Aide de Camp: Asymmetric Dual Core Design for Power and Energy Reduction

Soraya Ghiasi, Dirk Grunwald
Dept. of Computer Science
University of Colorado, Boulder
Boulder, CO 80309-0430

Abstract

Power consumption and heat dissipation are becoming extremely important as processors continue to become more complex and are implemented in smaller and smaller technologies. Many applications do not require the full processing power of modern chips. The question then arises of how we can optimize for both applications that require the full processing power of modern designs and those that don't.

A symmetric dual-core design that combines two high performance processors provides more computing power than needed by many desktop applications. An *asymmetric* dual-core design will allow the processor to operate under heavy loads at a performance near that of two equal cores, but with considerably less power consumption and heat dissipation. The reconfigurable component of an asymmetric design allows an additional degree of freedom when dealing with operating systems scheduling and thermal overload situations. During one mode of normal operation both cores are powered on and may have jobs scheduled to them. It is also possible to shut down the high powered core and continue processing on the less power-intensive core during thermal overload situations. In addition, an asymmetric design can be used as an alternative to existing complexity-effective designs with application driven scheduling policies that determine when and where an application runs. We believe an asymmetric dual-core design is less complex to design than many proposed complexity-effective designs; however, dual core designs can also incorporate proposed complexity-effective designs.

We introduce the concept of general purpose asymmetric dual core processors and show they can be used to reduce power, energy, energy-delay, and static power, as well as to control thermal overloads. We believe that reusing existing processor designs is a straightforward way to build asymmetric dual-cores with the additional benefit of allowing the reuse of debugged and validated designs. Our simulated results show asymmetric dual-core designs can provide power, energy and energy-delay product savings. We analyzed the impact of process scaling on abstracted processors loosely based upon Intel Pentium Pro, Pentium II, and Pentium III processors. We found that the scaled processors reduce power consumption and energy usage by at least 80% and provide energy delay product savings of at least 35% when compared against a more complex, modern processor. Similarly, static power and energy losses are reduced by a factor of five. When used as a second

processor for reducing thermal and energy constraints, we find two possible design solutions both of which reduce the energy delay product by 25%.

1 Introduction

Power and power density are increasingly important design constraints for processors. Overall power is increasing because more functionality is being placed in processors; power density, the power per unit area, is increasing because processor area is decreasing faster processor power across different processor technology improvements. As process geometries continue to shrink, *static power* becomes an increasing factor in overall power use and leakage currents are the major contributor to static power.

At the same time, much of the functionality added to current processors provides little in the way of significant performance improvement for many applications. For example, specific applications may see improvement due to added functionality from a new branch predictor or instruction set extensions but most applications see improvement primarily from increases in processor speeds. To counter this, many researchers have examined mechanisms for *complexity effective designs*. These designs dynamically modify or throttle some aspect of processor operation in order to reduce the power usage of that component that is not used.

Current energy reduction techniques involve either running a processor at slower speed (frequency and voltage scaling), disabling unneeded components (functional unit gating) or using less complex components (complexity-effective designs).

Power consumption can be broken down into dynamic and static components. Dynamic power consumption is caused by switching, while static power consumption (*i.e.* "leakage") is an undesirable side-effect of CMOS technology. Voltage scaling reduces dynamic power consumption because $P_{dyn} \propto C f V_{cc}^2$ where C is the capacitance, f is the frequency, and V_{cc} is the core voltage. Reducing the frequency allows you to reduce processor voltage which results in an overall $P_{dyn} \propto V_{cc}^3$. Voltage scaling has a linear reduction on leakage power, rather than the cubic reduction seen with dynamic power. Static power can be estimated using $P_{static} = V_{cc}\, I_{leak}\, N\, k_{design}$ where V_{cc} is the core voltage, I_{leak} is the leakage current, N is the number of transistors, and k_{design} is a scaling factor representing device characteristics. Voltage scaling is implemented in a number of processors, including the Intel Pentium 4 series, AMD PowerNow, Intel XScale and the TransMeta Crusoe.

Clock gating is implemented in most processors for certain components; for example, floating point units are typically gated since they are only used in certain workloads. Clock gating functional units improves the dynamic power by reducing the total capacitance, but unless power to the functional unit can be completely disabled rather than simply not discharged, gating may not help leakage power. "Power Gating" can reduce leakage by gating the power supply to a component. Both clock and power gating can exacerbate the "current surge" problem by rapidly varying the current needed by the

processor; since there is limited current sourcing available, such gating techniques may need several cycles to fully precharge a disabled component.

The efficient use and management of speculation is one of the primary differences in the energy efficiency of different processor designs. Aggressive speculation techniques, and the structures to support them, are efficient only when the speculation is accurate. Complexity effective design seeks solutions that are either more effective for a given complexity than a traditional design or produces similar performance results for a less complex design. Complexity effective designs use a variety of techniques to either reduce speculation (*e.g.* through pipeline gating or similar techniques) or reduce the complexity of associated data structures. For example, if instruction fetch is being gated, either through pipeline gating or some other technique, a large issue window and the complex control logic for that issue window may be superfluous. A complexity effective design may then adjust the size of the issue window. Such techniques attempt to reduce the power used by adaptive components without expending significant additional power on determining *when* and *how* the component should be adapted. Most complexity effective designs influence dynamic rather than static power by not using (discharging) certain components, although some designs may reduce leakage by completely gating power or back-biasing when a component is not being used for long periods.

We propose an alternative to complexity effective designs called *Aide De Camp* (ADC). ADC is an example of an asymmetric multi-core processor design. Rather than augment a new processor with features such as issue-window or execution unit throttling [3, 8, 2], we propose to integrate two fully functional but structurally dissimilar processors on a single die. As an example, one might consider integrating a complex processor core (*e.g.* a Pentium 4) and a simpler but functionally similar core (*e.g.* a Pentium, Pentium II, *etc*). During periods of high computing demand which make good use of the speculative execution features of such aggressive processors, the higher functionality core is used. During periods of more modest demands, or when the aggressive speculation provides little benefit, the lower functionality processor is used.

The ADC designs we propose can make use of voltage scaling and functional unit gating. Although ADC designs can incorporate elements of current complexity effective adaptation designs, our intent is to replace fine-grain adaptation of microarchitectural structures with coarser changes. In this paper, we will argue that such coarse design changes are likely to be as effective in reducing dynamic power as are fine-grain adaptations and are more likely to lead to reductions in leakage power, which will soon dominate dynamic power in total power usage.

There are other compelling reasons to explore ADC designs as an alternative to fine-grain complexity-effective adaptations. First, ADC designs will be simpler, reducing the design complexity of critical components. Indeed, an ADC design can reuse existing core designs, where practical, after scaling those older designs for modern fabrication processes. Second, unlike fine-grain complexity-effective designs, an ADC design can use *both* cores when demand is high. By comparison, fine-grain com-

plexity effective designs can only be used by a single running process, and their associated control hardware is purely overhead when not being used.

In the rest of this paper, we explore the concept of general purpose asymmetric dual core processors and demonstrate their ability to reduce power, energy, energy-delay products and static energy as well as their applicability in the area of thermal control. We describe the motivation for ADC designs, using public information concerning actual modern implementations and scaling projections for those processors in §2. Throughout this paper, we assume that ADC processors will use existing processor cores, scaled to new fabrication processes; this means the two cores will typically run at different clock speeds. In §3, we discuss the related work. Section 4 describes our measurement and analysis methodology and we present concrete results in §5.

2 Motivation

Modern processor designers search for new ways to improve processor performance. Until recently, the techniques developed have focused primarily on improving the instruction level parallelism of a processor. Techniques such as speculative execution and out-of-order execution have been able to gain some improvements in performance for specific classes of applications. New branch predictors or or instruction set extensions may improve the performance of certain applications, but have no impact on many others. For many common applications, recent performance gains have come primarily from faster clocks. Designers have recently broadened their scope of interest and are now looking into techniques that improve thread level parallelism in addition to instruction level parallelism.

Thread level parallelism efforts are focused primarily in two different areas. The first of these efforts, Simultaneous MultiThreading, allows multiple contexts to execute in a single pipeline. Resources are shared, duplicated or partitioned between the different contexts. The second thread level parallelism design alternative focuses instead on chip multiprocessing or multi-core design where multiple cores are placed on a single die. Different threads or applications can be scheduled to each core. The cores can be identical or can differ from one another. A shared memory processor may be implemented as a multi-core design. Aide de Camp is a variant of multi-core design.

In selecting a second core for ADC's asymmetric dual-core design, it is necessary to take into account a number of trends in microarchitectural design. As designs have become more complicated, the amount of work done per pipeline stage has been reduced so that the load can be driven in a single clock cycle. Less work per pipeline stage has led to deeper pipelines. The load per stage is typically measured in fan-out-of-four (FO4) inverter delays. A design is not necessarily less complex if it has a low number of FO4 inverter delays. It is possible that its complexity has been spread out over multiple stages. For example, the Pentium 4 requires three stages for allocate and renaming of registers while the older Pentium Pro requires only one stage to perform the same work. The number of FO4 inverter

delays may change over the lifetime of a given processor as refinements are made to a critical element. These refinements can increase or decrease the FO4 inverter delays of a processor.

The load, measured in FO4 inverter delays, which can be driven in a reduced technology size will greatly limit the performance of the older, smaller, less complex core when it is reimplemented in the new process technology. We use publicly available information about the Intel process technologies for the rest of this analysis. Our sample calculation presented below determines the maximum drivable frequency as well as provides estimates for power and performance for the processor that has been reimplemented in a new, smaller technology. We use the term "process scaling" to refer to the act of taking a given processor implemented in an older technology and reimplementing it into a newer technology. The term "scaled processor" will refer to the processor that has undergone process scaling.

2.1 Calculating Maximum Drivable Frequencies

The maximum frequency at which a scaled processor can be driven is a function of both its FO4 per pipeline stage and the technology to which it is scaled. The recently released Northwood variant of the Pentium IV processor is implemented in P860 technology as is the second generation of Pentium 4 processors. P860 is a 0.13 μ technology with a gate length of 0.07 μ. The worst case FO4 gate delay for processors in this generation case can be calculated using the gate length [14]. We have conservatively chosen to use the worst case gate delay to determine maximum drivable frequencies. This decision will be reflected in larger energy-delay products, but widens the range over which voltage scaling may be used to achieve additional power savings.

$$FO4\ gate\ delay(worst\ case)\ =\ 500\ *\ gate\ length$$

$$gate\ length\ =\ 0.07\mu\ in\ P860\ (Pentium4)$$

$$FO4\ gate\ delay(worst\ case)\ =\ 35\,pS$$

The gate delay can then, in turn, be used to find the maximum frequency at which an older design may be driven. The cycle time is calculated using:

$$cycle\ time\ =\ FO4\ gate\ delay\ *\ FO4(processor)$$

For example, at the end of its life, the 80486 had a FO4 of 42 per clock cycle, leading to a minimum cycle time of 1470 pS and a maximum drivable frequency of 680 MHz when it is scaled to a .13 μ process.

$$cycle\ time\ =\ FO4\ gate\ delay\ *\ FO4(80486)$$

$$= 35\,pS * 42 = 1470$$

$$max\,drivable\,frequency = 1000/cycle\,time$$

$$= 680MHz$$

2.2 Power and Performance Estimation

The maximum drivable frequency is one important factor in determining performance. Unfortunately, the complexity of the chip design also plays a vital role and it is more difficult to compare older generations to new generations due to the unavailability of hardware or even benchmarking results on both platforms. In order to make a comparison, SPEC2000 results have been estimated based on SPEC95 results. An average scaling factor was found by averaging across processors for which both SPEC95 and SPEC2000 results were available. The following is again for the 486 implemented using a 0.13 μ technology:

$$Estimated\,Spec95\,int = 4.29$$

$$Estimated\,Spec2K\,int = 43$$

A performance ratio between the 486 and the Northwood can be found using the estimate of SPEC2000 on both processors.

$$\text{perf ratio to 2.8 GHz Pentium 4} = 43/833$$

$$= 5.1\%\text{of performance}$$

A less complex, older design reimplemented in a smaller technology size has one feature which makes it appealing despite its significantly lower performance - it uses much less power than more modern designs. Without additional details, it is difficult to estimate the power consumption of the reimplemented older design. It is, however, easy to derive a conservative upper bound. By using the voltage needed by the Pentium 4 running at its drivable frequency and the derived maximum drivable frequency for the scaled processor as well as applying scaling to the capacitance per unit area and the processor area, the power ratio between them can be estimated:

$$P \propto CV^2f$$

$$P_{scaled} = C_{unit\_area} * scaling\_factor * scaled\_area * V^2f$$

$$power\,ratio = \frac{P_{max}[486]}{P_{max}[Pentium\,4]}$$

process	year	circuit	gate	FO4	cycle	freq	orig freq	spec95	spec2k	%perf	%pow	cpu
P648	1989	1.0	1.0	42	1470	680	100	4.29*	43*	5.1	0.9	486
P852	1993	0.50	0.50	35	1225	816	133	4.01	40*	24.9	1.5	Pentium (75-266 MHz)
P854	1995	0.35	0.35	23	805	1242	200	8.2	83*	52.4	16	Pentium Pro
P856	1997	0.25	0.20	27	945	1058	266	10.8	110*	44	55	Pentium II
P858	1999	0.18	0.13	35	1225	816	800	N/A	399	39.8	16	Pentium III
P858	1999	0.18	0.13	16	16	1800	1400	N/A	486	63.5	84	Pentium 4
P860	2001	0.13	0.07	16	560	2800	2800	N/A	984	100	100	Pentium 4 Northwood

Table 1: Intel Process Generations Scaled to 0.13 μ. Values indicated with asterisks are estimates from past performance.

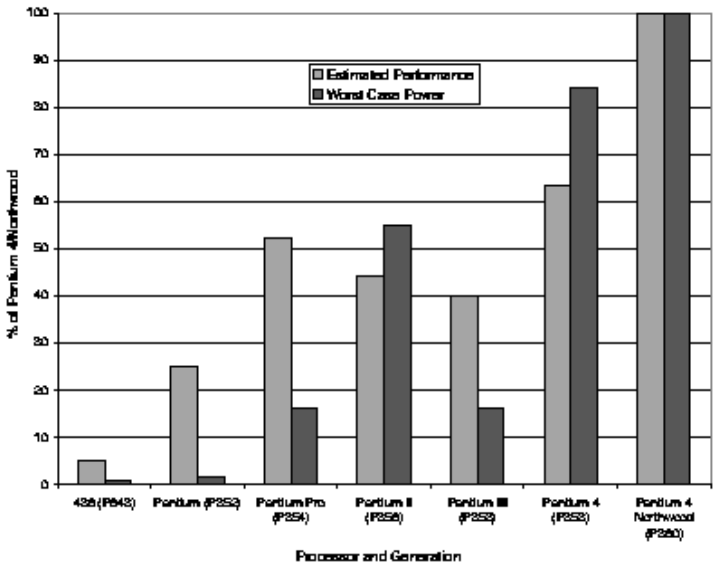


Figure 1: Relative Performance and Power for Intel Processors Scaled to 0.13 μ

$$= 0.9\% \, of \, the \, power \, of \, a \, 2.8 \, GHz \, Pentium \, 4$$

Table 1 shows the maximum drivable frequencies, performance ratios, and power ratios for a variety of Intel processor ranging from the 486 through the Pentium 4 Northwood (generations P648 through P860). All comparisons are done against a 2.8 GHz Pentium 4 Northwood; we have estimated performance of these scaled processors on the SPEC2K suite. FO4 per cycle estimates are from end of life if applicable, rather than beginning of life, to take advantage of developments made over the course of a processor's lifespan.

Figure 1 shows that the Pentium Pro appears to be a good selection for the less powerful core in an asymmetric dual-core design because it has a high percentage of the performance of a Pentium 4 (52.4%) at a much lower power (16%). We have also studied additional design points, but not in as

much detail as the Pentium Pro.

The method presented above produces an extremely conservative upper bound on both performance and power. It is possible, and indeed likely, that scaled processors will run faster than worst case gate delay analysis predicts. In addition, the actual voltage necessary to drive the scaled processor at its maximum frequency will generally be lower than the full voltage of the Pentium 4 Northwood running at its maximum frequency. For example, Intel produces a 0.13 μ 1.4 GHz Pentium III that consumes .5 Watts of power at 1.15 volts. However, the above analysis shows that even under conservative assumptions, there are design points worth considering for an asymmetric dual core design.

3 Related Works

Dual core designs have typically focused on either symmetric or asymmetric approaches. Symmetric designs make use of two identical cores. Asymmetric designs instead allow for specialized cores. Little work appears to have been done on non-specialized asymmetric designs. Intel and others have done work on scaling processors between generations but have not placed new and older, scaled processors together on the same die. Some implementations of the "Itanium" processor design do incorporate a small IA-32 processor on-die, but this is used to execute IA-32 instructions.

Our work work builds upon scaling ideas presented by Ho, et al [14]. They introduce the concept of worst case gate delay and emphasize the trends that occur when processors are scaled to a new generation. It also takes into account FO4 work by Hrishikesh, et al [15] which suggests the optimal FO4 per pipeline stage is only six to eight FO4 inverter delays. Their work suggests some modifications to existing designs, such as segmented instruction windows, to enable future microprocessors to deal with large number of pipeline stages that will be required. Hartstein, et al [1] explored optimizations of pipeline depths and found different classes of applications had different optimal depths. These works, taken together, indicate modern processors will continue to have longer pipelines, but also, that the pipelines of a given depth may not be optimal for all applications used on that platform.

Voltage and frequency scaling can be used to reduce power consumption and handle thermal overloads. Voltage scaling relies on the idea of running the processor as slowly as possible to meet the processing demands of applications. Weiser et al. [25] proposed some of the earliest speed setting algorithms which have since been refined and supplemented by *avgn* by Pering at al. [20, 21], Lorch and Smith's PACE [18] and others [9, 22]. We do not dynamically adjust the voltage or frequency of either processor on our asymmetric dual core. We do, however, make use of the same underlying assumption that they do. Both approaches rely on the fact that $P \propto CV^2f$.

Our work differs from efforts in the area of complexity effective design. Complexity effective design typically attacks power issues by a wide range of techniques. These range from speculation control techniques such as pipeline gating [19, 3] and dynamic IPC adjustment to dynamically resizing

Processor Abstractions				
Intel Processor	FO4 delay	Cache Size	Pipeline Depth	Abstracted Name
Pentium Pro (P854)	Low	Moderate	Moderate	LMM
Pentium II (P856)	Moderate	Large	Moderate	MLM
Pentium III (P858)	High	Moderate	Moderate	HMM
Pentium 4 Northwood (P860)	Very Low	Large	Large	VLL

Table 2: Processor abstractions for Intel processor generations P854 to P860.

structures such as caches [2], instruction windows[10], and issue queues[8]. Our approach instead focuses on using a less complex design that has been scaled to a new technology. Complexity effective designs are more reconfigurable than our approach, but they could be used in conjunction.

Static power consumption will be equivalent to dynamic power consumption within a few generations [7, 26]. As core voltages are lowered, it is necessary to also lower threshold voltages to keep sufficient noise margins. Subthreshold leakage increases as the threshold voltage is dropped $Pstatic = V_{cc}\ I_{leak}\ N\ k_{design}$. Without adjusting the size of the transistors and simply "scaling" an existing design, the V_{cc} of the scaled design will be lower than that of the original design. With the same transistor designs, the scaled design will be limited in the possible speed it can achieve; although these are faster than the original design, they would not be as fast as a processor using transistors designed for that process technology. Thus, relative to current processors, a "scaled processor" can have lower V_{cc} and significantly lower transistor counts (N).

Efforts to reduce leakage power rely primarily on circuit techniques such as multi-V_t[11, 16] or variable V_t[4] designs and leakage biasing [13, 12]. Another technique powers down unused circuitry by gated V_{dd} [27, 23].

4 Methodology

We used Wattch[5], a SimpleScalar [6] 3.0-based power model, to simulate processors inspired by the Intel Pentium Pro, Pentium II, Pentium III and Pentium 4 processors in a 0.13 μ technology. Wattch already had much of the information necessary to support scaling across generations. We adjusted the area scaling factor and frequency to match the requirements of the scaled processors. We also modified the core and threshold voltages to match the now known values for the P860 process, as well as reasonable estimates of the scaled voltage at the reduced frequencies that our scaled processors run.

Lack of complete information detailing the processors and shortcomings in the simulation infrastructure prevent modeling Intel processors exactly. However, our results do illuminate general trends which should be seen when using Intel processors. The important features of the various Intel designs

Pipeline Simulation Configurations				
Parameters	LMM	MLM	HMM	VLL
Machine Width	4-wide fetch, 4-wide issue, 4-wide commit			
Window Size	64 entry RUU 32 entry load/store queue			128 entry RUU 128 entry load/store queue
Branch Misprediction	min. recovery latency 12 cycles			min. recovery latency 19 cycles
L1 Icache	8K 4 way	16K 4 way	16K 4 way	12K μop trace cache
L1 Data Cache	8K 4 way 32 byte lines 3 cycle hit latency	16K 4 way 32 byte lines 3 cycle hit latency	16K 4 way 32 byte lines 3 cycle hit latency	8K 4 way 3 μops 2 cycle hit latency
L2 Cache Combined	256K 4 way 32 byte lines 25 cycle hit latency	512K 4 way 32 byte lines 25 cycle hit latency	256K 4 way 32 byte lines 25 cycle hit latency	512K 8 way 128 byte lines 10 cycle hit latency
Memory	128 bit wide 41 cycle hit latency	128 bit wide 38 cycle hit latency	128 bit wide 29 cycle hit latency	128 bit wide 92 cycle hit latency
BTB	512 entry, 4-way set-associative, 32 entry return address stack			4096 entry, 4-way set-associative, 32 entry return address stack
TLB	64 entry (I), 64 entry (D), 4-way set-associative, 30 cycle miss latency			128 entry (I), 128 entry (D), 4-way set-associative 30 cycle miss latency
Functional Units and Latency (total/issue)	1 Int ALU (1/1), 1 Int Mult (2/2) / Div(2/2), 2 Load/Store (2/1), 1 FP Add (5/3), 1 FP Mult (6/5) / Div (6/5) / Sqrt (6/5)			2 Int ALU (1/1), 1 Int Mult (2/2) / Div(2/2) 4 Load/Store (2/1), 1 FP Add (5/3), 1 FP Mult (6/5) / Div (6/5) / Sqrt (6/5)

Table 3: Pipeline parameters for our scaled processors.

can be abstracted and the abstracted processors studied. The key features of interest for Aide de Camp are the FO4 delay, the cache sizes, and the pipeline depth. The preliminary results take into account the first two features, but do not account for the third.

The Intel processors and their abstracted names are presented in Table 2. FO4 is divided into four categories: high (> 30), moderate ($25 - 30$), low ($20 - 25$), and very low (< 20). Cache sizes for the cores of interest fall into the moderate (256KB) and large (512KB) categories. Pipeline depth has increased from a moderately deep pipeline of 14 stages for the Pentium Pro through Pentium III to a Pentium 4's longer pipeline of 20 stages. These abstractions ignore microarchitectural differences between the different generations, but lack of detailed information makes it difficult to use these differences to distinguish between processors. We are primarily interested in studying processors similar to those of the Pentium Pro through Pentium 4 generations. A Pentium Pro is abstracted to a low FO4 (L), moderately sized cache (M), and a moderate pipeline depth (M) known henceforth as an LMM core. The derivation of other abstractions is performed in a similar manner. Table 2 highlights the fact that many design points remain unexplored. The assumption that the second core will be implemented using existing intellectual property limits the scope of study to those design points which have a real world correspondence.

We also set our simulation parameters of our abstracted processors to match the scaled Intel processors as closely as possible. Cache sizes and associativities, TLB sizes and associativities, BTB sizes and associativities, latencies, register sizes, and numbers of functional units were all chosen to match

the Intel processors found in Table 1. These parameters are found in Table 3. They are close to those of the actual processors, but they are not exact, especially in the area of machine width.

4.1 Evaluation Criteria

Our goal was to find a scaled processor that provides reasonable performance while significantly reducing both dynamic and static power consumption.

4.1.1 Core Selection

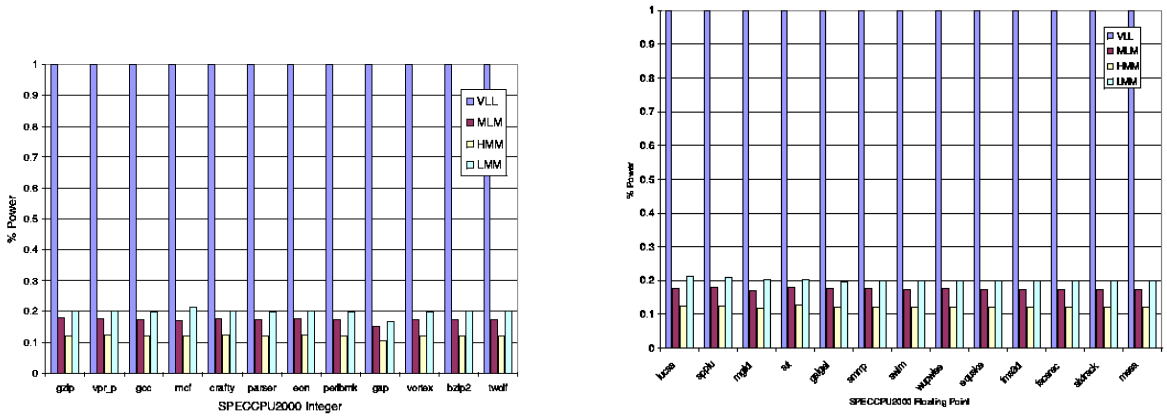
We present results for power, energy and energy delay products for all four processors we studied. Leakage power is also considered because of our desire to directly attack the problem of static power consumption. For these criteria we ran SPECCPU2000 on each processor under consideration. Maximum power dissipation per area is a measure of how the core temperatures may compare to one another. Throughput is an additional consideration and was used as our final arbiter. These results are used to select a core from our candidates.

In addition, these results can be used to illustrate situations where only the scaled core is run or situations where both the base core (VLL) and the scaled core are run. The first may occur in cases of thermal override, laptops operating off battery, or when the processing requirements are known to be less than those of the base processor. The second situation occurs when both core are run to achieve additional performance or when classes of applications are offloaded to the less powerful core.

4.1.2 Thermal Density Reduction

We also consider a scenario in which a single execution stream is switched between the two cores. This is similar to a symmetric design where instructions are scheduled for a time to the first core and later scheduled on the second core. The two cores are not in use simultaneously except during whatever time may be necessary to exchange information about the architectural state. It also presents an alternative to the situation where a complexity adaptive processor adjusts processor components, data structures or algorithms to meet the less intensive needs of an application.

We study a worst case situation in which our processors swap back and forth $1,000$ times per second. This is ten times a common operating system scheduling quanta, i.e., every millisecond rather than every ten milliseconds. Sherwood, et al., found that time varying behavior of applications could be seen when application behavior was examined on a 100 million committed instruction interval [24]. Many complexity effective design techniques respond to more frequent changes that occur on the order of tens of thousands of cycles to millions of cycles. Aide de Camp's slowest configuration (HMM) switches every 800,000 cycles on a one millisecond switch. Aide de Camp's fastest configuration



(a) Integer Benchmarks (b) Floating Point Benchmarks

Figure 2: Power Consumption of Scaled Processors on SPECCPU2000

(VLL) switches 2.8 million cycles on a one millisecond switch. Both are within an acceptable range for response intervals.

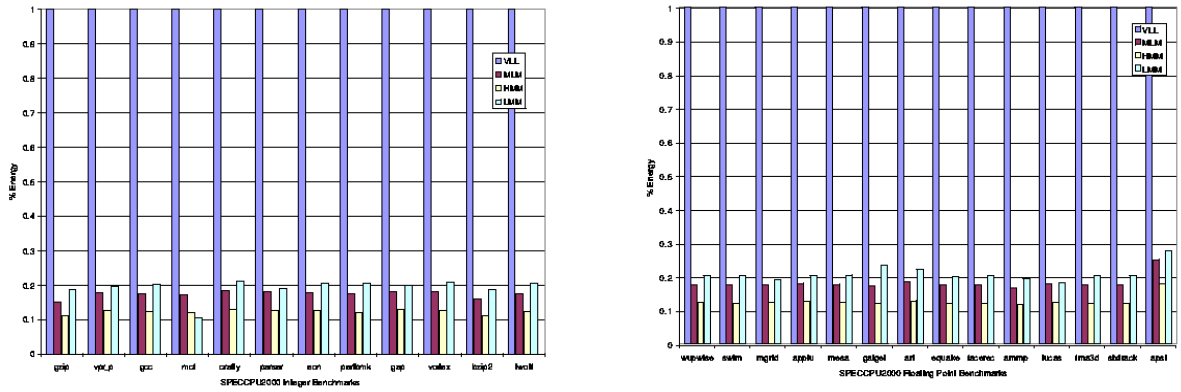
We examine two cases. In the first case, the powered down processor snoops the other processor's L2 cache. In the second case, the new processor must go to main memory for all of its cache misses. No work is done during L1 ICache and L1 DCache misses. The results presented come from two separate runs that have been interwoven to emulate switching between cores. Switching overhead has been added for each switch to take into account an upper bound on the amount of lost time incurred per switch. The simulations will be rerun when the Aide de Camp v2 simulator is completed.

We do not consider policies for determining *when* switching should occur. These have been studied by many others and can be adapted to the current situation. We analyze a worst case scenario to find the lowest bound on possible performance.

5 Results

5.1 Core Selection

We find the LMM core makes an excellent choice for Aide de Camp. It occupies a small area on the die, has a low complexity in comparison to more modern designs, and can be driven at a reasonable frequency. Its moderately sized caches are adequate to support its processing needs. It provides good savings in power and energy. It also has an energy-delay product of 46% of the primary VLL core. These results are explained in more detail below.



(a) Integer Benchmarks (b) Floating Point Benchmarks

Figure 3: Energy of Scaled Processors on SPECCPU2000

5.1.1 Power, Energy and Energy Delay Product

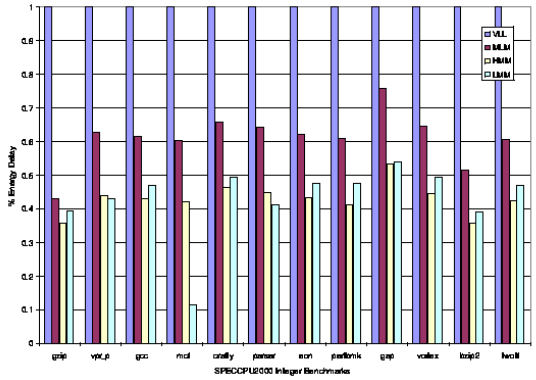
Figures 2 (a) and (b) show the per cycle power for the studied scaled processors. All results are normalized against the per cycle power of a 0.13 μ VLL design. The HMM design has the lowest simulated power, but both the LMM and MLM designs also perform well when considered against the base VLL processor. This result is true across SPECCPU2000 integer and floating point benchmarks. Across all benchmarks we find the normalized power costs to be 17% for a MLM core, 12% for a HMM core, and 20% for a LMM. All of these provide significant power savings over the VLL core.

Although power is an important consideration for packaging, it is also necessary to consider energy for total battery life. Figures 3 (a) and (b) report the *energy* needed for an application and thus take into account the longer time it takes to complete an application using a slower processor. The HMM core provides excellent energy savings. The MLM and LMM cores do less well, but still consume much less energy than the VLL core for a given application. On average, the MLM core uses 18% of the energy of the VLL core. The HMM core uses only 12% and the LMM core uses 20%.

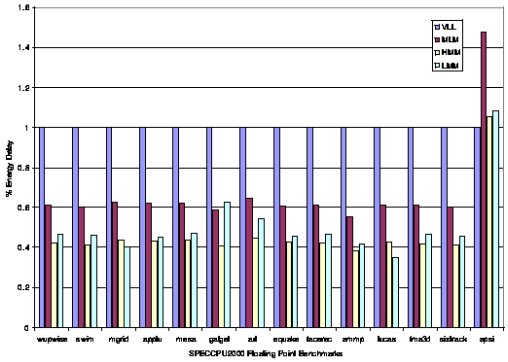
The energy-delay product can be used to estimate how much performance has been traded off to save power. This trade-off should be minimized because of the impact it can have on user experience. Figures 4 (a) and (b) indicate all three cores have energy delay products that are significantly lower than the VLL core. The LMM and HMM core have energy delay products of 46% and 45% of a VLL core. The MLM core trades off slightly more performance for its energy reduction, but still produces an energy delay product that is 65% of a VLL core.

5.1.2 Static Power and Leakage

Another important criteria to consider when selecting a scaled processor for ADC is its success in reducing the leakage power and energy. Most work takes into account only power, but for this work it

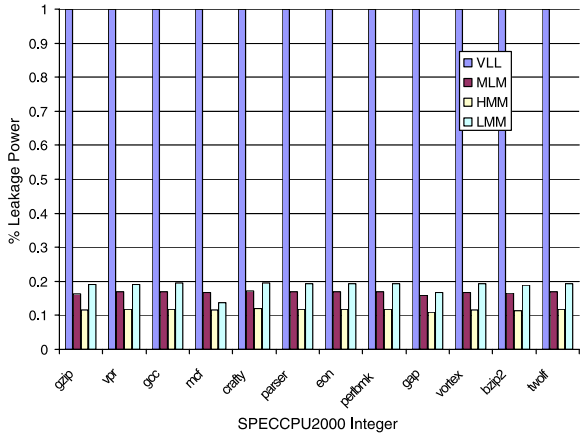


(a) Integer Benchmarks

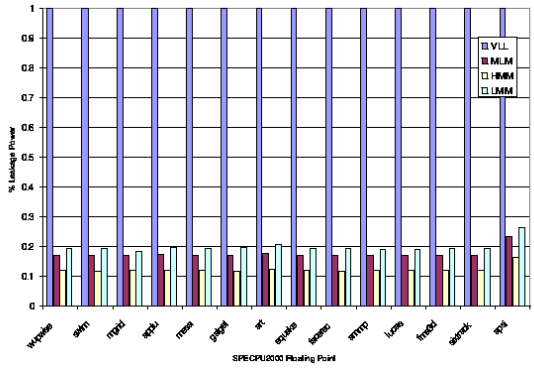


(b) Floating Point Benchmarks

Figure 4: Energy-Delay Products of Scaled Processors on SPECCPU2000



(a) Integer Benchmarks



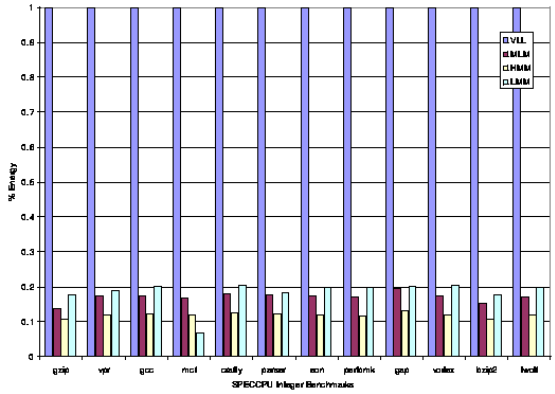
(b) Floating Point Benchmarks

Figure 5: Static Power Consumption of Scaled Processors on SPECCPU2000

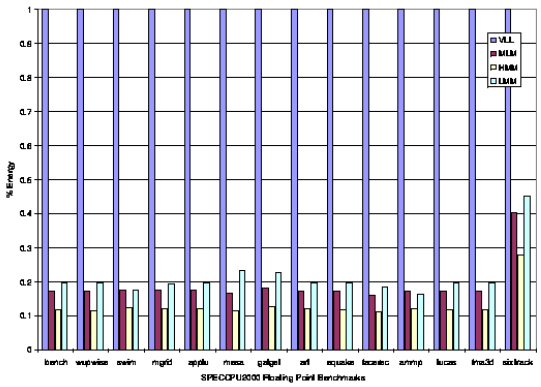
is necessary to consider leakage energy as well because design decisions lead to intentionally running more slowly. By increasing the duration of an application's lifetime, its leakage energy may also be increased, even if its leakage power is decreased.

Figures 5(a) and 5(b) show the HMM core reduces the static power the most. On average, it loses only 12% of the leakage power that is lost by the primary VLL core. The MLM and LMM cores lose 17% and 19% respectively. This is somewhat misleading because while the leakage power lost is considerably less, it is proportionally the same across all the processors considered. The LMM core loses the least with static power contributing 57% to the total power, while the VLL core represents the other end of the range where its static power contributes 61% to the total power cost. The range of values is easily within the simulator error of Wattch, especially when the non-ideal match between real-world processors and our simulator is taken into account.

Leakage energy takes into account the low clock frequencies of our scaled processors. Less energy

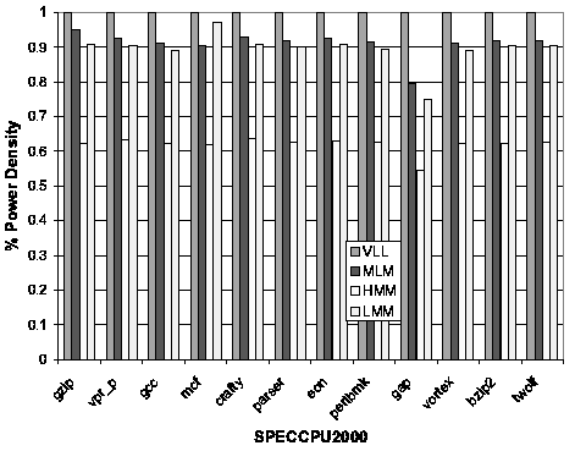


(a) Integer Benchmarks

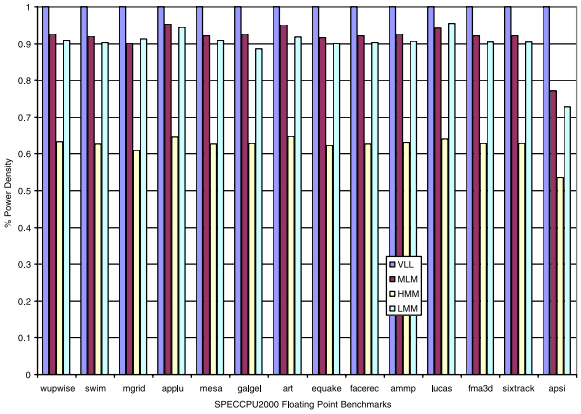


(b) Floating Point Benchmark

Figure 6: Static Energy of Scaled Processors on SPECCPU2000



(a) Integer Benchmarks



(b) Floating Point Benchmark

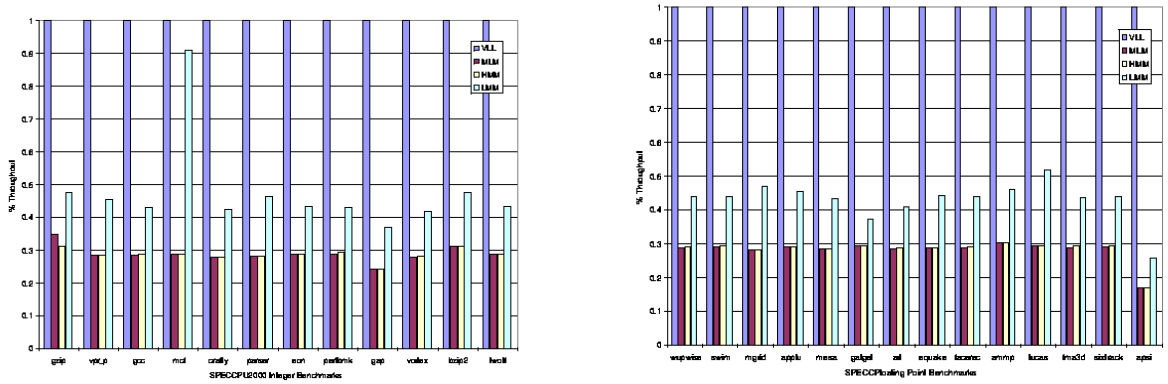
Figure 7: Power Densities of Scaled Processors on SPECCPU2000

wasted due to leakage is desirable. Figure 6 shows all three candidate cores reduce the amount of leakage energy. The LMM and MLM cores waste only 20% and 19% of the leakage energy of a VLL core, respectively. The HMM core does even better, wasting only 13% of the leakage energy of a VLL core.

5.1.3 Power Density

Our initial power density results indicate that any of the less powerful cores have a reduced power density when compared to a VLL core. The MLM and LMM cores both reduce the power density by about 10%. The HMM core reduces it by approximately 35%. The results are shown in Figures 7 (a) and (b).

Three of the four simulated processors had power densities within the margin of error of the simu-



(a) Integer Benchmarks (b) Floating Point Benchmark

Figure 8: Throughput relative to the VLL core on SPECCPU2000 Benchmarks

lator so it is not possible to say with certainty that the MLM and LMM cores would provide benefit.

Although the ADC processor uses less energy and will thus dissipate less power overall, thermal stress may still arise from the thermal density. This means there is the potential that thermal situations may simply move to the second processor when the first processor is brought down. However, the reduced clock speed and lower transistor counts of the second core alleviates this problem to some degree. We will revisit the issue in future work.

5.1.4 Throughput

Throughput can be used as a final arbiter when possible solutions are similar. If other metrics are comparable, the core with the highest throughput should be selected because it completes the most work per unit time. Figure 8 indicates the LMM core has the highest throughput of the three candidate cores. It achieves a throughput of 45% of a VLL core. The MLM and HMM cores both achieve throughputs of only 29% when compared to a VLL core.

5.1.5 Aide de Camp Core Selection

Based on the above results, we believe an LMM core will provide a reasonable second processor. It meets the requirements of reduced power, energy and leakage. It also presents a better user experience because it does not slow down processing as much as the other two processors under consideration. Its higher clock speed and lower complexity, as evidenced by its low FO4, makes it a good choice for the ADC processor. The above results indicate the best choice may depend on the specific design goals that led to the use of an ADC processor. If thermal density reduction had been the primary goal, a different selection would have been made.

5.2 Thermal Density Reduction – Striping

We present only the results for the ADC processor selected above, a design which contains an LMM core and a VLL core, and compare it against both a "scaled" LMM processor and a VLL processor. Wilf Pinfold at Intel has proposed using a striping technique across two identical cores to reduce thermal densities [17]. We instead studied the effect of striping across the two asymmetric cores of ADC. We are interested in these results for two reasons. The first is to study the efficacy of using this technique to reduce the thermal density. We do not have enough results yet to say with certainty that this has reduced the thermal density, but it has reduced the power density. The power model still needs to be extended to support the idea that the unused core cools during the time it is unused. The power results included below are a measure of the thermal density, but without a notion of cooling they are potentially misleading.

Second, we are interested in how many cycles are lost in the worst case when scheduling decisions are made based on time intervals, rather than on changing application requirements. For this work, we assume scheduling changes between the two processor modes (running on fast CPU *vs.* running on the slower CPU) occur at fixed intervals. Given this assumption the "worst case" behavior would be for the scheduler to switch from one scheduling mode (*e.g.* use slow CPU) to the other mode (*e.g.* use fast CPU). This incurs the maximum amount of overhead due to switching execution state from one CPU to the other. The results from this second goal are presented below.

5.2.1 Filling the L1 Cache from L2

In this case, the two processors share L2 cache data by snooping each other's L2 traffic. Processor A powers down everything, except the L2 cache which snoops Processor B's L2 bus traffic to maintain consistency. Although this means both L2 caches are always used, the contribution of the second level cache to the overall power was shown to be low in our simulations. Results from the Wattch simulator indicate only 3% of the VLL core's total power is consumed by the L2 cache. In the LMM core, the L2 consumes only 2% of the total.

Figure 9 indicates there can be a significant power savings involved in using ADC as a replacement for more conventional complexity adaptive techniques. These graphs use a switch rate of $1,000$ context switches between the cores per second. Since we are evaluating this technique in the absence of a particular scheduling policy, these results show the performance and energy when execution simply switches from one CPU to the other. This results in 22% less power than using the simulated VLL core alone. Figure 10 shows the energy reduction achieved by using ADC. On average, the energy savings was 44%. Figure 11 demonstrates that even with the performance trade-off involved, the energy-delay product is, on average, 76% of that of a VLL core.

These results indicate that it is possible to switch at a moderate frequency without regard to pro-

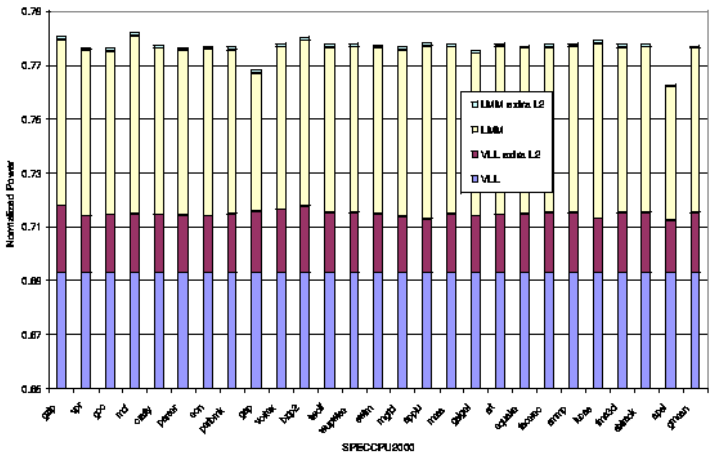


Figure 9: ADC Power Consumption when filling from the L2 Cache

gram characteristics and still achieve good power, energy and energy-delay product savings; an intelligent scheduling policy could adjust the schedule to either increase performance or reduce energy. The VLL core contributes much of the overall work completed, but the performance penalty for running with an ADC processor is not large enough to offset the power savings. We also found that the switching time is insignificant in comparison to the useful work time at this switching rate.

5.2.2 Filling the L1 Cache from Main Memory

We previously showed that it was possible to run ADC when switching between processors while leaving the L2 cache powered on. Here we consider the efficacy of turning off the L2 of the non-running processor. It is now necessary to refill the L1 caches from main memory, which has a larger overhead than refilling the caches from the L2 cache. The application is still run on both processors using scheduling which disregards program characteristics.

Figures 12-14 show power, energy and energy-delay products comparable to those of using an L2 cache to refill the L1 caches because the overhead time is still insignificant in comparison to the overall work time. Filling from main memory actually performs marginally better than filling from the L2 cache because it has slightly lower power and energy needs. This design alternative does not have to constantly power both L2 caches, slightly reducing power.

As memory latencies increase, we believe that a design that shares a single L2 cache would provide the best power and performance tradeoff. However, we wanted to explore the possibility of using complete designs rather than changing the L2 cache. In any case, our experiments show that either configuration appears to provide reasonable power performance tradeoffs.

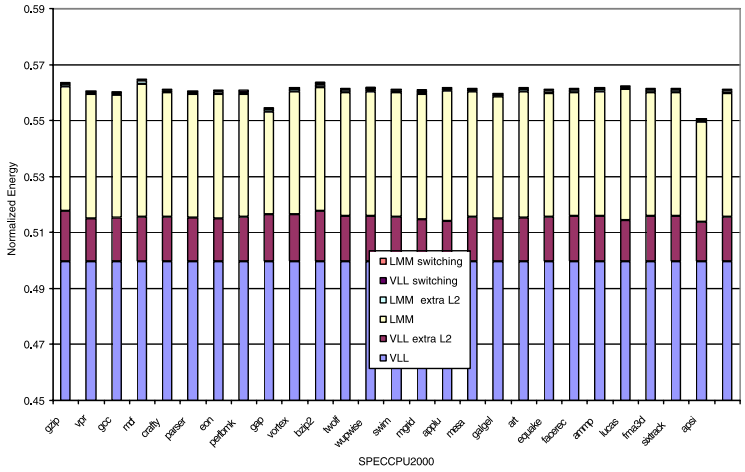


Figure 10: ADC Energy when filling from the L2 Cache

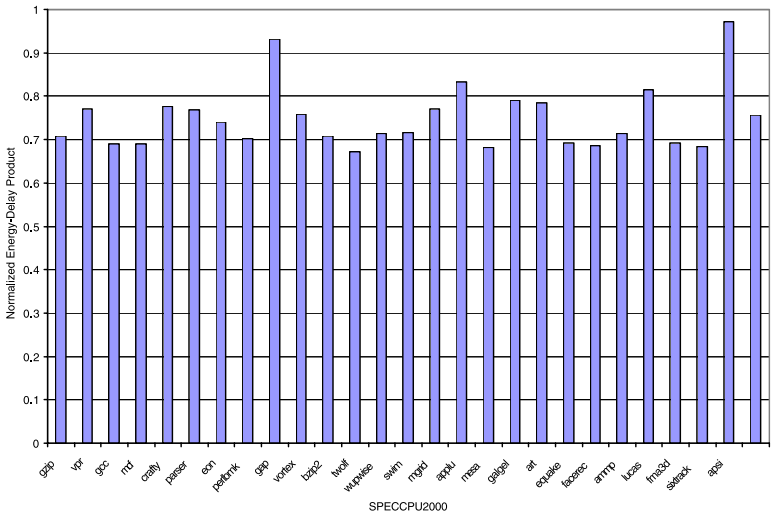


Figure 11: ADC Energy-Delay Product when filling from the L2 Cache

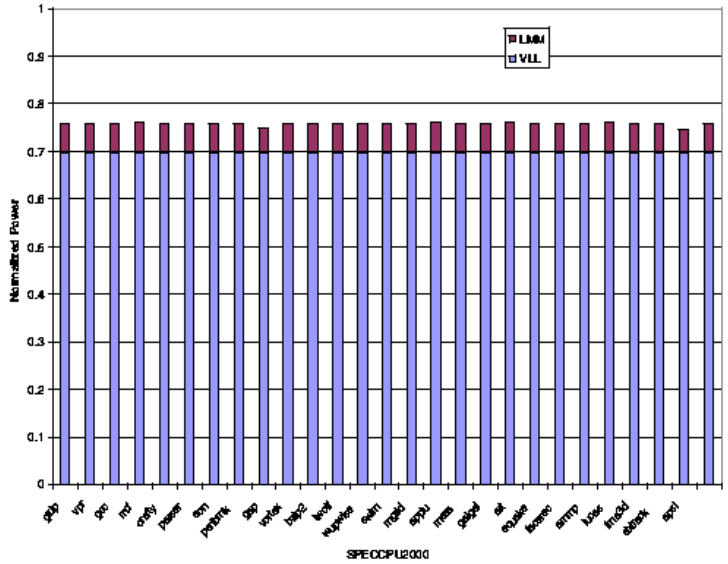


Figure 12: ADC Power Consumption when filling from Main Memory

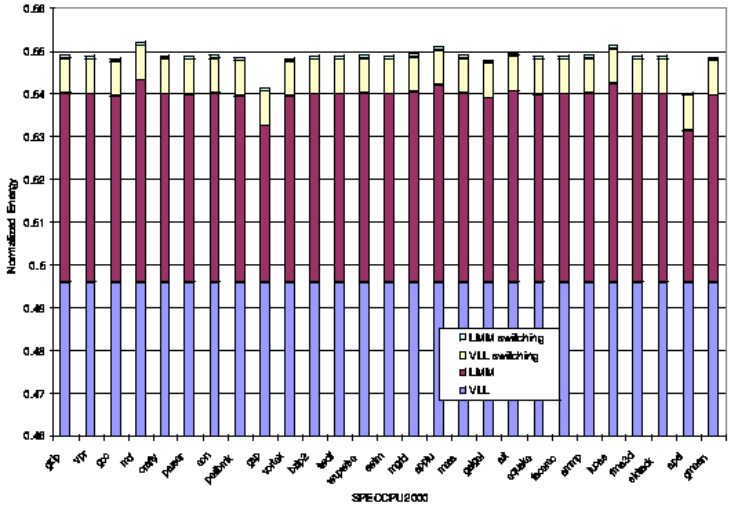


Figure 13: ADC Energy when filling from Main Memory

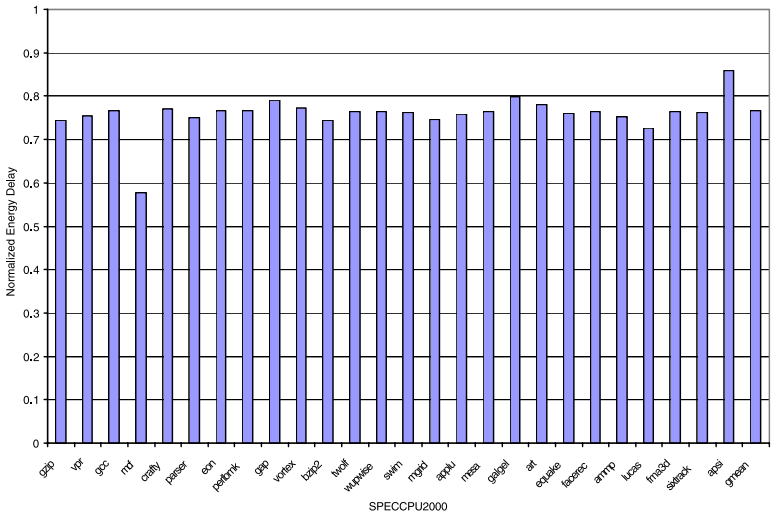


Figure 14: ADC Energy-Delay Product when filling from Main Memory

5.2.3 Switching Overhead

Finally, we consider the problem of switching overhead. Our above results were based upon a switching rate of $1,000$ per second, or every millisecond. As the frequency of switching increases, the amount of time available to perform useful work decreases. It is important to consider cases where more frequent switching occurs because microarchitectural-directed adaptations can occur much more frequently than OS-directed adaptations. ADC must be able to address the adaptation needs that have been found by prior work in complexity adaptive designs.

Figure 15 demonstrates this trade-off on the energy-delay product of switching between the two processors in both configurations. This graph shows the results for the "gzip" benchmark. Alternating execution using ADC is effective up until about $10,000$ switches per second when filling the L1 from memory. At that point, the cost of the additional overhead for switching between processors begins to overwhelm the energy savings of using ADC stripping. With $10,000$ switches per second, VLL core can switch every $230,000$ cycles and the ADC LMM core to switch every $100,000$ cycles. Because this is a worst case analysis, more frequent switching may be possible; in particular, more frequent scheduling would be possible when filling the L1 cache from L2 rather than memory, as seen in Figure 15. When filling the L1 caches from the L2 cache, it is possible to achieve about $15,0000$ switches per second.

This worst case analysis shows it is possible to run ADC with frequent switches. A more realistic analysis would allow some work to be completed while the L1 caches are filled by misses. This will be handled by a later version of the Aide de Camp simulator. Even in this case there is a limit to the frequency of switching because the useful work time does eventually become zero as the switching frequency increases.

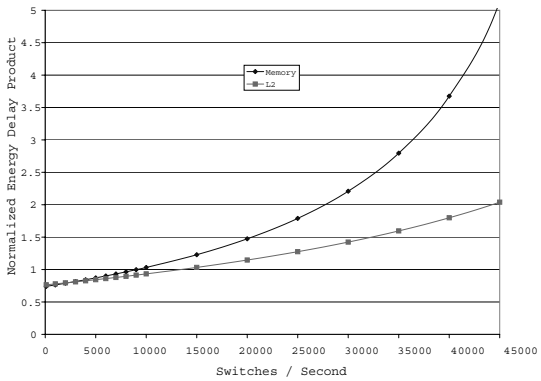


Figure 15: ADC Energy-Delay Product And Switching Frequency

6 Conclusions

Our preliminary results indicate Aide de Camp provides a versatile solution to power problems. It is able to significantly reduce the power and energy demands of an application without increasing the energy delay product. It also is able to partially address the problem of static power consumption by reducing the total leakage power lost. Our results indicate that for current process technologies, a design leveraging a low FO4 load, moderately sized caches and a moderately deep pipeline can be scaled to the current process and paired with a modern processor.

ADC can be run in a variety of different situations and modes to match those situations. It may be run as a single processor to limit thermal shutdown or to extend battery life. In such situations we found that our ADC processor can reduce average power consumption and energy usage by 80% and has an energy delay product close to that of a design similar to the Pentium 4. ADC also reduces the leakage power and energy by 80%.

ADC may also be run as a complexity-adaptive mechanism with different policies for assigning an application to a processor. Even in the situation where an application is striped across two processors to reduce the power density, ADC provides benefit until the frequency of switching becomes too high to hide the overhead. We propose two possible solutions to maintaining L1 cache contents, either filling L1 data from the L2 cache or the memory system. We find that we are able to reduce power by 22%, energy by 44% and the energy-delay product by 24%. These results used an oblique scheduling policy; an informed scheduling policy would likely perform better.

Aide de Camp shows considerable promise. We plan to extend our work in this area by focusing on its use in controlling thermal overloads.

References

[1] T. R. Puzak A. Hartstein. The optimum pipeline depth for a microprocessor. In *29th Annual International Symposium on Computer Architecture*, May 2002.

[2] D. Albonesi. Dynamic IPC/Clock Rate Optimization. In *Proceedings of the 25th International Symposium on Computer Architecture*, pages 282–292, Barcelona, Spain, June 1998.

[3] R.Iris Bahar and Srilatha Manne. Pipeline Gating: Speculation Control for Energy Reduction. In *Proceedings of the 28th International Symposium on Computer Architecture*, pages 218–229, Goteberg, Sweden, June 2001.

[4] A. Bhavnagarwala, A. Kapoor, and J. Meindl. Dynamic-threshold cmos sram cells for fast portable application. In *13th Annual IEEE International ASIC/SOC Conference*, 2000.

[5] David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimization. In *Proceedings of the 27th International Symposium on Computer Architecture*, pages 83–94, Vancouver, Canada, June 2000.

[6] D.C. Burger and T.M. Austin. The SimpleScalar Tool Set, Version 2.0. *Computer Architecture News*, 25(3):13–25, 1997.

[7] J. Adam Butts and Gurindar S. Sohi. A static power model for architects. In *International Symposium on Microarchitecture*, pages 191–201, 2000.

[8] A. Buyuktosunoglu, S. Schuster, D. Brooks, P. Bose, P. Cook, and D.H. Albonesi. An Adaptive Issue Queue for Reduced Power at High Performance . In *Workshop on Power Aware Computer Systems*, Boston, November 2000.

[9] K. Flautner, S. Reinhardt, and T. Mudge. Automatic performance setting for dynamic voltage scaling. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MOBICOM)*, July 2001.

[10] D. Folegnani and A. Gonzalez. Reducing Power Consumption of the Issue Logic. In *Workshop on Complexity Effective Design*, Vancouver, Canada, June 2000.

[11] F. Hamzaoglu, Y. Ye, A. Keshavarzi, K. Zhang, S. Narendra, S. Borkar, and V. De M. Stan. Dual-vt sram cells with full-swing single-ended bit line sensing for high-performance on-chip cache in 0.13 um technology generation. In *IEEE Symposium on Low Power Electronics*, 2000.

[12] Seongmoo Heo and Krste Asanovic. Dynamic fine-grain leakage reduction using leakage-biased bitlines. In *29th Internation Symposium on Computer Architecture*, 2002.

[13] Seongmoo Heo and Krste Asanovic. Leakage-biased domino circuits for dynamic fine-grain leakage reduction. In *IEEE Symposium on VLSI Circuits*, 2002.

[14] Ron Ho, Kenneth W. Mai, and Mark A. Horowitz. The Future of Wires. In *Proceedings of the IEEE, VOL. 89, NO. 4*, pages 490–504. IEEE, April 2001.

[15] M.S. Hrishikesh, Norman P. Jouppi, Keith I. Farkas, Doug Burger, Stephen W. Keckler, and Premkishore Shivakumar. The optimal useful logic depth per pipeline stage is 6-8 fo4. In *29th Annual International Symposium on Computer Architecture*, May 2002.

[16] K. Itoh, A.R. Fridi, A. Bllaouar, and M.I. Elmasry. A deep sub-v, single power-supply sram cell with multi-vt, boosted storage node and dynamic load. In *IEEE Symposium on VLSI Circuits*, 1996.

[17] Michael Kanellos. At Intel - the chip with two brains. *C—Net news.com*, Aug 2002.

[18] Jacob Lorch and Alan Jay Smith. Improving Dynamic Voltage Scaling Algorithms With PACE. In *Proceedings of the ACM SIGMETRICS '01 International Conference on Measurement and Modeling of Computer Systems*, 2001.

[19] S. Manne, A. Klauser, and D. Grunwald. Pipeline Gating: Speculation Control for Energy Reduction. In *Proceedings of the 25th International Symposium on Computer Architecture*, pages 122–131, Barcelona, Spain, June 1998.

[20] Trevor Pering, Tom Burd, and Robert Brodersen. The Simulation of Dynamic Voltage Scaling Algorithms. In *IEEE Symposium on Low Power Electronics*, 1998.

[21] Trevor Pering, Tom Burd, and Robert Brodersen. Voltage Scheduling in the lpARM Microprocessor System. In *Proceedings of the 2000 International Symposium on Low Power Design*, August 2000.

[22] Padmanabhan Pillai and Kang G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *Proceedings of the 18th Symposium on Operating Systems Principles*, October 2001.

[23] M.D. Powell, S.H. Yang, B. Falsafi, K. Roy, and T.N. Vijaykumar. Gated-vdd: A circuit technique to reduce leakage in deep-submicron cache memories. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 2000.

[24] T. Sherwood and B. Calder. Time Varying Behavior of Programs. Technical Report UCSD-CS99-630, Dept. of Computer Science and Engineering, University of California San Diego, August 1999.

[25] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for Reduced CPU Energy. In *First Symposium on Operating Systems Design and Implementation*, pages 13–23, November 1994.

[26] Se-Hyun Yang, Michael D. Powell, Babak Falsafi, Kaushik Roy, and T. N. Vijaykumar. An integrated circuit/architecture approach to reducing leakage in deep-submicron high-performance i-caches. In *HPCA*, pages 147–158, 2001.

[27] S.H. Yang, M.D. Powell, B. Falsafi, K. Roy, and T.N. Vijaykumar. An integrated circuit/architecture approach to reducing leakage in deep-submicron high-performance i-caches. In *High Performance Computer Architecture*, 2001.