of the processing elements. The goal of achieving high performance with low energy consumption is non-trivial. The demand for extracting a high performance-to-power ratio, enforced for power management strategies. One of the most popular mechanisms is Dynamic Voltage Frequency Scaling (DVFS), in which the processor's clock frequency and supply voltage are changed in tandem by software during the course of operation. In DVFS, a processor is set to use the most appropriate frequency and voltage level at any given interval, spreading bursty workload over time and avoiding fast and idle scenarios that consume more energy than is truly required for the computation. The parallelism exposed by multicore architectures poses new challenges as well as open up new design opportunities for DVFS compared to single core systems. The multicore/multi threaded systems give rise to additional scenarios such as the interdependence of threads mapped on different cores (software pipelining), shared resources such as a communication mechanism, shared caches etc. In general, primary components of the DVFS are (i) Observer (ii) Regulator. The Observer monitors the workload in a specified time intervals and forecast the future workload conditions. The regulator selects the appropriate performance level based on the forecasted workload. Naturally, run time estimation accompany an error and hence the primary focus of this work is the DVFS system design for soft real time systems without any a priori assumption on the workload and its characteristics.

In the realm of soft real time multicore systems, the programming of such systems presents additional challenges due to the timeliness constraints of the tasks. It is essential that the tasks get processed within the predetermined deadlines; this, in turn, necessitates efficient DVFS mechanism, which depends on an accurate knowledge of the amount of workload for a processing element. However, the workloads are not known in advance; therefore, an accurate estimation of the workload is of paramount significance. Most of the related work in the field of

DVFS assumes that the work load or their characteristics are known quantities. However, these assumptions are unfair since the workload is a run-time property depending on numerous factors, ranging from the executing tasks to the run-time state of the underlying platform. It exhibits purely stochastic characteristics due to application-dependencies (e.g. the variation in the amount and type of data input to the application), platform-dependencies (due to the micro-architectural attributes of the platform influencing queuing, caching etc.) and environment-dependencies (such as databases accesses, communication links etc.). The mathematical models proposed under this research work obviate the need for a priori knowledge, and constitute a general solution orthogonal to the underlying architecture, the executing tasks and the statistical properties of the workload with efficient performance, not only in terms of resource utilization, but energy consumption as well.

As mentioned previously, the workload of a core in a multicore systems depends on previous workload conditions and workloads of adjacent cores. Thus, the observer is modeled as a function of 'temporal and spatial correlation' factors. In the real world scenarios, it is extremely difficult if not impossible to determine and model all parameters. Only limited parameter vectors can be used to model the workload because of the practical considerations that only the net effect of these dependencies is modeled. The developed periodic DVFS scheme in this thesis considers the workload conditions in the uniformly sampled interval (e.g. 100 ms). At the beginning of each interval, the observer observes the workload conditions for a core and forecast the future workload. The un-modeled factors, whose affects are even unknown, show dependence on the modeled factors because the modeled and un-modeled factors belong to the same process. To capture the source of error, estimator is modeled in the non-linear state space. The state space model introduces state variables to compensate for the modeled factors. The fundamental advantage of the state space model is the extensibility to the joint state space model to forecast the workloads and model parameters simultaneously, which in result eliminate fundamental assumptions/constraints posed on workload estimation. The state space model is solved recursively by Extended Kalman Filter (EKF), which is well suited for nonlinear and non-stationary workload conditions. The regulator takes the reference utilization (selectable by the user) and forecasted workload to compute the necessary frequency gain. The uncertainty in the previous estimation is used as an uncertainty factor in order to obtain better decision for the gain. Based on the regulator gain, the core frequency step is adjusted.

The periodic invocation of DVFS manager abstracts the execution time of multiple tasks in a single workload interval. This abstraction may hamper the quality of a forecast as multiple sub processes are considered as a single process. To alleviate this and for the stochastic assumptions of EKF, we resorted to tackle the problem at the task graph level. The proposed model exploits the ‘temporal and spatial correlations’ among tasks in the software pipeline. As the tasks are working on the same data set (under the same application), the execution time of the former would influence the execution time of the latter. Hence, the tasks exhibit ‘spatial correlations’. The ‘temporal correlation’ refers to the correlation exhibited by the execution times of the same task upon its repeated instantiations in successive scheduling intervals. Moreover, the tasks experience phases of high computational loads followed by low computational loads and vice versa. The estimation of execution time suffers from a large error in case of abrupt phase change. At design time, it is not known when such a phase shift will occur, due to the dynamic nature of the data to be processed. In this context, an accurate estimation of the execution time necessitates modeling the local as well as the global phase. The proposed estimator is built upon a non-linear, non-Gaussian state-space model, to cater the un-modeled parameters and phase changes. In the given approach, the coefficients are adapted at run-time alongside the estimation, rendering a joint state-space model. The joint state space model is solved recursively independent of stochastic properties of execution time by the online Monte Carlo method. The forecasted execution times are used to select the appropriate performance level in the next scheduling interval and then Earliest Deadline First (EDF) scheduler schedules tasks.

Besides the intricacy of the hardware and software aspects of the architecture, another aspect of modern multicore processing platforms is the high scale of integration employed. Reduction in the logic gate size has increased the probability of the occurrence of ‘soft-errors’. These errors require re-computations therefore, act as bottlenecks to performance. The occurrence of these errors is linked to the processor’s operating voltage i.e. power consumption. Hence, the solutions employed to improve the performance-to-power tradeoff, such as (DVFS), must incorporate ways to cater for the soft errors as well. The decrease in performance level increases the probability of soft error, a model based on ‘soft error rate’ (SER) for different performance levels is developed. By assuming a uniformly distributed application checkpoint mechanism for error detection/correction, the execution times of the tasks are adjusted and incorporated in the scheduling decision, to keep the error probability constant. The resultant execution times are utilized to schedule the tasks for minimum energy consumption by the EDF scheduler. The results obtained show significant improvement compared to the state-of-the-art.

Bibliography

[A2004] Sean Eron Anderson, “http://www-graphics.stanford.edu/~seander/bithacks.html, Bit Twiddling Hacks”, 2004.

[A2006] K.J.Astrom,“Introduction To Stochastic Control Theory”, Dover Publishing, New York, 2006.

[AAE2003] M. Anis, S. Areibi, and M. Elmasry, “Design and Optimization of Multi-Threshold CMOS (MTCMOS) Circuits”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp. 1324–1342, Oct. 2003.

[ABD$^+$2003] D. Albonesi, R. Balasubramonian, S. Dropsho, S. Dwarkadas, E. Friedman, M. Huang, V. Kursun,G. Magklis, M. Scott, G. Semeraro, P. Bose, A. Buyuktosunoglu, P. Cook, and S. Schuster, “Dynamically Tuning Processor Resources with Adaptive Processing”, IEEE Computer, pp. 43–51, , 2003.

[ACM$^+$2003] N. AbouGhazaleh, B. Childers, D. Mosse, R. Melhem, and M. Craven, “Energy Management for Realtime Embedded Applications with Compiler Support”, In Proceedings of the Proceedings of the 2003 ACM SIGPLAN Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES), pp. 284-293, 2003.

[AIC$^+$2002] A. Azevedo, I. Issenin, R. Cornea, R. Gupta, N. Dutt, A. Veidenbaum, and A. Nicolau, “Profile based Dynamic Voltage Scheduling using Program Checkpoints”, In Proceedings of the conference on Design, automation and test in Europe (DATE’02), pp. 168-175, 2002.

[ACPI2010] http://www.acpi.info/, "Advanced Configuration and Power Interface Specification 4.0a", 2010.

[ASE$^+$2004] A. Andrei, M. Schmitz, P. Eles, Z. Peng, and B. Al-Hashimi, “Overhead-Conscious Voltage Selection for Dynamic and Leakage Power Reduction of Time-Constraint Systems”, In ”Proc. Design, Automation and Test in Europe (DATE)”, pp. 518–523, Feb 2004.

[AVS2008] F. Abate, M. Violante and L. Sterpone, “A New Mitigation Approach for Soft Errors in Embedded Processors”, IEEE Nuclear Science, pp. 2063-2069, 2008.

[B1990] William L. Brogan, “Modern Control Theory”, 3rd Edition, Prentice Hall, 1990.

[B1994] E.J. Bomhoff, “Financial Forecasting for Business and Economics”, Academic Press, San Diego 1994.

[B1998] E.Brookner,“Tracking and kalman filtering made easy” John Wiley & Sons-Interscience, 1998.

[B1999] N. Bergman, “Recursive Bayesian estimation: Navigation and tracking applications,” Ph.D. dissertation, Linköping Univ., Linköping, Sweden,1999.

[B2000] F. Bellosa, “The benefits of event-driven energy accounting in power-sensitive systems”, In Proceedings of 9th ACM SIGOPS European Workshop, September 2000.

[B2001] Ronal S. Burns, “Advanced Control Engineering”, Butterworth-Heinemann 2001.

[B2005] L. A. Barroso, “The Price of Performance”, ACM Queue, pp 48-53, 2005.

[BBG$^+$2005] L. Benini, D. Bertozzi, A. Guerri and M. Milano, “Allocation and Scheduling for MPSoCs via Decomposition and No-Good Generation”, In ”Proc. of International Joint Conference on Artificial Intelligence (IJCAI)”, pages 1517–1518, 2005.

[BBT$^+$2001] N. K. Bambha, S. S. Bhattacharyya, J. Teich, and E. Zitzler, “Hybrid global/ local search strategies for dynamic voltage scaling in embedded multiprocessors”, Proceedings of the CODES, pp. 243-248, 2001.

[BBY$^+$2009] Sung-Yong Bang, Kwanhu Bang, Sungroh Yoon, Eui-Young Chung, “Run-Time Adaptive Workload Estimation for Dynamic Voltage Scaling”, IEEE Trans. on CAD of Integrated Circuits and Systems, number=9, volume=28, 2009.

[BD1996] Craig Boutilier and David Poole, “Computing optimal policies for partially observable decision processes using compact representations”, In Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI Press/The MIT Press, pp 1168-1175, 1996.

[BH1992] R.G. Brown, and P. Y. C. Hwang, “Introduction to Random Signals and Applied Kalman Filtering”, Second Edition, John Wiley & Sons Inc, 1992.

[BJ2008] W.Lloyd Bircher, Lizy K. John, “Analysis of dynamic power management on Multicore processors”, ICS '08: Proceedings of the 22nd annual international conference on Supercomputing, pp. 327—338, 2008.

[BJS2007] S.Borkar, N. P. Jouppi and P. Stenström, “Microprocessors in the Era of Terascale Integration”, IEEE/ACM DATE, pp. 237-242, 2007.

[BLV$^+$2004] W. Bircher, J. Law, M. Valluri, and L. K. John, “Effective Use of Performance Monitoring Counters for Run-Time Prediction of Power”, Technical Report TR-041104-01, University of Texas at Austin, Nov. 2004.

[BM2001] D. Brooks and M. Martonosi, “Dynamic thermal management for high-performance microprocessors”, In Proceedings of the Seventh International Symposium on High-Performance Computer Architecture (HPCA-7), January 2001.

[BPB$^+$1999] R. Bogliolo, A. Paleologo, L. Benini, G. D. Micheli, “Policy optimization for Dynamic power management”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 18, Issue 6, pp. 813–833, 1999.

[BSB$^+$2001] A. Buyuktosunoglu, S. Schuster, D. Brooks, P. Bose, P. W. Cook, and D. H. Albonesi, “An Adaptive Issue Queue for Reduced Power at High Performance”, In Proceedings of the First International Workshop on Power-Aware Computer Systems (PACS’00), 2001.

[BVL$^+$2005] W. L. Bircher, M. Valluri, J. Law, and L. K. John, “Runtime identification of microprocessor energy saving opportunities”, In Proceedings of the 2005 International Symposium on Low Power Electronics and Design (ISLPED), pp. 275-280, 2005.

[C2001] R. Carmona, “Statistical Analysis of Financial Data, with an implementation in S-plus”, Springer, 2004.

[C2002] N. H. Chan, “Time Series: Applicatios to Finance”, John Wiley & Sons, New York, 2002.

[C2006] M. Chin, “Desktop CPU Power Survey”, In SPCR Forum, 2006.

[CAT$^+$2001] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, “Managing Energy and Server Resources in Hosting Centers”, in ACM Symposium on Operating Systems Principles, pp. 103–116, 2001.

[CC2008] Charles K. Chui, Guanrong Chen, “Kalman Filtering with Real-Time Applications”, Springer, 2008.

[CCV$^+$2008] E. Chun, Z. Chishti, T.N. Vijaykumar, “Shapeshifter: Dynamically changing pipeline width and speed to address process variations”, International Symposium on Microarchitecture, MICRO-41, pp. 411–422, 2008.

[CK2007] J.J.F. Commandeur and S. Koopman. “Time series analysis using state space models”, Oxford University Press, USA, 2007.

[CKE2000] T. L. Cignetti, K. Komarov, and C. S. Ellis, “Energy Estimation Tools for The Palm”, in International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp. 96–103, 2000.

[CL2006] C.-B. Cho and T. Li, “Complexity-based Program Phase Analysis and Classification”, In Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT), pp. 105-113, 2006.

[CKL1994] Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman, “Acting optimally in partially observable stochastic domains”, In Proceedings of the Twelfth National Conference on Artificial Intelligence, pp. 1023-1028, 1994.

[CM2005] G. Contreras and M. Martonosi, “Power Prediction for Intel XScale Processors Using Performance Monitoring Unit Events”, In Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), 2005.

[CMB2002] E.-Y. Chung, G. D. Micheli, and L. Benini, “Contents Provider-Assisted Dynamic Voltage Scaling for Low Energy Multimedia Applications,” in International Symposium on Low Power Electronics and Design, pp. 42–47, 2002.

[COJ2001] J. Cook, R. L. Oliver, and E. E. Johnson, “Examining performance differences in workload execution phases”, In Proceedings of the IEEE International Workshop on Workload Characterization (WWC-4), 2001.

[CRM2004] O. Celebican, T. S. Rosing, and V. J. M. III, “Energy Estimation of Peripheral Devices in Embedded Systems,” in Great Lakes symposium on VLSI, pp. 430–435, 2004.

[CS2001] A. P. Chandrakasan and A. Sinha, “JouleTrack: A Web Based Tool for Software Energy Profiling”, In Proceedings of the 38th Design Automation Conference (DAC’01), pp. 220-225, 2001.

[CSC$^+$2007] Jike Chong, Nadathur Rajagopalan Satish, Bryan Catanza-ro,Kaushik Ravindran, Kurt Keutzer, “Efficient Parallelization of H.264 Decoding with Macro Block Level Scheduling”, Multimedia and Expo, IEEE ICMCS Conference, pp. 1874-1877, 2007.

[CSP2004] K. Choi, R. Soma, and M. Pedram, “ Dynamic Voltage and Frequency Scaling based on Workload Decomposition”, In Proceedings of International Symposium on Low Power Electronics and Design (ISLPED), pp. 174-179, 2004.

[D1998] A. Doucet, “On sequential Monte Carlo methods for Bayesian filtering,” Dept. Eng., Univ. Cambridge, UK, Tech. Rep., 1998.

[D2010] Annette J. Dobson, “An Introduction to Generalized Linear Models”, Second edition, CRC press, 2010.

[DB2008] Richard C. Dorf, Robert H. Bishop, “Modern Control Systems”, Prentice Hall, 2008.

[DF1996] Val Donaldson and Jeanne Ferrante, “Determining Asynchronous Acyclic Pipeline Execution Times”, IPPS, pp. 568-572, 1996.

[DFG+2001] A.Doucet, N.Freitas, N.Gordon, A. Smith, “Sequential Monte Carlo Methods in Practice”, Springer, 2001.

[DK2001] James Durbin, Siem Jan Koopman, “Time Series Analysis by State Space Models”, Oxford University Press, USA, 2001.

[DMV+2001] V. Delaluz, M. Kandemir, N. Vijaykrishnan, A. Sivasubramaniam, and M. J. Irwin,“Hardware and Software Techniques for Controlling DRAM Power Modes,” IEEE Transactions on Computers, vol. 50, pp. 1154–1173, November 2001.

[DMZ2002] A. Dudani, F. Mueller, and Y. Zhu, “Energy Conserving Feedback EDF Scheduling for Embedded Systems with Real-time Constraints”, In LCTES/SCOPES ’02: Proceedings of the joint conference on Languages, compilers and tools for embedded systems, pp. 213-222, 2002.

[DS2002] A. Dhodapkar and J. Smith, “Managing multi-configurable hardware via dynamic working set analysis”, In 29th Annual International Symposium on Computer Architecture, pp. 233-246, 2002.

[DS2003] A. Dhodapkar and J. Smith, “Comparing Program Phase Detection Techniques”, In 36th International Symp. on Microarchitecture, pp. 217-227, 2003.

[EMM2002] E. Mootaz Elnozahy, R. Melhem, and D. Moss´e, “Energy efficient duplex and tmr real-time systems”, In Proc. of The IEEE Real-Time Systems Symposium, pp.256-266, 2002.

[ENP+2003] D. Ernst, S. D. Nam Sung Kim, S. Pant, T. Pham, R. Rao, C. Ziesler, D. Blaauw, T. Austin, and T. Mudge, “Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation, In Proceedings of the 36th International Symp. on Microarchitecture, pp. 7-18, 2003.

[ERK+2006] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, “Full-system Power Analysis and Modeling for Server Environments”, In Proceedings of the Workshop on Modeling Benchmarking and Simulation (MOBS), 2006.

[ESY+2005] L. Eeckhout, R. Sundareswara, J. Yi, D. Lilja, and P. Schrater., “Accurate Statistical Approaches for Generating Representative Workload Compositions”, In Proceedings of the IEEE International Symposium on Workload Characterization, 2005.

[F2005] Bernard Friedland, Control System Design: An Introduction to State-Space Methods, Dover Publications, 2005.

[FF2010] Richard M. Feldman , Ciriaco Valdez-Flores , “Applied Probability and Stochastic Processes”, Springer, 2010.

[FFB+2000] K. I. Farkas, J. Flinn, G. Back, D. Grunwald, and J. M. Anderson, “Quantifying the Energy Consumption of a Pocket Computer and a Java Virtual Machine,” in International conference on Measurements and Modeling of Computer Systems, pp. 252-263, 2000.

[FKM+2002] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge, “Drowsy Caches: Simple Techniques for Reducing Leakage Power”, In Proceedings of the 29th International Symposium on Computer Architecture (ISCA-29), 2002.

[FMJ+2007] J. Friedrich, B. McCredie, N. James, B. Huott, B. Curran, E. Fluhr, G. Mittal, E. Chan, Y. Chan, D. Plass, S. Chu, H. Le, L. Clark, J. Ripley, S. Taylor, J. Dilullo, and M. Lanzerotti, “Design of the POWER6 Microprocessor”, In IEEE International Solid-State Circuits Conference (ISSCC 2007), 2007.

[FS1999] J. Flinn and M. Satyanarayanan, “Energy-Aware Adaptation for Mobile Applications,” in ACM Symposium on Operating Systems Principles, pp. 48–63, 1999.

[GA2008] Mohinder S. Grewal,Angus P. Andrews, “Kalman filtering theory and practice using Matlab”, John Wiley & Sons, 2008.

[GAD+2006] M. Golden, S. Arekapudi, G. Dabney, M. Haertel, S. Hale, L. Herlinger, Y. Kim, K. McGrath, V. Palisetti, and M. Singh, “A 2.6GHz Dual-Core 64b x86 Microprocessor with DDR2 Memory Support”, In IEEE International Solid-State Circuits Conference (ISSCC 2006), 2006.

[GCW1995] K. Govil, E. Chan, and H. Wasserman, “Comparing algorithms for dynamic speed-setting of a low-power CPU”, Proceedings of the First ACM International Conference on Mobile Computing and Networking, 1995.

[GJ2005] R. Gupta, R. Jejurikar, “Dynamic Slack Reclamation with Procrastination Scheduling in Real-Time Embedded Systems”, In Design Automation Conference (DAC), 2005.

[GK2001] F. Gruian and K. Kuchcinski, “ LEneS: Task Scheduling for Low-Energy Systems Using Variable Supply Voltage Processors”, In Proc. ASP-DAC’01, pp. 449–455, 2001.

[GK2006] S. Gurun and C. Krintz, “A Run-Time, Feedback-Based Energy Estimation Model For Embedded Devices”, In Proceedings of the International Conference on Hardware-Software Codesign and System Synthesis (CODES+ISSS), pp. 28-32, 2006.

[GKS$^+$2008] D. P. Gulati, C. Kim, S. Sethumadhavan, S. W. Keckler, and D. Burger, “Multitasking workload scheduling on flexible-core chip multiprocessors,” in Proceedings of the 17th international conference on Parallel architectures and compilation techniques, pp. 187–196, 2008.

[GSJ$^+$2004] S. P. Gurrum, S. K. Suman, Y. K. Joshi, and A. G. Fedorov, “Thermal Issues in Next-Generation Integrated Circuits”, IEEE Transactions on Device and Materials Reliability, pp. 709-714, 2004.

[GSK$^+$2003] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke, “Reducing Disk Power Consumption in Servers With DRPM”, IEEE Computer, vol. 36, pp. 59–66, December 2003.

[H1989] A. C. Harvey, “Forecasting, Structural Time Series Models and the Kalman Filter”, Cambridge University Press, Cambridge, 1989.

[H1996] Monson H. Hayes, “Statistical Digital Signal Processing and Modeling”, John Wiley & Sons, 1996.

[H1997] Hwei P. Hsu, “Probability, Random Variables and Random Processes”, McGraw-Hill (Schaum's Outlines), 1997.

[H2001] Simon O. Haykin, “Adaptive Filter Theory”, 4th edition, Prentice Hall, 2001.

[H2007] I. Ahmand, “Easy and Efficient Disk I/O Workload Characterization in VMware ESX Server”, in Proc. of International Symposium on Workload Characterization, pp. 149-158, 2007.

[H2009] Aydin Hakan , “Reliability-Aware Energy Management for Periodic Real-Time Tasks”, IEEE Transactions on Computers, 2009.

[HBA2003] S. Heo, K. Barr, and K. Asanovic, “Reducing Power Density through Activity Migration”, In Proceedings of International Symposium on Low Power Electronics and Design (ISLPED), pp. 217-222, 2003.

[HCG$^+$2006] T. Heath, A. P. Centeno, P. George, L. Ramos, Y. Jaluria, and R. Bianchini, “ Mercury and freon: Temperature emulation and management in server systems”, In Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 106-116, 2006.

[HHI2008] http://iphome.hhi.de, H.264/AVC Software Coordination, 2008

[HJK$^+$2003] M. Horowitz, A. Joch, F. Kosssentini, and A.Hallapuro, “H.264/AVC baseline profile decoder complexity analysis,” IEEE Trans. Circuits and Syst. Video Technology., vol. 13, no.7, 2003.

[HJK2005] C. Hu, D. Jimenez, and U. Kremer, “Toward an Evaluation Infrastructure for Power and Energy Optimizations”, In Workshop on High-Performance, Power-Aware Computing, 2005.

[HK2003] C.H. Hsu and U. Kremer, “The design, implementation, and evaluation of a compiler algorithm for CPU energy reduction”, In Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation, pp.38-48, 2003.

[HKM2002] Z. Hu, S. Kaxiras, and M. Martonosi, “Let Caches Decay: Reducing Leakage Energy Via Exploitation of Cache Generational Behavior,” ACM Transactions on Computer Systems, vol. 20, May 2002, pp. 161–190.

[HM2004] J. Hu and R. Marculescu, “Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints”, In Proc. Design, Automation and Test in Europe (DATE), pp. 234–239, 2004.

[HM2007] S. Herbert and D. Marculescu, “Analysis of Dynamic Voltage/Frequency Scaling in Chip-Multiprocessors”, In Proceedings of the 2007 International Symposium on Low Power Electronics and Design, pp. 38-43, 2007.

[HP2003] J. L. Hennessy and D. A. Patterson, “Computer Architecture: A Quantitative Approach”, Morgan Kaufman Publishers, Third Edition, 2003.

[HPH2004] T. Heath, E. Pinheiro, J. Hom, U. Kremer, and R. Bianchini, “Code Transformations for Energy-Efficient Device Management”, IEEE Transactions on Computers, pp. 974–987, 2004.

[HRS2003] M. J. Hind, V. T. Rajan, and P. F. Sweeney, “Phase Shift Detection: A Problem Classification”, IBM Researh Report RC-22887, IBM T. J. Watson, 2003.

[HRT2003] M. Huang, J. Renau, and J. Torrellas, “Positional Adaptation of Processors: Application to Energy Reduction”, In Proceedings of the International Symp. on Computer Architecture, pp. 157-168, 2003.

[HS2000] P. Hazucha and C. Svensson, “Impact of CMOS technology scaling on the atmospheric neutron soft error rate”, IEEE Trans. on Nuclear Science, pp. 2586–2594, 2000.

[HSK$^+$2006] Y. Hotta, M. Sato, H. Kimura, S. Matsuoka, T. Boku, and D. Takahashi, " Profile-based Optimization of Power Performance by using Dynamic Voltage Scaling of a PC cluster," in Proc. of Parallel and Distributed Processing Symposium, pp. 8-16, 2006.

[I2007] Canturk Isci, "Workload Adaptive Power Management with live phase monitoring and prediction" PhD Dissertation, Princeton University, 2007.

[IBM2005] C. Isci, A. Buyuktosunoglu and M. Martonosi, "Long-term load phases duration prediction and application to DVFS", IEEE Micro, pp. 39-51, 2005.

[IH2009] Nabeel Iqbal and Jörg Henkel, "Efficient constant-time entropy decoding for H.264", IEEE/ACM Design Automation and Test in Europe (DATE), pp. 1440-1445, 2009.

[IH2010] Nabeel Iqbal, J.Henkel , "SETS: Stochastic execution time scheduling for multicore systems by joint state space and Monte Carlo", IEEE International conference on computer aided design (ICCAD), 2010.

[IAH2010] N.Iqbal, M.A Siddique, J.Henkel, "DAGS: Distribution Agnostic Sequential Monte Carlo Scheme for Task Execution Time Estimation", IEEE/ACM 13th Design Automation and Test in Europe Conference (DATE), 2010.

[IM2001] A. Iyer and D.Marculescu, "Power aware microarchitecture resource scaling", In Proceedings of Design Automation and Test in Europe (DATE), pp. 190-196, 2001.

[IM2002] A. Iyer and D. Marculescu, "Power Efficiency of Voltage Scaling in Multiple Clock, Multiple Voltage Cores," in Proc. of International Conference on Computer Aided Design (ICCAD), pp. 379-386, 2002.

[INT2008] Intel® Core™2 Extreme Quad-Core Mobile Processor and Intel® Core™2 Quad Mobile Processor on 45-nm Process", document number 320390, www.intel.com, 2008.

[IRH1986] R. K. Iyer, D. J. Rossetti, and M. C. Hsueh, "Measurement and modeling of computer reliability as affected by system activity", ACM Trans. Comput. System, pp. 214–237, 1986.

[ISH2010] Nabeel Iqbal, M.A. Siddique, J. Henkel, "RMOT: Recursion in Model Order for Task Execution Time Estimation in a Software Pipelin", IEEE/ACM Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010.

[ISH2011] N. Iqbal, M.A Siddique, J. Henkel, “SEAL: Soft Error Aware Low Power Scheduling by Monte Carlo State Space Under the Influence of Stochastic Spatial and Temporal Dependencies”, 48th ACM/EDA/IEEE Design Automation Conference (DAC´11), 2011.

[ITU2012] ITUT “H.264 : Advanced video coding for generic audiovisual services” available from http://www.itu.int/rec/T-REC-H.264-201201-I/en ,2012

[IZP1999] M. A. Iverson, F.O. Zguner, and L.C. Potter, “Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment”, IEEE Transaction on Computers vol. 48, pp 1374-1379, 1999.

[J1993] O. L. R. Jacobs, “Introduction to Control Theory”, 2nd Edition. Oxford University Press, 1993.

[J2009] Hwisung Jung, “Stochastic power and thermal management techniques for multicore systems”, PhD Dissertations, University of souther California, USA, 2009.

[JM2001] R. Joseph and M. Martonosi, “Run-time power estimation in high performance microprocessors”, In International Symposium on Low Power Electronics and Design, pages 135–140, 2001.

[JP2008] H. Jung and M. Pedram, “Continuous Frequency Adjustment Technique based on Dynamic Workload Prediction”, in Proc. of International Conference on VLSI Design, pp.415-420, 2008.

[JWP$^+$2005] P. Juang, Q. Wu, L.-S. Peh, M. Martonosi, and D. Clark, “Coordinated, Distributed, Formal Energy Management of Chip Multiprocessors”, In Proceedings of International Symposium on Low Power Electronics and Design, pp. 127-130, 2005.

[K1960] R. E. Kalman, “A new approach to Linear Filtering and Prediction Problems”, Transactions of the ASME-Journal of Basic Engineering, 82 (series D), pp.35-45, 1960.

[KBL$^+$2008] R. Knauerhase, P. Brett, T. Li, B. Hohlt, and S. Hahn, “Using OS Observations to Improve Performance in Multi-core Systems”, in Proc. of IEEE Micro, Vol. 28, Issue 3, pp. 54-66, 2008.

[KFJ$^+$2003] R. Kumar, K. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen, “Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction”, In Proceedings of the 36th International Symp. on Microarchitecture, pp.81-92, 2003.

[KGS2004] E. Kursun, S. Ghiasi, and M. Sarrafzadeh, “Transistor Level Budgeting for Power Optimization”, In Proceedings of the 5th International Symposium on Quality Electronic Design (ISQED’05), pp. 116-121, 2004.

[KGW+2008] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, “System level analysis of fast, per-core dvfs using on-chip switching regulators,” in 2008 IEEE 14th International Symposium on High Performance Computer Architecture, pp. 123–134, 2008.

[KHM2001] S. Kaxiras, Z. Hu, and M. Martonosi, “Cache decay: Exploiting generational behavior to reduce cache leakage power”, In Proceedings of the 28th International Symposium on Computer Architecture (ISCA-28), pp. 240-251, 2001.

[KHR2000] U. Kremer, J. Hicks, and J. Rehg, “Compiler-Directed Remote Task Execution for Power Management”, In Proceedings of the Workshop on Compilers and Operating Systems for Low Power (COLP’00), 2000.

[KL2003] S. Kang and Y. Leblebici, “CMOS digital integrated circuits analysis and design”, McGraw-Hill Series in Electrical and Computer Engineering, 2003.

[KM2008] Stefanos Kaxiras, Margaret Martosoni, “Computer Architecture Techniques for Power Efficiency”, Morgan and Claypool Publishers, 2008.

[KR2002] C. H. Kim and K. Roy, “Dynamic Vth Scaling Scheme for Active Leakage Power Reduction”, In Proceedings of the conference on Design, automation and test in Europe (DATE’02), pp. 163-167, 2002.

[KS2009] K. Pradheep Kumar and A. P. Shanthi , “Application of non-uniform laxity to EDF for aperiodic tasks to improve task utilization on multicore platforms”, CoRR 2009.

[KSN2007] M. Kondo, H. Sasaki, and H. Nakamura, “Improving fairness, throughput and energy efficiency on a chip multiprocessor through DVFS” vol. 35, no. 1. New York, NY, USA: ACM, pp. 31-38, 2007.

[KST2004] R. Kalla, B. Sinharoy, and J. Tendler, “A Dual-Core Multithreaded Processor”, IEEE Micro, pp. 40–47, 2004.

[KTJ+2005] R. Kumar, D. M. Tullsen, N. P. Jouppi, and P. Ranganathan, “Heterogeneous chip multiprocessors”, IEEE Computer, Vol. 38, No. 11, pp. 32-38, 2005.

[KTR+2003] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas, “Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction”, In Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 81-87, 2003.

[KYS+2006] Yong-Hwan Kim, Yoon-jong Yoo, Jeongho Shin, Byeongho Choi, and Joonki Paik, “Memory-Efficient H.264/AVC CAVLC for Fast Decoding”, IEEE Trans. On Consumer Electronics, Vol.52, No3, 2006.

[L1996] P. E. Landman. High-level power estimation, “In Proceedings of the 1996 International Symposium on Low Power Electronics and Design (ISLPED), pp. 29-35, 1996.

[L2005] Edward A. Lee, “Building unreliable system out of reliable components: A real time story”, http://www.eecs.berkeley.edu/Pubs, 2005.

[LB2006] B. Lee and D. Brooks, “Accurate and Efficient Regression Modeling for Microarchitectural Performance and Power Prediction”, In Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-XII), pp. 185-194, 2006.

[LBK+2007] T. Li, D. Baumberger, D. A. Koufaty, and S. Hahn, “Efficient operating system scheduling for performance-asymmetric multi-core architectures,” in SC ’07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing, pp. 1-11, 2007.

[LC1998] J. S. Liu and R. Chen, “Sequential Monte Carlo methods for dynamical systems”, Statist. Assoc., vol. 93, pp. 1032-1044, 1998.

[LCS2005] J. Lau, S. Schoenmackers, and B. Calder, “Transition Phase Classification and Prediction”, In 11th International Symposium on High Performance Computer Architecture, pp. 278-289, 2005.

[LEM+2001] S. Lee, A. Ermedahl, S. L. Min, and N. Chang, “An accurate instruction-level energy consumption model for embedded RISC processors”, In LCTES/OM, pp. 1-10, 2001.

[LJ2002] J. Luo and N. Jha, “Static and Dynamic Variable Voltage Scheduling Algorithms for Real-time Heterogeneous Distributed Embedded Systems”, Asia and South Pacific Conference on VLSI Design, pp. 719-726, 2002.

[LJ2003] T. Li and L. K. John, “Run-time Modeling and Estimation of Operating System Power Consumption”, in ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pp. 160-171, 2003.

[LM2001] Y-H. Lu and G. De. Micheli, “Comparing System-Level Power Management Policies”, IEEE Design & Test of Computers, Vol. 18, Issue 2, pp. 10-19, 2001.

[LM2006] J. Li and J. Martinez, "Dynamic Power-Performance Adaptation of Parallel Computation on Chip Multiprocessors", Proceedings of the 12th International Symposium on High-Performance Computer Architecture (HPCA-12), pp. 77-87, 2006.

[LR1994] P. E. Landman and J. M. Rabaey, "Black-box Capacitance Models for Architectural Power Analysis", In Proceedings of the International Workshop on Low Power Design, 1994.

[LR1995] P. E. Landman and J.M. Rabaey, "Activity-sensitive Architectural Power Analysis for the Control Path", Proceedings of the International Workshop on Low Power Design, pp. 93-98, 1995.

[LS2000] S. Lee and T. Sakurai, "Run-time Voltage Hopping for Low-power Real-time Systems" Proceedings of the Design Automation Conference (DAC'00), pp. 806-809, 2000.

[LSP$^+$2005] J. Lau, J. Sampson, E. Perelman, G. Hamerly, and B. Calder, "The Strong Correlation between Code Signatures and Performance", In IEEE International Symposium on Performance Analysis of Systems and Software, pp. 236-247, 2005.

[LTC2001] O.Y.-H. Leung, Chi-Ying Tsui, R.S.-K. Cheng, "Reducing power consumption of turbo-code decoder using adaptive iteration with variable supply voltage", IEEETransactions on Very Large Scale Integration (VLSI) Systems, pp. 34-41, 2001.

[LTK2004] L.F. Leung, C-Y. Tsui, and W-H. Ki, "Minimizing Energy Consumption of Multiple-Processors-Core Systems with Simultaneous Task Allocation, Scheduling and Voltage Assignment", Proceedings of Asia South Pacific Design Automation (ASP-DAC), pp. 647–652, 2004.

[LVK2005] P. Li, B. Veeravalli and A. A. Kassim, " Design and implementation of parallel video encoding strategies using divisible load analysis", IEEE Trans. Circuits and Systems for Video Technology, pp. 1098-1112, 2005.

[LWO2004] M. Liu, W.-S. Wang, and M. Orshansky, "Leakage Power Reduction by Dual-Vth Designs Under Probabilistic Analysis of Vth Variation", In Proceedings of the 2004 International Symposium on Low Power Electronics and Design (ISLPED), pp.2-7, 2004.

[M1965] G. E. Moore, "Cramming more components onto integrated circuits", In Electronics, pp. 114-117, 1965.

[M2005] A.Merkel, “Balancing Power Consumption in Multiprocessor Systems”, System Architecture Group, University of Karlsruhe, Karlsruhe. PhD Dissertation, 2005.

[MA2010] Alex F. Mill and J. Anderson, “A stochastic framework for multiprocessor soft real time scheduling”, IEEE RTAS, pp. 311-320, 2010.

[MAA$^+$2008] C. Meenderinck, A. Azevedo, M. Alvarez, B. Juurlink, and A. Ramirez, “ Parallel Scalability of H.264”, In Workshop on Programmability Issues for Multi-Core Computers (MULTIPROG), 2008.

[MBS2004] Dragorad Milovanovic, Zoran Bojkovic and Andreja Samcovic, “Video Coding with H.264/AVC : Tools performance and complexity”, WSEAS AMTA-EE-ICAI-MCBC-MCBE Conference, Venice, Italy, 2004.

[MEP2002] Sorin Manolache, Petru Eles, Zebo Peng. “Schedulability analysis of multiprocessor real-time applications with stochastic task execution times”, IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD).

[MFM$^+$2002] S. Martin and K. Flautner and T. Mudge and D. Blaauw, “Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Lower Power Microprocessors under Dynamic Workloads”, IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD), pp. 721-725, 2002.

[MJD$^+$2008] K. Meng, R. Joseph, R. P. Dick, and L. Shang, “Multi-optimization power management for chip multiprocessors,” in Proceedings of the 17th international conference on Parallel architectures and compilation techniques, pp. 177–186, 2008.

[ML2008] Jeonhak Moon, Seongsoo Lee, “Design of H.264/AVC Entropy Decoder Without Internal ROM/RAM Memories”, IEEE ISCCSP, 2008.

[MME2004] R. Melhem, D. Moss´e, and E. (Mootaz) Elnozahy, “The interplay of power management and fault recovery in realtime systems”, IEEE Trans. on Computers, pp. 217–231, 2004.

[MOI1996] H. Mehta, R. M. Owens, and M. J. Irwin, “Energy characterization based on clustering”, IEEE Design Automation Conference (DAC’96), pp. 702-707, 1996.

[MRH$^+$2007] B. Mochocki, D. Rajan, X.S. Hu, C. Poellabacer, K. Otten, and T. Chantem, “Network-Aware Dynamic Voltage and Frequency Scaling”, Real Time and Embedded Technology and Application Symposium, pp. 215-224, 2007.

[MS1979] Maybeck, Peter S., “Stochastic Models, Estimation, and Control”, Volume 1, Academic Press, Inc. 1979.

[MSB+2008] M.C. Maury, A. Shah, F. Blajojevic, D.S. Nikolopoulos, B.R. de Supinski, M schulz, “Prediction models for multi-dimensional power-performance optimization on many cores”, ACM PACT, pp. 250-259, 2008.

[MSS+2003] G. Magklis, M. Scott, G. Semeraro, D. Albonesi, and S. Dropsho, “Profile-based Dynamic Voltage and Frequency Scaling for a Multiple Clock Domain Microprocessor”, In Proceedings of the 30th International Symposium on Computer Architecture (ISCA-30), pp. 14-27, 2003.

[MWK+2006] T. Y. Morad, U. C. Weiser, A. Kolodnyt, M. Valero, E. Ayguade, “Performance, power efficiency and scalability of asymmetric cluster chip multiprocessors”, Computer Architecture Letters, Vol. 5, No. 1, pp. 14-17, 2006.

[NIS+2010] D.Nikolov, Urban Ingelsson, Virendra Singh and Erik Larsson, "Estimating Error-probability and its Application for Optimizing Roll-back Recovery with Checkpointing", IEEE DELTA, pp. 281-285, 2010.

[NIS+2011] Dimitar Nikolov, Urban Ingelsson, Virendra Singh and Erik Larsson, “Level of Confidence Study for Roll-back Recovery with Checkpointing”, IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops pp.59-64, 2011.

[NKH+2006] P. Nagpurkar, C. Krintz, M. Hind, P. Sweeney, and V. Rajan, “Online Phase Detection Algorithms”, In Proceedings of the International Symposium on Code Generation and Optimization (CGO), pp. 111-123, 2006.

[OH2005] K. Olukotun and L. Hammond, “The Future of Microprocessors”, ACM Queue, pp. 26-29, 2005.

[OW1997] A.V.Oppenheim, and A.S.Willsky, “Signals and Systems”, 2nd Edition, Prentice Hall, Upper Saddle River, NJ, 1997.

[P1991] Athanasios Papoulis, “Probability, Random Variables and Stochastic Processes”, 3rd Edition, McGraw-Hill, 1991.

[P2001] Peyton Z. Peebles , “Probability, Random Variables And Random Signal Principles”, McGraw-Hill 2001.

[PB2004] E. Pinheiro and R. Bianchini, “Energy Conservation Techniques for Disk Array based Servers,” in International Conference on Supercomputing, pp. 68-78, 2004.

[PBB1998] T. Pering, T. Burd, and R. Brodersen, “Dynamic voltage scaling and the design of a low-power microprocessor system”, Power Driven Micro architecture Workshop, attached to ISCA98, 1998.

[PBB2000] T. Pering, T.Burd, R. Brodersen, “Voltage scheduling in the IpARM Microprocessor system”, International Symposium on Low Power Electronics and Design (ISLPED '00), pp. 96-101, 2000.

[PKG2001] D. Ponomarev, G. Kucuk, and K. Ghose, “Reducing Power Requirements of instruction Scheduling Through Dynamic Allocation of Multiple Datapath Resources”, In Proceedings of the 34th Annual International Symposium on Microarchitecture (MICRO-34), pp. 90-101, 2001.

[PL2007] N. Pettis and Y.-H. Lu, “Improving Quality-of-Service of File Migration Power Management Policies in High-Performance Servers,” in International Conference on Parallel and Distributed Systems, pp. 1-8, 2007.

[PLM$^+$2012] V. Petrucci, O. Loques, D. Mosse, R. Melhem, N.A. Gazala, S. Gobriel, “Thread Assignment Optimization with Real-Time Performance and Memory Bandwidth Guarantees for Energy-Efficient Heterogeneous Multi-core Systems”, Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 263-272, 2012.

[PS2001] P. Pillai and K. G. Shin, “Real-time Dynamic Voltage Scaling for Low-power Embedded Operating Systems”, in ACM Symposium on Operating Systems Principles, pp. 89-102, 2001.

[R1987] B. Ripley, “Stochastic Simulation”, John Wiley & Sons, 1987.

[R2000] Rambus Inc., “Rambus 128/144-Mbit Direct RDRAM data sheet,” June 2000.

[R2003] Iain E.G. Richardson, “H.264 and MPEG-4 Video Compression – video coding for next generation multimedia”, John Wiley & Sons, 2003.

[RAB$^+$2005] Martino Ruggiero, Andrea Acquaviva, Davide Bertozzi, Luca Benini, “Application-Specific Power-Aware Workload Allocation for Voltage Scalable MPSoC Platforms”, International Conference on Computer Design, pp. 87-93, 2005.

[RGA$^+$2006] Martino Ruggiero, Pari Gioia, Guerri Alessio, Luca Benini, Milano Michela, Davide Bertozzi, Alexandru Andrei, “A Cooperative, Accurate Solving Framework for Optimal Allocation, Scheduling and Frequency Selection on Energy-Efficient MPSoCs,System on Chip”, International Symposium pn Chip, 2006.

[RJ1998] J. Russell and M. Jacome, “Software power estimation and optimization for high performance, 32-bit embedded processors”, International Conference on Computer Design, pp. 328-333, 1998.

[RLF2007] B. Rountree, D. K. Lowenthal, S. Funk, V. W. Freeh, B. R. de Supinski, and M. Schulz, “Bounding energy consumption in large-scale mpi programs”, ACM/IEEE conference on Supercomputing, pp. 1–9, 2007.

[RLS+2009] B. Rountree, D. K. Lownenthal, B. R. de Supinski, M. Schulz, V.W. Freeh, and T. Bletsch, “Adagio: making dvs practical for complex hpc applications”, International conference on Supercomputing, pp. 460–469, 2009.

[RML+2001] R. Ronen, A. Mendelson, K. Lai, S.-L. Lu, F. Pollack, and J. P. Shen, “Coming Challenges in Microarchitecture and Architecture”, Proceedings of the IEEE, pp. 325–340, 2001.

[RW1994] C. Ruemmler and J.Wilkes, “An Introduction to Disk Drive Modeling,” IEEE Computer, vol. 27, pp. 17-28, 1994.

[RWB2009] K. K. Rangan, G.-Y. Wei, and D. Brooks, “Thread Motion: Fine-Grained Power Management for Multi-Core Systems”, In Proceedings of the 36th Annual International symposium on Computer architecture, pp 302-313, 2009.

[S2005] R. Schmidt, “Liquid Cooling is Back”, Electronics Cooling, 11(3), Aug. 2005.

[S2007] Per Stenström, “The Paradigm Shift to Multi-cores: Opportunities and Challenges”, International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, 2007.

[SA2005] L. Spracklen and S. G. Abraham, “Chip Multithreading: Opportunities and Challenges”, In 11th International Symposium on High Performance Computer Architecture (HPCA-11), pp 248-252, 2005.

[SAC2010] R.A. Shafik, B. M. Al-Hashimi and K. Chakrabarty , “Soft error-aware design optimization of low power and time-constrained embedded systems”, IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010.

[SAK+2009] R. A. Shafik, B. M. Al-Hashimi, S.Kundu, A.Ejlali “Soft Error-Aware Voltage Scaling Technique for Power Minimization in Application-Specific MPSoC”, Journal of Low Power Electronics (JOLPE), 2009, pp. 145-156.

[SGM2008] R.Sridharan, N.Gupta and R.N.Mahapatra, “Feedback-controlled reliability-aware power management for real-time embedded systems”, IEEE Design Automation Conference (DAC), pp.185-190, 2008.

[SIK+2007] H. Sasaki, Y. Ikeda, M. Kondo, and H. Nakamura, “An intra-task DVFS technique based on statistical analysis of hardware events”, in Conference On Computing Frontiers, pp. 123–130, 2007.

[SKL2001] D. Shin, J. Kim, and S. Lee, “Low-Energy Intra-Task Voltage Scheduling Using Static Timing Analysis”, IEEE Design Automation Conference (DAC’01), pp. 438-443, 2001.

[SM2010] R. Sridharan, R. Mahapatra, “Reliability aware power management for dual-processor real-time embedded system”, IEEE Design Automation Conference (DAC), pp. 819-824, 2010.

[SMB$^+$2002] G. Semeraro, G. Magklis, R. Balasubramonian, D. Albonesi, S. Dwarkadas, and M. Scott, “Energy-Efficient Processor Design Using Multiple Clock Domains with Dynamic Voltage and Frequency Scaling”, In Proceedings of the 8th International Symposium on High-Performance Computer Architecture (HPCA-8), pp.29-42, 2002.

[SNT1994] T. Sato, M. Nagamatsu, and H. Tago, “Power and Performance Simulator: ESP and Its Applications for 100 MIPS/W Class RISC Design”, International Symposium on Low Power Electronics and Design (ISLPED), 1994.

[SPC2001] T. Sherwood, E. Perelman, and B. Calder, “Basic block distribution analysis to find periodic behavior and simulation points in applications”, International Conference on Parallel Architectures and Compilation Techniques, pp. 3-14, 2001.

[SPE2000] http://www.spec.org/benchmarks.html, Standard Performance Evaluation Cororation, SPEC CPU 2000.

[SPH$^+$2002] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, “Automatically Characterizing Large Scale Program Behavior”, Tenth International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 45-57, 2002.

[SS2003] A. Srivastava and D. Sylvester, “Minimizing Total Power by Simultaneous Vdd/Vth Assignment”, Asia South Pacific design automation conference, pp. 400-403, 2003.

[SSC2003] T. Sherwood, S. Sair, and B. Calder, “Phase tracking and prediction”, International Symposium on Computer Architecture (ISCA-30), pp. 336-349, 2003.

[SSF2000] S. Sudharsanan, Sriram, P. , Frederickson, H. , Gulati, A. , “Image and video processing using MAJC 5200”, IEEE International Conference on Image Processing, 2000.

[STA+2002] M. Schmitz, Marcus T, Al-Hashimi, Bashir M and Eles, Petru, “Energy-Efficient Mapping and Scheduling for DVS Enabled Distributed Embedded Systems”, IEEE Design, Automation and Test in Europe Conference (DATE2002), pp. 514-521, 2002.

[SYK2001] Donghwan Son, Chansu Yu, Heung-Nam Kim, “Dynamic voltage scaling on MPEG Decoding”, International Conference on Parallel and Distributed Systems (ICPADS 2001), pp. 633- 640, 2001.

[SYM+2007] N. Sakran, M. Yuffe, M. Mehalel, J. Doweck, E. Knoll, and A. Kovacs, “Implementation of the 65nm Dual-Core 64b Merom Processor”, IEEE International Solid-State Circuits Conference (ISSCC 2007), 2007.

[T2001] Harry L. Van Trees, “Detection, Estimation, and Modulation Theory, Part I”, John Wiley & Sons-Interscience, 2001 .

[T2006] G. Theocharous, “Machine Learning for Adaptive Power Management,” Intel Technology Journal, Vol. 10, Issue 4, pp.299-310, 2006.

[TMW1994] V. Tiwari, S.Malik, and A.Wolfe, “Power analysis of embedded software: A first step towards software power minimization”, IEEE Transactions on VLSI Systems, pp. 437–445, 1994.

[TRJ2003] T. K. Tan, A. Raghunathan, and N. K. Jha, “Software Architectural Transformations: A New Approach to Low Energy Embedded Software”,IEEE conference on Design, Automation and Test in Europe (DATE’03), pp. 146-151, 2003.

[U2006] United States Environmental Protection Agency, “ENERGY STAR Program Requirements for Computers”, Version 4.0. Oct. 2006.

[VGJ2003] Ankush verma , Brinda Ganeshc and Bruce Jacob , “Control-theoratic approach to dynamic voltage scheduling”, International conference on Compilers architecture and synthesis for embedded systems, pp. 255-266, 2003.

[VGS+2003] Ankush Varma, Brinda Ganesh, Mainak Sen, Suchismita Roy Choudhury, Lakshmi Srinivasan, and Bruce Jacob, “Control theoratic approach to dynamic voltage scheduling“, CASES, pp. 255-266, 2003.

[VJG2003] Van der Tol, E., Jaspers, E., Gelderblom, R, “ Mapping of H.264 Decoding on a Multiprocessor Architecture”, In Proc. SPIE Conf. on Image and Video Communications and Processing, pp. 707-718, 2003.

[W2008] Kai-Chiang Wu , “Power-aware soft error hardening via selective voltage scaling”, IEEE International Conference on Computer Design, 2008.

[WAS2010] Jonathan A. Winter, David H. Albonesi, Christine A. Shoemaker, "Scalable Thread Scheduling and Global Power Management for Heterogeneous Many-Core Architectures", International Conference on Parallel Architecture and Compilation Techniques (PACT 2010), 2010.

[WB2001] Greg Welch and Gary Bishop , "An Introduction to Kalman Filter", http://www.cs.unc.edu/~welch, 2001.

[WB2002] A.Weissel and F. Bellosa, "Process cruise control: Event-driven clock scaling for dynamic power management", International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES 2002), pp. 238-246, 2002

[WBB2002] A. Weissel, B. Beutel, and F. Bellosa, "Cooperative IO- A Novel IO Semantics for Energy-Aware Applications," in Operating Systems Design and Implementation, pp. 117-130, 2002.

[WJM$^+$2005] Q. Wu, P. Juang, M. Martonosi, and D.W. Clark, "Voltage and Frequency Control with Adaptive Reaction Time in Multiple-Clock Domain Processors," in Proc. of Symposium on High-Performance Computer Architecture, pp. 178-189, 2005.

[WS2003] Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra, "Overview of the H.264/AVC Video Coding Standard", IEEE Transactions on Circuits and Systems for Video Technology, 2003.

[WWD$^+$1994] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy", Preceding of the First USENIX Symposium on Operating Systems Design and Implementation (OSDI '94), pp. 13-23, 1994.

[XL2006] C. Xian and Y.-H. Lu, "Energy Reduction by Workload Adaptation in a Multi-Process Environment," IEEE Design Automation and Test in Europe, pp. 514–519, 2006.

[XLL2007] C. Xian, Y.-H. Lu, and Z. Li, "Adaptive Computation Offloading for Energy Conservation on Battery-Powered Systems," International Conference on Parallel and Distributed Systems, pp. 1-8, 2007.

[XMM2003] F. Xie, M. Martonosi, and S. Malik, "Compile-Time Dynamic Voltage Scaling Settings: Opportunities and Limits", ACM Conference on Programming Language Design and Implementation (PLDI 2003), pp. 49-62, 2003.

[XMM2005] F. Xie, M. Martonosi, and S. Malik, “Bounds on Power Savings Using Runtime Dynamic Voltage Scaling: an Exact Algorithm and a Linear-Time Heuristic Approximation”, in International Symposium on Low Power Electronics and Design, pp. 287-292, 2005.

[XTS+2008] Feng Xia, Yu-Chu Tian, Y. Sen, J. Dong, “Control-theoretic dynamic voltage scaling for embedded controllers”, IET Computers and Digital Techniques, pp. 377-385, 2008.

[YDT2008] S. Yaldiz, A. Demir and S. Tasiran, “Stochastic modeling and optimization for energy management in multicore systems: A video decoding case study”, IEEE trans. on computer aided design, pp. 1264-1277, 2008 .

[YN2006] W. Yuan and K. Nahrstedt, “Energy-Efficient CPU Scheduling for Multimedia Applications,” ACM Transactions on Computer Systems, vol. 24, pp. 292-331, 2006.

[ZBS2005] Huichai Zhang ,M.V. Basin ; M. Skliar, “Kalman Filter for Continuous State-Space System with Continuous, Multirate, Randomly Sampled and Delayed Measurement”, American Control Conference, 2005.

[ZC2003] Y. Zhang and K. Chakrabarty, “Energy-aware adaptive checkpointing in embedded real-time systems. IEEE Design, Automation and Test in Europe (DATE), 2003.

[ZC2008] D.Zhu,H.Aydin and J.J Chen, “Optimistic Reliability Aware Energy Management for Real-Time Tasks with Probabilistic Execution Times”, IEEE RTSS, pp. 313-322, 2008.

[ZEL+2002] H. Zeng, C. S. Ellis, A. R. Lebeck, and A. Vahdat, “ECOSystem: Managing Energy As A First Class Operating System Resource,” in International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 123-132, 2002.

[ZM2004] Y. Zhu and F. Mueller, “Feedback EDF Scheduling Exploiting Dynamic Voltage Scaling”, IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 84-93, 2004.

[ZMM2004] D. Zhu, R. Melhem, and D. Moss´e, “The effects of energy management on reliability in real-time embedded systems”, In Proc. of the Int’l Conf. on Computer Aidded Design (ICCAD), 2004.

[ZSG+2003] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R.Wang, “Modeling Hard-Disk Power Consumption,” Conference on File and Storage Technologies, pp. 217-230, 2003.