An Efficient Wakeup Design for Energy Reduction in High-Performance Superscalar Processors

Kuo-Su Hsiao and Chung-Ho Chen
Department of Electrical Engineering, National Cheng Kung University
No.1, University Road, Tainan 701, Taiwan
newjimmy@ee.ncku.edu.tw, chchen@mail.ncku.edu.tw

ABSTRACT

In modern superscalar processors, the complex instruction scheduler could form the critical path of the pipeline stages and limit the clock cycle time. In addition, complex scheduling logic results in the formation of a hot spot on the processor chip. Consequently, the latency and power consumption of the dynamic scheduler are two of the most crucial design issues when developing a high-performance microprocessor. We propose an instruction wakeup scheme that remedies the speed and power issues faced with conventional designs. This is achieved by a new design that separates RAM cells from the match circuits. This separated design is such that the advantages of the CAM and bit-map RAM schemes are retained, while their respective disadvantages are eliminated. Specifically, the proposed design retains the moderate area advantage of the CAM scheme and the low power and low latency advantages of the bit-map RAM scheme.

The experimental results show that the proposed design saves power consumption by 80% compared to the traditional CAM-based design and 18% to the bit-map RAM design, respectively. In speed, the proposed design reduces an average of 77% in the wakeup latency compared to the conventional CAM-based design and an average of 33% reduction of the latency of the bit-map RAM design. For an 8-issue superscalar processor, the proposed design reduces the power consumption of the conventional wakeup logic by 80%, while simultaneously increasing the Instruction Count per nano-second (IPns) by a factor of approximately 2.5 times with a moderate area cost.

Categories and Subject Descriptors

C.1 **Processor Architectures:** Single Data Stream Architectures
C.5.3 **Microcomputers:** Microprocessors

General Terms: Performance, Design, Experimentation.

Keywords

Low Power, High Performance, Issue Window, Wakeup Logic

CF'05, May 4–6, 2005, Ischia, Italy.

1. INTRODUCTION

Power consumption and latency of instruction scheduler are the key issues in today's microprocessor design. Nowadays, design of high performance processor tends towards the specification of wide issue width and large issue window. However, this leads to an increasingly complex dynamic instruction scheduler.

Considering the clock cycle time, although a pipelined dynamic scheduler can increase the clock frequency, the instruction wakeup and instruction selection operations should be an atomic operation to avoid significant performance degradation. It has been shown that the latencies associated with the wakeup and selection operations form the critical path of the pipeline stages [1]. In addition to clock cycle time, power consumption is another important issue for processor design. The power consumption associated with the complex dynamic scheduler constitutes a significant portion of the processor power consumption. Table 1 shows the power breakdown of the Compaq Alpha 21264 processor [2], and indicates that, second only to the global clock, the instruction issue unit is the most power hungry component of the processor. Similarly, Figure 1 reveals that the out-of-order scheduler of the Intel Pentium 4 processor (NetBurst) accounts for approximately 40% of the total power consumption [4].

Table 1. Power breakdown of the Alpha 21264 processor

Component	Percentage
Global clock	32
Instruction issue unit	18
Caches	15
FP execution unit	10
INT execution unit	10
MMU	8
I/O	5
Miscellaneous logic	2

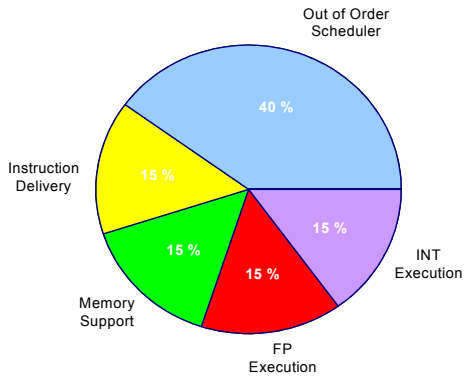

Figure 1. Power breakdown for the Intel Pentium 4 processor.

As shown in Figure 2, approximately 50% of the wakeup operations do not result in any instruction wakeup for the first (left) or the second (right) source operand. About 44% of the wakeup operations wake up only one instruction. Matching all of the tags in the issue window only to wake up a few instructions is inefficient in terms of both time and energy. In this paper, we propose a wakeup design that remedies the power consumption and the latency issues faced with conventional designs.

The remainder of this paper is organized as follows. Section 2 reviews dynamic schedulers used in superscalar processors. Section 3 details two conventional schedulers followed by the proposed one. Section 4 presents the experimental methodology and the evaluation results. Section 5 provides a brief review of previous related work, and finally, Section 6 presents the conclusion.

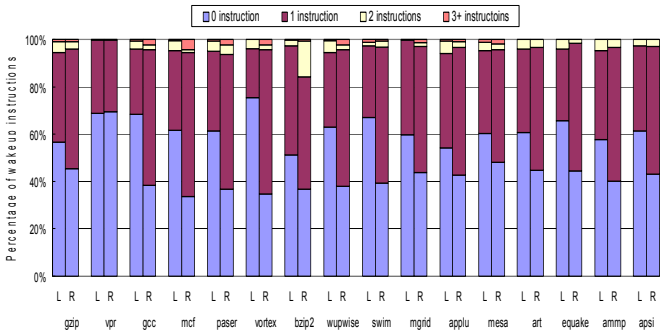

Figure 2. Number of instructions woken up per each broadcast result for the left and right source operand in a 16-issue 128-entry processor.

2. DYNAMIC SCHEDULER FOR SUPERSCALAR PROCESSORS

In this section, we provide background discussions for the operations of dynamic schedulers used in superscalar processors.

Figure **3** depicts a baseline model of a superscalar processor. The fetch unit retrieves multiple instructions from the instruction cache and uses a branch predictor to assist in fetching instructions speculatively over basic blocks during a clock cycle. Subsequently, the instructions are decoded and their register designators are renamed for resolving WAR (write after read) and WAW (write after write) dependences. In the issue window, instructions wait for their source operands to become available for execution. The wakeup and select logics schedule instructions for out-of-order execution. Instructions are committed in program order to ensure correct completion of the executing program.

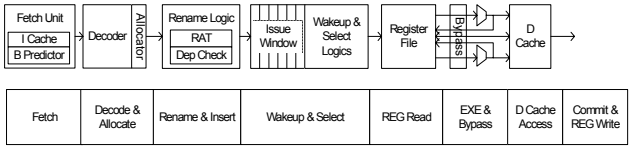

Figure 3. Baseline model of a superscalar processor.

2.1 Instruction Wakeup and Selection

The wakeup logic is responsible for waking up the instructions in the issue window when their source operands become available. When an instruction is going to complete execution, its destination tag is forwarded to the wakeup logic to wake up the relevant dependent instructions. Once both source operands are available, the wakeup logic sends a request for execution to the instruction select logic for this instruction. The detail of the wakeup design is discussed in Section 3.

The instruction select logic is responsible for selecting the appropriate instructions for execution from the instructions that have both their source operands available. Once a functional unit becomes available, the select logic then directs a ready instruction to that unit for execution. Many selection policies have been presented for the case where the number of ready instructions exceeds the capacity of the available functional units [3], for instance, the oldest first selection algorithm [1]. When an instruction is selected for execution, the select logic responds with a grant signal. If the size of the issue window is n, then the grant signals are numbered from 0 to n-1, respectively.

3. WAKEUP LOGIC DESIGNS

We begin first by introducing a conventional CAM-based design [1] and a bit-map RAM design [10] for the wakeup logic and then detail our approach.

3.1 CAM-Based Scheme

Conventional wakeup logic implementation is based on CAM (content-addressable memory) structure [1]. A CAM-based wakeup logic is shown in Figure 4. Once an instruction has been renamed, the allocated destination tag is inserted into the destination tag RAM (upper part of Figure 4a). The renamed left and right source tags are inserted into the left and right source tag fields (Tag L and Tag R) of the issue window respectively. The ready bits (Rdy L and Rdy R) are employed to indicate whether the corresponding source operands are available or not.

The grant signals from the select logic are used to index the destination tag RAM to retrieve the corresponding destination tags. The result tag bits are then driven on the tag bus (Tag 1 to Tag w) to the CAM array, which matches them with all the left and right source tags in the issue window.

Figure 4b depicts a single cell of the CAM array. Two chained inverters represent the memory cell for storing one bit of the source tag data, while two NMOS transistors form the

comparison circuit required to match the source operand bit with the result tag bit. Note that in this configuration, the memory cell and comparison circuit are closely integrated together.

With the match line being pre-charged high, if a match between the incoming tag and the source tag occurs, the match line remains high. Specifically, if result tag j ($1 \le j \le w$, w is the issue width) matches with a source operand tag, then match line j remains high. If one of the result tags (Tag 1 to Tag w) matches with the source tag, the ready bit should be set to indicate that this operand is available. In this way, the final ready signal (Rdy) is formed by the OR operations of all the match lines. In contrast, if the result tag bit is not equal to the corresponding cell bit, one of the two comparison circuits will be turned on to pull the match line down, that is, a mismatch occurs.

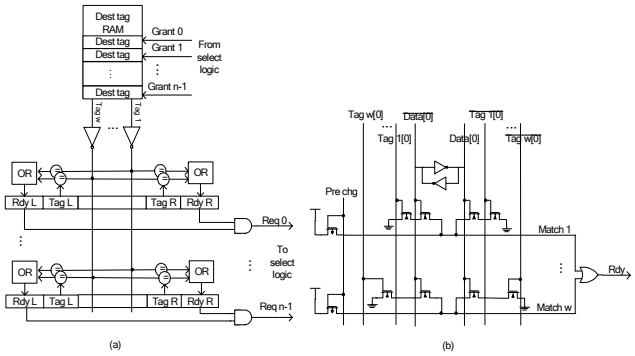

Figure 4. (a) Wakeup logic using CAM structure. (b) CAM cell closely integrated with the matching circuit.

3.2 Bit-Map RAM Scheme

Although only a few instructions are woken up, a CAM-based design actually matches the result tags with all of the source tags in the issue window. Clearly, this operation is both time and energy inefficient. The RAM scheme reduces the power requirements during the wakeup process [10].

Figure 5 depicts a wakeup logic implemented by using bit-map RAM structure. Two bit-map RAMs are used to store the dependence between instructions (left and right source operands) in the form of bit position. Each bit of the RAM structure represents dependence between two instructions. For example, the bit pointed to by i^{th} column and j^{th} ($0 \le i, j \le n-1$) row in the bit-map RAM indicates that instruction j requires a source operand from the output of instruction i. That is, all of the instructions that depend on instruction i form the column vector i. In this way, the wakeup operation can be simplified to a RAM-read operation. The read signal (grant line) selects a column of the bit vectors for output.

The bit-map RAM is a square memory array of which the number of row and column are identical to the size of the issue window. The two bit-map RAMs handle the instruction wakeup operations for the left and right operands.

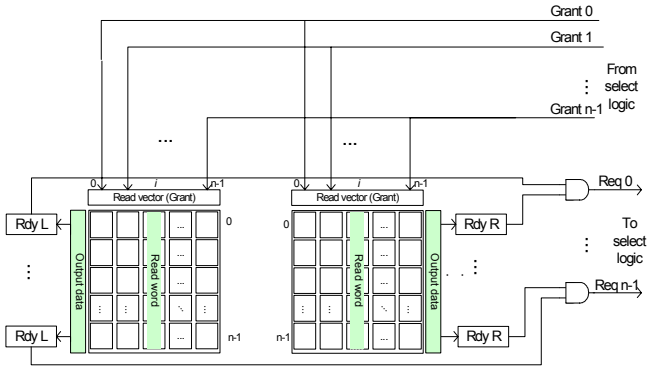

Figure 5. Bit-map wakeup design.

After renamed, the data dependence between the producer and the consumer instruction is inserted into the bit-map RAMs. As shown in Figure 6, the destination entry number (instruction j, for instance) is used to index the RAM structure to select the write word line. The renamed source tag (the entry number of instruction i that produces the result as the source operand for instruction j) is decoded to drive the corresponding write bit line. Consequently, the bit, which is located at the i^{th} column and the j^{th} row in the bit-map RAM, is set to indicate that instruction j depends on instruction i.

In the wakeup operation, the grant signal (grant i, for instance) from the select logic drives the corresponding read word line to select a column (the i^{th} column) of the RAM cells. If the cells in the selected column have been set previously, they pull down the read bit lines (only the j^{th} bit line is pulled down, in this example). The outputs of the selected column drive the ready bits accordingly. Thus, the wakeup operation is accomplished by means of a read operation.

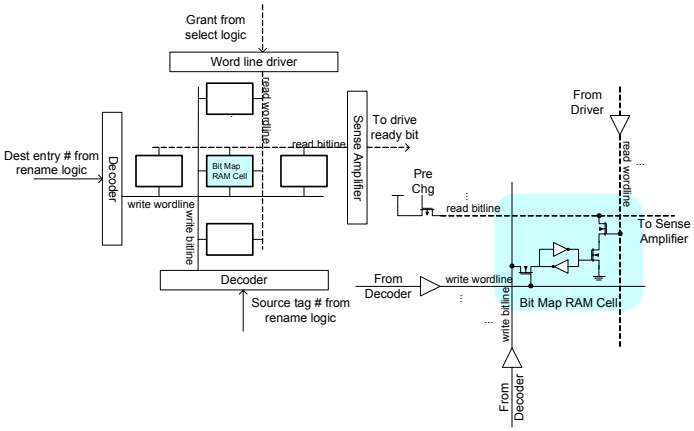

Figure 6. Bit-map RAM circuit.

3.3 A Complexity-Efficient Design

Although the RAM-based wakeup design is more energy efficient and faster than the CAM-based scheme, the large area cost of the bit-map RAM may prohibit its usage in a processor with a large issue window. In addition, using a large bit-map RAM structure may lead to excessive wire delay in the process of future technology [16]. To overcome this problem, we develop new instruction wakeup logic, which is able to reduce the area used while achieving low power and high speed requirements.

Figure 7 presents the proposed wakeup logic. Two sets of the wakeup structure are used to handle instruction wakeup operations for the left and right source operand respectively. This wakeup logic is designed to match the source tags in the issue window directly with the grant lines from the select logic in an efficient manner. Each entry of the wakeup design employs the proposed 32-bit wakeup circuit as shown in Figure 8 to perform wakeup operation.

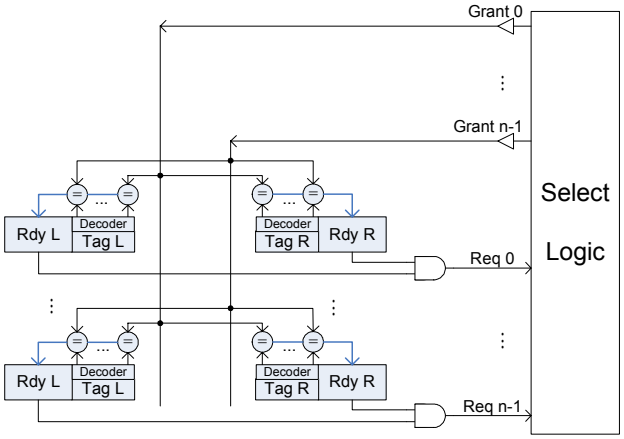

Figure 7. The proposed wakeup logic.

In Figure 8, the 32-bit wakeup circuit matches 5-bit source tag with 32 grant lines. The two chained inverters stand for a RAM cell. There are five data cells for storing source tag and a valid bit data cell to indicate whether this tag is valid or not. The 5-bit source tag is decoded into 32 decoded output lines that are connected to 32 match circuits. The grant input lines are represented by the vertical lines (grant0 to grant31), each of which runs through the match circuits to match with corresponding decoded line. Each match circuit consists of two nMOSFET transistors that match the decoded line with the grant line.

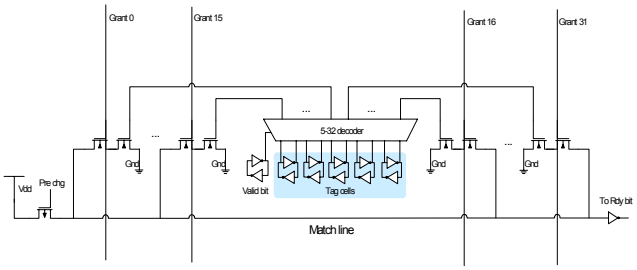

Figure 8. A 32-bit wakeup circuit.

We have designed the layout of the wakeup design for an 8-issue processor. In this layout, the decoder and the RAM cells for source tag are placed in the center of the wakeup module. The source tag consists of 6 RAM cell bits that has 8 write ports (for 8-issue processor). Two sets of 16 match circuits are placed at the left and right sides of the source tag respectively to match 32 grant lines with 32 decoded lines.

Since the 32-bit wakeup circuit module only has 32 grant input lines, for a larger issue window, the wakeup logic can be built up by merging multiple lanes of 32-bit wakeup circuit to accommodate more grant input lines. For example, wakeup logic for a 128-entry issue window can be built up by using four lanes of the 32-bit wakeup circuit as shown in Figure 9. The 128 grant lines are divided into 4 groups. Each group goes into a lane of the wakeup logic.

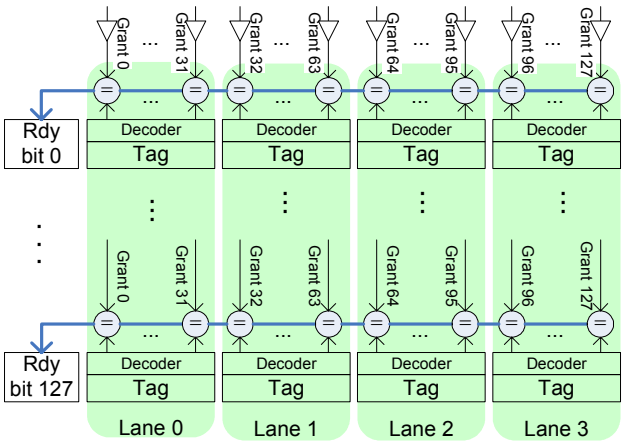

Figure 9. Proposed wakeup logic for 128-entry issue window.

After renamed, the source tag is inserted into the appropriate location of the wakeup logic in the following way. The destination tag is used to select one of the 128 entries of the wakeup logic. Besides, the most significant two bits of the renamed source tag are used to select one group out of the four to which the least significant five bits of the source tag are written. In other words, in each entry only the tag data in the selected lane is valid while the tag data in the rest of the lanes are invalid. The written 5-bit tag is decoded to drive the corresponding decoded line that is connected to one of the 32 match circuits. In each entry, only one decoded line is driven to wait for the corresponding grant line while the remainders are set low to turn off the other match circuits.

As stated previously, the select logic sends a grant signal to trigger wakeup operations for the left and right source operand. The grant signal is delayed for the appropriate cycles of this execution and then is sent to the left and right wakeup structure to wake up instructions that depend on this result. The grant lines go through all the entries of the wakeup structure. In each entry, each grant line connects to its match circuit that matches this grant line with the corresponding decoded line. When both the grant line and the corresponding decoded line become active, the match circuit pulls the match line low to indicate that this operand is available. If both the left and the right ready bits of an instruction are set, this instruction sends a request for execution to the select logic as described previously.

Compared to the CAM-based design, the proposed wakeup logic does not need to read the destination tags and hence the wakeup operation is faster. Additionally, the inputs of our design are grant lines, not destination tags. At most, only w grant lines are driven rather than x tag bits are driven. Where

w = issue width,

x = issue width $\times$ tag length.

Furthermore, only the match line that depends on the execution result is active to drive the ready bit. Therefore, the proposed wakeup design is not only faster than the CAM-based design, but also more energy efficient.

Compared to the bit-map RAM scheme, our design separates the RAM cells from the match circuits in order to use only a small area for the match circuits. Although the area of the match circuits in this design grows proportionally with the square of the size of the issue window, the area of the separated RAM cells grows only linearly with the increasing of the issue window. Hence, the proposed design drastically reduces the area requirement unlike the case of bit-map RAM that all grows proportionally with the square of the issue window size.

4. EXPERIMENTAL EVALUATION

In this section, we present the evaluation methodology and the results of power consumption, timing analysis, and area cost for the three designs.

4.1 Experimental Methodology

We used the Wattch simulator [6] to simulate the power consumptions of the three wakeup designs. The Wattch simulator is a tool based on SimpleScalar [7] for extracting the relative power consumptions of individual components in a superscalar processor. In Wattch, the data cell of source tag in the proposed design was modeled as conventional RAM cell and the match circuit of the proposed design was modeled as the match circuit in CAM structure. In addition, the RAM cell of bit-map structure was extended from the conventional RAM cell. Other parameters for the Wattch simulator include 1GHz clock frequency, 1.8V Vdd, and 0.18μm technology process.

Table 2 presents the architectural parameters for three different issue-width processors evaluated in this study. Architectural simulation was conducted using 7 integer and 9 floating point benchmark programs as specified by the SPEC2000 [8]. The test input set was used for all the selected SPEC2000 benchmark programs. Additionally, five Media-bench programs [11] were employed to obtain an even more comprehensive evaluation of the three designs. All the ported benchmark programs were compiled with full optimization (-O4) and were run to completion.

The Avant! Hspice tool was used to extract timing data for the circuit level designs. The Hspice circuit model was based on the one proposed by Ernst and Austin [5]. Finally, the parameters of CMOS transistors and wires in three designs all conformed to the design rules of TSMC 0.18μm process. The area sizes of the cells in three designs were based on the model in [9].

4.2 Power Consumption Statistics

Figure 10 presents the comparisons of power consumptions for the three wakeup logics in an 8-issue processor that has 64 entries issue window. The results indicate that the proposed design consumes the least amount of energy. Specifically, its power consumption is only 20% that of the CAM-based design and 80% that of the bit-map RAM design.

Additionally, the power consumptions of the three designs were also evaluated for 4-issue and 16-issue processors; the results are shown in Table 3. For the 16-issue 128-entry issue window processor, the proposed design only consumes approximately 14% of the power taken by the CAM-based design and approximately 46% of the power consumed by the bit-map RAM design. Regarding the 4-issue 32-entry issue window processor, the proposed scheme consumes about 26% of the power of the conventional CAM-based design. Because of the small load capacitances of the word line and bit line, the bit-map RAM scheme has the best power consumption of the three wakeup designs in a 32-entry 4-issue processor.

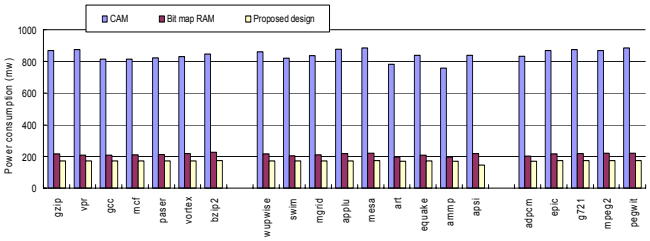

Figure 10. Power consumptions for the three designs.

Table 2. Architectural parameters

Dispatch, issue, commit width	4	8	16
Issue window size/ LSQ size	32/ 8	64/ 16	128/ 32
Functional units	4 IALU, 1 IMUL, 2 FALU, 1 FMUL, 2 LSU	8 IALU, 2 IMUL, 4 FALU, 2 FMUL, 4 LSU	16 IALU, 4 IMUL, 8 FALU, 4 FMUL, 8 LSU
L1 I-cache/ L1 D-cache	4-way, 64KB, 32-byte line, 1-cycle latency/ 4-way, 64KB, 32-byte line, 1-cycle latency		
L2 cache/ TLB	4-way, 256KB, 64-byte line, 10-cycle latency/ 4-way, 128-entry, 4KB page size		
Memory width and latency	64-bit wide, 75 cycle latency, 4-cycle burst		
Branch predictor	Combination of bimodal and 2-level global predictor/ 1024-entry chooser/ 4-way, 1024-entry BTB/ 2048-entry bimodal 8-bit history, 2048-entry level 2 16-entry RAS(return address stack)/ 8-cycle penalty		

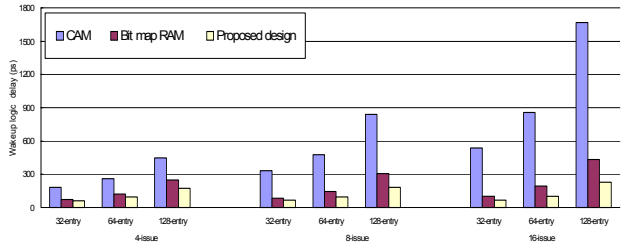

Figure 11. Latency of three different wakeup logics (ps)

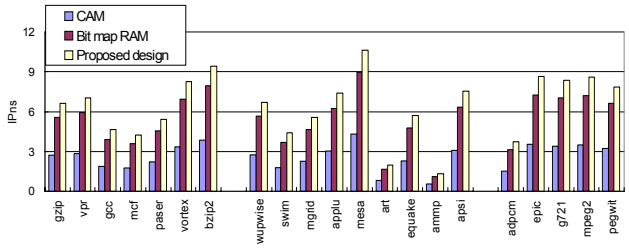

Figure 12. IPns for 8-issue 64-entry issue window processor.

4.3 Circuit Timing Result

Figure 11 shows that the latency of the proposed wakeup logic is 20% to 50% less than that of the bit-map RAM design and 60% to 90% less than that of the CAM-based design for the various configurations of processor. That is, the speed of the proposed wakeup design is the fastest among the three designs. Figure 12 presents the IPns (instructions per nano-second) for processors with three different wakeup schemes. The results reveal that the IPns of the processor with the proposed wakeup design is about 2.5 times greater than that with the CAM-based design and 1.2 times greater than that with the bit-map RAM design.

4.4 Area Size Tradeoff

Figure **13** shows the area sizes of the three designs. The area of the CAM-based design is the smallest of the three for the 4-issue and 8-issue processors. For the 16-issue processor, the size of the CAM structure increases because it needs 16 ports for destination tags and 16 write ports for source tags for each CAM cell. In this case, the area of the CAM-based design is larger than that of the proposed design.

The area of the bit-map RAM wakeup logic becomes unacceptably large as the window size increases since it is proportional to the square of the issue window size. The area of the 128-entry bit-map RAM design is approximately 5 to 6 times of the CAM-based design.

Table 3. Power consumption statistics (mw)

	4-issue 32-entry processor			8-issue 64-entry processor			16-issue 128-entry processor		
	CAM	Bit-map RAM	Proposed design	CAM	Bit-map RAM	Proposed design	CAM	Bit-map RAM	Proposed design
gzip	204	43	52	870	216	173	4623	1388	634
vpr	205	45	53	873	209	171	4604	1406	637
gcc	194	41	52	813	209	171	4259	1358	631
mcf	191	41	52	814	210	172	4319	1373	632
paser	196	42	52	822	212	172	4335	1367	632
vortex	200	43	52	831	218	173	4312	1399	636
bzip2	203	45	53	848	227	175	4450	1424	639
Int. avg.	199	43	52	839	214	172	4457	1392	635
wupwise	202	43	52	860	217	173	4511	1401	636
swim	193	40	51	819	206	171	4353	1360	631
mgrid	196	42	52	836	211	172	4399	1368	632
applu	205	43	52	878	218	173	4681	1403	636
mesa	207	44	53	886	212	174	4717	1417	638
art	181	37	51	782	198	169	4207	1326	627
equake	195	41	52	839	209	171	4440	1365	631
ammp	181	38	51	756	197	169	3950	1307	624
apsi	205	43	52	839	218	146	4695	1406	637
F.P. avg.	196	41	52	833	210	169	4439	1373	632
adpcm	196	39	51	832	202	170	4384	1321	626
epic	205	43	53	868	217	174	4583	1399	636
g721	205	43	52	873	220	174	4599	1389	635
mpeg2	205	45	53	868	222	174	4622	1430	640
pegwit	208	45	53	884	221	175	4661	1426	636
Med. avg.	204	43	52	865	216	173	4570	1393	635
Total avg.	199	42	52	842	213	171	4474	1385	634

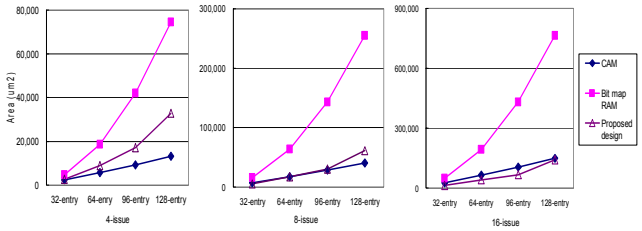

Figure 13. Areas of three wakeup logics (μm^2).

On the other hand, the proposed wakeup logic uses much less area than the bit-map RAM design. In the 16-issue processor, the area of the proposed wakeup logic is the smallest among the three designs.

5. RELATED WORK

Many previous researches have attempted to reduce the complexity of the dynamic scheduler. Folegnani and González presented a scheduler that dynamically manages the size of issue window and gates off needless (e.g. empty entries and those entries that are ready for execution) wakeup activities [12]. Ernst and Austin proposed a scheduler that employs less tag comparators to reduce the complexity of the scheduler. This scheduler also has a last tag speculator to reduce the frequency of tag matching [5]. Huang and et al. proposed an index-based scheduler [14], which employs producer instruction pointer and consumer instruction pointer to index the instructions that should be woken up, to improve the energy efficiency of the scheduler. However, this scheduler must work together with a conventional CAM structure.

Kim and Lipasti proposed a sequential wakeup mechanism to reduce the complexity of scheduler [15]. In this mechanism, the last-arrival operand is placed into the fast wakeup entry and two (left and right) source operands of an instruction are woken up in two sequential steps. The sequential wakeup logic enables a higher clock frequency by reducing the load capacitance of tag driver in the wakeup logic. Henry and et al. presented a cyclic segmented prefix (CSP) circuit to improve the performance of wakeup logic [17]. Ernst and et al. also proposed a Cyclone scheduler that predicts the operand arrival time and schedules instructions in a countdown cyclic queue. This scheduler reduces the area cost of scheduler and boosts the clock frequency with a small IPC degradation [13]. Other works [18][19][20] reduced the complexity of scheduler by scheduling dependent instructions into data-flow based issue window.

6. CONCLUSIONS

This paper presents an energy-efficient wakeup design for the dynamic scheduler of high-performance superscalar processors. The proposed wakeup scheme reduces the wakeup latency and power consumption by matching source tags directly with the grant lines. This design eliminates the read operation of destination tag and reduces the number of tag lines to be driven in wakeup operation. In addition, this scheme is more efficient for tag matching since only one match circuit is turned on for each match. Furthermore, the proposed design separates the RAM cells from the match circuits to reduce the area cost.

The simulation results show that on average for three different issue-width processors, the proposed design saves power consumption by 80% and 18% respectively compared to the conventional CAM-based wakeup logic and the bit-map RAM design. The proposed design enables an average saving of 77% in the wakeup latency compared to the conventional CAM-based design and an average latency saving of 33% compared to the bit-map RAM design. For area cost, the proposed design is only 29% of the bit-map RAM design and approximately 1.7 times of the CAM-based design for a 128-entry issue window processor.

In conclusion, the results have confirmed that the proposed wakeup scheme greatly improves the power consumption and reduces the latency of conventional wakeup schemes with a moderate area cost.

ACKNOWLEDGEMENT

The work in this paper is in part supported by the research grant under NSC 93-2220-E-006-004, Taiwan, ROC.

7. REFERENCES

[1] S. Palacharla, N. P. Jouppi, and J. E. Smith, "Quantifying the Complexity of Superscalar Processors," *Tech.Rep. CS-1328, University of Wisconsin-Madison*, May 1997.

[2] M. K. Gowan, L. L. Biro, and D. B. Jackson, "Power considerations in the design of the alpha 21264 microprocessor," *Proceedings of the Design Automation Conference (DAC)*, June 1998.

[3] M. Butler and Y. N. Patt, "An Investigation of the Performance of Various Dynamic Scheduling Techniques," *25th Annual International Symposium on Microarchitecture*, December 1992.

[4] S.H. Gunther, "personal communication," *Intel Corporation*, May 2003.

[5] D. Ernst and T. M. Austin, "Efficient dynamic scheduling through tag elimination," *29th Annual International Symposium on Computer Architecture*, May 2002.

[6] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," *27th Annual International Symposium on Computer Architecture*, June 2000.

[7] D. Burger and T. M. Austin, "The SimpleScalar tool set, version 2.0," *Tech. Rep. CS-1342, University of Wisconsin-Madison*, June 1997.

[8] SPEC System Performance Evaluation Committee, www.spec.org.

[9] G. Reinman and N. P. Jouppi, "CACTI 2.0: An Integrated Cache Timing and Power Model," *Tech. Rep. Compaq Western Research Lab*, February 2000.

[10] M. Goshima et al., "A High-Speed Dynamic Instruction Scheduling Scheme for Superscalar Processors," *34th Annual International Symposium on Microarchitecture*, December 2001.

[11] C. Lee, M. Potkonjak, and W. Mangione-Smith, "MediaBench: A Tool for Evaluating Multimedia and

Communications Systems," *30th Annual International Symposium on Microarchitecture*, December 1997.

[12] D. Folegnani and A. Gonzalez, "Energy-Effective Issue Logic," *28th Annual International Symposium on Computer Architecture*, July 2001.

[13] D. Ernst, A. Hamel, and T. Austin, "Cyclone: A Broadcast-Free Dynamic Instruction Scheduler with Selective Replay," *30th Annual International Symposium on Computer Architecture*, June 2003.

[14] M. Huang, J. Renau, and J. Torrellas, "Energy-Efficient Hybrid Wakeup Logic," *International Symposium on Low Power Electronics and Design*, August 2002.

[15] I. Kim and M. H. Lipasti, "Half-Price Architecture," *30th Annual International Symposium on Computer Architecture*, June 2003.

[16] R. Ho, K. W. Mai, and M. A. Horowitz, "The Future of Wires," *Proceedings of the IEEE, 89(4):490-504*, April 2001.

[17] D. S. Henry, B. C. Kuszmaul, G. H. Loh, and R. Sami, "Circuits for Wide-Window Superscalar Processors," *27th Annual International Symposium on Computer Architecture*, June 2000.

[18] S. Palacharla, N. P. Jouppi, and J. E. Smith, "Complexity-effective superscalar processors," *24th Annual International Symposium on Computer Architecture*, June 1997.

[19] P. Michaud and A. Seznec, "Data-flow prescheduling for large instruction windows in out-of-order processors," *7th IEEE International Symposium on High Performance Computer Architecture*, January 2001.

[20] S. E. Raasch, N. L. Binkert, and S. K. Reinhardt, "A Scalable Instruction Queue Design Using Dependence Chains," *29th Annual International Symposium on Computer Architecture*, May 2002.