

Energy Efficient Asymmetrically Ported Register Files

Aneesh Aggarwal
ECE Department
University of Maryland
College Park, MD 20742
aneesh@eng.umd.edu

Manoj Franklin
ECE Department and UMIACS
University of Maryland
College Park, MD 20742
manoj@eng.umd.edu

Abstract

Power consumption in the register file (RF) forms a considerable fraction of the total power consumption in a chip. With increasing instruction window sizes and issue widths, RF power consumption will suffer a significantly large growth. Using the fact that many of the register values are small and require only a small number of bits for representation, we propose a novel asymmetrically ported RF (to reduce RF power consumption), in which some of the ports can only read/write small-sized values. We experiment with both monolithic and partitioned versions of asymmetrically ported RFs. The power savings in the RF with partitioned asymmetrically ported RF reach as high as 60%. These reductions in RF power consumption come with about 40% improvement in RF access-time and a negligible impact on IPC (Instructions per Cycle).

1. Introduction

Dynamic power consumption plays a key role in the design of high performance processors [10, 19]. The Register File (RF) is known to consume a significant portion of the total power dissipated in a chip [4, 9, 13, 25]. High RF power consumption is because of the large register files used to obtain high performance. In fact, a study [24] has shown that for current out-of-order superscalar processor designs such as MIPS R10000 [21] and Alpha 21264 [14], RF consumes the largest fraction of the total chip power consumption. The study also showed that with increase in issue width, the RF has an energy-per-instruction growth that is second only to that of the instruction issue window. Apart from high power consumption, longer access times for larger RFs is now becoming a major hurdle in clock speed improvement [1, 6, 20]. Hence, techniques to reduce RF power consumption and access time are very important.

Figure 1 shows the organization of a register file (RF). It consists of a 2-dimensional array of memory cells with rows equal to the number of physical registers and columns equal to the word length. The number of word-lines in each row is equal to the number of read and write ports. The

number of bit-lines in each column is twice the number of read and write ports (each port consists of a Bit line and a Bit' line). To read from or write a value into a register entry, one of the word-lines for that entry is activated. The bit-lines connected to the word-line read the value from or write the value into the memory cells of that entry.

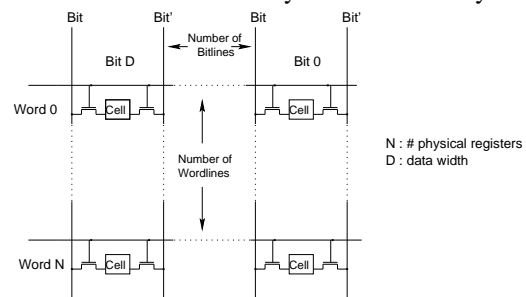


Figure 1. RF Organization

The major power consumers in an RF structure are the bit-lines and the word-lines [3] because of their large lengths and numbers, which in turn is because of: a large number of ports used for performing multiple RF accesses in a cycle, a large number of physical registers, and large data widths. Large number of ports and physical registers are required for exploiting Instruction Level Parallelism. For example, a processor with an issue width of 8 may need to make 16 read and 8 write accesses per cycle to the physical RF and requires up to 120 physical registers for obtaining significant performance [8]. Long bit-lines and word-lines also imply long delays for driving them.

In this paper, we focus on reducing RF power consumption by reducing the capacitance that gets charged and discharged in each access to the RF. For this, we use the property that many of the values stored and read from the RF are small in size (i.e., represented by a small number of bits with a large number of leading zeroes). Hence, in our RF organization, called *asymmetrically ported* RF, fewer ports are provided to read the high order bits, while more ports are provided to read the low order bits. We experiment with both monolithic and partitioned *asymmetrically ported* RFs. We found that the partitioned version reduces the dynamic

RF power consumption to the extent of 60% with a small IPC (Instructions per Cycle) loss of about 3%. We also found that the technique also reduces the RF access-time.

The rest of the paper is organized as follows. Section 2 provides the intuition behind *asymmetrically ported* RFs. Section 3 presents an *asymmetrically ported* RF and its implementation. Section 4 presents evaluation of the new RFs. In section 5, we study the sensitivity of *asymmetrically ported* RFs’ performance to the issue width. We discuss related work in Section 6. Section 7 concludes the paper.

2. Motivation

2.1. Experimental Setup

Our base-line architecture model, shown in Figure 2, is similar to the micro-architecture of MIPS R10000 [21], Alpha 21264 [14], and HP PA-8000 [15]. We assume a single cycle register read/write, which may be an optimistic assumption for a large RF [20]. We use the *Wattch*¹ simulator [3], a power estimator based on the *SimpleScalar* simulator [5], simulating a 32-bit PISA architecture. *Wattch* provides an accurate account (which is only about 10% off) of the power consumption in a chip [3]. The hardware features and default parameters that we use are given in Table 1. For benchmarks, we use a collection of 9 SPEC2000 integer programs (*gzip*, *vpr*, *mcf*, *parser*, *vortex*, *bzip2*, *twolf*, *crafty*, *gcc*), using *ref* inputs, and 2 media benchmarks (*cjpeg* and *djpeg*), using *test* inputs. The benchmarks are compiled (with an optimization flag of -O3) using the compiler that comes with *SimpleScalar*. In this paper, the scheme is applied to the Integer RF accessed by the Int FUs, hence only the integer benchmarks are used. The statistics are collected for 500M instructions after skipping the first 500M instructions for the SPEC2000 benchmarks and for all the instructions (about 15M) for the media benchmarks.

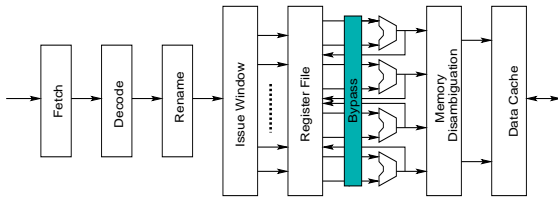


Figure 2. Superscalar Configuration Pipeline

2.2. Issue Slots with Small-Sized Operands

We first measure the average percentage of integer instruction issue slots per cycle with (i) both, (ii) one, and (iii) no operands greater than a particular size. Figure 3 gives the statistics for operand sizes of 8, 16, and 24 bits. For instance, the third bar for *cjpeg* shows that about 18% issue slots have both operands less than or equal to 8 bits, about 27% issue slots have exactly one operand greater than

Parameter	Value
<i>Technology</i>	0.18 μ m
<i>Fetch/Commit</i>	8 instructions/cycle
<i>Int/FP Window Size</i>	128/64 instructions
<i>Int/FP Phy. RF</i>	128/64 entries
<i>Int/FP Issue Width</i>	8/4 instructions/cycle
<i>L1 - I-Cache</i>	128K, direct-mapped
<i>L1 - D-Cache</i>	128K, 4-way assoc.
<i>L2 - Cache</i>	unified 512K, 4-way assoc.
<i>Mem. Reorder</i>	16 entries
<i>Functional Units (FUs)</i>	Int. : 8 ALU, 1 Div/Mult. FP : 8 ALU, 1 Div/Mult.

Table 1. Default Experimental Parameters

8 bits, about 7% issue slots have both operands greater than 8 bits, and about 48% issue slots are not used.

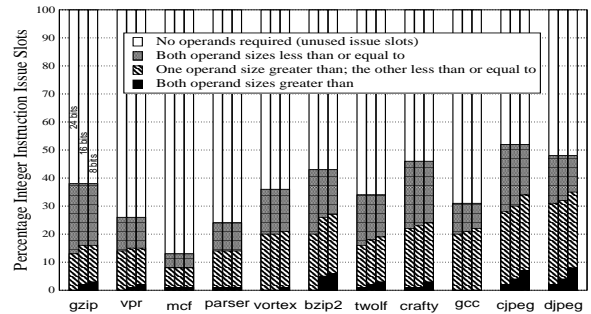


Figure 3. Integer Issue Slots per cycle with operands greater than 8, 16, and 24 bits

If the unused issue slots are assumed to issue instructions with operand sizes of 0, most of the integer issue slots (80% and 65% for the SPEC2000 and media benchmarks, respectively) have both the operands less than or equal to 8 bits. Most of the remaining issue slots have only 1 operand greater than 8 bits. Many issue slots with 1 wide operand are due to the load instructions accessing array elements, global variables, local variables etc., where the effective address is typically generated by adding a small value to a 32-bit address. This also explains why their percentage does not vary significantly from 8 to 16 to 24 bits. In fact, on the average, only about 2-4% of the issue slots have both operands greater than 8 bits. An even smaller percentage of issue slots have both operands greater than 16, and 24 bits. This means that most of the RF accesses are for small-sized values, and do not require normal-sized read/write ports.

3. Asymmetrically Ported RF

Traditionally RF ports have been symmetric in that all bit positions have the same number of read/write ports (Figure 1). However, for a small-sized value, many of the leading bit-lines need not be activated. This may be achieved with some form of clock-gating [2], where only the relevant bit-lines are activated. Although some power savings can be obtained by clock gating, there is room for much more sav-

¹In *Wattch*, the reservation stations forming the issue queue also act as an RF. We modified *Wattch* to have a separate issue queue and a RF.

ings. We propose an *asymmetrically ported* RF in which the bit-lines for the leading bit positions are entirely removed in some of the ports. We call these ports as *narrow ports* and the rest as *normal ports*. Narrow ports can only read/write small-sized values, whereas normal ports can read/write any value. The internal organization of such an *asymmetrically ported* RF is given in Figure 4. In Figure 4, the LSB (least significant bit) has 4 bit-lines for realizing 2 ports, whereas the MSB has only 2 bit-lines, realizing a single port.

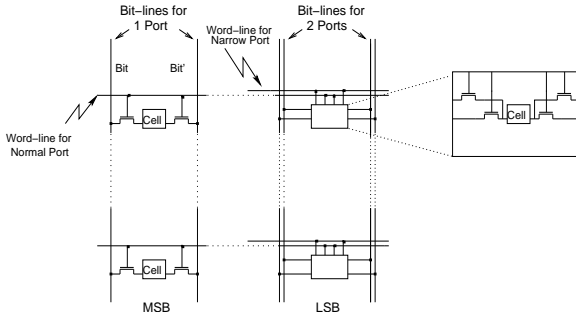


Figure 4. Asymmetrically Ported RF

Since FUs have dedicated read/write ports into the RF (for minimal RF access time), in an *asymmetrically ported* RF, some FUs are provided with narrow read ports. An FU with two narrow read ports into the RF can only operate on narrow-width operands, and thus will produce a narrow-width result, except for Integer Multipliers. Hence, all the Int ALU FUs with two narrow read ports are provided with narrow write ports. An FU with normal ports can execute any instruction, irrespective of the size of the operands.

3.1. Implementation Details

Issuing the instructions to the appropriate FUs (e.g., instructions with normal-width operands to FUs having normal-width ports) is performed by the select logic, which consists of a tree of arbiters (Figure 5(a)) [16]. Inputs to the select logic are *request* signals from the instructions that are ready. In each arbiter cell, the *anyreq* output signal is raised only if any of the input signals are high. If the *enable* signal is high, the highest priority request is granted the FU.

Figure 5(b) shows the detailed circuit for *grant1* signal generation for the case where *req0* has the highest priority, then *req1*, then *req2*, and so on. In this design, while the request signals are traversing to the root of the arbiter tree, their priority is precomputed in the arbiter cells. The logic equation for *grant1* is equation (1). For *asymmetrically ported* RFs, it is changed to equation (2) for an FU with 2 narrow read ports and to equation (3) for an FU with 1 narrow and 1 normal read ports.

$$grant1 = \overline{req0} \cap req1 \cap \overline{enable} \quad (1)$$

$$grant1_{nn} = (\overline{req0} \cap S01 \cap S02) \cap req1 \cap \overline{enable} \cap S11 \cap S12 \quad (2)$$

$$grant1_{nw} = \overline{req0} \cap (S01 \cup S02) \cap req1 \cap \overline{enable} \cap (S11 \cup S12) \quad (3)$$

Here each request is accompanied by 2 bits, S1 and S2 (e.g., *req0* is accompanied by S01 and S02), requiring 2 additional bits for each instruction in the IW. S1 indicates whether operand1 is wide (S1 = “0”) or narrow (S1 = “1”) and S2 indicates whether operand2 is wide (S2 = “0”) or narrow (S2 = “1”). In equation (2), *grant1_{nn}* is raised if *req0* is not accompanied with narrow operands and *req1* is accompanied with two narrow operands. Figure 5(c) shows the modified *precompute priority* part for the *grant1_{nn}* signal generator. The *with enable* part remains the same. Only the lower most arbiter cells (*level 1* in Figure 5(a)) need to be modified. The *Precompute priority* part for the *grant1_{nw}* signal generator can be modified accordingly.

A set of 1-bit latches (equal to the number of physical registers) keeps track of the size of the RF values. When a value is written into the RF, the size of the value is known, and the 1-bit latch for the register storing the value is set accordingly. We investigate 2 schemes based on how the bits S1 and S2 for each request are set:

- **Rename Time Only (RTO):** When an instruction is decoded and renamed, the size of the available operands is known from the 1-bit latches, and the bits S1 and S2 are set accordingly. An instruction requiring only one operand is assumed to have the other operand of size 0 and available, and the bit for the operand is set. An unavailable operand is assumed to be wide and the bit for the operand is reset.
- **Wakeup Time Also (WTA):** The bits for the operands available at dispatch-time are still set as in the *RTO* scheme. If an operand value is not available at dispatch time, it becomes available when the producing instruction executes. For such operands, the bits are set based on the FU used by the producer instruction. So, if the producer executes on an FU having 2 narrow read ports, then the operand is narrow, else it is normal.

The result produced by an Int ALU with 2 narrow operands can be at the most only 1 bit more than the width of the operands. Hence, for correct bypass of the values, the width of the narrow read and write ports is made 1 bit more than the maximum width of the narrow operands. On the other hand, a value wider than the maximum width of narrow operands is considered wide. For instance, for 16-bit narrow operands, the width of the narrow read and write ports is 17 bits, and a value of width 17 bits or more is wide. Henceforth, only the width of narrow operands is mentioned and it is implicit that narrow read and write ports are 1 bit wider. Also, when a value is written to the register file using narrow write ports, the higher order bits are reset, so that a read of the value using normal ports results in the correct value.

3.2. Overheads and Merits

Overheads: The additional hardware required includes the additional transistors in the select logic, the two bits in each

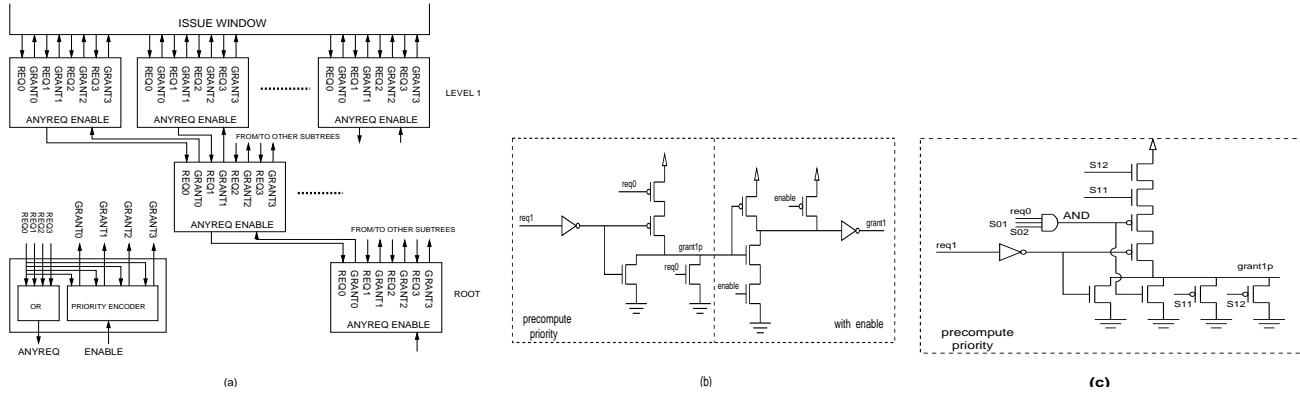


Figure 5. (a) Select Logic Arbiter Tree and a Single Arbiter in Detail; (b) grant1 Signal Generation; (c) Precompute priority for $grant1_{nn}$ signal generator for an FU with 2 narrow ports

IW entry, and the 1-bit latches (equal to the size of the RF). The power consumed in the additional hardware is negligible (within 1%, based on our experimental studies) as compared to the RF power savings. For measuring the power consumed in the 1-bit latches, we use the energy values given in [11] for the various latches. The additional transistors in the select logic consume almost zero power because only a few of them (for the instructions requesting a FU) get activated every cycle. The additional bits in the IW also consumed negligible power in our experiments.

Merits: The additional hardware does not lie in the critical path. No additional delays are introduced in writing to the RF because the size of the value can be determined in parallel. The additional transistors in the select logic are in the precompute stage which is triggered when the request signals are traversing the arbiter tree; thus not affecting the delay in the select logic. On the other hand, a considerable RF access time reduction is achieved (Section 4.3), which is critical to obtaining better performance [1]. Considerable RF power reduction is also achieved.

4. Experimental Evaluation

We evaluate both monolithic and partitioned *asymmetrically ported* RFs using the setup given in Table 1.

4.1. Monolithic Asymmetrically Ported RF

We experiment with 3 integer RF configurations (Table 2), with narrow operands of width 16 and 8 bits (normal operands are 32-bit wide). Table 3 gives the percentage savings in RF power consumption for these configurations averaged over all the benchmarks. These results include the power consumed in the additional hardware and are reported with conditional clocking [3], where the ports not used in a cycle consume almost no power.

In Table 3, the power savings increase with decrease in the width and increase in the number of narrow ports. For the 8-bit narrow operands, RF power savings with Config 2 is about 35%. With Config 3, RF power savings reach almost 50%. RF power consumption savings with the WTA

Config.	Int ALU FUs (total 8) with		
	2 narrow read ports	1 narrow, 1 normal read ports	2 normal read ports
Config 1	4	None	4
Config 2	4	2	2
Config 3	6	1	1

Table 2. Integer RF Configurations Simulated

scheme is slightly less than that with the RTO scheme, because the WTA scheme makes more RF accesses per cycle as it has a higher IPC than RTO, as we will see later in the section. But, the overall energy consumed in WTA is less than that consumed with RTO.

Scheme	Config 1		Config 2		Config 3	
	16b	8b	16b	8b	16b	8b
RTO	27%	33%	31%	39%	40%	50%
WTA	27%	33%	30%	37%	38%	48%

Table 3. RF power savings with conditional clocking for 16-bit and 8-bit narrow operands

Figure 6 gives the IPCs obtained for the 3 configurations using 8-bit narrow operands. In Figure 6:

- Almost no IPC loss is observed for Configs 1 and 2 with the WTA scheme. This is because, most of the issue slots require operands with size less than or equal to 8 bits (Figure 3). Also, even if a few instructions are delayed due to unavailability of normal-ported FUs (our experiments showed that only about 1% of the total instructions issued were delayed by a cycle), they will soon be issued in the subsequent cycles.
- On the other hand, with the RTO scheme, the IPC impact is higher. This is because, as the operand sizes of many instructions are not known at dispatch time, it assumes these operands to be wide, which increases the number of instructions operating on wide operands.
- For Config 3, as expected, the IPC impact increases (more for the RTO scheme), particularly for relatively larger IPC benchmarks such as *bzip2*, *gzip*, *vortex*, *twolf*, *cjpeg*, and *djpeg*.

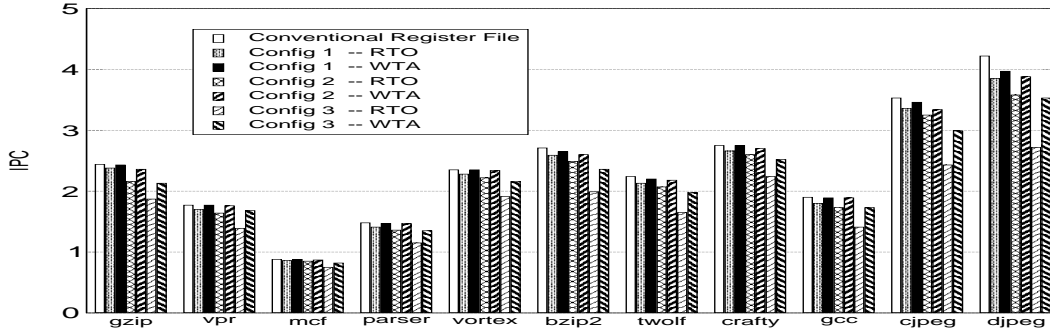


Figure 6. Performance (IPC)

- Higher IPC loss for media benchmarks is because of their high base IPCs and more wide operands (Figure 3), resulting in more normal-ported FUs being required.

The results show that, for the WTA scheme, almost 40% power savings is obtained without any IPC loss (for Config 2), whereas for 50% power savings, an IPC loss of 5-8% is incurred for the SPEC2000 benchmarks.

4.2. Partitioned Asymmetrically Ported RF

Asymmetrically ported RFs reduce RF power consumption by reducing the number of bit-lines and the length of the word-lines. RF power consumption can be further reduced by partitioning the *asymmetrically ported* RF, thereby reducing the length of the bit-lines. We experiment with 2 configurations of partitioned *asymmetrically ported* RF (Configs 1 and 2 from Table 2), for both 8-bit and 16-bit narrow operands using the WTA scheme. In both the configurations, the RF is partitioned into 4 sub-structures, and all FUs can access all sub-structures. We found that with 16-bit narrow operands, Config 1 reduces RF power consumption by about 45% and Config 2 reduces by 50%. For 8-bit wide narrow operands, the corresponding numbers are 55% and 60%. The IPC loss incurred with *partitioned asymmetrically ported* RF equal to that for the monolithic version because all FUs can access all RF sub-structures.

With a partitioned RF, the number of ports in each RF sub-structure can be reduced because the RF accesses are divided among the RF sub-structures. This means that access to a particular RF sub-structure is limited only to a few FUs (and not all the FUs). This can again be implemented using the select logic to steer the instructions to appropriate FUs (as is done in Section 3.2). We found that when the number of FUs accessing each sub-structure is reduced to 4 (8 read/4 write ports), the RF power savings increase to more than 70% with an IPC loss of not more than 5%. The individual benchmark results are not presented here because the results did not vary significantly across the benchmarks.

4.3. Timing Results

The total RF access-time consists of four components: the time taken to decode the register number, the time taken to drive the word-line, the time taken to pull the bit-line low,

and for the sense amplifier to detect the change in the bit-line and produce the corresponding output. We use the analysis in [16] to measure the RF access-times. For a monolithic *asymmetrically ported* RF with Config 2 in Table 3, using 8-bit narrow operands, the access time reduces by 25%. This results from reduced word-line lengths requiring less delays in driving them. The corresponding number for a partitioned *asymmetrically ported* RF is about 40%. The additional reduction comes from reduced bit-line lengths. This means that, with *asymmetrically ported* RFs, not only is a significant reduction in RF access time achieved, but it is also achieved with a negligible IPC impact, resulting in a significant potential improvement in the performance [1].

5. Sensitivity to Issue Width

In section 2.2, one reason for the large number of issue slots with small-sized operands was the unused issue slots. The number of unused issue slots and the percentage of issue slots with wider operands will vary with varying issue width. Hence, we experiment with 4 different issue widths of 2, 4, 6 and 8. The number of Int ALU FUs are also varied accordingly, while keeping the other parameters constant. For an issue width of 2, 1 FU has only narrow ports and the other has normal ports, for an issue width of 4, 2 FUs have narrow ports and 2 FUs have normal ports, and for an issue width of 6, 3 FUs have narrow ports, 1 FU has 1 narrow and 1 normal port, and 2 FUs have normal ports. We use the Config 2 from Table 2 for an issue width of 8. Figure 7 presents the reduction in IPC (compared to a *symmetrically ported* RF with the same issue width) for the all the configurations using the WTA scheme and 8-bit narrow operands.

In Figure 7, the impact of an *asymmetrically ported* RF on IPC remains almost the same for issue widths of 8, 6, and 4, but increases slightly for an issue width of 2. With *asymmetrically ported* RFs, IPC is affected when narrow-ported FUs go waste for want of ready instructions with narrow operands while other ready instructions with wide operands are not able to issue. For instance, for an issue width of 2, an *asymmetrically ported* RF impacts IPC when more than 1 instruction with wide operands are ready but no instruction with narrow operands is ready. Issue slots were not frequently wasted in our experiments. Hence, *asymmetrically ported* RFs remain effective with reduced issue widths also.

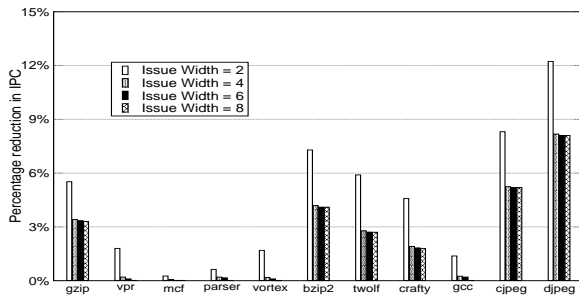


Figure 7. Percentage Reduction in IPC

6. Related Work

Different reduced latency and energy-efficient RFs have been proposed. Software controlled two-level RFs have been proposed in [18, 23]. Cruz, et al [6] propose a two-level hierarchical RF. They use a fully associative upper level RF which stores the critical register values. A similar idea was also proposed in [22]. Balasubramonian, et al [1] propose a two-level RF which adds a L2 RF to the conventional L1 RF. Registers are written to the L2 RF when the number of physical registers falls below a pre-set threshold.

Partitioning or replication of register files has been proposed in the context of clustered processors [7, 14, 17]. Partitioned register files have also been proposed in [12] for a VLIW processor, and in [1] for superscalar processors. Balasubramonian, et al [1] also propose reducing the number of ports per bank. Brooks et al [2] use small-sized operand values to reduce power consumption in functional units. To the best of our knowledge, *asymmetrically ported* RF organization is the first of its kind that exploits small-sized operand values to reduce power consumption in RFs.

7. Summary and Conclusions

RF power consumption is a major contributor to the total chip power consumption for microprocessors with large issue queues and issue widths. In this paper, we presented a novel *asymmetrically ported* RF that uses the observation that many accesses to the RF are for small-sized data values. In this RF organization, some of the ports can only read/write the lower significant bits of a value. We investigated both monolithic and partitioned asymmetrically ported RFs. With the partitioned *asymmetrically ported* RF, the RF power savings reach as high as 60% with a 3% IPC loss and a 40% reduction in RF access-time. We also found that *asymmetrically ported* RFs remain effective for smaller issue widths too. These studies indicate that asymmetrically ported RFs are good candidates for reducing the power consumption in processors. This organization can also be used for other memory structures in the processor that store small-sized values.

References

[1] R. Balasubramonian, et. al., "Reducing the Complexity of the Register File in Dynamic Superscalar Processors," *Proc. Micro-34*, 2001.

[2] D. Brooks and M. Martonosi, "Dynamically Exploiting Narrow Width Operands to Improve Processor Power and Performance," *Proc. HPCA*, 1999.

[3] D. Brooks, V. Tiwari and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," *Proc. ISCA*, 2000.

[4] T. D. Burd and B. Peters, "Power analysis of a microprocessor: A study of an implementation of the MIPS R3000," *ERL Technical report*, University of California, Berkeley, May 1994.

[5] D. Burger and T. M. Austin, "The SimpleScalar Tool Set, Version 2.0," *Computer Arch. News*, June 1997.

[6] J.-L. Cruz, et. al., "Multiple-Banked Register File Architectures," *Proc. ISCA-27*, 2000.

[7] K. Farkas, et. al., "The Multicluster Architecture: Reducing Cycle Time Through Partitioning," *Proc. ISCA-24*, 1997.

[8] K. Farkas, et. al., "Register File Considerations in Dynamically Scheduled Processors," *Proc. HPCA*, 1996.

[9] D. R. Gonzales, "Micro-RISC architecture for the wireless market," *IEEE Micro*, July/August 1999.

[10] M. K. Gowan, et. al., "Power Considerations in the Design of the Alpha 21264 Microprocessor," *Proc. DAC*, 1998.

[11] S. Heo, et. al., "Activity-Sensitive Flip-Flop and Latch Selection for Reduced Energy," *Proc. Conf. on Advanced Research in VLSI*, March 2001.

[12] J. Janssen and H. Corporaal, "Partitioned Register File for TTAs," *Proc. Micro-28*, 1995.

[13] A. Kalambur and M. J. Irwin, "An extended addressing mode for low power," *Proc. IEEE Symposium on Low Power Electronics*, August 1997.

[14] R. Kessler, "The Alpha 21264 Microprocessor," *IEEE Micro*, March/April 1999.

[15] A. Kumar, "The HP PA 8000 RISC CPU," *IEEE Micro*, April 1997.

[16] S. Palacharla, "Complexity-Effective Superscalar Processors," *PhD thesis*, University of Wisconsin, 1998.

[17] S. Rixner, et. al., "Register Organization for Media Processing," *Proc. HPCA*, 2000.

[18] J. Swensen and Y. Patt, "Hierarchical Registers for Scientific Computers," *Proc. ICS*, 1988.

[19] V. Tiwari, et. al., "Reducing Power in High-performance Microprocessors," *Proc. DAC*, 1998.

[20] D. M. Tullsen et. al., "Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor," *Proc. ISCA*, 1996.

[21] K. C. Yeager, "The MIPS R10000 Superscalar Microprocessor," *IEEE Micro*, April 1996.

[22] R. Yung and N. Wilhelm, "Caching Processor General Registers," *Proc. ICCD*, 1995.

[23] J. Zalamea, et. al., "Two-Level Hierarchical Register File Organization for VLIW Processors," *Proc. Micro-33*, 2000.

[24] V. Zyuban and P. Kogge, "Optimization of High-Performance Superscalar Architectures for Energy Efficiency," *Proc. ISLPED*, 2000.

[25] V. Zyuban and P. Kogge, "Split register file architectures for inherently low power microprocessors," *Power Driven Microarchitecture Workshop at ISCA98*, June 1998.