A Runtime Framework for Robust Application Scheduling With Adaptive Parallelism in the Dark-Silicon Era

Nishit Kapadia, *Member, IEEE*, and Sudeep Pasricha, *Senior Member, IEEE*

***Abstract*— With deeper technology scaling accompanied by a worsening power wall, an increasing proportion of chip area on a chip multiprocessor (CMP) is expected to be occupied by dark silicon. At the same time, design challenges due to process variations and soft errors in integrated circuits are projected to become even more severe. It is well known that spatial variations in process parameters introduce significant unpredictability in the performance and power profiles of CMP cores. By mapping applications onto the best set of cores, process variations can potentially be used to our advantage in the dark-silicon era. In addition, the probability of occurrence of soft errors during application execution has been found to be strongly related to the supply voltage and operating frequency values, thus necessitating reliability awareness within runtime voltage scaling schemes in contemporary CMPs. In this paper, we present a novel framework that leverages the knowledge of variations on the chip to perform runtime application mapping and dynamic voltage scaling to optimize system performance and energy, while satisfying dark-silicon power constraints of the chip as well as application-specific performance and reliability constraints. Our experimental results show average savings of 10%–71% in application service times and 13%–38% in energy consumption, compared with prior work.**

***Index Terms*— Application parallelism, dark silicon, multicore scheduling, process variations, soft-error reliability.**

I. Introduction

With increasing transistor miniaturization, circuit densities have drastically increased, and the critical charge, which is the minimum charge capable of a bit flip in a memory cell or a logic cell, has significantly decreased [1]–[3]. This phenomenon has caused newer process technologies to be more susceptible to transient faults due to the effects of radiation, e.g., alpha particle and neutron strikes. The rate of such transient faults at runtime has thus been increasing with technology scaling [4]. Studies have also shown that hardened flip-flops are only 30%–50% more resilient than unprotected ones, at sub-40-nm nodes [14], i.e., circuit-level hardening may not sufficiently suppress soft errors. Thus, system-level runtime approaches to cope with transient faults and complement circuit-level techniques will be increasingly essential in newer process technologies.

Simultaneously, unpredictability in leakage power and circuit delay due to variability in modern fabrication processes has become a serious concern. In emerging chip multiprocessors (CMPs), spatially correlated systematic within-die (WID) variations manifest across multiple cores, creating core-to-core variations [5]. At the same time, die-to-die (D2D) variations remain quite significant [6]. Both WID and D2D variations have random and systematic components. Although several prior works, such as [35]–[37], have proposed variation-aware design-time application-mapping frameworks, it is generally quite difficult to predict the variation profile of a fabricated chip at design time [19]. It is, however, possible to extract the variation map of a chip at runtime from the chip-frequency profile obtained using ring-oscillator-based delay-sensors [20], [21]. Thus, runtime approaches to mitigate adverse effects of process variations become possible, and will be vital for scaled technologies.

The slowdown of power scaling with technology scaling, due to leakage and reliability concerns [7], [8], has led to a rise in chip power densities, and created the dark-silicon phenomenon—a significant fraction of the chip needs to be shut down (i.e., dark) at any given time to satisfy the chip power budget. With the extent of dark silicon increasing every technology generation (30%–50% for 22 nm) [9], [10], designs are becoming increasingly power-limited rather than area-limited. Runtime power-saving techniques, such as dynamic voltage scaling (DVS), are thus becoming increasingly important.

Given these multiple daunting design challenges, there is a critical need for a system-level solution that can simultaneously and adaptively manage the constraints imposed by dark silicon, process variations, and soft-error reliability, while executing applications. In this paper, we address this need by proposing a novel runtime variation- and reliability-aware application-scheduling framework that employs dynamically adaptable application degrees of parallelism (app-DoPs) to minimize average application service times and energy, while meeting a chip-wide dark silicon power constraint (DS-Pc) and application performance and reliability constraints, in the presence of process variations. Our novel contributions in this paper are as follows.

1) We design a novel runtime application-mapping framework for the emerging dark-silicon-constrained design regime, improving over traditional mapping approaches that are typically optimized for the area-constrained

Manuscript received February 2, 2016; revised June 13, 2016; accepted July 19, 2016. Date of publication August 12, 2016; date of current version January 19, 2017. This work was supported in part by NSF under Grant CCF-1252500 and Grant CCF-1302693 and in part by SRC.

N. Kapadia is with Synopsys Inc., Hillsboro, OR 97124 USA (e-mail: nishit.kapadia@synopsys.com).

S. Pasricha is with the Electrical and Computer Engineering Department, Colorado State University, Fort Collins, CO 80523 USA (e-mail: sudeep@colostate.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2016.2594238

design regime.
2) Our framework simultaneously manages all dynamically arriving applications while adapting app-DoPs to optimally utilize the system power slack (difference between DS-Pc and current system power dissipation).
3) We design a novel heuristic to integrate within the application-mapping process a DVS mechanism that is constrained not just by performance but also by application-reliability requirements.
4) The traditionally disjoint design steps of region-selection and task-to-tile mapping are seamlessly integrated in our framework to co-optimize memory communication and leakage power while exhibiting awareness of the runtime CMP-environment.
5) Our combined mapping and DVS approach takes advantage of both D2D and WID variations, performing WID variation-aware mapping onto cores with the optimal power and performance characteristics, and D2D variation-aware chip-wide DVS where faster (leakier) chips would need lower V_{dd} and slower chips may run at higher V_{dd}.

II. Related Work

A few recent works have begun to focus on dark-silicon aware design methodologies for emerging CMPs. Allred *et al.* [9] and Turakhia *et al.* [17] propose design-time frameworks for the synthesis of heterogeneous CMPs to extract better energy efficiency and performance in the dark-silicon regime. However, these works do not consider the effects of reliability and the impact of process variations on CMP performance and power dissipation. Raghunathan *et al.* [10] exploit the variation profile for runtime application mapping onto a homogeneous CMP die, to maximize performance and reduce leakage power given a fixed dark-silicon power budget. But the authors do not consider overheads of intercore communication or application-reliability requirements. Chou *et al.* [23] and Fattah *et al.* [24] consider runtime mapping of a queue of incoming applications, and propose mapping techniques to fit the maximum number of applications possible on a homogeneous CMP die, while minimizing intercore communication distances. However, these approaches do not consider reliability and system power dissipation, and are geared toward traditional area-constrained designs.

Several other efforts (see [11], [13], and [38]) have proposed design-time fault-tolerant scheduling frameworks that assume fault-detection mechanisms implemented on the multicore platform. Hardening techniques, such as task reexecution and replication, are utilized in these works to probabilistically meet task-completion deadlines for real-time tasks of varying criticalities. However, these efforts do not consider dark silicon and process variation challenges, runtime adaptation support, or a dynamically parallel application workload.

Some recent works advocate varying the DoP of multithreaded applications at runtime to adapt to changes in the execution environment (power/performance profiles, core availability, and so on) of a CMP while optimizing metrics, such as power and energy-delay product. Given enough parallelism within the application, [7] showed that increasing DoP is a more energy-efficient way of boosting performance, compared with hiking the core-frequency/voltage values. Variable DoPs can be implemented by saving multiple versions of application code at compile time, and selecting the DoP at runtime that is most appropriate for the execution environment, or via more sophisticated techniques [18]. The variable-DoP runtime scheduling frameworks in [15] and [16] report improvements over scheduling techniques that employ statically fixed DoPs, and search for the best combination of voltage, frequency, number of cores, and number of threads, to optimize power and performance of a single application on a CMP. Our proposed framework is different from these efforts in that we assume such information to be preprofiled at design time and the focus is on runtime management of app-DoPs in a multiapplication power-constrained system. A more recent work [22] shows performance improvements by adapting app-DoPs at runtime to varying arrival rates. The authors in this paper focus on asymmetric multicore processors, assume execution of a single application at any given time, and primarily focus on cluster migration to improve performance; in contrast, our proposed runtime framework focuses on symmetric multicore processors and addresses the problem of simultaneous execution of multiple applications. Moreover, unlike the frameworks in [15], [16], and [22], this paper also considers the effects of process variations as well as the design implications of system reliability.

More recently, a few works have proposed reliability- and variation-aware runtime application mapping and DVS frameworks for multicore systems. Salehi *et al.* [39] advocate using different precompiled code versions for every task with varying levels of reliability, and maximizes the reliability of the currently mapped task given the power budget available at runtime. But this paper does not consider multithreaded applications or DoP adaptation, nor does it consider any intertask communication. Our prior work, VARSHA [40], aims to minimize average application service times by performing DoP adaptations and exhibiting awareness of the runtime CMP environment (variation profile, application reliability/performance, and dark silicon). This framework performs application mapping on contiguous and nonoverlapping rectangular regions on the CMP die. As in most prior works, VARSHA assumes that the traffic generated in the network-on-chip (NoC) due to accesses to off-chip memory, i.e., communication traffic between compute cores and on-chip memory controllers (MCs), is negligible. Under such an assumption, mapping applications within rectangular regions could simplify design and enable communication isolation. However, such an approach that restricts application mapping onto rectangular regions exclusively has the following disadvantages: 1) the scope of power optimization is restricted while selecting cores that dissipate lower leakage power and meet performance constraints and 2) performing region selection without the awareness of memory traffic could potentially result in much longer communication routing paths to MCs, which in turn would induce network congestion and increase communication latencies.

In this paper, we improve upon the above mentioned shortcomings of the VARSHA framework by: 1) abandoning the restrictive region-selection methodology and 2) considering memory traffic in the tile-selection (region-selection) and mapping scheme. Our improved tile-selection approach essentially balances leakage-power savings with communication optimization in the presence of memory traffic in the NoC. In other words, we select CMP tiles that dissipate lower leakage power while simultaneously minimizing the traffic footprint (traffic-fp) of the mapped application. The integrated tile-selection and task-to-tile mapping approach proposed in this paper (discussed in Section V) is much better suited to effectively tradeoff between both the objectives.

III. Motivational Example

In this section, we illustrate the advantages of some of the key aspects of our framework with the help of a small motivational example. We consider a scenario in which applications arrive at runtime at a service queue to be served. The example assumes applications App-1 and App-2 arriving and being mapped on the CMP at time $t = 0$ s, and a third application App-3 arriving at $t = 1$ s. In this section, we show [in Fig. 1(a)–(c)] how different approaches proposed in prior works perform the application mapping in comparison to our proposed approach [shown in Fig. 1(d)]. The locations of MCs on the chip corners serving the respective applications are also shown in Fig. 1. A DS-Pc of 45 W is assumed for the 6×6 CMP utilized in this example.

Raghunathan *et al.* [10] exploit the variation profile [shown in Fig. 1(e)] of the CMP die and select the tiles with the least leakage power dissipation (with highest effective V_T values), as shown in Fig. 1(a). But this paper does not consider the intercore communication in the NoC fabric, thereby producing mapping solutions with high communication power and latency overheads. On the other hand, prior works, such as [23] and [24], search for square or near-convex regions on the CMP die to map arriving application at runtime, so that a maximum number of applications can fit on the die area, while also minimizing intercore communication distances [as shown in Fig. 1(b)]. Clearly, such a variation-unaware approach is suited for traditional area-constrained designs. Therefore, frameworks proposed in [10] and [24] would map App-1 and App-2 at $t = 0$, as shown in Fig. 1(a) and (b) with a nominal DoP of 8. At $t = 1$ s, mapping of App-3 at its nominal DoP of 8 is stalled (until App-1 finishes), because the projected power values exceed the DS-Pc of 45 W.

Mapping solutions produced with our prior work VARSHA [40] and the proposed VARSHA++ framework in this paper are shown in Fig. 1(c) and (d). Our frameworks that adapt app-DoPs to extract maximum system performance (i.e., minimum application service times) for a given DS-Pc increase the DoP of App-1–12. When App-3 arrives at $t = 1$ s, our frameworks map it with a reduced DoP of 4 with a zero-wait time while meeting the DS-Pc. Thus, in our frameworks, app-DoPs are hiked from their nominal values opportunistically to minimize runtimes, and app-DoPs are reduced to cut down on wait times,

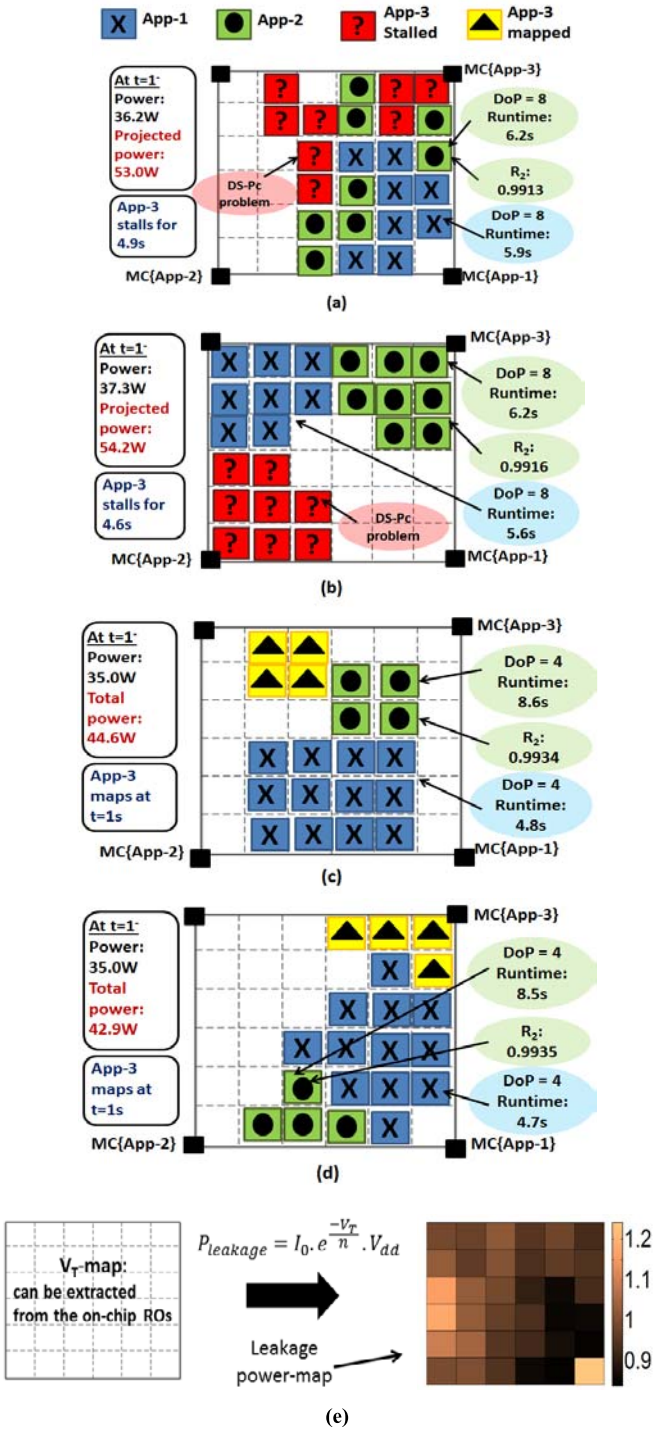

Fig. 1. Motivational example using a 6×6 CMP where App-1 and App-2 are simultaneously mapped at time $t = 0$ and App-3 arrives at $t = 1$ s. DS-Pc = 45 W is assumed. Application-mapping solutions obtained with (a) [10], (b) [24], (c) VARSHA [40], (d) our proposed VARSHA++ framework, and (e) extracted V_T map and corresponding leakage-power profile (in watts) of the chip.

thereby minimizing average application service times (service time = run time + wait time). As the application mapping in VARSHA++ is not restricted to contiguous regions [as shown in Fig. 1(d)], it produces solutions with applications mapped closer to corresponding MCs, and thus offers superior NoC-communication optimization in the presence of memory traffic, in comparison to VARSHA [40]. In addition, observe

that VARSHA++ produces much better leakage profiles due to the flexibility of selecting noncontiguous mapping regions on the die.

We define reliability of the ith application as $R_i = \{1-$*Probability of one or more soft errors during the execution of App-i*$\}$. In this paper, we assume applications with different minimum reliability constraints running simultaneously on a CMP. Rc_i represents the reliability constraint of the ith application. The application reliability (R_i) primarily depends on V_{dd} and frequency of the cores [as shown in (1)–(3)]. In addition, R_i depends on the app-DoP. Although application execution time typically reduces with higher DoP (as long as the DoP is below an application-specific performance saturation point), the chip area susceptible to soft errors increases. Therefore, for fixed values of voltage and frequency, soft-error probability increases (i.e., application reliability decreases) with increasing app-DoP. Our frameworks [shown in Fig. 1(c) and (d)] exhibit reliability awareness by reducing the DoP of App-2 (with a relatively stringent reliability constraint) from the nominal value of 8–4, thereby meeting Rc_2 of 0.993 [Fig. 1(b)]. However, Rc_2 is violated when the reliability-unaware frameworks [10] and [24] are employed [shown in Fig. 1(a) and (b)].

In summary, the variation-aware framework [10] optimizes computation leakage energy by mapping onto low leakage tiles (resulting in overall energy of 89.9 J in this example), while the variation-unaware framework [24] optimizes communication energy by mapping applications within contiguous squarelike regions (resulting in overall energy of 90.9 J), whereas VARSHA [40] finds contiguous rectangular regions on the die with minimum estimated leakage power, thereby cutting down on both communication and computation energy (resulting in overall energy of 74.5 J). On the other hand, our proposed VARSHA++ framework in this paper improves on VARSHA by: 1) further optimizing NoC communication profiles with memory-traffic awareness and 2) simultaneously producing better leakage profiles by relaxing the constraint of mapping onto contiguous regions (resulting in overall energy of 69.7 J). Note that our frameworks also employ an energy-saving mechanism to opportunistically scale V_{dd} levels while meeting performance and reliability constraints of all applications. This feature is omitted from the above example for brevity.

IV. Models and Problem Formulation

A. Reliability Modeling

Computer systems are susceptible to both transient and permanent faults, the latter being caused by fabrication defects or wear out. We only consider the impact of transient faults on reliability and do not consider permanent faults in this paper. It is assumed that permanent faults could either be detected during the testing phase or are mitigated by hardware-redundancy techniques. To model the dependence of raw soft (transient) error rate, raw-soft error rate (SER) (λ), in a hardware component (core or NoC router) on voltage and frequency values, we use the relationship proposed in [4]

$$\lambda(\omega_j) = \lambda_0 \cdot 10^{\frac{d(1-\omega_j)}{1-\omega_{\min}}} \tag{1}$$

where λ_0 is the SER corresponding to the highest voltage and frequency values ($\omega_{\max}$) and ω_j is the average of the normalized values of the jth combination of voltage and frequency (such that $\omega_{\max} = 1$). For any compute core, we use a value of 10^{-6} errors/s for λ_0 and assume $d = 3$ as in [11] and [13]. However, we use $\lambda_0 = 10^{-6}/3$ for NoC routers, given that our router area is roughly a third of the area of a compute core. The reliability of a core or an NoC router is given by

$$\mathbb{R}(\omega_j) = e^{-\lambda(\omega_j)\cdot\tau} \tag{2}$$

where τ is the execution time of the component (time duration that the component stays active). Assuming n = app-DoP, reliability of the ith application running on a CMP can be given as the product of all $2n$ (n routers and n compute cores) component reliabilities

$$R_i = \prod_{2n} R(\omega_j). \tag{3}$$

Prior works [1], [2] have shown that at technology nodes of 32 nm and below, process variations have almost no effect on SER. Therefore, in this paper, we also assume no dependence of V_T variations on SER, instead exploiting variations for speed and power benefits only.

B. Inputs, Assumptions, and Problem Objective

We assume the following inputs to our problem.

1) A CMP with a regular mesh-based 2-D NoC, with Ttiles: $T = (d^2)$, where d is the mesh dimension, and each tile consists of a compute core and an NoC router.
2) A set S of candidate supply voltage (V_{dd}) levels for the chip.
3) A set of NV_T maps (N test chips) incorporating the effects of WID and D2D variations with continuous distribution over the die; the estimated leakage power profile for a given value of V_{dd} can be obtained from a V_T map [using the relation shown in Fig. 1(e)].
4) Application sequence s of length l, made up of η different applications, with arbitrary application interarrival times.
5) Application task graphs for the set $P = \{P_1, P_2, \ldots P_\eta\}$ of DoPs for all applications; an application i possesses $|P_i|$ viable DoPs; an application has a maximum DoP value beyond which performance does not improve (or gets worse). Such suboptimal DoP values are ignored.
6) Vertices of each task graph with execution times of compute cores and edges with intertask communication volumes; execution time and volume values are assumed available from offline profiling.
7) Energy-optimal frequency constraint $\{f_1, f_2, \ldots, f_\eta\}$, minimum reliability constraints $\{Rc_1, Rc_2, \ldots, Rc_\eta\}$, and designated MCs $\{MC_1, MC_2, \ldots, MC_\eta\}$ for all η applications.
8) A chip-wide dark-silicon power dissipation constraint (DS-Pc).

We make the following assumptions in this paper.

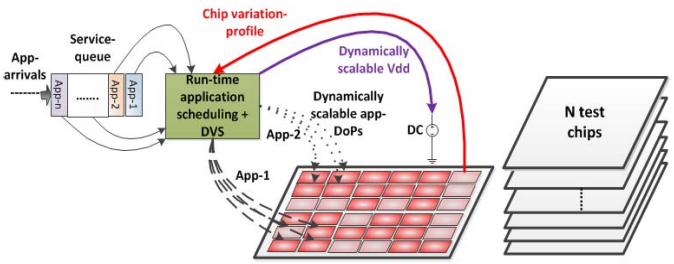

Fig. 2. High-level overview of our proposed application-scheduling + DVS framework (VARSHA++) for CMPs.

1) There exists one-to-one mapping between tasks and cores.
2) Variation-map data for a chip is available at runtime, in terms of the threshold voltage (V_T) distribution, from the chip-frequency profile obtained using distributed ring-oscillator-based delay sensors [20].
3) There exists a chip wide supply voltage that can be scaled using DVS at runtime, avoiding the overheads of implementing DVS at a per-core granularity.
4) As the applications are not necessarily mapped in contiguous regions of the CMP die, there exists interapplication interference, which is quantified in this paper with cycle-accurate NoC simulation.
5) MCs are placed at the four corners of the CMP die; all memory traffic in the NoC for any application is directed to and from a single MC.
6) All compute cores executing an application run at the same frequency to avoid imbalances during multithreaded execution [10], while the NoC fabric runs at a fixed NoC frequency.

Problem Objective: Given the above inputs and assumptions, our objective is to perform runtime application scheduling and DVS on a given CMP platform, such that the average application service time and average energy (across all N test chips) are minimized, while all application-specific operating frequency and reliability constraints, as well as CMP platform-specific DS-Pc are satisfied.

V. VARSHA++ Framework: Overview

VARSHA++ represents a holistic runtime application-mapping and DVS framework that processes all waiting applications in the service queue of a CMP with the goal of: 1) selecting a set of applications to be mapped in the current mapping interval; 2) determining the DoPs of the selected applications; 3) determining the set of tiles on the CMP to map specific application tasks to; and 4) determining the V_{dd} level of the CMP, while leveraging the knowledge of the process variation profile of the chip under consideration.

Fig. 2 shows the key aspects of our VARSHA++ framework. The knowledge of the chip-variation profile is continuously utilized in the scheduling and DVS steps. Assuming equal priority for all incoming applications, the application-scheduling step consists of: 1) determining the DoP (out of the $|P_i|$ DoPs) for each waiting application in the service queue and 2) mapping the appropriate task graphs onto the tiles of the CMP. For a given V_{dd}, scaling-up of app-DoPs is constrained by the available power slack (difference between DS-Pc and current system-power dissipation), application-reliability constraints, and available tiles that meet the application-frequency constraints. At any given time, the scaling-down of V_{dd} (to save power/energy) is constrained by the frequency and reliability constraints of the applications running on the CMP, whereas scaling-up of V_{dd} (to boost app-DoPs) is constrained by the DS-Pc for the CMP.

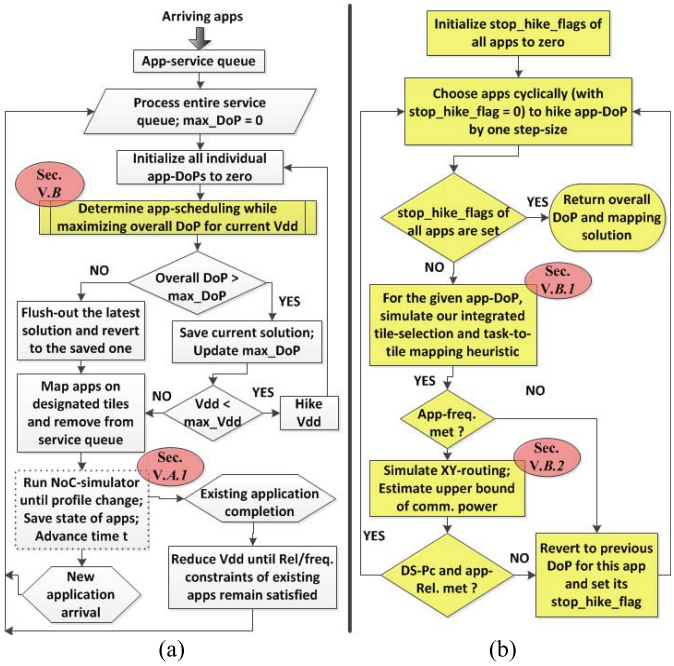

Fig. 3. Design flows for the proposed VARSHA++ framework. (a) V_{dd}-level selection (Section V-A). (b) Determination of application schedule for the current V_{dd} level (Section V-B).

The proposed framework is effectively executed in two nested procedures: 1) V_{dd}-level selection (outer loop), triggered on an arrival or a departure of any application and 2) determination of application schedule for the current V_{dd} level (inner loop). These procedures are discussed in detail in Sections V-A and V-B, respectively, and the corresponding design flows for the procedures are shown in Fig. 3(a) and (b), respectively.

A. V_{dd}-Level Selection

To extract maximum performance from the applications being considered for mapping at any instant of time, the first-order objective is to maximize overall (aggregate) DoP for all applications in the system, at the current time. Recall that an application typically has a maximum viable DoP, and higher DoPs can cause performance to degrade (e.g., due to high synchronization overheads)—such higher DoP configurations are ignored by our framework (Section IV-B). Our V_{dd}-level selection heuristic [Fig. 3(a)] selects the V_{dd} level that yields the maximum overall DoP. As a second-order power/energy saving objective, on completion of any application, the heuristic also reduces V_{dd} to the lowest allowable level that would not introduce any violations in frequency and reliability constraints of existing (already running) applications.

We assume all incoming applications are buffered in a service queue [*App-service queue* in Fig. 3(a)]. On arrival or completion of any application, the V_{dd}-selection heuristic is triggered, which processes the entire service queue. The V_{dd}-selection heuristic iteratively invokes the application-schedule determination procedure (discussed in Section V-B), which produces the mapping solution with the highest overall DoP corresponding to the current V_{dd} level. The V_{dd} level is hiked iteratively (in increments of 0.1 V) and the highest overall DoP (sum total of DoPs of all executing applications) obtained thus far is recorded at each step (*max_DoP*). This continues until either the overall DoP reduces compared with *max_DoP*, in which case the immediately preceding solution with the highest overall DoP is reverted to, or the maximum allowable V_{dd} level (*max_*V_{dd}) is reached. Note that the overall DoP may increase with increasing V_{dd} levels, as more applications satisfy frequency and reliability constraints for higher DoPs; at the same time, the chip power will reach the DS-Pc quicker at higher V_{dd} levels, thereby limiting overall DoP. Therefore, our search for the optimal V_{dd} level culminates when the increase in overall DoP is limited by the DS-Pc. Finally, the best application-mapping solution with the highest overall DoP is mapped to the CMP. The voltage supply is changed to the selected V_{dd} level, and the mapped applications are then removed from the service queue.

1) NoC Simulation: To accurately estimate latencies (and thus application completion times) due to intraapplication communication as well as the memory traffic in the presence of possible interapplication interference (note that individual NoC routers route packets from multiple applications) on the mesh-based NoC fabric, we employ a cycle-accurate NoC simulator [41]. A dimension-order routing algorithm for mesh-based networks, *XY* routing [43], where packets are routed first in the x-direction and then in the y-direction to the destination is utilized. In the event of either arrival or completion of an application, the state of the computation and communication profile of the CMP could potentially change. Therefore, at this time, the current NoC-simulation instance is terminated; time t of the simulation engine is advanced appropriately, and the current state of CMP is saved, i.e., the number of packets received by the destination core of each communication flow is recorded. This enables the NoC simulation to accurately resume at its next invocation. Note that the dotted outline of the NoC-simulation block [in Fig. 3(a)] signifies that this step simulates the runtime NoC behavior and is not needed when utilizing our framework on real hardware, where NoC latency, in the presence of possible runtime congestion, for a communication event can be easily obtained after the communication event finishes.

B. Determination of Application Schedule

Given a specific execution environment for the CMP (including the V_{dd} level, available power slack, and variation profile), the objective of the application-schedule determination heuristic is to maximize the overall DoP, while simultaneously considering all applications in the service queue and satisfying application-frequency and -reliability constraints. Fig. 3(b) shows the design flow of this heuristic. Starting at the least possible DoP value of zero (DoP of zero leaves the application unmapped at the current time), applications are considered cyclically for hiking of DoP to their next higher valid DoP level. Here, to extract maximum performance from the CMP, we choose applications for hiking of DoP in order of their compute intensiveness, because of the relatively smaller communication delay and communication power overheads for compute-intensive applications at higher DoPs. Also, as performance generally does not scale in a perfectly linear manner with increasing app-DoPs, application execution turns out to be more energy efficient at lower DoPs due to lower synchronization and communication performance overheads as well as lower power dissipation, i.e., running four applications simultaneously, each with DoP $= 4$, is typically more energy efficient than running them one after the other with DoP $= 16$ each. Therefore, in our heuristic, we change (hike) app-DoPs symmetrically across all applications to improve performance and energy efficiency collectively for the applications.

To produce an optimal mapping for the application under consideration (with a specific DoP), the following steps are performed: 1) integrated tile-selection and task-to-tile mapping given the task graph for the current DoP (Section V-B1) and 2) communication-flow routing and delay/power analysis (Section V-B2). After the steps mentioned earlier, the mapping is evaluated for overall power footprint and reliability of the applications. Satisfaction of the application-frequency constraints is also checked during the tile-selection and application-mapping step. As shown in Fig. 3(b), the DoP hike of any application could fail due to potential violation(s) in application-frequency constraints, application-reliability constraints, or DS-Pc. When an attempted DoP hike is stalled for any application, its *stop_hike_flag* is set to preclude it from future DoP-hike consideration, and the feasible mapping with the preceding DoP is finalized for this application.

In Sections V-B1 and V-B2, we discuss the core details of our application-scheduling approach. Section V-B1 discusses our integrated tile-selection and mapping heuristic and its theoretical run time complexity. Section V-B2 discusses our routing and communication power evaluation.

1) Integrated Tile-Selection and Mapping Heuristic: We first motivate the need for an integrated tile-selection and task-to-tile mapping approach with a small example. Given the V_T map for a die, the maximum frequency that each core on the die can be reliably clocked at depends upon the V_T values as well as the V_{dd} values, and the relationship can be expressed as

$$f_{max} = \frac{\mu(V_{dd} - V_T)^{\alpha}}{C_0 \cdot V_{dd}} \tag{4}$$

where α and μ are technology-dependent constants and C_0 is switching capacitance of the critical path [30]. In our approach, we utilize the knowledge of both frequency and leakage-power profiles of the chip. Note that dynamic power remains unaffected by V_T variations and is thus not considered in this step. Our objective is to find the set of tiles on the mesh, such that leakage power and communication latency and energy are minimized, while satisfying the frequency constraint of the

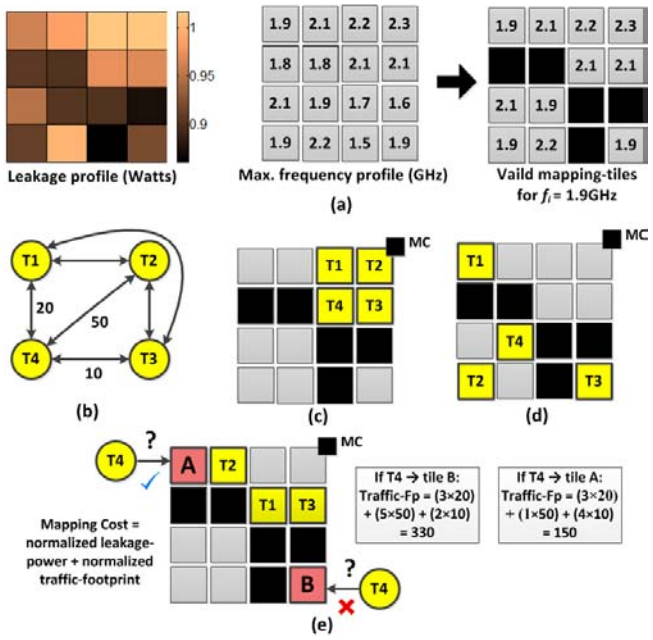

Fig. 4. Example of our integrated tile-selection and application-mapping heuristic. (a) Leakage power map and feasible mapping tiles for application-frequency constraint of 1.9 GHz. (b) Application task graph with app-DoP = 4. (c) Solution minimizing communication energy/latency. (d) Solution minimizing computation energy. (e) Our proposed solution.

application being mapped.

Fig. 4 shows an illustration of our approach for a 4×4 mesh-based CMP. The app-DoP equal to 4 [shown in Fig. 4(b)] with a frequency constraint of 1.9 GHz can only be mapped onto feasible tiles that satisfy its minimum frequency constraints [as shown in Fig. 4(a)]. Fig. 4(c) shows a mapping that exclusively minimizes the communication overheads in terms of latency and energy by minimizing the communication Manhattan distances (MDs) for intertask communication as well as memory communication. On the other hand, Fig. 4(d) shows an application-mapping solution that exclusively minimizes the computation leakage power by choosing tiles with minimum leakage from the chip-leakage profile shown in Fig. 4(a).

Fig. 4(e) shows our solution. Let us assume that tasks T1–T3 have already been mapped (as shown) using a mapping cost function that considers both the core leakage power and the resulting communication overheads. Now, tile A and tile B, both of which correspond to the same MD (in terms of number of hops) from the appropriate MC and similar leakage powers, are feasible candidates to map the task T4. But the two candidates produce significantly different traffic-fp's on the underlying NoC fabric, as shown in Fig. 4(e). Note that similar optimization metrics have been used in prior works, such as [42]. The traffic-fp is calculated as

$$\text{traffic-fp} = \sum_{\forall \text{ flows}} \text{MD} \times comm.\ volume. \tag{5}$$

Mapping T4 to tile A reduces the communication latency and thereby the application runtimes. Note that, in this example, the communication volumes to and from the MCs are ignored as the MDs of tile A and tile B from the appropriate MC are equal. Also note that the traffic-fp metric can only consider communication flows associated with tasks that have already been mapped, in addition to communication flows to and from the appropriate MC. Our approach, therefore, is able to intelligently tradeoff communication overheads with computation energy, considering leakage power as well as NoC traffic-fp during the tile-selection step. Now, we present the details of our integrated tile-selection and mapping approach. The main idea behind our integrated tile-selection and mapping heuristic is that by considering the leakage (variation) characteristics of CMP tiles and communication characteristics of applications in tandem, superior power and latency profiles are achievable. The pseudocode of the heuristic is shown in Algorithm 1. The problem objective is to optimize both the computation and the communication profiles of the application i to be mapped, given the application characteristics (task graph, app-DoP, f_i, and MC_i) and the CMP-platform characteristics (power slack, V_{dd}, and variation profile) at current time t. We first sort the tasks in decreasing order of communication volumes (ingress and egress volumes) to be mapped in that order (step 1). In order to limit the memory traffic in the NoC, we restrict our mapping search space to a square region in the CMP-corner adjoining MC_i (step 2). Note that the application is not necessarily mapped contiguously within the square region. The size of this square region, *Sq_size*, is determined by the app-DoP or the number of tiles that application i maps to, and can be expressed as

$$Sq\_size = \max(\lceil(\sqrt{c \times \text{DoP}})\rceil^2, min\_sq\_size) \tag{6}$$

where c and *min_sq_size* are constants. To avoid frequent application stalls due to unavailability of tiles when mapping the smallest app-DoPs, we use *min_sq_size* as the smallest region size of the search space. Now, we map the first task from the sorted list onto an unmapped tile that satisfies the f_i constraint with the smallest cost C1 (step 3). For mapping the rest of the tasks (step 4), the traffic-fp corresponding to each of the available tiles (meeting the f_i constraint) within the square region is calculated (step 5) as explained earlier in this section (5). Using these traffic-fp values, the mapping cost C2 is calculated for each valid tile, and the task is mapped onto the tile with minimum cost C2 (step 6).

Theoretical time-complexity analysis of our tile-selection and mapping heuristic: The size of the square region to

Algorithm 1 Integrated Tile-Selection and Mapping Heuristic

inputs: Application task graph of the appropriate DoP and CMP platform characteristics at current time t

1: sort the tasks in the order of decreasing communication volumes – to be mapped sequentially in this order
2: designate a square region (of size given by Eq. 6) in the CMP-corner adjoining the appropriate MC for mapping the application
3: map the first task onto an available tile on the square region with the least cost C1= normalized leakage-power + normalized MD from MC
4: for all remaining tasks in the sorted list, **do {**
5: compute the *traffic-fp* w.r.t. already mapped tasks (and MC) corresponding to mapping the current task onto all available tiles
6: map the task onto the tile on the square region with the least cost: C2 = normalized leakage-power + normalized traffic-fp
7: }

output: mapping solution of application task graph on to the CMP die

be evaluated (for cost C1 or C2) for mapping each task is proportional to the app-DoP and there are app-DoP numbers of tasks to be mapped. Also, to calculate the traffic-fp of each candidate tile, up to app-DoP communication flows may have to be evaluated. Thus, the time complexity for our integrated mapping heuristic would be of the order of O((app-DoP)^3). Note that app-DoP is typically a small integer (between 4 and 16 in our experiments) and can be treated as a constant.

2) Communication Power Estimation: Similar to numerous prior works (see [24]), we use low-cost *XY*-routing to route the communication flows of applications. Average dynamic power of NoC routers and links (corresponding to different voltages, communication loads, and router sizes) is assumed to be saved in the read-only (or nonvolatile) memory on the CMP die, and accessible by our framework at runtime. Router leakage powers, estimated from the chip-variation profile, are added to the appropriate dynamic power values to produce the total communication power for running each application. Based on the active times (execution times) of routers and compute cores, the application reliability is computed using (1)–(3). For our analyses, the energies and runtimes of applications are calculated from component powers and active times.

C. Complexity Analysis of VARSHA++ Framework

Our application-schedule determination heuristic (discussed in Section V-B), which maximizes the DoP of all applications being processed at the current V_{dd} level, attempts to find mapping solutions for the w waiting applications for up to $|P_i|$ DoP levels. This heuristic could be required to execute for at most $|S|$ (total number of candidate V_{dd} levels) times. $|S|$ and $|P_i|$ are small constant integers in our experiments. Also, at any given time, only a small number of applications (up to w) are generally expected to be processed given the DS-Pc. Therefore, whenever the service queue is processed in our framework, the number of times that our integrated tile-selection and application-mapping heuristic would need to be invoked is bounded by a relatively small constant integer.

In addition, app-DoP number of tasks in an application would require up to (app-DoP)^{2}communication flows to be routed. With XY-routing, the path (in terms of number of *X*-hops and *Y*-hops) can be easily computed from the coordinates of the destination core in constant time. Also, given the application-mapping area [as shown in (6)], any routing path is up to $(\sqrt{2 \times c})\times$ app-DoP number of hops long. Therefore, both our tile-selection and application-mapping step (discussed in Section V-B1), and our communication-routing step are bound by O((app-DoP)^3 or constant time complexity. As these steps correspond to the highest theoretical time complexity in the design flow, the time complexity of our framework can be bounded in constant time. In other words, our framework is scalable with respect to the CMP size (number of tiles T), and thus amenable for use at runtime.

VI. Experiments

Our experiments were conducted using $\eta = 14$ different parallel application benchmarks: seven from the SPLASH-2 benchmark suite [25] (*cholesky, fft, lu, ocean, radix, radiosity,* and *raytrace*), and seven from the PARSEC benchmark suite [26] (*vips, swaptions, fluidanimate, dedup, streamcluster, canneal*, and *blackscholes*). We consider DoPs that are multiples of 4, up to 16, where a DoP of 8 is considered the nominal DoP value, as a reasonable tradeoff between speed and energy. As discussed earlier, every application has a unique maximum viable DoP beyond which further performance gains cannot be achieved. Our task graphs are modeled based on the intercore communication characterization from [28] and based on our observations of traces and communication patterns between the respective pair of cores. The reliability constraints of different applications are set in the range: 0.99–0.999.

We assume the ARM Cortex-A9 processors [27] as the baseline CMP compute cores, which support five operating voltage levels ($|S| = 5$): 0.8, 0.9, 1.0, 1.1, and 1.2 V. Note that although we consider chip-wide DVS, our framework can be easily adapted when per-core voltage scaling is used. Under such a scenario, the framework will not generate application schedules corresponding to multiple V_{dd} levels, given that each core can run at the lowest allowable supply voltage. The application-specific energy-optimal core frequencies range from 1300 to 1900 MHz, based on the level of compute intensity of the tasks assigned to cores. We consider an 81-core mesh-based CMP platform with dimensions 9×9 for our experiments. The dark-silicon power constraint (DS-Pc) is set to 50 W. In our integrated tile-selection and mapping heuristic (discussed in Section V-B1), we use a value of 1.5 for the constant c [in (6)] when computing the size of the square region to map any application. However, a minimum size (*min_sq_size*) of 16 tiles is used for this region to avoid excessive stalling of applications at the highest arrival rates. The delay overhead for V_{dd} scaling (PLL lock time) is estimated to be less than 5 μs, similar to [12], which is negligible compared with the granularity of application runtimes (2–9 s each) in our experiments.

To investigate the applicability of our approach to CMP dies with diverse variation profiles, we use either 100 or 1000 test chips ($N = 100$ or $N = 1000$) in different experiments. The V_T maps corresponding to these test chips are generated using the open-source tool [29] (based on systematic and random WID-variation model in [30]). The values of 0.3 and 0.09 are used for the statistical mean and a standard deviation of the parameter V_T, respectively, and a correlation range (ϕ) of 0.5 is used (as recommended in [30]). A normally distributed V_T bias representing the D2D variation component is superimposed onto these V_T maps; the standard deviation of the D2D V_T is assumed to be 6%, as in [31]. Each V_T map represents a 27×27 grid of points corresponding to nine ring oscillator test sites for each core on the CMP. For a given V_T map, the maximum core frequencies are calculated by using the V_T-max values, and the core-leakage powers are calculated using the V_T-avg values (out of the 9 V_T values per core). We use the following standard relationship between V_T and V_{dd} to calculate the leakage power of cores:

$$P_{\text{leakage}} = I_0 \cdot e^{\frac{-V_T}{nKT/q}} \cdot V_{\text{dd}} \tag{7}$$

where I_0 is the technology-dependent subthreshold leakage, KT/q is the thermal voltage, and n is the subthreshold swing coefficient [44].

The power values of routers and links (32-bit wide) for different voltages and frequencies at varying communication loads, for the 32-nm technology node, are obtained from ORION 2.0 [32]. Note that the router power values obtained are for nominal V_T, and are scaled for varying V_T values.

A. Experimental Results

We compare the results obtained from our proposed VARSHA++ framework with those obtained from using runtime application-mapping frameworks proposed in [10], [24], and [40] (VARSHA). A variation- and dark-silicon-aware mapping technique is proposed in [10], whereas [24] advocates for a traditional area-constrained design approach. We implemented these prior works to the best of our understanding. Our experiments considered two unique application sequences (Seq-A and Seq-B) that represent an ordering of arriving application instances, with instances randomly chosen from among the 14 applications considered. For each sequence, we vary the interarrival times of application instances randomly within the following ranges: 0–2 s (Seq-1A and Seq-1B), 0–4 s (Seq-2A and Seq-2B), 0–8 s (Seq-3A and Seq-3B), and 0–16 s (Seq-4A and Seq-4B). We assume $l = 100$ application instances in any application sequence. In the first set of experiments, we consider a set of 100 test chips, representing different variation profiles; our results (in Fig. 5, and Tables I and II) show the mean values across all 100 test chips.

Fig. 5 shows results comparing VARSHA++ with the frameworks from [10], [24], and [40]. The prior works [10] and [24] assume fixed nominal app-DoPs. Our VARSHA++ framework and [40] adapt app-DoPs in accordance with the application interarrival rates to minimize the application service times. Observe in Table I that for both sequences, the average app-DoP reduces with increasing interarrival rates for both of our frameworks. At higher interarrival rates when applications with nominal DoPs cannot be quickly serviced due to the DS-Pc constraint, our frameworks cut down application wait times significantly by reducing DoPs [as shown in Fig. 5(a)—Seq-1A,B and Seq-2A,B], although the application runtimes tend to increase due to the reduction in DoPs. On the other hand, at lower interarrival rates, with on average fewer applications to be serviced simultaneously, our framework opportunistically hikes the app-DoPs also to minimize runtimes [as shown in Fig. 5(b)—Seq-3A,B and Seq-4A,B]. In comparison with [24], we obtain on average 71% savings in average service times with our VARSHA++ framework. Note that maximum savings are obtained when the interarrival rates are most stringent, as shown for Seq-1A and Seq-1B in Fig. 5(a). The communication-unaware framework in [10] maps applications onto regions of noncontiguous tiles to optimize computation power exclusively (without consideration of communication traffic in the NoC), resulting in longer runtimes due to longer communication latencies. Compared with [10], we obtain 71.5% savings in average service times with our VARSHA++ framework.

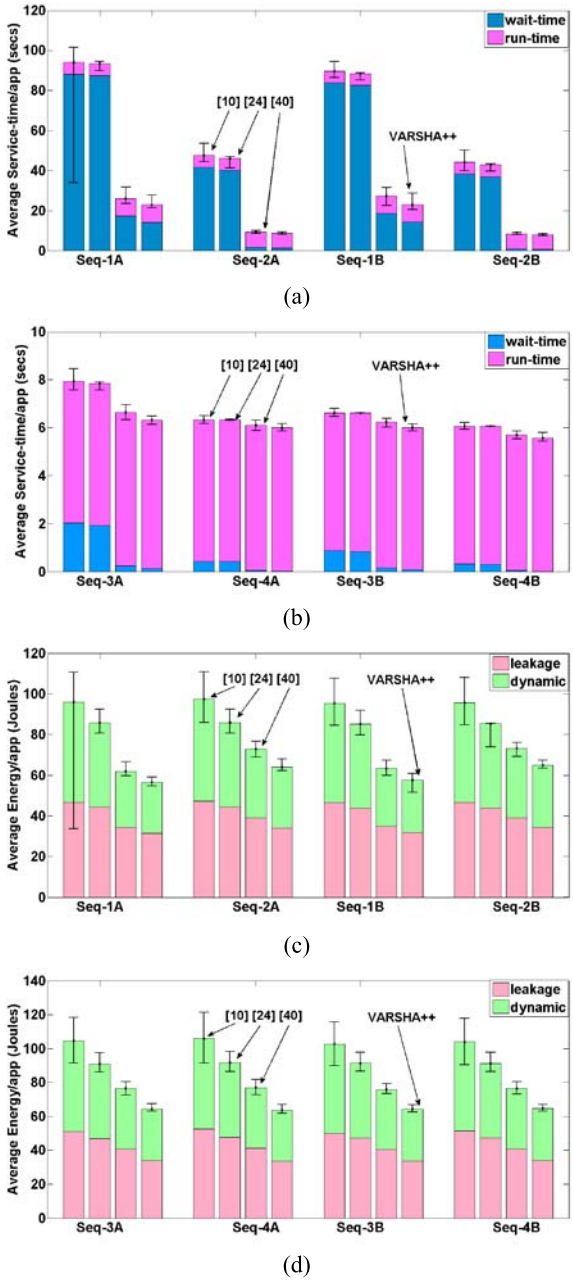

Fig. 5. Results of our proposed framework (VARSHA++) versus frameworks in [10], [24], and [40]. (a) Average service time per application instance (wait time+run time) for Seq-1 and Seq-2. (b) Average service time per application instance (wait time + run time) for Seq-3 and Seq-4. (c) Average energy per application instance (leakage + dynamic) for Seq-1and Seq-2. (d) Average energy per application instance (leakage + dynamic) for Seq-3 and Seq-4. Bars: mean values of service times and energies across 100 test chips, while variation in service times and energies is shown by confidence intervals.

TABLE I

MEAN VALUES ACROSS 100 TEST CHIPS FOR AVERAGE DOP PER APPLICATION INSTANCE FOR [40] VERSUS VARSHA++

	Seq-1A	Seq-2A	Seq-3A	Seq-4A	Seq-1B	Seq-2B	Seq-3B	Seq-4B
[40]	4.21	5.84	8.54	9.57	4.10	5.47	8.98	10.32
VARSHA++	4.24	6.49	9.13	9.86	4.13	6.15	9.41	10.70

Our frameworks also opportunistically lower V_{dd} levels while ensuring that frequency and reliability constraints remain satisfied for the set of existing applications.

TABLE II

MEAN VALUES ACROSS 100 TEST CHIPS FOR WEIGHTED-AVERAGE V_{dd} (V_{w-avg}) FOR [40] VERSUS VARSHA++

	Seq-1A	Seq-2A	Seq-3A	Seq-4A	Seq-1B	Seq-2B	Seq-3B	Seq-4B
[40]	1.10	1.15	1.09	0.97	1.13	1.16	1.08	0.95
VARSHA++	1.10	1.13	1.07	0.96	1.10	1.13	1.06	0.94

The weighted-average V_{dd} metric (V_{w-avg}) represents the average value of V_{dd} throughout the execution of the entire application sequence for a given chip, and is defined as

$$V_{\text{w-avg}} = \frac{((V_{\Delta t1} \times \Delta t1) + (V_{\Delta t2} \times \Delta t2) + \cdots + (V_{\Delta tp} \times \Delta tp))}{\sum_{i=0}^{i=p} \Delta ti} \quad (8)$$

where Δti is the ith time interval during which the V_{dd} level stays constant at a value of $V_{\Delta ti}$, and the execution of the entire application sequence s takes p such time intervals. Table II shows the mean V_{w-avg} across the 100 test chips for both of our frameworks. In the absence of DVS in [10] and [24], in our implementation of these frameworks, we assume the highest voltage of 1.2 V, which is just high enough to meet all application-frequency constraints across all test chips.

Although [10] saves on leakage-power dissipation in computation cores by performing variation-aware mapping, it produces solutions with much longer communication paths along with high V_{dd}, resulting in highest service times and energy values. In comparison, the variation-unaware framework [24] consumes higher computation power but minimizes communication latency and energy by mapping applications onto contiguous regions (as shown in Fig. 5). With variable app-DoPs in our frameworks, energy efficiency decreases with increasing app-DoPs (due to increased communication and sublinear increase in computation performance), while decreasing V_{dd} levels generally cause an increase in energy efficiency. Therefore, in Fig. 5(c) and (d), we observe variation in energy values with different interarrival rates. The maximum energy savings are obtained for the most stringent interarrival rates, as shown for Seq-1A and Seq-1B in Fig. 5(c), due to opportunistic reduction of DoPs. We find from Fig. 5 that VARSHA++ produces on average, 37.6% and 29.3% savings in mean energy (32% and 27% savings in mean leakage energy) and, 71.5% and 71% savings in mean application service time (88% and 87% savings in mean application wait times), over [10] and [24], respectively.

In general, our proposed VARSHA++ framework, which co-optimizes NoC communication (including memory traffic) in addition to leakage power, results in even higher CMP performance compared with [40] due to the following reasons: 1) higher leakage-power savings generate higher power slacks, enabling execution of applications at higher DoPs on average, thereby cutting down on application service times and 2) shorter communication-routing paths produce shorter application runtimes, which in turn enables application-reliability constraints to be satisfied at lower V_{dd} levels (thus saving even more power) and higher app-DoPs (thus boosting performance further). The higher average app-DoPs (by up to 12.4%) and lower V_{w-avg} values (by up to 2.6%) obtained with VARSHA++ in comparison to [40] are shown in Tables I and II, respectively. We obtain improvements of 10.1% in mean application service time (21.7% in mean wait time) and 13.4% in mean energy (14.1% in mean leakage energy), in comparison to the framework proposed in [40].

TABLE III

ENERGY BREAKUP IN TERMS OF COMMUNICATION AND COMPUTATION ENERGY FOR [40] AND VARSHA++ FOR APPLICATION SEQUENCE SEQ-2A

	Avg. comm. energy (J)	**Avg. compute energy (J)**	**Avg. energy (J)**
[40]	20.84	52.08	72.92
VARSHA++	11.55	52.45	64.00
Improvement	44.6%	-0.7%	12.2%

TABLE IV

AVERAGE NUMBER OF RELIABILITY-CONSTRAINT (*Rc*) VIOLATIONS ACROSS THE 100 TEST CHIPS OBTAINED USING THE RELIABILITY-UNAWARE FRAMEWORKS [10] AND [24]

	Seq-1A	Seq-2A	Seq-3A	Seq-4A	Seq-1B	Seq-2B	Seq-3B	Seq-4B
[10]	13.9	13.9	13.4	13.4	13.8	13.7	13.7	13.4
[24]	13.4	13.4	7.15	6.95	12.8	12.8	7.0	5.0

We also present results for the breakdown of energy improvements in terms of communication and computation energy for VARSHA++ in comparison to [40]. For brevity, we focus on one application sequence (Seq-2A). As shown in Table III, VARSHA++ results in significant reduction in the communication energy consumption and only a slight increase in computation energy consumption even with 11% higher average DoP value (as shown in Table I). This general trend of greater communication energy savings for VARSHA++ compared with [40] is also observed for the other sequences.

We note that for applications with relatively more stringent reliability constraints, it may not be possible to support nominal DoP even at the highest V_{dd} level (1.2 V). Therefore, when using reliability-unaware frameworks with no DoP adaptivity, reliability constraints (*Rc*) of such applications may be violated. Table IV shows the average number of application instances incurring *Rc* violations, across the 100 test chips for schedules produced by [10] and [24]. With longer routing distances, [10] consumes more routing resources compared with [24]; thus even with the same app-DoPs (nominal DoPs), the estimated failure rates are greater for [10]. Observe that higher number of *Rc* violations may occur for high interarrival rates. This is due to the higher levels of NoC traffic congestion and higher execution times of hardware components with increased die utilization at higher application interarrival rates. Our reliability-aware frameworks, both [40] and VARSHA++ framework, result in no *Rc* violations because of their ability to dynamically reduce app-DoPs as well as hike V_{dd} levels in accordance with application reliability requirements.

We also perform another set of experiments to investigate the effects of D2D variations in terms of energy efficiency obtained using different frameworks. As discussed earlier, our frameworks, both [40] and VARSHA++, combine a DVS

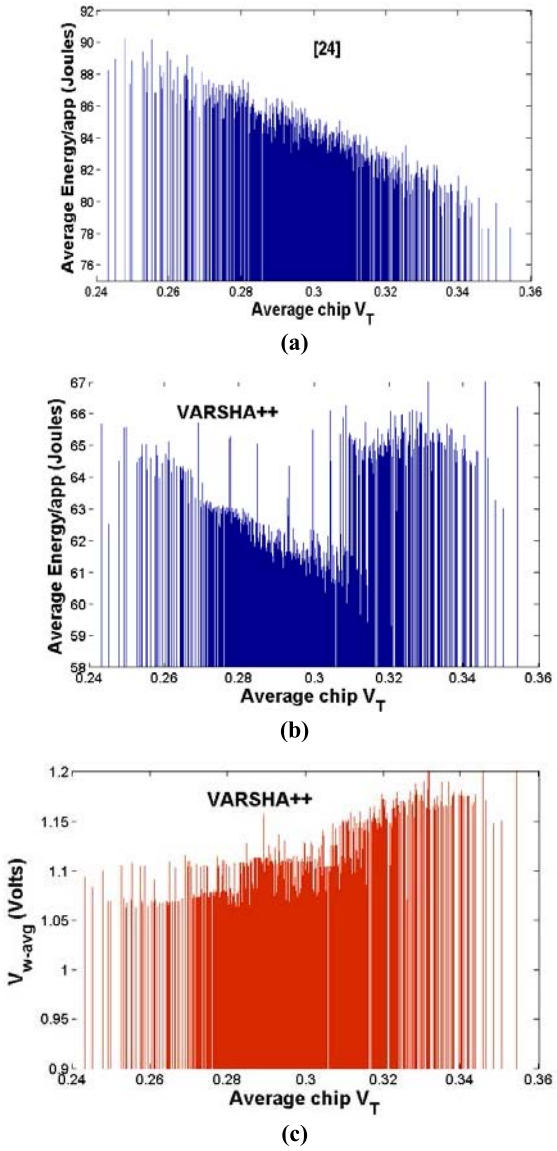

Fig. 6. Effects of D2D variation in terms of average energy/app obtained for 1000 test chips for Seq-2A. (a) With [24], performance degradation due to D2D variations cannot be mitigated efficiently with high V_{dd}, and faster chips generally result in much worse energy efficiency. (b) Due to intelligent DVS scheme integrated within the VARSHA++ framework, the best energy efficiency is obtained for test chips with moderate leakage and performance profiles. (c) V_{w-avg} distribution with VARSHA++.

scheme with an application-mapping methodology. This facilitates the mitigation of the adverse effects of D2D variations by enabling faster chips to utilize lower V_{dd} values and slower chips to utilize higher V_{dd} values while satisfying reliability and frequency constraints at all times. On the other hand, prior works, such as [10] and [24], do not employ DVS in their application-mapping methodology, instead utilizing a fixed high value of V_{dd} for all test chips, thereby resulting in significantly worse energy efficiencies over the entire set of test chips. Thus, in this set of experiments, we select one of our frameworks (VARSHA++) that employs DVS in order to mitigate effects of D2D variations, and a prior work [24] that does not employ DVS to analyze the impact of how D2D variations are addressed by these frameworks. Also, here, we consider a much bigger test set (of a 1000 test chips) while considering just one application sequence, Seq-2A.

Fig. 6 shows the average energy consumed per application for all 1000 test chips across VARSHA++ and [24]. The experimental results indicate that for the framework from [24] [Fig. 6(a)], where a fixed high value of V_{dd} is set pessimistically for the worst case (highest) V_T values, slower chips (those with high V_T and lower leakage) could prove to be advantageous as they would be left with greater power slack to accommodate more applications at any given time (i.e., less percentage of dark silicon), thus resulting in better service times and energy profiles. We observe that the energy efficiency obtained using [24] generally steadily reduces, as the leakage power of test chips increases [going from right to left in Fig. 6(a)].

Conversely, the VARSHA++ framework can intelligently mitigate the adverse effects of D2D variations, resulting in much higher energy efficiency, as shown in Fig. 6(b). With VARSHA++, given a DS-Pc, faster chips (that also dissipate higher leakage power) can usually utilize lower V_{dd} levels to minimize energy, and slower chips (that dissipate lower leakage power) can utilize higher V_{dd} levels to improve performance, as shown in Fig. 6(c). We observe that test chips with moderate leakage and performance profiles produce desirable energy results, whereas chips that are too leaky or too slow yield worse results, as shown in Fig. 6(b). This phenomenon can be explained as follows. Although fast chips could utilize lower voltage levels, the average power dissipation is generally high with a higher leakage power component. At the same time, average power dissipation is also high for slow chips (with high V_T values) due to high supply voltage levels. On the contrary, chips with moderate V_T values correspond to moderately low leakage power and V_{dd} levels, resulting in lower average power dissipation per core, which results in potentially higher silicon utilization (lower dark-silicon levels) given a fixed DS-Pc. Higher utilization can result in higher average app-DoPs and shorter application runtimes. Thus, with generally lower power dissipation and slightly higher levels of utilization, we observe better energy efficiency for test chips with moderate V_T values. Note that as we hike app-DoPs in multiples of 4 in our experiments, we see a discrete change in the general trend of average energy values at around average chip V_T of 0.31 V due to a step change in utilization levels in Fig. 6(b).

VII. Conclusion and Future Work

In this paper, we proposed an application-mapping and DVS framework, which represents one of the first efforts to integrate reliability and variation awareness in a runtime variable DoP-based application-scheduling methodology to enhance performance of multicore systems in the dark-silicon era. We combine the traditionally disjoint steps of region-selection and task-to-tile mapping to co-optimize memory communication, NoC traffic, and leakage power. Tailored for deeply scaled technologies, our lightweight runtime framework generates highly optimized application-mapping solutions. Our experimental results show that the proposed framework produces

average savings of 10%–71% in application service times, 13%–38% in energy, and avoids reliability violations unlike the state-of-the-art prior works that suffer from reliability violations in up to 14% of application instances arriving at runtime.

We note that as our tile-selection and mapping heuristic favors tiles with proximity to MCs, cores closer to the corners of the chip are expected to have higher utilization, and thus are prone to greater wear out. We leave addressing the aging aspects of our proposed framework for future work. In the future, we intend to devise new runtime application-mapping and dynamic voltage frequency scaling techniques to simultaneously optimize soft-error reliability and lifetime reliability in multicore chips, while also considering the overheads of task reexecution due to soft errors.

REFERENCES

[1] A. Kaouache, F. Wrobel, F. Saigné, A. D. Touboul, and R. D. Schrimpf, "Analytical method to evaluate soft error rate due to alpha contamination," *IEEE Trans. Nucl. Sci.*, vol. 60, no. 6, pp. 4059–4066, Dec. 2013.

[2] N. Gaspard *et al.*, "Effect of threshold voltage implants on single-event error rates of D flip-flops in 28-nm bulk CMOS," in *Proc. IEEE Int. Rel. Phys. Symp.*, Apr. 2013, pp. SE.7.1–SE.7.3.

[3] S.-I. Abe *et al.*, "Neutron-induced soft error analysis in MOSFETs from a 65 nm to a 25 nm design rule using multi-scale Monte Carlo simulation method," in *Proc. IEEE Int. Rel. Phys. Symp.*, Apr. 2012, pp. SE.3.1–SE.3.6.

[4] D. Zhu, R. Melhem, and D. Mosse, "The effects of energy management on reliability in real-time embedded systems," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2004, pp. 35–40.

[5] E. Humenay, D. Tarjan, and K. Skadron, "Impact of process variations on multicore performance symmetry," in *Proc. ACM/IEEE Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, Apr. 2007, pp. 1–6.

[6] L.-T. Pang, K. Qian, C. J. Spanos, and B. Nikolić, "Measurement and analysis of variability in 45 nm strained-Si CMOS technology," *IEEE J. Solid-State Circuits*, vol. 44, no. 8, pp. 2233–2243, Aug. 2009.

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

[8] S. Borkar, "Design perspectives on 22 nm CMOS and beyond," in *Proc. IEEE/ACM Design Autom. Conf. (DAC)*, Jul. 2009, pp. 93–94.

[9] J. Allred, S. Roy, and K. Chakraborty, "Designing for dark silicon: A methodological perspective on energy efficient systems," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2012, pp. 255–260.

[10] B. Raghunathan, Y. Turakhia, S. Garg, and D. Marculescu, "Cherry-picking: Exploiting process variations in dark-silicon homogeneous chip multi-processors," in *Proc. ACM/IEEE Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2013, pp. 39–44.

[11] A. Das, A. Kumar, B. Veeravalli, C. Bolchini, and A. Miele, "Combined DVFS and mapping exploration for lifetime and soft-error susceptibility improvement in MPSoCs," in *Proc. ACM/IEEE Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2014, pp. 1–6.

[12] A. Bashir *et al.*, "Fast lock scheme for phase-locked loops," in *Proc. IEEE Custom Integr. Circuits Conf.*, Sep. 2009, pp. 319–322.

[13] M. A. Haque, H. Aydin, and D. Zhu, "Energy-aware task replication to manage reliability for periodic real-time applications on multicore platforms," in *Proc. Int. Green Comput. Conf. (IGCC)*, Jun. 2013, pp. 1–11.

[14] T. D. Loveless *et al.*, "Neutron- and proton-induced single event upsets for D- and DICE-flip/flop designs at a 40 nm technology node," *IEEE Trans. Nucl. Sci.*, vol. 58, no. 3, pp. 1008–1014, Jun. 2011.

[15] J. Li and J. F. Martinez, "Dynamic power-performance adaptation of parallel computation on chip multiprocessors," in *Proc. 12th Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Feb. 2006, pp. 77–87.

[16] Y. Ding, M. Kandemir, P. Raghavan, and M. J. Irwin, "A helper thread based EDP reduction scheme for adapting application execution in CMPs," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. (IPDPS)*, Apr. 2008, pp. 1–14.

[17] Y. Turakhia, B. Raghunathan, S. Garg, and D. Marculescu, "HaDeS: Architectural synthesis for heterogeneous dark silicon chip multi-processors," in *Proc. 50th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, May 2013, pp. 1–7.

[18] A. Raman, H. Kim, T. Oh, J. W. Lee, and D. I. August, "Parallelism orchestration using DoPE: The degree of parallelism executive," in *Proc. 32nd ACM SIGPLAN Conf. Program. Lang. Design Implement. (PLDI)*, Jun. 2011, pp. 26–37.

[19] L. Zhang, L. S. Bai, R. P. Dick, L. Shang, and R. Joseph, "Process variation characterization of chip-level multiprocessors," in *Proc. 46th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2009, pp. 694–697.

[20] X. Wang, M. Tehranipoor, S. George, D. Tran, and L. Winemberg, "Design and analysis of a delay sensor applicable to process/environmental variations and aging measurements," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 8, pp. 1405–1418, Aug. 2012.

[21] A. A. M. Bsoul, N. Manjikian, and L. Shang, "Reliability- and process variation-aware placement for FPGAs," in *Proc. ACM/IEEE Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2010, pp. 1809–1814.

[22] B. Raghunathan and S. Garg, "Job arrival rate aware scheduling for asymmetric multi-core servers in the dark silicon era," in *Proc. ACM Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES)*, Oct. 2014, Art. no. 14.

[23] C.-L. Chou, U. Y. Ogras, and R. Marculescu, "Energy- and performance-aware incremental mapping for networks on chip with multiple voltage levels," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 10, pp. 1866–1879, Oct. 2008.

[24] M. Fattah, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Smart hill climbing for agile dynamic mapping in many-core systems," in *Proc. 50th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, May 2013, pp. 1–6.

[25] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," in *Proc. 22nd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 1995, pp. 24–36.

[26] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *Proc. ACM 17th Int. Conf. Parallel Archit. Compil. Techn. (PACT)*, Oct. 2008, pp. 72–81.

[27] ARM. *ARM Cortex-A9*, accessed on Feb. 10, 2016. [Online]. Available: http://www.arm.com/products/processors/selector.php

[28] N. Barrow-Williams, C. Fensch, and S. W. Moore, "A communication characterisation of SPLASH-2 and PARSEC," in *Proc. IEEE IISWC*, Oct. 2009, pp. 86–97.

[29] *VARIUS Model*, accessed on accessed Feb. 10, 2016. [Online]. Available: http://www.cse.ohiostate.edu/~teodores/arch/tools/

[30] 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.

[31] B. Li, L.-S. Peh, and P. Patra, "Impact of process and temperature variations on network-on-chip design exploration," in *Proc. 2nd ACM/IEEE Int. Symp. Netw.-Chip (NOCS)*, Apr. 2008, pp. 117–126.

[32] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi, "ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration," in *Proc. ACM/IEEE Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, Apr. 2009, pp. 423–428.

[33] N. A. Kapadia and S. Pasricha, "VISION: A framework for voltage island aware synthesis of interconnection networks-on-chip," in *Proc. 21st ed. Great Lakes Symp. VLSI (GLSVLSI)*, 2011, pp. 31–36.

[34] N. Kapadia and S. Pasricha, "A framework for low power synthesis of interconnection networks-on-chip with multiple voltage islands," *Integr., VLSI J.*, vol. 45, no. 3, pp. 271–281, Jun. 2012.

[35] N. Kapadia and S. Pasricha, "VERVE: A framework for variation-aware energy efficient synthesis of NoC-based MPSoCs with voltage islands," in *Proc. IEEE 14th Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2013, pp. 603–610.

[36] L. Huang and Q. Xu, "Performance yield-driven task allocation and scheduling for MPSoCs under process variation," in *Proc. 47th ACM/IEEE Design Autom. Conf. (DAC)*, Jun. 2010, pp. 326–331.

[37] N. Kapadia and S. Pasricha, "Process variation aware synthesis of application-specific MPSoCs to maximize yield," in *Proc. IEEE 27th Int. Conf. VLSI Design, 13th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2014, pp. 270–275.

[38] S.-H. Kang, H. Yang, S. Kim, I. Bacivarov, S. Ha, and L. Thiele, "Static mapping of mixed-critical applications for fault-tolerant MPSoCs," in *Proc. 51st ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2014, pp. 1–6.
[39] M. Salehi *et al.*, "dsReliM: Power-constrained reliability management in dark-silicon many-core chips under process variations," in *Proc. ACM 10th Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES)*, Oct. 2015, pp. 75–82.
[40] N. Kapadia and S. Pasricha, "VARSHA: Variation and reliability-aware application scheduling with adaptive parallelism in the dark-silicon era," in *Proc. ACM/IEEE Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2015, pp. 1060–1065.
[41] *Nirgam Simulator*. [Online]. Available: http://nirgam.ecs.soton.ac.uk/
[42] W. Jang, D. Ding, and D. Z. Pan, "A voltage-frequency island aware energy optimization framework for networks-on-chip," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2008, pp. 264–269.
[43] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. San Mateo, CA, USA: Morgan Kaufmann, 2004.
[44] J. Rabaey, *Low Power Design Essentials*. New York, NY, USA: Springer, 2009.

Nishit Kapadia (S'12–M'16) received the master's degree in electrical and computer engineering from California State University, Northridge, CA, USA, in 2005, and the Ph.D. degree in electrical and computer engineering from Colorado State University, Fort Collins, CO, USA, in 2016.

He was a VLSI Faculty Member with the Department of Electronics, Sardar Vallabhbhai National Institute of Technology at Surat, Surat, India, from 2006 to 2008. He is currently a Senior R&D Engineer with Synopsys Inc., Hillsboro, OR, USA. His current research interests include CAD for multicore 2-D and 3-D systems, networks-on-chip, power-, thermal-, variation-, and reliabilty-aware design methodologies, and VLSI automation.

Sudeep Pasricha (M'02–SM'13) received the B.E. degree in electronics and communication engineering from the Delhi Institute of Technology, Delhi, India, in 2000, and the M.S. and Ph.D. degrees in computer science from the University of California at Irvine, Irvine, CA, USA, in 2005 and 2008, respectively.

He is currently an Associate Professor and a Monfort Professor of Electrical and Computer Engineering with Colorado State University, Fort Collins, CO, USA. His current research interests include the areas of energy efficiency, security, and fault tolerant design for embedded systems and high-performance computing.

Dr. Pasricha is a Senior Member of ACM. He is or has been an Organizing Committee Member and/or Technical Program Committee Member of various IEEE/ACM conferences, such as DAC, DATE, CODES+ISSS, NOCS, ISQED, VLSI Design, and GLSVLSI. He was a recipient of the 2015 IEEE/TCSC Mid-Career Researcher Award for Excellence, the 2014 George T. Abell Outstanding Mid-Career Faculty Award, the 2013 AFOSR Young Investigator Award, the 2012 ACM SIGDA Technical Leadership Award, and the best paper awards at the ACM SLIP 2016, ACM GLSVLSI 2015, the IEEE AICCSA 2011, the IEEE ISQED 2010, and ACM/IEEE ASPDAC 2006 conferences.