Co-Simulating Region-Based Dynamic Voltage Scaling for FPGA Architecture Design

Johannes Pfau*, Jiro Hernandez*, Maximilian Reuter†, Klaus Hofmann† and Jürgen Becker*
*Institut für Technik der Informationsverarbeitung, Karlsruher Institut für Technologie
†Integrierte Elektronische Systeme, Technische Universität Darmstadt
Email: pfau@kit.edu, maximilian.reuter@ies.tu-darmstadt.de,
klaus.hofmann@ies.tu-darmstadt.de, becker@kit.edu

***Abstract*—With rising importance of low power technology, researchers explore FPGA power management ranging from static leakage reduction to Dynamic Voltage Scaling (DVS). VPR, the framework commonly used for FPGA architecture exploration, provides power estimation for static approaches, but assumes operating conditions to be constant both temporally and spatially. This limitation prevents analysis of both DVS and fine grain, region based power management.**

To remedy this, we propose power regions and a co-simulation framework for VPR. We demonstrate two strategies to statically reduce power in those regions: The a priori strategy specifies low-power regions ahead of time and considers them during application placement. The a posteriori strategy places the application normally, then determines available timing slack in each region to find regions suitable for low-power operation.

Integrating QuestaSim and VPR is proposed to support temporally varying operating conditions. This enables analysis of DVS schemes, where an application simulated in QuestaSim controls operating conditions for VPR, which determines architecture power. Furthermore, operating conditions can be modeled over time, enabling simulation of spatial and temporal external influences. Framework usage is demonstrated with simulation of process variation in an FPGA architecture consisting of 16 power regions. We show how two regions can be switched off entirely and that circuit delay in other regions can increase up to four times without causing timing violations for a given sample application.

***Index Terms*—Reconfigurable Logic, Design Tools, Power Estimation**

I. Introduction

In recent years, various power saving strategies for integrated circuits have been introduced to reduce energy demands and cope with excess heat. For Field Programmable Gate Arrays (FPGAs) in particular, the power saving mechanisms can be grouped into two categories: First, there are application centric power saving techniques. These include generic concepts such as reduction of overall resource usage in a design and extend to application specific high-level techniques, such as disabling unused parts of a design dynamically. Apart from those, there are techniques focusing on the circuit boards hosting the FPGA. One such technique is reduction of the FPGA core voltage, if reduced performance is acceptable. Common to all of those techniques is the requirement of explicit intervention of the application designer. The second category and focus of this publication are application independent power optimizations in FPGA architectures. Apart from well-established circuit and transistor-level optimizations for FPGAs, there are various system level power saving options. Those include scaling of the overall supply voltage of the FPGA or using body-biasing for a whole FPGA. Here, gains are limited by missing application knowledge. Commonly, choosing the right supply voltage or body bias voltage is therefore left to the application designer.

This work was supported by the German Research Foundation (DFG) within the PARFAIT2 project (DFG 326384402).

Using novel transistor technologies for FPGAs can potentially enable further power optimizations. Reconfigurable Field Effect Transistors (RFETs) provide secondary gates which can be used to influence the transistor's threshold voltage [1]–[8], enabling fine grain adjustment of the FPGA performance on a per-region level [9]. Dynamic systems perform these adjustments over time, measure available timing slack and adjust the voltage for a region accordingly [10]. Such approaches can also be used to counter process variation and aging.

To evaluate these power management schemes during FPGA architecture design, an FPGA architecture power estimator is required. For detailed results, the FPGA architecture can be realized in Electronic Design Automation (EDA) tools and comprehensive circuit power analysis can be performed, e.g. using SPICE simulators. Designing such a circuit representation is labor-intensive, and it can't be used for quick Design Space Exploration (DSE) of FPGA architectures. Alternatively, a system level power estimator such as the one included in Versatile Place and Route (VPR) may be used. The results are less exact, but do not require a complete circuit design for the FPGA architecture, allowing quick DSE. VPR however estimates power for the whole FPGA, assuming identical operating conditions in all FPGA regions. Furthermore, VPR always performs static simulation: It assumes operating conditions don't change over time, making simulation of dynamic power management schemes difficult. Additionally, to perform realistic power estimation in DVS, the FPGA user application needs to be simulated as well: The timing slack depends mainly on the static placement and routing of the application. But when dynamic compensation schemes are used or external influences such as temperature vary over time, a dynamic simulation of the application is required.

We therefore propose four extensions for VPR, providing a framework to efficiently estimate power for static and dynamic power management:

- Power domain regions: Allowing different areas in the FPGA to possess different operating conditions.
- A priori region assignment: Assign different operating conditions to different regions before placement.
- A posteriori region assignment: Calculate available slack in each region after placement and determine the lowest supply voltages which still achieve timing closure.
- VPR and QuestaSim co-simulation: Simulating DVS controllers in QuestaSim while estimating power in VPR.

II. Related Work

Various region based power management schemes for FPGAs have been described in literature. In [11], the authors present a dual power supply architecture and an algorithm to assign voltages depending on available slack, but do not explore different power region sizes. An approach for power gating is given in [12], whereas [13] describes placement for a coarse grain architecture with multiple power regions. With regard to DVS, [10] describes how to measure delay of the FPGA circuits and [14] provides experimental results for voltage adjustment based on such measurements.

All of those approaches are either static only, i.e. do not consider application power over time, or they have been evaluated experimentally, instead of in simulation. With only experimental evaluation, comparison and design space evaluation of different architectures is difficult. To the best of our knowledge, there is no system level simulator available to evaluate DVS schemes with programmed FPGA applications. Our framework fills this gap and allows to compare various power control strategies. For example, the framework can be used to compare different voltage adjustment algorithms or to simulate custom scenarios, such as aging, local temperature increase, and more.

III. FPGA Regions

To support locally varying operating conditions such as different supply voltages, we divide the FPGA into power regions. A power region is an area on the FPGA containing one or multiple primitive blocks, usually Configurable Logic Blocks (CLBs). Each region can be in one of multiple predefined modes, which usually map to operating conditions such as supply voltage levels. This technology independent approach can model various power management techniques, including power gating, body biasing, voltage scaling and backgate voltage adjustment for RFETs.

Primitive blocks, switches and other elements in the FPGA architecture will exhibit different propagation delays, depending on the operating conditions. The VPR architecture description format was therefore extended to map these different values to region modes. In addition, two changes were implemented in the VPR power estimator: Where property values are given in the architecture file, such as for *absolute* modeling where power values are specified, or where *pin-toggle* power is specified, those are changed to tables. For transistor level modeling such as the *auto-size* model, VPR

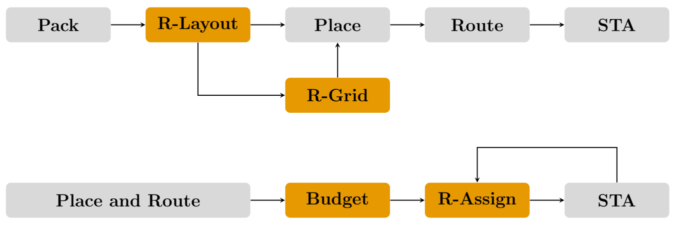


Fig. 1. Integration into VPR standard flow. Top: A priori approach. Bottom: A posteriori approach.

obtains per transistor data from a tech file. We extend this functionality to load different tech files depending on each region's current mode.

Lookup Table (LUT) and therefore CLB delays are largely dependent on the supply voltage and therefore the most critical component to consider when reducing operating voltage [15]. Clocking and routing networks are currently assumed to have an independent power supply. For logic block delay calculation, the algorithms use the delay from the architecture file according to the region mode. In the placement phase, we then ensure that the cost function also considers the now region-specific, adjusted delays. This is mainly important for a priori region assignment, where regions have a fixed, specified voltage during placement. Based on its current mode, each region's power usage for logic blocks is also estimated independently.

IV. A Priori Assignment

In this scheme, the region layout is predefined in the FPGA architecture description, including the initial mode each single region is in. VPR itself will not adjust the modes of the regions, but simply calculate the initial placement according to this grid. The top part of Fig. 1 shows how a new region layout generation stage is inserted in the VPR flow after packing. As VPR supports determining an application-specific FPGA size, FPGA size information is in general only available after packing has been completed. The *R-Layout* pass reads the architecture description to build the grid (*R-Grid*) and make it available for the placement phase. In the placement phase, VPR uses this information to place logic in low-power or high performance regions. For this, the placement cost function has been extended with a third term as shown below:

$$\Delta C = \lambda \frac{\Delta T_{\text{cost}}}{T_{\text{cost,prev}}} + (1-\lambda)\frac{\Delta W_{\text{cost}}}{W_{\text{cost,prev}}} + \mu \frac{\Delta P_{\text{cost}}}{P_{\text{cost,prev}}} \qquad (1)$$

T_{cost} and W_{cost} are unmodified timing and wiring cost components as previously used in VPR, P_{cost} is the newly added power component. As originally the case for T_{cost} and W_{cost}, a normalized difference of the values for new and old placements is used for P_{cost} as well.

V. A Posteriori Assignment

The bottom diagram in Fig. 1 shows the implementation of the a posteriori region assignment mode. In this approach, modes are assigned to regions after the design has been placed using the standard VPR flow. For the placement phase, there

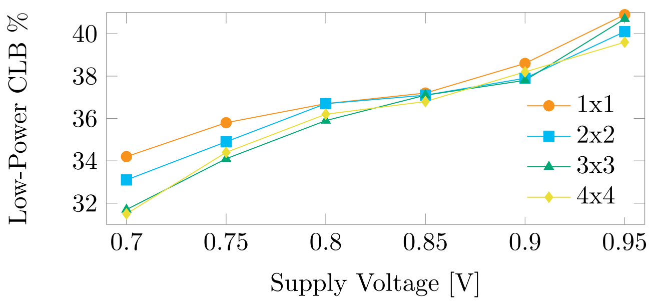


Fig. 2. A priory region assignment: Fraction of CLBs in low power mode for different region sizes and low power supply levels. The supply voltage in high-performance regions is 1 V.

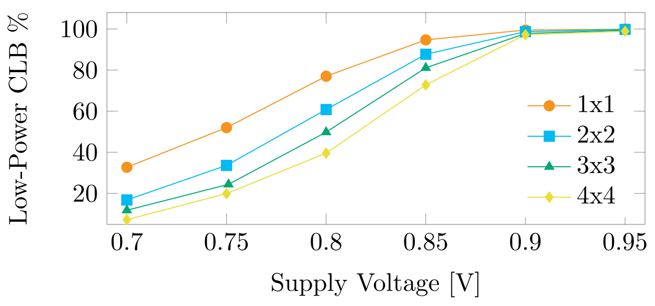


Fig. 3. A posteriori region assignment: Fraction of CLBs in low power mode for different region sizes and low power supply levels. The supply voltage in high-performance regions is 1 V.

are two options to consider: Either all regions are considered to be in low power mode when the design is placed. After placement, the voltages of regions containing critical paths are increased, until the paths meet the timing constraints. Alternatively, the design is placed assuming all regions are in high-performance mode. After the initial placement, the voltage for each region is reduced until the design fails to achieve timing closure. Experiments show that the latter approach yields better results. Our implementation of the algorithm largely follows the variant proposed by [11] for dual power supply FPGAs.

Two new processing phases are introduced in Fig. 1: Slack budget calculation and dynamic region assignment. In slack budget calculation, all paths going through a region are enumerated and their timing slacks are collected. The minimum slack is then assigned as the slack budget of that region. After determining the budgets of all regions, they are sorted accordingly and the dynamic region allocation phase is executed: VPR iterates through all regions, starting with those with the highest slack budget. It assigns a lower voltage to the region and recalculates the overall timing. If timing closure is still achieved, the change is accepted and processing continues with the next region. Otherwise, the change is reverted before continuing with the next region. When all regions are processed, the algorithm concludes with a final timing analysis.

VI. Static Analysis

To demonstrate static region approaches, we evaluate a simple architecture supporting two power modes. For the a priory strategy, the modes have been statically assigned in an alternating pattern. The architecture specification is based on the 40 nm *k6_frac_N10_40nm* architecture shipped with Verilog-to-Routing (VTR). It contains only basic logic elements and no memory, allowing easier comparison. To extend the architecture, we obtained delay values for various supply voltages in a similar 45 nm Predictive Technology Model (PTM) using COFFEE 2 [16] and HSPICE. Power and delay values obtained with 1 V supply voltage were used for the high performance power regions. For the low power regions, varying power and delay values were evaluated. As benchmarks for evaluation, we used the MCNC benchmarks [17] used by VPR. To summarize the results on system-level, we determine the amount of used CLBs which are in low-power mode and compare this to the baseline architecture. The result figures show averages for the evaluated benchmarks.

Fig. 2 depicts results for the a-priory strategy, showing that between 30 % and 40 % of the CLBs have been placed into low power regions. Larger region sizes, which need less hardware resources in implementation, do not strongly affect the amount of low-power CLBs in this evaluation. As the target frequency was not set (best-effort), these results come with a decrease of the maximum clock frequency of factor 1.1 at 0.95 V up to 1.8 at 0.7 V. Design space exploration of real architectures should evaluate more patterns for the regions and needs to assess whether setting a fixed target frequency is necessary.

Unlike a priori assignment, a posteriori assignment will not affect maximum application frequencies: The algorithm essentially modifies the distribution of timing slacks, reducing the slack of non-critical paths. For an exemplary evaluation, we slightly modify the architecture to specify only one region type with two possible voltage/power/delay combinations. The high performance mode continues to operate at 1 V and the low power mode uses varying voltage, power and delay. Fig. 3 shows the average amount of cells in low-power for the a posteriori strategy. It illustrates that a large amount of CLBs can be operated in low power mode, but results depend heavily on the parameters chosen: A low-power supply voltage of 0.9 V will enable almost all CLBs to be in low-power mode. At 0.75 V for most region sizes, less than 25 % of the CLBs are placed in this mode. The other factor largely influencing the results is region size: A finer granularity in voltage adjustments allows for better results, with per-CLB voltage adjustment yielding the best results. For practical FPGA design, such fine-grain voltage selection will have to balanced with the need of excessive additional resources.

VII. Dynamic Co-Simulation

Fig. 4 shows the co-simulation extensions to enable evaluation of DVS schemes. The overall simulation setup realizes a control loop, which can be used to evaluate the DVS controller or a user provided model of external influences, named *PVTA*

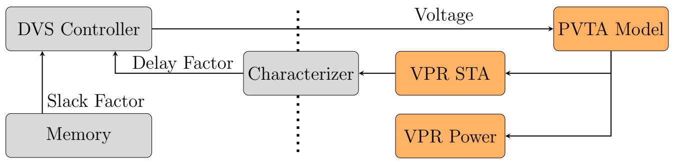


Fig. 4. The QuestaSim and VPR based co-simulation for dynamic, adaptive voltage scaling systems.

Model. The left-hand side of the figure is part of the simulated FPGA architecture and is modeled as HDL code using QuestaSim. It consists of the *DVS controller*, which adjusts the performance of regions dynamically, *Memory* which stores characterizations of user application timing requirements in each region, and the *Characterizer* which assesses the current performance within a region. The simulation framework does not dictate any concrete implementation of those components, giving maximum flexibility to user supplied models. Slacks for user applications in various regions are calculated using VPR when placing the application.

The right-hand side of Fig. 4 is part of the simulation environment, implemented in C++, and interfaces to the HDL code using QuestaSim's Foreign Language Interface (FLI). It takes the requested voltage from the architecture's *DVS Controller* and passes it to a user-provided *PVTA Model* written in LUA. The model calculates a location-dependent timing degradation factor to adjust the delay of each CLB. Using this information, a static timing analysis of the target application can be performed in VPR. This allows to assess the target application's timing requirements under changing operating conditions. The results are also forwarded to the *Characterizer*, which then estimates the current delay in each region.

For power analysis, model values are forwarded to the *VPR Power* estimator, most notably the current voltage for a region. Additionally, VPR's power analysis was extended with region support: It calculates per-region power for all primitive blocks. As previously mentioned, we support using different *tech* files for each region and point in time, enabling full flexibility regarding simulation of DVS systems.

VIII. Dynamic Analysis

In the following section, we will provide an example showing how the simulation framework can be used for FPGA architecture design. We introduce a DVS architecture, which dynamically measures delays in FPGA regions and adjusts the regions' supply voltages to match the user applications' required slacks in each region. To also demonstrate external influences, we describe how to model process variation in the framework. We limit the discussion to process variation because of the limited space for description of the Process-, Voltage- and Temperature Variation and Aging (PVTA) effects and simulation models. The simulator can be used for voltage variation, temperature variation and aging in the same way given the respective models.

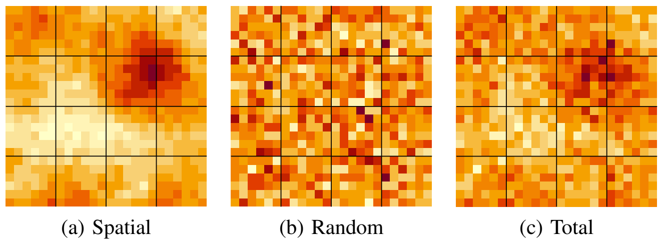


Fig. 5. Simulated intra-die process variation showing the transistor threshold voltage V_{th} for a 24x24 grid. The spatially correlated part is shown in (a), the uncorrelated part in (b) and (c) depicts the resulting total variation. Brighter color indicates smaller V_{th}.

Process Variation Introduction

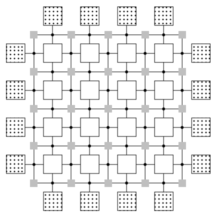
Process variation effects can be classified into certain categories [18], [19]: At first level, it can be categorized into systematic variation and non-systematic, random variation. Random variation can be further separated into inter-die variation and intra-die variation, where intra-die variation can be distinguished as pure random variation or spatially correlated variation. Systematic errors can be largely corrected during manufacturing [18], so we're not further considering them in this demonstration. Inter-die variation can be modeled as variation between the fabricated ICs later on, so we will only model intra-die variation here. Pure random intra-die variations are caused by random dopant fluctuations and line edge roughness [18]. Spatially correlated variation occurs in various processing steps, including lithography, etching and chemical mechanical polishing. The effects are mainly caused by these processes changing progressively across the chip [18].

Models for process variation are usually technology dependent. As the required technology parameters are not always available, this can make adaption of some models difficult. Certain models also require detailed modeling of the circuit for which process variation is analyzed, e.g. in SPICE. As a primary aim of our simulator is to quickly evaluate different architectures, we want to avoid this extended modeling here. We therefore use the VARIUS process variation model [20], as it requires few parameters and provides exemplary parameter values which match the empirical study in [21]. The simulation can be easily adapted to more advanced models such as [22]–[25], when parameter values for these are available. A detailed analysis of a single specific FPGA architecture could alternatively provide detailed implementations of basic building blocks such as LUTs and can use resulting advanced models. Variation models offered as part of Process Design Kits (PDKs) however often do not cover spatially correlated variation.

The VARIUS model considers two main factors in process variation, threshold voltage V_{th} and the effective gate length L_{eff}. Its authors note that variation in any parameter can be written as:

$$\Delta P = \Delta P_{\mathrm{D2D}} + \Delta P_{\mathrm{WID}} = \Delta P_{\mathrm{D2D}} + \Delta P_{\mathrm{rand}} + \Delta P_{\mathrm{sys}} \quad (2)$$

Where ΔP_{D2D} is inter-die variation, ΔP_{WID} intra-die variation


(a) Architecture

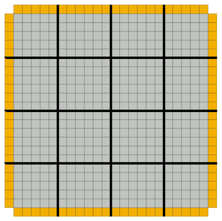

(b) 24x24 Grid

Fig. 6. VPR's *k6_frac_N10_40n* flagship architecture. Only CLBs are needed for the sample application, other blocks have been removed to improve clarity.

and ΔP_{rand} and ΔP_{sys} are the random and spatially correlated components of intra-die variation. VARIUS describes ΔP_{rand} and ΔP_{sys} as normal distributions and treats them as additive. ΔP_{D2D} can also be modeled as an additive, chip-wide offset. The spatial correlation between two points on the chip is then modelled using the correlation

$$corr(P_{\vec{x}}, P_{\vec{y}}) = \rho(r) \qquad r = |\vec{x} - \vec{y}| \tag{3}$$

and a spherical model for the correlation function $\rho(r)$:

$$\rho = \begin{cases} 1 - \frac{3r}{2\Phi} + \frac{r^3}{2\Phi^3} & (r \leq \Phi) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The model for spatially correlated L_{eff} is given as:

$$L_{\text{eff}} = L^0_{\text{eff}} \left(1 + \frac{V_{\text{th}} - V^0_{\text{th}}}{2V^0_{\text{th}}}\right) \tag{5}$$

Using this model and the authors' parameter values $\Phi = 0.5$, $\sigma/\mu = 0.063$ for random and systemic V_{th}, and $\sigma/\mu = 0.032$ for random L_{eff}, we obtained the variation maps for V_{th} in Fig. 5. The maps were generated for a 24x24 grid, which is the amount of FPGA CLBs used in this demonstration. Our simplified analysis here assumes that the grid spans the whole chip. If this is not the case, spatial correlation between neighboring CLBs is qualitatively larger.

Target FPGA Architecture

In this demonstration, we again use a modified *k6_frac_N10_40nm* FPGA architecture. As our benchmark application does not use any other blocks, we remove all but the CLBs and Input- / Output- Blocks (IOBs) for clarity. The resulting architecture is shown in Fig. 6a. We further assume the FPGA architecture provides some power vs. performance adjustment in 4x4 separate regions. Depending on the technology, such performance adjustment can be realized in different ways: Bulk technology enables power supply voltage adjustment or body biasing, using the body effect.

$$V_{\text{th}} = V_{\text{th0}} + \gamma \left(\sqrt{|2V_{\text{fp}}| + V_{\text{SB}}} - \sqrt{|2V_{\text{fp}}|}\right) \tag{6}$$

Eq. (6) shows how the threshold voltage is affected by substrate voltage V_{SB} [26]. The result depends on the technology specific factors V_{fp} as well as γ, and is non-linear. Furthermore, Silicon on Isolator (SOI) devices or other technology may require different modeling [27]. In the remaining demonstration, we assume the dependency has been modeled so that a factor can be used to linearly scale the threshold voltage.

For a system analysis, those transistor-level factors must be transformed to variation in logic cell delay. We again follow the VARIUS model

$$T_{\text{g}} \propto \frac{V\left(1 + V_{\text{th}}/V^0_{\text{th}}\right)}{\left(V - V_{\text{th}}\right)^{\alpha}} \tag{7}$$

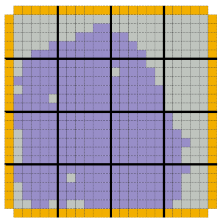
and use their observation that this equation is nearly linear regarding V_{th} in the relevant parameter range. This model is based on the analytical expression of inverter delay. Whereas more accurate delays could be obtained through SPICE simulation of the affected logic blocks, SPICE models usually do not consider spatial variation. For this demonstration, we therefore avoid technology specific SPICE analysis and use the VARIUS model to derive the degradation factor.

FPGA User Application

As target application, we use the *diffeq1* benchmark included in VPR and set the FPGA size to 24x24 as shown in Fig. 6b. VPR is then used to place the target application as shown in Fig. 7a. This placement does not utilize CLBs in two regions, so these can be powered off entirely. Based on the initial placement and independent of any technology factors, VPR then calculates the slack factor in each region: It first finds the routing path $c(r)$ of the placed application with the minimum slack in each region. The slack factor for region r is then calculated as

$$k_{\text{slack,r}} = \frac{\text{slack}_{c(r)} + \text{delay}_{c(r)}}{\text{delay}_{c(r)}} \tag{8}$$

This factor describes how much a path can be slowed down while still achieving timing closure. Fig. 7b shows the factors resulting for this demonstration. In this case, the critical path spans 7 regions, which therefore all have the same slack factor of 1.17. The remaining regions have varying factors from 1.39 to 4.73.


(a) Placement

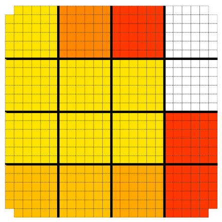
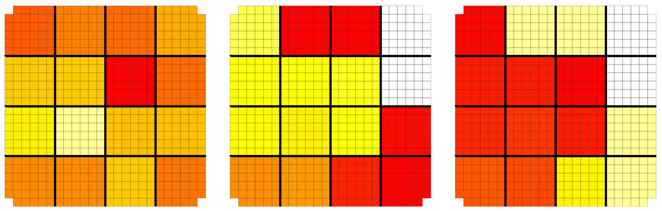

(b) Slack Factor

Fig. 7. User application placement and available slack in each region. (a) shows how the benchmark application has been placed onto the FPGA and into regions. (b) depicts the slack factor for each region. Brighter color indicates a lower factor, i.e. less available slack in the critical path. Unused regions are depicted in white color.



(a) t_0:Delay Factor (b) t_1:Delay Factor (c) t_1:Voltage

Fig. 8. Delay factors and region voltage at start and end of simulation. Unused regions are depicted in white color. (a) and (b) show the current delay factors at the respective point of time. Brighter color indicates a smaller factor, i.e. delays in the region are smaller compared to the nominal case. (c) shows the voltage factors used to achieve the delay factors in (b). Brighter color indicates a lower factor, i.e. the threshold voltage in that region has been increased. At the initial time t_0, all region factors are 1.0.

Simulation Results

In a practical realization, those values could be stored within the bitstream and provided to the target architecture. In the simulation case, our framework will provide this value to the region power controller written in VHDL or Verilog and simulated in QuestaSim. The architecture then needs to provide a way to characterize the actual delay in a region relative to the normal delay. Various hardware implementations for such characterizations have been presented in literature [28]–[35]. For the simulation, we implement a simple timing analysis for a fixed path (bottom left to top-right) in each region. The simulation determines the expected delay of the path under nominal conditions $p(r, nom)$ and at specific time points $p(r, t)$ using the currently applied voltage adjustment and process variation:

$$k_{\text{delay,r,t}} = \frac{\text{delay}_{p(r,t)}}{\text{delay}_{p(r,nom)}} \tag{9}$$

This factor can be used to scale path delays within a region to get the expected real delay. Whereas strictly speaking, this estimation is only valid for the measured path, due to spatial correlation other paths within a region are affected similarly. For the demonstration application, Fig. 8a and Fig. 8b show $k_{\text{delay,r,t}}$ at the beginning and at the end of the simulation. It can be seen that in the beginning, the measured delay largely conforms to the process variation map of Fig. 5c. After some time, the FPGA architecture's DVS controller (in this demonstration a simple proportional controller) has adjusted the power supply in each region. It therefore made the measured delay variation match the applications slack factor in Fig. 7b. The adjustment in the voltage factor needed to achieve this is shown in Fig. 8c. This result is mostly a combination of the process variation map and the slack factor, as the voltage adjustment has to both cancel the process variation and adapt the local performance to application requirements.

The simulation can be used to further determine the power savings for each region using the adapted VPR power estimator. In addition, our framework can be used to validate the assumption that the measured path is representative for the whole region: By performing additional static timing analysis for the target application at different time steps in the simulation, the slack values of all paths can be validated. The results for this demonstration are shown in Fig. 9, where regions are numbered from bottom-left to top-right. Analysis of the minimal slack in each region most importantly shows that all paths still achieve timing closure. It can also be seen that most critical paths have reduced slack, as expected. This also applies for the averages in most regions, showing that all paths are affected similarly. Another important observation is the fact that some regions even provide larger slack values. This is caused by the initial measurement being based on the initial VPR placement, which can not include device specific variation. In some cases, the device specific variation might however provide better performance in some regions than the worst case corner model that was used for this initial placement. Another observation can be made that the slack values of critical paths have not been reduced to zero. This is largely caused by some paths traversing more critical regions: For example, the worst path in region 3 could be scaled by factor 4.73. The analysis assumes it is scaled by this factor in all regions it traverses. However, the path also passes through region 2, which has a more critical path and can only be scaled by 1.59. The effect is also further explained by the characterized paths not being identical to the application paths.

IX. Conclusion

In this publication, we presented VPR extensions to enable power estimation of more complex power management schemes in FPGA architectures. We further introduced a framework for the evaluation of DVS schemes and presented an FPGA example architecture which compensates process variation effects using local supply voltage variation.

Our framework enables quick evaluation of various DVS schemes and design-space-exploration of dynamic power management FPGA architectures. Furthermore, easy integration of other PVTA models allows to obtain technology specific results with more specialized simulation models. For example, we intend to use this framework in future work to evaluate various architectures with real technology power characterizations derived from RFET semiconductor technologies.

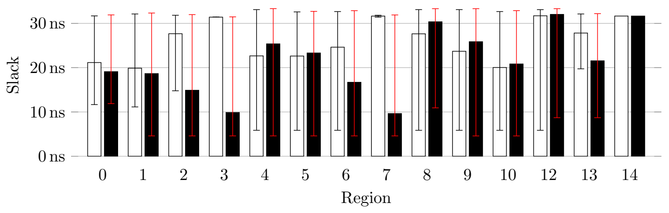


Fig. 9. Available slacks for each a region. Bars depict the mean over the slacks of all paths in a region and error bars show the minimum and maximum slack values. The nominal initial distribution after placement, ignoring the device specific variation model, is shown as white bars. Black bars show the final distribution, including the variation model and adjusted voltage factors at t_1.

References

[1] Y.-M. Lin, J. Appenzeller, J. Knoch, and P. Avouris, “High-performance carbon nanotube field-effect transistor with tunable polarities,” *IEEE transactions on nanotechnology*, vol. 4, no. 5, pp. 481–489, 2005.

[2] A. Heinzig, T. Mikolajick, J. Trommer, D. Grimm, and W. M. Weber, “Dually active silicon nanowire transistors and circuits with equal electron and hole transport,” *Nano letters*, vol. 13, no. 9, pp. 4176–4181, 2013.

[3] S.-M. Koo, Q. Li, M. D. Edelstein, C. A. Richter, and E. M. Vogel, “Enhanced Channel Modulation in Dual-Gated Silicon Nanowire Transistors,” *Nano Letters*, vol. 5, no. 12, pp. 2519–2523, 2005, pMID: 16351207. [Online]. Available: http://dx.doi.org/10.1021/nl051855i

[4] J. Trommer, A. Heinzig, U. Mühle, M. Löffler, A. Winzer, P. M. Jordan, J. Beister, T. Baldauf, M. Geidel, B. Adolphi *et al.*, “Enabling energy efficiency and polarity control in germanium nanowire transistors by individually gated nanojunctions,” *ACS nano*, vol. 11, no. 2, pp. 1704–1711, 2017.

[5] J. Zhang, M. De Marchi, D. Sacchetto, P.-E. Gaillardon, Y. Leblebici, and G. De Micheli, “Polarity-controllable silicon nanowire transistors with dual threshold voltages,” *IEEE Transactions on Electron Devices*, vol. 61, no. 11, pp. 3654–3660, 2014.

[6] S. Nakaharai, T. Iijima, S. Ogawa, S. Suzuki, K. Tsukagoshi, S. Sato, and N. Yokoyama, “Electrostatically-reversible polarity of dual-gated graphene transistors with he ion irradiated channel: Toward reconfigurable cmos applications,” in *2012 International Electron Devices Meeting*. IEEE, 2012, pp. 4–2.

[7] S. Nakaharai, M. Yamamoto, K. Ueno, Y.-F. Lin, S.-L. Li, and K. Tsukagoshi, “Electrostatically reversible polarity of ambipolar alpha-mote2 transistors,” *ACS nano*, vol. 9, no. 6, p. 5976—5983, Jun. 2015. [Online]. Available: https://doi.org/10.1021/acsnano.5b00736

[8] P. Wu, T. Ameen, H. Zhang, L. A. Bendersky, H. Ilatikhameneh, G. Klimeck, R. Rahman, A. V. Davydov, and J. Appenzeller, “Complementary black phosphorus tunneling field-effect transistors,” *ACS nano*, vol. 13, no. 1, pp. 377–385, 2018.

[9] J.-M. Park, J.-H. Bae, J.-H. Eum, S. H. Jin, B.-G. Park, and J.-H. Lee, “High-density reconfigurable devices with programmable bottom-gate array,” *IEEE Electron Device Letters*, vol. 38, no. 5, pp. 564–567, 2017.

[10] C. Chow, L. Tsui, P. Leong, W. Luk, and S. Wilton, “Dynamic voltage scaling for commercial fpgas,” in *Proceedings. 2005 IEEE International Conference on Field-Programmable Technology, 2005.*, 2005, pp. 173–180.

[11] A. Gayasen, K. Lee, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and T. Tuan, “A dual-vddlow power fpga architecture,” in *Field Programmable Logic and Application*. Springer Berlin Heidelberg, 2004, pp. 145–157.

[12] A. A. M. Bsoul and S. J. E. Wilton, “An fpga architecture supporting dynamically controlled power gating,” in *2010 International Conference on Field-Programmable Technology*, 2010, pp. 1–8.

[13] C. Li, Y. Dong, and T. Watanabe, “New power-efficient fpga design combining with region-constrained placement and multiple power domains,” in *2011 IEEE 9th International New Circuits and systems conference*, 2011, pp. 69–72.

[14] J. M. Levine, E. Stott, and P. Y. Cheung, “Dynamic voltage & frequency scaling with online slack measurement,” in *Proceedings of the 2014 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA ’14. New York, NY, USA: Association for Computing Machinery, 2014, p. 65–74.

[15] I. Ahmed, L. L. Shen, and V. Betz, “Optimizing fpga logic circuitry for variable voltage supplies,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 4, pp. 890–903, 2020.

[16] S. Yazdanshenas and V. Betz, “Coffe 2: Automatic modelling and optimization of complex and heterogeneous fpga architectures,” *ACM Trans. Reconfigurable Technol. Syst.*, vol. 12, no. 1, jan 2019.

[17] S. Yang, “Logic synthesis and optimization benchmarks user guide,” *Technical Report 1991-IWLS-UG-Saeyang*, 1991.

[18] V. Champac and J. Garcia Gervacio, *Timing Performance of Nanometer Digital Circuits Under Process Variations*. Cham: Springer International Publishing, 2018, vol. 39.

[19] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer, “Statistical timing analysis: From basic principles to state of the art,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 4, pp. 589–607, 2008.

[20] S. R. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, “Varius: A model of process variation and resulting timing errors for microarchitects,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 21, no. 1, pp. 3–13, 2008.

[21] A. Keshavarzi, S. Duvall, J. Brews, V. De, G. Schrom, S. Tang, S. Ma, K. Bowman, S. Tyagi, K. Zhang, T. Linton, and N. Hakim, “Measurements and modeling of intrinsic fluctuations in mosfet threshold voltage,” in *Proceedings of the 2005 international symposium on Low power electronics and design - ISLPED ’05*, K. Roy and V. Tiwari, Eds. New York, New York, USA: ACM Press, 2005, p. 26.

[22] H. Ghasemzadeh Mohammadi, P.-E. Gaillardon, and G. de Micheli, “Efficient statistical parameter selection for nonlinear modeling of process/performance variation,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 12, pp. 1995–2007, 2016.

[23] A. Lange, C. Sohrmann, R. Jancke, J. Haase, B. Cheng, A. Asenov, and U. Schlichtmann, “Multivariate modeling of variability supporting non-gaussian and correlated parameters,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 2, pp. 197–210, 2016.

[24] J. Freeley, D. Mishagli, T. Brazil, and E. Blokhina, “Statistical simulations of delay propagation in large scale circuits using graph traversal and kernel function decomposition,” in *2018 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. IEEE, 2018, pp. 213–219.

[25] P. Manfredi and R. Trinchero, “A probabilistic machine learning approach for the uncertainty quantification of electronic circuits based on gaussian process regression,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 8, pp. 2638–2651, 2022.

[26] R. J. Baker, *CMOS circuit design, layout, and simulation*, fourth edition ed., ser. IEEE Press series on microelectronic systems. Piscataway, NJ and Hoboken, New Jersey: IEEE Press and Wiley, 2019, vol. 22.

[27] S. Clerc, T. Di Gilio, and A. Cathelin, Eds., *The Fourth Terminal: Benefits of Body-Biasing Techniques for FDSOI Circuits and Systems*, 1st ed., ser. Integrated Circuits and Systems. Cham: Springer International Publishing and Imprint Springer, 2020.

[28] V. von Kaenel, P. Macken, and M. Degrauwe, “A voltage reduction technique for battery-operated systems,” *IEEE Journal of Solid-State Circuits*, vol. 25, no. 5, pp. 1136–1140, 1990.

[29] T. D. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen, “A dynamic voltage scaled microprocessor system,” *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1571–1580, 2000.

[30] M. Agarwal, B. C. Paul, M. Zhang, and S. Mitra, “Circuit failure prediction and its application to transistor aging,” in *25th IEEE VLSI Test Symmposium (VTS’07)*. IEEE, 2007, pp. 277–286.

[31] M. Fojtik, D. Fick, Y. Kim, N. Pinckney, D. M. Harris, D. Blaauw, and D. Sylvester, “Bubble razor: Eliminating timing margins in an arm cortex-m3 processor in 45 nm cmos using architecturally independent error detection and correction,” *IEEE Journal of Solid-State Circuits*, vol. 48, no. 1, pp. 66–81, 2013.

[32] V. Huard, F. Cacho, F. Giner, M. Saliva, A. Benhassain, D. Patel, N. Torres, S. Naudet, A. Jain, and C. Parthasarathy, “Adaptive wearout management with in-situ aging monitors,” in *2014 IEEE International Reliability Physics Symposium*. IEEE, 2014, pp. 6B.4.1–6B.4.11.

[33] M. Altieri Scarpato, “Digital circuit performance estimation under pvt and aging effects,” Theses, Université Grenoble Alpes, 2017. [Online]. Available: https://theses.hal.science/tel-01773745

[34] I. Miro-Panades, E. Beigne, O. Billoint, and Y. Thonnart, “In-situ fmax/vmin tracking for energy efficiency and reliability optimization,” in *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2017, pp. 96–99.

[35] D. Sengupta and S. S. Sapatnekar, “Estimating circuit aging due to bti and hci using ring-oscillator-based sensors,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 10, pp. 1688–1701, 2017.