Dark Silicon is Sub-Optimal and Avoidable

Hamza Bin Sohail

Mithuna Thottethodi

T. N. Vijaykumar

TR-ECE-11-22

November 1, 2011

School of Electrical and Computer Engineering

1285 Electrical Engineering Building

Purdue University

West Lafayette, IN 47907-1285

Dark Silicon is Sub-Optimal and Avoidable

Abstract

Several recent papers argue that due to the slowing down of Dennard's scaling of the supply voltage future multicore performance will be limited by dark silicon where an increasing number of cores are kept powered down due to lack of power. Customizing the cores to improve power efficiency may incur increased effort for hardware design, verification and test, and degraded programmability. In this paper, we show that dark silicon is sub-optimal in performance and avoidable, and that a gentler, evolutionary path for multicores exists.

We make the key observations that (1) previous papers examine voltage-frequency-scaled designs on the power-performance Pareto frontier whereas the frontier extends to a new region derived by frequency scaling alone where voltage-scaled designs are infeasible, and (2) because memory latency improves only slowly over generations, performance of future multicores' workloads will be dominated by memory latency. Guided by these observations and a simple analytical model, we exploit (1) the sub-linear impact of clock speed on performance in the presence of memory latency, and (2) the super-linear impact of throughput on queuing delays. Accordingly, we propose an evolutionary path for multicores, called successive frequency unscaling (SFU). *Compared to dark silicon. SFU keeps powered significantly more cores running at clock frequencies on the extended Pareto frontier that are succesively lowered every generation to stay within the power budget. The higher active core count enables more memory-level parallelism, non-linearly offsetting the slower clock and resulting in more performance than that of dark silicon. For memory-intensive workloads, full SFU, where all the cores are powered up, performs 81% better than dark silicon at the 11 nm technology node. For enterprise workloads where both throughput and response times are important, we employ* controlled SFU (C-SFU) *which moderately slows down the clock and powers many, but not all, cores to achieve 29% better throughput than dark silicon at the 11 nm technology node. The higher throughput non-linearly reduces queuing delays and thereby compensates for the slower clock, resulting in C-SFU's total response latency to be within +/- 10% of that of dark silicon.*

1. Introduction

Historically, CMOS scaling has reduced transistor area and per-transistor dynamic power by about half and has improved switching speed by about 40% from one technology generation to the next. Specifically, Dennard's scaling of the supply voltage has allowed doubling the number of transistors without significantly worsening the dynamic power [8]. Recently, however, on one hand, Dennard's scaling has slowed down significantly due to its undesirable side-effects of higher leakage, narrower noise margins, and worse reliability (e.g., supply voltage reduces only by 2% now). On the other hand, transistor count, and hence the number of cores in a multicore, continue to double. Consequently, there is an exponential divergence between the core count and the per-core power Unfortunately, the total chip power budget cannot be increased due to limits on cooling and power delivery (i.e., the total chip power will remain constant). These trends imply an increasing power shortage in future generations.

A recent paper [9] analyzes these trends and asserts that an increasing number of cores must be kept powered down in future generations and that future multicore performance is fundamentally limited by *dark silicon*. Indeed, the imminence of dark silicon has been presaged for some time now [1] [1]. To alleviate the resulting performance loss, other papers [7, 10, 15] have suggested customizing the cores for specific functionalities to improve power efficiency so that more cores can be powered with the same power budget. Unfortunately, the customization option puts multicores on a potentially arduous path of (i) requiring customization to provide exponential improvements of power efficiency (i.e., every generation better than the previous), (ii) incurring increased effort for hardware design, verification, and test, and (iii) potentially degraded programmability.

In this paper, we show that dark silicon is sub-optimal in performance and avoidable, and that a gentler, evolutionary path exists where customization may be optional but not essential. Previous dark silicon papers [9, 10] examine design points along the power-performance Pareto frontier covering a large space of large and small core designs and voltage-frequency-scaled operating points. The papers assert that the dark-silicon configuration, in which a subset of the cores run at the Pareto-optimal clock speed while the rest are kept powered down, achieves the optimum performance for a given technology generation and power budget.

We make the key observations that (i) because voltage-scaled design points are infeasible, the Pareto frontier extends to a new region derived by frequency scaling alone; and (ii) because memory lags far behind processor clocks in speed, performance of most realistic workloads for future multicores will be dominated by memory latency and not processor clock speed (most future multicores with 16 or more cores are destined for servers with memory-intensive workloads). Our key result combines these two observations to show that lower frequencies on the new extended Pareto frontier enable powering of more or, in many cases, all cores of a multicore which achieve more overlap of memory latency and better performance than dark-silicon configurations (note that our claim is better, but not optimum, performance). We show that our results hold despite techniques for reducing, hiding, or tolerating memory latency via 3-D stacked memory, out-of-order issue, and simultaneous multithreading,

[1] The term "dark silicon" appears to have been coined by ARM CTO Mike Muller.

respectively.

We arrive at this result by exploiting two known non-linear effects, the first of which is captured by a simple analytical model for multicore performance. Our model shows that in the degenerate case of absence of memory latency, more cores running at slower clocks perform *as well as* fewer running at faster clocks under the same power budget as long as the workloads are sufficiently parallel (a condition also necessary for multicores in general). In the presence of memory latency, however, more cores running at slower clocks perform *better than* fewer cores running at faster clocks. This reversal occurs because of the non-linear impact of clock speed on performance in the presence of memory latency where more active cores achieve more overlap of memory latency so that the dominant memory component of execution time reduces far more than the slight increase in the smaller non-memory component due to the slower clock. In other words, our key insight is that slow silicon is better than dark silicon as long as the slow silicon makes memory accesses. While voltage scaling has historically exploited the non-linear (cubic) relationship between power and voltage, we propose that clock-performance non-linearity be exploited in the post-Dennard era. Due to this fundamental non-linearity, dark-silicon performance limit can be exceeded for many realistic multicore workloads.

Based on our model's predictions, we propose a gentler, evolutionary path for multicores than customized architectures, called *successive frequency unscaling (SFU)*, where more cores than dark silicon (and in many cases, all cores) are kept powered and run at *successively* slower clocks every generation to bridge the exponential divergence between the core count and per-core power in the post-Dennard era. While the linear relationship between power and frequency is well known, this paper is the first to propose successively slower clocks; dynamic voltage and frequency scaling (DVFS) dynamically changes the clock speed up or down for good power-performance but does not employ successively slower clocks.

We employ SFU in two contexts with different performance metrics. In the first context of workloads where job execution time is the only metric (e.g., scientific applications), we employ full SFU wherein we unscale frequency to power all the cores. Surprisingly, despite considerably slower clocks in later generations (e.g., sub-GHz) full SFU exceeds the dark silicon performance limit. Not surprisingly, however, SFU does not completely close the gap between dark silicon and a power-unconstrained system due to the slower clock. In the other context of enterprise workloads (e.g., on-line transaction processing) where both throughput and response latency matter, the slower clock of full SFU would degrade single-thread performance, and hence response latency. Accordingly, we employ *controlled successive frequency unscaling (C-SFU)* which moderately slows down the clock and powers many, if not all, cores to achieve better throughput than dark silicon. C-SFU avoids degrading response latency despite the moderate clock slowdown by exploiting the second non-linearity that the higher throughput of C-SFU non-linearly reduces the queuing component of response latency

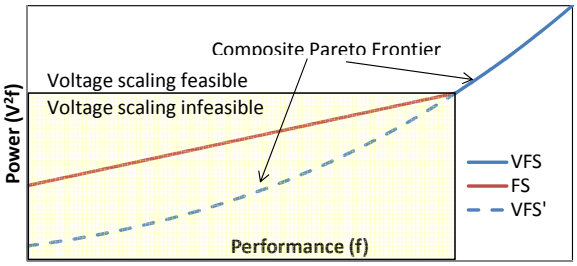

Figure 1: Power-performance Pareto frontier for perfect memory

and thereby compensates for the slower clock. We emphasize the simplicity of SFU which achieves better performance while incurring virtually no design effort or complexity, and thereby provides a viable evolutionary path for multicores.

The key contributions of this paper are:

- We refute the widely-believed inevitability and optimality of dark silicon. While our results may seem obvious in hindsight, dark silicon was published in ISCA and selected in 2012 Top Picks implying that our contrary results are not known; further, refutation, by definition, cannot be incremental over the assertion it refutes; and
- We propose the unusual idea of successively slower cores to stay within the power budget in the post-Dennard era.

The key results of this paper are:

- for memory-intensive workloads, SFU performs 81% better than dark silicon at the 11 nm technology node whereas for compute-intensive, workloads SFU performs similarly; and
- for response-time-sensitive enterprise workloads, C-SFU achieves 29% better throughput than dark silicon at the 11 nm technology node while maintaining the total response latency including queuing delays to be within +/- 10%.

The rest of the paper is organized as follows. Section 2 discusses our intuition and qualitative arguments behind SFU. Section 3 presents a simple model for multicore performance to provide quantitative corroboration of our intuition. Sections 4 and 5 validate our model using simulations of commercial and scientific workloads. Finally, we conclude in Section 6.

2. Intuition behind successive frequency unscaling

We now offer a high-level intuition on how our design circumvents the dark silicon limit on performance. In [9], the authors consider a large number of design points on the power-performance Pareto frontier (each point corresponds to a core design operating at some fixed voltage and frequency). Effectively, they consider multicore systems built with cores that achieve the best power-performance. Consequently, any attempt to achieve higher performance must face the following logical constraints. First, the new design point may not lie *on* the Pareto frontier because those points have already been examined in [9]. Second, the new design point (say X) may not lie *inside* the frontier because of the nature of the Pareto frontier. In such cases, one can always find other design points on the frontier (say Y) that *dominate* X; that is Y operates at the same power

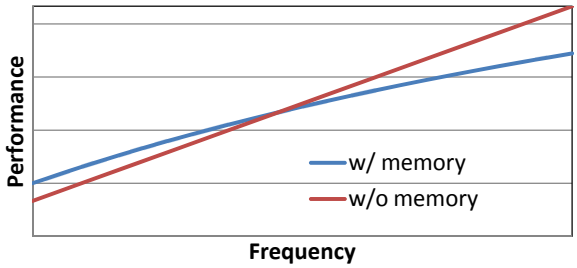

Figure 2: Frequency-Performance relationship (with and without memory latency effects)

as X while achieving higher performance. Then, we can simply replace cores of type Y with those of type X in the multicore and achieve higher performance for the same power budget.

Our design adheres to the above logical constraints. However, we show that the Pareto frontier may be extended to regions in the power-performance space that Esmailzadeh *et al.* [9] explicitly remove from consideration. Figure 1 illustrates the relationship between power on the Y-axis (proportional to $V^2 f$) and performance with perfect one-cycle memory on the X-axis (proportional to frequency f) under voltage-and-frequency scaling (VFS) and frequency-only scaling (FS). Note, the axes are shifted; the origin in Figure 1 does not correspond to the zero-power-performance point. We consider memory latency effects later in this section. The VFS curve illustrates the cubic relationship between power and perfect-memory performance. However, because of the inherent limits on voltage scaling, the VFS curve cannot extend indefinitely (the dashed VFS curve shown in the VFS-infeasible region). Esmailzadeh *et al.* rule out FS as an option by arguing that VFS dominates FS, which is indeed true in the region where VFS is feasible. However, in the VFS-infeasible region, we may use lower power-performance operating points by scaling down frequency under frequency scaling (FS) alone where power and perfect-memory performance are linearly related. Such FS design points are also on the Pareto frontier because there are no feasible VFS design points that dominate the FS design points. Thus, the Pareto frontier may be viewed logically as being a composite of (1) the old VFS frontier and (2) the new FS-only frontier.

Such scaling down of frequency – frequency unscaling – reduces the per-core power consumption and thus enables spreading the power budget over a larger number of cores. However, such spreading is useful only if it achieves higher overall multicore performance than dark silicon. Recall from Section 1 that SFU outperforms the dark silicon limit because of the non-linear impact of memory latency on the power-performance tradeoff.

Figure 2 qualitatively illustrates the relationship between performance (Y-axis) and frequency (X-axis) with and without memory effects. As with Figure 1, the axes of Figure 2 are also shifted; the origin in Figure 2 does not correspond to the zero-frequency-and-performance point. Not surprisingly, performance-frequency relationship is effectively sub-linear when memory latency effects are included because the fixed memory latency remains unchanged when frequency is scaled. When frequency is scaled up (to the right of the intersection in Figure 2), we observe a less-than-proportional increase in performance. However, when scaling down the frequency (to the left of the intersection), the same sub-linearity is advantageous in the sense that a large reduction in frequency results in less-than-proportional reduction in performance. Recall our earlier observation that the power-frequency relationship under frequency-only scaling is linear. This difference in the impact of frequency unscaling on power (linear) and performance (sub-linear) implies that our design loses only a little in performance due to its scaled-down frequency while gaining much more in power to keep many more cores active than dark silicon. These active cores achieve more overlap of memory latency, far exceeding the performance loss due to the lower frequency and enabling frequency unscaling to achieve higher performance than the dark-silicon limit. We emphasize that the non-linear impact of memory latency is crucial; if the impact were linear, frequency unscaling would yield no benefit.

The above discussion provides the basic qualitative arguments on the power-performance tradeoffs of frequency unscaling. Next, in Section 3, we develop a quantitative model of the combined impact of frequency unscaling and memory latency effects on overall multicore performance.

3. A multicore power-performance model

Our goal is to show that keeping all the cores powered and running at successively slower clocks over generations to stay within the power budget achieves higher performance than the dark silicon configuration where a subset of the cores are kept powered and running at faster clocks while the rest of the cores are powered down. To arrive at this result, we build a simple analytical model of multicore performance. Our model is derived from Amdahls' Law [4] and more recent revisits of Amdahl's Law in the context of multicores [11, 9]. However, recall from Section 1 that the key reason for our better performance is more cores achieving higher overlap of memory latency. Accordingly, our model specifically includes memory latency effects in addition to the usual serialization effects. While real-world applications may serialize arbitrary number of the threads for arbitrary amounts of time, our model follows the previous models in making the simplifying assumption that the parallel parts are fully parallel (i.e., no inter-thread dependence) and serial parts are fully serial (i.e., only one thread runs). Further, while voltage scaling has slowed down significantly in the near past, it has not stopped. However, continued voltage scaling even slowly is hard due to a multitude of inexorable trends such as higher leakage, lower noise margins, lower reliability especially of latches and SRAM cells, and higher susceptibility to soft errors. Consequently, we assume that voltage simply does not scale over technology generations.

3.1. Model

We first describe our model for a multicore that is not constrained by power. Then, we modify this model to include power constraints either via dark silicon (DS) or successive frequency unscaling (SFU). Let

- s be the serial portion of sequential execution time (i.e., $1-s$ is the parallel portion);
- c be the factor by which the clock frequency improves every technology generation (e.g., if the clock speed improves by 20% then c = 1.2); and
- m be the fraction of sequential execution time due to memory latency (i.e., $1-m$ is the non-memory, compute fraction).

To clarify our exposition, we relate the fraction m to the off-chip miss rate and main memory latency as follows: Assuming the number of off-chip misses per kilo instructions (MKPI) is r, the per-access average exposed main memory latency is mem_{lat} in processor clocks, and the processor clock cycles per instruction (CPI) with perfect on-chip caches (i.e., 0% off-chip miss rate) is $instr_{lat}$ then

$$m = r \times mem_{lat} / (1000 \times instr_{lat} + r \times mem_{lat}).$$

For example, assuming an off-chip miss rate of 2% which usually corresponds to r of 5, mem_{lat} of 400, and $instr_{lat}$ of 0.5 gives m = 0.80. We note that mem_{lat} denotes exposed memory latency, and hence covers both in-order- and out-of-order-issue cores though the latter's $mem_(lat)$ and m values would be smaller than the former's.

To simplify the model, we assume that

- both the parallel and serial portions of the application incur the same fraction m of execution time due to memory latency;
- employing more parallel cores does not change the fraction m of execution time due to memory;
- the factor c is constant across generations while in reality clock speed improvements may reduce in later generations resulting in an overall average of c;
- the cores do not employ simultaneous multithreading (SMT) (we include SMT later); and
- memory bandwidth scales with the number of cores (we revisit this assumption in our results).

Assume that execution time is split up into serial and parallel portions each of which include non-memory, compute fraction and memory fraction so that the total execution time can be expressed as

$$s \times ((1-m)+m) + (1-s) \times ((1-m)+m)$$

Over n technology generations, the non-memory, compute fraction $1-m$ scales as $(1-m)/c^n$ due to clock speed improvements while the memory fraction m remains unchanged. This scaling occurs for both the serial and parallel portions so that the serial portion scales as

$$s \times (\frac{1-m}{c^n} + m)$$

and the parallel portion scales as

$$(1-s) \times (\frac{1-m}{c^n} + m).$$

In addition, the parallel portion gets further sped up by a factor of 2 every generation due to the doubling of the core count so that the parallel portion scales overall as

$$\frac{(1-s) \times (\frac{1-m}{c^n} + m)}{2^n}.$$

Thus, after n generations since the last uniprocessor, a power-unconstrained (PU) multicore achieves a net speedup of

$$\frac{1}{s \times (\frac{1-m}{c^n} + m) + \frac{(1-s) \times (\frac{1-m}{c^n} + m)}{2^n}} \quad (1)$$

We note that while the $1-m$ compute terms in both the serial and parallel portions diminish exponentially over generations due to faster clocks, the m memory term in the parallel portion diminishes exponentially due to more cores' higher memory-level parallelism. For memory-intensive workloads, m is generally greater than $1-m$ making the number of cores more important than the clock speed for performance (from the above example, m is 0.80). As we will see shortly, this difference is the key contrast between DS and SFU where the former keeps only a subset of the available cores powered whereas the latter keeps all the cores powered albeit at a slower clock.

For the DS multicore which is constrained by power, let p be the factor by which the per-core dynamic power scales every technology generation (e.g., if power reduces by 20% then p = 0.8). Leakage power, which generally increases with lower supply voltage and higher transistor count, remains a constant fraction of the total power budget in post-Dennard generations (e.g., 25-30%) due to two reasons: (1) voltage does not scale anymore and (2) the transistor speed is reduced to compensate for the doubling of transistor count every generation (e.g., by fine-tuning the transistor threshold voltage). Recent commercial microprocessors follow this methodology to keep leakage under check [6]. Because the total power budget and leakage are constant across generations, the dynamic power budget is also constant. Further, we assume that p, like c, is constant across generations while in reality dynamic power improvements may reduce in later generations resulting in an overall average of p.

The number of cores that can be powered under the dark silicon option after n generations is $(2 \times 0.5/p)^n = (1/p)^n$. The DS multicore differs from the PU multicore only in the number of active cores —- $(1/p)^n$ versus 2^n; the DS configurations enjoy identical clock speed improvements and incur similar memory latency effects. Therefore, the DS multicore's net speedup after n generations is

$$\frac{1}{s \times (\frac{1-m}{c^n} + m) + \frac{(1-s) \times (\frac{1-m}{c^n} + m)}{1/p^n}} \quad (2)$$

We see that the DS multicore exploits significantly less memory-level parallelism than the PU multicore due to fewer active cores. This limitation considerably degrades performance for realistic multicore workloads which are memory-intensive.

To satisfy the power constraint, SFU successively scales down the clock over generations so that the per-core power from one generation to the next is half allowing twice as many cores to be powered. Let α be the factor by which the clock speed is unscaled every generation on top of the factor c provided by technology scaling, so that $\alpha \times p = 0.5$. The SFU multicore differs

from the power unconstrained multicore only in the scaling of the clock speeds — c^n versus $(\alpha \times c)^n$, while maintaining the same number of active cores (i.e., 2^n). Consequently, the SFU multicore's net speedup after n generations is

$$\frac{1}{s \times (\frac{1-m}{(\alpha \times c)^n} + m) + \frac{(1-s) \times (\frac{1-m}{(\alpha \times c)^n} + m)}{2^n}} \quad (3)$$

Comparing DS and SFU multicores, we consider all four components of execution time (the denominators in the above performance expressions): serial-non-memory ($s \times (1-m)$ terms), serial-memory ($s \times m$ terms), parallel-non-memory ($(1-s) \times (1-m)$ terms), and parallel-memory ($(1-s) \times m$ terms). DS reduces the serial-non-memory component by the factor of c^n and is better than SFU which reduces by the smaller factor of $(\alpha \times c)^n$. However, this component is likely to be small for parallel, memory-intensive workloads. DS and FSU are equal in the serial-memory and parallel-non-memory (the second and third) components. The equality in the second component is obvious. To see the equality in the third component, DS reduces the component by a factor of p^n/c^n whereas SFU reduces by a factor of $2^n/(\alpha \times c)^n$ where $\alpha \times p = 0.5$. DS reduces the (fourth) parallel-memory component by a factor of $(1/p)^n$ which is worse than SFU's factor of 2^n, highlighting our insight that slow silicon is better than dark silicon in the presence of memory latency (e.g., if p = 0.8, then DS's and SFU's factors are 1.25^n and 2^n, respectively, giving SFU a significant advantage in memory-level parallelism). As discussed above, in memory-intensive parallel workloads, the parallel-memory component is likely to dominate the other components, magnifying SFU's advantage.

Due to these reasons, SFU performs better than DS and it is possible and better to keep all the cores powered for memory-intensive workloads (i.e., DS is both avoidable and sub-optimal). The above quantitative analysis closely matches the qualitative analysis in Section 2.

3.1.1. Controlling single-thread latency: SFU, as proposed, improves (1) overall execution time of parallel workloads (e.g., scientific workloads) and (2) throughput of enterprise workloads (e.g., on-line transaction processing). However, full SFU requires scaling the clock by $(\alpha \times c)$ every generation which may degrade single-thread performance, and hence response latency, of enterprise workloads. This degradation may be considerable despite the sub-linear impact of clock on performance particularly in later generations where the clock is slowed down significantly with full unscaling. To address this issue, we exploit the other non-linear impact of throughput on queuing delays where higher throughput super-linearly reduces the queuing delay component of response latency as dictated by queuing theory, and thereby compensates for the slower clock. Based on this non-linear relationship, we propose to reduce the unscaling factor α compared to full SFU, still allowing more cores to be powered than DS, though not all the cores like full SFU. We choose α so that the resultant degradation of single-thread execution time is matched by the gain in the queuing delay due to higher throughput, resulting in similar total response latency and higher throughout as compared to DS configurations. We propose to apply such controlled SFU (C-SFU) only for response-latency-sensitive workloads and not for others.

Table 1: Clock and active core scaling with technology generations ($c = 1.07$, $p = 0.74$, $\alpha = 0.68$)

Parameter	Arch.	G0	G1	G2	G3	G4
Clock (GHz)	PU,DS	3.2	3.42	3.66	3.92	4.19
	FU	3.2	2.33	1.69	1.23	0.90
Active cores	PU,FU	4	8	16	32	64
	DS	4	6	8	12	16

Because C-SFU allows only a subset of the cores to be powered, we consider the option of converting the area of the remaining unpowered cores, which would otherwise be dark silicon, into additional last-level cache. While the same design can choose dynamically between full or controlled SFU based on the workload, this option cannot be applied dynamically and, if employed, would require different designs for full and controlled SFU. We consider this conversion option for DS configurations as well. Because memory is more dense in area than logic, this conversion adds transistors whose leakage is accounted for by giving up a few cores.

3.1.2. Effect of SMT: The above model does not include SMT. It may seem that because SMT reduces each core's exposed memory latency via better thread overlap, SFU's opportunity would decrease with SMT. However, SMT increases the number of threads to improve thread ovelap but does not change each thread's compute-memory overlap or performance (ignoring any extra cache misses due to SMT's increased cache pressure). SMT's thread overlap is no different than that achieved by the cores of a multicore (i.e., one core's memory latency is overlapped by the other cores' computation and memory accesses) and the non-linear impact of memory latency on power-performance tradeoff holds in a multicore irrespective of the number of cores, as seen in Equation 1. Therefore, the non-linearity holds for SMT as well, implying that the above model stays valid for SMT and that SFU opportunity would remain unchanged with SMT. In reality, SMT's increased cache pressure may actually increase cache misses and thereby increase the opportunity. We include SMT in all our results.

As an aside, we note that unlike SMT, out-of-order issue does increase each thread's compute-memory overlap and therefore, may decrease SFU's opportunity. We evaluate this issue in our results.

3.2. Model's predictions

To illustrate our model's predictions, we analyze four regions of the workload-characteristics space: (1) mostly serial and memory-unintensive (high s and low m), (2) mostly serial and memory-intensive (high s and high m), (3) highly and memory-unintensive (low s and low m), and (4) highly parallel and memory-intensive (low s and high m). In Figures 3(a) through

(d), we show the four regions with $s = 0.8$ and $m = 0.2$, $s = 0.8$ and $m = 0.8$, $s = 0.01$ and $m = 0.2$, and $s = 0.01$ and $m = 0.8$, respectively. The figure compares the PU (Equation 1), DS, (Equation 2), and SFU (Equation 3) multicores across technology generations. Each graph plots the speedups for the systems over a 4-core multicore, which corresponds to our generation 0, in the Y axis, and technology generations past our generation 0 on the X axis. We assume that the clock and power improvement factors across generations, c and p, are 1.07 and 0.74, respectively, in line with the conservative scaling trends in [7]. Based on this p, the frequency unscaling factor, α, is 0.68. Using these scaling values, Table 1 shows the clock frequencies and the number of active cores across generations (G0 through G4) for PU, DS, and SFU multicores.

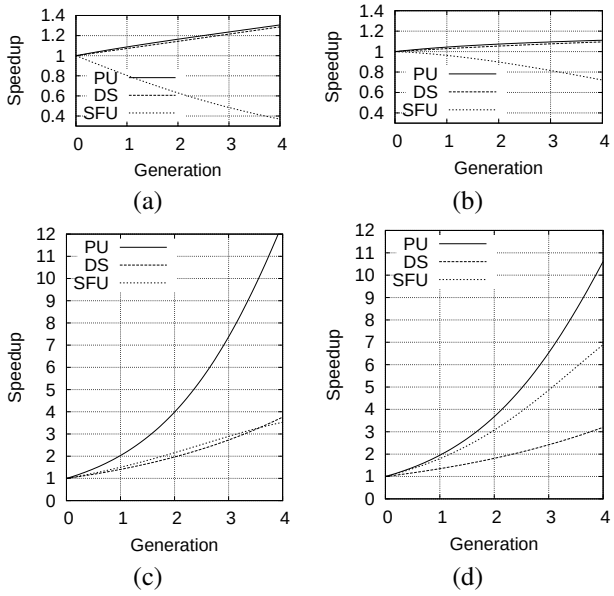

Figure 3: Speedups across generations of PU, DS, and SFU multicores over a four-core multicore (generation 0) for the following workloads: (a) mostly serial and memory-unintensive, (b) mostly serial, and memory-intensive, (c) mostly parallel and memory-unintensive, and (d) mostly parallel and memory-intensive

Figure 3(a) shows that for serial workloads ($s = 0.8$), the active core count does not matter and the DS multicore performs as well as the PU multicore. On the other hand, the SFU multicore performs worse due to its slower clock slowing down the serial-non-memory component. However, serial workloads are unimportant for future multicores which are destined for servers where multi-threaded workload is the norm. As such, the speedups are low even for the PU multicore due to the highly-serial workload, placing SFU within normal range. Further, power is not a problem for such workloads which need only a few cores.

The same analysis holds for Figure 3(b) though the speedups are even lower compared to those of Figure 3(a) because of the higher impact of memory latency which does not scale.

In Figure 3(c) which shows parallel, memory-unintensive workloads, the PU multicore performs better than both the DS and SFU multicores which perform similarly. The major non-memory component of this workload is affected by both the DS multicore's fewer active cores and the SFU multicore's slower clock, as compared to the PU multicore (Table 1). The DS and SFU multicores perform similarly due to their performance-equivalence for the parallel-non-memory component as shown above in Section 3.1. However, memory-unintensive workloads are likely to be uncommon for servers targeted by multicores.

Finally, Figure 3(d) shows that for parallel, memory-intensive workloads, the PU multicore performs better than the DS multicore for the same reason as Figure 3(c). Despite the lack of power constraints, the PU multicore does not achieve linear speedups with the number of cores due to the serial portions (Amdahl's Law effect). The SFU multicore performs better than the DS multicore due to higher memory-level parallelism achieved by SFU's more, albeit slower, cores in the major parallel-memory component of this workload. Nevertheless, SFU does not fully close the gap between the PU and DS multicores due to SFU's slower clock (Table 1) which affects both the serial-non-memory and parallel-non-memory components in the workload. Thus, this analysis corroborates the main results of the paper that dark silicon is sub-optimal and avoidable for many memory-intensive workloads. We show in Section 5 that our real-world commercial and scientific workloads closely track Figure 3(d).

Table 2: Scaling factors relative to Gen0 (45nm)

Parameter	Gen0	Gen1	Gen2	Gen3	Gen4
Tech Node (nm)	45	32	22	16	11
Clock (c)	1.00	1.10	1.19	1.25	1.30
Vdd	1.00	0.93	0.88	0.86	0.84
Capacitance	1.00	0.75	0.56	0.42	0.32
Dyn. Power (p)	1.00	0.71	0.52	0.39	0.29
SFU factor (α)	1.00	0.70	0.48	0.32	0.22

4. Experimental Methodology

We run full-system simulations using Wisconsin GEMS [12] built on top of Simics [3]. We simulate SPARC-based multicores running Solaris 10. In the remainder of this section, we describe the technology and architecture parameters for performance and power, and the benchmarks used in our experiments.

Technology Scaling: Recall that our technology scaling parameters are from the conservative projections by Borkar *et al.* [7]. Based on the projections, Table 2 shows the improvement factors over 45nm technology node for clock frequency, supply voltage (Vdd), capacitance, and per-core power. Recall that the clock and power factors are denoted as c and p in Section 3.1, The table also shows SFU factors (α in Section 3.1 where $\alpha * p = 0.5$).

Multicore architecture: We assume a tiled organization in which each multicore "tile" comprises a core, private L1 I- and D-caches, and a local bank of the shared, unified L2. The tiles are organized as a 2D-mesh network with memory controllers

are at the edges of the mesh network.

Core: While our baseline generation assumes in-order-issue, two-way SMT cores (line 1 in Table 3), we also show some brief results for out-of-order-issue cores. Because our scaling numbers in Table 2 are based on Intel designs citeborkar-scaling we use two-way SMT cores which is common for Intel designs so that the scaling numbers match the core's architecture. Based on the per-core power and clock factors from Table 2, we list the number of active cores and the actual clock frequencies for power-unconstrained (PU), dark-silicon (DS), and SFU multicores in Table 4. The core counts for PU and SFU scale as per Moore's Law. Recall from Section 2 that (1) no further voltage scaling is feasible and dark-silicon does not employ frequency scaling alone and instead employs fewer active cores ($(1/p)^n$ cores), and (2) frequency unscaling occurs on top of the clock frequency for the power-unconstrained multicore of each generation. While c and p are assumed to be constants in Section 3.1, Borkar's scaling assumes that the factors vary from one generation to the next. Consequently, clock frequencies in Table 1 and Table 4 differ slightly.

Cache Capacity and Access Latency: We hold key parameters of the L1 cache and the per-core L2 bank (capacity, block size, associativity and access latency in cycles) constant across generations (see lines 2 – 8 in Table 3). The aggregate L1 and L2 capacities double every generation as per Moore's Law. Though dark-silicon multicores keep many of their cores powered down, they use *all* of the shared L2 cache including the banks in the inactive cores' tiles (i.e., all three designs, PU, DS, and SFU, have the same amount of L2 cache in each generation).

We conservatively assume that the access latencies in ns of L1 and L2 slice scale at the same rate as overall frequency improvement. Consequently the access latencies in cycles, stays constant across generations for PU and DS (17 cycles, line 8 in Table 3). In SFU, the successively slower clocks imply correspondingly fewer number of cycles for L2 slice access (line 7, Table 4).

On-chip Network: We assume a simple, dimension-order routed, 2D mesh network (line 9 in Table 3). Network latencies (number of network hops) for remote L2 banks increase with the network size which grows across generations (line 8 in Table 4). Our simulator models a simple 1-cycle router delay per hop. In addition, we assume link latency scales slower than logic, as per the well-documented wire-delay phenomenon. Conservatively, we assume a modest 1-cycle increase in latency at Gen3 for power PU and DS. Because SFU sees slower clocks, its link latency in cycles reduces by one in later generations even after accounting for slower wires (lines 9 – 10 in Table 4).

DRAM Latency and Bandwidth (including impact of die-stacking): There are two sources of DRAM latency improvement: technology scaling every generation and one-time move to 3-D die-stacking. We assume that the former yields the usual 2% per generation (optimistic estimate as trends indicate no improvement in DRAM latency [2]) and show the resultant memory latency in cycles for PU, DS, and SFU in lines 11 – 12 in Table 4. We assume that the latter achieves the expected 30% latency reduction [10, 5] which we evaluate in Section 5.4.

Table 3: Invariant parameters across generations

	Parameter	Values
1	SMT contexts per core	2
2	Private L1D size (KB)	64
3	Private L1I size (KB)	64
4	Private L1 associativity	4
5	L1 access (cycles)	3
6	Shared L2 associativity	32
7	Shared L2 block size (bytes)	64
8	L2 slice (cycles) (PU/DS)	17
9	Network Topology	2D mesh
10	Channel width (bits)	64
11	Mem. Ctrl. Queue (entries)	32
12	Page mode	closed

We assume that the number of DRAM banks (i.e., the internal DRAM bandwidth) doubles every generation, in line with the scaling of DRAM density (line 13 in Table 4). We show our main results assuming that processor-memory bandwidth scales across generations (i.e., the number of memory channels scale from 1 to 16 across four generations, see line 14 in Table 4). This assumption is in anticipation of the imminent deployment of 3-D stacking technology. Further, a recent paper [14] shows that compression and other techniques can allow conventional pin bandwidth to scale beyond 100 cores, covering many future generations. Nevertheless, we include results constraining the number of memory channels in Section 5.3.

Leakage: As explained in Section 3.1, recent commercial microprocessors increase transistor threshold voltages and decrease transistor speeds to hold leakage at a constant 25-30% of the constant total power budget [6]. Consequently, the total dynamic power budget is also a constant. Borkar's conservative scaling, and therefore Table 2, include the effects of such adjustments. Accordingly, we assume a dynamic power budget of 112 W for all generations.

Workloads: We run the commercial and scientific workloads shown in Table 5. While the commercial workloads are naturally memory-intensive, the scientific workloads include both memory- and compute-intensive categories. To account for the general trend of data increasing with cache sizes across generations, we scale up the workload size across generations so that the off-chip miss rate (misses per kilo instructions) remains about the same across generations as shown in Table 5. Even higher miss rates would give more opportunity for frequency-unscaled multicores to achieve even more memory latency overlap than dark-silicon multicores. Consequently, the constant miss rate is a conservative choice. While the scientific workloads are amenable to easy scale-up by increasing the datasets, meaningfully scaling up the commercial workloads' datasets to 128 cores requires significant amount of domain expertise to achieve realistic settings (e.g., tune various benchmark parameters, choose scalable versions, and avoid software bottlenecks). Instead, we

Table 4: Parameter scaling with technology generations

	Parameter	Values					
		Gen0	Gen1	Gen2	Gen3	Gen4	Scaling Comments
1	Number of cores (PU/SFU)	4	8	16	32	64	2X per gen
2	Number of cores (DS)	4	6	8	12	16	Power-limited
3	Clock (GHz) (PU/DS)	3.2	3.52	3.81	4	4.16	As per Table 2
4	Clock (GHz) (SFU)	3.2	2.48	1.83	1.28	0.89	As per Table 2
5	Shared L2 size (MB)	4	8	16	32	64	2X per gen
6	Shared L2 banks	4	8	16	32	64	2X per gen
7	L2 slice (cycles) (SFU)	17	12	9	6	4	Same as f scaling
8	Network	2x2	2x4	4x4	4x8	8x8	
9	Link latency (DS/PU)	2	2	2	3	3	Slower than f scaling
10	Link latency (SFU)	2	2	2	1	1	Slower than f scaling
11	Memory (DS/PU)	320	344	368	380	400	2% reduction per gen
12	Memory (SFU)	320	240	180	124	86	2% reduction per gen
13	Number of DRAM banks	16	32	64	128	256	2X per gen
14	Number of memory channels	1	2	4	8	16	2X per gen
15	Number of instances (DS)	1	1	2	4	4	Problem size scaling
16	Number of instances (SFU/PU)	1	1	2	4	8	Problem size scaling

Table 5: Workloads: Description and Characteristics

Commercial Workloads	**MPKI**
SPECjbb: version 2005, Java-based 3-tier client/server system workload with emphasis on the middle tier. Java server VB version 1.5 with parallel garbage collection. We simulate a system with 24 warehouses (~600 MB).	7.1
Online Transaction Processing (OLTP): models database transactions of a wholesale parts supplier. We use PostgreSQL 8.3.7 database system and DBT-2 test suite which implements TPC-C benchmark. We reduced number of items and districts per warehouse and customers per district to allow a larger number of warehouse. We use a database of 25,000 warehouses (~5GB). We simulate 128 concurrent database connections.	2.4
Apache: version 2.2.9, a static web server workload with repository of 20,000 files (~500 MB). SURGE is used to generate web requests by simulating 1600 clients, each with 25ms think time between requests.	17.9
Scientific Workloads (memory-intensive)	
FFT: is a Splash benchmark that computes Fourier transforms. We run the transpose computation of 4 Million complex numbers (~64 MB) for generation 0 to 2 and 16 Million complex numbers for generation 3 and 4.	10.2
canneal: is a Parsec benchmark that models cache-aware annealing to optimize routing cost of a chip design. We use the native dataset (~100 MB). Systems for generations 0 to 2, 3 and 4 optimize 1, 2 and 4 chips respectively.	4.0
Streamcluster: is a Parsec benchmark that performs online clustering of an input stream. We use 1 million 128-dimensional points, 5000 intermediate centers (~100 MB) for generation 0 to 3 and and 2 million 128-dimensional points , 20000 intermediate centers (~ 275 MB) for generation 4.	4.8
Scientific Workloads (compute-intensive)	
Dedup: is a Parsec benchmark that performs next-generation compression with data deduplication. We use the simmedium dataset for all generations.	0.3
Fluidanimate: is a Parsec benchmark that models fluid dynamics for animation purposes with Smoothed Particle Hydrodynamics (SPH) method. We use the simmedium dataset for all generations.	0.6
Blackscholes: is a Parsec benchmark that computes option pricing with Black-Scholes Partial Differential Equation (PDE). We use the simmedium dataset across all generations.	1.5

scale up the commercial workloads by consolidating multiple instances of the same benchmark as shown in lines 15 – 16 of Table 4 (i.e., homogeneous consolidation as done in [13]). This methodology is in line with the trend in software consolidation especially on large multicores [13].

5. Experimental Results

We begin with our main results (Section 5.1) in which we compare across technology generations the performance of power-unconstrained (PU), dark-silicon (DS), and SFU multicores running our commercial and scientific workloads. While the previous comparison uses in-order-issue processors, we evaluate out-of-order issue processors in Section 5.2 to demonstrate that their latency-hiding ability does not change our results. While the main results assume that processor-memory bandwidth (i.e., the number of memory channels) scales across generations, we show the effect of constraining the bandwidth in Section 5.3. In Section 5.4, we study the effects of faster memory (e.g., via 3-D stacking). Finally, in Section 5.5, we compare the re-

sponse time (i.e., single-thread performance) and throughput achieved by controlled unscaling (C-SFU) and DS for commercial, transaction-processing workloads.

5.1. Performance

In Figure 4, we compare PU, DS, and SFU multicores across technology generations. We evaluate full SFU here and cover C-SFU later in Section 5.5. The figure shows performance normalized to that of a generation-0, four-core, PU multicore on the Y axis and technology generations past our generation 0 as well as the benchmarks on the X axis. The normalized performance for our consolidated commercial workloads (Section 4)is defined as the improvement in transaction rate whereas that for our scientific workloads is defined as improvement in execution time. Recall that PU and DS multicores use the same clock frequencies whereas the SFU multicore uses slower clock, and that while the power-unconstrained and SFU (full unscaling) multicores keep all the cores active, the DS multicore has fewer active cores.

From Figure 4, we observe that for all the benchmarks DS's relative performance improves slowly over generations with dark-silicon lagging well behind PU, as also shown in [9]. This lag is due to the difference in the number of active cores between PU and DS. By keeping all the cores active, albeit at slower clocks, SFU performs significantly better than DS for the memory-intensive benchmarks (i.e., *specjbb*, *OLTP*, *apache*, *FFT*, *canneal*, and *streamcluster* in Table 5), as predicted in Figure 3(d). For the compute-intensive benchmarks (i.e., *dedup*, *fluidanimate*, and *black-scholes* in Table 5), SFU performs similar to DS, as predicted in Figure 3(c). Because these benchmarks and *streamcluster* do not scale to generation-4 (64, 2-way SMT cores in PU) in our set up, we exclude those numbers. Overall, SFU achieves an average 81% improvement over DS for memory-intensive workloads (generation-4 geometric means). Recall from Section 3.2 that our model predicts this improvement based on the reasoning that the higher memory-level parallelism enabled by the greater number of active cores in SFU than in DS more than offsets the slower clock in SFU.

To corroborate the above reasoning, we show the misses per thousand instructions (MPKI) for the generation-0, PU multicore (see Table 5), and the average number of parallel memory requests for PU, DS, and SFU multicores (see Table 6). Because we commensurately scale the cache and workload sizes across generations Section 4, the MPKI for the other generations are similar; and because all the three multicores have the same amount of cache, the MPKI for the other multicores (DS and FU) are similar (not shown). As expected, the memory-intensive benchmarks' usual but non-trivial MPKI (but well below the excessive point of thrashing) combined with the greater number of parallel memory requests in SFU than in DS indicate that SFU achieves higher memory-level parallelism, and hence higher performance. Because the compute-intensive benchmarks have low MPKI (Table 5), the memory-level parallelism is low for both systems which , therefore, perform similarly. The performance results, which are the main claims of the paper, show that DS is sub-optimal and avoidable, and that slow silicon is better than dark silicon as long as the slow silicon makes memory accesses.

SFU, however, does not catch up to PU because while both systems have the same number of active cores and hence achieve the same amount of memory-level parallelism, the slower clock in (full) SFU results in slower compute portion. Nevertheless, the gap between SFU and PU is narrow in the earlier generations and is wide only in the fourth generation, supporting our claim in Section 1 that SFU provides a gentle, evolutionary path for multicores which may need to be augmented with customized cores only in later generations.

Because DS and SFU do not differ for compute-intensive benchmarks, these benchmarks are not interesting for, and hence are excluded from, the rest of our analysis.

5.2. Impact of out-of-order issue

Our results in Section 5.1 use in-order-issue cores whereas out-of-order-issue cores can hide some memory latency via memory-level parallelism. Because the key advantage of SFU over DS is higher amount of memory-level parallelism, the memory-latency hiding ability of out-of-order-issue cores may affect the advantage. To resolve this issue, we show in Figure 5 performance for generation-2 DS, SFU, and PU multicores normalized to that of a generation-0, four-core, PU multicore (Y axis) running our memory-intensive workloads (X axis), where all the systems employ out-of-order-issue cores. We vary the number of SMT contexts as 2 and 4 in the X axis. Because out-of-order-issue simulations are long-running, we show results only for generation 2 (4-way SMT runs for *streamcluster* did not complete). From the figure, we see that the results for out-of-order issue closely match those for in-order issue generation-2 in Figure 4 except for *FFT* which saturates the memory bandwidth in all the systems, as evidenced by excessive queuing at the memory controllers. Overall, SFU retains its significant advantage over DS, resulting in 23% improvement with 2-way SMT (for in-order issue generation 2, the improvement is 39% in Figure 4). Further, SFU's advantage over DS increases from 2-way to 4-way SMT due to increased cache pressure (Section 3.1.2). The lack of difference between the results for in-order issue and out-of-order issue stems from the fact that despite its memory-latency hiding ability, out-of-order-issue cores cannot bridge the large gap in the memory-level parallelism between SFU and DS.

5.3. Impact of processor-memory bandwidth

Our results in Section 5.1 assume that the processor-memory bandwidth in terms of the number of memory channels scale from 2 to 16 across the four generations. Because SFU exploits higher memory-level parallelism requiring more processor-memory bandwidth than DS, we study the impact of processor-memory bandwidth on SFU. Figure 6 shows performance on the Y axis for generation-4 PU, DS, and SFU multicores normalized to that of a generation-0, four-core, PU multicore (all in-order issue) running our memory-intensive workloads (X axis) while varying the number of memory channels (also on the X axis) as 8 and 16 (default). We choose generation 4 because it has the highest bandwidth demand. From the figure, we see that

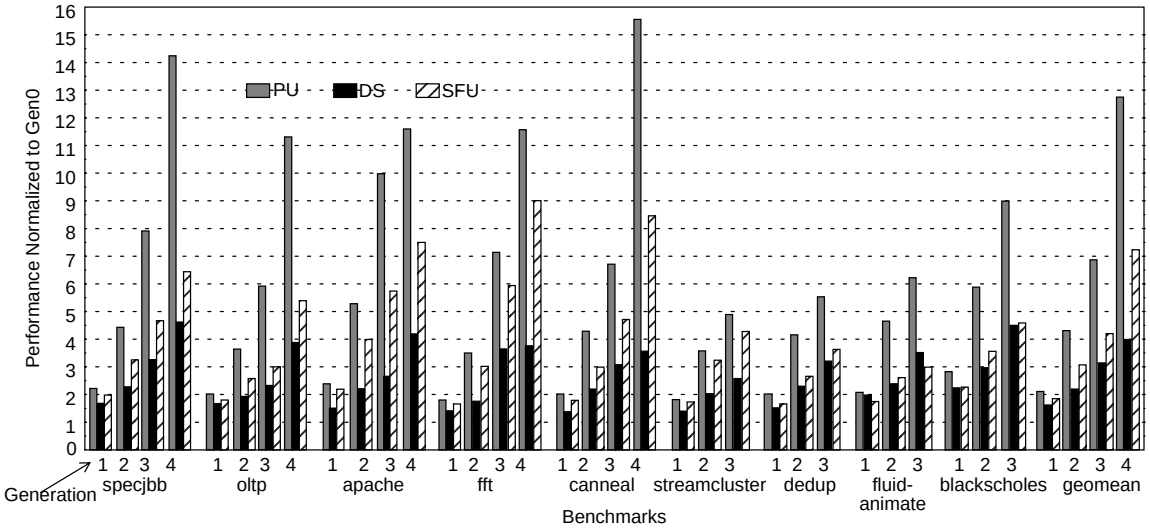

Figure 4: Speedups of power-unconstrained, dark-silicon, and successively-frequency-unscaled multicores over a generation-0, four-core, power-unconstrained multicore.

Table 6: Measured average number of parallel memory accesses

Benchmarks	Gen0	Gen1			Gen2			Gen3			Gen4		
		PU	DS	FU	PU	DS	FU	PU	DS	FU	PU	DS	FU
specjbb	1.4	2.5	1.8	2.3	4	1.8	2.6	6.8	1.9	3.1	19.6	2.0	5.5
oltp	1.3	1.5	1.4	1.5	2.1	1.5	1.6	2.7	1.6	1.8	6.4	1.7	2.6
apache	1.5	2.8	1.9	2.7	4.5	2.0	3.1	7.4	2.2	4.5	23.0	2.4	5.9
fft	1.4	2.6	1.9	2.6	4.9	1.9	4.0	9.3	2.1	8.9	32.5	3.4	18.7
canneal	1.5	2.9	1.9	2.8	4.9	2.0	3.6	6.5	2.1	5.1	19.7	2.2	7.1
streamcluster	1.2	1.7	1.4	1.7	2.4	1.5	2.1	4.2	1.7	3.5	17.4	2.1	6.4

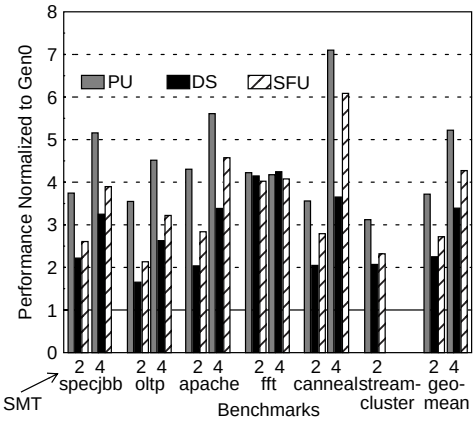

Figure 5: Impact of out-of-order issue

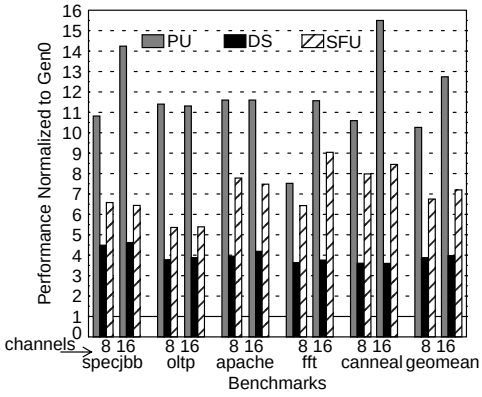

Figure 6: Impact of processor-memory bandwidth

SFU outperforms DS by a significant margin (74% on average) even with 8 channels though the margin increases with 16 channels for bandwidth-hungry applications like *FFT*. As discussed in Section 4, these amounts of bandwidth seem achievable by the time of generation 4, given the imminence of 3-D stacking and potential of compression and other techniques [14].

5.4. Impact of faster, 3-D stacked memory

Memory latencies also improve with 3-D stacking. Because SFU targets memory latency effects, we consider the effect of a 30% reduction in memory latency. Previously reported latency reduction for 3-D stacking ranges from 20% to 30% [5]. Figure 7 illustrates the effect of faster, 3-D stacked memory on performance (Y axis) for generation-4 PU, DS, and SFU multicores normalized to that of generation-0 PU multicore (all in-order

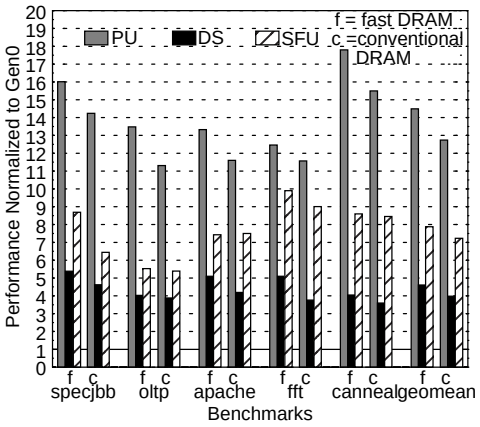

Figure 7: Impact of faster, 3-D stacked memory

issue) running our memory-intensive workloads (X axis). To facilitate comparison, we show performance with both faster memory (left three bars for each benchmark on the X axis) and conventional memory (right three bars for each benchmark). Not surprisingly, all configurations see a performance improvement from the latency decrease. However, the reduction in latency is not enough to eliminate the memory latency problem. As such, SFU is still on average 63% better than DS.

5.5. Single Thread Latency

Recall from Section 3.1.1 that to restore single-thread performance for response-time-sensitive workloads, controlled SFU (C-SFU) trades-off throughput by activating fewer additional cores than dark-silicon (but not as many as full SFU) for higher clock frequency than full SFU (but not as high as DS). To compensate for the remaining clock slowdown over DS, C-SFU exploits the super-linear reduction in queuing delays due to the higher throughput than DS. Also, recall the option of converting the unactivated cores into additional last-level cache for both C-SFU and dark silicon (DS). We empirically determine a C-SFU configuration that corresponds to acceptable throughput-latency trade-off as listed in Table 7 which shows the number of cores and frequency for C-SFU and DS with the "big-cache" option as well as full SFU and DS, all for generation 4 (where SFU has the biggest clock disadvantage). This C-SFU configuration reduces the clock slowdown compared to DS from full SFU's 4.7x (i.e., 0.89 GHz) to 40% (i.e., 2.48 GHz), affording 23 cores (i.e., 40% more than DS's 16) instead of full SFU's 64. To account for extra cache's leakage (Section 3.1.1), C-SFU's core count drops from 23 to 20 and DS's from 16 to 14. Due to the non-linear impact of memory latency, C-SFU's 40% slower clock amounts to 18% single-thread slowdown over DS.

Throughput: Figure 8 shows throughput (transactions per second) for C-SFU + big cache, DS + big cache, full SFU, and DS normalized to that of a gen0 system (Y-axis) for our commercial workloads which are sensitive to response times (X axis). We see that C-SFU performs better than DS by 29% (46% without big-cache) and almost as well as full SFU. Further, while DS improves marginally with big-cache (DS + big cache versus DS),

Table 7: Controlled SFU Configurations

Configuration	Cores	Clock (GHz)	L2 (MB)
DS + big cache	14	4.16	114
C-SFU + big cache	20	2.48	108
DS	16	4.16	64
full SFU	64	0.89	64

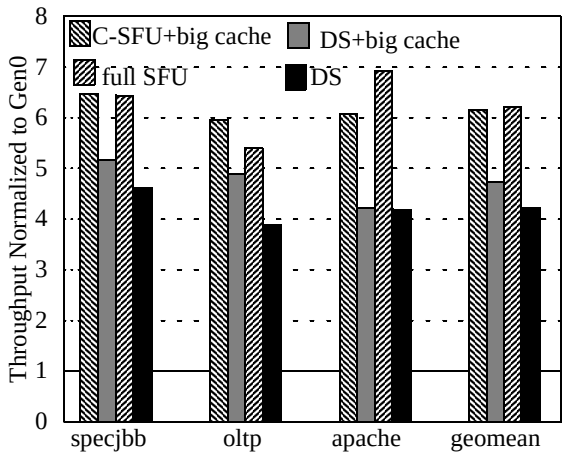

Figure 8: Normalized throughput

the improvement is not enough to match C-SFU.

Response Time: To analyze queuing delays, we use an M/M/m multiserver queuing model to relate the response time to throughput. We set the number of servers m to be 40 for C-SFU (20, 2-way SMT cores) and 28 for DS (14, 2-way SMT cores), and the service time (i.e., single-thread execution time) to be 1.18 for C-SFU and 1.0 for DS. Figure 9 plots the response time (Y-axis) vs. throughput (X-axis) curves for C-SFU and DS. The throughput is normalized to the saturation throughput of DS. The typical L-shaped curves show that at low loads, the response time is almost entirely service time (i.e., little queuing delay) and at high loads, the response time grows with the queuing delay, increasing rapidly near saturation. Typically, systems are operated in the flat, pre-saturation region to achieve reasonable response times and good throughput (a proxy for utilization). While C-SFU's response time is longer than DS's at low loads, cores can be turned off at low loads without throughput loss making power constraints irrelevant. However, in the fairly typical region of load, C-SFU stays within +/-10% of the response time of DS. Thus, C-SFU exploits the non-linear impact of higher throughput on queuing delays to compensate for its slower clock.

6. Conclusion

Researchers have responded to the slowing of Dennard's scaling by arguing that dark silicon inevitably imposes a performance limit and advocating for customization to harness more performance at the same power budget. Either approach imposes a heavy penalty; dark silicon necessitates a significant performance penalty, and customization places multicores on potentially ar-

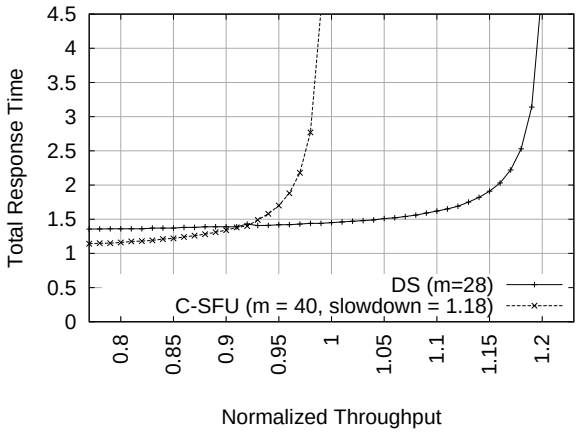

Figure 9: Total response time versus throughput

duous path of considerable design/programmability cost. We showed that dark silicon is sub-optimal in performance and avoidable, and that a gentler, evolutionary path for multicores exists.

This path, called successive frequency unscaling (SFU), involves successively scaling down the clock frequency with each technology generation. SFU is based on the insights that (1) frequency unscaling lowers per-core power where voltage-frequency scaling is infeasible, enabling more cores to be activated than dark silicon; and (2) typical multicore workloads are memory-bound and benefit from the increased memory-level parallelism achieved by the higher active core count. Guided by these insights and a simple analytical model, SFU exploits two non-linearities: (1) the sub-linear impact of clock speed on performance for memory-bound workloads and (2) the super-linear impact of throughput on queuing delays. The first non-linearity implies that SFU's increased memory-level parallelism more than offsets the slower clock so that for memory-intensive workloads, full SFU, where all the cores are powered up, performs 81% better than dark silicon at the 11 nm technology node. The second non-linearity comes into play for enterprise workloads where both throughput and response times are important and full SFU's aggressive clock slowdown may penalize response times. To address this issue, we proposed controlled SFU (C-SFU) which moderately slows down the clock and powers many, but not all, cores to achieve 29% better throughput than dark silicon at the 11 nm technology node. The higher throughput non-linearly reduces queuing delays and thereby compensates for the slower clock, resulting in C-SFU's total response latency to be within +/- 10% of that of dark silicon.

Finally, while SFU narrows the gap between dark-silicon and power-unconstrained mutlicores, the gap is not eliminated. We leave exploring other techniques to bridge the gap between SFU and power-unconstrained multicores for future work.

References

[1] "Arm-cto:power surge could create 'dark silicon'." [Online]. Available: http://www.eetimes.com/electronics-news/4085396/ARM-CTO-power-surge-could-create-dark-silicon-

[2] "Ddr4: Double the speed, double the latency?" [Online]. Available: http://www.chipestimate.com/techtalk.php?d=2011-11-22

[3] "Wind river simics." [Online]. Available: http://www.virtutech.com

[4] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," *Proceedings of the April 1820 1967 spring joint computer conference*, vol. 23, no. 4, pp. 483–485, 1967. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1465482.1465560

[5] B. e. a. Black, "Die stacking (3d) microarchitecture," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, 2006, pp. 469–479.

[6] S. Borkar, Personal communication, 2012.

[7] S. Borkar and A. A. Chien, "The future of microprocessors," *Commun. ACM*, vol. 54, pp. 67–77, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1941487.1941507

[8] R. Dennard, F. Gaensslen, V. Rideout, E. Bassous, and A. LeBlanc, "Design of ion-implanted mosfet's with very small physical dimensions," *Solid-State Circuits, IEEE Journal of*, vol. 9, no. 5, pp. 256–268, 1974.

[9] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proceeding of the 38th annual international symposium on Computer architecture*, ser. ISCA '11. New York, NY, USA: ACM, 2011, pp. 365–376. [Online]. Available: http://doi.acm.org/10.1145/2000064.2000108

[10] N. Hardevellas, M. Ferdman, B. Falsafi, and A. Ailamaki, "Toward dark silicon in servers," *IEEE Micro*, vol. 31, no. 4, pp. 6–15, 2011.

[11] M. D. Hill and M. R. Marty, "Amdahl's law in the multicore era," *Computer*, vol. 41, pp. 33–38, 2008.

[12] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, "Multifacet's general execution-driven multiprocessor simulator (gems) toolset," *SIGARCH Comput. Archit. News*, vol. 33, pp. 92–99, November 2005.

[13] M. R. Marty and M. D. Hill, "Virtual hierarchies to support server consolidation," in *Proceedings of the 34th annual international symposium on Computer architecture*. New York, NY, USA: ACM, 2007, pp. 46–56.

[14] B. M. Rogers, A. Krishna, G. B. Bell, K. Vu, X. Jiang, and Y. Solihin, "Scaling the bandwidth wall: challenges in and avenues for cmp scaling," in *Proceedings of the 36th annual international symposium on Computer architecture*, ser. ISCA '09. New York, NY, USA: ACM, 2009, pp. 371–382. [Online]. Available: http://doi.acm.org/10.1145/1555754.1555801

[15] G. Venkatesh, J. Sampson, N. Goulding-Hotta, S. K. Venkata, M. B. Taylor, and S. Swanson, "Qscores: trading dark silicon for scalable energy efficiency with quasi-specific cores," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-44 '11. ACM, 2011, pp. 163–174.