Optimization of High-Performance Superscalar Architectures for Energy Efficiency*

V. Zyuban and P. Kogge
Computer Science & Eng. Dept., University of Notre Dame, IN 46556, USA

Abstract

In recent years reducing power has become a critical design goal for high-performance microprocessors. This work attempts to bring the power issue to the earliest phase of high-performance microprocessor development. We propose a methodology for power-optimization at the micro-architectural level. First, major targets for power reduction are identified within superscalar microarchitecture, then an optimization of a superscalar micro-architecture is performed that generates a set of energy-efficient configurations forming a convex hull in the power-performance space. The energy-efficient families are then compared to find configurations that dissipate the lowest power given a performance target, or, conversely, deliver the highest performance given a power budget. Application of the developed methodology to a superscalar micro-architecture shows that at the architectural level there is a potential for reducing power up to 50%, given a performance requirement, and for up to 15% performance improvement, given a power budget.

Introduction

Until heat removal from a workstation micro-chip became a major issue, high performance microprocessors remained free of power conservation requirements. As a result, microprocessor architecture tradeoff decisions have traditionally been based on cost-performance analysis. Such features as wide out-of-order issue, speculation, register renaming, multiple branch prediction, multi-level caches, memory disambiguation, etc. have become standard, and little work has been done so far to improve the energy efficiency of structures providing these architectural features.

This work attempts to bring the power issue to the earliest phase of high-performance microprocessor development, where there is still a significant potential for power reduction.

Since performance is the main factor that sells in this high-end processor market, solutions compromising performance, such as those used in low power embedded microprocessors, are not an option. It is the goal of this paper to explore architectural level solutions to the power problems in high-performance superscalar microprocessors that achieve a power reduction without compromising performance.

The organization of the paper is as follows: Sections 1 analyses the critical design points of a superscalar microarchitecture and discusses energy models. Section 2 applies the energy model to the superscalar architecture and describes the developed optimization procedure along with obtained results. Section 3 suggests an architectural solution to the power growth problem and evaluates the potential of the proposed technique. Section 4 summarizes the paper.

*This work was supported in part by the National Science Foundation under Grant No.MIP–95–03682.

ISLPED '00, Rapallo, Italy.

1 Energy Models

1.1 Key power consumers in superscalar micro-architecture

Future growth in performance for modern superscalar CPUs is predicated on higher and higher widths of instruction issue. Therefore, when doing power analysis of high-performance future microprocessors, particular attention should be given to those structures in a microarchitecture where energy dissipation per instruction grows with increasing issue width. Among them are [10]: rename logic which includes a map table, storing logical-to-physical register translations; issue window which keeps track of dependencies between instructions and stores instructions until they are ready for issue; memory disambiguation unit which keeps track of load-store dependencies and allows out-of-order issue of memory instructions; data bypass mechanism and multiported register file.

The baseline architecture model, shown in Fig. 1, is similar to the micro-architecture of the MIPS R10000 [14], Alpha 21264 [4] and HP PA-8000 [8]. We chose this scheme for its scalability and logical simplicity [15]. In terms of dissipated energy it is comparable to the reservation station model [11], used in other high performance out-of-order processors such as Intel Pentium Pro, PowerPC, and SPARC64.

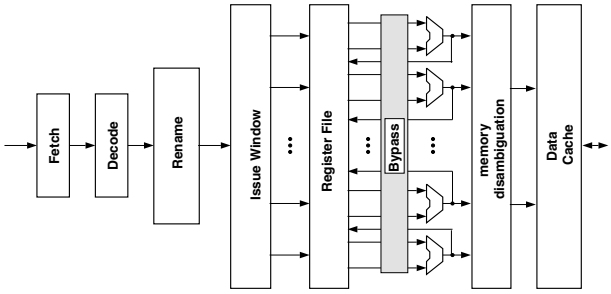

Figure 1: Baseline superscalar model.

Most of the structures listed above include multiported memory macros whose storage size and number of ports grow with increasing instruction-level parallelism. For example, the number of read ports from the register file and from the register rename map table is a product of the number of read operands per instruction and the issue width, and the number of ports from the issue window matches the issue width. Also, the number of entries in the physical register file, issue window, and memory disambiguation unit tends to grow approximately linearly with the issue width to support higher degree of out-of-order execution, [3]. Taking into account growth both in storage size and the number of ports, we expect that the power portion of structures that include multiported on-chip memories will grow rapidly in the future.

The above list does not include any of the functional units because their complexity is independent of the issue width. Nevertheless, we need to have simple and reliable energy models for the functional units, because much of the CPU power is still dissipated there.

1.2 Energy models

In [15] we developed energy models for the key structures discussed above. The models express energy for all possible types of accesses to each structure in terms of micro-architectural parameters. Different types of accesses typically dissipate different amounts of energy. For example, there is a read and write access energies in the register file [16], while the issue window has in addition a *match* access, when an instruction generating a result broadcasts the result tag to all instructions in the issue window. There are even more access types to the memory disambiguation unit [15].

These models have been incorporated into a very detailed architectural-level simulator [15], written based on the SimpleScalar tool set [1]. The simulator measures the architectural speed of a simulated out-of-order superscalar processor and counts the exact numbers of accesses of every type to every structure in the micro-architecture. The data measured (over SPEC95) are then applied to the energy models to estimate the average energy dissipated per instruction, as well as total power of a simulated processor.

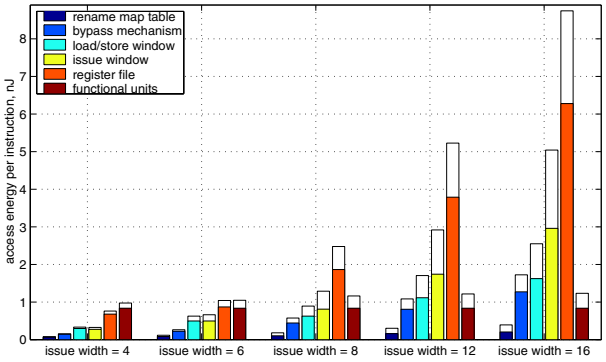

Figure 2: Average energies dissipated per committed instruction (measured on SPEC95); 0.35μ feature size, $V_{dd} = 3.3V$.

Fig. 2 shows the results of applying the energy models to five superscalar designs with issue width ranging from 4 to 16. The shaded bars represent the energy per instruction, assuming that every fetched instruction is committed. In a real processor with speculation instructions fetched from mispredicted paths are flushed from the pipeline at various stages. The white bars show the total energy per committed instruction, taking into account the energy dissipated by instructions flushed from the pipeline. Notice that the portion of energy dissipated by flushed instructions grows with increasing issue width, because wider processors rely on deeper speculation to exploit more Instruction Level Parallelism (ILP).

As can be seen from Fig. 2, the energy per instruction grows faster than linearly with the issue width for all the key structures, while for functional units it grows much slower, the growth being determined by the increasing portion of instructions flushed from the processor.

Table 1: Estimated energy growth parameters.

Structure	Energy growth parameter
register rename logic	$\gamma = 1.1$
instruction issue window	$\gamma = 1.9$
memory disambiguation unit	$\gamma = 1.5$
multiported register file	$\gamma = 1.8$
data bypass mechanism	$\gamma = 1.6$
functional units	$\gamma = 0.1$

The energy-per-instruction growth in each structures can be described as $E \sim (IW)^{\gamma}$, where IW is the issue width, and γ is an energy growth parameter, specific for every structure. Table 1 shows the energy growth parameters for these structures, based on a polynomial curve fit in Fig. 2.

2 Inherent Energy Inefficiency of Superscalar Design

2.1 Energy-Delay Product

The energy-delay product, $E \times D = \frac{energy}{instruction} \times \frac{cycles}{instruction}$, which can be expressed in terms of IPC as $E \times D = \frac{Energy/cycle}{IPC^2}$, is a reasonable metric for energy efficiency at the micro-architectural level [6]. We call a micro-architecture *energy-efficient* if its energy-delay metric does not grow with increasing architectural performance, IPC.

According to Table 1, the energy dissipated per cycle in every structure can be described as $Energy/cycle \sim IPC \times (IW)^{\gamma}$. By substituting this into the energy-delay formula, we get

$$E \times D = \frac{Energy/cycle}{IPC^2} \sim \frac{IPC \times (IW)^{\gamma}}{IPC^2} = \frac{(IW)^{\gamma}}{IPC}. \quad (1)$$

To reduce this expression we need to establish a relation between the issue width of a processor, IW and the architectural speed, IPC. Assuming that $IPC = IW^{\alpha}$ yields

$$E \times D \sim (IW)^{\gamma - \alpha} \quad (2)$$

If the IPC grew linearly with the issue width ($\alpha = 1$) then to achieve *energy-efficiency* it would be sufficient that the energy growth parameters of all structures be no higher than one, $\gamma \leq 1$. For example, the register file built using the Port Priority Selection technique combined with differential reads and low swing write, according to [16], would be considered an *energy-efficient* design, because the energy growth parameter for this design was found in [16] to be $\gamma = 0.96$. On the other hand, the register file built using the conventional technique would not be energy efficient, because, according to Table 1, its energy growth parameter is $\gamma = 1.8$.

However, since in real machines IPC increases less than linearly with an increase in the issue width ($\alpha < 1$), the condition $\gamma \leq 1$ may not be sufficient for energy efficiency. In fact, for an Amdahl's law-like α of 0.5 [12, 15] we have:

$$E \times D \sim (IW)^{\gamma - \frac{1}{2}} \sim (IPC)^{2\gamma - 1} \quad (3)$$

This formula shows that only those structures that have $\gamma \leq 0.5$ are energy efficient. Thus, even the use of the most energy-efficient register file circuitry known to the authors [16] that reduces access energy at a cost of much higher design complexity does not solve the power growth problem, since it leaves the energy dependence parameter γ well above 0.5.

2.2 Energy-Efficiency Optimization Methodology

There are many parameters involved in the micro-architecture, and different combinations of architectural parameters result in design points with different performance and energy efficiency. This section describes our methodology to optimize the superscalar architecture for energy efficiency.

We found that it is practically impossible to determine a single optimal design point for an architecture, because the optimality criteria highly depend on the orientation of a processor. There can be high-end configurations targeted at achieving the maximum performance. On the other hand, there can be low-end configurations targeted at achieving the minimum energy dissipation per instruction. Between these two extremes, there are configurations targeted at achieving a reasonably high performance with a reasonably low power. To address this diversity of possible optimality criteria we performed an energy-performance optimization for the whole power-performance design trade-off space. In other words,

optimal configurations were sought for the whole range of optimization targets.

To better understand the energy-performance design trade-off space, it is helpful to define an *energy-efficient* configuration as a configuration that delivers the highest performance among all configurations dissipating the same power. An alternative definition is that it is the one that dissipated the least power among all configurations that deliver the same performance. The definitions are equivalent.

Thus, for every micro-architecture there is not just one optimal configuration, but a whole set of *energy-efficient* configurations, called hereafter an *energy-efficient family*. Each configuration from the *energy-efficient family* is optimal in terms of some energy-performance optimality criterion, $E \times D^{\beta}$, $\beta > 0$. If plotted on in power-versus-performance coordinates, energy-efficient configurations form a *convex hull* of all possible configurations of a given architecture.

Having the *energy-efficient family* of configurations can be very useful in making power-performance trade-offs, because it provides an insight to the power-performance design tradeoff space. For example, it allows a designer to immediately find the highest performance for a given energy budget, achievable for an architecture in study, or estimate the lowest power dissipation, given a minimal performance requirement.

Every particular configuration of a microarchitecture is specified by a large number of architectural parameters, for example our architectural simulator accepts about 50 input parameters. However, not all architectural parameters have equally strong effect on the energy-performance tradeoff. Selection of values for most of the parameters is affected by implementability issue.

We found that the most fundamental parameters that have a strong affect on both power and the out-of-order processor capabilities and, thus, performance, are the size of the physical register file, the size of the instruction issue window and the size of the load/store issue window (memory disambiguation unit). We treated these parameters as independent variables in the optimization process. The rest of the architectural parameters were set to the values that were found to be optimal over a wide region of the energy-performance tradeoff space, or to values selected for implementability considerations [15].

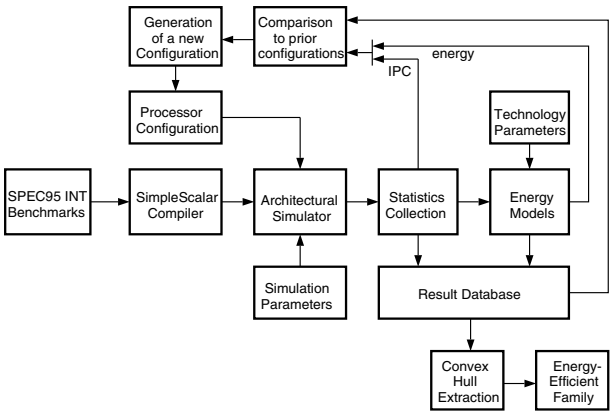

Figure 3: Methodology for constructing energy-efficient families.

To construct the energy-efficient configuration families we started with an initial configuration, and then used multidimensional optimization algorithms to find configurations minimizing the *energy-delay* product, *energy-delay-square* metric, and *energy-square-delay* metric. The configurations optimizing each of these metrics obviously belong to the energy-efficient family of the architecture in study. The *energy-delay* optimal configuration is the one that is optimal in the conventional sense of $\frac{Power}{Performance^2}$, as discussed above. The *energy-delay-square* optimal configuration represents a power conscious, performance oriented configuration, while the *energy-square-delay* optimal configuration represents a low-power oriented processor.

Upon obtaining these three energy-efficient design points, we scaled the architectures by linear interpolation. Local optimizations were further performed at every new point on the interpolation intervals. Then the energy-efficient family curves were extended through linear extrapolation beyond the *energy-delay-square* optimal points to see the maximum performance that every architecture is capable of delivering, and beyond the *energy-square-delay* optimal points to get the minimal power, if the performance is not an issue. Figure 3 summarizes this methodology.

In the energy-efficiency optimization process every evaluation involves running eight SPEC95 benchmarks on the architectural simulator, which takes about 18 hours on UltraSparc 300MHz workstation. Taking into account such a high cost of every evaluation, we used the Nelder-Mead simplex method [9] which typically requires only one or two evaluation per iteration.

Good intuition on the interdependence of architectural configuration parameters had been developed in the process of development of the simulator, resulting in good initial guesses for the optimization process. As a result, only from ten to twelve iterations were sufficient to obtain the *energy-delay* optimal, *energy-delay-square* optimal and *energy-square-delay* optimal points. This required approximately 50 evaluations. To complete the energy-efficient families another 50 evaluations were needed.

2.3 Results and Analysis

The most straightforward way to analyze the constructed energy-efficient families is to represent each configuration as a dot on the energy-per-cycle versus IPC graph. If multiplied by clock, the axes would transform to power and performance, respectively.

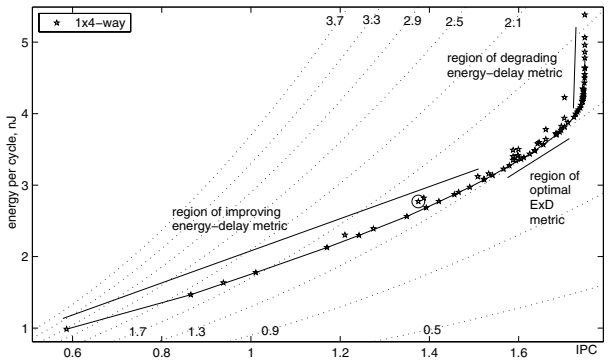

Figure 4: Energy-efficient family and all simulated configurations for the 4-way architecture.

Figures 4 and 5 present the energy-efficient families for the simulated 4-way, 6-way and 8-way superscalar architectures. All simulated configurations are shown with stars. Circled stars represent the initial guesses. Solid lines connect the points on the convex hulls, and, thus, represent the energy-efficient family curves.

Dotted lines correspond to constant energy-delay product, $E \times D = x\frac{nJ \times cycle}{instr^2}$, with x incrementing in steps of 0.4. In the energy-per-cycle versus IPC coordinates the constant energy-delay product lines are represented by parabolas. Points lying on lower equal energy-delay product lines represent configuration with lower energy-delay product. Charts for 12-way and 16-way superscalars look similar [15].

For all architectures the energy-delay product is very high for low-end configurations, because the processor functional resources are underutilized, and the delay term in the energy-delay product is high. Then, as we increase the out-of-order issue capabilities, the energy-delay product improves, until it reaches the minimum. At this point, the functional resources of the processor are well utilized, while the energy overhead of structures needed to support these out-of-order issue capabilities is reasonably low.

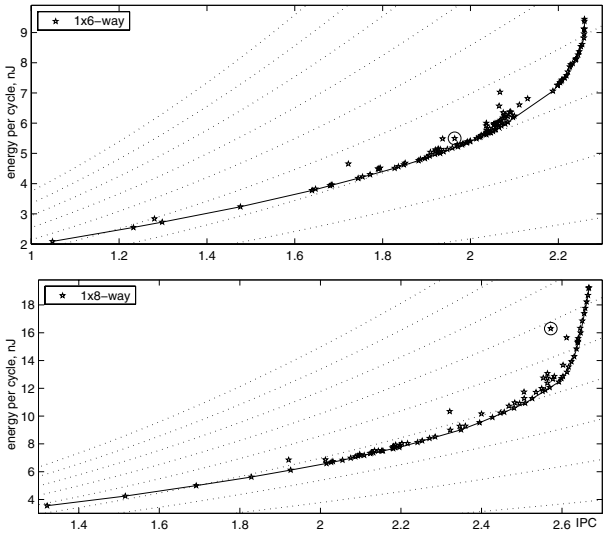

Figure 5: Energy-efficient family and all simulated configurations for the 6-way and 8-way architectures.

As we try to further increase the performance by pushing the out-of-order capabilities of the processor, the performance growth begins to saturate, while the energy overhead of structures needed to support the growing out-of-order capabilities explodes. Thus, pushing performance to the limit results in a very fast growth of the energy-delay product at the high-end portion of the energy-efficient family curves.

By looking at the positions of the circled stars, representing the initial guesses for the energy-efficiency optimization process, one can appreciate the value of energy-efficiency optimization. Most of the initial guesses turned out to be close to the optimal curves, because they were based on extensive analysis preceding the experiment. However, without performing the energy-efficiency analysis, a combination of architectural parameters picked up based on intuition or pure performance analysis may be as much as 30% off the optimal curve (which we found to be the case in many academic works), which means that there exists another configuration delivering the same performance, at 30% less power. By constructing the energy efficient family a designer can easily control how far a particular configuration is off the optimal curve.

We made a number of observations by analyzing the configurations on the energy-efficient family curves. First, the optimal sizes of the major structures, even if the energy-delay-square product is chosen for the optimization target, are significantly smaller than those assumed in the majority of academic architectural studies. The reason is that in most academic architectural studies all architectural parameters are pushed to achieve the limit performance, neglecting the power issue. This approach makes the resulting architectures very inefficient in terms of energy dissipation.

The optimal sizes of major architectural structures typically grow as the issue width of the processor increases, because the resulting enhancement of the out-of-order capabilities allows the processor to more fully take advantage of the growing issue capabilities and functional resources, and the resulting performance gain over-weighs the associated growth of the energy dissipated by th

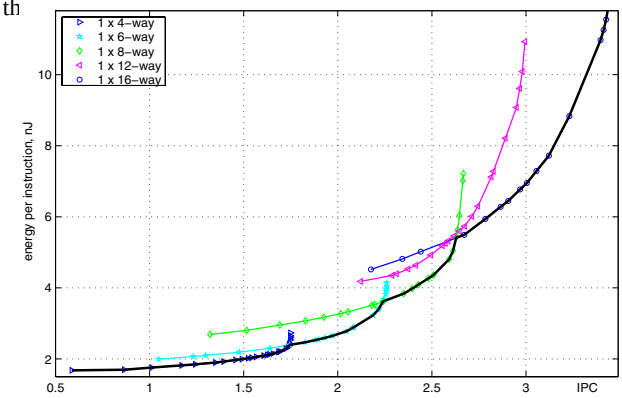

Figure 6: Global energy-efficient family for the centralized superscalar architecture.

Figures 4 through 5 indicate that configurations with a fixed issue width can work efficiently only over a certain performance range. To find what issue widths allow configurations that dissipate the least amounts of energy per instruction, given a performance target over the whole range of achievable IPC, we plotted on the same graph all the *energy-efficient* curves for architectures with a given number of clusters. Figure 6 graphs energy-per-instruction versus IPC for all the *energy-efficient* curves for all simulated micro-architectures.

Highlighted curves in the graphs on Figure 6 represent configurations with the lowest amounts of energy dissipation per instruction given a performance target. We will refer to the configurations on these highlighted curves as *global energy-efficient* families. A non-linear growth of the energy per instruction in Fig. 6 indicates that the centralized superscalar architecture, even when optimized for the energy efficiency, is inherently energy inefficient. The next section demonstrates the potential for improving the energy efficiency at the architectural level using a decentralization approach.

3 Improving Energy Efficiency through Decentralization

3.1 Multicluster Architectures

One way to address the energy growth problem at the micro-architectural level is to replace the today's tightly coupled superscalar CPU with a set of clusters, so that all key energy consumers are split between clusters. Ideally, instead of accessing centralized structures in the traditional superscalar design, instructions scheduled to individual cluster would access local structures most of the time. The primary advantage of accessing a collection of local structures instead of centralized ones is that the number of ports and entries in each local structure is much smaller, which makes them simple and low power.

The problem of the growth of centralized structures in a superscalar architecture has been recognized by some researchers primarily because of the associated increase in access time and implementability problems, and a few decentralized architectures have been proposed [10, 2, 5, 13, 3, 7]. However, access times have been the primary concern thus far, not energy costs.

In [15] we propose a new version of the multicluster architecture, tailored to achieving a higher energy efficiency. It addresses the access energy growth problem in the key problematic points of the superscalar micro-architecture, in particular, register file, issue window, memory disambiguation unit and bypass mechanism.

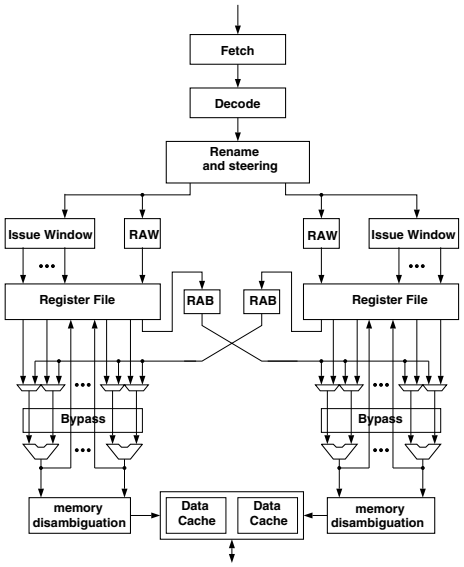

Figure 7: Overview of the proposed multi-cluster architecture.

We propose an architecture where today's tightly coupled superscalar CPU, Fig. 1, is replaced with a set of clusters, as shown in Fig. 7. Each cluster could be either a simple in-order single-issue machine, multiple-issue machine, or it could be an out-of-order machine. Our simulations showed, however, that in order to compensate for the overhead of inter-cluster communication and get a significant improvement in the energy-delay metric over the tradition superscalar processor, every cluster must be a powerful out-of-order superscalar machine by itself. The optimal number of clusters and their configurations have been determined by simulation targeted to the minimization of the energy-delay metric [15].

Each cluster, Fig. 7 is provided with a local instruction issue window, local physical register file containing a subset of physical registers, a set of execution units, local memory disambiguation unit, and one bank of the interleaved data cache.

Each architectural register named by an instruction is renamed to a physical register in one of the local register files. A global mapping table is used for renaming source registers. For the destination register named by an instruction a physical register is picked up from the free list of the cluster to which it is dispatched. A simple heuristic is used to assign instructions to clusters [15].

After renaming, the renamed instructions are dispatched to instruction windows in individual clusters. The availability of source operands mapped to the local register file is monitored in the conventional way. To monitor the availability of a source operand mapped to a remote register file, an appropriate request is placed to the Remote Access Window (RAW) corresponding to the register file holding the needed operands. Also, a free entry is allocated in the corresponding Remote Access Buffer (RAB) that will accept the value read from the remote register file through one of *remote access ports*, and will keep it until the instruction that needs it is issued.

Issued instructions get their source operands from either the local register file, or from the appropriate RAB. One clock cycle penalty is incurred for accessing a remote operand. This extra cycle of latency is the main reason why a multi-cluster architecture cannot reach the same IPC as a centralized architecture for the same total issue width and the same total sizes of major hardware resources. The other reason is that centralized resources are always better utilized that distributed ones.

The number of the remote access ports and the number of entries in the RAW and RAB have been determined for every configuration through the energy-efficiency optimization process. The typical inter-cluster traffic on SPEC95 benchmarks was measured to be about 0.2 remote register file access per instruction, and from 1 to 3 remote access ports were found be sufficient to handle it. The optimal number of entries in RAW and RAB were found to be in the range from 4 to 8.

Memory instructions are allocated entries in the disambiguation unit of the cluster to which they are issued. The selection of a cluster to which a memory instruction is dispatched is based on the memory bank prediction, done in the decode stage.

3.2 Energy-Efficiency of Multicluster Architecture

To optimize the multicluster architecture for energy efficiency we used the same optimization methodology, Fig. 3. There are six independent variables involved in the optimization process, including the sizes of the register file, issue window, memory disambiguation window, remote access window and remote access buffer, as well as the number of remote access ports.

Having constructed *global energy-efficient* families for single-cluster and multi-cluster architectures allows a knowledgeable comparison of these architectures for energy efficiency. Figure 8 shows the global energy-efficient families for single-cluster, two-cluster and four-cluster architectures plotted on the same graph in energy-per-instruction versus IPC coordinates.

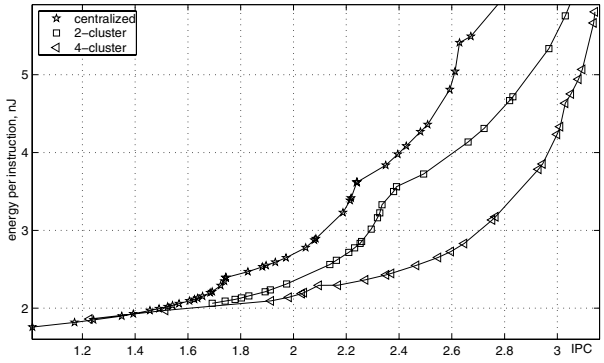

Figure 8: Global energy-efficient families for single-cluster, two-cluster, and four-cluster architectures.

An important result that can be derived from Figure 8 is that using multi-cluster configurations for building higher-performance processors allows us to further increase the architectural speed of the processors, while keeping the micro-architecture *energy-efficient* in the sense of the of the energy-delay product, $E \times D$, as discussed in Section 2.1. Indeed, for keeping the micro-architecture *energy-efficient* it is sufficient to guarantee that the energy dissipation per instruction grows no faster than linearly with increasing IPC. This can be achieved by using the single-cluster 1×4-way and 1×6-way configurations for delivering IPC below 2, the two-cluster 2×4-way and 2×6-way configurations for IPC in the range from 2 to 2.5, and the four-cluster configurations for achieving architectural speed up to 3 instructions per cycle. This provides an energy efficient micro-architecture road-map for the next several generations of high-performance processors.

Figure 9 (upper graph) shows the *global energy-efficient* families plotted in energy-per-cycle versus IPC coordinates. It indicates that given the same power budget the multi-cluster architecture allows configurations that can deliver up to 20% higher performance

than the best configurations with the centralized architecture, while the typical performance improvement for the IPC range from 2 to 3 is around 15%. Conversely, given a performance target, configurations the multi-cluster architecture can deliver the required performance, dissipating only half the power dissipated by the best configurations of the single-cluster architecture.

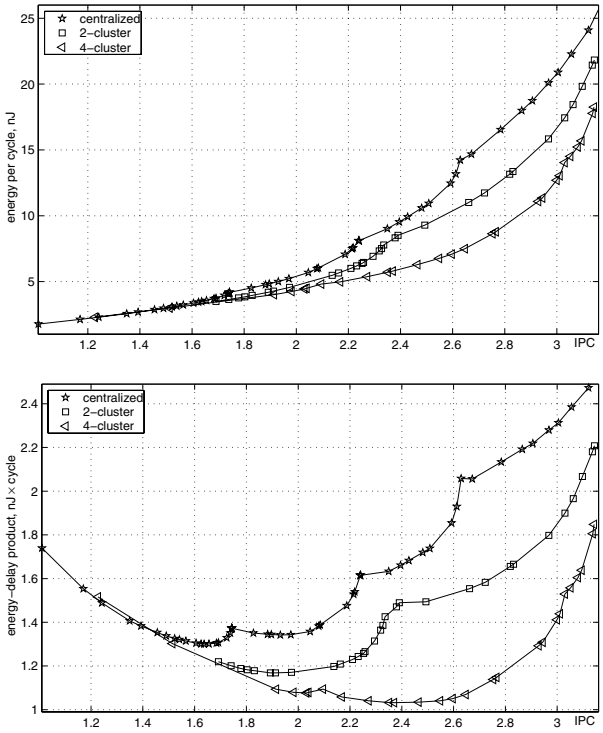

Figure 9: Global energy-efficient families for single-cluster, two-cluster, and four-cluster architectures.

The plots showing the energy-delay product for the global energy-efficient families in Fig. 9 (lower graph) prove that exploiting Instruction-Level Parallelism (ILP) generally improves the energy-delay metric as long as the energy overhead of the hardware is moderate. As the energy overhead of exploiting ILP becomes too high, further pushing the IPC by increasing the processor out-of-order capabilities causes the energy-delay product to grow. Thus, for any architecture there exists a certain value of IPC for which the energy-delay metric reaches its minimum.

The lower graph on Fig. 9 shows that architectures with more clusters reach the minimum energy-delay metric at higher values of IPC. The minimum energy-delay product achieved by the two-cluster architecture is lower than that achieved by the single-cluster architecture, and the minimum energy-delay product of the four-cluster architecture has the lowest value of approximately $\frac{1nJ \times cycle}{instruction^2}$, achieved at IPC = 2.5 instructions per cycle.

4 Conclusions

A methodology has been proposed for power-optimization at the micro-architecture level. Major targets for power reduction have been identified, then an energy-efficiency optimization of a superscalar micro-architecture has been performed to construct a family of *energy-efficient* configurations. Analysis of centralized superscalar architecture has been performed, indicating the inherent energy inefficiency of the traditional approach. Multicluster architecture has been suggested as a possible solution to the problem. Comparison of energy-efficient families of centralized and multicluster micro-architectures shows that at the architectural level there is a potential for reducing power of the traditional design up to 50%, given a performance requirement, and for up to 15% performance improvement, given a power budget.

References

[1] D. Burger and T. Austin. The SimpleScalar tool set, version 2.0. Technical Report 1342, University of Wisconsin-Madison, Computer Science Department, 1997.

[2] R. Colwell et al. A VLIW architecture for a trace scheduling compiler. *IEEE Transactions on Computers*, 37(8):967–979, August 1988.

[3] K. Farkas. *Memory-System Design Considerations for Dynamically-Scheduled Microprocessors*. PhD thesis, University of Toronto, 1997.

[4] James Farrell and Timothy Fischer. Issue logic for a 600-MHz out-of-order execution microprocessor. *IEEE Journal of Solid-State Circuits*, 33(5), May 1998.

[5] M. Franklin. *The Multiscalar Architecture*. PhD thesis, University of Wisconsin-Madison, November 1993.

[6] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277–1283, September 1996.

[7] R. Kessler. The Alpha 21264 microprocessor. *IEEE Micro*, 19(2):24–36, March 1999.

[8] A. Kumar. The HP PA8000 RISC CPU. *IEEE Micro*, 17(27-32), April 1997.

[9] J. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7(4):308–313, 1965.

[10] S. Palacharla, N. Jouppi, and J. Smith. Complexity-effective superscalar processor. *Proceedings of the 24th Annual International Symposium on Computer Architecture*, pages 206–218, June 1997.

[11] J. Smith and G. Sohi. The microarchiteture of superscalar processors. *Proceedings of the IEEE*, December 1995.

[12] M. Tremblay, D. Greenley, and K. Normoyle. The design of the microarchitecture of UltraSPARC-I TM. *Proceedings of the IEEE*, pages 16531–1663, December 1995.

[13] S. Vajapeyam and T. Miltra. Improving superscalar instruction dispatch and issue by exploiting dynamic code sequences. *Proceedings of 24th Annual International Symposium on Computer Architecture*, June 1997.

[14] N. Vasseghi, K. Yeager, et al. 200-MHz superscalar RISC microprocessor. *IEEE Journal of Solid-State Circuits*, 31(11):1675–1685, November 1996.

[15] V. Zyuban. *Inherently Lower-Power High-Performance Superscalar Architectures*. PhD thesis, University of Notre Dame, January 2000.

[16] V. Zyuban and P. Kogge. The energy complexity of register files. *IEEE Symposium on Low Power Electronics and Design*, pages 305–310, August 1998.