# VARSHA: Variation and Reliability-Aware Application Scheduling with Adaptive Parallelism in the Dark-Silicon Era

Nishit Kapadia, Sudeep Pasricha
Department of Electrical and Computer Engineering
Colorado State University, Fort Collins, CO, U.S.A.
nkapadia@colostate.edu, sudeep@colostate.edu

***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. In this work, we propose a novel framework that leverages the knowledge of variations on the chip to perform run-time 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 35%-80% in application service-times and 13%-15% in energy consumption, compared to the state-of-the-art.*

## 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- 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 run-time 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-40nm nodes [14], i.e., circuit-level hardening may not sufficiently suppress soft-errors. Thus, system-level run-time 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 (C2C) 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, and it is 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 run-time from the chip-frequency-profile obtained using ring-oscillator based delay-sensors [20], [21]. Thus, run-time 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, giving rise to the dark-silicon phenomenon – a significant fraction of the chip needs to be shut-down at any given time to satisfy the chip power-budget. With the extent of dark-silicon increasing every technology-generation (30%-50% for 22nm) [9], [10], designs are becoming increasingly power-limited rather than area-limited. Run-time 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 run-time application scheduling framework (VARSHA) that employs dynamically adaptable application degrees of parallelism (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 are summarized below:

- We design a novel run-time application-mapping methodology for the emerging dark-silicon-constrained-design-regime, improving over traditional mapping approaches optimized for the area-constrained-design-regime;
- Our framework simultaneously manages all dynamically arriving applications while adapting application-DoPs to optimally utilize the system-power-slack (difference between DS-Pc and current system power dissipation);
- 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;
- Our combined mapping and DVS approach is also enhanced to take advantage of both D2D and WID variations, performing WID variation-aware mapping on to cores with the optimal power/performance characteristics, and D2D variation-aware chip-wide DVS where faster (leakier) chips would need lower $V_{dd}$ levels and slower chips may run at higher $V_{dd}$ levels.

## II. RELATED WORK

A few recent works have explored dark-silicon aware CMP design methodologies. Allred et al. [9] and Turakhia et al. [17] propose design-time frameworks for 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 impact of process variations on CMP performance and power profiles. Raghunathan et al. [10] exploit the variation-profile for run-time application-mapping on to 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 inter-core communication and application reliability requirements. Chou et al. [23] and Fattah et al. [24] consider run-time 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 inter-core communication distances. However, these approaches do not consider reliability and system power, and are geared towards traditional area-constrained designs.

Some recent works advocate varying the degree of parallelism (DoP) of multithreaded applications at run-time, to adapt to changes in the execution environment (power/performance-profiles, core-availability etc.) of a CMP while optimizing metrics such as power and energy-delay product (EDP). Given enough parallelism within the application, [7] showed that increasing DoP is a more energy-efficient way of boosting performance, compared to hiking the core-frequency/voltage values. Variable DoPs can be implemented by saving multiple versions of application-code at compile-time, and choosing the DoP at run-time that is most appropriate for the execution environment; or via more sophisticated techniques [18].

This research is supported by grants from SRC, NSF (CCF-1252500, CCF-1302693), and AFOSR (FA9550-13-1-0110).

The variable-DoP run-time 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 VARSHA framework is different from these efforts in that we assume such information to be pre-profiled at design-time and the focus is on run-time management of application-DoPs in a multi-application power-constrained system. Besides, unlike VARSHA, the frameworks in [15], [16] consider neither the effects of process variations nor the design implications of system-reliability.

## III. Motivational Example

In this section, we illustrate the advantages of some of the key aspects of our VARSHA framework with the help of a small example. We consider a scenario in which applications arrive at run-time 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=0s, and a third application App-3 arriving at t=1s.

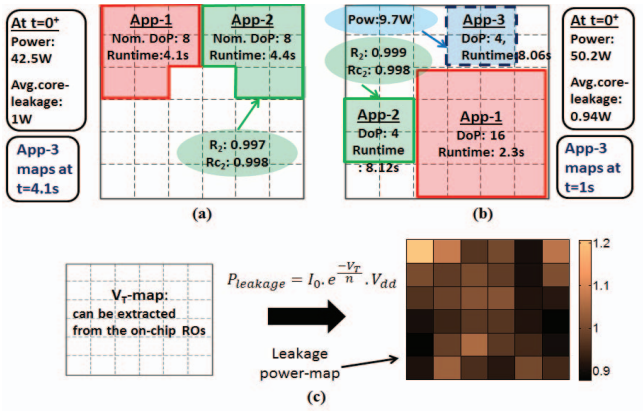


**Figure 1: Motivational example using a 6×6 CMP where App-1 and App-2 are simultaneously mapped at time t=0 and App-3 arrives at t=1s. A DS-Pc = 60W is assumed: (a) Application-scheduling representative of prior works [23], [24]; (b) Application-scheduling obtained by our proposed VARSHA framework; (c) Extracted $V_T$-map and corresponding estimated leakage-power profile (in Watts) of the chip.**

Prior works such as [23], [24] search for square or near-convex regions on the CMP die to map arriving application at run-time, so that a maximum number of applications can fit on the die-area, while also minimizing inter-core communication distances. In this work, we note that in the modern dark-silicon era, performance is generally not limited by the chip-area but by the dark-silicon power-constraint (DS-Pc) of the chip. Therefore, while frameworks proposed in prior works would map App-1 and App-2 at t=0 as shown in figure 1(a) with a nominal DoP of 8, our framework, which adapts application-DoPs (app-DoPs) to extract maximum system-performance (i.e., minimum application service-times) for a given DS-Pc, increases the DoP of App-1 to 16 (as shown in figure 1(b)). When App-3 arrives at t=1s, our framework maps it with a reduced DoP of 4 with a zero-wait time while meeting the DS-Pc of 60W (figure 1(b)), whereas [23] and [24] map App-3 at its nominal DoP of 8 (not shown), and require waiting until App-1 finishes at t=4.1s, so that the DS-Pc is not violated. Thus in our framework, app-DoPs are hiked from their nominal values opportunistically, to minimize runtimes, and app-DoPs are reduced to cut-down on wait times thereby minimizing average application service times (service time=runtime + wait time).

We define reliability of the $i^{th}$ application as $R_i$ = {1-Probability of one or more soft-errors during the execution of App-i}. In this work, we assume applications with different minimum reliability constraints running simultaneously on a CMP. $Rc_i$ represents the reliability constraint of the $i^{th}$ application. The application-reliability ($R_i$) primarily depends on $V_{dd}$ and frequency of the cores (as shown in equations (1)-(3) in section *IV*). Additionally, $R_i$ depends upon 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 framework exhibits reliability-awareness by reducing the DoP of App-2 (with a stringent reliability-constraint) from the nominal value of 8 to 4, thereby meeting $Rc_2$ (figure 1(b)). However, $Rc_2$ is violated when the reliability-unaware frameworks [23], [24] are employed (figure 1(a)).

Furthermore, due to the variation-awareness of our framework, applications are mapped to regions with minimum core-leakage powers (see figure 1(c)). Therefore, the average leakage power per core is 1W for the variation-unaware frameworks (figure 1(a)), and 0.94W for our framework (figure 1(b)). Even though the initial allocation's power dissipation is higher with our framework (at t=0, in figure 1), our lower service time and variation-aware mapping results in less overall energy consumption for the applications - 184J compared to 213J for [23], [24]. Our framework also employs 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 simple example for brevity.

## IV. 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 work. 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 error rate, raw-SER (λ), in a hardware component (core or router) on voltage and frequency values, we use the relationship proposed in [4]:

$$\lambda(\omega_j) = \lambda_0 . 10^{\frac{d.(1-\omega_j)}{1-\omega_{min}}} \quad \text{..... (1)}$$

where $\lambda_0$ is the SER corresponding to the highest voltage and frequency values ($\omega_{max}$), and $\omega_j$ is the average of the normalized values of the $j^{th}$ combination of voltage and frequency (such that $\omega_{max}$=1). For any compute-core, we use a value of $10^{-6}$ errors/sec for $\lambda_0$ and assume d=3 as in [11], [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 a router is given by:

$$\mathbb{R}(\omega_j) = e^{-\lambda(\omega_j).\tau} \quad \text{..... (2)}$$

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

$$R_i = \prod_{2n} \mathbb{R}(\omega_j) \quad \text{..... (3)}$$

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

### B. Inputs, Assumptions, and Problem Objective

We assume the following inputs to our problem:

- A CMP with a regular mesh-based 2D network-on-chip (NoC), with $T$ tiles: $T = (d^2)$, where $d$ is the mesh dimension, and each tile consists of a compute core and a NoC router;
- A set $S$ of candidate supply voltage ($V_{dd}$) levels for the chip;
- A set of $N$ $V_T$-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 figure 1).

- Application sequence $s$ of length $\ell$, made up of η different applications, with arbitrary application inter-arrival times;
- 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 DoP values are ignored;
- Vertices of each task-graph with execution-times of compute cores and edges with inter-task communication volumes; execution time and volume values are assumed available from offline profiling;
- Energy-optimal frequency constraint $\{f_1, f_2, \ldots, f_\eta\}$ and minimum reliability-constraints $\{Rc_1, Rc_2, \ldots, Rc_\eta\}$ for all η applications;
- A chip-wide dark-silicon power dissipation constraint (DS-Pc).

We make the following assumptions in our work:

- There exists one-to-one mapping between tasks and cores;
- Applications are mapped contiguously on non-overlapping rectangular regions of the CMP die, for inter-application isolation;
- All cores executing an application run at the same frequency, to avoid imbalances during multi-threaded execution [10];
- Variation-map data for a chip is available at run-time, in terms of the threshold-voltage ($V_T$) distribution, from the chip-frequency-profile obtained using ring-oscillator based delay sensors [20];
- A chip wide supply voltage that can be scaled using DVS at run-time, as the overheads of implementing DVS at a per–core granularity are deemed to be prohibitively expensive [22].

Problem Objective: Given the above inputs and assumptions, our objective is to perform run-time 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.

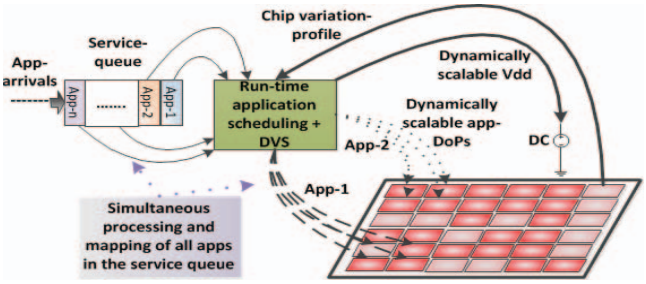


**Figure 2: Overview of our application-scheduling + DVS framework**

## V. VARSHA FRAMEWORK: OVERVIEW

Figure 2 illustrates the key aspects of our proposed 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 *(i)* determining the DoP (out of the $|P_i|$ DoPs) for each waiting application in the service-queue, and *(ii)* mapping the appropriate task-graphs on to the tiles of the CMP. For a given $V_{dd}$, scaling-up of application-DoPs (*app-DoPs*) is constrained by the available power-slack (difference between DS-Pc and current system-power), application-reliability constraints, and the available tiles meeting 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.

The VARSHA framework is effectively executed in two nested procedures: *(i)* $V_{dd}$-level selection (outer loop), triggered on an arrival or a departure of any application; and *(ii)* 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 figure 3(a) and figure 3(b), respectively.

### A. $V_{dd}$-level Selection

To extract maximum performance from the applications being considered for mapping at any time instant, the first-order objective in VARSHA is to maximize overall DoP of the system (sum-total of all *app-DoPs*). Recall that an application typically has a maximum viable DoP, and higher DoPs can cause performance to degrade (due to high synchronization overheads) – such higher DoP configurations are ignored (section *IV.B*). Our $V_{dd}$-selection heuristic (figure 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, our framework 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. 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 (in increments of 0.1V) until either the overall DoP reduces, 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 hiked to the selected $V_{dd}$-level, and the mapped applications are then removed from the service-queue.

### 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. Figure 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 power overheads for compute-intensive applications at higher DoPs. Also, we hike *app-DoPs* symmetrically across all applications because execution of an application is generally more energy-efficient at lower DoPs (due to lower parallelization- and communication-overheads). For instance running four applications simultaneously, each with DoP=4, is typically more energy-efficient than running them one after the other with DoP=16 each.

To produce an optimal mapping for the application under consideration (with a specific DoP), the following three steps are performed: *(i)* rectangular-region selection on the CMP for the current DoP (section *V.B.1*); *(ii)* mapping of the corresponding task-graph to the selected region (section *V.B.2*); *(iii)* communication-flow routing and delay/power analysis (section *V.B.3*). After the above steps, the mapping is evaluated for overall power footprint and reliability of the applications. Satisfaction of the application-frequency constraints are checked during the rectangular-region-

selection step. As shown in figure 3(b), DoP-hike of any application could fail due to potential violation(s) in application-frequency and –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.

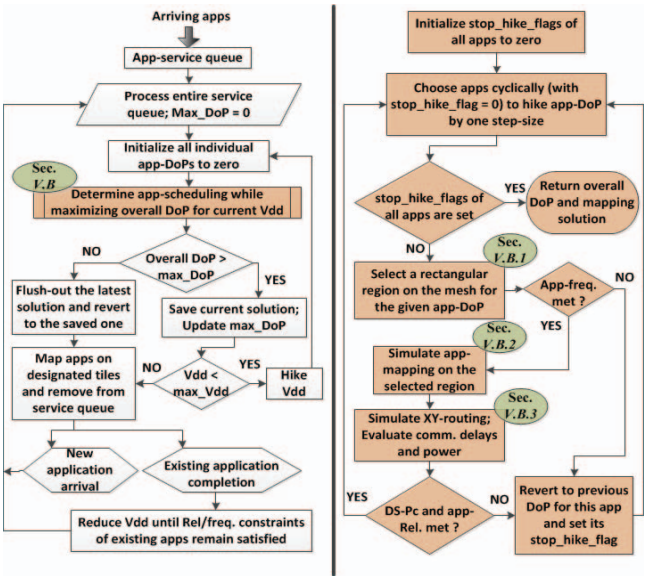


**Figure 3: Design-flows for the VARSHA framework: (a) $V_{dd}$-level selection (discussed in section *V.A*); (b) Determination of application-schedule for the current $V_{dd}$-level (discussed in section *V.B*).**

### *B.1 Rectangular Region Selection*

We consider application-mapping on rectangular regions, of pre-determined dimensions corresponding to each possible DoP value in set *P*. Therefore, all intra-application communication is contained within a rectangular region. This provides inter-application isolation, and eliminates communication cross-interference overhead. Given the $V_T$-map, the maximum frequency that each core can be reliably clocked at depends upon the $V_T$ and $V_{dd}$ values, given by:

$$f_{max} = \frac{\mu(V_{dd}-V_T)^{\alpha}}{c_0.V_{dd}} \quad \ldots\ldots (4)$$

where α and μ are technology-dependent constants, and $C_0$ is switching capacitance of the critical path [30]. In our region selection method, 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 region on the mesh (of pre-defined dimensions) that dissipates the least total leakage-power and all cores within which satisfy the frequency-constraint of the application being mapped. To this end, we perform a simple exhaustive search over all tiles on the mesh as the left-upper corner of the given rectangular region. If the rectangle is not a square, then both of its orientations need to be checked to find the optimal rectangular region. Figure 4 shows an illustration of our region-selection method for a 4×4 mesh.

<u>*Theoretical time-complexity analysis*</u>: At most *T* tiles (total tiles on the chip) are considered for the prospective rectangular region. For non-squares, e.g., a 2×4 rectangle, rectangles of both orientations need to be evaluated at each tile. Note that *app-DoP* (relatively small integer *c* – treated as constant) number of tiles are to be evaluated for frequency and leakage-power at each of these iterations. This gives a linear-time complexity for the region-selection step: O(2*cT*).

### *B.2 Application Mapping*

After the rectangular region (of size equal to app-DoP) on the mesh has been selected, our mapping heuristic maps the appropriate application-task-graph on to the CMP tiles. We employ an incremental-mapping approach that is carried out in two steps:

1) Starting with the task with highest communication-volume, list tasks in decreasing order of inter-task communication volumes with all preceding tasks in the list;
2) Map tasks on the selected region, in the order of this list – starting with one of the center tiles on the rectangular-region. To map any task, select the tile that would incur least traffic foot-print with the already mapped tasks. We define communication-traffic foot-print of a core (mapped task) as: $\sum MD_j \times vol_j$, where $vol_j$ is the communication-volume of the $j^{th}$ flow entering or leaving the core and $MD_j$ is the Manhattan distance between the source and destination of the $j^{th}$ flow.

<u>*Theoretical time-complexity analysis*</u>: Step (1) sorts *app-DoP* tasks – the time-complexity could be bounded by $O((app\text{-}DoP)^2)$. In step (2), *app-DoP* number of tasks are mapped and each could consider a maximum of *app-DoP* number of tile-locations. Thus this step could be loosely bounded by $O((app\text{-}DoP)^2)$. As *app-DoP* is a relatively small integer, it can be considered a constant (between 4 and 16 in our experiments). Thus, our application-mapping has constant time-complexity. Note that mapping-information can potentially even be saved with the task-graph information at compile-time, thus saving valuable run-time resources and overhead.

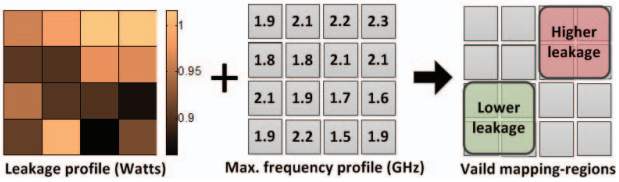


**Figure 4: Two feasible regions are shown for the application frequency constraint of 1.9 GHz. Our method chooses the one with lower leakage.**

### *B.3 Communication Delay and Power Estimation*

Similar to numerous prior works such as [24], we use the XY-routing scheme to route the communication-flows of applications. The communication-delays with congestion-overheads are calculated from the application-frequency (routers and links run at the rated application-frequency) and link BWs. Profiling of compute- and communication-delays could potentially be performed at design-time [33]-[35]. Dynamic powers of routers and links (corresponding to different voltages, frequencies, communication-loads, and router-sizes), are assumed to be saved in the read-only (or non-volatile) memory on the CMP die. 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 equations (1)-(3). For our analyses, the energies and run-times 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 VARSHA framework, the number of times that our region-selection heuristic would need to be invoked is bounded by a relatively small constant integer. As the region-selection step, which has the highest theoretical time-complexity in the VARSHA design-flow, finishes in linear-time (as discussed in section *V.B.1*), the time-complexity of our framework is linear, with respect to the CMP mesh-size, *T*.

## VI. EXPERIMENTS

Our experiments were conducted using η=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 trade-off 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 for all applications at different DoPs are modeled based on the system-traces generated by running topology-agnostic gem5 simulations [28]. The reliability constraints of different applications are set in the range: 0.99 to 0.999. We assume the ARM Cortex-A9 processors [27] as the baseline CMP compute cores, which support five operating voltage levels ($|S|$=5): 0.8V, 0.9V, 1.0V, 1.1V, and 1.2V. The application-specific energy-optimal core frequencies range from 1300 MHz to 1900 MHz, based on the level of compute intensity of the tasks assigned to cores. We use a 100-core mesh for the CMP platform with dimensions 10×10. The dark-silicon power-constraint (DS-Pc) is set at 100W. The delay-overhead for $V_{dd}$-scaling (PLL lock time) is estimated to be less than 5µs, similar to [12], which is negligible compared to the granularity of application-runtimes (2-9 seconds each) in our experiments.

To investigate the applicability of our approach to CMP dies with diverse variation-profiles, we use 1000 test-chips ($N$=1000), in our experiments. The 1000 $V_T$-maps 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 ($R$) 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 30×30 grid of points corresponding to 9 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). 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. Results

We compare the results obtained from our VARSHA framework with those obtained from using run-time application mapping frameworks proposed in recent prior works [10] and [24]. 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 inter-arrival times of application-instances randomly within the following ranges: 0 to 1 seconds (Seq-1A and Seq-1B), 0 to 2 seconds (Seq-2A and Seq-2B), 0 to 4 seconds (Seq-3A and Seq-3B), and 0 to 8 seconds (Seq-4A and Seq-4B). We assume $\ell$=100 application-instances in any application-sequence. Also, our results (as shown in figure 5 and Table 1) show the mean-values across a 1000 test-chips.

The prior works [10] and [24] assume fixed nominal app-DoPs. Our framework adapts app-DoPs in accordance with the application inter-arrival rates to minimize the application service-times. Observe in Table 1 that for both sequences, the average app-DoP reduces with increasing inter-arrival-rates for VARSHA. At higher inter-arrival rates when applications with nominal DoPs cannot be quickly serviced due to the DS-Pc constraint, our VARSHA framework cuts down application wait-times significantly by reducing DoPs (as shown in figure 5(a) - Seq-1A,B and Seq-2A,B), although the application run-times tend to increase due to the reduction in DoPs. On the other hand, at lower inter-arrival rates, with on average fewer applications to be serviced simultaneously, our framework opportunistically hikes the application-DoPs to minimize run-times (as shown in figure 5(a) - Seq-3A,B and Seq-4A,B). In comparison with [24], we obtain 27%-43% savings in average service-times with our VARSHA framework. Note that maximum savings are obtained when the inter-arrival-rates are most stringent, as shown for Seq-1A and Seq-1B in figure 5(a). The communication-unaware framework in [10] maps applications on to large rectangular regions of non-contiguous tiles, resulting in longer run-times due to longer communication-latencies. Compared to [10], we obtain 70% - 87% savings in average service-times with our VARSHA framework.

**Table 1: Mean values across all 1000 test-chips for average DoP per application-instance and weighted-average $V_{dd}$ for VARSHA framework**

| | Seq-1A | Seq-2A | Seq-3A | Seq-4A | Seq-1B | Seq-2B | Seq-3B | Seq-4B |
|---|---|---|---|---|---|---|---|---|
| DoP | 4.5 | 10.3 | 14.1 | 14.5 | 5.1 | 10.7 | 13.7 | 14.2 |
| $V_{w\text{-}avg}$ | 1.09 | 1.10 | 1.07 | 1.07 | 1.07 | 1.11 | 1.09 | 1.07 |

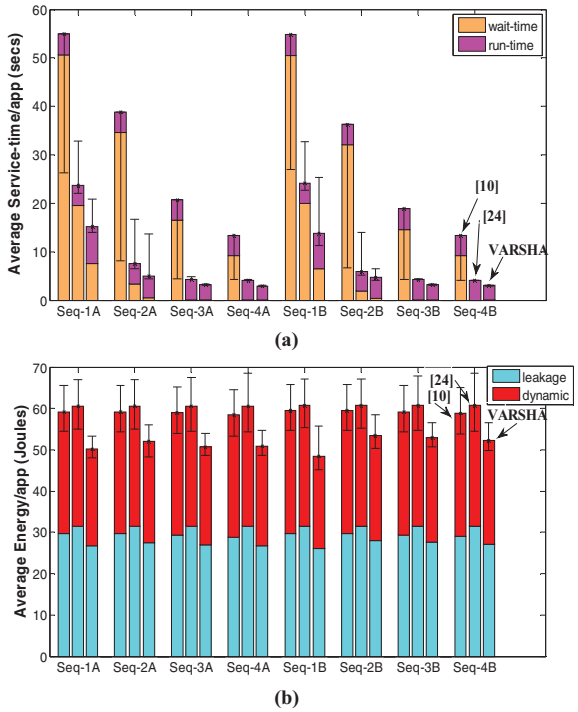


**Figure 5: Results of VARSHA framework vs. [10] and [24]: (a) Average service-time per application-instance (wait-time + run-time); (b) Average energy per application-instance (leakage + dynamic). The bars represent mean values of service-times and energies across 1000 test-chips, while variation in service-times and energies is shown by confidence intervals.**

Our framework also opportunistically lowers $V_{dd}$-levels while ensuring that frequency and reliability constraints remain satisfied for the set of existing applications. The weighted-average $V_{dd}$ metric ($V_{w\text{-}avg}$) represents the average value of $V_{dd}$ throughout the execution of the entire application-sequence for a given chip, and is defined as:

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

where, $\Delta ti$ is the $i^{th}$ 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 1 shows the mean $V_{w\text{-}avg}$ across all 1000 test-chips. In the absence of DVS in [10] and [24], in our implementation of these frameworks, we assume the highest voltage of 1.2V, which is just high enough to meet all application-frequency constraints for the 1000 test-chips. Despite the high $V_{dd}$ and much longer communication paths, [10] saves leakage-power by performing variation-aware mapping. In the absence of such variation-awareness, [24] consumes the highest energy (as shown in figure 5(b)). In our VARSHA framework, energy-efficiency decreases with increasing app-DoPs (due to increased communication and sub-linear increase in computation-performance), while decreasing $V_{dd}$-levels generally cause an increase in energy-efficiency. Therefore, in figure 5(b), we observe variation in energy values with different inter-arrival rates. The maximum energy savings are obtained for the most stringent inter-arrival rates, as shown for Seq-1A and Seq-1B in figure 5(b), due to opportunistic reduction of DoPs. We find from figures 5(a), (b) that VARSHA produces on average, **13%** and **15%** savings in mean energy (8% and 14% savings in mean leakage energy) and, **80%** and **35%** savings in mean application service time (93% and 66% savings in mean application wait-times), over [10] and [24] respectively.

Our experiments indicate that for the frameworks from [10] and [24], where a fixed high value of $V_{dd}$ is used, slower chips (usually with lesser leakage) could prove to be advantageous as they would be left with greater power-slack to accommodate more applications at any given time (less %dark-silicon), thus resulting in better service-times and energies. Conversely, in our VARSHA framework, 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. We observed that test-chips with moderate leakage and performance profiles produce desirable service-times and energy results, whereas chips that are too leaky or too slow yield worse results, as indicated by the upper-bounds of the confidence-intervals in figure 5 (a) and 5(b) for the VARSHA framework.

**Table 2: Reliability-constraint (Rc) violations (average per sequence)**

| | Seq-A | Seq-B |
|---|---|---|
| [10] | 8 | 9.5 |
| [24] | 6 | 8 |

For applications with relatively more stringent reliability-constraints, it may not be possible to support nominal-DoP even at the highest $V_{dd}$-level (1.2V). Therefore, when using reliability-unaware frameworks with no DoP-adaptivity, reliability-constraints (Rc) of such applications may be violated. Table 2 shows the Rc-violations across 1000 test-chips for schedules produced by [10] and [24]. With longer routing distances, [10] consumes far more routing resources compared to [24], thus even with the same app-DoPs (nominal DoPs), the estimated failure-rates are greater for [10]. Our reliability-aware VARSHA framework, results in no Rc-violations because of its ability to dynamically reduce app-DoPs as well as hike $V_{dd}$-levels in accordance with application reliability-requirements.

## VII. Conclusion

VARSHA represents one of the first efforts to integrate reliability and variation-awareness in a run-time variable degree-of-parallelism (DoP) application-scheduling methodology to enhance performance of multi-core systems in the new dark-silicon era. Tailored for deeply scaled technologies, our proposed VARSHA framework generates highly optimized application-mapping solutions, while exhibiting linear run-time complexity. Our experimental results show that VARSHA produces savings of **35%-80%** in application service-times, **13%-15%** in energy, and avoids reliability violations unlike state of the art prior works that suffer from reliability violations in up to **11%** of application instances arriving at run-time.

## References


[1] A. Kaouache et al., "Analytical method to evaluate soft error rate due to alpha contamination," IEEE Trans. Nucl. Sci., 60(6), 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," IEEE IRPS, 2013.

[3] N. Kapadia, S. Pasricha, "Process Variation Aware Synthesis of Application-Specific MPSoCs to Maximize Yield", IEEE VLSID, 2014.

[4] D. Zhu, R. Melhem, D. Mosse, "The effects of energy management on reliability in real-time embedded systems," Proc. ICCAD, Nov. 2004.

[5] E. Humenay, D. Tarjan, K. Skadron, "Impact of process variations on multicore performance symmetry," Proc. DATE, pp. 1653-1658, 2007.

[6] L. Pang et al. "Measurement and Analysis of Variability in 45 nm Strained-Si CMOS Technology," IEEE JSSC, Aug. 2009.

[7] J. Lee, N. Sung Kim, "Optimizing total power of many-core processors considering voltage scaling limit and process variations," ISLPED, '09.

[8] S. Borkar "Design perspectives on 22nm CMOS and beyond" DAC '09.

[9] J. Allred, S. Roy, K. Chakraborty, "Designing for dark silicon: a methodical perspective on energy efficient systems," ISLPED, 2012.

[10] B. Raghunathan et al., "Cherry-picking: Exploiting process variations in dark-silicon homogeneous chip multi-processors," Proc. DATE, 2013.

[11] A. Das et al., "Combined DVFS and mapping exploration for lifetime and soft-error susceptibility improvement in MPSoCs," DATE, 2014.

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

[13] M. Haque, et al., "Energy-aware task replication to manage realiability for periodic real-time applications on multicore platforms," IGCC, 2013.

[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. on Nucl. Sci., 58(3), pp. 1008-1014, June 2011.

[15] J. Li, J. F. Martinez, "Dynamic power-performance adaptation of parallel computation on chip multiprocessors," HPCA, pp: 77-87, 2006

[16] Y. Ding et al., "A helper thread based EDP reduction scheme for adapting application execution in CMPs," IEEE IPDPS Apr. 2008.

[17] Y. Turakhia et al., "HADES: Architectural synthesis for heterogeneous dark silicon chip multi-processors," Proc. DAC, June 2013.

[18] A. Raman et al., "Parallelism orchestration using DoPE: the degree of parallelism executive," PLDI, pp: 26-37, June 2011.

[19] L. Zhang et al., "Process variation characterization of chip-level multiprocessors," Proc. DAC, July 2009.

[20] X. Wang et al., "Design and analysis of a delay sensor applicable to process/environmental variations and aging measurements," IEEE TVLSI, 20(8), pp: 1405-1418, Aug. 2012.

[21] A.A.M. Bsoul, N. Manjikian, L. Shang, "Reliability- and process variation-aware placement for FPGAs," DATE, pp: 1809-1814, 2010.

[22] S. Dighe et al., "Within-die variation-aware dynamic-voltage-frequency-scaling with optimal core allocation and thread hopping for the 80-core TeraFLOPS processor," IEEE JSSC, 46(1), pp: 184-193, Jan. 2011.

[23] C. Chou, U. Ogras, R. Marculescu, "Energy- and performance-aware incremental mapping for networks on chip with multiple voltage levels," IEEE TCAD, 27(10), pp: 1866-1879, Oct. 2008.

[24] M. Fattah et al., "Smart hill climbing for agile dynamic mapping in many-core systems," Proc. DAC, June 2013.

[25] S.V. Woo et al., "The SPLASH-2 programs: characterization and methodological characterization," Proc. ISCA, pp. 24-36, May 1995.

[26] C. Bienia, S. Kumar, J.P. Singh, K. Li, "The PARSEC benchmark suite: characterization and architectural implications," Proc. PACT, 2008.

[27] ARM, http://www.arm.com/products/processors/selector.php

[28] N. Binkert et al., "The gem5 simulator," SIGARCH, 39(2), May 2011.

[29] VARIUS model, http://www.cse.ohiostate.edu/~teodores/arch/tools/

[30] S. Sarangi et al., "VARIUS: A Model of Process Variation and Resulting Timing Errors for Microarchitects," IEEE TSM, (21)1, 2008.

[31] B. Li, L. Peh, P. Patra, "Impact of process and temperature variation on network-on-chip design exploration," NOCS, Apr. 2008.

[32] A. Kahng, et al., "ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration," DATE, 2009.

[33] N. Kapadia, S. Pasricha, "VISION: a framework for voltage island aware synthesis of interconnection networks-on-chip", GLSVLSI, 2011.

[34] N. Kapadia, S. Pasricha, "A framework for low power synthesis of interconnection networks-on-chip with multiple voltage islands", Integration, the VLSI Journal, 45(3):271-281, Jun 2012.

[35] N. Kapadia, S. Pasricha, "VERVE: a framework for variation-aware energy efficient synthesis of NoC-based MPSoCs with voltage islands", IEEE ISQED, 2013.