Title
Scaling the Utilization Wall: The Case for Massively Heterogeneous Multiprocessors

Authors
Ahn, Ikkjin
Goundling, Nathan
Sampson, John
et al.

Publication Date
2009-09-03

Peer reviewed

Scaling the Utilization Wall: The Case for Massively Heterogeneous Multiprocessors

Abstract

Multi-core processors have emerged as the leading solution to the power and scalability concerns that processor designers currently face. This transition addresses microarchitectural scalability issues, but it only delays the onset of the power scalability problem. Due to limitations on threshold voltage scaling, in a few process generations, processors will only be able to make use of a small fraction of a silicon die at full frequency at once. This "utilization wall" will prevent massively multi-core processors from effectively employing more than a small subset of cores at once. If we cannot utilize the full array of homogeneous cores, then the utility of building them comes into question.

This paper explores *massively heterogeneous CMPs*, an approach to processor design that can continue to scale performance in spite of the utilization wall. Such designs will comprise 10s to 100s to even 1000s of heterogeneous specialized processing elements (SPEs), ranging from small ASIC circuits to large speculative out-of-order general purpose processors. Massively heterogeneous CMPs combine these SPEs with an execution model that allows each part of a program to run on the SPE that can execute it most efficiently. Although the utilization wall dictates that massively heterogeneous CMPs (like all future processors) may use only a small fraction of the die at once, it uses that fraction very efficiently.

This paper explores the architectural challenges that arise in designing general-purpose massively heterogeneous CMPs. Our results demonstrate that massively heterogeneous systems can extend performance scaling by realizing large gains (up to $7\times$) in performance and efficiency relative to more modestly heterogeneous and homogeneous designs. The paper also presents an ASIC-based SPE case study that demonstrates the ability of such systems to provide large efficiency gains even for irregular integer applications.

1 Introduction

Power and microarchitectural scalability concerns have motivated a transition toward multi-core designs. Although multi-core designs address many microarchitectural scalability issues, they only defer the power scalability problems. As a result of threshold voltage scaling limitations, the percentage of transistors that a fixed-area design can switch at full frequency is dropping substantially with each generation of Moore's Law. In a few generations, processors will only be able to make use of a small fraction of the die at one time at full speed. We term this impending scaling problem *the utilization wall*.

The utilization wall will force designers onto one of three paths. First, they can build smaller (in area) designs so that power consumption and transistor budgets remain constant. Second, they can build successively lower-frequency designs that use all of the area. Third, they can design processors that use only a small fraction of the processor at once, but dynamically vary which fraction of the chip is active in order to maximize performance and efficiency. The first two scenarios would mark the end of the benefits that increased integration has historically delivered. Only the third scenario provides a means to continue scaling performance in the face of fixed power budgets and current limits on technology.

This paper proposes a new style of processor design called *massive heterogeneity* that explores this third scenario. Massively heterogeneous multiprocessors comprise a heterogeneous array of 10s to 100s to even 1000s of *specialized processing elements* (SPEs), a small subset of which will be active at any time. The SPEs in a massively heterogeneous system vary widely, ranging from specialized processors dedicated to particular loop nests to 8-way issue DSPs, graphics accelerators, and out-of-order superscalars. This work explores the challenges in exploiting the power-efficiency and performance that these SPEs provide to improve the performance of *general purpose* applications. We describe an execution model that allows processes to migrate between SPEs, allowing the set of active SPEs to vary over time to match changing program behavior. In addition, data from the literature and a preliminary hardware design study show that massively heterogeneous designs can improve power efficiency by up to 10× and 7× compared to conventional homogeneous designs and less aggressively heterogeneous approaches [47, 46]. Moreover, massively heterogeneous systems can realize these gains for a very wide range of applications, even irregular integer codes.

This paper makes the case for massively heterogeneous multiprocessors as a solution to the utilization wall and explores the architectural challenges that arise in designing massively heterogeneous massively heterogeneous systems with potentially 1000s of specialized cores. It also evaluates massively heterogeneous systems relative to other heterogeneous and homogeneous multiprocessors.

Specifically, the paper makes the following contributions:

- It describes the class of massively heterogeneous multiprocessors and demonstrates that they offer a promising approach to overcoming the utilization wall, allowing massively heterogeneous designs to profitably exploit up to 6.7× more transistors compared to less-aggressively heterogeneous designs.
- It demonstrates that although massively heterogeneous systems are composed largely of specialized processors, they can realize performance and efficiency gains even if workloads vary (i.e., they are general-purpose).
- It examines the trade-offs between different levels of SPE specialization and their impact on massively heterogeneous CMP performance.
- It proposes a baseline architecture and execution model to address engineering concerns in massively heterogeneous systems.

Highly heterogeneous processors raise interesting research questions about programming, operating systems, and software engineering. Indeed, recent efforts (e.g., [39, 49, 32]) have developed several promising approaches to address these issues. We do not address these issues in detail, but the general research area continues to grow, particularly due to the increasing prevalence of SoC chips in cell phones and other embedded systems.

The remainder of this paper is organized as follows. Section 2 examines the challenges and opportunities presented by highly heterogeneous processors and examines related work. Section 3 proposes an architecture and execution model for use in massively heterogeneous systems. Sections 4 and 5 present our methodology and the results of our evaluation of the massively heterogeneous approach to processor design. Section 6 concludes.

2 The utilization wall and heterogeneity

This section describes the technology-based case for massively heterogeneous multiprocessors. Then, we describe the relationship between our exploration of massively heterogeneous CMPs and previous work on heterogeneity.

2.1 The Utilization Wall

The percentage of transistors that a fixed-area design can switch at full frequency is decreasing exponentially with Moore's Law, due to problems in threshold voltage scaling. We call this phenomenon *the utilization wall*. This section describes the cause and structure of the utilization wall and shows that it is an inherent problem in MOSFET-based microprocessors.

Classical scaling dictates that device voltages scale in proportion to the dimension of the device. Unfortunately, further reducing the transistor threshold voltage, V_t, is no longer possible, because it increases leakage current exponentially. Making matters worse, fundamental limits from MOSFET physics suggest that improvements to transistor design or materials cannot offer sustainable repeated improvements in leakage[1]. Process designers have no choice but to stop scaling V_t with successive process generations.

Although fixing V_t can prevent leakage, it creates a chain-reaction that prevents the other aspects of the transistor from scaling. The next victim in the chain is the supply voltage, V_{dd}. Reducing V_{dd} with the feature size reduces per-transistor switching power, allowing the total power for the devices that can fit in a given area to remain constant. However, transistor speeds are determined by the ratio between V_{dd} and V_t, and performance and power-delay product degrade substantially as this ratio falls below the "sweet spot" of $V_{dd}/V_t = 3$.

Because of its negative impact on performance, reducing V_{dd} without lowering V_t is also not a long-term solution to the power problem. With fixed V_t and V_{dd}, the power consumption of a fixed sized chip, clocked at frequency proportional to transistor switching speed increases as S^2, where S is the scaling factor from one process generation to the next. Table 1 summarizes these scaling relationships and shows the difference between conventional "ideal" scaling (column 2) and the current leakage-limited regime (column 3).

The table shows that, given a fixed power budget, the percentage of transistors that can be switched at full frequency (i.e., the amount of silicon that can be in use) is going to decline drastically with each process generation, approximately as $1/S^2$. This ratio leads us to conclude that an exponentially diminishing fraction of silicon area will be usable at full frequency in future process technology.

This utilization wall has already begun to appear in current CMOS technology. Threshold and operating voltages are scaled only slightly with each generation, and recently announced quad-core designs run at lower clock rates than their dual-core predecessors in the same family. The introduction of new VLSI integration techniques like 3-D or large- or multi- die chips will only accentuate this problem.

2.2 Trade-offs in heterogeneous processors

A processor can provide heterogeneity in the instruction sets and programming models it supports and/or the microarchitectural implementations it provides. The amount of heterogeneity at each level substantially determines the challenges that the processor must overcome and the extent to which it can address the utilization wall.

[1]The leakage relationship is governed by the subthreshold slope, and lowering it would reduce leakage. Currently the subthreshold slope is around 90-110 mV/decade, and the *ideal* minimum value possible at room temperature is 60 mV/decade, which limits the potential benefits of improved MOSFET design.

Parameter	Classical Scaling	Leakage Limited	Parameter	Classical Scaling	Leakage Limited
power budget	1	1	I_{sat} (saturation current)	$1/S$	1
chip size	1	1	F (device frequency)	S	S
V_{dd} (supply voltage)	$1/S$	1	D (device/area)	S^2	S^2
V_t (threshold voltage)	$1/S$	1	p (device power)	$1/S^2$	1
t_{ox} (oxide thickness)	$1/S$	$1/S$	P (full die, full frequency power)	1	S^2
W, L (transistor dimensions)	$1/S$	$1/S$	U (utilization at fixed power)	1	$1/S^2$
C_{gate} (gate capacitance)	$1/S$	$1/S$			

Table 1: **Scaling trends** Application of CMOS scaling theory to current-day constraints, assuming fixed power and chip area. The classical scaling column assumes that V_t can be lowered arbitrarily. In the leakage limited case, constraints on V_t, necessary to prevent unmanageable leakage currents, hinder scaling.

We provide a two-dimensional taxonomy for heterogeneous multi-processors based on their architectural and microarchitectural heterogeneity. A four letter mnemonic, aAmM, identifies each processor class where a is the number of instruction sets or programming models the processor supports and m is the number of microarchitectures present. We use "*" to denote "more than one."

Below we describe the classes of processors covered by this taxonomy, previous related work, the challenges that each class presents, and existing approaches for addressing those challenges.

1A1M These are conventional, homogeneous chip multi-processors. They are commercially available and have received extensive attention from research and industry. Their uniformity prevents them from addressing the utilization wall.

1A*M Recent work on single-ISA, heterogeneous multiprocessors demonstrates that 1A$\star$M machines can provide both power savings [46, 41] and performance improvements [47]. Researchers have also examined scheduling issues [25, 48], and microarchitectural optimizations [40]. These machine provide some relief from the utilization wall by allowing some flexibility in the types of cores in use at any one time. However, the use a single ISA and programming model limits the level of heterogeneity they can exploit.

***A1M** Researchers have proposed several multi-ISA, single micro-architecture processors, but they cannot address the power scaling problems outlined in Section 2.1. SmartMemories [50] can emulate a range of different architectures using the same hardware. Chimaera [70], GARP [43], PRISC [56], and the work in [31] augment a general-purpose processor with reconfigurable logic to provide extensible ISAs.

***A*M** Multi-ISA, multi-microarchitecture machines are the most aggressively heterogeneous multiprocessors and they raise the most serious challenges. Our goal is to explore processors in this class that support many (> 10) instruction sets and microarchitectures and that target general-purpose computing.

A range of commercial $\star$A$\star$M processors and research prototypes are available or have been proposed. These include Sony's Cell [44], IRAM [54], and the hardware used in [39]. These machines augment a general purpose core with vector co-processors to accelerate multimedia applications. These machines incorporate just two types of core (i.e., they are 2A2M machines).

In addition to the complexity that diverse hardware presents, $\star$A$\star$M machines also require significant changes in the operating system, compiler, and, depending on the amount of compiler support, applications. Researchers have already begun

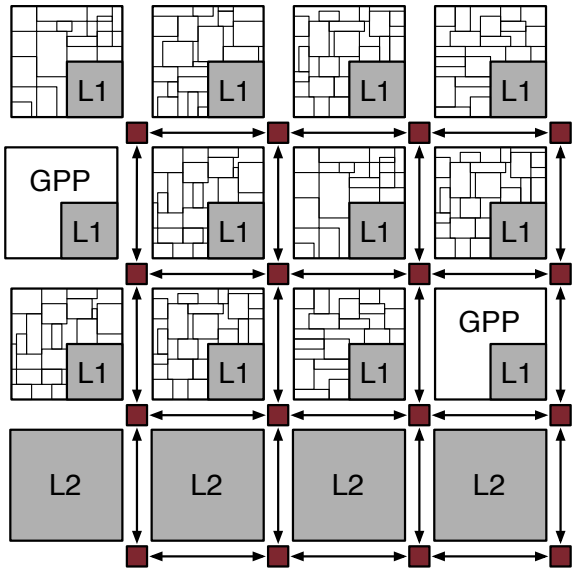


Figure 1: **A massively heterogeneous CMP** The high-level design of a massively heterogeneous multiprocessor. SPE complexes attach to a grid-based interconnect which provides access to general-purpose processors (GPP) and an on-chip L2. A single SPE complex comprises many SPEs of varying generality and complexity. Only one SPE in the complex may be active at a time.

to address the `*A*M` programming problem. EXOCHI [39] and Merge [49] provide general frameworks for programming heterogeneous systems. They allow programmers to target multiple SPEs in the same program using a uniform set of abstractions. VEAL [32] provides a light-weight dynamic translation layer that allows computation targetted toward any potential accelerators to be expressed in a single baseline instruction set.

3 Massively heterogeneous architectures

Organizing 100s or 1000s of heterogeneous SPEs into a general-purpose processor raises both software and hardware issues. Below, we briefly discuss the execution model for massively heterogeneous CMPs and then discuss the high-level organization, interconnect, and memory architecture of massively heterogeneous systems.

3.1 Execution model

A program executing on a massively heterogeneous CMP migrates between SPEs as its behavior changes. The tool-chain and run-time environment combine to create the mapping between different sections of the program and the available SPEs. Programmers and compilers will use a range of tools (e.g. library interfaces, code similarity measurement technology, and performance prediction models) to identify which code segments will run most efficiently on different SPEs. The operating system or run-time schedules code onto the available SPEs taking into account the physical location of the SPEs and other applications competing for the same SPEs. The mapping of program to SPEs can also change at runtime to account for observed changes in program behavior. Constructing the software infrastructure to support these tasks is challenging, but several proposals [39, 49, 32] have already begun exploring this area and we expect it to be a fruitful area of research.

3.2 Hardware organization

Figure 1 depicts a small massively heterogeneous CMP comprised of twelve *SPE complexes* and four banks of shared L2 cache connected by a grid-based on-chip interconnect. Together the complexes, cache banks, and network resemble recently proposed

tiled processors such as WaveScalar [63], Raw [65], or TRIPS [57]. Instead of uniform tiles, however, the complexes (i.e., the "tiles") in a massively heterogeneous system contain many SPEs. The mix of SPEs in each complex is different. Broadly, the SPEs fall into two categories: *Single-purpose* and *Multi-purpose* Specialists.

Single-purpose specialists are ASIC-like pieces of hardware that target particular power-critical algorithms, loops or applications, based on their prevalence within pre-existing workloads. Examples include compression, encryption (e.g., a SSL accelerator in a web server-oriented), or back-end rasterizers. We expect single-purpose specialists will be automatically synthesized from languages like C and C++, and have demonstrated that this is feasible in many cases (see Section 5.5).

Multi-purpose specialists, on the other hand, are intended to function across a wider variety of code-bodies (including those not available at design-time), but are not necessarily Turing complete. Presently available multi-purpose specialists include general purpose processors, graphics card pixel processors [52, 24], vector-accelerators, and molecular dynamics simulation accelerators [35]. We expect that, as the utilization wall becomes more acute, we will see an increasing diversity in both single-purpose and multi-purpose specialists.

Massively heterogeneous systems organize SPEs into complexes based on related functions to allow pipeline-sequential style communication. SPEs likely to be used in sequence are placed in the same complex, so that they can efficiently communicate through the local L1 cache.

3.3 Interconnect

Two unique characteristics of massively heterogeneous systems combine to influence the design of the on-chip interconnect. First, because of the utilization wall, only a fraction of the SPEs will be active at any time. This means that while the number of end-points in the network will be high, the amount of simultaneous communication per end-point will be low. Second, many SPEs are much smaller than even the smallest wormhole router, so conventional on-chip interconnects are not suitable – the chip would be nothing but routers.

Massively heterogeneous systems satisfy these constraints with a hierarchical interconnect built to accommodate the execution model. Within a complex, we introduce the constraint that only one SPE at a time can be active. This allows us to pair down the intra-complex interconnect by eliminating arbitration, flow-control and buffering logic, which is responsible for much of the power and area. It also helps address power density concerns.

Massively heterogeneous systems connect SPE complexes with a conventional mesh-style wormhole-routed interconnection network, similar to those found in tiled architectures like Raw [65]. SPEs in different complexes are permitted to operate simultaneously, managed by the operating system.

3.4 Memory

SPEs communicate using coherent shared memory. To facilitate speedy exchange and forwarding of data between SPEs as program execution migrates, SPE complexes contain a coherent L1 cache shared among the SPEs in that complex. Since only a single SPE may be active in the complex at one time, there is no contention between SPEs for access and the cache does not require multiple ports.

A conventional coherence protocol keeps the L1s synchronized with each other, the L2 cache, and main memory. In addition, SPEs may have their own L0 caches or local memories customized for the SPE's cache behavior. Our execution model for massively heterogeneous systems requires that programs appear to execute on a conventional processor. To provide

this appearance while ensuring maximum freedom in SPE design, massively heterogeneous systems adopt release consistency to coordinate communication and synchronization.

4 Exploring massively heterogeneous designs

The effectiveness of massively heterogeneous multiprocessors hinges on their ability to combine many SPEs into a single processing system that can efficiently execute general-purpose programs. This section presents our approach to resolving two challenges that arise in massively heterogeneous designs. First, we analyze the key trade-off between SPE generality and efficiency by examining the performance and power-efficiency that existing general-purpose, partially-specialized, and fully-specialized SPEs offer.

Next, we present a genetic algorithm-based toolchain for selecting a set of SPEs to include in a massively heterogeneous design and then evaluating that design. By automating the high-level design of massively heterogeneous systems, we can quickly explore a large space of massively heterogeneous designs. We use this tool-chain to do so in Section 5.

4.1 Understanding SPE performance trade-offs

Ultimately, we expect massively heterogeneous multiprocessors to include 100s or 1000s of different SPEs. Performance data for such a large variety of processors is not currently available. However, the literature does provide performance and power data for several dozen SPEs that span a wide range in the application areas they target and the degrees of specialization they provide. We collected performance, area, and power consumption data for many of these processors.

Making direct comparisons using the published data is challenging because the designers use different implementation technologies and different benchmarks to measure performance. We overcame this obstacle by focusing on 23 SPE designs from the literature and 25 applications (Table 2) that are either well-known or drawn from benchmark suites such as Spec2000 [60], EEMBC [38], or MineBench [55]. We chose this set of applications and SPEs to maximize the number of direct comparisons possible while still including a range of hardware designs and workloads.

We used a 90 nm Pentium 4 processor scaled to 22 nm technology as a performance "Rosetta stone." We collected P4 performance data for all the applications and used those data to normalize the performance of other SPEs. To account for different process technologies, we scaled each design to 22 nm using the data in Table 3.

For tiled machines, we combined data from WaveScalar [64] and Raw [65]. We partitioned the workloads into three groups (the "Cat." column of Table 2 (bottom)): Embarrassingly parallel (P), strictly serial (S), or mixed (M). Using the available data, we computed the performance of different tiled array sizes by assuming that performance scales by the number of tiles (P), is constant (S), or scales by the square root of the number of tiles (M). We counted each "domain" in the WaveScalar processor as one tile.

The resulting data are not perfect, but they do highlight the *range* of performance and power efficiency numbers we can expect from future SPEs. The table shows, for instance, that performance variation ranges from less than 2× for SpecINT workloads to over 9,600× for convolution encoding. Likewise, the power consumption varies from 0.23W for the i.MX21 to 85W for a GPU. For ASICs, the power consumption (data not shown) is as low as 7mW (for the convolution encoder). The area required for the SPEs varies just as widely (85.7mm^2 for the GPU, 7.5e-5mm^2 for the convolution encoder).

The data also provide insight into the trade-off between generality and power consumption. This trade-off is critical in

SPE	Description	Power(W)
General Purpose		
P4[1]	Pentium 4 Xeon	61.7
MC7448[2]	Freescale MC7448 "PowerPC G4"	15.44
750-CL[3]	2-issue, in order, IBM PowerPC	4.12
i.MX21[4]	Freescale embedded Arm Core	0.23
Tiled		
Tiled 1-16[64, 65]	WaveScalar, Raw	1.86 - 30
Specialized		
Scale[45]	The Scale processor	
Cell[30]	Sony Cell processor	22.05
CSX600[34]	ClearSpeed FP accelerator	6.41
DSP [5, 33]	TI-TMS320C6416-720 and Bison	1.1
GPU[6]	NVidia 8800GTX	85.73

ASICs	Power(W)
Conv. Encoder[66]	2e-03
8b/10b Block Encoder[66]	6e-03
Conv. Encoder (EEMBC)	7e-03
Autocorrelation (EEMBC)	7.5e-02
8x8 DCT	0.35
FFT (EEMBC)	0.35
1024-point Complex-FFT	0.5
GROMACS	3.5
Raytracing	3.5
Scan	0.5
Viterbi	0.3
Bio. Sequence Matching	8e-02

Workload	Cat.	Perf. Range		Workload	Cat.	Perf. Range		Workload	Cat.	Perf. Range	
		Min.	Max.			Min.	Max.			Min.	Max.
EEMBC				**SPEC 2000**				**Others**			
Text	S	0.07	1	Equake	P	0.62	1	DCT 8 X 8	P	1	2.71
Ptr. Chase	S	0.04	1	MCF	S	1	2	1024-point FFT	M	1	117.6
Ospf	M	0.08	2.23	Art	M	1	1.1	Scan	P	1	14
Conv. Enc.	P	0.003	28.93	**Others (cont.)**				GROMACS	P	1	105
Auto Corr.	P	0.01	2.32	8b/10b block enc.	P	1	60.56	Raytracing	P	1	8.41
Fixed pt. Bit Alloc.	M	0.05	2.59	802.11A Conv. Enc.	P	0.53	20.97	CSLC	P	1	33
Vitrebi	P	0.08	2.75	SGEMM	P	1	21	CityBusy	M	1	20
FFT	M	0.06	4.65	DGEMM	P	1	7.06	Cloth Sim.	M	1	20
				SpMM	P	0.05	6.4	Bio Sequen match	P	1	94

Table 2: **Kernels, applications, and SPEs** We gathered performance data for 23 processors, co-processors, and ASICs (top) and 25 applications and kernels (bottom). The "Cat." column in the workload table describes the parallizability of the application: S=Serial (non-parallizable), P=Parallel, M=mixed. The "Perf. range" columns measure performance relative to a Pentium 4. Data from [62, 59, 33, 45, 7, 67, 8, 34, 53, 9, 10, 11, 58, 37, 29, 36, 23, 12, 61, 66, 42, 22, 13, 14, 15, 28, 51, 26, 16, 69, 17, 67, 27, 58, 18, 19, 20]. Some values are estimates.

Process (nm)	250	180	130	90	65	45	32	22
V_{dd}	2.0	1.75	1.5	1.4	1.35	1.3	1.25	1.2
A	129	67	35	17	8.7	4.2	2.1	1.0
$1/T_{FO4}$	0.09	0.12	0.17	0.24	.34	.49	.69	1.0

Table 3: **Scaling methodology** Our scaling methodology extends the model in Table 1 by incorporating V_{dd} data from Intel product lines. To account for on-going transistor improvements such as strained silicon, high K, FinFET and improved doping profiles, we allow the process to maintain consistent I_{on}/I_{off} ratios (and thus constant leakage levels, which are set as a parameter by Intel process engineers) and linear improvements in FO4 delay even as V_{dd} is lowered. We assume performance scales with FO4.

massively heterogeneous designs because they strive for power-efficient execution of general-purpose workloads. Our data show that moving from single-purpose SPEs to multi-purpose (but still specialized) SPEs leads to a large increase in power consumption. However, increasing generality further leads to less dramatic reductions in efficiency and performance. For instance, for matrix multiplication a fully-specialized ASIC is $\sim 50\times$ more efficient than the CSX600 (a dedicated 96-way FP co-processor), but the CSX600 is only $6\times$ more power efficient than a fully-general tiled processor. In addition, the CSX600 outperforms the tiled machine on all the applications in its target domain for which we have data. We observe this trend for other applications as well: The gap in efficiency between single-purpose and substantially multi-purpose designs is much larger than between those designs and fully general-purpose designs. However, the designers of massively heterogeneous systems must understand the "shape" of the trade-off between generality and power efficiency in more detail. In particular, it would be useful to understand whether very modest levels of generality extract the same cost in terms of power efficiency.

4.2 Tool-chain

The design space of massively heterogeneous multiprocessors is much larger than for conventional multiprocessors. To explore the space of massively heterogeneous designs quickly, we have developed a tool-chain for generating and evaluating massively heterogeneous multiprocessor designs given a workload and set of available SPEs. By varying the inputs to the tool-chain we can generate massively heterogeneous multiprocessors crafted from different sets of SPEs and targeted at different applications and area budgets. We use this tool-chain in Section 5 to compare the massively heterogeneous approach to other approaches and to understand the impact of SPE and application mix on the effectiveness of massively heterogeneous systems.

The tool-chain takes four inputs:

1. **Workload description:** A list of kernels that make up the workload along with the fraction of time the workload spends executing each kernel.
2. **Performance data:** A table of performance, power, and area data for each SPE running each kernel.
3. **Area and power constraints:** Limits on the amount of area and power the processor can consume.
4. **Communication patterns:** A matrix of communication requirements between kernels.

The first stage of the tool-chain uses a genetic algorithm (GA) to select the set of SPEs (a configuration for a given massively heterogeneous system) that is best suited to the workload. The GA avoids the need to exhaustively search the space of massively heterogeneous CMP configurations. While the GA does not guarantee an optimal design, it achieves the same result as exhaustive search for small problem cases with small diversity in core types. To verify bigger problem cases, for each power budget, we defined a heuristic upper-bound using performance of a configuration with unconstrained area budget. With enough area budget, the GA finds configurations with less than 5% performance degradation.

We use the data in Section 4.1 as input to the GA. When data for an SPE/application pair are not available, we estimate it based on the performance of similar SPEs and similar applications. Our data show that these estimates have little impact on our results: Randomly perturbing the estimated values by up to 10% results in smaller than 2% changes in our performance measurements, and all the trends remain the same.

To evaluate a configuration, the GA combines measurements of both throughput and latency-oriented performance. To measure throughput, the GA generates a random sequence of 400 jobs (kernel instances) drawn uniformly at random from our benchmark suite. The scheduler attempts to maximize throughput while staying within the power budget.

To evaluate latency performance, the GA models executing the kernels in the workload serially on the highest performance SPE for each kernel. The GA combines the latency and throughput measures to determine the overall quality of a configuration.

We control the trade-off between latency and throughput with a parameter called α. For parallel-centric designs, $\alpha = 0$. For serial-centric designs, $\alpha = 1$. The α parameter corresponds to the trade-off in all multi-processors between throughput and latency. In conventional CMPs, a large value of α would lead to a few, powerful cores (e.g., the Power6) while a small value would lead to a larger number of simple cores (e.g., Sun's Niagara). Unless otherwise stated, we provide data for $\alpha = 0.33$.

If two configurations achieve the same level of performance, the algorithm favors the smaller one, so the final processor might not utilize its entire area budget.

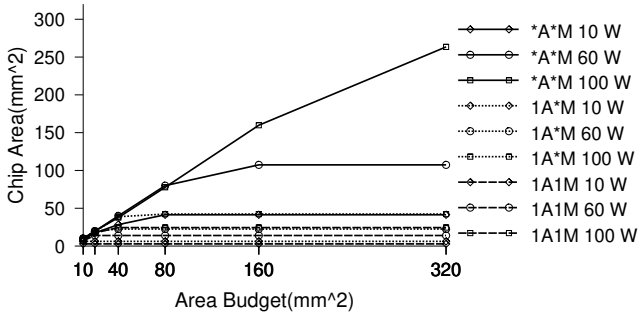


Figure 2: **Putting silicon to good use** Extensive heterogeneity allows massively heterogeneous designs to profitably translate more silicon into performance. For `1A1M` and `1A*M` designs, processors larger than 24mm^2 or 42mm^2 do not offer performance gains. Massively heterogeneous designs realize gains up to 260mm^2.

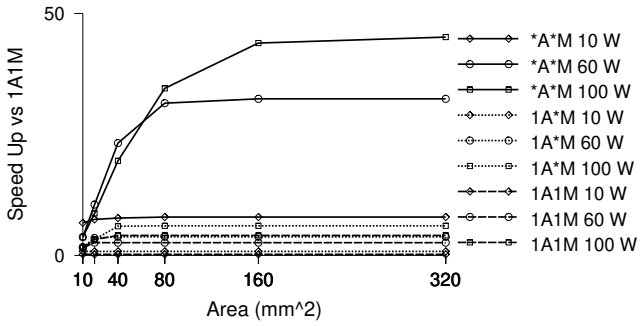


Figure 3: **Power, area, and performance for three different processor classes** Performance for `1A1M`, `1A*M`, and `*A*M` processors all increase with increasing die size. However, massive heterogeneity (`*A*M`) provides much larger gains: up to 7.5× and 11.3× relative to `1A1M` and `1A*M`, respectively.

We also use our toolchain to generate the `1A1M` and `1A*M` machines we use in our comparisons in the next section. To generate `1A1M` configurations we limit the GA to choosing the best single general-purpose processor (this amounts to an exhaustive search). For `1A*M`, we restrict the choice of SPEs to the four general-purpose processors in Table 2.

5 Results

This section uses our GA-based toolchain to quantify the differences between massively heterogeneous multiprocessors and other types of heterogeneous processors. Section 5.1 measures massively heterogeneous multiprocessors' efficiency and performance improvements and investigates the trade-offs between area, power, and performance. It also compares massively heterogeneous CMP performance to `1A1M` (conventional, homogeneous CMP) and `1A*M` (single-ISA heterogeneous) designs. Section 5.2 demonstrates the ability of massively heterogeneous designs to provide benefits on a range of workloads. Section 5.3 measures the impact of different degrees of SPE specialization on massively heterogeneous designs. Section 5.4 examines trade-offs between parallel and sequential performance in massively heterogeneous multiprocessors. Finally, Section 5.5 describes a case study that argues that massive heterogeneity techniques are applicable to integer workloads and that hardware complexity in massively heterogeneous processors is tractable.

5.1 Efficiency and performance

Figure 2 demonstrates massively heterogeneous systems' ability to overcome the utilization wall and translate large transistor budgets into performance gains, even in the face of strict power budgets. For instance, at 100W, `1A1M` and `1A*M` designs larger than 24mm^2 or 42mm^2, respectively, do not offer additional performance (due to the utilization wall). Massively heterogeneous designs can extract performance and efficiency gains from 6-10× more silicon. At 10W, the differences are even larger: massively heterogeneous designs can effectively utilize up to 14× more silicon. The diversity in the available SPEs (i.e., those in Table 2) limits the amount of area massively heterogeneous designs can utilize. For larger selections of SPEs, we would expect to see even greater gains.

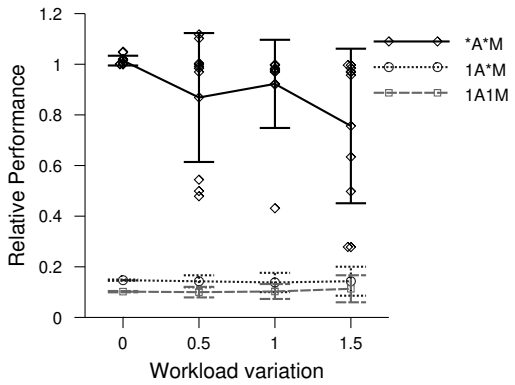


Figure 4: **Workload variation and massively heterogeneous system performance**The performance of a particular massively heterogeneous multiprocessor degrades gracefully for workloads that differ widely from the workload the GA designed it for. The variation in performance grows, but even in the worst case, the massively heterogeneous machine out-performs less heterogeneous designs.

Using specialized SPEs also translates into increased parallelism. A 100W, 260mm^2 massively heterogeneous design uses, on average, 34.4 SPEs simultaneously, compared to just 6.9 and 13.8 for the `1A1M` and `1A*M` machines (data not shown). `1A1M` designs fare especially poorly in power-limited regimes because they cannot put additional area to any good use. Indeed, depending on the power budget, `1A1M` scalability stalls out at between 3 and 7 cores on one chip. After those points, further increases in area (and transistor budgets) are useless.

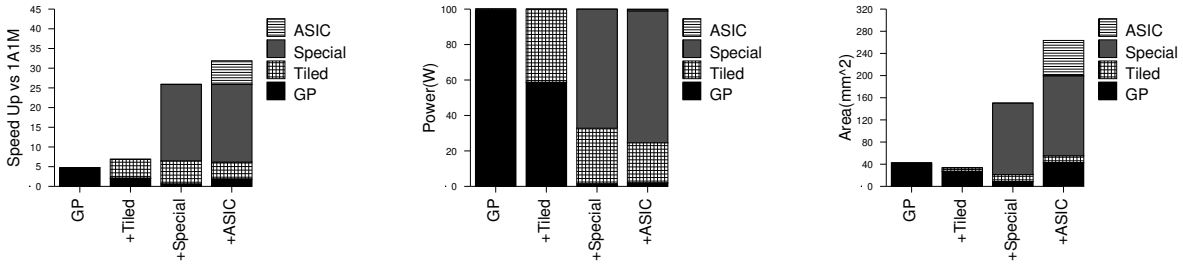
Massively heterogeneous designs can also translate increased transistor counts into performance gains more effectively than other types of heterogeneous processors. Figure 3 explores this effect in detail. It shows massively heterogeneous, `1A1M`, and `1A*M` performance for a range of power and area budgets. Each line represents a different power budget with performance normalized to the 10mm^2, 10W, `1A1M` design comprised of general-purpose processors.

For all three architecture types, performance increases with power and area budgets. The data show that `1A*M` designs outperform `1A1M` designs for a given area and power budget. These results corroborate previous results on the single-ISA heterogeneous systems [47, 46].

Massively heterogeneous systems out-perform both of the other classes by a significant margin. At the low end, at 100W, 10mm^2 massively heterogeneous design outperforms the corresponding `1A1M` and `1A*M` designs by 2.5× and 2.2×, respectively. Incorporating highly specialized processors allows massively heterogeneous CMPs to widen the margin as the chip size grows: For 100W, 320mm^2 designs, the massively heterogeneous approach achieves 11.3× the performance of the `1A1M` and 7.5× the performance of the `1A*M`.

5.2 General purpose performance

Massively heterogeneous multiprocessors must provide speedups on a very wide range of workloads. The toolchain we outlined in Section 4.2 generates architectures based on a particular workload model given as input. Figure 4 measures the sensitivity of massively heterogeneous system performance to variations in workload. The figure shows the change in the performance for a particular massively heterogeneous multiprocessor, A, as the workload characteristics diverge from the workload the GA used to design the processor. For each point, we create a workload according to a skewed probability distribution over the kernels in Table 2. The X-axis measures the degree of skew. The maximum possible value is 2. We use the GA to generate a customized



(a) (b) (c)

Figure 5: **Diversity's impact** Different core types contribute differently to different aspects of massively heterogeneous designs. Graphs (a), (b), and (c) compare the contributions of each core type to overall processor performance, power consumption, and area. The "GP" corresponds to `1A*M` machines. In all cases, the power budget is 100W and the area budget is 320mm^2.

massively heterogeneous multiprocessor for that workload. The Y-axis measures A's performance relative to that processor. The lines and error bars show the average performance and standard deviation for each level of skew.

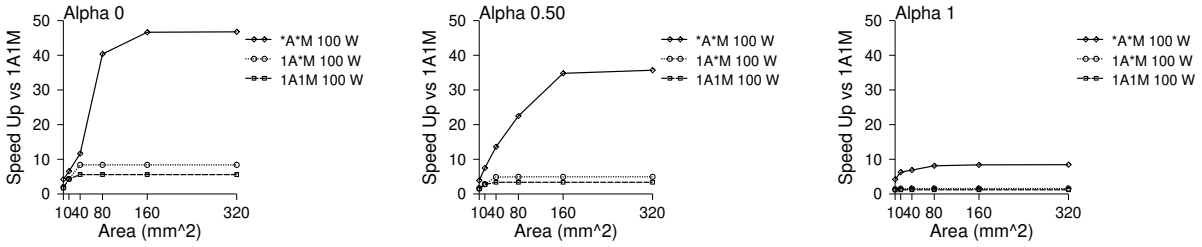
The data show that this massively heterogeneous design achieves good performance on a range of workloads beyond the training workload used by the genetic algorithm. The data also show that although variation in performance is smaller for `1A1M` and `1A*M` designs, the maximum performance is lower. Indeed, even in the worst case we measured (workload variation = 1.5), the massively heterogeneous multiprocessor outperformed the other designs by 2.4-13×.

The data illustrate what we believe will be a common theme in massively heterogeneous system performance: The variation in performance for massively heterogeneous multiprocessors across applications will be much larger than for `1A*M` and `1A1M` machines. Two key goals in designing massively heterogeneous systems will be to ensure that 1) any program can run at a reasonable speed on the design and 2) most programs can benefit from the specialized cores available. We can address the first goal by providing a capable selection of general purpose processors. Attaining the second goal will require shrewdly crafting the specialized co-processors so that they are useful on a range of applications.

5.3 The impact of diversity

The SPEs listed in Table 2 represent a wide range of processor types that represent very different trade-offs in performance, area, and power. Likewise, the contributions that these cores make to massively heterogeneous multiprocessors' overall characteristics vary widely.

Figure 5 demonstrates these differences by dividing the SPEs into four groups, general purpose (GP), tiled, specialized (special), and application specific (ASIC), and measuring the effects of adding each group in turn. For instance, for a massively heterogeneous CMP with a 100W power budget and 320mm^2 area budget, ASICs provide 19% of the performance, but consume just 1% of the power and 24% of the area. In contrast, the specialized processors provide 62% of the performance but consume 74% the power and 55% of the area. ASICs provide greater gains in performance and efficiency than specialized processors, but at the cost of reduced flexibility. Other data (not shown) demonstrates that more specialized SPEs become more valuable as power budgets shrink. For the 320mm^2 case discussed above, ASICs provide fully 50% of performance for processors with 10W power budgets.



(a) (b) (c)

Figure 6: **Trading off latency and throughput in massively heterogeneous systems** Multiprocessor performance for parallel (a), hybrid (b), and serial (c) workloads. Massive heterogeneity provides the largest gains for parallel workloads ($\alpha = 0$), but heterogeneity is still beneficial for serial workloads ($\alpha = 1$)

5.4 Latency and bandwidth trade-offs

All multiprocessors make a trade-off between optimizing for latency vs. bandwidth. In `1A1M` machines the key variable is the single-threaded performance of the individual, homogeneous cores (i.e., simple cores for bandwidth-oriented machines, and larger more aggressive cores for latency-oriented machines). For massively heterogeneous and `1A*M` machines, the choices are more complex since the *mix* of cores comes into play as well. As described in Section 4, our GA uses a weighting factor, α, to trade off between parallel- and serial-centric designs.

Figure 6 uses α to show how massively heterogeneous multiprocessors perform under parallel and serial-centric workloads. Graphs (a), (b), and (c) in the figure show performance for purely serial-centric ($\alpha = 1$), hybrid ($\alpha = 0.5$), and purely parallel-centric ($\alpha = 1$) processors running on correspondingly serial, hybrid, and parallel workloads. The data show that the massively heterogeneous approach provides larger gains for parallel than serial workloads. For the parallel case, the massively heterogeneous approach benefits from both the performance and power advantages of its low-power SPEs, since more SPEs than general purpose cores can run simultaneously. In the serial case, it can only exploit the performance benefits that the SPEs provide. However, even for strictly serial workloads, the massively heterogeneous approach outperforms `1A1M` and `1A*M` machines by up to 3.6×.

5.5 SPE design case study

The goal of massively heterogeneous systems is to improve the efficiency and performance of general-purpose applications by using a large number of different SPEs. It is well-known that specialized processors and ASICs are well-suited to signal-processing, multi-media, and some scientific workloads. However, the availability of SPEs for irregular, integer applications (e.g., those in SpecINT) is less clear. Designing the many SPEs that a massively heterogeneous system requires is also a daunting task. We must be able to generate SPEs with relative ease.

To address both these challenges, we performed an in-depth study of bzip2 from SPEC2000 [60]. We tasked one graduate student, who had limited previous hardware design experience, with designing and synthesizing two SPEs, each accelerating a specific function in bzip2. The conversion from C to Verilog only used transformations that a straight-forward compiler could implement, demonstrating that even a simple hardware compiler can create SPEs that deliver substantial benefits in a massively heterogeneous system.

5.5.1 Accelerated functions

We targeted two of the three most heavily executed functions in bzip2. `undoReversibleTransformation_fast` accounts for 23.7% of execution time on our baseline Pentium 4 and is called 155 times for the reference *graphic* input. Our SPE accelerates the run-length decoder and corresponding CRC computation, covering about 75 lines of source code. The resulting SPE runs at 500MHz. It executes 3.6× faster than the baseline processor (including cache effects), occupies only 0.062mm^2, and requires 55.5× less power (including L2 accesses).

The second function, `generateMTFValues`, accounts for 11.8% of execution time and is called 155 times on the reference input. The primary loop contains 50 lines of code. The SPE runs at 500MHz and runs 2.8% faster than the baseline processor while using only 0.113mm^2 area and requiring 55.6 times less power. Figure 7 shows the placed and routed layout of the `generateMTFValues` SPE (excluding the associated cache).

The combined area of the two SPEs and their caches is 0.175mm^2 – only 0.13% of the area of the baseline processor.

5.5.2 Transformations

Our SPEs implement functions that take few arguments and operate primarily on a set of global and local arrays. To support location-independent execution, we transformed global names into parameters passed to the SPE and stored in registers for the duration of an execution of the procedure. We assume the chip containing the SPEs has an on-chip communication network for passing start/return commands and scalar parameters between processors. Since the number of parameters is small (< 10 scalar values), and the number of function calls is small compared to execution time, we do not consider the power and delay effects of parameter passing.

5.5.3 System modeling and results

To evaluate the performance and power advantages of bzip2 SPEs, we model a system consisting of the Pentium 4 (90nm, 2.8GHz) used for our baseline analysis, augmented with the two SPEs described above. Both SPEs and the general purpose processor share the L2. Caches for these SPEs are write-through, no-write-allocate, thus allowing the L2 to enforce coherence for the SPEs using invalidations.

We used a 90nm CAD flow to synthesize our designs. We generated power and performance numbers for SPEs through a trace-based approach that provides PrimePower [21], a CAD power tool, with dynamic transition counts to determine SPE power and cycle count measurements and likewise feeds a cache simulator to model memory system power and delay effects. We use Cacti 4.2 [68] to provide power data for our memory modeling. We assume the baseline processor operates at its TDP of 84W throughout its execution.

Given that the SPEs are very small and operate at 500MHz, the memory system consumes most of the power. We use Cacti to model leakage and per-access energy for tag and data arrays at both L1 and L2. The cache simulator provides miss and access counts, which we combine with the Cacti and PrimePower-derived values to compute total SPE execution power requirements.

At 500MHz, access to the L2 cache is only a small number of cycles. Thus, the small cache associated with an SPE is as much a power- and bandwidth-saving device as a performance enhancer. The cache for `generateMTFValues` is fully associative, containing eight 16-byte lines. We use a simple stride-1 prefetcher to exploit the streaming nature of one load. The `undoReversibleTransformation_fast` SPE features a 16-entry fully associative L1 cache with 16-byte lines, and a six-entry stride-based prefetcher (length-one history for each of six entries).

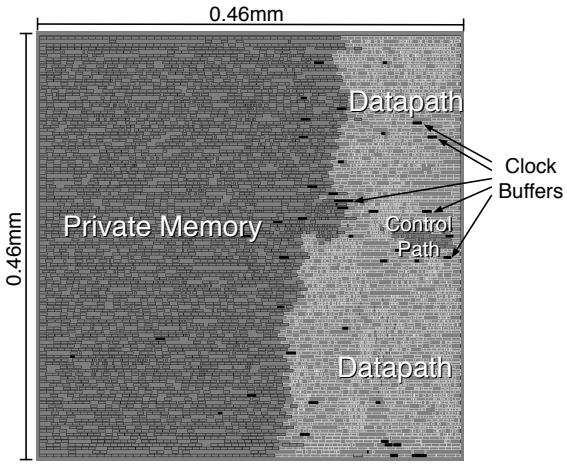


Figure 7: **Placed and Routed** `generateMTFValues` **SPE.** Major components of the SPE are shaded and labeled. Black cells dispersed throughout the SPE are clock buffers. The circuit occupies 0.113mm^2 in 90nm (or 0.007mm^2 scaled to 22nm).

5.5.4 Discussion

In only six weeks, one graduate student with minimal previous hardware design experience was able to design and synthesize two SPEs that performe as well or better than our baseline processor and consume vastly less area and power. On a system augmented with both SPEs, bzip2 executes 35% more energy-efficiently (21% faster and with 21.6% less power). It is clear that small, power-efficient SPEs offering viable performance are relatively easy to produce. Given their minute area requirements, a 320mm^2, 22nm die, with a broad selection of the larger, more general-purpose SPEs in Table 2 and a 64MB cache, could still accommodate several thousand ASIC SPEs.

6 Conclusion

We have proposed massively heterogeneous CMPs, a new approach to processor design that exploits highly-heterogeneous SPEs to mitigate the impact of tight power budgets and provide performance that scales with increased transistor density. Massively heterogeneous systems go beyond previous proposals for heterogeneous processors to include 10s, 100s, or even 1000s of SPEs on a single die. This paper has described the design space of heterogeneous processors and explored the architectural issues that arise in the design of massively heterogeneous machines. Our data demonstrate the potential of massively heterogeneous CMPs to provide significant performance gains across a wide range of applications and its ability to exploit a great diversity of SPEs. Finally, our ASIC-based case study shows that building specialized processors that provide substantial performance, power, and area savings for a particular application is relatively easy.

References

[1] http://download.intel.com/design/Xeon/datashts/30235501.pdf.
[2] http://www.freescale.com/files/32bit/doc/data_sheet/MPC7448EC.pdf?fpsp=1.
[3] http://www-01.ibm.com/chips/techlib/techlib.nsf/techdocs/-F33B5691BBB8769872571D10065F7D5/$file/-750cldd2x_ds_v2.4_pub_29May2007.pdf.
[4] http://www.freescale.com/files/32bit/doc/data_sheet/MC94MX21.pdf?fpsp=1.
[5] http://focus.ti.com/docs/prod/folders/print/tms320c6416.html.
[6] http://www.nvidia.com/page/8800_tech_specs.html.
[7] http://www.eembc.org/benchmark/score/scorereport.asp.
[8] http://www.cs.unc.edu/Research/ProjectSummaries/numericalalgorithms07.pdf,July 2007.
[9] http://ag.csail.mit.edu/ps3/lectures/6.189-lecture2-cell.pdf.
[10] http://gcc.gnu.org/ml/fortran/2005-12/msg00083.html, July 2007.
[11] http://www.ics.uci.edu/~fastmm/FMM-Reference/reference.html, July 2007.
[12] http://www.gpgpu.org/sc2006/workshop/presentations/Buck_NVIDIA_Cuda.pdf, July 2007.

[13] http://iram.cs.berkeley.edu/papers/2000.HotChips.VIRAM.pdf.
[14] http://www.cerc.utexas.edu/vlsi-seminar/spring05/slides/2005.02.16.hph.pdf.
[15] http://halcyon.usc.edu/~smohanty/papers/MILAN_SantaFe.ppt.
[16] http://adsabs.harvard.edu/abs/1990milc....1..249S, July 2007.
[17] http://iram.fr/IRAMFR/TA/backend/corchip/chip.doc, July 2007.
[18] www.sun.com/products-n-solutions/edu/events/archive/hpc/2006presentations/Mo06_JohnGustafson.pdf, July 2007.
[19] http://www.spec.org/cpu2000/results/res2006q1/cpu2000-20060109-05361.html, July 2007.
[20] http://www.spec.org/cpu2000/results/res2006q1/cpu2000-20060109-05360.html, July 2007.
[21] http://www.synopsys.com/products/primetimepx/ptpx_ds.html.
[22] J. H. Ahn, et al. "Evaluating the Imagine Stream Architecture." In *ISCA*, 2004.
[23] Alias Systems. "Alias cloth technology demonstration for the cell processor.", 2005. http://www.research.ibm.com/cell/whitepapers/alias_cloth.pdf.
[24] ATI website. http://www.ati.com.
[25] S. Balakrishnan, et al. "The impact of performance asymmetry in emerging multicore architectures." In *ISCA*, 2005.
[26] E. Basha. "Fast fourier transform on a 3d fpga." In *M.S. Thesis*, 2005.
[27] R. G. Belleman, et al. "High performance direct gravitational n-body simulations on graphics processing units – ii: An implementation in cuda." 2007.
[28] C. Benthin, et al. "Ray Tracing on the CELL Processor." *Technical Report, inTrace Realtime Ray Tracing GmbH, No inTrace-2006-001 (submitted for publication)*, 2006.
[29] J. Bolz, et al. "Sparse matrix solvers on the gpu: conjugate gradients and multigrid." *ACM Trans. Graph.*, 2003.
[30] T. Chen et al. "Cell broadband engine architecture and its first implementation.", November 2005. http://www-128.ibm.com/developerworks/power/library/pa-cellperf/.
[31] N. Clark, et al. "An architecture framework for transparent instruction set customization in embedded processors." In *ISCA*, 2005.
[32] N. Clark, et al. "Veal: Virtualized execution accelerator for loops." In *ISCA*, 2008.
[33] J. Clay Gloster, et al. "Optimizing the design of a configurable digital signal processor for accelerated execution of the 2-d discrete cosine transform." In *HICSS*, 2006.
[34] ClearSpeed. "CSX600 datasheet.", 2006. http://www.clearspeed.com/docs/resources/CSX600_Product_Brief.pdf.
[35] D. E. Shaw et al. "Anton, a special-purpose machine for molecular dynamics simulation." In *ISCA*, 2007.
[36] B. D'Amora, et al. "High-performance server systems and the next generation of online games." *IBM Syst. J.*, 2006.
[37] Y. Dou, et al. "64-bit floating-point fpga matrix multiplication." In *FPGA*, 2005.
[38] Embedded Microprocessor Benchmark Consortium. "Eembc benchmark suite." http://www.eembc.org.
[39] P. W. et al. "Exochi: architecture and programming environment for a heterogeneous multi-core multithreaded system." In *PLDI*, 2007.
[40] R. K. et al. "Core architecture optimization for heterogeneous chip multiprocessors." In *PACT*, 2006.
[41] E. Grochowski, et al. "Best of both latency and throughput."
[42] M. Harris, et al. "Parallel prefix sum (scan) with cuda." In H. Nguyen, ed., *GPU Gems 3*. Addison Wesley, Aug. 2007.
[43] J. R. Hauser et al. "Garp: A MIPS Processor with a Reconfigurable Coprocessor." In K. L. Pocek et al., eds., *FCCM*, 1997.
[44] J. Kahle. "The CELL processor architecture." In *MICRO*, 2005.
[45] R. Krashinsky, et al. "The vector-thread architecture." *SIGARCH Comput. Archit. News*, 2004.
[46] R. Kumar, et al. "Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction." In *MICRO*, 2003.
[47] R. Kumar, et al. "Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance." In *ISCA*, 2004.
[48] J. Li et al. "Power-performance considerations of parallel computing on chip multiprocessors." *ACM Trans. Archit. Code Optim.*, 2005.
[49] M. D. Linderman, et al. "Merge: a programming model for heterogeneous multi-core systems." In *ASPLOS XIII: Proceedings of the 13th international conference on Architectural support for programming languages and operating systems*, 2008.
[50] K. Mai, et al. "Smart memories: a modular reconfigurable architecture." In *ISCA*, 2000.
[51] 2007. http://download.nvidia.com/developer/presentations/2006/gdc/2006-GDC-NVIDIA-Havok_FX.pdf.
[52] nVidia website. http://www.nvidia.com.
[53] S. Olivier, et al. "Porting the gromacs molecular dynamics code to the cell processor." In *IPDPS*, 2007.
[54] D. Patterson, et al. "A case for intelligent RAM." *IEEE Micro*, April 1997.
[55] J. Pisharath, et al. "Nu-minebench 2.0. technical report." Tech. Rep. CUCIS-2005-08-01, Center for Ultra-Scale Computing and Information Security, Northwestern University, August 2006.
[56] R. Razdan et al. "A high-performance microarchitecture with hardware-programmable functional units." In *MICRO*, 1994.
[57] K. Sankaralingam, et al. "Exploiting ILP, TLP, and DLP with the Polymorphous TRIPS architecture." In *ISCA*, 2003.
[58] L. Shang, et al. "Dynamic power consumption in virtex(tm)-ii fpga family." In *FPGA*, 2002.
[59] T. Sherwood, et al. "Automatically characterizing large scale program behavior." *SIGOPS Oper. Syst. Rev.*, 2002.
[60] SPEC. "SPEC CPU 2000 benchmark specifications.", 2000. SPEC2000 Benchmark Release.
[61] J. Suh, et al. "A performance analysis of pim, stream processing, and tiled processing on memory-intensive signal processing kernels." In *ISCA*, 2003.
[62] S. Swanson. *The WaveScalar Architecture*. Ph.D. thesis, University of Washington, Seattle, WA, USA, June 2006.
[63] S. Swanson, et al. "WaveScalar." In *MICRO*, 2003.
[64] S. Swanson, et al. "Area-performance trade-offs in tiled dataflow architectures." In *ISCA*, 2006.
[65] M. B. Taylor, et al. "Evaluation of the Raw Microprocessor: An Exposed-Wire-Delay Architecture for ILP and Streams." In *ISCA*, 2004.
[66] D. Wentzlaff et al. "A Quantitative Comparison of Reconfigurable, Tiled, and Conventional Architectures on Bit-level Computation." In *FCCM*, April 2004.
[67] S. Williams, et al. "The potential of the cell processor for scientific computing." In *CF*, 2006.
[68] N. Wilton, S.J.E.; Jouppi. "Cacti: an enhanced cache access and cycle time model." *Solid-State Circuits, IEEE Journal of*, May 1996.
[69] S. Woop, et al. "Estimating performance of a ray-tracing asic design." In *Proceedings of IEEE Symposium on Interactive Ray Tracing 2006*, September 2006.
[70] Z. A. Ye, et al. "CHIMAERA: A High-Performance Architecture with a Tightly-Coupled Reconfigurable Functional Unit." In *ISCA*, 2000.

Appendix

Hardware	Cat.	P4	ASIC			MC7448	750-CL	i.MX21	Tiled:16	Tiled:8	Tiled:4	Tiled:2	Tiled:1	CELL	CSX600	DSP	GPU
			Perf	Area	Power												
Process (μm)		**0.022**				**0.022**	**0.022**	**0.022**	**0.022**	**0.022**	**0.022**	**0.022**	**0.022**	**0.022**	**0.022**	**0.022**	**0.022**
Area (mm^2)		**8.1**				**3.49**	**0.95**	**0.05**	**3.82**	**1.91**	**0.96**	**0.48**	**0.24**	**13.2**	**6.44**	**0.34**	**25.09**
Freq (GHz)		**11.6**				**7.1**	**3.3**	**1.6**	**2.5**	**2.5**	**2.5**	**2.5**	**2.5**	**1.3**	**1.5**	**4.2**	**6.25**
Power(W)		**61.7**				**15.44**	**4.12**	**0.23**	**30**	**15**	**7.5**	**3.75**	**1.87**	**22.05**	**6.41**	**1.1**	**85.73**
EEMBC																	
Text	S	**1**				**0.9**	**0.54**	**0.07**	0.32	0.32	0.32	0.32	**0.32**	0.2	0.1	0.4	0.1
Ptr. Chase	S	**1**				**0.81**	**0.36**	**0.04**	0.53	0.53	0.53	0.53	**0.53**	0.2	0.1	0.5	0.01
Ospf	M	**1**				**0.96**	**0.36**	**0.08**	8.9	6.3	4.46	3.15	**2.23**	0.2	0.1	0.5	0.1
Conv. Enc.	P	**1**	100	7.53E-05	7.00E-03	**28.93**	**0.04**	**0.003**	95	47.5	23.75	11.87	**5.93**	4	0.1	**0.02**	0.01
Auto Corr.	P	**1**	40	0.015	0.075	**1.08**	**0.1**	**0.01**	37.15	18.58	9.29	4.64	**2.32**	8	12	**0.43**	8
Fixed pt. Bit Alloc.	M	**1**				**1.36**	**0.35**	**0.05**	10.36	7.33	5.18	3.66	**2.59**	1	0.4	**0.38**	0.5
Vitrebi	P	**1**	16	0.1	0.3	**2.75**	**0.12**	**0.03**	4.92	2.46	1.23	0.62	**0.31**	18	7.05	**0.08**	17
FFT	M	**1**	60	0.12	0.35	**4.65**	**0.35**	**0.06**	9.35	6.6	4.67	3.3	**2.34**	20	20	**0.45**	4
SPEC 2000																	
Equake	P	**1**				0.9	0.4	0.05	10	5	2.5	1.25	**0.62**	2	2	0.1	2
MCF	S	**1**				0.9	0.7	0.5	2	2	2	2	**2**	1.5	0.2	0.1	0.2
Art	M	**1**				0.9	0.1	0.01	4.4	3.1	2.2	1.56	**1.1**	8	4	0.06	2
Others																	
8b/10b block enc.	P	**1**	**60.56**	1.13E-04	6.00E-03	8	0.04	0.002	**30**	15.2	7.6	3.8	**1.92**	4	0.1	0.02	0.01
802.11A Conv. Enc.	P	**1**	**20.97**	2.51E-05	2.00E-03	2	0.03	0.002	**6.88**	3.5	1.8	0.95	**0.53**	4	0.1	0.2	0.5
SGEMM	P	**1**				0.9	0.1	0.01	1.6	0.8	0.4	0.2	0.1	**21**	8	0.4	**6**
DGEMM	P	**1**				0.8	0.05	0.005	0.8	0.4	0.2	0.1	0.05	**2.9**	**7.06**	0.2	0.3
SpMM	P	**1**				0.9	0.4	0.05	6.4	3.2	1.6	0.8	0.4	2.5	3	0.15	2.5
DCT 8 X 8	P	**1**	60	0.128	0.35	3	0.35	0.06	5.59	2.8	1.4	0.7	0.35	20	12	**2.71**	4
1024-point FFT	M	**1**	150	0.18	0.5	1	0.35	0.06	1.4	1	0.7	0.49	0.35	30	**117.6**	0.4	**11**
Scan	P	**1**	15	0.2	0.5	1.1	0.45	0.2	7.2	3.6	1.8	0.9	0.45	7	6	10	**14**
GROMACS	P	**1**	105	13	3.5	3	0.3	0.05	4.8	2.4	1.2	0.6	0.3	**4**	10	0.3	100
Raytracing	P	**1**	60	11.1	3.5	1.2	0.3	0.04	4.8	2.4	1.2	0.6	0.3	**8.41**	7	0.5	**2.2**
CSLC	P	**1**				2.78	0.23	0.04	**33.4**	16.7	8.35	4.17	2.09	20.5	10	0.43	5
CityBusy	M	**1**				1.5	0.2	0.01	0.8	0.57	0.4	0.28	0.2	**1.25**	2.5	0.2	**20**
Cloth Sim.	M	**1**				1.5	0.2	0.01	0.8	0.57	0.4	0.28	0.2	**6.25**	10	0.2	**20**
Bio Sequen match	P	**1**	60	0.02	0.08	0.2	0.5	0.07	8	4	2	1	0.5	4	1	0.4	**94**

Table 4: **SPE power, performance, and area** The SPEs we have collected data for span an enormous range in power consumption, size, and performance for the kernels in the table. Values in bold are scaled from data in [62, 59, 33, 45, 7, 67, 8, 34, 53, 9, 10, 11, 58, 37, 29, 36, 23, 12, 61, 66, 42, 22, 13, 14, 15, 28, 51, 26, 16, 69, 17, 67, 27, 58, 18, 19, 20], others are estimated (see Section 4).