Processor Power Reduction Via Single-ISA Heterogeneous Multi-Core Architectures

Rakesh Kumar* Keith Farkas† Norman P Jouppi† Partha Ranganathan† Dean M. Tullsen*

*Department of Computer Science and Engineering
University of California, San Diego
{rakumar,tullsen}@cs.ucsd.edu

†HP Labs
Palo Alto, CA
{keith.farkas,norm.jouppi,partha.ranganathan}@hp.com

***Abstract*— This paper proposes a single-ISA heterogeneous multi-core architecture as a mechanism to reduce processor power dissipation. It assumes a single chip containing a diverse set of cores that target different performance levels and consume different levels of power. During an application's execution, system software dynamically chooses the most appropriate core to meet specific performance and power requirements. It describes an example architecture with five cores of varying performance and complexity. Initial results demonstrate a five-fold reduction in energy at a cost of only 25% performance.**

***Index Terms*— chip multiprocessor, low-power architecture**

I. Introduction

As processors continue to increase in performance and speed, processor power consumption and heat dissipation have become key challenges in the design of future high-performance systems. For example, Pentium-class processors currently take well over 100W and processors in the year 2015 are expected to take close to 300W. Increased power consumption and heat dissipation typically leads to higher costs for thermal packaging, fans, electricity, and even air conditioning. Higher-power systems can also have a greater incidence of failures.

This paper proposes a *single-ISA heterogeneous multi-core architecture* to reduce processor power dissipation. Prior chip-level multiprocessors (CMP) have been proposed using multiple copies of the same core (i.e., homogeneous), or processors with co-processors that execute a different instruction set. We propose that for many applications, core diversity is of higher value than uniformity, offering much greater ability to adapt to the demands of the application(s). We present a multi-core architecture where all cores execute the same instruction set, but have different capabilities and performance levels. At run time, system software evaluates the resource requirements of an application and chooses the core that can best meet these requirements while minimizing energy consumption.

The motivation for this proposal is that different applications have different resource requirements during their execution. For example, some applications may have a large amount of instruction-level parallelism (ILP), which can be exploited by a core that can issue many instructions per cycle (i.e., a wide-issue superscalar CPU). The same core, however, might be wasted on an application with little ILP, consuming significantly more power than a simpler core that is better matched to the characteristics of the application.

Previous work on power-related optimizations for processor design can be broadly classified into two categories: (1) work that uses voltage and frequency scaling of the processor core to reduce power [3], [7], (2) work that uses "gating" – the ability to turn on and off portions of the core – for power management [1], [6], [4]. Our heterogeneous multi-core architecture does not preclude the use of these techniques and can potentially address the drawbacks of these techniques to provide much greater power savings.

One way to implement a heterogeneous multi-core architecture is to take a series of previously implemented processor cores, modify their interfaces, and combine them into a single multiprocessor. Given the growth between generations of processors from the same architectural family, the entire family can typically be incorporated on a die only slightly larger than that required by the most advanced core.

In this paper, we consider implications of this single-ISA heterogeneous architecture, with particular attention to one example architecture which includes five representative cores (three in-order cores and two out-of-order cores) from an ordered complexity/performance continuum.

II. Architecture

This section gives an overview of a potential heterogeneous multi-core architecture and core-switching approach.

The architecture consists of a chip-level multiprocessor with multiple, diverse processor cores. These cores all execute the same instruction set, but include significantly different resources, and achieve different performance and energy efficiency on the same application. During an application's execution, the operating system software tries to match the applications to the different cores so as to make the best use of the available hardware while maximizing energy efficiency for a given performance requirement or goal.

A. Choice of cores.

Our heterogeneous multi-core architecture is based on the hypothesis that the performance difference between the cores varies across different workloads. In other words, the "best" core (defined, for now, as some desired combination of power and performance) for one application may not be best for another. One application may benefit greatly from wide issue and dynamic scheduling, another benefits from neither. Thus,

The authors would like to thank the Semiconductor Research Corporation who provided part of the funding for this project, and the anonymous reviewers for helpful suggestions.

Manuscript submitted: 30 Mar. 2003. Manuscript accepted: 7 Apr. 2003. Final manuscript received 17 Apr. 2003.

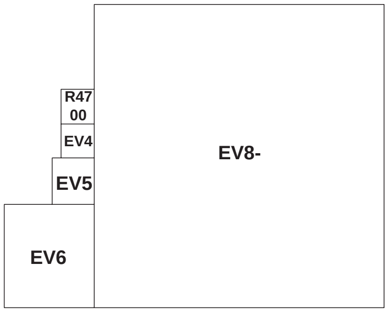

Fig. 1. Relative sizes of the cores used in the study

the latter gains nothing from the extra power required for it to run on a high-performance processor.

To provide an effective platform for a wide variety of application execution characteristics, the cores on the heterogeneous multi-core processor should cover both a wide and evenly spaced range of the complexity/performance design space. This initial study considers a design that takes a series of previously implemented processor cores with slight changes to their interface – this preserves one of the key advantages of the CMP architecture, namely the effective amortization of design and verification effort. For breadth, we include both a single-threaded version of the EV8 (Alpha 21464), referred to as EV8-, and the MIPS R4700, a processor targeted at very low-power applications. To fill out the design space, we also include the EV4 (Alpha 21064), EV5 (Alpha 21164), and EV6 (Alpha 21264). Core switching is greatly simplified if the cores can share a single executable, so we assume a variant of the R4700 that executes the Alpha ISA. Finally, we assume the five cores have private L1 data and instruction caches and share a common L2 cache, phase-lock loop circuitry, and pins.

We chose the cores of these off-the-shelf processors due to the availability of real power and area data for these processors, except for the EV8 where we use projected numbers.

Figure 1 shows the relative sizes of the cores used in the study, assuming they are all implemented in a 0.10 micron technology (the methodology to obtain this figure is described in the next section). It can be seen that the resulting core is only modestly (within 15%) larger than the EV8- core by itself.

For our initial results, we assume only one application runs at a time on only one core. Because we assume a maximum of one thread running, the multithreaded features of EV8 are not needed. Hence, these are subtracted from the model, as discussed in Section III. In addition, this assumption means that we do not need more than one of any core type. Finally, since only one core is active at a time, we implement cache coherence by ensuring that dirty data is flushed from the current core's L1 data cache before execution is migrated to another core.

This particular choice of architectures also gives a clear ordering in both power dissipation and expected performance. This allows the best coverage of the design space for a given number of cores and simplifies the design of core-switching algorithms.

B. Switching of workloads between cores.

The second hypothesis in our study is that different cores have varying energy efficiencies for the same workload. Typical programs go through phases with different execution characteristics – the best core during one phase may not be best for the next phase. This observation motivates the need to dynamically switch cores in mid execution to take full advantage of our heterogeneous architecture.

There is a cost to switching cores, so we must restrict the granularity of switching. One method for doing this would switch only at operating system timeslice intervals, when execution is in the operating system, with user state already saved to memory. If the OS decided a switch was in order, it would trigger a cache flush to save all dirty cache data to the shared L2, power up the new core, and signal the new core to start at a predefined OS entry point. The new core would then power down the old core and return from the timer interrupt handler. The user state saved by the old core would be loaded from memory into the new core at that time, as a normal consequence of returning from the operating system.

In this work, we assume that unused cores are completely powered down, rather than left idle. Thus, unused cores suffer no static leakage or dynamic switching power. This does, however, introduce a latency for powering a core back up. We estimate that a given processor core can be powered up in approximately one thousand cycles of the 2.1GHz clock. This assumption is based on the observation that when we power down a processor core we do not power down the phase-lock loop that generates the clock for the core. Rather, in our multi-core architecture, the same phase-lock loop generates the clocks for all cores. Consequently, the power-up time of a core is determined by the time required for the power buses to charge and stabilize. In addition, to avoid injecting excessive noise on the power bus bars of the multi-core processor, a staged power up would likely be used. Experiments confirm that switching cores at operating-system timer intervals ensures that the switching overhead has virtually no impact on performance, even with the most pessimistic assumptions about power-up time, software overhead, and cache cold start effects.

III. Methodology

This section discusses the various methodological challenges of this research, including modeling the power, the real estate, and the performance of the heterogeneous multi-core architecture.

A. Modeling of CPU Cores

As discussed earlier, the cores we simulate are roughly modelled after cores of R4700, EV4 (Alpha 21064), EV5 (Alpha 21164), EV6 (Alpha 21264) and EV8-. EV8- is a hypothetical single-threaded version of EV8 (Alpha 21464). The data on the resources for EV8 was based on predictions made by Joel Emer [2] and Artur Klauser [5], conversations with people from the Alpha design team, and other reported data. The data on the resources of the other cores are based on published literature on these processors.

TABLE I

CONFIGURATION OF THE CORES

Processor	R4700	EV4	EV5	EV6	EV8-
Issue-width	1	2	4	6 (OOO)	8 (OOO)
I-Cache	16KB, 2-way	8KB, DM	8KB, DM	64KB, 2-way	64KB, 4-way
D-Cache	16KB, 2-way	8KB, DM	8KB, DM	64KB, 2-way	64KB, 4-way
Branch Pred.	Static	2KB,1-bit	2K-gshare	hybrid 2-level	hybrid 2-level (2X EV6 size)
Number of MSHRs	1	2	4	8	16

The multi-core processor is assumed to be implemented in a 0.10 micron technology. The cores have private first-level caches, and share an on-chip 3.5 MB 7-way set-associative L2 cache. At 0.10 micron, this cache will occupy an area just under half the die-size of the Pentium 4. All the Alpha cores (EV4,EV5,EV6,EV8-) are assumed to run at 2.1GHz. This is the frequency at which an EV6 core would run if its 600MHz, 0.35 micron implementation was scaled to a 0.10 micron technology. All of the Alpha cores were designed to run at high frequency, so we assume they can all scale to this frequency (if not as designed, processors with similar characteristics certainly could). On the other hand, the R4700 is not designed primarily for high clock rate; thus, we assume it is clocked at 1 GHz.

Table I summarizes the configurations that were modelled for various cores. We did not faithfully model every detail of each architecture, but we were most concerned with modeling the approximate spaces each core covers in our complexity/performance continuum. However, all architectures are modelled as accurately as possible, given the parameters in Table I, on a highly detailed instruction-level simulator. The various miss penalties and L2-cache access latencies for the simulated cores were determined using CACTI [11].

Note that while we took care to model real architectures that have been available in the past, we could consider these as just sample design points in the continuum of processor designs that could be integrated into a heterogeneous multiple-core architecture. These existing designs already display the diversity of performance and power consumption desired.

B. Modeling Power

Table 2 shows our power and area estimates for the cores. Power dissipation for all implemented cores is derived from published numbers, forcing us to start with peak power data obtained from datasheets and conference publications. While this basis ensures that our power estimates are high, we believe that the typical power for each core scales roughly with peak power. This gives us an adequate yardstick to determine the initial feasibility of this approach.

To derive the peak power dissipation in the core of a processor from the published numbers, the power consumed in the L2-caches and at the output pins of the processor must be subtracted from the published value. Details of this methodology will be described in more detail in later publications.

The second column in Table II summarizes the power consumed by the cores at 0.10 micron technology. As can be seen from the table, the EV8- core consumes almost 200 times the power and 80 times the real estate of the R4700 core. CACTI was also used to derive the energy per access of the shared L2-cache, for use in our simulations.

TABLE II

PEAK POWER AND AREA STATISTICS OF THE CORES

Core	Core-power	Core-area	Power/area
	(Watts)	(mm^2)	Watt/mm^2
R4700	0.30	2.80	0.11
EV4	1.83	2.87	0.64
EV5	4.43	5.06	0.88
EV6	10.80	24.5	0.37
EV8-	60.11	236	0.26

C. Estimating Chip Area

Table II also summarizes the area occupied by the cores at 0.10 micron (also shown in Figure 1). The area of the cores (except EV8-) is derived from published photos of the dies after subtracting the area occupied by I/O pads, interconnection wires, BIU (bus-interface unit), L2 cache, and control logic. Area of the L2 cache of the multi-core processor is estimated using CACTI.

The die size of EV8 was predicted to be 400 mm^2 [8]. To determine the core size of EV8-, we subtract out the estimated area of the L2 cache (using CACTI). We also account for reduction in the size of register files, instruction queues, reorder buffer, and renaming tables to account for the single-threaded EV8-. The area data is then scaled for the 0.10 micron process.

D. Modeling Performance

Benchmark execution is simulated using SMTSIM, a cycle-accurate, execution-driven simulator that simulates an out-of-order, simultaneous multithreading processor [10]. SMTSIM executes unmodified, statically linked Alpha binaries. The simulator was modified to simulate a multi-core processor comprising five heterogeneous cores sharing an on-chip L2 cache and the memory subsytem. Because the R4700 does not execute Alpha binaries, what we are modeling is an R4700-like architecture targeted to the Alpha ISA.

In all simulations in this research we assume a single thread of execution running on one core at a time. Switching execution between cores involves flushing the pipeline of the "active" core and writing back all its dirty L1 cache lines to the L2 cache.

IV. INITIAL RESULTS

Figure 2 shows results for the SPEC application *applu*. Performance and power are modeled for each processor, with the ratio (IPS^2/W) (essentially, the inverse of power-delay product) shown on the Y axis. The bold line shows the core at each interval which minimizes the power-delay product over that interval, with the constraint that we never choose a core that sacrifices more than 50% performance relative to EV8-

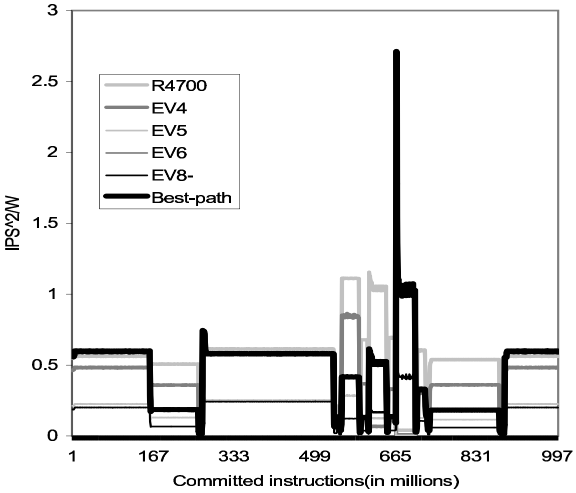

Fig. 2. *Oracle* switching for best energy-delay – *applu*. IPS^2/W varies inversely with energy-delay product

over an interval. In this figure, four different cores are used for some interval. Compared to a single-core architecture (e.g., one that only contained the EV8- core), this configuration could ideally reduce the power-delay product by 74% (a nearly 4X improvement in IPS^2/W). This comes from a combination of a 25% performance loss and a 81% energy savings (that's a five-fold reduction in energy). Relaxing the (50%) performance constraint would allow even higher energy-delay savings, but would make greater performance sacrifices to do so. More conservative constraints are also possible, of course – another design point yields a 36% reduction in energy with a 4% performance loss. It is trivial to adapt these techniques to optimize other metrics besides power-delay product (depending on the actual priorities of the architecture or application), and we have experimented with some of those. It should be noted that the hardware architecture need not change for varying power/performance tradeoffs – it is only necessary for the switching algorithm to change.

V. Related Work

There has been a large body of work on power-related optimizations for processor design. These can be broadly classified into two categories: (1) work that uses voltage and frequency scaling of the processor core to reduce power, (2) work that uses ”gating” – the ability to turn on and off portions of the core – for power management.

Voltage and frequency scaling reduces the parameters of the entire core [3], [7]. While this reduces power, the power reductions are uniform - across both the portions of the core that are useful for this workload as well as the portions of the core that are not. Gating-based power optimizations [1], [6], [4] provide the option to turn off (gate) portions of the processor core that are not useful to a workload. For example, half of the banks in the branch predictor could be turned off in the example above. However, this kind of gating does not address the power consumption in driving wires across the inactive areas of the processor core.

The architecture proposed in this paper addresses the drawbacks of gating by effectively designing multiple processor cores each optimized for a particular energy efficiency for a particular performance.

Several other studies have also identified the differences in the behavior characteristics across different applications and different phases between applications [9].

VI. Conclusions and Future Work

This paper seeks to gain some initial insights into the energy benefits available for a new architecture, that of a heterogeneous set of cores on a single multi-core die, sharing the same ISA. To do this, we constrain the problem to a single application switching among cores to optimize some function of energy and performance.

We show that a sample heterogeneous multi-core design with five cores capable of executing the Alpha ISA has the potential to increase energy efficiency in one benchmark by a factor of four (by reducing energy by a factor of five, and performance by only 25%).

We will be fully exploring the design space of a single thread utilizing a heterogeneous chip to maximize energy efficiency, including algorithms and heuristics for thread movement. We will explore the best configurations of individual cores, assuming more freedom to alter each core. We will explore both performance and power implications of multiple threads running on a multi-core design, including multithreaded cores.

References

[1] D. H. Albonesi. Selective cache-ways: On demand cache resource allocation. In *IEEE/ACM International Symposium on Microarchitecture (MICRO-32)*, 1999.

[2] J. Emer. EV8:the post-ultimate alpha. In *PACT Keynote Address(http://research.ac.upc.es/pact01/keynotes/emer.pdf*, 2001.

[3] K. Govil, E. Chan, and H. Wasserman. Comparing algorithms for dynamic speed-setting of a low-power cpu. In *1st Int'l Conference on Mobile Computing and Networking*, Nov. 1995.

[4] D. Grunwald, A. Klauser, S. Manne, and A. Pleskun. Confidence estimation for speculation control. In *25th Annual International Symposium on Computer Architecture*, June 1998.

[5] A. Klauser. Trends in high-performance microprocessor design. In *Telematik-2001*, 2001.

[6] S. Manne, A. Klauser, and D. Grunwald. Pipeline gating: Speculation control for energy reduction. In *25th Annual International Symposium on Computer Architecture*, June 1998.

[7] T. Pering, T. Burd, and R. Brodersen. The simulation and evaluation of dynamic voltage scaling algorithms. In *Proceedings of 1998 International Symposium on Low Power Electronics and Design*, Aug. 1998.

[8] J. M. Rabaey. The quest for ultra-low energy computation opportunities for architectures exploiting low-current devices. In *UC Berkeley Solid State Seminar*, Apr. 2000.

[9] T. Sherwood, E. Perelman, G. Hammerley, and B. Calder. Automatically characterizing large-scale program behavior. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 2002.

[10] D. Tullsen. Simulation and modeling of a simultaneous multithreading processor. In *22nd Annual Computer Measurement Group Conference*, Dec. 1996.

[11] S. Wilton and N. Jouppi. CACTI: an enhanced cache access and cycle time model. In *IEEE Journal of Solid State Circuits, Vol 31, No. 5*, May 1996.