Received January 18, 2019, accepted February 13, 2019, date of publication February 21, 2019, date of current version March 26, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2900477

Energy Optimization for Large-Scale 3D Manycores in the Dark-Silicon Era

SOHAIB MAJZOUB[1], (Senior Member, IEEE), RESVE A. SALEH[2], (Fellow, IEEE), IMRAN ASHRAF[3], (Member, IEEE), MOTTAQIALLAH TAOUIL[3], (Member, IEEE), AND SAID HAMDIOUI[3], (Senior Member, IEEE)

[1]Department of Electrical and Computer Engineering, University of Sharjah, Sharjah 27272, United Arab Emirates
[2]Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada
[3]Laboratory of Computer Engineering, Delft University of Technology, 2628 Delft, The Netherlands

Corresponding author: Sohaib Majzoub (smajzoub@sharjah.ac.ae)

ABSTRACT In this paper, we study the impact of the idle/dynamic power consumption ratio on the effectiveness of a multi-V_{dd}/frequency manycore design. We propose a new tool called LVSiM (a Low-Power and Variation-Aware Manycore Simulator) to carry out the experiments. It is a novel manycore simulator targeted towards low-power optimization methods including within-die process and workload variations. LVSiM provides a holistic platform for multi-V_{dd}/frequency voltage island analysis, optimization, and design. It provides a tool for the early design exploration stage to analyze large-scale manycores with a given number of cores on 3D-stacked layers, network-on-chip communication busses, technology parameters, voltage and frequency values, and power grid parameters, using a variety of different optimization methods. LVSiM has been calibrated with Sniper/McPAT at a nominal frequency, and then the energy-delay-product (EDP) numbers were compared after frequency scaling. The average error is shown to be 10% after frequency scaling, which is sufficient for our purposes. The experiments in this work are carried out for different Idle/Dynamic ratios considering 1260 benchmarks with task sizes ranging from 4000 to 16 000 executing on 3200 cores. The best configurations are shown to produce on average 20.7% to 24.6% EDP savings compared to the nominal configuration. Traditional scheduling methods are used in the nominal configuration with the unused cores switched off. In addition, we show that, as the Idle/Dynamic ratio increases, the multi-V_{dd}/frequency approach becomes less effective. In the case of a high Idle/Dynamic ratio, the minimum EDP can be achieved through switching off unused cores as opposed to using a multi-V_{dd}/frequency approach. This conclusion is important, especially in the dark-silicon era, where switching cores on and/or off as needed is a common practice.

INDEX TERMS 3D-stacked chip, dark-silicon, dynamic power, energy-delay-product, frequency scaling, idle power, low-power design, manycore, multicore, process variation, simulator, voltage scaling, voltage selection, within-die variation.

I. INTRODUCTION

Over the last decade, we have witnessed a major shift in processor design from a single complex monolithic processor towards the manycore design paradigm, which uses a large array of simpler processors. The manycore design provides a promising solution to numerous design challenges. Technology scaling issues, process variation, thermal impact, power density, and the market demand for battery-operated devices are all major obstacles that hinder performance and power budget improvements. However, manycore designs provide better localized control over power and thermal impact. The manycore design also relies on throughput instead of raw processor speed for performance improvement [1]–[10].

There are many compute-intensive problems in areas of machine learning that can benefit greatly using manycore designs, but power consumption of such designs is a big problem. In the current mobile computing era, power consumption of a thousand-core chip is one of the key challenges. In addition, the process variability of small feature technologies is creating irregular distributions of power and speed among cores [11]–[13]. Furthermore, as technology

The associate editor coordinating the review of this manuscript and approving it for publication was Gian Domenico Licciardo.

continues to scale down to 10nm and below, the dark-silicon phenomenon becomes more prominent [2], [14]. Dark silicon is defined by the inevitable fact that a large portion of the chip may have to be switched-off, i.e., ''dark'' all the time, to meet the power budget. The active part is determined based on the workload requirements. The needed apparatus to switch cores on and off is now part of the chip that controls the power consumption. Another common practice is the use of multi-voltage and frequency techniques to ''dim down'' cores that are running non-critical tasks, and this is the subject of our paper

The contribution of this paper is twofold. The first contribution is towards the efficacy of multi-V_{dd}/frequency design with respect to the Idle/Dynamic power ratio. A multi-V_{dd}/frequency design is shown to be effective primarily when the dynamic power dominates over the idle power, and we seek to quantify this effect. Specifically, we find that it is effective when the ratio is two times higher or above. However, if the idle power is the dominant power component, an On/Off core switching mechanism is sufficient to produce the best power savings. This mechanism is in line with system-level energy optimization methods that exploit the dark silicon phenomenon. Extensive experiments are carried out in this work to demonstrate this outcome.

The second contribution is LVSiM which was used to carry out the analysis. It is a manycore simulator that follows a holistic simulation approach. The nature of such a manycore platform is necessarily complex. In our view, the analysis should be carried out from the application perspective, considering system, microarchitecture, and circuit, and device level issues, to deliver a proper evaluation [6], [15]–[21]. Many proposed simulators attempt to tackle only a few of these aspects at a time. For instance, Sniper/McPAT [22] is a power, area and timing model that is combined with a multicore simulator for application and architecture development. Others, such as [9], [23], and [24], target low power and/or speed exploration and modelling. However, a holistic simulation environment that includes the aforementioned factors is needed. Similar to most application-level simulators, the main challenge is always the tradeoff between accuracy and complexity [25]. This could be even more challenging for large-scale manycore simulators when dealing with thousands of cores [26], [27]. In the literature, The suggestion has been made to use simple high-level core modelling and shift the details into other elements or issues under investigation, such as system-level power and performance evaluation in the early stages of the design cycle [28]. In the case of LVSiM, we follow a similar approach to focus on system-level assessment of low-power techniques while including process variations for manycore designs. LVSiM can be used as a vehicle to carry out system-level experiments targeting low-power methods under process, voltage, and temperature variations for large-scale manycores. Unlike existing simulators, LVSiM is developed targeting large number of cores with simple modelling methods to be used in early design stages for initial evaluation.

To establish a baseline, we compared the power numbers calculated using Sniper/McPAT [22] and LVSiM. In particular, LVSiM is first calibrated using the nominal power numbers extracted from Sniper/McPAT. Then, LVSiM experiments are carried out with frequency scaling. The power numbers produced by LVSiM after frequency scaling are then compared with Sniper/McPAT and the accuracy results are reported herein.

Beyond the improvements in accuracy, some of the techniques used in LVSiM are the improvements of the work done in [10]. Crucial changes have been implemented in LVSiM to improve simulation speed and to provide a complete simulation platform. LVSiM is programmed using the C language, as opposed to MATLAB, which was used in the prior work, to improve simulation speed and to provide a free open-source tool to the research community. Another key change is eliminating the use of genetic algorithms (GA) as a data routing optimization method and instead using XYZ-routing [29] to improve simulation speed. The capacity of the tool to handle larger applications and more cores was increased. Specifically, two optimization rounds were removed as a result of removing GA. This allowed the scaling up of problem sizes from 1000 to 16,000 tasks, and an increase in the manycore sizes from 1600 up to 3200 cores. Scaling to these levels permitted LVSiM to use improved and novel approaches to tackle low-power optimization for large-scale problems.

This paper is organized as follows. Section II presents motivation and review of previous work. Section III presents the low power optimization techniques available in the simulator. Mapping applications to cores is discussed in Section IV. Section V presents the process variation modeling used by LVSiM. Section VI describes the LVSiM simulator platform. Finally, Section VII provides the experimental setup, with calibration to Sniper/McPAT, and summarizes the results and findings.

II. RELATED WORK

In this section, we review existing manycore simulators targeted towards low power and process variation. Of particular note in the description of previous work are the power models, problems sizes, and complexity of the problems. The size of any optimization problem dealing with large-scale manycore platforms presents a major issue. Unlike traditional simulators, a manycore simulator requires special consideration of complexity versus accuracy. A complex model might provide high accuracy, but it may result in prohibitively high simulation time [25]–[27]. Thus, the model must be targeted towards specific aspects, such as low-power and performance evaluation, simulation time, area estimation, cache policy, network topology, etc. This provides a focused direction for the relevant optimization techniques with an acceptable accuracy-speed compromise [25], [26]–[30]. These tradeoffs are evident in most of the prior work.

Binkert *et al.* [15] developed the gem5 simulator. It is a merger of two simulators, namely M5, which provides a configurable processor framework, and GEMS, a memory design simulator. The gem5 tool provides a full system with detailed instruction-set architecture (ISA) and memory models. It also utilizes a parallel discrete-event simulation (PDES) environment to improve simulation speed. It is mostly used to explore cache and memory design in a multicore platform. It has been reported that it has poor performance, especially when a large number of cores are used [18], [31]. Certner *et al.* [17] proposed a manycore simulator supporting scalability and fast simulation time. They used a discrete-event simulator model to support up to 1024 cores.

In [16], Carlson *et al.* presented Sniper. The simulator runs in parallel on a multicore design to speed up the simulation time. The number of cores considered in the work was up to 16 cores. Li *et al.* [32] proposed a power, area, and timing model, namely McPAT. The model is used for design space exploration of multicore platforms. At the system-level, the model includes a network-on-chip (NoC), shared caches, memory controllers, and multiple domain clocking. The critical-path timing, area, dynamic, short-circuit, and leakage power are calculated at the circuit-level. Different device types including bulk CMOS, SOI and dual-gate transistors are used. The McPAT model has to be integrated with a manycore simulator for power and performance estimation. The authors demonstrated a multicore design with up to 64 cores.

Kim *et al.* [33] presented an imitation learning (IL) vs. reinforcement learning (RL) comparison to improve the efficiency of dynamic voltage and frequency scaling of voltage islands in a manycore platform. The authors used a combination of gem5 [15] and McPAT [32] as a platform to carry out the experiments. The paper uses a maximum of 64 cores in their manycore simulation environment.

Heirman *et al.* [22] described a multicore hardware/software design exploration platform that combines Sniper [16], and McPAT power modeling [32], with custom DRAM power models. The authors indicate that the proposed platform is fast for multicore design because it uses analytical models. They showed that this platform could predict timing performance and power numbers with absolute errors of around 22% and 8%, respectively. The paper noted that the maximum number of cores considered is 16 cores.

Kodaka *et al.* [9] developed a method to predict the needed number of cores to satisfy workload changes. The method optimizes for low power without any performance degradation. The authors used dynamic voltage and frequency scaling (DVFS) and power gating to achieve their goal. They used a maximum of 32 cores to demonstrate their method.

Lai *et al.* presented PoweRock [23], a flexible Dynamic Power Management (DPM) tool. The authors proposed a profile-guided DVFS with a power prediction model and a scalable architecture. PoweRock produced roughly 37% and 25% of energy and energy-delay product (EDP) savings, respectively. The number of cores considered in the experiments was 48 cores.

Cai *et al.* [34] proposed a model to minimize energy consumption under performance constraints. They also considered process variations in the work. They state that the power reduction obtained is 31% and the throughput increase is 11%. The number of cores considered in the experiments was up to 128 cores. Stamelakos *et al.* [35] proposed a dual-voltage platform to mitigate process variation impact in a manycore design operated at near-threshold (NT) supply. A dual-voltage rail (DVR) is used to power the cores. The paper notes a 50% improvement in performance under the same power budget. The maximum number of cores considered was 128. Lee and Kim [36] addressed the issue of low power under process variation. The paper discusses replacing core speed by increased throughput. The authors show a 65% reduction in power for highly parallel applications. The number of cores considered was up to 8 cores. Miller *et al.* [37] present another DVR approach to mitigating the core-to-core variation in which the authors show a 50% improvement in performance with the same power budget. They used a 64-core platform.

Drego *et al.* [38] used a near-optimal search algorithm to select a proper voltage value (of two available voltages) to mitigate core-to-core speed variation. The authors show 6% to 16% in energy savings. The paper also addresses the switched off cores to save on energy while meeting the performance constraints. The proposed methodology assumed manycore platforms with 100 and 1000 cores. Rahmani *et al.* [39] proposed a multi-objective dynamic power management for NoC. The method uses fine-grained voltage and frequency scaling and power gating considering core reliability. The authors claim to have minimized the aging effects and to have extended the core lifetime and boosted the overall throughput. The paper considered a manycore design with 144 cores to demonstrate the proposed method.

Jeyapaul *et al.* [40] proposed UnSync-CMP, a customizable and redundant Chip Multiprocessor (CMP). The platform uses redundancy to handle cache soft errors. The proposed platform shows a 34.5% power reduction, 20% speed improvement, and 13.3% less area overhead. The authors used a platform with 8 cores. Mercati *et al.* proposed in [41] multi-rate predictive controllers to dynamically adjust the GPU computational resources to maximize energy savings while meeting the timing target. The paper claims a 25% energy saving compared to existing methods without performance overhead.

Many other papers present different schemes to address power reduction using different simulators and methods for voltage and frequency scaling. Some of these papers addressed other issues such as scalability, memory hierarchy, thermal issues, and process variations [8], [11], [24], [39], [41]–[54].

III. LOW POWER OPTIMIZATION IN LVSiM

We propose a new open-source tool called LVSiM[1] (a Low-Power and Variation-Aware Simulator for Manycores) to

[1]http://dx.doi.org/10.21227/tnc7-3d33

address low-power optimization, including within-die process and workload variations. LVSiM provides a holistic platform for multi-V_{dd}/frequency voltage island designs for large-scale manycore designs. The problem of assigning voltage and frequency values to cores, scheduling tasks into time slots without conflicts and routing traffic to satisfy the power and performance requirement is an NP-hard problem [10]. Scaling up the problem into thousands of cores to deliver the required timing deadlines of thousands of tasks within a limited power budget requires new methodologies and development platforms. In this section, we discuss the applicability of existing methods to large-scale manycores, and propose new methods to handle the power reduction problem.

At one extreme, the voltage and frequency assignment to cores can be fixed prior to chip fabrication without possible changes after fabrication. This method has a simple power delivery system. It is usually used when the processor is designed for a specific application and its workload is known before fabrication. This method may not be suitable for a processor with manycore capabilities as it has very limited flexibility. The other extreme case would be to dynamically change the supply voltage and the frequency for each core, i.e., DVFS. If this method is implemented at the core level, it incurs considerable timing and energy overhead, and high design complexity [8], [10], [12], [34], [42], [55]–[58].

An intermediate solution is to limit the number of voltages or frequencies to a small number and allow an energy-aware operating system to select the proper values for each core. If needed, the core supply can be adjusted when switching between different applications. This reduces the timing and energy overhead and simplifies the circuit design. This compromise method provides the needed flexibility with acceptable design complexity for manycore platforms. It can be realized using a multi-voltage rail system and a multi-clocking network. Each voltage-frequency domain (VFD) consists of cores using the same voltage and frequency values.

A typical method described in the literature for multicore is to map the application onto cores and then attempt to slow the cores down, i.e., reduce the voltage and/or frequency of cores to eliminate any slack [10]. Thus, the voltage and frequency values can be specified for each core after scheduling tasks on cores. However, this does not scale well with large manycore designs. When the traffic cost is also included, the complexity of the optimization problem becomes very high. It is more appropriate to set the cores' voltage and frequency (based on estimating the application needs) before application mapping. Then, tasks with the same slack are grouped together and executed on cores within the same VFD. This approach is used in LVSiM.

A. POWER AND DELAY MODELING

Accurate power and delay modeling of cores in a large-scale manycore design is important. However, the power model complexity can create a bottleneck for acceptable simulation speed. Thus, we use the alpha-power model to model critical path delay in voltage/delay scaling. The equation is given by:

$$D = \frac{C_o}{k} \times \frac{V_{dd}/2}{(V_{dd} - V_t)^{\alpha}} \tag{1}$$

where $\boldsymbol{C_o}$ is the capacitive load along the critical path, $\boldsymbol{k}$ is the technology-dependent variable, $\boldsymbol{V_{dd}}$ is the supply voltage, $\boldsymbol{V_t}$ is the threshold voltage and $\boldsymbol{\alpha}$ is a tuning parameter. The dynamic and idle power, total core power, router and link power are modelled as follows:

$$\begin{aligned} P_{dynamic} &= C_L V_{dd}^2 (\beta . f) && (2)\\ P_{idle} &= (I_o e^{-\frac{qV_t}{nkT}}) V_{dd} && (3)\\ P_{core} &= P_{dynamic} + P_{idle} + P_{shifters} && (4)\\ P_{router,\ link} &= P_{router} + P_{link} + P_{shifters} && (5) \end{aligned}$$

where $\boldsymbol{C_L}$ is the capacitive load of the core, $(\boldsymbol{\beta.f})$ is the average switching rate, $\boldsymbol{I_o}$, $\boldsymbol{q}$, $\boldsymbol{n}$, and $\boldsymbol{k}$ are known technology parameters and constants, $\boldsymbol{T}$ is the temperature, $\boldsymbol{P_{shifters}}$ is the power consumed by level shifters between domains, and $\boldsymbol{P_{router,link}}$ is the power consumed by the router and links at every cross-section of the Network-on-Chip. The energy and the Energy-Delay-Product (EDP) for the given power equations are then calculated using the following equations:

$$\begin{aligned} Energy &= Power \times Core\ Cycle\ Time && (6)\\ EDP &= Energy \times Number\ of\ Cycles && (7) \end{aligned}$$

The given models are sufficient for our purposes and have been used extensively in the literature to validate different system-level power and delay optimization techniques [23], [26], [35], [36], [49], [59], [60]. In our case, we calibrate our initial power numbers to Sniper/McPAT prior to the analysis to ensure that the results deliver acceptable accuracy, as described later in section A.

B. CORE REDUCTION PROBLEM

A key step in the optimization process is to determine the number of cores for each Voltage-Frequency Domain (VFD). A VFD is defined as a set of cores with the same voltage and frequency values, as shown in equation (8), where $\boldsymbol{C}$ is any core with voltage and frequency $\boldsymbol{V_i}$ and $\boldsymbol{F_j}$, respectively.

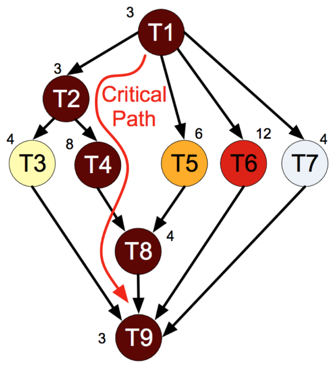
$$VFD_{ij} = \left\{ \forall C : C \in (V_i, F_j) \right\} \tag{8}$$

A typical method used for multicores is to schedule tasks with the same slack on as many cores as are available to meet the timing deadline. This would be without any constraints on the number of cores used for each VFD. Instead of keeping cores idle, it is assumed that it is better to use these cores with reduced voltage and frequency values to run non-critical tasks. This was acceptable in small multicores before the dark-silicon problem.

In the dark silicon era, some cores have to be switched off to maintain the total power budget. The core reduction problem is defined as maximizing the off cores to reduce the power while meeting the performance requirements. Specifically, it is defined as follows: a set of cores $\boldsymbol{C}$ within

VFD_{ij} are identified to be switched off. The tasks running on these switched off cores have to migrate into a higher order VFD, i.e., with higher voltage and frequency values, to maintain the timing deadline. This core reduction step is meant to minimize the number of cores that are being used while the timing deadline is met.

FIGURE 1 and FIGURE 2 illustrate the nature of the problem. Assume that for the given standard task graph (STG) of FIGURE 1. with tasks T1-T9, the voltages of the cores are scaled down to eliminate the associated timing slacks shown in the table of Fig. 1. An initial implementation of the task graph using multi-V_{dd} is shown in Fig. 2 (a) with each core having its own voltage values. Note that each VFD here is assumed to have a single core in this example for illustrative purposes only. Specifically, the five VFDs are {C_1}, {C_2}, {C_3}, {C_4}, and {C_5}. However, another implementation without voltage scaling is also possible with fewer cores as shown in FIGURE 2 (b). Thus, it is possible to migrate tasks



Task	Time Before Scaling	Slack	Time after Scaling
T1	3	0	3
T2	3	0	3
T3	4	8	12
T4	8	0	8
T5	6	5	11
T6	12	3	15
T7	4	11	15
T8	4	0	4
T9	3	0	3

Traffic	12 Edges
Parallelism Factor	47/21 = 2.23
Critical Path	21

FIGURE 1. A task graph, with task timing and dependencies.

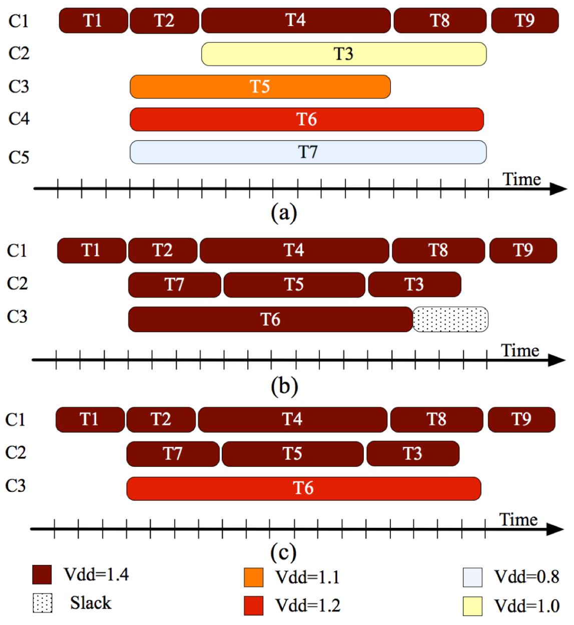


FIGURE 2. Core reduction and voltage scaling (a) typical mapping after voltage scaling, (b) optimized mapping without voltage scaling (c) optimized mapping with voltage scaling.

T_3, ***T_5*** and ***T_7*** to ***C_1***, without any voltage scaling, and then switch off ***C_4*** and ***C_5***. In this case, switching off two cores provides more power savings than powering all cores with voltage scaling. The final VFDs are {***C_1***}, {***C_2***}, and {***C_3***}. Although On/Off switching can result in timing and power overhead, it is an inevitable necessity in the dark-silicon era. Voltage scaling can be implemented after switching off the maximum number of cores as shown in FIGURE 2 (c).

C. CORE REDUCTION OPTIMIZATION

The core reduction and optimization described above is carried out as follows. Initially, tasks with the same slack time, i.e., those utilizing the same voltage and frequency values with no timing overlaps, are scheduled on the same core within a given VFD. The outcome of this step is multiple VFDs, each with a number of cores running a number of tasks. Then, the number of cores within a given VFD is reduced. A core within a VFD can be removed if all its tasks can be scheduled onto another core in a higher level VFD (i.e., with higher voltage or frequency values, so that the timing deadline is always met), assuming that the incremental power increase is minimal. The optimization target is to minimize the number of cores per VFD. This step is referred to as the Core Reduction step. However, this can be implemented in a number of different ways.

One of the following four approaches can be selected in LVSiM for core reduction during low power optimization:

1- ***Core to Multiple Domains (C2MD):*** tasks of removed core can be scheduled on any destination core of any destination VFD.

2- ***Core to Single Domain (C2SD):*** tasks of removed core have to be scheduled on any, but all within the same destination VFD.

3- ***Core to Single Core (C2SC):*** tasks of the removed core have to be scheduled on any, but all within the same destination core in the same destination VFD.

4- ***Tasks to Multiple Domains (T2MD):*** tasks of the to-be-reduced VFD are scheduled on any destination core of any destination VFD.

The methods used in core reduction range from rigid to highly flexible. But it is not clear at the outset which method will be the best overall. However, all methods require that the destination VFD be of a higher V_{dd}/frequency order to preserve performance. The number of cores, tasks, and VFDs greatly affects the optimization speed and outcomes. C2MD selects a core to be removed from the source VFD based on the lowest power overhead, i.e., the minimum power increase if this core is removed. It removes the core and moves its tasks to a higher level VFD such that the power is kept at minimum and the performance is not affected. The tasks of the removed core do not have to be scheduled on the same core or in the same VFD, hence the notion of multiple domains. This method gives the freedom to the tasks of the removed core to be scheduled anywhere, as long as the power overhead is minimal.

Another option is to limit the scheduling of the tasks to the same VFD. Thus, the C2SD method forces all tasks of the removed core to be scheduled in the same destination VFD. Otherwise the core cannot be removed. The third method, namely C2SC, forces all tasks of the removed core to be scheduled on the same destination core in the same destination VFD. Finally, the last method, namely T2MD, is the most flexible of all methods. It does not look at cores within the VFD. Instead, it attempts to remove as many tasks as possible from a given source VFD to any higher level destination VFD. In the process, many cores will be removed from the source VFD. After the optimization is completed, all VFDs are expected to have the minimum number of cores to execute all tasks and to satisfy the timing deadline.

D. VFD REDUCTION PROBLEM

Initially, the cores within a VFD are assigned voltage and frequency values selected from user-defined voltage and frequency lists. The assignment is based on the available slack for each task running on these cores. The number of voltages and frequencies used by cores is then reduced to a smaller number. This is because only a limited number of voltages and frequencies can be implemented in the actual hardware, a problem usually referred to as the voltage and frequency selection problem [10]. Alternatively, it is referred to as VFD reduction, or VFD merging. The work described below extends the Removal Cost Method (RCM) proposed in our prior work [10] to perform VFD.

First, the power consumption of a given VFD is calculated. Then, the ***removal cost*** (***RC***) of a given (voltage, frequency) pair is defined as shown in equation (9):

$$RC_{ij} = NTP\left[VFD\left(V_i, F_j\right) \rightarrow VFD(V_k, F_l)\right] - CTP \quad (9)$$

where ***NTP*** and ***CTP*** stand for the **New Total Power** and **Current Total Power**, respectively. The ***RC*** is defined as the increase in total power consumption if two domains, $\boldsymbol{VFD(V_i,F_j)}$ and $\boldsymbol{VFD(V_k, F_l)}$, are merged together into the higher level domain (or $\boldsymbol{VFD(V_i, F_j)}$ is removed), $\boldsymbol{VFD(V_k,F_l)}$, where the voltage $\boldsymbol{V_k > V_i}$ and the frequency $\boldsymbol{F_l > F_j}$ to satisfy the timing deadline.

The removal cost is calculated for all available VFDs. Then, the VFD with the minimal cost is removed. In other words, a VFD with a given voltage/frequency value is removed if it has the minimal impact on power consumption, i.e., minimal power increase if removed. After removing $\boldsymbol{VFD(V_i,F_j)}$, its tasks are added to $\boldsymbol{VFD(V_k, F_l)}$. The number of cores in $\boldsymbol{VFD(V_k,F_l)}$may increase as a result of allocating more tasks to this domain.

Given the initial number of available voltages and frequencies, $\boldsymbol{N_v}$ and $\boldsymbol{N_f}$, respectively, the total number of VFDs is equal to $\boldsymbol{N_v * N_f}$. Looping through all VFD might be computationally expensive. Thus, LVSiM provides a compromise for improving the speed. This step can be done in three different ways: reducing voltages and then reducing frequencies (VFR), reducing frequencies and then voltages (FVR), or reducing individual VFDs (SVFR). For example, in the VFR approach, the user can choose to reduce the voltages first. This way the removal cost is going to be for a given voltage value instead of a single VFD. Thus, all VFD with the same voltage are going to be combined into one removal cost value. The voltage with the minimum removal cost is removed. After the voltages are reduced into the required number, the frequencies are reduced in the same way in a consecutive loop. A similar method is used in FVR, except that the frequency-based reduction is carried out first, and then voltages. The SVFR is the most computationally expensive approach as it uses the combined approach.

IV. APPLICATION MAPPING ONTO CORES

In this section, the methods used to schedule tasks on to cores are discussed. LVSiM provides different approaches to evaluate the priority of each task during scheduling, and the proper core location to execute this task, as discussed below.

A. TASK SCHEDULING PRIORITY

The scheduler used in LVSiM is a modified As-Soon-As-Possible (ASAP) scheduler. Tasks are scheduled on cores as soon as the data and control dependencies are satisfied. The scheduler prioritizes ready tasks based on which one should go first. There are two types of priorities: user-defined STG priority and scheduler-defined task priority. The tasks of a higher STG priority are given higher precedence while scheduling over the rest of the tasks.

Task priority is applied within the same STG. Tasks on the critical path have to be scheduled first as they have the highest priority. Non-critical tasks are prioritized using different methods based on available slack and/or based on the number of successors (i.e., task's children). Although all available slacks of all tasks are supposed to be eliminated during the voltage and frequency scaling, some will inevitably remain after the core and VFD reduction steps. Tasks with more slack are assigned lower priority. Alternatively, a task with more successors should also be considered to have a higher priority. Rather than specifying the scheduling method in LVSiM, one of the following five prioritization options may be chosen by the user:

1- ***Basic Task Priority (BTP):*** Ready tasks are scheduled based on their order of appearance in the task list after random shuffling, without any consideration of available slack or number of children.
2- ***Children-based Task Priority (CTP):*** After random shuffling, ready tasks with more children are scheduled first. Prioritizing with respect to the number-of-children assumes that finishing a task (with more children) would allow more successor tasks to be scheduled.
3- ***Slack-based Task Priority (STP):*** After random shuffling, ready tasks with smaller slack times are scheduled first.
4- ***Slack-Children Task Priority (SCTP):*** After random shuffling, tasks with smaller slack times are scheduled

first. Those with the same slack are scheduled based on the number of children.

5- ***Children-Slack Task Priority or CSTP:*** After random shuffling, ready tasks with more children are scheduled first. Those with the same number of children are scheduled based on the slack.

Tasks are selected for scheduling based on task priority. The root tasks, i.e., tasks with no predecessors, are scheduled first. Once a core is selected for a given task, the task delay is recalculated to reflect the new changes due to the impact of process and voltage variations on the selected core. XYZ-routing is used to handle data traffic [29], [61].

B. TASK CORE SELECTION

A core has to be assigned to run a given task. It has to belong to the proper VFD, such that the timing requirements of the given task are met. Root tasks are scheduled on any available core. Ideally, successor tasks should be scheduled as close as possible to their predecessors to minimize traffic time. The core used for a successor task is selected based on the center of gravity of all predecessors. Since the XYZ-routing is used, the value of the X, Y, and Z coordinates of the selected core are calculated using the average traffic/distance from the cores of all predecessor tasks (pre-tasks). The amount of communication between a given task and its pre-tasks is used as a weighting factor when calculating the coordinates of the core. If the core at the calculated coordinates does not belong to the proper VFD, the closest core within the proper VFD is selected. Three possible methods can be selected as criteria for core selection. The following three options are available in LVSiM:

1- ***Any Free Core (AFC):*** tasks are scheduled on any free core in the order of appearance.

2- ***Center-of-Gravity Core (CoGC):*** the core is selected based on the center of gravity. If the core is busy, then the task is not scheduled within the current cycle time and the task must wait for a core to be available.

3- ***Center-of-Gravity with Waiting List (CoGWL):*** the core is selected based on the center of gravity. If the core is busy, the task is pushed into a FIFO waiting list for this core.

V. PROCESS VARIATION CALCULATION

The manycore paradigm is only possible with small-feature technologies, where billions of transistors are fabricated on the same chip. Consequently, as the dimensions of the transistors get smaller, the precision at which the transistor can be fabricated has deteriorated significantly. In a manycore platform, the problem manifests itself in speed and power discrepancy among cores. Many papers proposed methods to mitigate the impact of process variation. Furthermore, thermal impact due to core overuse, and voltage drop due to workload distribution, are other important issues to consider as well.

Process variation is calculated using its two components: systematic and random. The systematic component is calculated using a multivariate normal distribution. The spatial correlation due to systematic effects is captured using a distance dependent model with a spherical correlation shape function [10], [62].

The manycore is divided into smaller regions, where each core is comprised of a given number of these regions specified in the simulator configuration file. Each region has its normal distribution for V_t (threshold voltage) and L_{eff} (effective gate length) with 0 mean and standard deviation σ_{sys}. The random component is represented by standard deviation σ_{rand}. The random and systematic components are then added together.

LVSiM can generate the process variation profiles (PVP) internally. Process variation parameters are user-defined. Each PVP file contains the normalized frequencies of the cores within the manycore design. The file also contains the mean threshold voltage for each core. The normalized frequency is used to calculate the core speed and dynamic power. The mean V_t is used during idle power calculations.

Temperature variation is generated using an external tool, namely HotSpot [63]. LVSiM generates the needed configuration files to be used by HotSpot. The simulator then invokes Hotspot to calculate the thermal impact numbers, where these numbers are read by the simulator. If HotSpot did not run for any reason, the thermal impact is ignored. In this paper, we do not use any thermal impact calculations, and HotSpot is not used, but the capability is available in the simulator.

The voltage variation due to workload is calculated using a resistive mesh, where the current drawn by a core is represented by a current source. The delay and power numbers are re-calculated during simulation using the calculated voltage [10].

VI. LVSiM SIMULATION ENVIRONMENT

In this section, we discuss the experimental setup utilizing the LVSiM simulation platform. The optimization flow used by LVSiM is shown in FIGURE 3. The simulator is configured

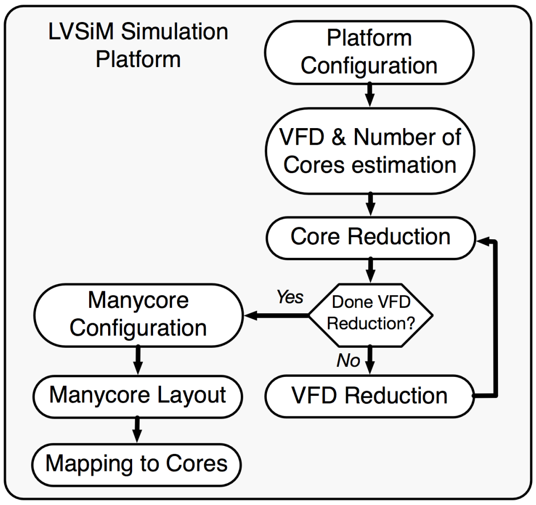


FIGURE 3. LVSiM simulation platform.

using the Simulator Configuration File (SCF). The SCF is the main configuration file used to set the simulation environment and parameters used to build the manycore design. In this file, the user can specify the links to application files, set the values of the process and voltage variation parameters, as well as nominal power and frequency numbers, voltage and frequency scaling parameters, and power optimization methods used during simulations.

As shown in FIGURE 4, seven files are read by LVSiM, some of which are mandatory while others are optional depending on user preference. The mandatory files are the Simulator Configuration File (SCF), and Standard Task Graph (STG). The optional files are the Process Variation Profile (PVP), Hotspot Configuration File (HCF), Core Voltage Layout (CVL), Core Frequency Layout (CFL), and finally Heterogeneity Specification File.

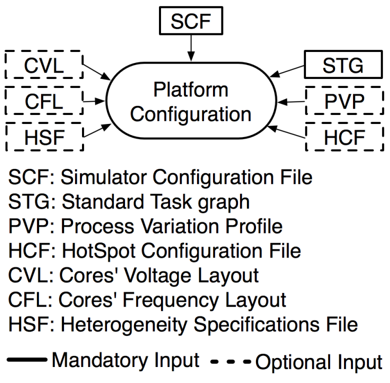

SCF
CVL
CFL
HSF
Platform Configuration
STG
PVP
HCF
SCF: Simulator Configuration File
STG: Standard Task graph
PVP: Process Variation Profile
HCF: HotSpot Configuration File
CVL: Cores' Voltage Layout
CFL: Cores' Frequency Layout
HSF: Heterogeneity Specifications File
Mandatory Input
Optional Input

FIGURE 4. Platform configuration input files.

In the next step, the application is analyzed to determine the initial power and speed characteristics. The tasks and the number of cores for each Voltage-Frequency Domain (VFD) are estimated. Then, a core reduction method is used to reduce the number of cores per VFD. One of the methods discussed in section III can be used to reduce the number of cores. The VFD reduction is then performed to reduce the number of domains in the design to a pre-defined number. The core reduction step is performed again to further reduce the number of cores.

The manycore size, number of layers, process variation impact on core's frequency, core's equivalent threshold voltage due to variation, and core's voltage drop values, are then used to create the manycore configuration. Next, the layout of VFD onto cores is performed. LVSiM provides two options for VFD layout in the SCF. The first option is stacking the VFD, **Stacked Domain Layout** (**SDL**), one after the other, starting with the one with the highest voltage and frequency values. For instance, assuming the first VFD to have ten cores, then the first ten cores of the first row are assigned to this VFD. The remaining VFDs are laid out in the same way. The second option is to alternate the voltage and frequency values for cores to cover all VFDs, referred to as **Alternating Domain Layout** (**ADL**). For instance, assuming two VFDs, the odd-numbered cores are assigned to the first VFD and the even-numbered cores are assigned to the second VFD.

In this version of LVSiM, the voltage and frequency values of the routers and links are going to either follow the cores' voltage and frequency or set to the nominal values. Internally, LVSiM uses separate variables for the NoC voltage and frequency values than that of the cores. This is in anticipation of having different optimization methods for the NoC in future versions.

Once the voltage and frequency are assigned to each core, the simulator schedules the tasks on cores. Each task has to execute on a free core with proper voltage and frequency values.

The simulator produces data and results in all stages. For instance, the simulator does voltage and frequency optimization for all tasks of the STGs. For each task, it saves the core used for execution, pre-tasks, real and assumed starting and finishing time, real execution time, power consumed, waiting time, and other relevant data. This data is saved into a separate file to keep a full record of the task execution behavior. The second file produced by the simulator saves the cores' activity during execution such as dynamic and idle power and frequency values of each core, number of times each router and link of the NoC is used. It also saves the final total power, STG data such as parallelism factor, critical path, total number of cores used, total number of cores switched off, and number of cycles took to finish execution. It saves the outcomes of each stage in a chronological order. Last but not least, if the user chooses to generate the PVP internally, a file for each PVP is generated based on the parameters specified in the SCF provided by the user.

VII. EXPERIMENTS AND ANALYSIS

In this section, we describe the experimental setup used in the simulation. First, LVSiM accuracy validation with respect to power estimation is presented. Then, we show the power saving results for large-scale benchmarks and discuss the outcomes. The main purpose of the experiments is to identify the set of options, or configurations, that provide the highest power savings. There are many options to choose from, and many combinations of these options. We chose to evaluate the most promising configurations for use in LVSiM. Another key question we address is how the ratio of Idle/Dynamic power affects the decision to use multiple VFDs or a single VFD. In particular, if there is little activity in a design, the leakage power would be high and it is better to use a single VFD and shut off unused cores, i.e., create areas of dark silicon. On the other hand, if many cores are busy, the dynamic power would be very high so the use of multiple VFD would be appropriate. We used LVSiM to explore and quantify this tradeoff.

A. LVSiM/SNIPER POWER NUMBERS' VALIDATION

In order to validate LVSiM, we compared the Energy-Delay-Product (EDP) numbers generated by LVSiM with Sniper/McPAT [22], [64], [65]. Note that Li *et al.* [65]

also used EDAP (Energy-Delay-Area-Product) and EDA^{2}P (Energy-Delay-Area^2-Product) as metrics, while we only used EDP. because LVSiM does not estimate the area. Sniper provides a middle ground between analytical architectural models and a cycle-accurate simulator. Analytical models tend to be fast but they do not capture all architectural details. On the other hand, cycle-accurate simulators utilize detailed architectural models to precisely simulate the architecture. However, doing so slows down the simulator, which limits the number of configurations that can be evaluated, especially for realistic workloads.

Sniper utilizes interval simulation techniques to improve its accuracy while maintaining simulation speed. Interval simulation raises the level of abstraction by replacing the cycle-accurate core-level simulation model with a mechanistic analytical model [66]. The analytical model estimates core-level performance by analyzing timing intervals between two events.

Sniper generates dynamic activity in the form of instruction-level statistics and utilizes the Multicore Power, Area, and Timing (McPAT) framework [65] for power and area modeling for manycore architectures. McPAT uses technology projections from International Technology Roadmap for Semiconductors (ITRS) for dynamic, idle, and short-circuit power. McPAT uses detailed models of various components of processors, i.e., cores, caches, NoC, and memory controllers. McPAT provides offline power and area estimates for full systems designed for various technologies.

As discussed earlier, LVSiM uses a much simpler simulation environment with basic models to calculate the energy, power and performance numbers as it is intended for use at a very early design stage. Thus, LVSiM requires a calibration cycle with an existing tool to deliver acceptable accuracy. LVSiM takes in the application abstracted as a standard-task-graph (STG) and maps it to cores. In order to generate STGs for realistic benchmarks, we utilized MCProf [67], which is an open-source, run-time memory and communication profiler. MCProf uses the Intel Pin [68] dynamic binary instrumentation framework to perform measurements at various granularity levels. MCProf tracks instructions, routines and memory accesses to maintain a producer-consumer relationship which can then be expressed as a flat-profile call-graph or a task-graph. Using a Python script, we converted this information to the required STG format which can be readily utilized by LVSiM to generate the power profile of a given application.

Table 1 lists the five benchmarks used to compare the power numbers from Sniper/McPAT and LVSiM. The total number of tasks is listed for each case. The critical path length, measured in cycles, is the processing time of the tasks that lie on the critical path. The parallelism factor is the sum of all task computation times divided by the critical path time.

The number of cores used in both LVSiM and Sniper platforms is 16. The nominal power was extracted from Sniper/McPAT for each of the five benchmarks. LVSiM was then calibrated through injecting the nominal power of a given

TABLE 1. Benchmarks used to compare LVSiM and Sniper/McPAT.

Benchmark Name and Description	Tasks	Critical Path Length (cycles)	Parallelism
vecOps: Vector Operations	10	300434	1.5
canny: edge detection algorithm	44	85139859	6.2
bmmult: matrix Multiplication	136	107400976	100.5
fft: Fast Fourier Transform	99	19631649	2.2
fluid: fluid dynamics solver for game engines [69]	294	1216152983	2.3

benchmark that was obtained into LVSiM's configuration file, namely the SCF (as discussed in section VI). Simulation using LVSiM was then carried out with frequency scaling and then the total EDP was computed. Consequently, the EDP difference error between LVSiM and Sniper/McPAT was calculated for the five benchmarks at seven different frequencies. The error difference between LVSiM and Sniper/McPAT are reported in FIGURE 5 and Table 2.

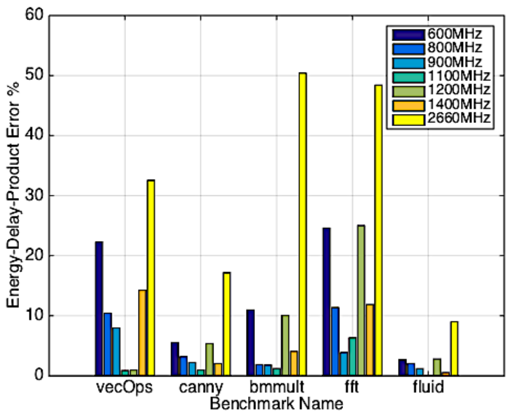


FIGURE 5. Sniper/McPAT versus LVSiM percentage EDP error, where 1000MHz is used for calibration.

The EDP error is shown after calibrating LVSiM with respect to Sniper/McPAT at a frequency of 1000MHz. The nominal power numbers were fixed after calibration and then the frequency was scaled to 600MHz, 800MHz, 900MHz, 1100MHz, 1200MHz, 1400MHz and 2660MHz. In general, the errors increase as the scaled frequency gets further from the nominal value, i.e., 1000MHz. The highest error is shown at 2660MHz, more than double the nominal frequency, for all benchmarks. Table 2 shows that the benchmark with the minimum error is the *fluid* benchmark, with an average error rate of 2.3%. The benchmark with the maximum error is the *fft* benchmark with an average error rate of 18.7%. Moreover, the frequency with the minimum error is the 1100MHz, with average error of 1.8%. The frequency with the maximum error is 2660MHz, with an average of 31.5%. Finally, the total average error across all benchmarks and frequencies is 10.2%, which is within acceptable bounds for use in LVSiM.

TABLE 2. Error % between LVSiM (calibrated at 1000MHz) and Sniper/McPAT.

Benchmark Name	Targeted Scaling Frequency (MHz)							Average Error % per Benchmark
	600	800	900	1100	1200	1400	2660	
vecOps	22%	10%	8%	0.8%	0.9%	14%	32.5%	12.3%
canny	5.5%	3%	2%	0.9%	5%	2%	17%	5%
bmmult	11%	1.8%	1.8%	1.2%	10%	4%	50%	11%
fft	24.6%	11%	3.8%	6.3%	25%	11%	48%	18.7%
fluid	2.6%	2%	1%	0%	2.7%	0.5%	9%	2.3%
Average Error % per Frequency	13%	5.7%	3.4%	1.8%	8.8%	6.5%	31.5%	Total Error Average = 10.2%

B. LARGE-SCALE EXPERIMENTS

The purpose of the work is to carry out experiments on very large manycore designs. This type of experimentation is not possible on other existing tools as discussed in section II. LVSiM is configured to run a 3D manycore architecture with 2 dies stacked on top of each other. Each die has a total of 40x40 cores, creating a platform with 3200 cores. Each layer has a two-dimensional router mesh. The routers are connected through vertical links to the upper layer to create the XYZ network-on-chip used by the simulator.

In earlier sections, we described the myriad of options available currently in LVSiM. The total number of different configurations that can be implemented by LVSiM is 360: 3 for VFD reduction (section III, D), 4 for core reduction (section III, C), 5 for task scheduling priority (section IV, A), 3 for core selection (section IV B), and 2 for core layout (section VI). This presents a problem, since it is not clear at the outset which combinations of options will emerge as the best for a given application. To reduce the size of the configuration space, we manually selected 38 meaningful configurations in order to generate a manageable set of results. These configurations are spread over all options in an attempt to conclude most cases. Table 3 shows the 38 LVSiM configurations used in this work. We indicate the options in the column headings (using acronyms given in the sections listed above) and use "x" to indicate its use in each configuration (listed as rows 1 to 38). Note that configurations 1, 2, 3, 4, and 5 are cases where multi-V_{dd}/frequency is not used; hence, the only power reduction gain is through switching off unused cores. The allowed values of the voltage levels vary from 0.6V to 1.4V in 0.05V increments, and the normalized frequencies from 0.15 to 1.0 with step of 0.05. These values are later reduced during VFD reduction to 2 voltages and 4 frequencies.

The standard task graph (STG) benchmarks generated by Tobita and Kasahara [78] are used in our experiments. FIGURE 6 shows the characteristics of the 1260 different STGs used in the simulations. The purpose of this figure is simply to illustrate that STG samples were selected to cover a wide range of the application characteristics, such as highly parallel or highly serial, and high and low traffic. The traffic edges and critical paths are plotted against parallelism for each benchmark. Hence, each benchmark is represented as two data points in the figure. The critical path length

TABLE 3. LVSiM simulation configurations.

Simulation	Low Power Techniques							No Multi-Vdd	Scheduling on Cores								Layout	
	VFD Reduction			Core Reduction					Task Priority					Core Selection				
	VFR	FVR	SVFR[3]	C2MD	C2SD	C2SC	T2MD		BTP[4]	CTP	STP[5]	SCTP	CSTP	AFC[6]	CoGC[7]	CoGWL	SDL[1]	ADL[2]
1								x	x					x				
2								x		x					x			
3								x			x				x			
4								x		x						x		
5								x			x					x		
6	x			x					x					x				x
7		x		x					x					x				x
8			x	x					x					x				x
9	x				x				x					x				x
10	x					x			x					x				x
11	x						x		x					x				x
12	x			x						x				x				x
13	x			x							x			x				x
14	x			x								x		x				x
15	x			x									x	x				x
16	x			x					x						x			x
17	x			x					x							x		x
18	x			x						x					x			x
19	x			x							x				x			x
20	x			x								x			x			x
21	x			x									x		x			x
22	x			x						x						x		x
23	x			x							x					x		x
24	x			x								x				x		x
25	x			x									x			x		x
26	x			x						x					x		x	
27	x			x							x				x		x	
28	x			x								x			x		x	
29	x			x									x		x		x	
30	x			x					x					x			x	
31	x				x				x					x			x	
32	x					x			x					x			x	
33	x						x		x					x			x	
34			x	x							x				x		x	
35			x		x						x				x		x	
36			x			x					x				x		x	
37			x				x				x				x		x	
38			x	x							x			x			x	

1, 6, and 7: proposed by [10]
4, and 5: standard method used by [10], [70], [71]
2, and 3: proposed by [72]
3: standard method used by [7], [70], [72]–[77]

is the processing time of the tasks that lie on the critical path. The parallelism factor is the sum of all task computation times divided by the critical path time. The number of edges represents the degree of communication between tasks. The number of tasks used in the benchmarks varies from 4,000 to 16,000.

C. IDLE/DYNAMIC POWER RATIO RESULTS AND ANALYSIS

In this section, we study the impact of the nominal Idle/Dynamic power ratio on the best optimization method for minimizing power. Assuming the Idle/Dynamic power ratio to be **1x**, i.e., the total idle power is equal to the total dynamic power for the given application, FIGURE 7 shows the average normalized EDP (across all 1260 benchmarks) using all 38 configurations listed in Table 3. Configuration **1** is considered the nominal case and, as shown in FIGURE 7,

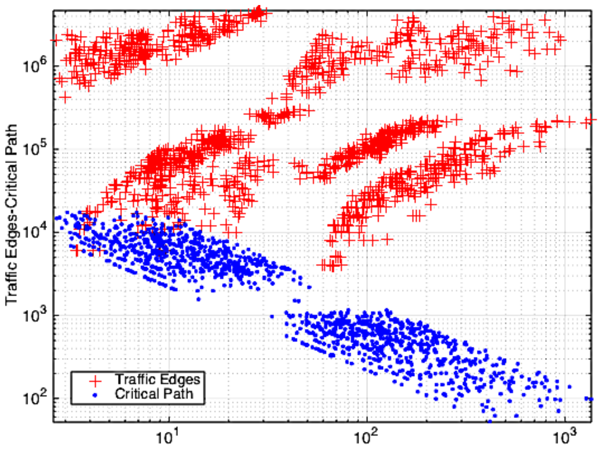


FIGURE 6. Characteristics of 1260 STG benchmarks used in the experiments, generated by Tobita and Kasahara [78].[2]

the idle power is equal to the dynamic power (1st bar in chart). Configurations **2, 3, 4,** and **5** (next 4 bars in chart) reduce power through switching off unused cores only, by using different scheduling techniques, and without any voltage or frequency scaling. There is a noticeable reduction in EDP, as expected, but this is impressive nevertheless, since VFDs were not used. In fact, none of the VFD cases (configurations **6 – 38**) performed better. The lowest power consumption case is configuration **2**. It uses children-based task priority, **CTP**, and center-of-gravity core selection, **CoGC**. This illustrates and emphasizes the effectiveness of dark silicon over VFDs by simply controlling power through switching off some of the unneeded cores.

FIGURE 8 shows the same average power for all configurations, but this time the Idle/Dynamic power ratio is set to **0.2x** for the nominal case. Reducing the Idle/Dynamic power ratio tends to favor multi-V_{dd}/frequency scaling approaches. For this case, many VFD configurations are able to beat the non-VFD cases. The minimum EDP number is produced using configuration **36**, which uses multi-V_{dd}/frequency design with **VFSR** for VFD reduction, **C2SC** for core reduction, **STP** for task priority, **CoGC** for core selection, and **SDL** for layout. A closer look at this configuration reveals that the simultaneous reduction of the voltage and frequency numbers, i.e., VFSR, is the most efficient method. As discussed earlier in section D, this method is the most flexible and it does seem to produce the best energy saving numbers. Moving tasks from removed cores to a single core, i.e. C2SC, seems to produce the best estimated number of cores to be used by the application, and this is surprising. As explained earlier in section C, this configuration is the least flexible, i.e., the core reduction is minimal. Allowing more cores to be used by the application seems to improve the performance and hence the EDP. The slack task priority, i.e., STP, is prioritizing the tasks during scheduling based on the task's slack; the more the slack the less the priority. STP seems to do better than CTP in the case of multi-V_{dd}/frequency design. Using the center of gravity, based on traffic weights, i.e. CoGC, when selecting a core during scheduling, appears to maximize the EDP. CoGC consistently shows the best results. Finally, the stacked domains layout (SDL) is showing better results when compared to the alternating domain layout (ADL).

Table 4 shows the configuration number that produces the best power savings given the Idle/Dynamic ratio. As the ratio goes down configuration **36** produces the best power savings compared to the nominal case, namely configuration **1**. As the Idle/Dynamic ratio increases, configuration **2** starts to take the lead in power savings. In summary, when the dynamic power is dominant, multi-V_{dd}/frequency design is shown to produce the best power savings, namely configuration **36**. On the other hand, if the idle power is the dominant power component, configuration **2** is shown to produce the best

[2]http://www.kasahara.elec.waseda.ac.jp/schedule/

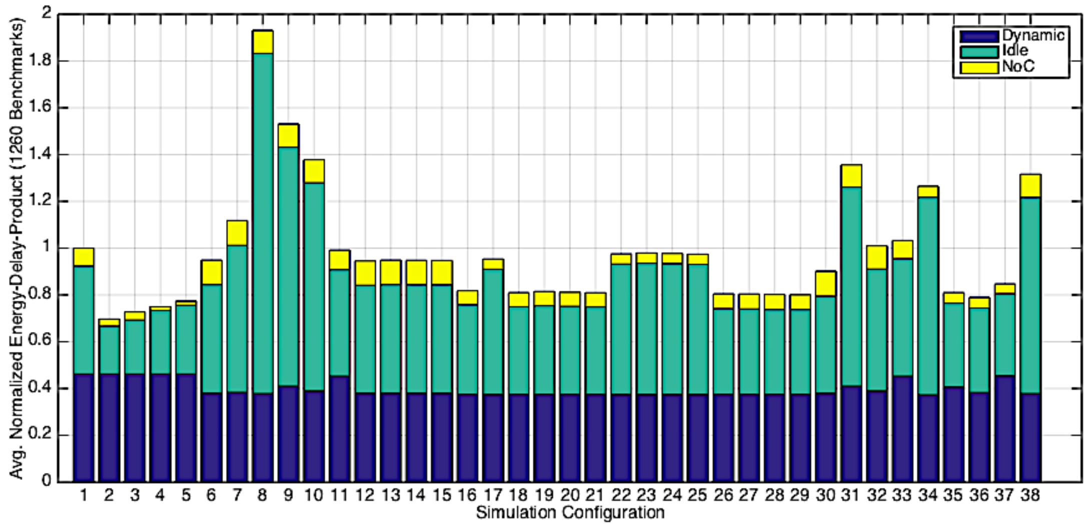


FIGURE 7. Average normalized EDP per configuration (shown in Table 3), averaged across 1260 benchmarks assuming idle/dynamic ratio = 1x.

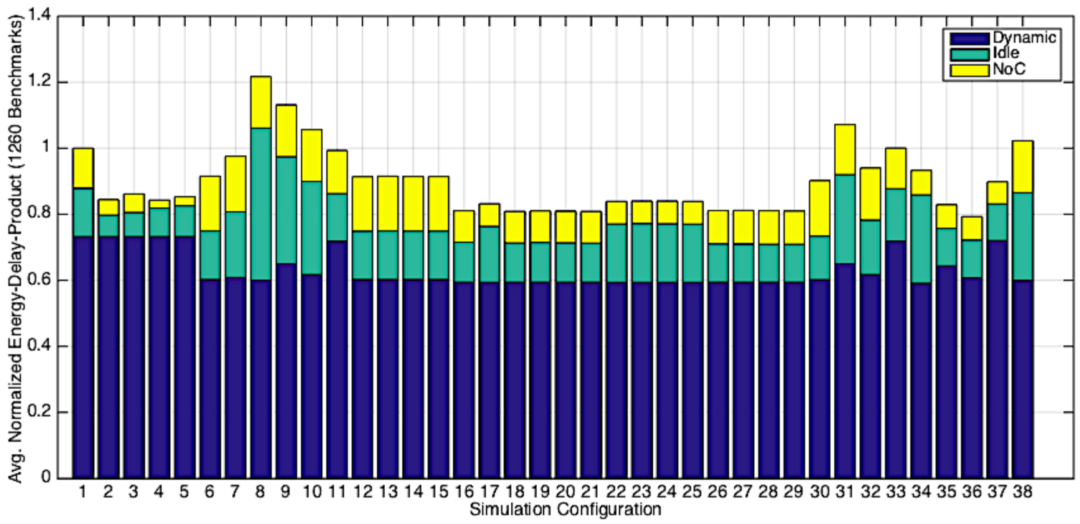


FIGURE 8. Average Normalized EDP per configuration (shown in Table 3), across 1260 benchmarks assuming Idle/Dynamic ratio = 0.2x.

TABLE 4. Configuration to produce maximum power saving with respect to Idle/Dynamic ratio.

Idle/Dynamic Ratio	Best Configuration	Power Saving
0.2x	36	20.68%
0.4x	36	20.81%
0.6x	2	24.62%
1.0x	2	30.34%
2.0x	2	38.31%

power savings. The transition point in our simulations is roughly when Idle/Dynamic is 0.5 in the nominal case. Therefore, the dynamic power consumption should be twice as high as the idle power for the VFD approach to be viable according to these results. Otherwise, it is better to shut off unused cores and use a single V_{dd}/frequency.

Further analysis of the results shows the reason behind the given outcomes. Table 5 provides the number of cores and the number of cycles averaged across all 1260 applications for each configuration. As shown, the number of cores is very close in both configurations, with about a 2.5% difference. Thus, the impact of the number of cores should be minimal. However, the difference in the number of cycles is substantial with 45530 cycles more for configuration 36, an increase of 40% compared to configuration 2. This execution time increase is due to voltage/frequency scaling, i.e., as the voltage and/or frequency scales down to save power, the execution time has to increase.

TABLE 5. Average number of cores and cycles of the configurations 2 and 36.

Configuration	Average Number of Cores	Average Cycle Count	Task Idle Time
36	406.93	159820	237.83
2	396.94	114290	64.9

Another important reason for the time increase is that VFD design adds more scheduling restrictions. A task must be scheduled on a core that exactly fits the task's pre-assigned voltage/frequency within a given VFD. This would limit the number of available cores for this task to be scheduled within a specific VFD. An implementation with no VFD design, i.e., no voltage/frequency scaling, allows any task to execute on any core without restrictions. This also has implications on traffic. A task might be forced to be scheduled far away from its pre- or post-tasks because its VFD is far away, which might incur extra waiting time for the traffic dependencies to be resolved. This issue is demonstrated with the task's idle time shown in Table 5. The task's idle time is defined as the time a task has to wait for the communication traffic. As shown, configuration 36 is showing a higher task's idle time as compared to configuration 2.

VIII. LIMITATIONS

Although, we compared the accuracy of LVSiM with Sniper/McPAT for frequency scaling, voltage scaling was not compared because Sniper/McPAT does not have this feature [64]. Generally, the issue of accuracy versus complexity in modelling complex systems is one of the major concerns for most simulators. LVSiM simulates thousands of tasks running on thousands of cores with multiple voltage and frequency domains, while considering workload and transistor level variability. Thus, it is hard to claim high absolute accuracy in this multi-layered simulation system. In this situation, the objective was to achieve relative accuracy when comparing different methods subjected to the same simulation environment. This is an acceptable compromise for validating proposed methods and techniques in a holistic simulation platform, such as LVSiM, in the very early stages of the design cycle.

Another issue in this work is the use of randomly generated benchmarks (produced by Tobita and Kasahara [78]). Most of the off-the-shelf benchmarks (representing actual applications) target multicores with a very limited number of cores. Therefore, these benchmarks, typically consisting of a few hundred tasks, cannot be used to explore the real potential and limitations of a manycore system. Even if multiple instances of the same small benchmark are simulated, they still do not represent a large benchmark with thousands of tasks. Using randomly generated benchmarks is acceptable for the stated objectives of this paper.

IX. CONCLUSION

In this paper, we studied the impact of the Idle/Dynamic power ratio on the effectiveness of multi-V_{dd}/frequency designs. The best techniques produced 20.7% and 24.6% Energy-Delay-Product savings, considering 0.2x and 0.6x Idle/Dynamic ratio. The results are the average of 1260 different benchmarks with a size range of 4,000 to 16,000 tasks. As the Idle/Dynamic ratio increases, a multi-V_{dd}/frequency becomes less effective, and a regime with switching unused cores off is sufficient to deliver the minimum power consumption. The crossover point appears to be when the dynamic power is twice the idle power. We have proposed LVSiM as a holistic manycore simulation tool to validate our claim. LVSiM takes an application and fully maps it onto cores while taking into account different low-power techniques and configurations, including the effects of intra-die process variations. LVSiM produces comprehensive data that can be used for thorough analysis of different low power techniques under process variations. An open-source version of LVSiM is available publicly.

ACKNOWLEDGMENT

The authors acknowledge the use of the HPC Facility at the University of Sharjah for part of the work reported in this paper. R. A. Saleh was with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada.

REFERENCES

[1] K. Asanovic *et al.*, "The landscape of parallel computing research: A view from berkeley," Dept. Elect. Eng. Comput. Sci., Univ. California Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2006-183, 2006, p. 19, vol. 18.

[2] M. B. Taylor, "A landscape of the new dark silicon design regime," *IEEE Micro*, vol. 33, no. 5, pp. 8–19, Sep./Oct. 2013.

[3] M. Mohamed, Z. Li, X. Chen, L. Shang, and A. R. Mickelson, "Reliability-aware design flow for silicon photonics on-chip interconnect," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 8, pp. 1763–1776, Aug. 2014.

[4] B. Raghunathan, Y. Turakhia, S. Garg, and D. Marculescu, "Cherry-picking: Exploiting process variations in dark-silicon homogeneous chip multi-processors," in *Proc. DATE*, 2013, pp. 39–44.

[5] U. R. Karpuzcu, N. S. Kim, and J. Torrellas, "Coping with parametric variation at near-threshold voltages," *IEEE Micro*, vol. 33, no. 4, pp. 6–14, Jul. 2013.

[6] T. Karnik *et al.*, "Resiliency for many-core system on a chip," in *Proc. ASP-DAC*, 2014, pp. 388–389.

[7] E. Garcia and G. R. Gao, "Strategies for improving performance and energy efficiency on a many-core," in *Proc. Int. Conf. Comput. Frontiers*, vol. 9, 2013, pp. 1–4.

[8] S. Afsharpour, M. Fazeli, and A. Patooghy, "Performance/energy aware task migration algorithm for many-core chips," *IET Comput. Digit. Techn.*, vol. 10, no. 4, pp. 165–173, 2016.

[9] T. Kodaka *et al.*, "A near-future prediction method for low power consumption on a many-core processor," in *Proc. DATE*, 2013, pp. 1058–1059.

[10] S. S. Majzoub, R. A. Saleh, S. J. E. Wilton, and R. K. Ward, "Energy optimization for many-core platforms: Communication and PVT aware voltage-island formation and voltage selection algorithm," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 29, no. 5, pp. 816–829, May 2010.

[11] L. Wanner *et al.*, "NSF expedition on variability-aware software: Recent results and contributions," *Inf. Technol.*, vol. 57, no. 3, pp. 181–198, 2015.

[12] S. Majzoub, "Reducing random-dopant fluctuation impact using footer transistors in many-core systems," *Integration*, vol. 48, no. 1, pp. 46–54, Jan. 2015.

[13] S. Borkar, "Design perspectives on 22 nm CMOS and beyond," in *Proc. DAC*, 2009, pp. 93–94.

[14] M. B. Taylor, "Is dark silicon useful?" in *Proc. DAC*, vol. 12, p. 1131, Jun. 2012.

[15] N. Binkert *et al.*, "The gem5 Simulator," *Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, 2011.

[16] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Sep. 2011, pp. 1–12.

[17] O. Certner, Z. Li, A. Raman, and O. Temam, "A very fast simulator for exploring the many-core future," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2011, pp. 443–454.

[18] D. J. R. Ferreira, "Analysis of many-core CPUs simulators," Instituto Superior Técnico, Universidade de Lisboa, Tech Rep., pp. 1–10. [Online]. Available: https://fenix.tecnico.ulisboa.pt/downloadFile/563345090413270/resumo.pdf

[19] S. Friederich, J. Heisswolf, and J. Becker, "Hardware/software debugging of large scale many-core architectures," in *Proc. Symp. Integr. Circuits Syst. Design*, 2014, pp. 45:1–45:7.

[20] W. Heirman, T. Carlson, S. Sarkar, P. Ghysels, W. Vanroose, and L. Eeckhout, "Using fast and accurate simulation to explore hardware/software trade-offs in the multi-core era," in *Proc. Int. Conf. Parallel Comput.*, 2012, pp. 343–350.

[21] W.-Y. Lee and I. H.-R. Jiang, "VIFI-CMP: Variability-tolerant chip-multiprocessors for throughput and power," in *Proc. Great Lakes Symp. VLSI*, 2009, pp. 39–44.

[22] W. Heirman, S. Sarkar, T. E. Carlson, I. Hur, and L. Eckhout, "Power-aware multi-core simulation for early design stage hardware/software co-optimization," in *Proc. Int. Conf. Parallel Archit. Compilation Techn.*, 2012, pp. 3–12.

[23] Z. Lai, K. T. Lam, C. -L. Wang, and J. Su, "PoweRock: Power modeling and flexible dynamic power management for many-core architectures," *IEEE Syst. J.*, vol. 11, no. 2, pp. 600–612, Jun. 2017.

[24] S. J. Hollis and S. Kerrison, "Swallow: Building an energy-transparent many-core embedded real-time system," in *Proc. DATE*, 2016, pp. 73–78.

[25] A. Akram and L. Sawalha, "×86 computer architecture simulators: A comparative study," in *Proc. IEEE 34th Int. Conf. Comput. Design (ICCD)*, Oct. 2016, pp. 638–645.

[26] D. M. Brooks *et al.*, "Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors," *IEEE Micro*, vol. 20, no. 6, pp. 26–44, Nov. 2000.

[27] T. Agerwala and S. Chatterjee, "Computer architecture: Challenges and opportunities for the next decade," *IEEE Micro*, vol. 25, no. 3, pp. 58–69, May 2005.

[28] Y. Fu and D. Wentzlaff, "PriME: A parallel and distributed simulator for thousand-core chips," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Mar. 2014, pp. 116–125.

[29] F. Dubois, A. Sheibanyrad, F. Pétrot, and M. Bahmani, "Elevator-first: A deadlock-free distributed routing algorithm for vertically partially connected 3D-NoCs," *IEEE Trans. Comput.*, vol. 62, no. 3, pp. 609–615, Mar. 2013.

[30] T. E. Carlson, W. Heirman, K. Van Craeynest, and L. Eeckhout, "Node performance and energy analysis with the sniper multi-core simulator," in *Tools for High Performance Computing*. New York, NY, USA: Springer, 2013.

[31] A. Butko *et al.*, "A trace-driven approach for Fast and accurate simulation of manycore architectures," in *Proc. ASP-DAC*, 2015, pp. 707–712.

[32] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. Micro*, 2009, p. 469.

[33] R. G. Kim *et al.*, "Imitation learning for dynamic VFI control in large-scale manycore systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 9, pp. 2458–2471, Sep. 2017.

[34] E. Cai, D.-C. Juan, S. Garg, J. Park, and D. Marculescu, "Learning-based power/performance optimization for many-core systems with extended-range voltage/frequency scaling," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 35, no. 8, pp. 1318–1331, Aug. 2016.

[35] I. Stamelakos, S. Xydis, G. Palermo, and C. Silvano, "Variation-aware voltage island formation for power efficient near-threshold manycore architectures," in *Proc. ASP-DAC*, 2014, pp. 304–310.

[36] J. Lee and N. S. Kim, "Optimizing total power of many-core processors considering voltage scaling limit and process variations," in *Proc. Int. Symp. Low Power Electron. Design*, 2009, pp. 201–206.

[37] T. N. Miller, R. Thomas, and R. Teodorescu, "Mitigating the effects of process variation in ultra-low voltage chip multiprocessors using dual supply voltages and half-speed units," *IEEE Comput. Archit. Lett.*, vol. 11, no. 2, pp. 45–48, Jul./Dec. 2012.

[38] N. Drego, A. Chandrakasan, D. Boning, and D. Shah, "Reduction of variation-induced energy overhead in multi-core processors," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 30, no. 6, pp. 891–904, Jun. 2011.

[39] A. M. Rahmani, M.-H. Haghbayan, A. Miele, P. Liljeberg, A. Jantsch, and H. Tenhunen, "Reliability-aware runtime power management for many-core systems in the dark silicon era," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 427–440, Feb. 2017.

[40] R. Jeyapaul, F. Hong, A. Rhisheekesan, A. Shrivastava, and K. Lee, "UnSync-CMP: Multicore CMP architecture for energy-efficient soft-error reliability," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 254–263, Jan. 2014.

[41] P. Mercati *et al.*, "Multi-variable dynamic power management for the GPU subsystem," in *Proc. 54th Annu. Design Automat. Conf. (DAC)*, vol. 17, 2017, Art. no. 2.

[42] R. Child and P. Wilsey, "Dynamically adjusting core frequencies to accelerate time warp simulations in many-core processors," in *Proc. Workshop Princ. Adv. Distrib. Simulation (PADS)*, Jul. 2012, pp. 35–43.

[43] H. Li *et al.*, "Energy-efficient power delivery system paradigms for many-core processors," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 36, no. 3, pp. 449–462, Mar. 2017.

[44] Y. Liu, G. Cox, Q. Deng, S. C. Draper, and R. Bianchini, "FastCap: An efficient and fair algorithm for power capping in many-core systems," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, no. 3, 2016, pp. 57–68.

[45] K. Ma, X. Li, M. Chen, and X. Wang, "Scalable power control for many-core architectures running multi-threaded applications," in *Proc. ISCA*, 2011, vol. 47, no. 3, pp. 449–460.

[46] A. K. Singh, P. Dziurzanski, and L. S. Indrusiak, "Value and energy optimizing dynamic resource allocation in many-core HPC systems," in *Proc. Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, 2016, pp. 180–185.

[47] M. Srivastav, M. Ehteshamuddin, K. Stegner, and L. Nazhandali, "Design of ultra-low power scalable-throughput many-core DSP applications," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 20, no. 3, 2015, Art. no. 34.

[48] C. Thompson, "On the simulation and design of manycore CMPs," Ph.D. dissertation, Inst. Comput. Syst. Archit., School Inform., 2014.

[49] P. Martin, L. Wanner, and M. Srivastava, "Runtime optimization of system utility with variable hardware," *ACM Trans. Embedded Comput. Syst.*, vol. 14, no. 2, 2015, Art. no. 24.

[50] P. Mercati, F. Paterna, A. Bartolini, L. Benini, and T. S. Rosing, "Dynamic variability management in mobile multicore processors under lifetime constraints," in *Proc. IEEE 32nd Int. Conf. Comput. Design (ICCD)*, Oct. 2014, pp. 448–455.

[51] L. Wanner, S. Elmalaki, L. Lai, P. Gupta, and M. Srivastava, "VarEMU: An emulation testbed for variability-aware software," in*Proc. Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS)*, Sep./Oct. 2013, pp. 1–10.

[52] P. Mercati, F. Paterna, A. Bartolini, L. Benini, and T. Š. Rosing, "WARM: Workload-aware reliability management in linux/android," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 36, no. 9, pp. 1557–1570, Sep. 2017.

[53] P. Mercati, F. Paterna, A. Bartolini, M. Imani, L. Benini, and T. Š. Rosing, "VarDroid: Online variability emulation in Android/Linux platforms," in *Proc. Int. Great Lakes Symp. VLSI (GLSVLSI)*, vols. 18–20, May 2016, pp. 269–274.

[54] A. Rahimi, L. Benini, and R. K. Gupta, "Hierarchically focused guardbanding: An adaptive approach to mitigate PVT variations and aging," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, 2013, pp. 1695–1700.

[55] A. Pathania, H. Khdr, M. Shafique, T. Mitra, and J. Henkel, "Scalable probabilistic power budgeting for many-cores," in *Proc. DATE*, 2017, pp. 864–869.

[56] A. Bartolini, C. Hankendi, A. K. Coskun, and L. Benini, "Message passing-aware power management on many-core systems," *J. Low Power Electron.*, vol. 10, no. 4, pp. 531–549, Dec. 2014.

[57] R. Puri, M. Choudhury, H. Qian, and M. Ziegler, "Bridging high performance and low power in processor design," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2014, pp. 183–188.

[58] C. Tan, A. Kulkarni, V. Venkataramani, M. Karunaratne, T. Mitra, and L.-S. Peh, "LOCUS: Low-power customizable many-core architecture for wearables," in *Proc. Int. Conf. Compliers, Archit., Sythesis Embedded Syst. (CASES)*, 2016, pp. 1–10.

[59] I. Stamelakos, S. Xydis, G. Palermo, and C. Silvano, "Throughput balancing for energy efficient near-threshold manycores," in *Proc. 26th Int. Workshop Power Timing Modeling, Optim. Simulation (PATMOS)*, 2016, pp. 64–69.

[60] I. Stamelakos, A. Khajeh, A. Eltawil, G. Palermo, C. Silvano, and F. Kurdahi, "A system-level exploration of power delivery architectures for near-threshold manycores considering performance constraints," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2016, pp. 484–489.

[61] S. Das, J. R. Doppa, P. P. Pande, and K. Chakrabarty, "Design-space exploration and optimization of an energy-efficient and reliable 3-D small-world network-on-chip," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 36, no. 5, pp. 719–732, May 2017.

[62] 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 Trans. Semicond. Manuf.*, vol. 21, no. 1, pp. 3–13, Feb. 2008.

[63] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 5, pp. 501–513, May 006.

[64] W. Heirman, A. Isaev, and I. Hur, "Sniper: Simulation-based instruction-level statistics for optimizing software on future architectures," in *Proc. 3rd Int. Conf. Exascale Appl. Softw.*, 2015, pp. 29–31.

[65] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, Dec. 2009, p. 469.

[66] D. Genbrugge, S. Eyerman, and L. Eeckhout, "Interval simulation: Raising the level of abstraction in architectural simulation," in *Proc. 16th Int. Symp. High-Perform. Comput. Archit. (HPCA)*, 2010, pp. 1–12.

[67] I. Ashraf, N. Khammassi, M. Taouil, and K. Bertels, "Memory and communication profiling for accelerator-based platforms," *IEEE Trans. Comput.*, vol. 67, no. 7, pp. 934–948, Jul. 2018.

[68] C.-K. Luk *et al.*, "Pin: Building customized program analysis tools with dynamic instrumentation," in *Proc. ACM SIGPLAN Conf. Program. Lang. Design Implement. (PLDI)*, 2005, vol. 40, no. 6, p. 190.

[69] J. Stam, "Real-time fluid dynamics for games," in *Proc. Game Developer Conf.*, 2003, vol. 18, no. 11, p. 17.

[70] W. Jang, D. Ding, and D. Z. Pan, "Voltage and frequency island optimizations for many-core/networks-on-chip designs," in *Proc. 1st Int. Conf. Green Circuits Syst. ICGCS*, 2010, pp. 217–220.

[71] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems: Survey of current and emerging trends," in *Proc. 50th ACM/EDAC/IEEE Design Automat. Conf. (DAC)*, May/Jun. 2013, pp. 1–10.

[72] S. Borkar, "Thousand core chips—A technology perspective," in *Proc. DAC*, 2007, pp. 749–754.

[73] E. Le Sueur and G. Heiser, "Dynamic voltage and frequency scaling: The laws of diminishing returns," in *Proc. Int. Conf. Power Aware Comput. Syst.*, 2010, pp. 1–8.

[74] C. Silvano, G. Palermo, S. Xydis, and I. Stamelakos, "Voltage island management in near threshold manycore architectures to mitigate dark silicon," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, 2014, pp. 1–6.
[75] U. Y. Ogras, R. Marculescu, P. Choudhary, and D. Marculescu, "Voltage-frequency island partitioning for GALS-based networks-on-chip," in *Proc. ACM/IEEE Design Automat. Conf.*, Jun. 2007, pp. 110–115.
[76] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2007, pp. 38–43.
[77] X. Wang, Z. Li, M. Yang, Y. Jiang, M. Daneshtalab, and T. Mak, "A low cost, high performance dynamic-programming-based adaptive power allocation scheme for many-core architectures in the dark silicon era," in *Proc. IEEE Symp. Embedded Syst. Real-Time Multimedia*, Oct. 2013, pp. 61–67.
[78] T. Tobita and H. Kasahara, "A standard task graph set for fair evaluation of multiprocessor scheduling algorithms," *J. Scheduling*, vol. 394, no. 5, pp. 379–394, 2002.

SOHAIB MAJZOUB (M'10–SM'17) received the B.E. degree in electrical engineering from the Computer Section, BAU, in 2000, the M.E. degree from AUB, Lebanon, in 2003, and the Ph.D. degreefrom the System-on-Chip research Laboratory, The University of British Columbia, Canada, in 2010. He worked for one year at the Processor Architecture Lab, Swiss Federal Institute of Technology, Lausanne, Switzerland. He worked for two years as an Assistant Professor with American University in Dubai, Dubai, United Arab Emirates. He then joined King Saud University, Saudi Arabia, in 2012. In 2015, he joined the University of Sharjah, United Arab Emirates, as a Faculty with the Electrical and Computer Engineering Department. His research interests include delay/power system modeling and low power manycore design. He is a member of the IEEE Computer Society, the Solid State Society, and a Circuits and Systems Member.

RESVE A. SALEH (M'79–SM'03–F'06) received the B.S. degree from Carleton University, Ottawa, Canada, and the M.S. and Ph.D. degrees from the University of California at Berkeley, Berkeley, all in electrical engineering.

He was a Professor and the NSERC/PMC-Sierra Chair with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada, in system-on-chip design and test. He has published more than 100 journal articles and conference papers. He co-authored a book *Design and Analysis of Digital Integrated Circuit Design: In Deep Submicron Technology*. He is a Professional Engineer of British Columbia. He received the Presidential Young Investigator Award, in 1990, from the National Science Foundation, USA. He served as a Technical Program Chair, in 1993, a Conference Chair, in 1994, and a General Chair, in 1995, for the Custom Integrated Circuits Conference. He held the positions of a Technical Program Chair, a Conference Chair, and a Vice-General Chair of the International Symposium on Quality in Electronic Design, in 2001. He has served as an Associate Editor for the IEEE TRANSACTIONS ON CAD.

Dr. Saleh spent nine years as a Professor with the Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign. He was a Founder of Simplex Solutions which developed CAD software for deep submicron digital design verification. He also taught for one year at Stanford University. He has worked for Mitel Corporation, Ottawa, Canada, Toshiba Corporation, Japan, Tektronix, Beaverton, OR, USA, and Nortel, Ottawa.

IMRAN ASHRAF received the Ph.D. degree in computer engineering from the Delft University of Technology, The Netherlands, in 2016, where he is currently a Postdoctoral Researcher and also with Quantum Computing Lab, QuTech. His research interests include advanced profiling, code parallelization, communication driven mapping of applications on multicore platforms, and compilation techniques for quantum computing.

MOTTAQIALLAH TAOUIL received the M.Sc. and Ph.D. degrees (Hons.) in computer engineering from the Delft University of Technology, Delft, The Netherlands. He is currently a Postdoctoral Researcher with the Dependable Nano-Computing Group, Delft University of Technology. His current research interests include reconfigurable computing, embedded systems, very large scale integration design and test, built-in-self-test, and 3-D stacked integrated circuits, architectures, design for testability, yield analysis, and memory test structures.

SAID HAMDIOUI (M'99–SM'11) worked for Intel Corporation, CA, USA, Philips Semiconductors R&D, Crolles, France, and for Philips/ NXP Semiconductors, Nijmegen, The Netherlands. He is currently a Chair Professor on dependable and emerging computer technologies with the Computer Engineering Laboratory, Delft University of Technology, The Netherlands. He holds one patent and has published one book and co-authored more than 170 conference and journal papers. He delivered dozens of keynote speeches, distinguished lectures, and invited presentations and tutorial at major international forums/conferences/schools and at leading semiconductor companies. His research interests include dependable CMOS nano-computing (including reliability, testability, and hardware security), and emerging technologies and computing paradigms (including 3D stacked ICs, memristors for logic and storage, and in-memory-computing). He is also a member of the Association for European NanoElectronics Activities/ENIAC Scientific Committee Council. He is an Associate Editor of the IEEE TRANSACTIONS ON VLSI SYSTEMS. He serves on the Editorial Board of the IEEE DESIGN & TEST, and of the *Journal of Electronic Testing: Theory and Applications*.