# Dark vs. Dim Silicon and Near-Threshold Computing Extended Results

**University of Virginia Dept. of Computer Science Technical Report CS-2013-01**

Liang Wang

Department of Computer Science

University of Virginia

lw2aw@virginia.edu

Kevin Skadron

Department of Computer Science

University of Virginia

skadron@cs.virginia.edu

## Abstract

*Due to limited scaling of supply voltage, power density is expected to grow in future technology nodes. This increasing power density potentially limits the number of transistors switching at full speed in the future. Near-threshold operation can increase the number of simultaneously active cores, at the expense of much lower operating frequency ("dim silicon"). Although promising to increase overall throughput, dim cores suffer from diminishing returns as the number of cores increases. At this point, hardware accelerators become more efficient alternatives. To explore such a broad design space, we have developed a framework called* Lumos *to analytically quantify the performance limits of many-core, heterogeneous systems operating at near-threshold voltage. Lumos augments Amdahl's Law with detailed scaling of frequency and power, calibrated by circuit-level simulations using a modified Predictive Technology Model (PTM) and factors in effects of process variations. While our results show that dim cores do indeed boost throughput, even in the presence of process variations, significant benefits are only achieved in applications with very high parallelism or with novel architectures to mitigate variation. A more beneficial*

*and scalable approach is to use accelerators. However, reconfigurable logic that supports a variety of accelerators is more beneficial than a dedicated, fixed-logic accelerator, unless 1) the dedicated kernel has overwhelming coverage across applications (e.g. twice as large as the total of all others), or 2) the speedup of the dedicated accelerator over the reconfigurable equivalent is significant (e.g. 10x-50x).*

## 1. Introduction

Threshold voltage scales down more slowly in current and future technology nodes to keep leakage power under control. In order to achieve fast switching speed, it is generally necessary to keep transistors operating at a considerably higher voltage than their threshold voltage. Therefore, the dwindling scaling on threshold voltage leads to a slower pace of supply voltage scaling. Since the switching power scales as $CV^2f$, Dennard Scaling can no longer be maintained and power density keeps increasing. Furthermore, cooling cost and on-chip power delivery implications limit the increase of a chip's thermal design power (TDP). As Moore's Law continues to double transistor density across technology nodes, total power consumption will soon exceed TDP, and if high supply voltage must be maintained, future chips will only support a small fraction of active transistors, leaving others inactive, a phenomenon referred to as "dark silicon" [10, 27].

Since dark silicon poses a serious challenge for conventional multi-core scaling [10], researchers have tried different approaches to cope with dark silicon, such as "dim silicon" [15] and customized accelerators [24]. Dim silicon, unlike conventional designs working at nominal supply, aggressively lowers supply voltage close to the threshold to reduce dynamic power. The saved power can be used to activate more cores to exploit more parallelism, trading off per-core performance loss with better overall throughput [11]. However, with near-threshold supply, dim silicon designs suffers from diminishing throughput returns as the core number increases. On the other hand, customized accelerators are attracting more attention due to their orders of magnitude higher power efficiency than general-purpose processors [5, 13]. Although accelerators are promising in improving performance with less power consumption, they are built for specific applications, have limited utilization on general-purpose applications, and sacrifice die

area that could be used for more general-purpose cores. Poorly utilized die area would be very costly if there are more efficient solutions, in terms of opportunity cost. Therefor, the utilization of each incremental die area must be justified with a concomitant increase in average selling price (ASP).

To investigate the performance potential of dim cores and accelerators, we have developed a framework called *Lumos*, which extends the well known Amdahl's Law, and is accompanied by a statistical workload model. We investigate two types of accelerators, application-specific logic (ASIC) and RL. When we refer to RL, we generically model both fine-grained reconfigurability such as FPGA and coarse-grained such as Dyser [12]. We also assume the reconfiguration overhead is overwhelmed by sufficient utilization of each single configuration. Our main conclusions are:

- Systems with dim cores achieve up to 2x throughput over conventional CMPs, but with diminishing returns.
- Hardware accelerators, especially RL, are effective to further boost the performance of dim cores.
- Reconfigurable logic is preferable over ASICs on general-purpose applications, where kernel commonality is limited.
- A dedicated fixed logic ASIC accelerator is beneficial either when: 1) its targeted kernel has a significant coverage (e.g. twice as large as the total of all other kernels), or 2) its speedup over RL is significant (e.g. 10x-50x).

## 2. Lumos

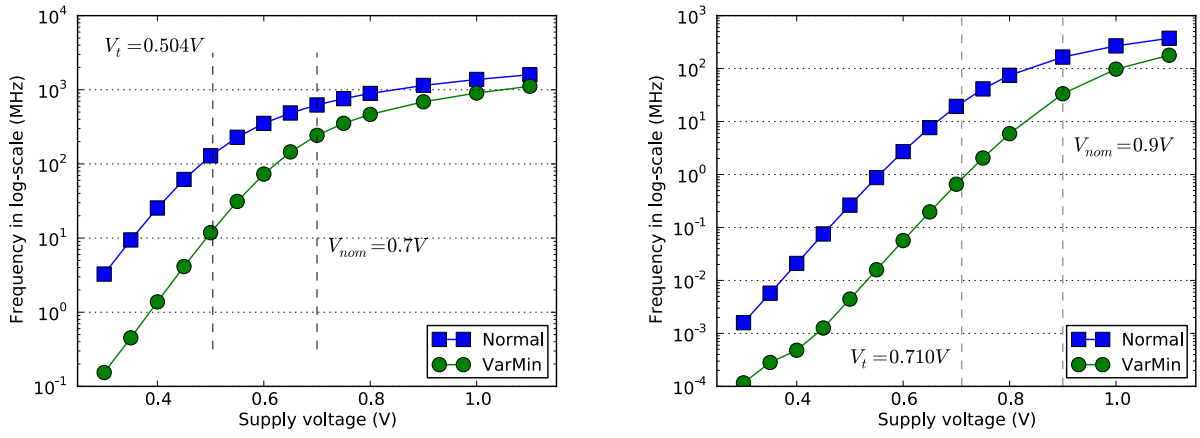
We use aggregate throughput under physical constraints as the primary performance metric. Instead of running extensive architectural simulation, we develop Lumos by extending the simple Amdahl's Law model by Hill and Marty [14] with physical scaling parameters calibrated by circuit simulations with a modified Predictive Technology Model [4]. Systems are studied across technology nodes ranging from 45nm through 16nm. A Niagara2-like in-order core is chosen as a single-core baseline design. The characteristics of this baseline core at 45nm are obtained from McPAT [19] and summarized in Table 1.

| Frequency ($GHz$) | Dynamic Power ($W$) | Leakage Power ($W$) | Area ($mm^2$) |
|---|---|---|---|
| 4.20 | 6.14 | 1.06 | 7.65 |

Table 1. Characteristics of a single Niagara-like in-order core at 45nm, obtained from McPAT [19].

Lumos extends Amdahl's Law with extensions of *voltage-frequency scaling* and *power modeling*, calibrated by circuit cimulations We use a 32-bits ripple carry adder to emulate the core frequency changes subject to various supply voltages. The adder is synthesized with a standard-cell library [16] at 45nm. Sizes of transistors in post-synthesized netlist are scaled to other technology nodes. These circuit netlists are then simulated using a modified version of Predictive Technology Model (PTM), which tunes transistor models with more up-to-date parameters extracted from recent publications [4]. Libraries in each technology node are available in two variants, a high-performance process and a low-power process. The primary difference is that low-power processes have a higher threshold, as well as a higher nominal supply voltage, than high-performance processes for every single technology node.

### 2.1. Frequency Modeling and Variation




(a) High-performance with $V_t = 0.504V$.

(b) Low-power with $V_t = 0.710V$.

Figure 1. A 32bits ripple carry adder (RCA) frequency scaling at 16nm, subjecting to process variation (VarMin). $V_t$ is the threshold voltage, and $V_{nom}$ is the nominal supply voltage.

Voltage-to-frequency scaling is modeled by interpolating empirical results from circuit simulations.

The switching speed of a transistor scales exponentially to the threshold voltage when it is operating at near-threshold voltage; therefore the working frequency of a circuit in near-threshold region is extremely sensitive to the threshold voltage. To investigate the fluctuation of frequency subject to process variations, the test circuit has been simulated with Monte-Carlo analysis, assuming a standard Gaussian distribution with 3-$\sigma$ on the threshold voltage of transistors. The results for 16nm are shown in Figure 1. For the high-performance process plotted in Figure 1a, the maximum performance penalty due to process variations is less than 10% when the circuit is operated at higher voltages around $1V$. However, when supply voltage approaches the threshold, frequency penalty increases significantly to more than 90%, which means an order of magnitude slow down on frequency. A similar trend is observed in Figure 1b for the low-power process, though the low-power process suffers from a larger percentage of frequency loss than the high-performance process even at a higher supply voltage, e.g. 1V.

### 2.2. Power Modeling

Core power consumption is modeled as two major sources, dynamic power due to transistor switching and static power due to leakage. Therefore, per-core power is given by

$$P_{\text{total}} = P_{\text{dynamic}} + P_{\text{leakage}} \tag{1}$$

Generally speaking, dynamic power is given by

$$P_{\text{dynamic}} = \alpha \cdot C_{eff} \cdot V_{dd}^2 \cdot f \tag{2}$$

where $\alpha$ is the activity factor, $C_{eff}$ is the effective capacitance, $V_{dd}$ is supply voltage, and $f$ is the core running frequency. We assume a constant activity factor and effective capacitance when the core frequency is scaled with various supply voltages. Therefore, dynamic power changes quadratically to supply voltage and linearly to frequency. According to [3], the static power of a system is given by

$$P_{\text{leakage}} = V_{dd} \cdot N \cdot k_{design} \cdot \hat{I}_{leak} \tag{3}$$

where $V_{dd}$ is supply voltage, $N$ is the number of transistors, $k_{design}$ is a device-specific constant, and $\hat{I}_{leak}$ is the normalized per-transistor leakage current. We assume the normalized per-transistor leakage current is proportional to the leakage current of a single transistor, which , according to [23], is given by

$$I_{leak} = I_0 \cdot 10^{\frac{V_{gs}-V_t+\eta V_{ds}}{S}} \cdot \left(1 - e^{-\frac{V_{ds}}{V_{th}}}\right) \tag{4}$$

where $V_t$ is the threshold voltage, $\eta$ is the drain-induced barrier lowering factor (DIBL), $V_{th}$ is the thermal voltage, and $n$ is a process-dependent constant. The thermal voltage at room temperature is around $28mV$, which is far less than the supply voltage of interest in this project, therefore $e^{-V_{ds}/V_t} \approx 0$. Because the transistor is at its static state when considering the static leakage power, $V_{gs}$ and $V_{ds}$ is roughly proportional to the supply voltage. As a result, the above equation can be reduced to

$$\hat{I}_{leak} \propto 10^{\frac{V_{dd}}{\hat{S}_{leak}}} \tag{5}$$

where $\hat{S}_{leak}$ is the aggregate scaling coefficient derived from fitting to the simulated results.

### 2.3. Performance Modeling

For the aggregate throughput performance ($pf_{\text{s}}$), we model it with Amdahl's Law as shown in Equation 6

$$\text{Speedup} = \frac{1}{\frac{1-\rho}{S_{\text{serial}}} + \frac{\rho}{n \cdot S_{\text{parallel}}}} \tag{6}$$

where $\rho$ is the parallel ratio of the studied workload, $S_{\text{serial}}$ is the serial part speedup over a baseline core, $S_{\text{parallel}}$ is the per-core speedup when the system works in parallel mode, and $n$ is the number of active cores running in parallel. For $S_{\text{serial}}$, only one core is active. In this case, the core is boosted to 1.3x nominal supply to achieve the best single core performance, denoted as $\hat{pf}_{\text{c}}$. When the system is in parallel mode, both $n$ and $S_{\text{parallel}}$ are determined by supply voltage. First, Lumos picks the optimal frequency with a given supply voltage; Second, it calculates per-core power consumption including both the switching power and the leakage power according to the frequency and the supply. Finally, the

number of active cores is the minimum restricted by the overall power and area budgets, which is given by

$$n(v) = \min(\frac{P}{p(v)}, \frac{A}{a}) \tag{7}$$

where $p(v)$ is the per-core power as a function of supply voltage, $P$ and $A$ are system budgets of power and area, respectively. As a result, Equation 6 can be rewritten as

$$\text{Speedup} = 1 \Big/ \frac{1-\rho}{\frac{\hat{pf}_c}{pf_0}} + \frac{\rho}{n(v) \cdot \frac{pf(v)}{pf_0}} \tag{8}$$

where $pf_0$ is the performance of a single baseline core, $pf(v)$ is the per-core performance as a function of supply voltage,

When considering process variation, a single core performance loss leads to poor throughput of a symmetric many-core system, in which the slowest core dictates the overall performance of the whole system. Adaptive Body Bias (ABB) is not considered here, but is important future work. However, the worst-case variation we used here sets an upper bound on the benefits of ABB. As a result, the parallel performance of a system is confined by the core with the worst performance, denoted as $pf(v)_{min}$. As for the power, we assume no fine-grained per-core level power management mechanisms; therefore, all cores contribute to the total power consumption through the whole period that the system is in parallel mode. We use the mean total power ($\overline{p}$) to estimate the per-core power, so Equation 7 is rewritten to

$$n(v) = \min(\frac{P}{\overline{p}}, \frac{A}{a}) \tag{9}$$

When considering process variations, the overall speedup in Equation 8 is rewritten to

$$\text{Speedup} = 1 \Big/ \frac{1-\rho}{\frac{\hat{pf}_c}{pf_0}} + \frac{\rho}{n(v) \cdot \frac{pf(v)_{min}}{pf_0}} \tag{10}$$

### 2.4. Technology Modeling

Technology scaling is built based on the modified PTM discussed earlier in this section. Two types of technology variants are investigated, a high-performance process and a low-power process. The two processes have different nominal voltage and threshold voltage, listed in Table 2.

| | | $45nm$ | $32nm$ | $22nm$ | $16nm$ |
|---|---|---|---|---|---|
| High | $V_{nom}$ | 1.0 | 0.9 | 0.8 | 0.7 |
| Perf. | $V_t$ | 0.424 | 0.466 | 0.508 | 0.505 |
| Low | $V_{nom}$ | 1.1 | 1.0 | 0.95 | 0.9 |
| Power | $V_t$ | 0.622 | 0.647 | 0.707 | 0.710 |

Table 2. Nominal supply voltage ($V_{nom}$) and threshold voltage ($V_t$) for each PTM technology variants. Voltage units are volt ($V$).

Inter-technology scaling mainly deals with scaling ratios for the frequency, the dynamic power, and the static power. From the circuit simulation, we compare the frequency, the dynamic power, and static power at the nominal supply for each technology nodes. We assume cores will have the same scaling ratio as the circuit we simulated. As shown in Table 3, the scaling factors of high-performance variants are normalized to 45nm, while the scaling factors of low-power variants are normalized to their high-performance counterparts.

| Tech. (nm) | Freq. | Switch Power | Static Power |
|---|---|---|---|
| 45 | 1.0 | 1.0 | 1.0 |
| 32 | 0.95 | 0.49 | 0.31 |
| 22 | 0.79 | 0.21 | 0.12 |
| 16 | 0.66 | 0.09 | 0.13 |

(a) HP, normalized to 45nm

| Tech. (nm) | Freq. | Switch Power | Static Power |
|---|---|---|---|
| 45 | 0.27 | 0.30 | 0.0084 |
| 32 | 0.28 | 0.32 | 0.042 |
| 22 | 0.26 | 0.34 | 0.23 |
| 22 | 0.26 | 0.40 | 0.49 |

(b) LP, normalized to HP counterparts

Table 3. Technology scaling for high-performance (HP) processes and low-power (LP) processes. The area scaling is assumed to be ideal so that the area of a single core shrinks by 2x per technology node.

### 2.5. Workload Modeling

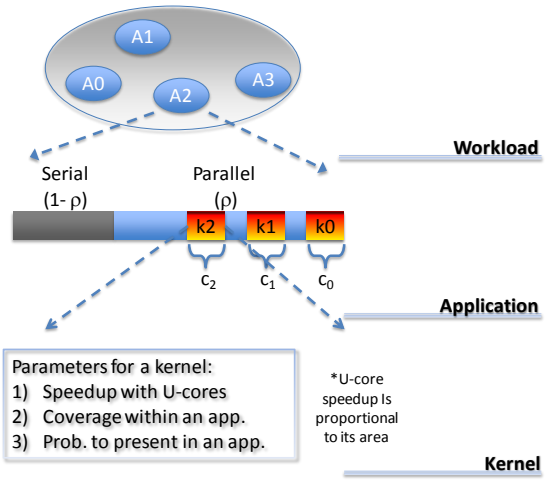



Figure 2. Workload modeling. A workload is composed by applications. An application may have several computing kernels, which can be accelerated by CMP, RL, and ASIC.

The workload model is illustrated in Fig. 2, where a workload is considered as a pool of applications. Each single application is divided into two parts, serial and parallel. Part of an application can be also partitioned into several computing kernels. These kernels can be accelerated by various computing units, such as multicore, possibly dim CPU cores, RL, and customized ASIC. We model the speedup and the power consumption of RL and customized ASIC for a given kernel by u-core parameters ($\eta$,$\phi$). As characterized in [5], a "u-core" is any unconventional core such as RL or customized ASIC block. $\eta$[1] is the relative power efficiency to a single, basic in-order core, and $\phi$ is the relative performance. We assume that u-cores are only used to accelerate kernels that are ideally parallelized. Therefore, we model the relative performance of a u-core proportional to its area. We add two more parameters to model the relationship between applications and kernels:

**Presence** ($\lambda$) A binary value to indicate whether or not a kernel present in an application.

**Coverage** ($\varepsilon$) The time consumption in percentage for a kernel when the whole application is running with a single base line core.

[1]In the original paper of [5], $\mu$ is used to denote relative performance of u-cores. However, $\mu$ is commonly used as the mean in statistics. Therefore, we use $\eta$ as an alternative to avoid confusions.

To model these two parameters, We have profiled the PARSEC benchmark suite [1] using Valgrind [22]. We use native input set for all applications except for x264, because Valgrind failed to capture source annotations for x264. We extract top kernels for every single application. Their presences and coverages, which are plotted in Figure 3, show trends:

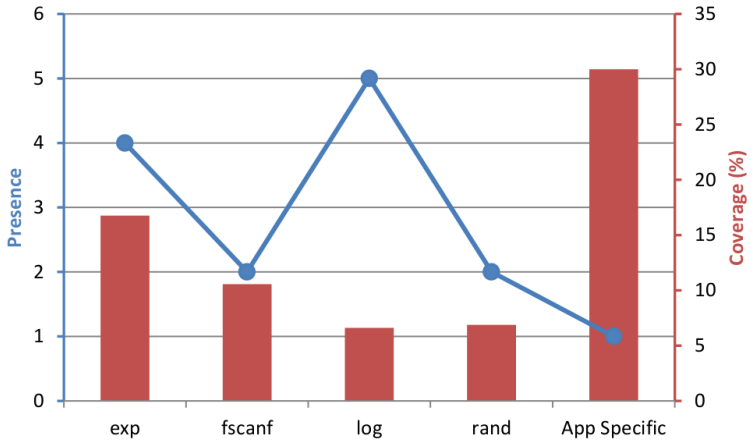



Figure 3. Kernels coverage and presence. Kernels are extracted from PARSEC applications, excluding x264. The coverage for exp, fscanf, log, and rand is averaged across their all occurrences. The coverage for application specific is the average of all application specific kernels.

- Kernels, such as library calls, have a larger probability of presence, but a lower coverage in a single application;
- Kernels, such as application-specific routines, have very limited presence, but a higher coverage once they present;
- Majority of kernels are application-specific.

Besides general-purpose benchmarks suite like PARSEC, we have observed a similar trend on domain-specific bencharmks suite such as SD-VBS [26] and Minebench [21]. Based on these observations, we propose a statistical approach to model kernels' speedup ($\eta$), presence ($\lambda$), and coverage ($\varepsilon$), which is shown in Equation 11.

$$\begin{cases} \eta & \sim \mathcal{N}(\mu_s, \sigma_s^2) \\ \lambda & = B(1, p) \qquad \text{where } p = PDF(\eta) \\ \varepsilon & \sim \mathcal{N}(\mu_c, \sigma_c^2) \end{cases} \tag{11}$$

The speedups ($\eta$) of kernels are modeled as following a normal distribution. Kernels, like library calls, are mapped to medium speedup close to the mean of distribution, while application-specific kernels are mapped to tails of the distribution. The left tail denotes application specific kernels that are hard to accelerate, such as control-intensive ones. The right tail denotes application specific kernels that are highly efficient on accelerators, such as compute-intensive streaming ones. The presences ($\lambda$) of these kernels are modeled as following a Bernoulli distribution with $PDF(\eta)$ as its parameter. The coverage ($\varepsilon$) is modeled as following another independent normal distribution. We extract parameters for speedups by normalizing reported values from recent publications in reconfigurable logic community. We assume a fixed scaling ratio 5x for ASIC performance to their corresponding RL implementations. We justify parameters for coverage with values we collected from PARSEC. Values of all parameters are summarized in Table 4

| $\mu_s$ RL | $\mu_s$ ASIC | $\sigma_s$ RL | $\sigma_s$ ASIC | $\mu_c$ | $\sigma_c$ |
|---|---|---|---|---|---|
| 40x | 200x | 10 | 50 | 40% | 10% |

Table 4. Summary of statistical parameters for kernel modeling

### 2.6. System Configuration

The system is modeled as a symmetric multi-core system composed of in-order cores. We assume the power and the area of the last-level cache (LLC) remains a constant ratio to the whole system. According to [19], we take the assumption that un-core components attribute to 50% of both the total thermal design power (TDP) and the die area; The bandwidth of the memory subsystem is assumed to be sufficient for future technology nodes by applying advanced techniques such as 3D-stacking, optical connection, multi-chip modules, etc. In the rest of this section, the power and the area of a system only refer to core components.

We have modeled three systems with different configurations of power and area by extracting data from commodity processors fabricated in 45nm. They include a small system representing desktop level processors [6], a medium system for normal server processors [7], and a large system that represents

the most aggressive server throughput processors [8]. One more parameter is introduced as the power density of a system, reflecting the maximum power allowed for a unit area. It is calculated by dividing the area from the TDP. Detailed parameters of system configurations are summarized in Table 5.

| System Type | Area ($mm^2$) | TDP ($W$) | P. Density ($W/mm^2$)† | # of Cores* |
|---|---|---|---|---|
| small | 107 | 33 | 0.308 | 14 |
| medium | 130 | 65 | 0.5 | 17 |
| large | 200 | 120 | 0.6 | 26 |

* Computed based on the single core area at 45nm from Table 1.
† P. density is defined as maximum power allowed for a unit area.

Table 5. System configurations. All numbers are for core components, attributed to 50% of the total TDP and the die area.

Within an application, we assume that each kernel takes considerable amount of computation time for a single run. Therefore, we can ignore various overhead in reconfiguring accelerators, as well as setting up the execution context. We leave the detailed overhead modeling and the modeling of transient kernel behaviors as future work.

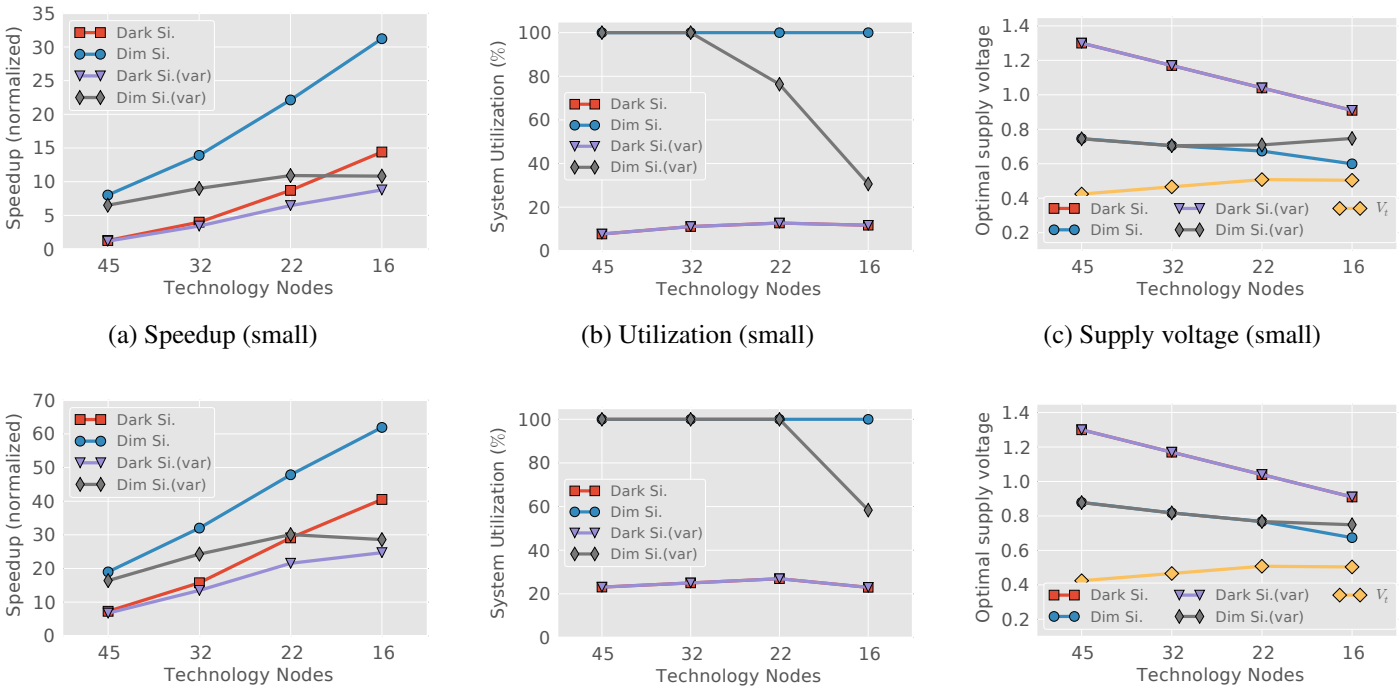
### 2.7. Lumos Release

Lumos is composed of a set of Python scripts that read in technology specific values mentioned in previous sections, as well as results from circuit simulations. Typically, Lumos needs a workload description as its input for analysis. This description is encoded in XML format, enumerating all applications that compose the workload. Each application is described by a parallel fraction, as well as possible kernels with their coverages in percentage. Each kernel is described by u-core parameters mentioned in section 2.5. We have pre-compiled a couple of descriptions for kernels and workloads following distributions as described in section 2.5. Lumos provides APIs to generate these XML descriptions by user-specified distribution parameters. Lumos requires the user to specify the physical constraints of a system, such as thermal design power (TDP) and area, then define their own system organization by

explicitly allocating area to supported accelerators. Lumos will allocate the rest of area to conventional cores to form a heterogeneous many-core system. When all these are ready, Lumos will apply the workload to the user-specified system and brute-force search system configurations to pick the one with the best TDP-constrained performance. More details on its use can be found in Lumos's documentation at http://liangwang.github.com/lumos. Lumos is released under a BSD open-source license.

## 3. Analysis

### 3.1. Effectiveness of Dim Silicon with Near-threshold Operation




(a) Speedup (small) (b) Utilization (small) (c) Supply voltage (small)

(d) Speedup (large) (e) Utilization (large) (f) Supply voltage (large)

Figure 4. Dark silicon vs. dim Silicon, regarding the throughput based speedup, die utilization, and the supply voltage with optimal throughput. Two system budgets are plotted, small and large.

Unlike dark silicon, which we define as maximizing the frequency of cores to improve the overall throughput at the expense of potentially turning some off, dim silicon aims to maximize chip utilization to improve overall throughput by exploiting more parallelism. Dark silicon systems apply the highest supply voltage, which is assumed to be 1.3x the nominal supply voltage in Lumos, to cores in parallel

mode; while dim silicon systems scale down supply and pick the optimal voltage according to the overall throughput. The comparisons between dark silicon (dark Si.) and dim silicon (dim Si.) are shown in Figure 4.

For performance, dim silicon beats dark silicon with up to 2X throughput improvement at 16nm. When variation comes into play, however, both systems experience throughput loss compared to non-variation cases respectively. Dim silicon systems suffer even more compared to dark silicon, revealing more vulnerability of dim silicon to process variations. Dim silicon is able to utilize more cores to achieve high utilization, up to 100% for the small system and the large system. Similarly, utilization of dim silicon systems is sensitive to the process variation. Being aware of variation, the small system with dim silicon sees the utilization drop to as low as 20%, quite close to the utilization obtained by dark silicon in the same technology node.

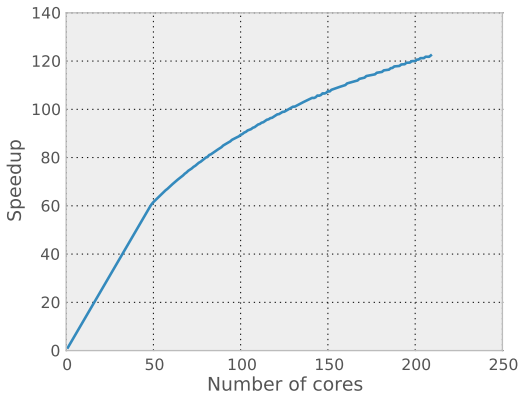
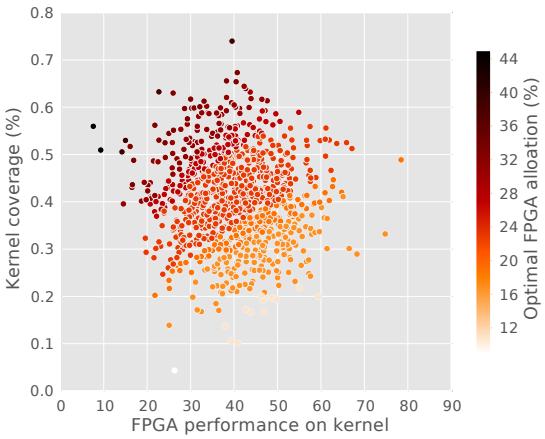



Figure 5. The speedup of IO cores at 16nm with HP process. The system is configured with large budget of 200 $mm^2$ in area and 120 $W$ in power. Diminishing returns are observed starting around 50 active cores.

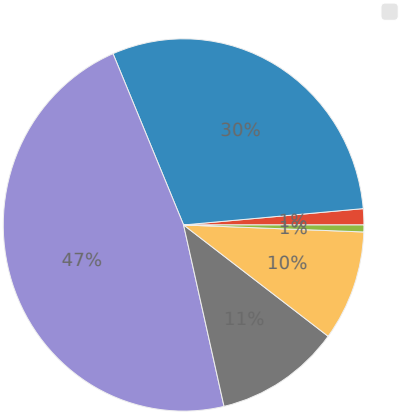
Although dim cores manage to deliver higher throughput, they suffer from diminishing returns as the number of active cores increases. As shown in Figure 5, a system with large budget, which is 200 $mm^2$ in area and 120 $W$ in power, starts to experience diminishing performance gains when the number of active cores pass over 50. This is the point when each core has to lower its supply voltage to stay within system-wide power budget. Therefore, the lower per-core frequency of dim cores compromises the speedup improvement coming from increasing cores. As a result, it is not cost effective to lower supply voltage aggressively closing to the threshold to reach the optimal throughput. The limited speedup

improvement from aggressive dim cores provides an opportunity to allocate some of the die area to more efficient customized hardwares, such as RL, and ASICs.

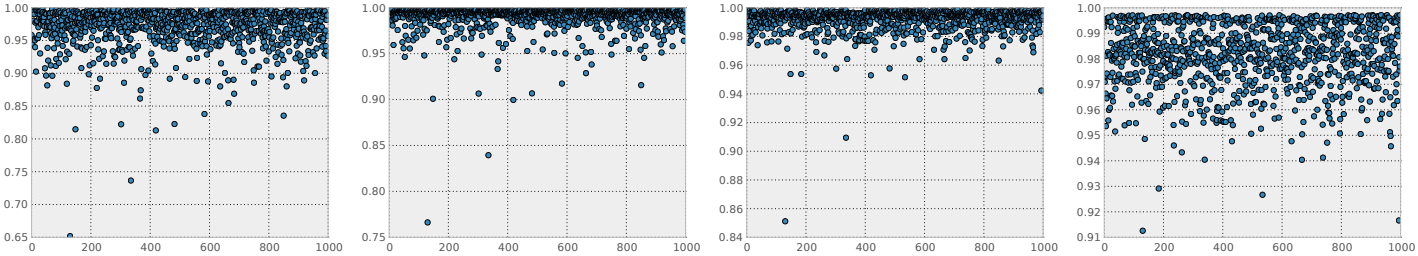
### 3.2. Dim Silicon with Reconfigurable Logic




(a) Optimal area allocation on RL




(b) Distribution



(c) 15% RL (d) 20% RL (e) 25% RL (f) 30% RL

Figure 6. Dim silicon plus RL. Optimal RL allocation for each application is plotted in (a), and the distribution of these optimal allocations are illustrated with a pie chart in (b). Besides, we show performance penalties for each application if the RL allocation is fixed across the workload, at 15%(c), 20%(d), 25%(e), 30%(f), respectively.

We start by looking at the scenario that every application in the workload has only a single kernel with considerable amount of coverage, e.g. an application-specific kernel. To do this, we generate a pool of kernels from a population of distribution described in Table 4, as well as applications associated with each of generated kernels. For each application, we adopt a brute-force approach to search for the optimal RL allocation ratio, with a step of 1% of total area budget. The results are shown in Figure 6. Despite various characteristics of each application, such as kernel speedup and coverage, a majority

of applications flavor RL allocations less than 30% (Figure 6a, 6b). This is because, RL with even a relatively small allocation manage to significantly accelerate the kernel, and the application performance is limited by the speedup dim silicon delivers on the rest part of the application, according to Amdahl's Law.

Considering a system with certain die area dedicated to RL, some kernels may suffer from performance penalties when the allocation is not large enough. We plot performance of such systems relative to the optimal speedup of each application in Figure 6c through 6f. Systems with smaller RL allocations, such as 15%, 20%, and 25%, have close-to-optimal performance for the most of applications, with a worst case performance loss of 35% with 15% allocation. While, on the other hand, the system with 30% RL allocation only experiences less than 5% penalties for most cases, and less than 10% for the worst-case. This indicates that the speedup is somewhat insensitive to RL allocation within a reasonable range.

We vary statistical parameters of speedup and coverage, with five other alternatives listed in Table 6. The sensitivity study on these alternatives is summarized in Figure 7. As the coverage drops down from high to low profile, the optimal area allocation drops down as well. This is because the lower the coverage of a single kernel, the less significant the kernel speedup at any allocation. As the RL speedup goes from fast to slow, the optimal area allocation goes up. This is because the less the speedup of the RL on kernels, the more area it requires to achieve a comparable performance.

| | Speedup | | | Coverage | |
|---|---|---|---|---|---|
| | Fast | Medium | Slow | High | Low |
| $\mu$ | 80 | 40 | 20 | 0.4 | 0.2 |
| $\sigma$ | 20 | 10 | 5 | 0.1 | 0.05 |

Table 6. Parameters for alternative distributions of speedup and coverage.

Then we take a look a more realistic scenario, with applications following characteristics defined by Equation 11. We choose ten synthetic kernels with speedup performance evenly spread from $\mu - 3\sigma$

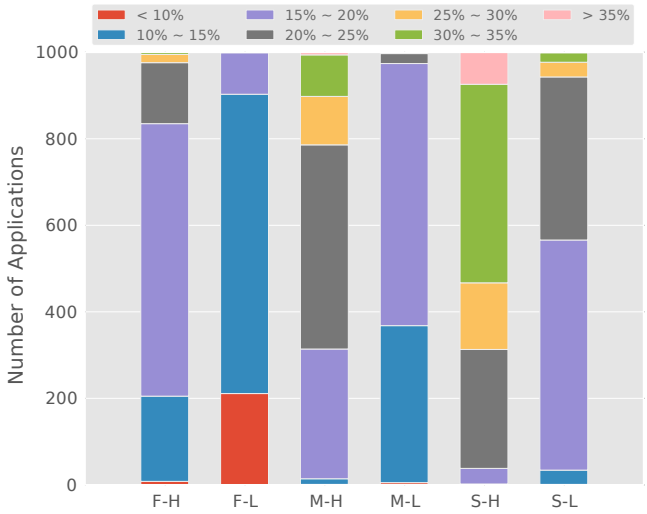



Figure 7. Optimal RL allocation with alternative distributions for speedup and coverage. Distributions for speedup cover three cases from slow to fast, while distributions for coverage cover two cases from low to high. Parameters for these distribution are detailed in Table 6.

through $\mu + 3\sigma$, representing various kernel performance and kernel's probabilities of presence in a specific application. We refer to these kernels with number 0 through 9: the speedup of kernels goes up from kernel 0 through kernel 9, while the probability of presence peaks at kernel 4 and 5, and is the smallest at kernel 0 and 9 (i.e. normal distribution). We draw a sample population of 500 applications. For each application, the sum of all kernels' coverage follows the given coverage distribution. We use the mean of speedups on all applications as the performance metric. We do a brute-force search to find the optimal RL allocation. The result suggests a optimal RL allocation around 20%, in Figure 8. A larger allocation on RL limits the number of available dim cores to accelerate the non-kernel parts of the application, delivering worse overall performance according to Amdahl's Law. For net speedup comparison of RL vs. dim silicon, see Figures 10-12 in following sections.

In summary, across a range of kernel characteristics, a relative small area allocation on RL, e.g. 20%-30%, is good enough to achieve the optimal throughput that is almost 3x larger than a non-RL system of only dim cores.

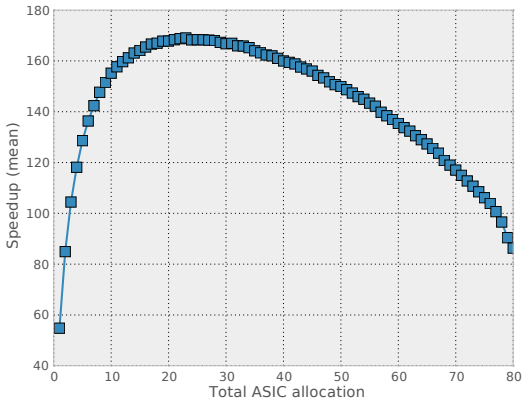
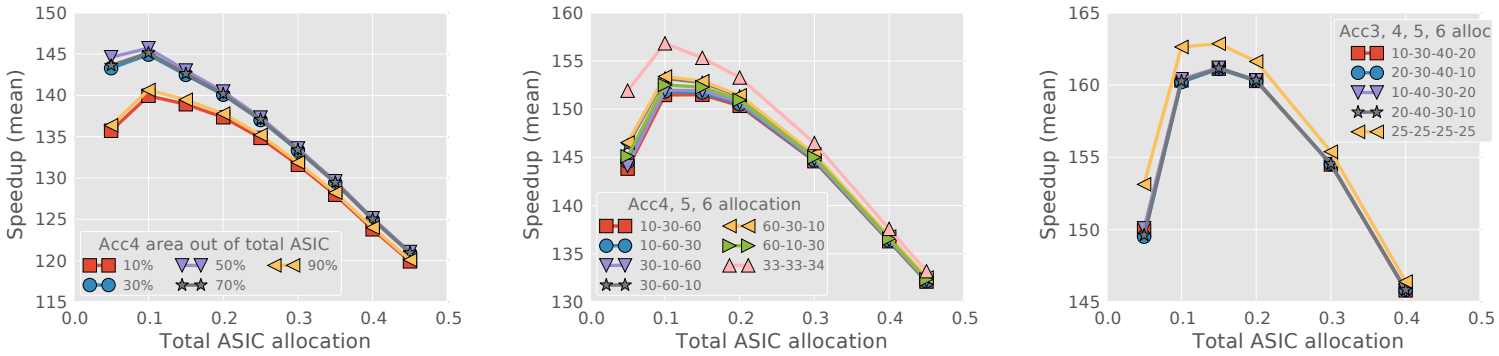



Figure 8. Speedup of various RL allocations. Optimal performance achieved at around 20% RL allocation.

### 3.3. Dim Silicon with ASICs

We use the same synthesized workload as described in the last subsection to study the performance implications with various fixed logic ASIC accelerators for these kernels. Although specialized accelerators can achieve tremendous speedup for application-specific kernels, limited overall coverage of its targeted kernel within a general-purpose workload leads to limited overall performance benefit for the workload as a whole. However, accelerators for library call kernels tend to be more beneficial in overall speedup of the workload due to their higher overall coverage. We plot potential allocations of 2, 3, and 4 accelerators for library call kernels in Figure 9. As the number of accelerators increases, the best achievable performance increases as well, from 140 to 155. With a given number of accelerators, the




(a) ASIC kernels of 4 and 5 (b) ASIC kernels of 4, 5, and 6 (c) ASIC kernels of 3, 4, 5, and 6

Figure 9. Speedup of a system composed by dim silicon and ASICs. We show configurations with tow accelerators ((a), three accelerators (b), and four accelerators (c). The X-axis is the total allocation to ASIC kernels, relative to the die area budget. The legend labels show the allocation of a kernel relative to the total ASIC allocation.

area allocation tends to be evenly distributed among all accelerators to achieve the best performance. The total area allocation to all hardware accelerators is limited to less than 20%.

### 3.4. Dim Silicon with Accelerators(RL and ASIC) on General-Purpose workload

In the previous two subsections, we have showed the benefits of accompanying conventional but dim cores with RL and ASIC accelerators, respectively. Both of them exhibit performance improvement over a baseline system composed by dim cores only. However, it is not necessary to achieve a better performance by combining both types of U-cores. As shown in Figure 10, the best performance is achieved with RL-only system organization. This counter-intuitive result comes from two reasons: First, we model the accelerator performance scaling proportional to its area, as well as we have a conservative assumption on the performance ratio between ASIC and RL (see Section 3.6 for the impact of alternative performance ratios). With these assumptions, RL will be more powerful on a kernel than the corresponding ASIC implementation as long as the RL is adequately larger than the ASIC accelerators, and a large RL allocation is justified by the high utilization achievable across multiple kernels. Consequently, the system ends up with a RL-only configuration. Second, with general-purpose workload, the average coverage of a kernel is small, due to either small coverages among applications (library-call kernels) or rare presence (application-specific kernels). This exaggerates the cost of a single ASIC accelerator which

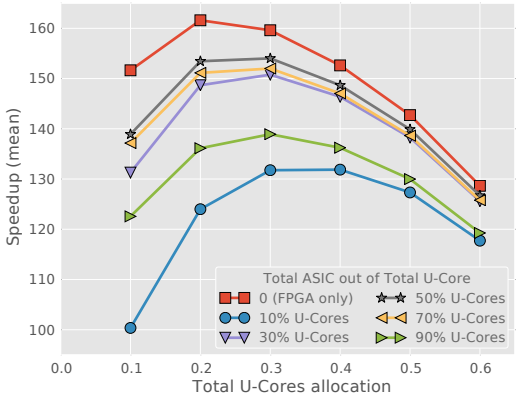
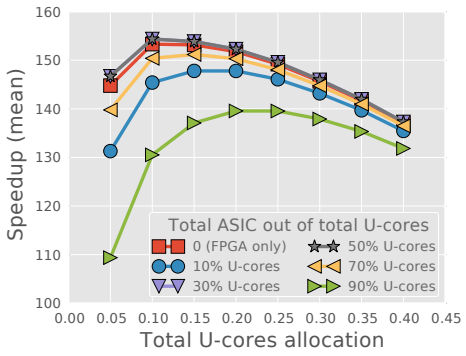



Figure 10. Speedup of a system composed by dim silicon, RL and ASICs. The X-axis is the total allocation to U-cores, including both RL and ASIC accelerators, relative to the die area budget. Legend labels indicate the total allocation of ASIC accelerators, relative to the total U-cores allocation. We assume an evenly distributed allocation among ASIC accelerators, since it shows the best performance in previous analysis.

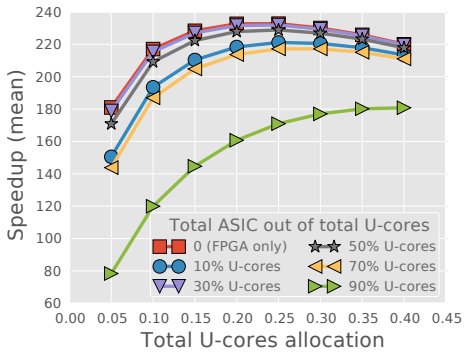
is only helpful for a specific kernel. As a result, the RL implementation is more favorable due to its versatility across kernels.

### 3.5. Benefit of ASIC Accelerators

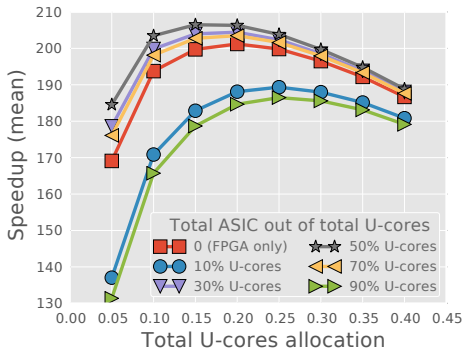
Although RL works better with general-purpose workloads, an ASIC accelerator becomes beneficial when its targeted kernel is common enough across applications in a workload. In order to capture this scenario, we add one more kernel to the set of ten kernels we have used in previous analyses. The coverage of this new kernel is fixed across all applications. We also hold the total coverage of all other kernels. By varying the coverage of the new kernel and all other kernels, we have generated several alternative workloads, within which the new kernel has a considerable amount of coverage across all applications.




(a) Fixed at 20%

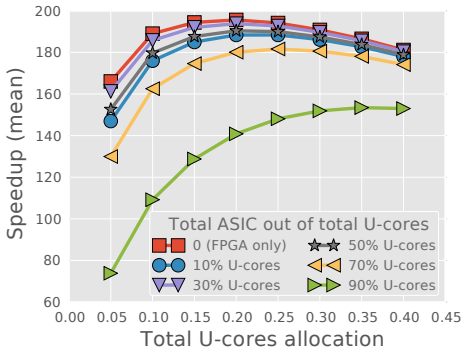



(b) Fixed at 30%

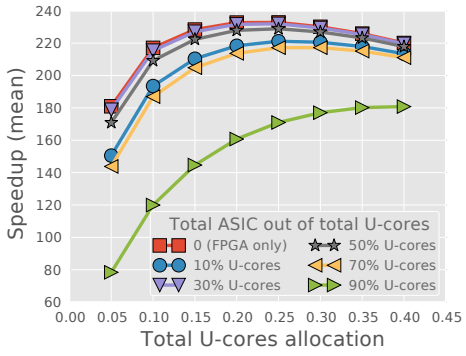



(c) fixed at 40%

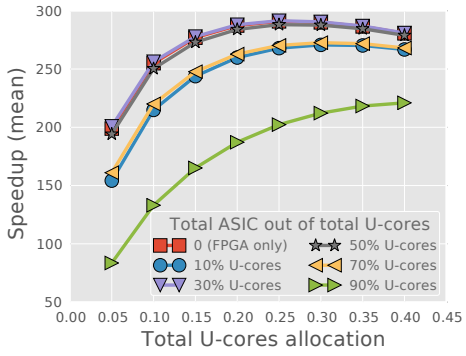
Figure 11. 10% coverage for all other kernels




(a) Fixed at 20%




(b) Fixed at 30%




(c) fixed at 40%

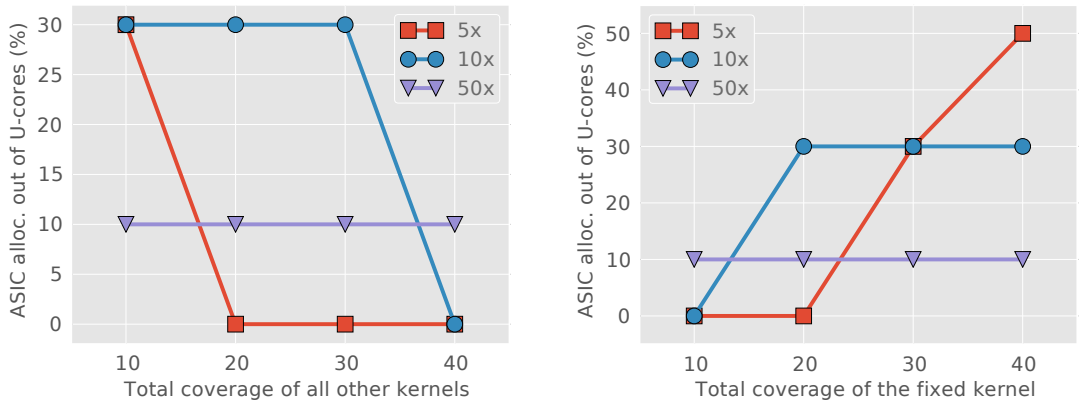
Figure 12. 30% coverage for all other kernels

We show results of 10% and 30% coverage of all other kernels in Figure 11 and 12, respectively. One of the most significant observations from these plots is that the dedicated ASIC accelerator is not beneficial at all until its targeted kernel covers more than the total of all other kernels within an application, e.g. in Figure 11a,11b,11c,12c. This observation is consistent with commercial MPSoC designs such as TI's OMAP4470 [17], which has dedicated accelerators for only image processing, video encoding/decoding, graphics rendering, since these functions are expected to be quite common for the target mobile device workloads.

### 3.6. Sensitivity on ASIC Performance Ratio

We are assuming that a fixed logic accelerator provides 5x better performance than the corresponding RL accelerator. This assumption could be quite conservative, especially when the reconfiguration overhead can not be amortized by RL's running time of a specific kernel. Alternatively, we increase the performance ratio to 10x and 50x, and plot the dedicated ASIC allocation for the kernel of a fixed coverage when the system achieves its optimal performance, regarding the total coverage of all other kernels in Figure 13a and fixed coverages of the kernel targeted by dedicated ASIC in Figure 13b.

In Figure 13a, the coverage of dedicated ASIC's kernel is fixed at 20%, while the total coverage of all other kernels varies from 10% through 40%. When the performance of dedicated ASIC accelerators




(a) 20% coverage of the fixed kernel

(b) 20% coverage of all the other kernels

Figure 13. Sensitivity study on ASIC performance ratio.

is merely 5x better than RL, the system ends up with zero allocation to the dedicated accelerator, unless the coverage of its targeting kernel (20%) is larger than the total coverage of all other kernels (10%). However, when the performance of dedicated accelerator boosts to 50x better than RL, the dedicated accelerator will always hold its place with 10% allocation out of all u-cores area. In the case of 10x performance ratio, dedicated accelerator is beneficial, unless the total coverage of all other kernels is as large as 40%, overwhelming the dedicated kernel's coverage of 20%. A similar trend is observed in Figure 13b, where the total coverage of all other kernels is fixed at 20% and dedicated kernel's coverage varies from 10% through 40%: dedicated kernel is favored when there is a huge gap between either: 1) the performance of the dedicated and the reconfigurable accelerator (e.g. 50x), or 2) the coverage of the dedicated kernel and the total coverage of all other kernels (e.g. 2x larger).

In summary, the benefit of ASIC accelerators is quite dependent to its relative throughput to RL accelerators on the same kernel. With a large throughput gap (e.g. 50x), ASIC accelerators are beneficial to be included in the optimal design, even when the targeted kernel has a limited presence across applications (e.g. as low as 10%). Otherwise, RL is preferable, as long as reconfiguration overhead can be neglected.

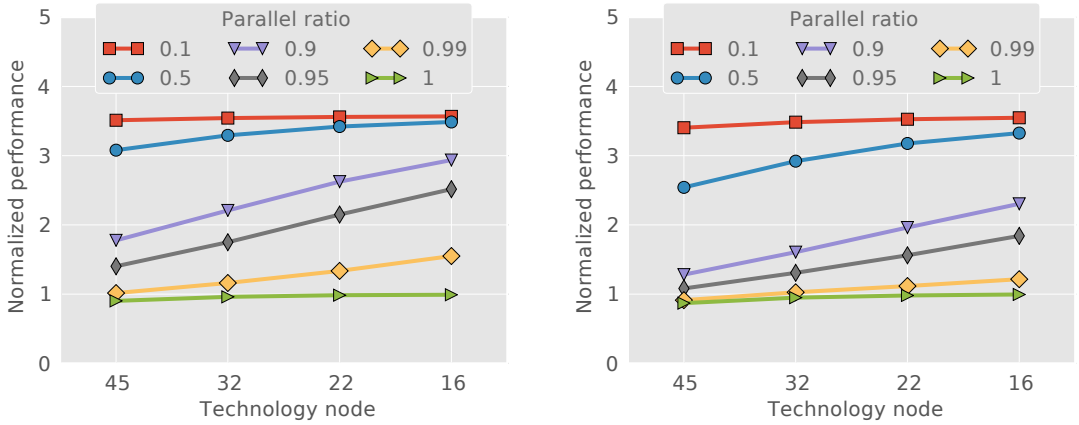
### 3.7. Alternative Serial Cores

Although massive parallelism has been observed in several computing domains, such as high performance computing, there are many applications with a limited parallel ratio. In this case, adding a "beefy" out-of-order (O3) core is more beneficial, especially when dim cores get diminishing returns on throughput. We extract the performance of the O3 core from SPEC2006 scores of a Core i7 processor and calculate its power and area using McPAT, which are normalized and summarized in Table 7.

| Perf. | Power | Area |
|---|---|---|
| 2.2x | 3.5x | 3.46x |

Table 7. Characteristics of an O3 core at 45nm, normalized to the in-order core at the same technology node.

We assume the same inter-technology scaling factors as we have used for in-order cores in all previous analyses. We also assume the serial core is gated off in parallel mode to save power for throughput dim




(a) Large chip with area of $200mm^2$

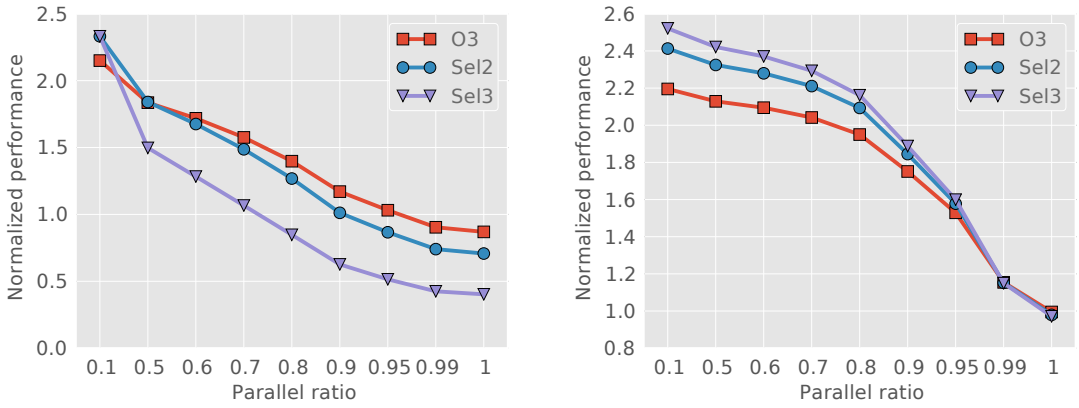
(b) Small chip with area of $107mm^2$

Figure 14. Relative performance by using O3 core to run the serial part of an application. System budget is large in (a) and small in (b), as characterized in Table 5. Y-axis is the performance normalized to the system at the same technology node, which only includes in-order cores and uses one of in-order cores as the serial core.

cores. As shown in Figure 14, even with an embarrassingly parallel application, the die area investment on dedicated O3 core is still beneficial. In the case of ideal parallelism of 100%, the performance loss by introducing a O3 core is around 14% at 45nm, While at 16nm, the performance loss is less than 1%, due to diminishing performance returns from a larger number of throughput cores, and lower percentage area impact of one O3 core.

Alternatively, the serial code can be executed by the best core selected from several out-of-order candidates, as proposed in [20]. In this work, Najaf et al. observed a 10% performance improvement by selecting from two alternative out-of-order cores, compared to an optimal-in-average out-of-order core. We model core-selectability with assumptions of 10% better performance but twice the area as an out-of-order core. To study the potential limits, we model an alternative organization of core-selectability with a total of three O3 cores to be selected, and assume another 5% performance boost by introducing more selections (a total of 15% over one O3 core "group"). We use the small system budget summarized in Table 5. As shown in Figure 15a, the original core-selectability (Sel2) shows less benefit as long as the parallel ratio is larger than 50% at 45nm. While, at 16nm (Figure 15b), the original core-selectability beats the conventional core with almost all parallel ratios. The core-selectability in alternative organiza-

tion (Sel3) delivers worse throughput at 45nm unless the application is almost serial with parallel ratio of 10%. However, when it comes to 16nm, the alternative core-selectability is better for almost all parallel ratios. This is, again, because the number of throughput cores is so large that more in-order cores suffer from diminishing performance returns. Therefore, the parallel performance penalty is limited when trading off a couple of in-order cores for single thread performance.




(a) 45nm (b) 16nm

Figure 15. Performance comparison among systems implementing the serial core as an out-of-order (O3) core, core-selectability of two O3 cores (Sel2), and core-selectability of three O3 cores (Sel3). The system budget is small from Table 5. Y-axis is the performance normalized to the system at the corresponding technology node, which only includes in-order cores and uses one of in-order cores as the serial core.

## 4. Related Work

The power issue in future technology scaling has been recognized as one of the most important design constraints by architecture designers [20, 27]. Esmaeilzadeh et al. performed a comprehensive design space exploration on future technology scaling with an analytical performance model in [10]. While primarily focusing on maximizing single-core performance, they did not consider lowering supply voltage, and concluded that future chips would inevitably suffer from a large portion of dark silicon. In [2], Borkar and Chien indicated potential benefits of near-threshold computing with aggressive voltage-scaling to improve the aggregate throughput. We evaluate near-threshold in more detail with the help of Lumos calibrated with circuit simulations. In [15], Huang et al. performed a design space exploration

for future technology nodes. They recommended dim silicon and briefly mentioned the possibility of near-threshold computing. Our work exploits circuit simulation to model technology scaling and evaluates in detail the potential benefit of improving aggregate throughput by near-threshold computing. Besides, there are numerous literatures studying the benefit of hardware accelerators as a response to dark silicon. Chung et al. studied the performance potentials of GPU, FPGA and ASIC in [5], regarding physical constraints of area, power, and memory bandwidth. Although very limited number of applications were studied in their work, our work corroborates that reconfigurable accelerators, such as FPGA, are more competitive than dedicated ASICs. Wu and Kim did a study across several popular benchmark suites regrading the targets to be accelerated in [28]. Their work suggested a large number of dedicated accelerators are required to achieve a moderate speedup due to the minimal functional level commonality in benchmark suites like SPEC2006. This is consistent with our observation that dedicated fixed logic accelerators are less beneficial due to limited utilization across applications in a general-purpose workload, and the importance of efficient, reconfigurable accelerators. In [25], Tseng and Brooks build an analytical model to study tradeoffs between latency and throughput performance under different power budgets and workloads. However, the model lacks the support of voltage and frequency scaling, especially near threshold and the capability of hardware accelerator performance modeling, limiting the design space explorations of future heterogeneous architectures. In [29], Zidenberg et al. propose MultiAmdahl model to study the optimal allocations to each computational units, including conventional cores and u-cores. The MultiAmdahl model can not support voltage scaling and near threshold effects, therefore its design space exploration capability is limited for heterogeneous systems under dark/dim silicon projections.

There are three challenges associated with near-threshold computing (NTC). First, the switching speed of a transistor slows due to small over-drive voltage. Second, devices are more sensitive to threshold variation when supply voltage is close to its threshold, leading to a significant increase in performance variations. Finally, variations in process, temperature and voltage make circuits less robust, especially for SRAM. In addition to pointing out those issues, Dreslinski et al. surveyed various techniques to accommodate those issues in [9]. They also mentioned the potential of NTC integration in ultra energy-

efficient servers to achieve high throughput. Our work studies NTC in detail with quantitative results to show the NTC could indeed be effective in high throughput computing. But unless the impact of variation is reduced, throughput boost will only be realized with embarrassingly parallel workloads. In [18], Krimer et al. demonstrated a near-threshold SIMD architecture with pipeline weaving for variation tolerance. Instead of energy-constrained throughput, our work focuses on power-constrained throughput. We do not target any specific systems or architectures, but more generally, our work studies near-threshold computing with systems variously configured in different technology nodes. It quantifies the potential of near-threshold computing as well as its limitations in context of power-constrained scaling.

## 5. Conclusions

In this paper, we develop *Lumos*, a framework for exploring the performance of heterogeneous systems operating at near-threshold (e.g. with dim cores). We find that dim cores manage to provide a moderate speedup up to 2x over conventional CMP architecture (suffering from dark silicon issue with further technology scaling). However, the poor per-core frequency of dim cores leads to a diminishing returns in throughput, creating opportunity for more efficient hardware accelerators such as RL and ASIC. Lumos models the general-purpose workload via statistical approach. We show that RL is more favorable with general-purpose applications, where commonality of kernels is limited. A dedicated ASIC accelerator will not be beneficial unless the average coverage of its targeted kernel is twice as large as that of all other kernels or its speedup over RL is significant (e.g. 10x-50x). However, it is hard to identify common function-level hotspots in realworld applications, and this is even true with domain-specific applications. In fact, the most important conclusion from this work is the need for efficient, on-chip, RL resources that can be rapidly reconfigured to implement a wide variety of accelerators.

## 6. Acknowledgements

We thank Mircea Stan and Martha Kim for their helpful comments. This work was supported by the SRC under GRC task 1972.001 and the NSF under grants MCDA-0903471 and CNS-0916908.

## References


[1] C. Bienia, S. Kumar, J. Singh, and K. Li. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *International Conference on Parallel Architectures and Compilation Techniques*, PACT '08, 2008. 10

[2] S. Borkar and A. A. Chien. The Future of Microprocessors. *Communication of the ACM*, 54(5), May 2011. 24

[3] J. A. Butts and G. S. Sohi. A Static Power Model for Architects. In *International Symposium on Microarchitecture*, MICRO '00, 2000. 5

[4] B. H. Calhoun, S. Khanna, R. Mann, and J. Wang. Sub-threshold Circuit Design with Shrinking CMOS Devices. In *International Symposium on Circuits and Systems*, ISCAS '09, March 2009. 3, 4

[5] E. S. Chung, P. A. Milder, J. C. Hoe, and K. Mai. Single-Chip Heterogeneous Computing: Does the Future Include Custom Logic, FPGAs, and GPGPUs? In *International Symposium on Microarchitecture*, MICRO '10, 2010. 2, 9, 25

[6] I. Corp. Intel® Core™2 Quad Processor Q9550S. http://ark.intel.com/products/40815. 11

[7] I. Corp. Intel® Xeon® Processor W5590. http://ark.intel.com/products/41643. 11

[8] O. Corp. Oracle SPARC T4 Processor. http://www.oracle.com/us/products/servers-storage/servers/sparc-enterprise/t-series/sparc-t4-processor-ds-497205.pdf. 12

[9] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge. Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits. *Proceedings of the IEEE, Special Issue on Ultra-Low Power Circuit Technology*, 98(2), February 2010. 25

[10] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Dark Silicon and the End of Multicore Scaling. In *International Symposium on Computer Architecture*, ISCA '11, 2011. 2, 24

[11] D. Fick, R. G. Dreslinski, B. Giridhar, G. Kim, S. Seo, M. Fojtik, S. Satpathy, Y. Lee, D. Kim, N. Liu, M. Wieckowski, G. K. Chen, T. N. Mudge, D. Sylvester, and D. Blaauw. Centip3De: A 3930DMIPS/W configurable near-threshold 3D stacked system with 64 ARM Cortex-M3 cores. In *IEEE International Solid-State Circuits Conference*, ISSCC '12, 2012. 2

[12] V. Govindaraju, C.-H. Ho, and K. Sankaralingam. Dynamically Specialized Datapaths for Energy Efficient Computing. In *International Symposium on High Performance Computer Architecture*, HPCA '11, 2011. 3

[13] R. Hameed, W. Qadeer, M. Wachs, O. Azizi, A. Solomatnikov, B. C. Lee, S. Richardson, C. Kozyrakis, and M. Horowitz. Understanding Sources of Inefficiency in General-purpose Chips. In *International Symposium on Computer Architecture*, ISCA '10, 2010. 2

[14] M. D. Hill and M. R. Marty. Amdahl's Law in the Multicore Era. *Computer*, 41(7), July 2008. 3

[15] W. Huang, K. Rajamani, M. R. Stan, and K. Skadron. Scaling with Design Constraints: Predicting the Future of Big Chips. *IEEE Micro*, 31(4), July 2011. 2, 24

[16] N. Inc. Nangate FreePDK45 Generic Open Cell Library. http://www.si2.org/openeda.si2.org/projects/nangatelib. 4

[17] T. Inc. OMAP4470 Mobile Application Processor. http://focus.ti.com/pdfs/wtbu/OMAP4470_07-05-v2.pdf. 21

[18] E. Krimer, R. Pawlowski, M. Erez, and P. Chiang. Synctium: a Near-Threshold Stream Processor for Energy-Constrained Parallel Applications. *IEEE Computer Architecture Letters*, 9(1), January 2010. 26

[19] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures. In *International Symposium on Microarchitecture*, MICRO '09, 2009. 3, 4, 11

[20] H. H. Najaf-abadi, N. K. Choudhary, and E. Rotenberg. Core-Selectability in Chip Multiprocessors. In *International Conference on Parallel Architectures and Compilation Techniques*, PACT '09, 2009. 23, 24

[21] R. Narayanan, B. Özışıyılmaz, J. Zambreno, G. Memik, and A. Choudhary. MineBench: A Benchmark Suite for Data Mining Workloads. In *IEEE International Symposium on Workload Characterization*, IISWC '06, 2006. 10

[22] N. Nethercote and J. Seward. Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation. In *Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '07, 2007. 10

[23] J. M. Rabaey, A. Chandrakasan, and B. Nikolić. *Digital Integrated Circuits: A Design Perspective*. Prentice Hall, second edition, 2003. 6

[24] M. B. Taylor. Is Dark Silicon Useful?: Harnessing The Four Horsemen of the Coming Dark Silicon Apocalypse. In *Design Automation Conference*, DAC '12, 2012. 2

[25] A. C.-N. Tseng and D. Brooks. Analytical Latency-Throughput Model of Future Power Constrained Multicore Processors. In *Workshop on Energy-Efficient Design*, WEED '12, 2012. 25

[26] S. K. Venkata, I. Ahn, D. Jeon, A. Gupta, C. Louie, S. Garcia, S. Belongie, and M. B. Taylor. SD-VBS: The San Diego Vision Benchmark Suite. In *IEEE International Symposium on Workload Characterization*, IISWC '09, 2009. 10

[27] G. Venkatesh, J. Sampson, N. Goulding, S. Garcia, V. Bryksin, J. Lugo-Martinez, S. Swanson, and M. B. Taylor. Conservation Cores: Reducing the Energy of Mature Computations. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '10, 2010. 2, 24

[28] L. Wu and M. A. Kim. Acceleration Targets: A Study of Popular Benchmark Suites. In *The First Dark Silicon Workshop*, DaSi '12, 2012. 25

[29] T. Zidenberg, I. Keslassy, and U. Weiser. MultiAmdahl: How Should I Divide My Heterogeneous Chip? *IEEE Computer Architecture Letters*, 11(2):65–68, 2012. 25