2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture

Single-Chip Heterogeneous Computing: Does the Future Include Custom Logic, FPGAs, and GPGPUs?

Eric S. Chung, Peter A. Milder, James C. Hoe, and Ken Mai
Computer Architecture Laboratory (CALCM)
Carnegie Mellon University, Pittsburgh, PA 15213
{*echung, pam, jhoe, kenmai*}*@ece.cmu.edu*

Abstract

To extend the exponential performance scaling of future chip multiprocessors, improving energy efficiency has become a first-class priority. Single-chip heterogeneous computing has the potential to achieve greater energy efficiency by combining traditional processors with unconventional cores (U-cores) such as custom logic, FPGAs, or GPGPUs. Although U-cores are effective at increasing performance, their benefits can also diminish given the scarcity of projected bandwidth in the future. To understand the relative merits between different approaches in the face of technology constraints, this work builds on prior modeling of heterogeneous multicores to support U-cores. Unlike prior models that trade performance, power, and area using well-known relationships between simple and complex processors, our model must consider the less-obvious relationships between conventional processors and a diverse set of U-cores. Further, our model supports speculation of future designs from scaling trends predicted by the ITRS road map. The predictive power of our model depends upon U-core-specific parameters derived by measuring performance and power of tuned applications on today's state-of-the-art multicores, GPUs, FPGAs, and ASICs. Our results reinforce some current-day understandings of the potential and limitations of U-cores and also provides new insights on their relative merits.

1. Introduction

Although transistor densities have grown exponentially from continued innovations in process technology, the power needed to switch each transistor has not decreased as expected due to slowed reductions in the threshold voltage (V_{th}) [1, 2, 3]. Therefore, the primary determinant of performance today is often power efficiency, not necessarily insufficient area or frequency. In addition to power concerns, off-chip bandwidth trends are also expected to have a major impact on the scalability of future designs [4]. In particular, pin counts have historically grown at about 10% per year—in contrast to the doubling of transistor densities every 18 months [5]. In this era of bandwidth- and power-constrained multicores, architects must explore new and unconventional ways to improve scalability.

The use of "unconventional" architectures today offers a promising path towards improved energy efficiency. One class of solutions dedicates highly efficient custom-designed cores for important computing tasks (e.g., [6, 7]). Another solution includes general-purpose graphics processing units (GPGPUs) where programmable SIMD engines provide accelerated computing performance [8, 9]. Lastly, applications on Field Programmable Gate Arrays (FPGA) can also achieve high energy efficiency relative to conventional processors [10].

Although energy efficiency plays a crucial role in increasing peak theoretical performance, the projected scarcity of bandwidth can also diminish the impact of simply maxing out the energy efficiency. For designers of future heterogeneous multicores, the selection of which unconventional approaches to incorporate, if any, is a daunting task. They must address questions such as: Are unconventional approaches worth the gains? Will bandwidth constraints limit these approaches regardless? When are (expensive) custom logic solutions more favorable than flexible (but less-efficient) solutions? *With the wide range of choices and degrees of freedom in this vast and little-explored design space, it is crucial to investigate which approaches merit further study.*

Building on the work of Hill and Marty [11], this paper extends their model to consider the scalability, power, and bandwidth implications of unconventional computing cores, referred throughout this paper as **U-cores**. In the application of our model, U-cores based on GPUs, FPGAs, and custom logic are used to speed up parallelizable sections of an application, while a conventional processor executes the remaining serial sections. In a similar spirit to Hill and Marty, the objectives of our model are not to pinpoint exact design numbers but to identify important trends worthy of future investigation and discussion.

To use our model effectively, various U-core-specific parameters must be calibrated beforehand. Unlike previous models that relied upon better understood performance and

1072-4451/10 $26.00 © 2010 IEEE
DOI 10.1109/MICRO.2010.36

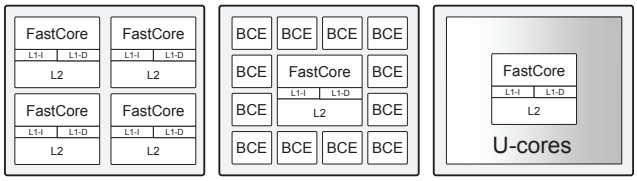

Figure 1: Chip models:
(a) Symmetric, (b) Asymmetric, (c) Heterogeneous.

power relationships between simple and complex cores (e.g., [11, 12, 13]), the relationships between conventional cores and U-cores are not as obvious. To obtain our U-core parameters, we measured the performance and power of tuned workloads targeting state-of-the-art CPUs, GPUs, and FPGAs. We also obtained estimates based on synthesized ASIC cores. Using our model, we study the trajectories of various hypothetical heterogeneous multicores under the constraints of area, power, and bandwidth as predicted by the International Technology Roadmap for Semiconductors (ITRS) [5]. Our results support several conclusions:

- *U-cores in general do provide a significant benefit in performance from improved energy efficiency despite the scarcity of bandwidth. However, in all cases, effectively exploiting the performance gain of U-cores requires sufficient parallelism in excess of* 90%.
- *Off-chip bandwidth trends have a first-order impact on the relative merits of U-cores. Custom logic, in particular, reaches the bandwidth ceiling very quickly unless an application possesses sufficient arithmetic intensity. This allows more flexible U-cores such as FPGAs and GPUs to keep up in these cases.*
- *Even when bandwidth is not a concern, U-cores such as FPGAs and GPGPUs can still be competitive with custom logic when parallelism is moderate to high (*$90\% - 99\%$*).*
- *All U-cores, especially those based on custom logic, are more broadly useful when power or energy reduction is the goal rather than increased performance.*

Outline. Section 2 provides background for the reader. Section 3 presents our modeling of U-cores. Section 4 covers the methods used for collecting performance and power. Baseline results are presented in Section 5. Using a calibrated model under ITRS scaling trends, the projections and key results are given in Section 6. We offer conclusions and directions for future work in Section 7.

2. Background

Our study of heterogeneous computing extends the analytical modeling for chip multiprocessors by Hill and Marty [11] to include U-cores based on unconventional computing paradigms such as custom logic, FPGAs, or GPUs. Figure 1 illustrates the chip models used in our study. The symmetric multicore in (a) resembles the architecture of commercial multicores today, consisting of identical high-performance microprocessors coupled with private and lower-level shared caches [14, 15, 16].

In (b), an asymmetric multicore consists of a single fast microprocessor for sequential execution, while nearby minimally sized baseline cores (we refer to these as Base-Core-Equivalent—BCE as termed in [11]) are used to execute parallel sections of code.

Finally, Figure 1 (c) illustrates a hypothetical heterogeneous multicore containing a conventional, sequential microprocessor coupled with a sea of U-cores. Although the illustration shows U-cores as a distinct fabric, they can also be viewed as being tightly coupled with processors themselves—e.g., in the form of specialized functional units. Since our studies focus on measurable technologies available today, we do not present results for the "dynamic" multicore machine model [11], which can hypothetically utilize all resources for both serial and parallel sections of an application. This effect is captured by our model if the resource in question is power or bandwidth.

2.1. Review of "Amdahl's Law for Multicore"

Before presenting the extensions of our model for single-chip heterogeneous computing, we first review Amdahl's law [17] and the analytical multicore model presented by Hill and Marty. Amdahl's law calculates speedup based on the original fraction of time f spent in sections of the program that can be sped up by a factor S. Speedup $= 1/(\frac{f}{S} + 1 - f)$.

The extended forms of Amdahl's Law for symmetric and asymmetric multicore chip models by Hill and Marty are reprinted below, which introduce two additional parameters, n and r, to represent total resources available and those dedicated towards sequential processing, respectively—in units of BCE cores. (Note: all speedups are relative to the performance of a single BCE core).

$$\text{Speedup}_{\text{symmetric}}(f,n,r) = \frac{1}{\frac{1-f}{perf_{seq}(r)} + \frac{f}{(n/r)\times perf_{seq}(r)}}$$

$$\text{Speedup}_{\text{asymmetric}}(f,n,r) = \frac{1}{\frac{1-f}{perf_{seq}(r)} + \frac{f}{perf_{seq}(r)+n-r}}$$

For all chip models, parallel work is assumed to be uniform, infinitely divisible, and perfectly scheduled. In their initial investigations [11], Hill and Marty use Pollack's Law [12] as input to their model, which observes that sequential processing performance from microarchitecture alone grows roughly with the square root of transistors used ($perf_{seq}(r) = \sqrt{r}$).

2.2. U-cores in Heterogeneous Computing

The focus of our study is the modeling of unconventional computing cores (U-cores) which were not considered previously. Our study covers three non-von-Neumann computing approaches that have shown significant promise in terms of energy-efficiency and performance: (1) **custom logic**,

(2) **GPGPUs**, and (3) **FPGAs**. Custom logic, in particular, can provide the most energy-efficient form of computation (typically 100-1000X improvement in either efficiency or performance) through application-specific integrated circuits (ASICs) customized to a specific task [18]. However, custom logic is costly to develop and cannot be easily re-purposed for new applications. Several proposals have suggested the co-existence of custom logic along with general-purpose processors in a single processing die (e.g., [6, 7]).

Flexible programmable accelerators such as GPGPUs have also been shown to significantly outperform conventional microprocessors in target applications [8, 9]. GPGPUs mainly derive their capabilities through SIMD vectorization and through multithreading to hide long latencies. In the space of flexible accelerators, programmable fabrics such as field-programmable gate arrays (FPGA) represent yet another potential candidate in single-chip heterogeneous computing. Unlike custom logic, FPGAs enable flexibility through programmable lookup tables (LUTs) that can be used to implement arbitrary logic circuits. In exchange for this flexibility, a typical 10-100× gap in area and power exists between custom logic and FPGAs [19].

2.3. Related Work

A substantial body of work has compared heterogeneous options such as custom logic, GPUs, FPGAs, and multicores (e.g., [20, 21, 22, 9]). However, most comparisons do not factor out area/technology differences or platform-specific limitations. Numerous projection studies based on ITRS have been previously carried out, focusing mainly on the scalability of chip multiprocessors (e.g., [23, 24, 25, 26, 4]). The growing body of research in single-chip asymmetric multicores is also highly relevant to this work (e.g., [27, 28, 29]).

In addition to multicores, many other types of heterogeneous designs exist such as domain-specific processors for digital signal processing [30], network off-loading [31], and physics simulations [32]. Notable products and prototypes include the IBM Cell processor [33] that integrates a PowerPC pipeline with multiple vector coprocessors, domain-specific computing prototypes [34, 35], and emerging integrated CPU-GPU architectures [36, 37].

Customization can also be facilitated through application-specific instruction processors (ASIPs). For example, the Xtensa tool from Tensilica [38] can generate custom ISAs, register files, functional units, and other datapath components that best "fit" an application. In the domain of reconfigurable computing, PipeRench [39] and RaPiD [40] were early projects that developed reconfigurable fabrics for specialized streaming pipelined computations, while Garp [41], Chameleon [42], PRISC [43], and Chimaera [44] combined reconfigurable fabrics with conventional processor cores. Tartan [45], CHIMPS [46], and other high-level synthesis tools can also target high-level code to FPGAs.

Table 1: Bounds on area, power, and bandwidth.

	Symmetric	Asym-offload	Heterogeneous
Area constraints	$n \leq A$	$n \leq A$	$n \leq A$
Parallel power bounds	$n \leq \frac{P}{r^{\alpha/2-1}}$	$n \leq P + r$	$n \leq \frac{P}{\varphi} + r$
Serial power bounds	$r^{\alpha/2} \leq P$	$r^{\alpha/2} \leq P$	$r^{\alpha/2} \leq P$
Parallel bandwidth bounds	$n \leq B\sqrt{r}$	$n \leq B + r$	$n \leq \frac{B}{\mu} + r$
Serial bandwidth bounds	$r \leq B^2$	$r \leq B^2$	$r \leq B^2$

Many extensions to Amdahl's Law [17] have been proposed. Gustafson's model [47] revisits the fixed size input assumption of Amdahl's Law. Moncrieff et al. model varying levels of parallelism and note the advantages of heterogeneous systems [48]. Paul and Meyer revisit Amdahl's assumptions and develop sophisticted models for single chip systems [49]. Morad et al. study the power efficiency and scalability of asymmetric chip multiprocessors [13]. Hill and Marty derive corollaries for single-chip multicores and argue for the importance of sequential processor performance [11]. Eyerman et al.'s multicore model introduced critical sections [50]. Woo and Lee [51] consider energy, power, and other metrics like performance per watt, while Cho and Melhem focus on energy minimization and power-saving features [52].

3. Extending Hill and Marty

In this section, we present extensions to the model by Hill and Marty to consider power and performance of single-chip heterogeneous multicore designs based on U-cores. In similar spirit to their investigations, our model is kept deliberately simple and omits details such as memory hierarchy/interconnect and how U-cores communicate.

3.1. Extending the Model for Power

To extend our models to include the effects of power limitations, we require a cost model that includes a power budget that cannot be exceeded in either serial or parallel phases of a workload. We first assume that a BCE core consumes an active power of 1 while fully executing, which includes both leakage and dynamic power. We assume that unused cores can be powered off entirely without the overhead of static power. To model a power budget, we introduce the parameter P, which is defined relative to the active power of a BCE core. To model power consumption of a sequential microprocessor relative to a BCE core, a simple power law is used to capture the super-linear relationship between performance and power using $power_{seq}(perf) = perf^{\alpha}$ where α was estimated to be 1.75 in [53]. Assuming Pollack's Law ($perf = \sqrt{r}$) [12], the power dissipation of the sequential core is related to the area consumed through $power_{seq} = perf^{\alpha} = (\sqrt{r})^{\alpha} = r^{\alpha/2}$.

Due to the higher power requirements of a sequential core, the speedup formula we use in our study later is a

modified asymmetric model called *asymmetric-offload*, which considers the case when the power-hungry sequential core is powered off during parallel sections.

$$\text{Speedup}_{\text{asymmetric-offload}}(f,n,r) = \frac{1}{\frac{1-f}{perf_{seq}(r)} + \frac{f}{n-r}}$$

With our new parameters and assumptions in place, the first three rows of Table 1 summarize how the original variables from Hill and Marty, n and r, are now bounded together by an area budget A (in units of BCE) and by power budget P. The interpretation of a "bounded" n is simply the maximum number of BCE resources that usefully contribute to overall speedup. For example—if the *n bounded by power* is less than the *n bounded by area*—power alone limits the number of resources that can contribute to overall speedup. Similarly, the serial power bounds shown in Table 1 also limit the amount of resources (r) that can be allocated for sequential performance.

3.2. Model Extension for Bandwidth

Apart from power and area, limited off-chip bandwidth is another concern that limits multicore scalability. We assume that a BCE core (at least) consumes the compulsory bandwidth of an application. We define this to be 1 for a particular workload. This assumption optimistically assumes that the working set size of an application or kernel can fit within on-chip memories and ignores details such as memory hierarchy organization. However, as we show later in Section 5, tuned algorithms with inputs that can fit within on-chip memories will typically achieve compulsory bandwidth (as measured in our workloads in Section 5). (The bounds in Table 1 can be easily modified if the application's bandwidth can be characterized as a function or constant factor of the compulsory bandwidth.)

For each workload with its own bandwidth requirements, a parameter B, in units of BCE compulsory bandwidth, defines the maximum off-chip memory bandwidth available. For bandwidth estimation of non-BCE processing cores, we assume that bandwidth scales linearly with respect to BCE performance—for example, a core twice as fast as a BCE core consumes a bandwidth of 2. Table 1 (bottom) summarizes how n and r are bounded to the maximum compulsory bandwidth supported by the machine.

3.3. Modeling U-cores

With power and bandwidth in place, we now introduce U-cores into our model. Recall from earlier that a U-core represents an unconventional form of computing such as custom logic, GPUs, or FPGAs. To model this, we begin with the assumption that a **BCE-sized** U-core can be used to execute exploitable parallel sections of code at a relative performance of μ compared to a BCE core. In addition, an actively executing U-core consumes a power of ϕ, which is in units of BCE power. Together, μ and ϕ characterize a design space for U-cores. For instance, a U-core with $\mu > 1$ and $\phi = 1$ can be viewed as an accelerator that consumes the same level of power as before. Similarly, if $\mu = 1$ but $\phi < 1$, the same level of performance is achieved as before but with lower power. For heterogeneous multicores based on U-cores, the new speedup formula is shown below:

$$\text{Speedup}_{\text{heterogeneous}} = \frac{1}{\frac{1-f}{perf_{seq}(r)} + \frac{f}{\mu(n-r)}}$$

We assume that the conventional microprocessor does not contribute to speedup during parallel sections of execution. Similar to Hill and Marty [11], we assume that parallel performance scales linearly with the resources used to execute parallel sections of code.

Table 1 (last column) shows how the n variable must also be similarly constrained by the power and bandwidth requirements of U-cores. (Note how lower ϕ values diminish the impact of power budget P, whereas higher μ values increase bandwidth consumption.) With our model in place, the next section describes the methodology used to obtain accurate parameters in order to perform our single-chip heterogeneous study using U-cores based on ASICs, FPGAs, and GPUs.

4. Methodology

To obtain reasonable *U-core parameters* (ϕ, μ), we measure performance and power of real systems. To satisfy the assumption in our model that U-core performance scales linearly with the amount of resources used, we ensured that all measured applications on a given system are compute-bound, such that performance increases would not be possible without more chip area.

Table 2 summarizes the various devices selected in our study. The Core i7-960 is a 4-way multicore from Intel [54] that serves as a baseline for calibrating BCE parameters. The GTX285 and GTX480 are two successive generations of high-performance, programmable GPUs from Nvidia [55], while the R5870 is a similarly-capable GPU from AMD [56]. The Virtex-6 LX760 is the largest FPGA available from Xilinx [57], and 65nm commercial synthesis flows are used to synthesize custom logic for each application.

In Table 2 we show the estimated area contribution from core- and cache-only components, which will be used later to estimate U-core parameters. For devices with available die photos, we subtracted non-compute areas such as on-die memory controllers and I/O. For the R5870 without a die photo, we based our estimates assuming a non-compute overhead of 25%. For the FPGA, we estimated the area per LUT and associated overhead to be roughly $.00191mm^2$, including the amortized overhead of flip-flops, RAMs, multipliers, and interconnect. To estimate areas for the ASIC cores, we used Synopsys Design Compiler [58] targeting a commercial 65nm standard cell library.

Table 2: Summary of devices.

		Core i7-960	GTX285	GTX480	R5870	V6-LX760	ASIC
Technology	*Year*	2009	2008	2010	2009	2009	2007
	Node	Intel/45nm	TSMC/55nm	TSMC/40nm	TSMC/40nm	UMC/Samsung/40nm	65nm
	Die area	263mm^2	470mm^2	529mm^2	334mm^2	-	-
	Core area	193mm^2	338mm^2	422mm^2	-	-	-
	Clock rate	3.2GHz	1.476GHz	1.4GHZ	1.476GHz	-	-
	Voltage	0.8-1.375V	1.05-1.18V	0.96-1.025V	0.95-1.174V	0.9-1.0V	1.1V
Platform	*System*	ASUS P6T	XFX 285	eVGA 480	XFX 5870	-	-
	Memory	3GB DDR3	1GB GDDR3	1.5GB GDDR5	1GB GDDR5	-	
	Bandwidth	32GB/sec	159GB/sec	177.4GB/sec	153.6GB/sec	-	-

Table 3: Summary of workloads.

	Core i7-960	GTX285	GTX480	R5870	LX760 ASIC
Dense Matrix Multiplication (MMM)	MKL 10.2.3	CUBLAS 2.3	CUBLAS 3.0/3.1beta	CAL++	Bluespec (by hand)
Fast Fourier Transform (FFT)	Spiral	CUFFT 2.3/3.0/3.1beta	CUFFT 3.0/3.1beta	–	Verilog (Spiral-generated)
Black-Scholes (BS)	PARSEC (modified)	CUDA 2.3	CUDA 3.1 ref.	–	Verilog (generated)

4.1. Overview of Workloads

The algorithms and workloads we study are shown in Table 3. Matrix-Matrix Multiplication (MMM), in particular, is a kernel with high arithmetic intensity and simple memory requirements. Fast Fourier Transform (FFT), on the other hand, possesses complex dataflow and memory requirements, while Black-Scholes (BS) was selected due its rich mixture of arithmetic operators. To satisfy compute-bound requirements, all kernels are assumed to be throughput-driven, i.e., many independent inputs are being computed. In all cases, we employed single-precision, IEEE-compliant floating point.

For the CPUs and GPUs, we used various tuned math libraries for Matrix-Matrix Multiplication (MMM). To obtain optimized FFT implementations for the Core i7, we utilized the Spiral framework [59]. For the Nvidia-based GPUs, we used the best results across three of the latest versions of CUFFT [60]. For Black-Scholes on the GPU, we used reference code from Nvidia. For the CPU, we used the multithreaded version from PARSEC [61] and added additional SSE optimizations. We were unable to obtain optimized FFT/BS for the R5870 and BS for the GTX480.

For the FPGA and ASIC, we developed an optimized implementation of MMM by hand. For FFTs, we used the Spiral framework to generate optimized synthesizable RTL [62]. Finally, for Black-Scholes, we developed a software tool to automatically create hardware pipelines from a high-level description of math operators. To ensure a fully-utilized FPGA, designs were scaled until timing could no longer be met. For the ASIC, we synthesized the same designs and estimated results using Synopsys Design Compiler 2009-06-SP5 configured with commercial 65nm standard cells. We used Cacti [63] to model SRAM area and power for the ASIC.

4.2. Power Methodology

To collect power data, a current probe was used to measure various devices while running applications in steady state. For the Core i7, we measured the EATX12V power rail, which only supplies power to the cores and private L1/L2 caches [64]. At time of publication, we did not possess a platform with the LX760 FPGA, and instead, relied on a smaller FPGA (LX240T/ML605) to perform our power measurements.

Although beyond the scope of this work, a significant amount of effort was placed into measuring GPU power consumption, due to the numerous non-computing related components (e.g., RAM). To achieve this, a set of microbenchmarks were designed to measure and subtract out non-compute power dissipation from on-die memory controllers and off-chip GDDR memory. Power results for the ASIC were estimated from synthesis tools.

5. Baseline Performance and Power

In this section we present baseline performance and power results. To explain how U-core parameters are derived, we use FFT as a case study, and present the remaining results for other workloads at the end. Starting with Figure 2 (top), all device performances are plotted from input sizes 2^4 to 2^{20}. However, without normalizing for die area, comparing any two devices at face value would not be fair in the context of designing a single chip multicore. Instead, Figure 2 (bottom) normalizes all performances to die area in 40nm/45nm. As expected, the ASIC FFT cores achieve nearly 100X improvement over the flexible cores (FPGA, GPU), and nearly 1000X improvement over the Core i7. Figure 3 shows the total power consumption of devices. A similar normalization step for comparing power is illustrated in Figure 4 (top). As expected, the ASIC achieves nearly two orders of magnitude improvement in power and 10X over the GPUs/FPGA.

A final step we took was to verify that the FFTs were compute-bound. To achieve this, we used CPU and GPU performance counters to measure off-chip bandwidth. A notable result for the GPU is shown in Figure 4 (bottom),

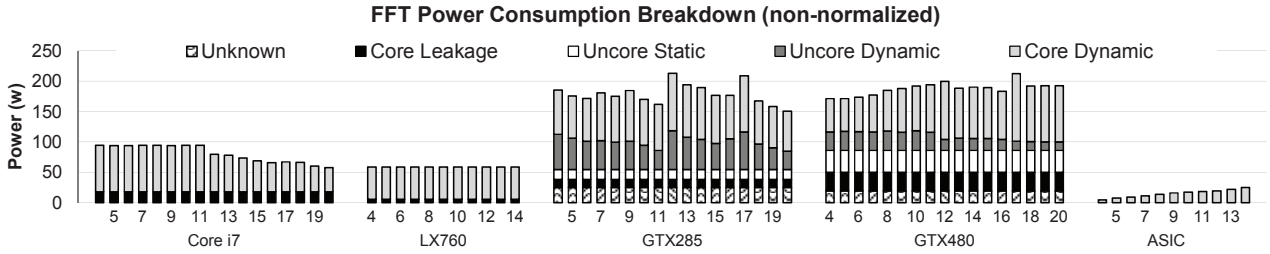

Figure 3: FFT Power Consumption (x-axis is FFT input size in $\log_2 N$).

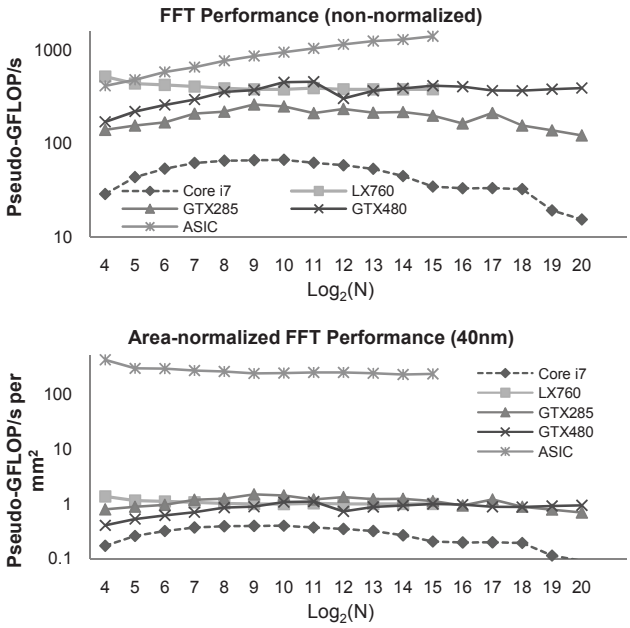

Figure 2: FFT performance in pseudo-GFLOPS/s (# FLOPS = $5N\log_2 N$).

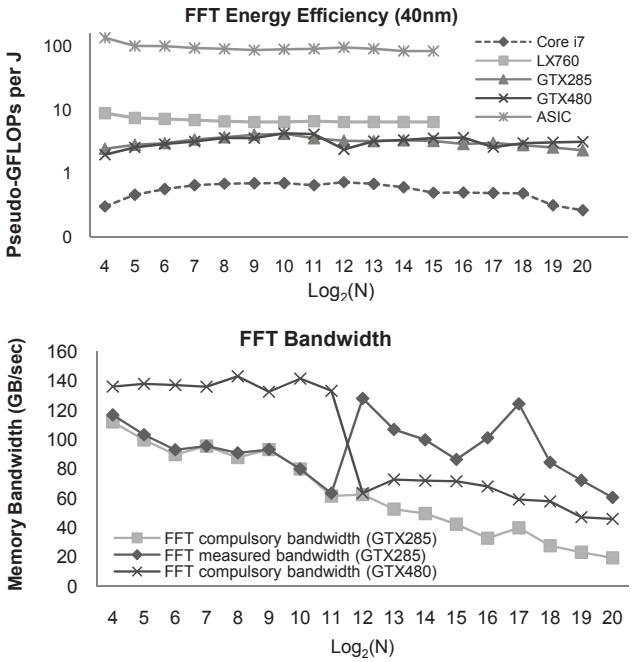

Figure 4: FFT energy efficiency and bandwidth.

where the GTX285 consumes compulsory bandwidth for input sizes until 2^{12} when the data no longer fits in on-chip memory. When data fits on-chip, the GTX285 does not reach the peak bandwidth of 159GB/sec, which suggests compute-bound performance. Interestingly, the GTX285 is still compute-bound for input sizes $\geq 2^{12}$ due to increasing arithmetic intensity and efficient out-of-core algorithms being employed. For the GTX480, we were unable to measure the bandwidth counters—thus, our results may not necessarily reflect compute-bound performance.

Summary of Results. The absolute performance and normalized results for remaining workloads (MMM and BS) are summarized in Table 4. For MMM, the R5870 GPU performed the best, achieving nearly 1.5 TeraFLOPs of performance. With a small die size, the R5870 achieves a high area-normalized performance almost comparable to the dedicated ASIC core (although still less energy-efficient). We were surprised that the GTX480 only achieved a 27% improvement in MMM over the GTX285 using the CUBLAS library (we expected a doubling of throughput); it possible that further software tuning is needed.

5.1. Deriving U-core Parameters

Recalling from Section 3.3, ϕ and μ are parameters that characterize a BCE-sized U-core, where ϕ is the relative power efficiency to a BCE core, and μ is the relative performance. To determine parameters of the BCE core, we treat the Core i7 as a fast core and derive BCE core parameters. To determine "r" for the Core i7, we base our estimates on an Intel Atom, which is a $26mm^2$ in-order processor in the 45nm process [65] (we subtract 10% for non-compute area). An r value of 2 roughly gives the equivalent size of a single Core i7 (about $193mm^2/(4$ cores)). Based on formulas that relate performance and power to area ($perf = \sqrt{r}$, $power = r^{\alpha/2}$), we derive $perf/mm^2$ and $perf/W$ for the BCE core (we assume $\alpha = 1.75$ [53]). These derived metrics are then used to compute ϕ and μ[1] for various U-cores in Table 5.

1. $\mu = \frac{x_{ucore}}{x_{corei7}\sqrt{r}}$ where $x_i = \frac{perf}{mm^2}$; $\phi = \frac{\mu \times e_{corei7}}{r^{(1-\alpha)/2} \times e_{ucore}}$ where $e_i = \frac{perf}{watt}$.

Table 4: Summary of results for MMM and BS.

		GFLOP/s	*(GFLOP/s)/mm²*	*GFLOP/J*
MMM	Core i7	96	0.50	1.14
	GTX285	425	2.40	6.78
	GTX480	541	1.28	3.52
	R5870	1491	5.95	9.87
	LX760	204	0.53	3.62
	ASIC	694	19.28	50.73
		Mopts/s	*(Mopts/s)/mm²*	*Mopts/J*
Black-Scholes	Core i7	487	2.52	4.88
	GTX285	10756	60.72	189
	GTX480	-	-	-
	R5870	-	-	-
	LX760	7800	20.26	138
	ASIC	25532	1719	642.5

Table 5: U-core parameters (ϕ = relative BCE power efficiency, μ = relative BCE performance).

		MMM	BS	FFT-64	FFT-1024	FFT-16384
GTX285	ϕ	0.74	0.57	0.59	0.63	0.89
	μ	3.41	17.0	2.42	2.88	3.75
GTX480	ϕ	0.77	-	0.39	0.47	0.66
	μ	1.83	-	1.56	2.20	2.83
R5870	ϕ	1.27	-	-	-	-
	μ	8.47	-	-	-	-
LX760	ϕ	0.31	0.26	0.29	0.29	0.37
	μ	0.75	5.68	2.81	2.02	3.02
ASIC	ϕ	0.79	4.75	5.34	4.96	6.38
	μ	27.4	482	733	489	689

6. Scaling Projections

In this section, we apply our model using U-core parameters from Table 5 to project the performance of single-chip heterogeneous computing devices based on the ITRS 2009 road map [5]. Figure 5 shows the long-term expected trends for pin count, Vdd, and gate capacitance. Assuming that clock frequencies do not increase, the reduction in power per transistor is expected to drop only by a factor of 5X over the next fifteen years (in contrast to the doubling of transistor density with each additional technology node). An even more

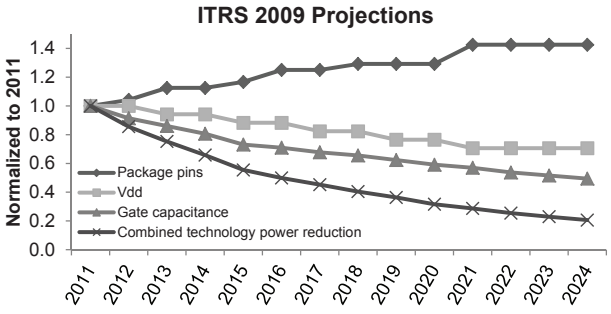

Figure 5: ITRS 2009 Scaling Projections (High-performance MPUs and ASICs [5]).

Table 6: Parameters assumed in technology scaling.

Year	**2011**	**2013**	**2016**	**2019**	**2022**
Technology node	40nm	32nm	22nm	16nm	11nm
Core die budget (mm^2)	432	432	432	432	432
Core power budget (W)	100	100	100	100	100
Bandwidth (GB/s)	180	198	234	234	252
Max area (BCE units)	19	37	75	149	298
Rel. pwr per transistor	1X	0.75X	0.5X	0.36X	0.25X
Rel. bandwidth	1X	1.1X	1.3X	1.3X	1.4X

serious challenge is the slowing of bandwidth, as evidenced by the gradual increases in projected pin counts (< 1.5X over fifteen years).

With these challenges in mind, we pose several critical questions: (1) under a bandwidth- and power-limited existence, would future single-chip multicores benefit significantly from having custom logic, FPGAs, and GPUs? (2) for the best possible performance, will custom logic always be the right answer under a power- and bandwidth-dominated regime? and (3) would our conclusions change if lowering total energy is the primary objective instead of maximizing performance?

To investigate these questions, we scale the performance of heterogeneous and non-heterogeneous multicores based on ITRS predictions. Table 6 summarizes various parameters and assumptions. We allow a maximum die budget of $576mm^2$ (e.g., Power7 [66]) and reserve 25% area for non-compute components (e.g., on-die memory controllers) [26]. A 100W power budget was selected for core- and cache-only components (not including power reserved for the non-compute components). We assume that clock frequencies do not scale further after 40nm.

For bandwidth, we assume that 180 GB/s can be optimistically reached in 2011 based on state-of-the-art high-end devices today (rounding up GTX480's 177GB/s). To estimate the compulsory bandwidth for FFT[2], we assume an input size of 1024, which gives $0.32\frac{bytes}{flop}$. For Black-Scholes, the compulsory bandwidth is $\frac{10bytes}{\text{option}}$. To compute compulsory bandwidth for MMM[3], we assume square matrix inputs blocked at $N = 128$, which gives $0.0313\frac{bytes}{flop}$. We exclude the ASIC-based MMM core from this constraint, since our design at 40nm can support $N \geq 2048$.

6.1. Overview of Results

The scaling projections for all U-core-based heterogeneous multicores (referred to as HET) and non-heterogeneous multicores (referred to as CMP) are shown starting with Figure 6. In each figure, we vary the f parameter, which indicates the fraction of time a particular kernel (e.g., FFT, MMM, BS) is being sped up by U-cores. To determine the optimal size of the sequential core, we sweep all values of r (sequential

2. FFT arithmetic intensity (32-bit) = $\frac{5N\log_2 N flops}{16\times Nbytes} = 0.3125\log_2 N$.

3. MMM arithmetic intensity (32-bit) = $\frac{2\times N^3 flops}{2\times 4N^2 bytes} = N/4$.

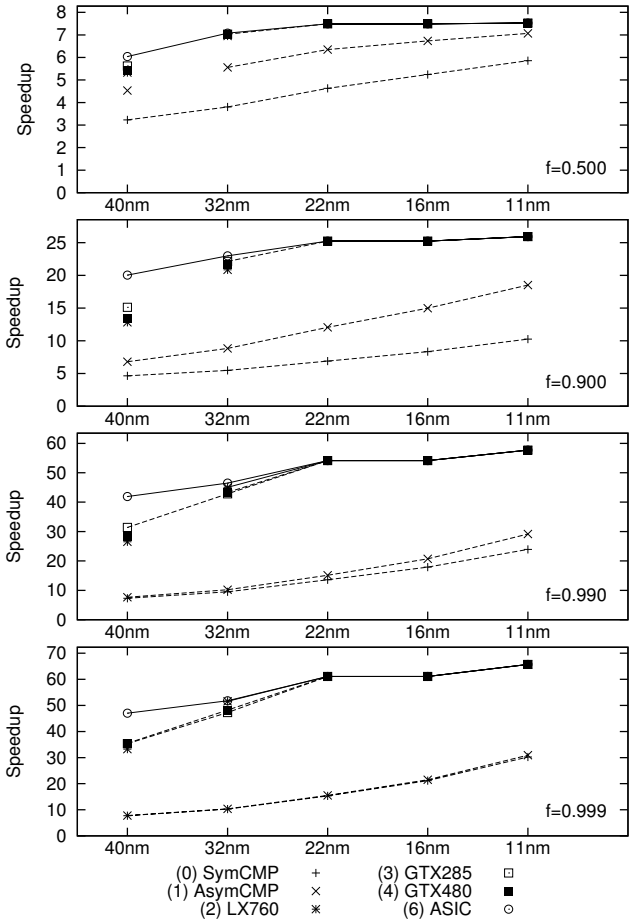

Figure 6: FFT-1024 projection.
(Dashed lines = power-limited, solid = bandwidth-limited)

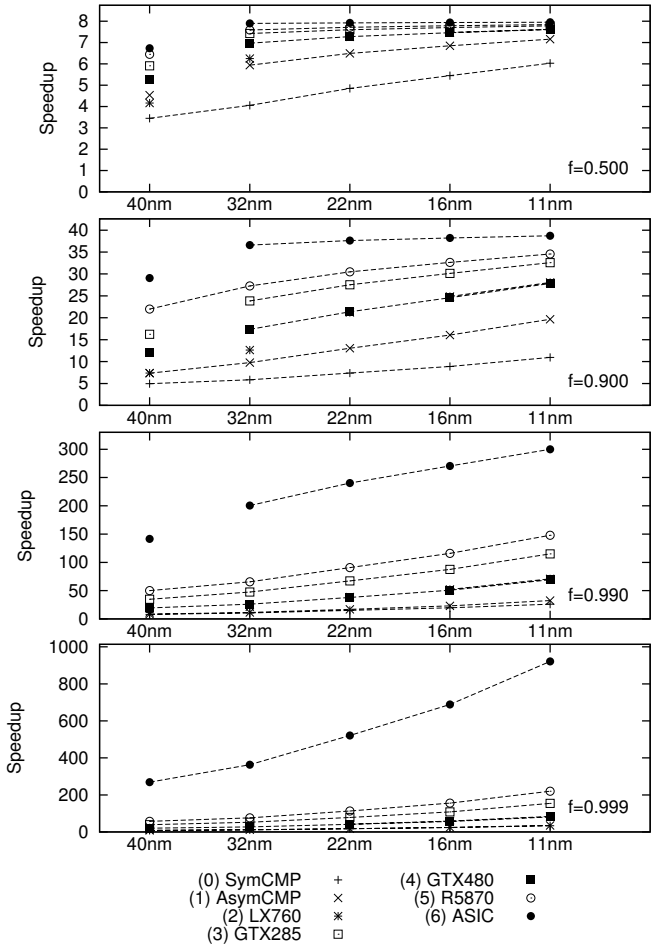

Figure 7: MMM projection.
(Dashed lines = power-limited, solid = bandwidth-limited)

core size) up to 16 for each particular design point and report the maximum speedup. All speedups are reported relative to 1 BCE core. To interpret the graphs, data points that are connected by **dashed lines** indicate designs that cannot be scaled any further due to power (i.e., the full area budget cannot be utilized). Data points that are connected by **solid lines** indicate bandwidth-limited designs. Finally, data points that **not connected by lines** are limited only by area.

FFT. Beginning with FFT-1024 in Figure 6, four different graphs are plotted with varying amounts of parallelism (f). At all values of f, the ASIC achieves the highest level of performance but cannot scale further due to bandwidth limitations. At $f = 0.5$, the lack of sufficient parallelism results in none of the HETs providing a significant performance gain over the CMPs—despite being more energy-efficient. The pronounced differences emerge when $f \geq 0.90$. Here, the HETs achieve more performance per unit power, which only becomes beneficial with sufficient parallelism. However, even with sufficient parallelism beyond $f \geq 0.99$, the lack of memory bandwidth limits any further speedup. It is interesting to note that the FPGA design reaches ASIC-like bandwidth-limited performance as early as 32nm—and similarly for the GPU designs, around 22nm and 16nm.

MMM. Results for MMM are shown in Figure 7. Unlike FFT, MMM possesses high arithmetic intensity and consumes relatively low memory bandwidth. For all values of f, the ASIC achieves the highest performance without becoming bandwidth-limited. However, unless $f \geq 0.999$, less-efficient approaches based on GPUs or FPGAs can still achieve speedups within a factor of two to five of the ASIC. It is also interesting to see how most designs are initially area-limited in 40nm and 32nm, but transition to becoming power-limited 22nm and after.

Black-Scholes. The results for Black-Scholes in Figure 8 are similar to FFT, where most HET designs quickly converge towards becoming bandwidth-limited. However, without sufficient parallelism ($f \leq 0.5$), even the conventional CMPs achieve speedups within a factor of two of the ASIC. Just as we observed in FFT, sufficient parallelism must exist

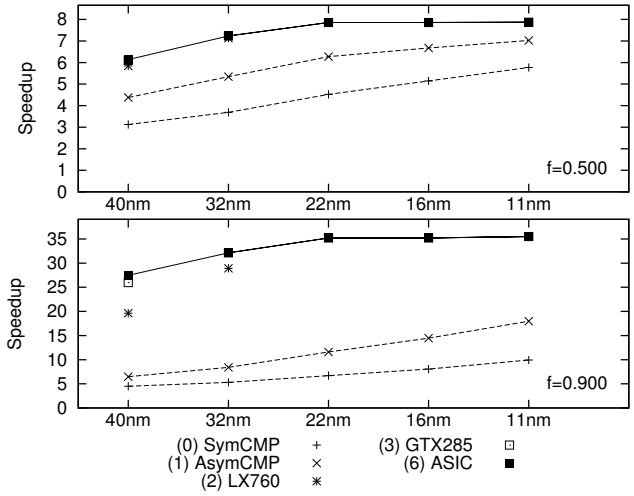

Figure 8: Black-Scholes projection.
(Dashed lines = power-limited, solid = bandwidth-limited)

($f \geq 0.9$) before the HETs can offer significant performance benefits over the CMPs. It is interesting to note again that the FPGAs and GPUs are also able to attain ASIC-like bandwidth-limited performance.

6.2. Projections under Alternative Scenarios

To investigate other scaling scenarios, we vary the following inputs to our model: (1) reduce initial bandwidth (in 40nm) to 90 GB/s, (2) increase bandwidth to 1 TB/s, (3) halve core-only area budget to $216mm^2$, (4) double power budget to $200W$, (5) decrease power to $10W$, and (6) increase core sequential power by setting α to 2.25. Below, we explain the rationale for each scenario and discuss results qualitatively.

Scenarios 1 and 2 (bandwidth). A starting bandwidth of 90 GB/s approximates a reduction in off-chip bandwidth costs (i.e., about half of what can be built into the highest-end designs in 40nm). We observe that FFT and BS on the FPGAs and GPUs converge even faster to the ASIC's performance in earlier technology nodes (i.e., becoming bandwidth-limited at 32nm). However, because of the low bandwidth ceiling in FFT, the CMPs also achieve within a factor of two of the ASICs at around 22nm (at any f). In BS, the large gap between HETs and CMPs still exists because the CMPs are unable to achieve close to bandwidth-limited performance.

To approximate disruptive technologies such as embedded DRAMs or 3D-stacked memories [67], our next scenario assumes a starting bandwidth of 1 TB/sec. For FFT, Figure 9 shows how most designs transition to becoming power-limited, with the ASIC still being bandwidth-limited from the start. At $f = 0.9$, all of the HETs provide further gains over the CMPs (around 2-3X speedup). However, the ASIC can only provide a significant speedup (about 2X) over the other HET approaches when $f \geq 0.999$.

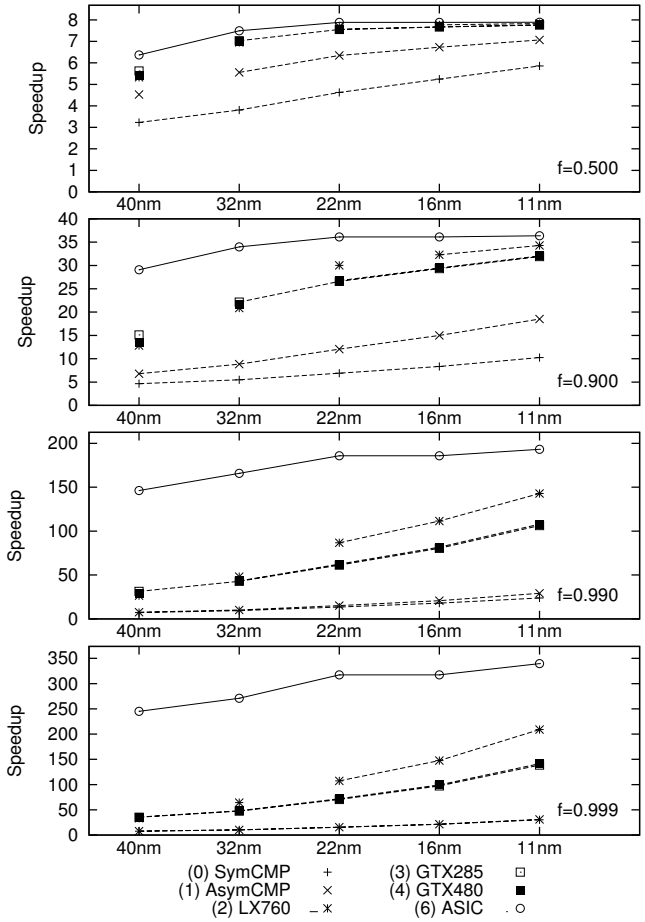

Figure 9: FFT-1024 projection given 1 TB/sec bandwidth.
(Dashed lines = power-limited, solid = bandwidth-limited)

Scenario 3 (area). Reducing the area budget approximates lower-cost manufacturing constraints (e.g., increasing yield). In this scenario, HETs in the earlier nodes (40nm, 32nm) become area-limited and provide less speedup over the CMPs. Surprisingly, in the later nodes ($\leq$22nm), most designs achieve similar performance to what was attained under the original area budget. The explanation is that most designs in the later nodes are limited by power to begin with, not because of insufficient area.

Scenarios 4 and 5 (power budget). Doubling the power budget to 200W approximates high-end systems with more capable cooling and power solutions. Under a higher power budget, the relative benefit of having energy-efficient HETs diminishes since the less efficient CMPs are able to close the gap in performance; this is especially noticeable in FFT when most HETs become bandwidth-limited. On the other hand, a 10W power budget approximates power-constrained devices such as laptops and mobiles. Here, we observe that only the ASIC-based HETs can ever approach bandwidth-limited performance, giving them an significant advantage

over the power-constrained HETs and CMPs.

Scenario 6 (serial power). Our last scenario approximates a sequential processor that consumes high power when scaled in performance (setting $\alpha = 2.25$). The trends are similar to the original projections with one notable exception. At low to moderate parallelism ($f \leq 0.9$), the speedups decrease significantly. Because the sequential processor's size is constrained by the serial power budget ($r^{\alpha/2} \leq P$), the optimal size of the core cannot be attained. Our results agree with Hill and Marty [11]—that is, improving sequential performance remains important. However, such performance cannot be achieved without accompanying improvements in sequential power efficiency.

6.3. Discussion

Our results show that the complex interplay between bandwidth, power, and parallelism has tremendous implications for various heterogeneous and non-heterogeneous computing approaches. To answer the first question—*would future multicores gain from having custom logic, FPGAs, and GPGPUs?*—our projections reveal that heterogeneous multicores based on U-cores do achieve significant performance gain over asymmetric and symmetric multicores—despite the scarcity of bandwidth. This performance is achieved with the increased power efficiency of U-cores and when sufficient parallelism exists ($f \geq 0.90$).

To answer the second question—*if custom logic is always the right answer*—a recurring theme we observed throughout was the impact of scarce bandwidth. Although custom logic provided the best performance under all circumstances, less efficient solutions such as GPUs and FPGAs were also able to reach bandwidth-limited performance for FFT and BS. Even in the case of MMM, which possessed the highest arithmetic intensity, the ASIC did not show significant benefits over the less efficient solutions unless $f > 0.99$.

Recalling the last question—*if energy should be prioritized over performance*—we briefly show an example that justifies why custom logic may be desirable even if parallelism is low or if memory bandwidth is a limitation. In Figure 10, the total energy consumed (normalized to BCE energy at 40nm) for various devices is plotted for MMM. Note that the energy decreases across generations are partially attributed to circuit improvements. At low levels of parallelism ($f = 0.5$), the opportunity to reduce the energy consumed is limited by the sequential core. However, at even moderate levels of parallelism ($f = 0.9 - 0.99$), the ASIC still achieves a significant reduction in energy relative to the other U-cores.

Although we do not quantify this in our study, custom logic and other low-power U-cores could potentially be used to reduce sequential power consumption or to efficiently improve sequential processing performance. First, if the goal is to achieve the same level of performance as a baseline

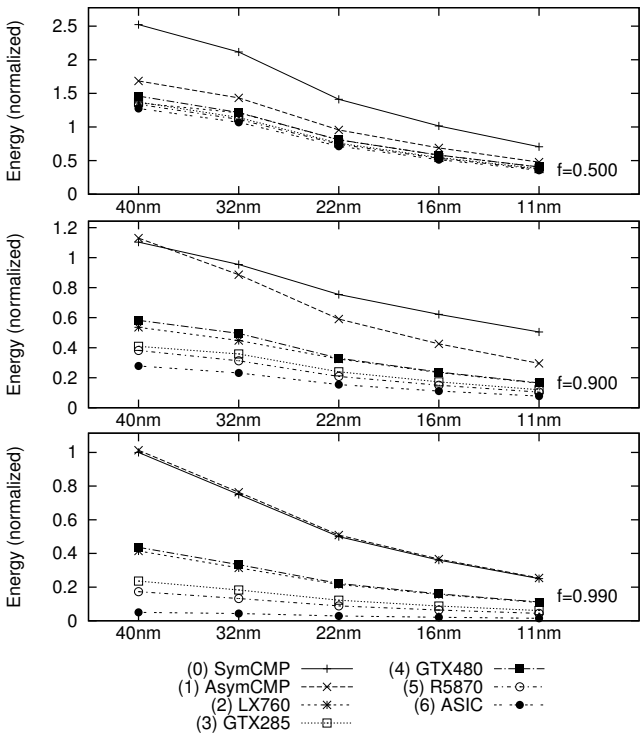

Figure 10: MMM Energy Projections (normalized to BCE).

system with processors, a U-core can be used to speed up parallel sections of an application while allowing the sequential processor to slow down with a significant reduction in power [1, 2]. Another previously proposed method [6] allows a power-hungry sequential processor to offload sections of serial code to custom logic. While custom logic can satisfy this role easily, reconfigurable fabrics such as FPGAs could also be used to fulfill this objective (e.g., [45, 46]). Finally, given the ability of custom logic and FPGA-like fabrics to extract multiple degrees of parallelism, U-cores based on these techniques may also be suitable for increasing sequential processor performance at a lower energy cost.

Relative U-core merits. Although our studies compared designs based on different U-cores, more compelling approaches would allow for mixing and matching. In our study, a simplifying assumption we made is that all U-cores and processors can expose identical levels of parallelism. As seen in practice (e.g., [22, 20, 21, 20]), GPGPUs have typically shown their strength for homogeneous data parallel tasks, while custom logic and FPGAs can be used to pipeline irregular data flows as well as data parallel tasks. Furthermore, the FPGA results we present should be treated conservatively given our selection of single-precision floating point applications (FPGAs lack dedicated FPUs.) With the abundance of area (but shortage of power) in the future, a compelling prospect is to fabricate different U-cores that are powered on-demand for suitable tasks. In similar spirit to

suggestions made by others such as Hardavellas [24] and Venkatesh et al. [6], specialized cores could be mixed and matched towards specific applications. For example, a high arithmetic intensity kernel such as MMM could be fabricated as custom logic alongside GPU- or FPGA-based U-cores used to accelerate bandwidth-limited kernels such as FFTs.

Model validity and concerns. By no means is our model infallible, nor our workloads universally representative. We remind that the objectives of this work are not to pinpoint exact design results, but to identify trends worthy of further investigations. Our selection of workloads, although limited, capture a spread of different characteristics. Our model, although simple, captures the salient constraints that any designer must be conscious of. The quality of our predictions will depend on two assumptions: (1) microarchitectures based on conventional CPUs, FPGAs, and GPGPUs do not change substantially in the future, and (2) ITRS predictions are correct. Obviously, the further we predict, the higher chance that some predictions will go askew. To check the quality of our predictions, we are pursuing further studies using older devices; data already collected from 55nm/65nm devices support the same conclusions from Section 6. Lastly, although programmability is an often raised concern, we deliberately excluded it from consideration. From our experience in developing high performance FFT libraries [59], the tuning of high-valued kernels on GPGPUs, FPGAs, or even CPUs will always require substantial investment.

7. Conclusions and Future Directions

To answer the question posed in the title of this paper—*whether the future should include custom logic, FPGAs, and GPGPUs*—the evidence from our measurements and from the projections made in our model show a strong case that the answer is **yes**. Our findings showed that: (1) sufficient parallelism must exist before U-cores offer significant performance gains, (2) off-chip bandwidth trends have a first-order effect on the relative merits of various U-cores (i.e., flexible U-cores can keep up with U-cores based on custom logic if bandwidth is a limitation), (3) even when bandwidth is not a concern, less efficient but more flexible U-cores based on GPUs and FPGAs are competitive with custom logic if parallelism is moderate to high ($90\% - 99\%$), and (4) U-cores, especially those based on custom logic, are more broadly useful if reducing energy or power is the primary goal.

There are numerous future directions worth pursuing. First, improved models and further case studies will be essential to sorting out the complex maze of choices a heterogeneous multicore designer must face. Models in the future should attempt to incorporate varying degrees of parallelism in an application, in order to capture how "suitable" certain types of U-cores might be under a given parallelism profile. Second, although significant research is underway for GPUs, little work has been done on integrating FPGAs with conventional multicores. Many issues require investigation such as how FPGAs should be integrated into existing software stacks and memory hierarchies. Finally, the most immediate challenge on the horizon is how to attack memory bandwidth limitations. While U-cores are effective at scaling the power wall, their long-term impact will increase even further if the bandwidth ceiling can be lifted through future innovations.

Acknowledgments

We thank the anonymous reviewers, members of CALCM, and John Davis for their feedback and suggestions. We thank Nikos Hardavellas for his influencing work on CMP scaling projections. Eric Chung is supported by a Microsoft Research Fellowship. We thank Spiral for providing FFT libraries and support. We thank Xilinx for their FPGA and tool donations.

References

[1] T. Mudge, "Power: A First-Class Architectural Design Constraint," *Computer*, vol. 34, no. 4, pp. 52–58, 2001.

[2] S. Borkar, "Getting Gigascale Chips: Challenges and Opportunities in Continuing Moore's Law," *Queue*, vol. 1, no. 7, pp. 26–33, 2003.

[3] M. Horowitz, "Scaling, Power and the Future of CMOS," in *Proceedings of the 20th International Conference on VLSI Design held jointly with 6th International Conference: Embedded Systems*. Washington, DC, USA: IEEE Computer Society, 2007, p. 23.

[4] B. M. Rogers *et al.*, "Scaling the Bandwidth Wall: Challenges in and Avenues for CMP Scaling," in *ISCA'09: Proceedings of the 36th Annual International Symposium on Computer Architecture*. New York, NY, USA: ACM, 2009, pp. 371–382.

[5] International Technology Roadmap for Semiconductors. http://www.itrs.net.

[6] G. Venkatesh *et al.*, "Conservation Cores: Reducing the Energy of Mature Computations," in *ASPLOS'10: Proceedings of the 15th International Conference on Architectural Support for Programming Languages and Operating Systems*. New York, NY, USA: ACM, 2010, pp. 205–218.

[7] S. Yehia *et al.*, "Reconciling Specialization and Flexibility Through Compound Circuits," in *HPCA'09: Proceedings of the 15th International Symposium on High Performance Computer Architecture*, 14-18 2009, pp. 277 –288.

[8] D. Luebke *et al.*, "GPGPU: General Purpose Computation on Graphics Hardware," in *SIGGRAPH '04: ACM SIGGRAPH 2004 Course Notes*. New York, NY, USA: ACM, 2004, p. 33.

[9] V. W. Lee *et al.*, "Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU," in *ISCA'10: Proceedings of the 37th Annual International Symposium on Computer Architecture*. New York, NY, USA: ACM, 2010, pp. 451–460.

[10] S. Hauck *et al.*, *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computing*. Morgan Kaufmann, 2008.

[11] M. D. Hill *et al.*, "Amdahl's Law in the Multicore Era," *Computer*, vol. 41, pp. 33–38, 2008.

[12] F. J. Pollack, "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies (keynote address)," in *MICRO'99: Proceedings of the 32nd Annual ACM/IEEE International Symposium on Microarchitecture*. Washington, DC, USA: IEEE Computer Society, 1999, p. 2.

[13] T. Y. Morad *et al.*, "Performance, Power Efficiency and Scalability of Asymmetric Cluster Chip Multiprocessors," *IEEE Comput. Archit. Lett.*, vol. 5, no. 1, p. 4, 2006.

[14] Intel Inc. (2009) Intel Microarchitecture, Codenamed Nehalem. http://www.intel.com/technology/architecture-silicon/nextgen.

[15] C. N. Keltcher *et al.*, "The AMD Opteron Processor for Multiprocessor Servers," *IEEE Micro*, vol. 23, no. 2, pp. 66–76, 2003.

[16] D. Wentzlaff *et al.*, "On-Chip Interconnection Architecture of the Tile Processor," *IEEE Micro*, vol. 27, no. 5, pp. 15–31, 2007.

[17] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *AFIPS'67 (Spring): Proceedings of the April 18-20, 1967, spring joint computer conference*. New York, NY, USA: ACM, 1967, pp. 483–485.

[18] W. J. Dally *et al.*, "Efficient Embedded Computing," *Computer*, vol. 41, no. 7, pp. 27–32, 2008.

[19] I. Kuon *et al.*, "Measuring the Gap Between FPGAs and ASICs," in *FPGA'06: Proceedings of the 2006 ACM/SIGDA 14th International Symposium on Field Programmable Gate Arrays*. New York, NY, USA: ACM, 2006, pp. 21–30.

[20] S. Che *et al.*, "Accelerating Compute-Intensive Applications with GPUs and FPGAs," in *SASP'08: Proceedings of the 2008 Symposium on Application Specific Processors*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 101–107.

[21] D. B. Thomas *et al.*, "A Comparison of CPUs, GPUs, FPGAs, and Massively Parallel Processor Arrays for Random Rumber Generation," in *FPGA'09: Proceeding of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. New York, NY, USA: ACM, 2009, pp. 63–72.

[22] N. Kapre *et al.*, "Accelerating SPICE Model-Evaluation using FPGAs," *Field-Programmable Custom Computing Machines, Annual IEEE Symposium on*, vol. 0, pp. 37–44, 2009.

[23] N. Hardavellas, "Chip-Multiprocessors for Server Workloads," Ph.D. dissertation, 2009, supervisors-Babak Falsafi and Anastasia Ailamaki.

[24] N. Hardavellas, "When Core Multiplicity Doesn't Add Up (Keynote)," in *International Symposium on Parallel and Distributed Computing*, Istanbul, Turkey, July 2010.

[25] S. Li *et al.*, "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," in *MICRO'09: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. New York, NY, USA: ACM, 2009, pp. 469–480.

[26] J. D. Davis *et al.*, "Maximizing CMP Throughput with Mediocre Cores," in *PACT'05: Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 51–62.

[27] R. Kumar *et al.*, "Heterogeneous Chip Multiprocessors," *Computer*, vol. 38, no. 11, pp. 32–38, 2005.

[28] E. Ipek *et al.*, "Core Fusion: Accommodating Software Diversity in Chip Multiprocessors," in *ISCA'07: Proceedings of the 34th Annual International Symposium on Computer Architecture*. New York, NY, USA: ACM, 2007, pp. 186–197.

[29] M. A. Suleman *et al.*, "Accelerating Critical Section Execution with Asymmetric Multicore Architectures," *IEEE Micro*, vol. 30, no. 1, pp. 60–70, 2010.

[30] Analog Devices. http://www.analog.com.

[31] Big Foot Networks, Inc. http://www.bigfootnetworks.com.

[32] Ageia. http://www.ageia.com.

[33] J. Kahle, "The Cell Processor Architecture," in *MICRO'05: Proceedings of the 38th Annual IEEE/ACM International Symposium on Microarchitecture*. Washington, DC, USA: IEEE Computer Society, 2005, p. 3.

[34] W. J. Dally *et al.*, "Merrimac: Supercomputing with Streams," in *SC'03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing*. Washington, DC, USA: IEEE Computer Society, 2003, p. 35.

[35] J. H. Ahn *et al.*, "Evaluating the Imagine Stream Architecture," in *ISCA'04: Proceedings of the 31st Annual International Symposium on Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 2004, p. 14.

[36] L. Seiler *et al.*, "Larrabee: A Many-core x86 Architecture for Visual Computing," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–15, 2008.

[37] AMD, Inc. http://www.fusion.amd.com.

[38] Tensilica, Inc. http://www.tensilica.com.

[39] S. C. Goldstein *et al.*, "PipeRench: A Coprocessor for Streaming Multimedia Acceleration," in *ISCA'99: Proceedings of the 26th Annual International Symposium on Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 1999, pp. 28–39.

[40] C. Ebeling *et al.*, "RaPiD - Reconfigurable Pipelined Datapath," in *FPL'96: Proceedings of the 6th International Workshop on Field-Programmable Logic, Smart Applications, New Paradigms and Compilers*. London, UK: Springer-Verlag, 1996, pp. 126–135.

[41] J. R. Hauser *et al.*, "Garp: a MIPS processor with a reconfigurable coprocessor," in *FCCM'97: Proceedings of the 5th IEEE Symposium on FPGA-Based Custom Computing Machines*. Washington, DC, USA: IEEE Computer Society, 1997, p. 12.

[42] Drew Wilson. (2000, Oct) Chameleon takes on FPGAs, ASICs. http://www.edn.com/article/CA50551.html?partner=enews.

[43] R. Razdan *et al.*, "A High-Performance Microarchitecture with Hardware-Programmable Functional Units," in *MICRO'94: Proceedings of the 27th Annual International Symposium on Microarchitecture*. New York, NY, USA: ACM, 1994, pp. 172–180.

[44] Z. A. Ye *et al.*, "CHIMAERA: A High-Performance Architecture with a Tightly-Coupled Reconfigurable Functional Unit," in *ISCA'00: Proceedings of the 27th Annual International Symposium on Computer Architecture*. New York, NY, USA: ACM, 2000, pp. 225–235.

[45] M. Mishra *et al.*, "Tartan: Evaluating Spatial Computation for Whole Program Execution," in *ASPLOS'06: Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*. New York, NY, USA: ACM, 2006, pp. 163–174.

[46] A. Putnam *et al.*, "Performance and Power of Cache-Based Reconfigurable Computing," in *ISCA'09: Proceedings of the 36th Annual International Symposium on Computer Architecture*. New York, NY, USA: ACM, 2009, pp. 395–405.

[47] J. L. Gustafson, "Reevaluating Amdahl's Law," *Commun. ACM*, vol. 31, no. 5, pp. 532–533, 1988.

[48] D. Moncrieff *et al.*, "Heterogeneous computing machines and Amdahl's law," *Parallel Computing*, vol. 22, no. 3, pp. 407 – 413, 1996. http://www.sciencedirect.com/science/article/B6V12-3VS5J8D-5/2/2363475aa2ac8e332e5fa124c9a9c3a1.

[49] J. M. Paul *et al.*, "Amdahl's Law Revisited for Single Chip Systems," *Int. J. Parallel Program.*, vol. 35, no. 2, pp. 101–123, 2007.

[50] S. Eyerman *et al.*, "Modeling Critical Cections in Amdahl's Law and its Implications for Multicore Design," in *ISCA'10: Proceedings of the 37th Annual International Symposium on Computer Architecture*. New York, NY, USA: ACM, 2010, pp. 362–370.

[51] D. H. Woo *et al.*, "Extending Amdahl's Law for Energy-Efficient Computing in the Many-Core Era," *Computer*, vol. 41, no. 12, pp. 24–31, 2008.

[52] S. Cho *et al.*, "On the Interplay of Parallelization, Program Performance, and Energy Consumption," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, pp. 342–353, 2010.

[53] E. Grochowski *et al.*, "Energy per Instruction Trends in Intel Microprocessors," in *Technology@Intel Magazine*, 2006.

[54] Intel, Inc. http://www.intel.com/products/processor/corei7/index.htm.

[55] Nvidia, Inc. http://www.nvidia.com.

[56] AMD Inc. http://www.amd.com.

[57] Xilinx, Inc. http://www.xilinx.com.

[58] Synopsys, Inc. http://www.synopsys.com.

[59] M. Puschel *et al.*, "SPIRAL: Code Generation for DSP Transforms," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 232 –275, feb. 2005.

[60] Nvidia. http://www.nvidia.com/object/cuda_home_new.html.

[61] C. Bienia *et al.*, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in *PACT'08: Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*. New York, NY, USA: ACM, 2008, pp. 72–81.

[62] P. A. Milder *et al.*, "Formal Datapath Representation and Manipulation for Implementing DSP Transforms," in *Proc. Design Automation Conference*, 2008, pp. 385–390.

[63] D. T. S. Thoziyoor *et al.*, "Cacti 4.0," HP Labs, Tech. Rep. HPL-2006-86, 2006.

[64] M. Schuette. (2008) Core i7 Power Plays. http://www.lostcircuits.com/mamb.

[65] Intel, Inc. http://www.ark.intel.com/Product.aspx?id=35635.

[66] R. Kalla *et al.*, "Power7: IBM's Next-Generation Server Processor," *IEEE Micro*, vol. 30, no. 2, pp. 7–15, 2010.

[67] L. A. Polka *et al.*, "Package Technology to Address the Memory Bandwidth Challenge for Tera-scale Computing," in *Intel Technology Journal, Volume 11, Issue 3*, 2007.