Wake-Up Logic Optimizations Through Selective Match and Wakeup Range Limitation

Kuo-Su Hsiao and Chung-Ho Chen, *Member, IEEE*

***Abstract*—This paper presents two effective wakeup designs that improve the speed, power, area, and scalability without instructions per cycle (IPC) loss for dynamic instruction schedulers. First, a wakeup design is proposed to aim at reducing the power consumption and wakeup latency. This design removes the READ of the destination tags from the wakeup path by matching the source tags directly with the grant lines. Moreover, this design eliminates the redundant matches during the wakeup operations by matching the source tags with only the selected grant lines. Next, the second design explores a metric called wakeup locality to further reduce the area cost of the wakeup logic. By limiting the wakeup ranges for the instructions in the issue window, this design not only reduces the area requirement but also improves the scalability. The experimental results show that this range-limited-wakeup design saves 76%–94% of the power consumption and reduces 29%–77% in the wakeup latency compared to the conventional CAM-based scheme with only 5%–44% of the area cost in a traditional RAM-based scheme. The results also show that this design scales well with the increase of both the issue width and the window size.**

***Index Terms*—Instruction scheduler, issue window, wakeup locality, wakeup logic.**

I. Introduction

MANY FACTORS such as power consumption, speed, and area used by the dynamic scheduler are important for designing a high-performance microprocessor, and the wakeup logic contributes the most limiting factors to the dynamic scheduler. In a dynamic scheduler, the wakeup logic traces the instruction dependences and wakes the instructions up when their source operands become available. There are two typical implementations for the wakeup logic—the CAM-based and RAM-based schemes which have their respective advantages and disadvantages.

The CAM-based wakeup designs impose a high complexity on the schedulers. In current schedulers, the primary wakeup logic is implemented by using the CAM structures that fully match all the source tags in the issue window with the result tags. However, the CAM structures consume a lot of power and slow down the wakeup latency due to considerable circuit activities and heavy load capacitance. In an effort to extract more instruction-level parallelism, scheduler designs often employ a larger window with more aggressive issue width. In other words, scheduler designs employ accordingly larger and more complex CAM structures that further deteriorate the power consumption and wakeup latency.

The scheduler becomes the major critical path, which limits the clock cycle time, of the pipeline stages mainly due to the complexity of the CAM-based wakeup logic. Although pipelining a dynamic scheduler can increase the clock frequency, the operations of instruction wakeup and instruction selection should be an atomic operation to avoid significant performance degradation. A recent study has shown that the latency associated with the instruction wakeup and selection forms the critical path of the pipeline stages [1]. Increasing both the window size and the issue width will continue to increase the burden to the clock cycle time.

Considering the energy issue, the power consumption associated with the CAM-based wakeup logic constitutes a significant portion of the processor power consumption and may lead to the use of costly cooling systems [2]. For example, the issue logic is the most power-hungry component of the Compaq Alpha 21464 processor; it is responsible for 46% of the total processor power [2]. Similarly, the out-of-order scheduler of the Intel Pentium 4 processor accounts for 40% of the total power consumption. As a result, using a complex CAM-based wakeup logic not only slows down the clock speed but also shifts more power budget to the scheduler.

In addition to the CAM design, an alternative wakeup design is the bit-map RAM scheme that records instruction dependences in the format of bit position in order to wakeup instructions through a READ operation. This design alleviates both the power consumption and the wakeup latency of the CAM scheme. However, the area requirement of this design grows proportional to the square of the issue window size, and the large area requirement may lead to more wire delay in future technologies.

In this paper, we propose two wakeup designs that improve the clock cycle time, power consumption, and area requirement without instructions per cycle (IPC) loss. First, we observe that most tag matches in the wakeup logic are unnecessary during the wakeup operation. As shown in Fig. 1, about 96% of the wakeup operations wake up only one or less (zero) instruction for the left or right source operand. Based on this observation, we propose a selective-match wakeup design, which activates only the match circuit that is selected by predecoding the source tag, to reduce power consumption and latency. Next, we note that most of the wakeup distances between two dependent instructions are short. Based on the selective match design, we introduce a

Manuscript received September 15, 2005; revised March 11, 2006. This work was supported in part by the National Science Council, Taiwan, under Grant NSC 94-2220-E-006-008.

K.-S. Hsiao is with the Department of Electrical Engineering, National Cheng Kung University, Tainan 701, Taiwan, R.O.C. (e-mail: newjimmy@ee.ncku.edu.tw).

C.-H. Chen is with the Department of Electrical Engineering and Institute of Computer and Communication Engineering, National Cheng Kung University, Tainan 701, Taiwan, R.O.C. (e-mail: chchen@mail.ncku.edu.tw).

Digital Object Identifier 10.1109/TVLSI.2006.884150

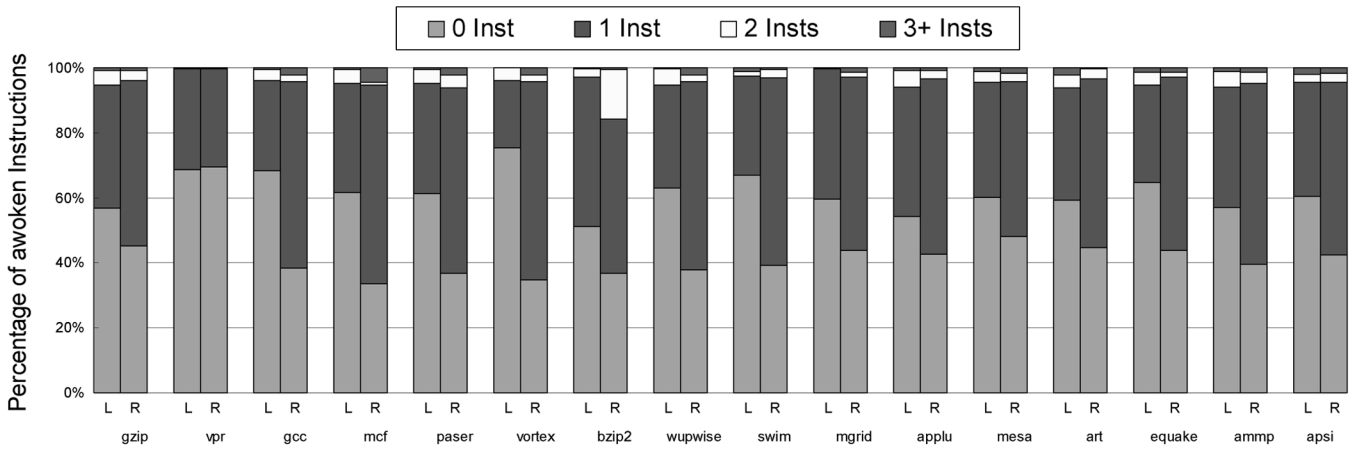

Fig. 1. Number of awoken instructions per wakeup operation for the left and right source operands in a 16-issue, 128-entry processor.

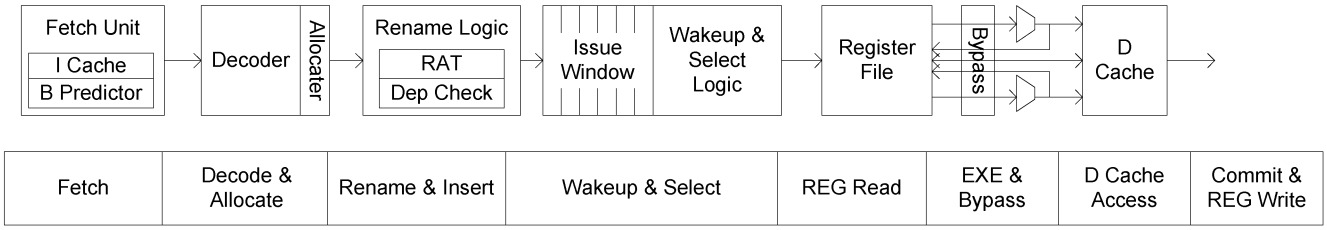

Fig. 2. Baseline superscalar processor model used in this paper.

range-limited-wakeup design that limits the wakeup ranges for the instructions in the issue window to reduce area requirement and improve scalability. The proposed wakeup designs achieve the following features—energy saving, high speed, low area requirement, no IPC degradation, and excellent scalability.

The remainder of this paper is organized as follows. Section II surveys dynamic schedulers used in superscalar processors. Section III reviews two primary wakeup designs. Section IV details the proposed wakeup designs. Section V presents the experimental methodology and the evaluation results. Section VI provides a brief review of previous related works. Finally, Section VII concludes this paper.

II. Dynamic Schedulers for Superscalar Processors

This section provides the background discussions of the dynamic schedulers used in superscalar processors.

A. Scheduling Model Overview

The baseline superscalar processor model is depicted in Fig. 2. The fetch unit retrieves multiple instructions from the instruction cache with a branch predictor to assist in fetching instructions speculatively over basic blocks during a clock cycle time. Subsequently, the instructions are decoded and their register designators are renamed for resolving WRITE after READ (WAR) and WRITE after WRITE (WAW) dependences. Next, the instructions are dynamically scheduled for out-of-order execution. Before execution, their source operands are read from the register file or bypassed from the functional units. Finally, the instructions are committed in program order to ensure the correct completion of the program. This scheduling model is implemented in HP PA8000 [4], Intel Pentium 4 [5], MIPS R10000 [6], Alpha 21264 [7], and its successors.

In the scheduler stage, the wakeup and select logic directs the instructions that are waiting for their source operands to become available or waiting for execution. The select logic selects appropriate instructions for execution from the instructions that have both their source operands ready. The wakeup logic is responsible for waking up the instructions that depend on the selected instructions.

Fig. 3 shows another processor model that dynamically schedules the instructions for out-of-order execution by using a reorder buffer and reservation stations. There are two major differences between this model and our baseline model. The first difference is that the register READ stage is placed before the scheduler stage in this model. In particular, after rename, the available source operands are read from the register file or the reorder buffer and then inserted into the reservation station together with the corresponding source tags. Therefore, the reservation station must have extra data fields to store the values of the source operands. The second difference is that the wakeup operation is triggered by the functional units. Specifically, after execution, both the result tags and result values are forwarded to the wakeup logic to wakeup-dependent instructions. The Intel P6 [8], PowerPC 604 [9], and HAL SPARC64 [10] are implemented based on this model. More discussions on this scheduling model can be found in [11].

B. Wakeup and Select Logic

Fig. 4 shows the scheduling flow of the wakeup and selection logic. After rename, the instructions are inserted into the issue

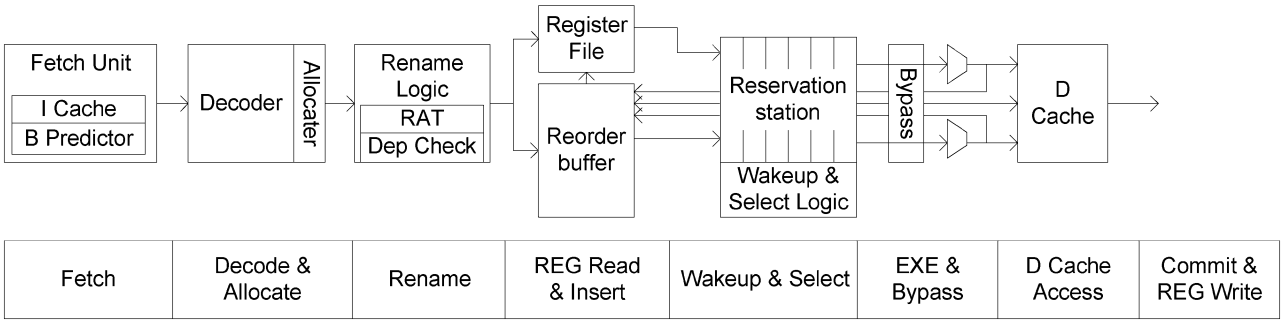

Fig. 3. Superscalar processor model that comes with a reorder buffer and reservation station.

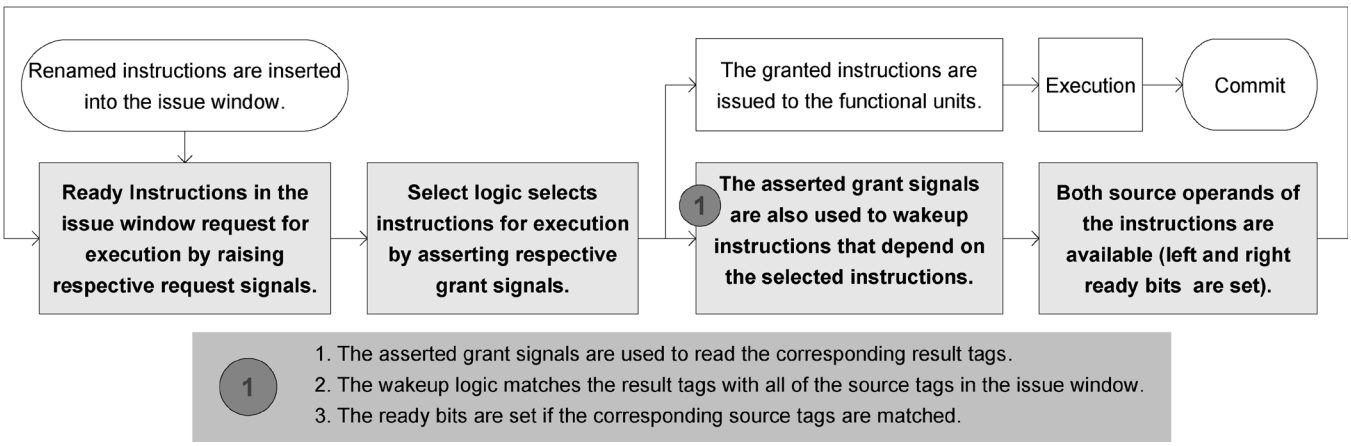

Fig. 4. Flow chart of dynamic scheduling.

window to wait for their operands or wait for execution. The ready instructions, which have both their source operands ready, send signals to the select logic to request for execution. The select logic selects the appropriate instructions for execution from the ready instructions. In particular, once a functional unit becomes available, the select logic then directs a suitable instruction to that unit for execution by asserting the corresponding grant signal. Many selection policies have been presented for the case where the number of ready instructions exceeds the capacity of the available functional units [12], [13], for instance, the oldest first selection algorithm [12].

In the baseline processor, the grant signals from the select logic are used not only to select instructions for execution but also to wake up the dependent instructions. When forwarding the grant lines for the wakeup operations, the grant lines are delayed according to the execution cycle time of the corresponding instructions. Only the grant lines with one cycle execution time are immediately used for the following wakeup operation. To wakeup the dependent instructions, first, the asserted grant lines are used as index addresses to READ their corresponding destination tags. Then, the destination tags are forwarded to the wakeup logic to match with the entire source tags in the issue window. For the matched source tags, their ready bits are set to indicate that the source operands are available. The instructions are marked ready for execution when both their source operands are available. The ready instructions ask for execution by sending the corresponding request signals to the select logic. More details on wakeup designs will be discussed in the following section.

III. Designs of Wakeup Logic: Two Primary Approaches

This section illustrates two typical designs of wakeup logic in more detail. One is the CAM-based wakeup logic [1] and the other is the RAM-based design [14].

A. CAM-Based Approach

Conventional implementation of the wakeup logic is based on the CAM structure as shown in Fig. 5(a). The destination tag RAM, shown at the top of the figure, is used to store the renamed destination tag. In addition, two CAM structures are used to match the destination tags with the left and right source tags, respectively. Two ready bits (Rdy L and Rdy R) are employed for each entry to indicate whether their corresponding operands are available. The wakeup operations begin when the select logic asserts the grant lines. Assuming n is the issue window size and w is the issue width. The select logic asserts at most w of the n grant lines to select w instructions for execution and to wakeup the instructions that depend on the selected instructions. The asserted grant lines are used to retrieve the corresponding result tags from the destination tag RAM. The result tags are then driven on the tag buses (Tag 1 to Tag w) to the CAM structures to match with the left and right source tags (Tag L and Tag R).

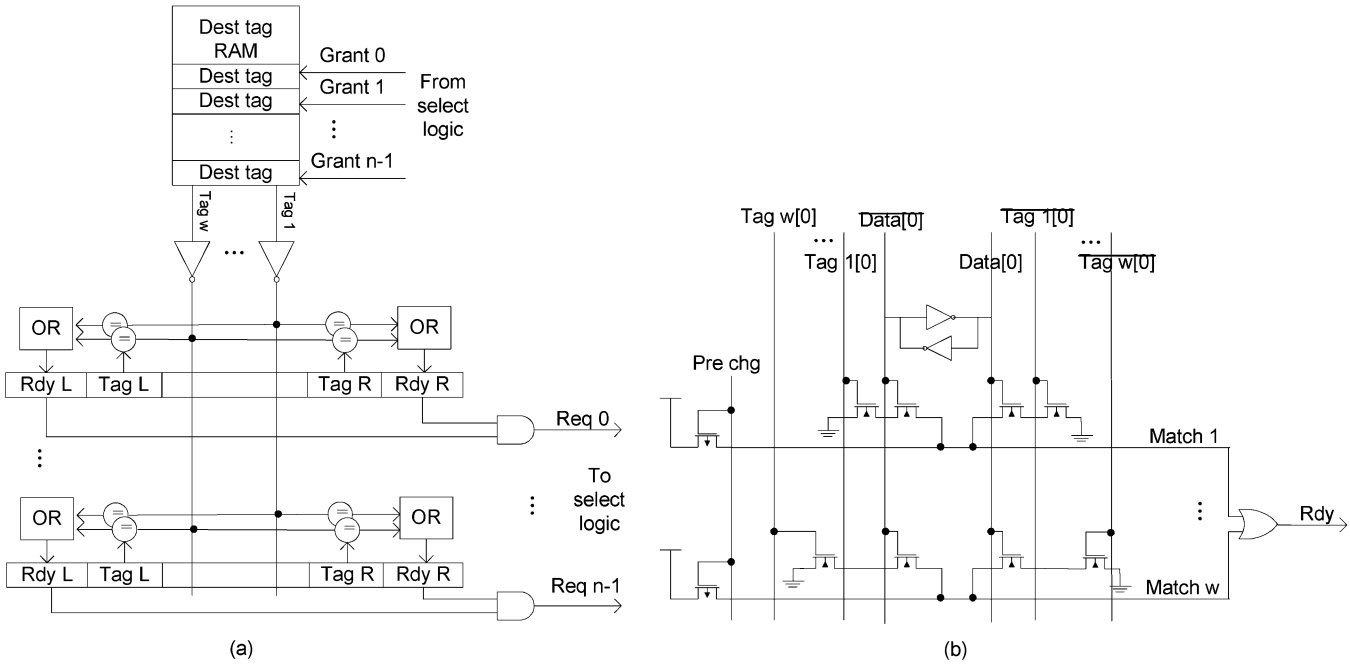

Fig. 5. (a) CAM-based wakeup logic. (b) Single cell of the CAM structure.

If there is a match, the corresponding ready bit is set to indicate that this operand is available. For each entry, the left and right ready bits are ANDed to generate one of the n execution requests (Req 0 to Req $n-1$).

The detail of how the destination tags match with the source tags can be illustrated with Fig. 5(b). At the center of the upper part, two chained inverters represent a memory cell that stores a single bit of the source tag (Data[0], for instance). On both sides of this memory cell, the corresponding bits ($\overline{\text{Tag}1[0]}$ to $\overline{\text{Tag}w[0]}$) of the result tags and their complements ($\text{Tag}1[0]$ to $\text{Tag}w[0]$) go vertically through the cell to match with the bit of the source tag. At the bottom of this figure, two NMOS transistors form a comparator and w pairs of comparators to match the w incoming tag bits with the data bit. There is a precharged match line, reacting to the match result. If the incoming tag bit does not match with the data bit, the corresponding match line is pulled down by one of the two comparators. The match line only remains high if all the bits of the result tag are matched with the corresponding bits of the source tag. Finally, the ready bit (Rdy) is produced by ORing all of the match lines. If one of the w result tags is matched with the source tag, the corresponding match line is high and the ready bit is set to indicate that this source operand is available now.

1) Gate-Off Scheme: Based on the CAM-based wakeup logic, the gate-off scheme dynamically disables the needless wakeup operations to reduce power consumption [15]. In this scheme, the entries of the issue window are classified into three types according to their status to gate-off unnecessary wakeup operations. First, the entries that have invalid instructions or have no instruction are called empty entries. These empty entries need not be searched during the wakeup operation. Second, the entries that have valid instructions with available operands are called ready entries. These ready entries need not be searched either during the wakeup operation because the operands are already available. Finally, the entries that have valid instructions with unavailable operands are the only entries that need to be searched during the wakeup operation. To put it precisely, the scheme avoids the source tags matching with the result tags by disabling the precharge of the match lines for the empty and ready entries in the CAM structures.

B. Bit-Map RAM Scheme

In addition to the CAM-based design, an alternative approach is implemented by using the bit-map RAM structure [14].

Fig. 6 shows this RAM-based wakeup design. This wakeup design employs two bit-map memory structures, of which the height and width are both identical to the issue window size, to handle the wakeup operations. In this RAM structure, each bit position represents the data dependence between two instructions. For example, the bit located at the intersection of the ith column and the jth $(0 \leq i, j \leq n-1)$ row in the bit-map RAM indicates that the jth instruction requires a source operand from the output of the ith instruction. In the same way, other instructions that depend on instruction i will set the corresponding bit in column i. Different from the structure of a general RAM which outputs a row of data, this bit-map RAM outputs a column of data. To wakeup the dependent instructions, the grant lines READ the corresponding columns from the bit-map RAM. If the bits on the selected column have been set, the corresponding ready bits are set to indicate that the result is now available for the waiting instructions.

The detail of the access to the bit-map RAM can be illustrated with Fig. 7. The source tags are inserted into the bit-map RAM in the following way. For each instruction, its entry number in the issue window is used as the address to select a row, driving the selected WRITE wordline as shown at the left side of the figure. Besides, the source tag of the instruction is decoded to assert the bitline for the selected row as shown at the bottom

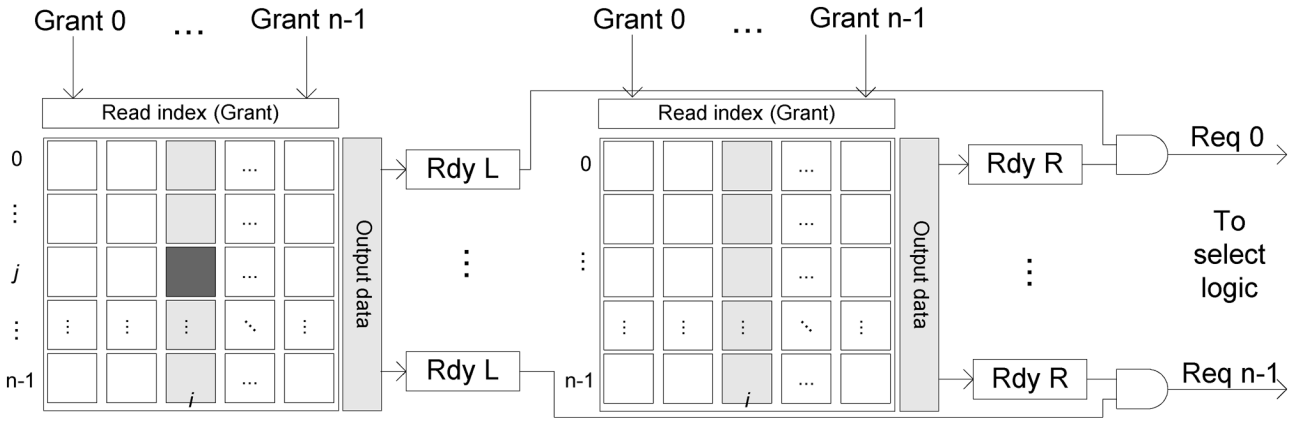

Fig. 6. Wakeup logic implemented by using the bit-map RAM.

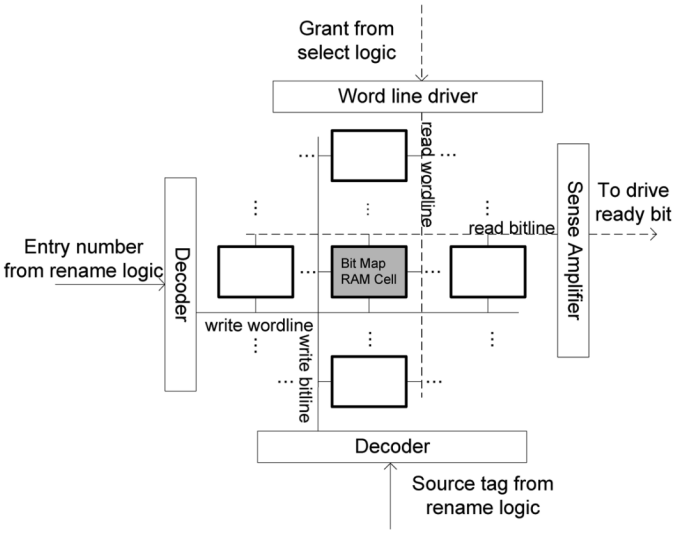

Fig. 7. Bit-map RAM circuit.

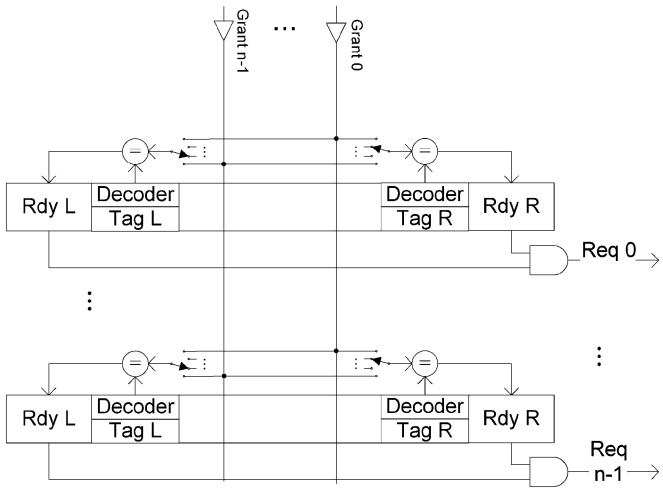

Fig. 8. Selective match wakeup logic.

of the figure. In other words, only one bit is set when writing a source tag into the wakeup logic. As an illustration, the instruction with its entry number $= j$ and source tag $= i$ is inserted in the following way. The entry number is used as the address to index the jth wordline and the source tag drives the ith bitline. Consequently, the cell located at the intersection of the ith column and the jth row is set high to specify that instruction j depends on instruction i.

On the other hand, the instruction wakeup operation is performed by means of a READ operation. The grant signal from the select logic drives the corresponding READ wordline to READ out a column of data. If the cells at the selected column have been set, they pull down their READ bitlines. These bits at the output column drive the ready bits according to their bit position. For example, the select logic asserts the ith grant line to select instruction i for execution. This grant line is also used to wake up the dependent instructions, driving the ith READ wordline to select column i for output. Because the jth cell at the selected column has been set previously, the jth READ bitline is pulled down by this cell, which then sets the corresponding ready bit.

This RAM-based wakeup design has the advantages in power dissipation and latency; however, it incurs a large area cost. The wire delay of the wakeup path increases as the logic size grows, leading to excessive wire delay in the processes of future technology [16]. Besides, the bit-map RAM may not suit for a large window processor due to its prohibitive area requirement.

IV. Optimizations of Wakeup Logic

In this section, we present two wakeup designs that are efficient in terms of energy usage, latency, area, and scalability without performance degradation. Based on the observation that the wakeup operation only wakes up a few instructions, our first design uses a novel selective match circuit that only matches the candidate grant line with the predecoded signal from the source tag. This scheme improves the wakeup latency as well as the energy usage of the scheduler. In addition, based on the observation that the wakeup distances between two dependent instructions are often short, the second design limits the wakeup ranges for the instructions in the issue window to improve the area cost and scalability without the compromise of IPC.

A. Reducing Energy and Latency via Selective Match

Fig. 8 shows the selective match wakeup design that trades the costly destination tag memory with a simple combinational

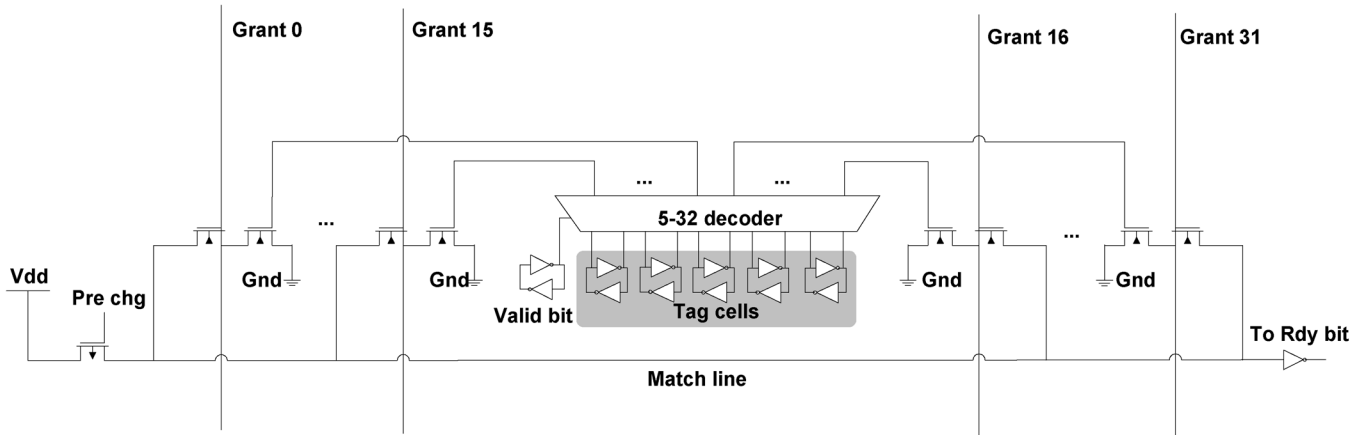

Fig. 9. Basic wakeup unit.

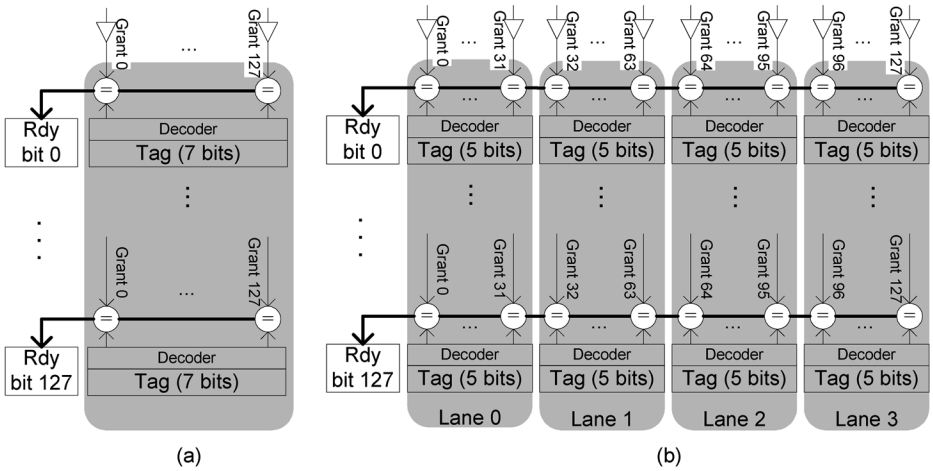

Fig. 10. (a) Using expanded wakeup units. (b) Using multiple lanes of basic wakeup units to form the wakeup logic for a 128-entry issue window.

circuit by using source tag predecoding (STP). Different from the CAM-based design, the grant lines are directly connected to the wakeup logic without the need for reading the destination tags. In addition, the source tag is decoded in advance before the tag match process. In this way, only the corresponding grant line is matched with the predecoded output of the source tag. This wakeup design is constructed by using the basic wakeup unit shown in Fig. 9.

The basic wakeup unit is the circuit that matches a 5-bit source tag with 32 grant lines. At the center of Fig. 9, the two chained inverters stand for a RAM cell. Five cells store the source tag and another one stores the valid bit that indicates whether this tag is valid. The 32 grant input lines, which are represented by the vertical lines (Grant 0 to Grant 31), are arranged equally on both sides of the tag cells. Each of the 32 grant lines is connected to the match circuit to match with the corresponding decoded line from the tag cells. The match circuit, which consists of two nMOSFET transistors, matches the grant line and the decoded line that are both connected to it. A match line, at the bottom of the figure, reacts to the match result.

For wakeup operation, the 5-bit tag cells are decoded to drive one of the 32 output lines. The asserted line activates the match circuit to wait for the corresponding grant line while the rest of the decoded lines turn off the 31 match circuits connected to them. When the waited grant line becomes high, the corresponding match circuit is turned on to pull the match line down. In short, only one of the 32 match circuits is activated to wait for the grant line.

There are two approaches for building up the wakeup logic for a large issue window design, for instance a 128-entry window. The first approach uses the expanded wakeup unit, which is an expansion of the basic wakeup unit, to build up the large wakeup logic. Fig. 10(a) shows an example of this approach for a 128-entry design. Each entry of the wakeup logic uses the expanded wakeup unit that matches the 7-bit source tag with the incoming 128 grant lines. In the expanded wakeup unit, the 7-bit source tag is predecoded by a 7-to-128 decoder. The 128 output lines of the decoder and the incoming 128 grant lines are matched as that in the basic wakeup unit.

In reality, arranging the 128 decoded lines in silicon circuit has posed an adverse effect on the area cost. For instance, with the experimental environment in which the technology process supports six metal layers, we employed four metal layers to route the 128 decoded lines. To reduce the area used, the 128 decoded lines can be divided into two groups going to the left and right sides to match with the grant lines, respectively. To route the 64 decoded lines for each direction, 16 rows of the

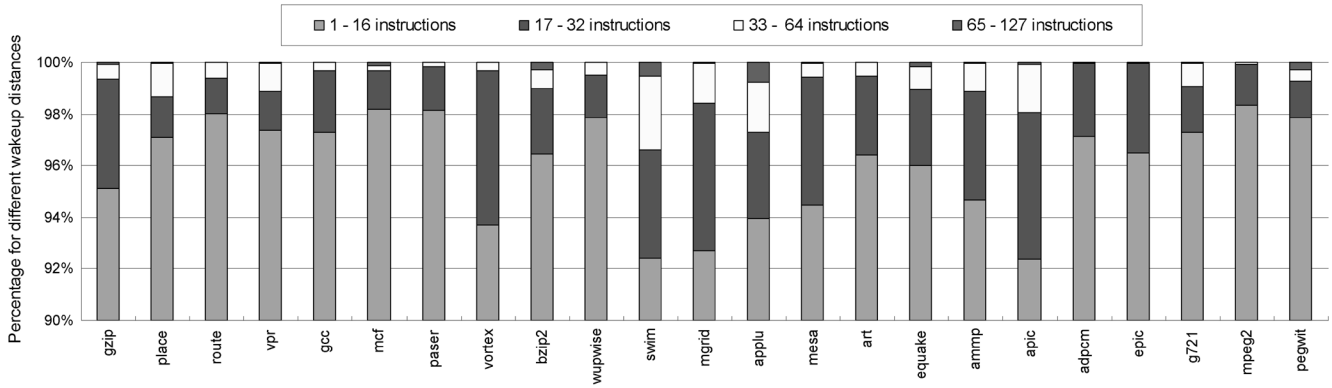

Fig. 11. Distribution of wakeup distances.

four metal layers were used. The area cost due to routing these decoded lines becomes unacceptable when the size of the issue window increases. As a result, in a four-issue processor the area cost of this approach is 133% that of the bit-map RAM scheme for a 128-entry issue window and it becomes 235% when the size of the issue window grows to 256 entries. Besides, the area due to routing the decoded lines increases the wire resistance of the grant line bus. The increased wire resistance deteriorates the wakeup latency and even the power consumption.

To tackle the previous problem, the second approach builds up the wakeup logic by merging multiple lanes of the basic wakeup units. For example, Fig. 10(b) depicts the wakeup logic that employs four lanes of the basic wakeup units for a 128-entry issue window. This structure handles the wakeup operations for the left or the right source operands in the issue window. The 128 grant input lines are divided into four groups as the inputs for the four lanes, respectively. Each lane of this wakeup logic, which consists of 128 entries of the basic wakeup unit, handles the wakeup operations for its incoming 32 grant lines. In this design, there are four basic wakeup units for each entry (row) and their match lines are connected together (wire OR) to drive the ready bit.

When an instruction is inserted in the issue window, the source tag of this instruction is written into the allocated entry in the following way. First, the most significant two bits of the source tag are used to select one from the four basic wakeup units in the allocated entry. Then, the least significant five bits are written into the tag field of the selected unit. The other three tag fields in this entry are set to be invalid. The 5-bit source tag is decoded to drive a decoded line that is corresponding to the source tag. For example, if the value of the source tag equals to i, the ith decoded line will be asserted to wait the ith grant line. For the next instruction, the next entry of the STP wakeup structure is allocated. In that entry, the written source tag drives the corresponding decoded line to wait for the grant line that is indexed by the source tag. In summary, only one tag field is valid even four basic wakeup units are employed for an entry. In each entry, only one decoded line would be asserted by the written source tag.

This design greatly improves the disadvantages of the CAM and RAM schemes. First, compared to the CAM-based design, the inputs to this wakeup logic are the grant lines rather than the outputs from the destination tag RAMs. In addition, at most, only w grant lines are driven rather than x tag bits are driven where

$$w = \text{issue width}$$

and

$$x = \text{issue width} \times \text{tag length}.$$

Moreover, only one match circuit is active for each entry and only the match line that depends on the execution result is activated. These optimizations make the STP design faster and more energy efficient than the CAM-based design. On the other hand, unlike the bit-map RAM area that grows with the square of the issue window size, the current design reduces the area cost by separating the tag RAM cells from the match circuits.

B. Improving Area Cost via Wakeup Range Limitation

The observation on the wakeup distances between two dependent instructions motivates the second optimization. Fig. 11 shows the statistics of the wakeup distances between two dependent instructions for the benchmark programs. The wakeup distance is the instruction count between two dependent instructions. On average, 96% of the wakeup operations come with the wakeup distances less than 17 instructions. Based on this observation, a compact wakeup design that supports the range-limited wakeup operation is proposed to reduce the area requirement and improve the scalability.

Fig. 12 shows the 128-entry example of the proposed compact wakeup design with a backup unit. The compact design, shown at the upper part, handles the wakeup operations only for the instructions with the wakeup distances in the limited wakeup range. In addition, a backup unit, shown at the lower part, is employed to deal with the instructions that their wakeup distances are out of range. In this design, the issue window is assumed to work as a circular ring. Thus, the grant lines interfaced with the compact sets are also arranged in a circular order.

The compact wakeup design consists of eight compact sets (S0–S7). Each compact set is formed by using 16 entries of the basic wakeup unit to handle the wakeup operations only for the

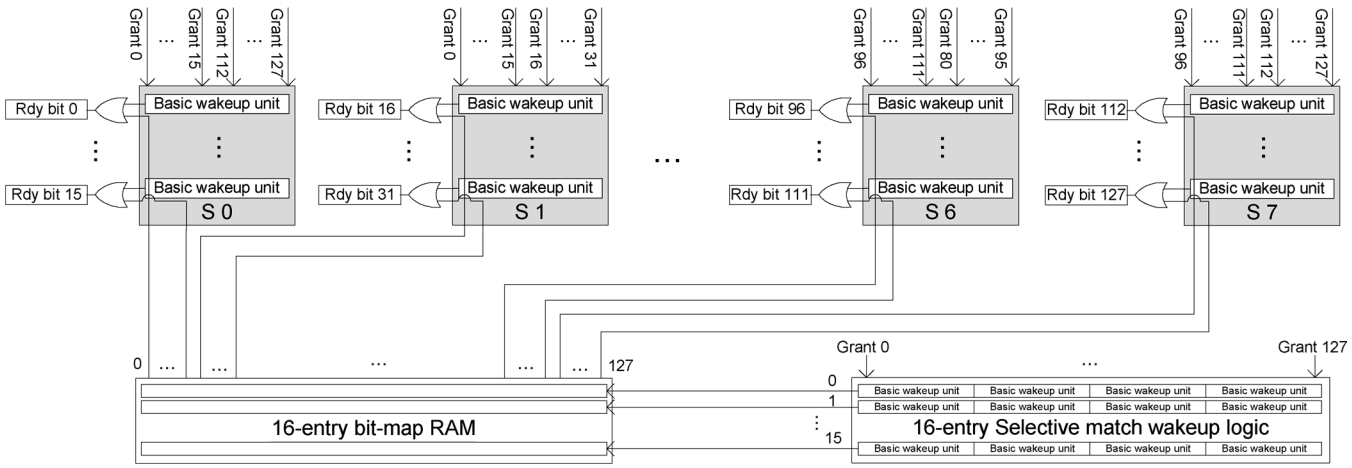

Fig. 12. Compact wakeup design for 128-entry issue window with a backup unit.

instructions assigned to it. For each compact set, the 32 grant input lines are arranged so that the instruction in the first entry of the compact set begins with a wakeup range of 16 instructions. For the instruction in the second entry, its wakeup range increases by one. Thus, the instruction at the last (16th) entry of the compact set has a wakeup range of 31 instructions. In summary, all the instructions in the compact set have the wakeup range of 16 instructions at least.

For example, the first compact set of the compact design handles the wakeup operations for the instructions in the issue window from entry 0 to entry 15. The instruction in entry 0 has the wakeup range of 16 instructions (from entry 112 to 127). The instruction in entry 1 has the wakeup range of 17 instructions (from entry 112 to 0, wrapping around). For the instruction in entry 15, the wakeup range increases to 31 instructions (from 112 to 14). In a similar fashion, the other compact sets handle the wakeup operations for the rest of the instructions in the issue window.

Because the compact set only stores the low-order five bits of the source tag, the connection of the grant lines to the compact set must conform to the order of the decoded line as shown in Fig. 9. Specifically, the 32 grant lines for a compact set are divided into two groups where the first 16 lines are represented by $G_{\mathrm{L}}(G_{\mathrm{L0}} - G_{\mathrm{L15}})$ and the rest are denoted as $G_{\mathrm{H}}(G_{\mathrm{H0}} - G_{\mathrm{H15}})$. The relationship for interfacing the grant lines with the compact sets can be written as follows:

$$G_L = \left\{ \begin{array}{ll} E-16, & \text{if } S = \text{odd} \\ E, & \text{if } S = \text{even} \end{array} \right\}$$

$$G_H = \left\{ \begin{array}{ll} E, & \text{if } S = \text{odd} \\ (E-16+\mathrm{IW}_{\mathrm{size}}) \bmod \mathrm{IW}_{\mathrm{size}}, & \text{if } S = \text{even} \end{array} \right\}$$

where E is the entry numbers of the compact set, $\mathrm{IW}_{\mathrm{size}}$ is the size of the issue window, and S is the number of the compact set. For example, the first compact set (S0) is connected to the grant lines numbered from 0 to 15, the same as the entry numbers of the first compact set, for the first half (G_L) and the grant lines numbered from 112 to 127 for the second half (G_H). For the second compact set (S1), the first half is the grant lines numbered from 0 to 15 while the second half is the grant lines numbered from 16 to 31, the same as the entry numbers of the second compact set. Fig. 12 also illustrates the rest of the connections.

TABLE I
VALID DISTANCE CODE FOR A COMPACT SET FOR A 128-ENTRY ISSUE WINDOW

Compact set	S 0	S 1	S 2	S 3	S 4	S 5	S 6	S 7
Distance codes	111 000	000 001	001 010	010 011	011 100	100 101	101 110	110 111

The access to this compact wakeup logic is different from the access to the STP design. After rename, the source tags are inserted into the compact sets only if the wakeup distances of the source tags are within the specified wakeup range. The wakeup distance of the instruction is easy to determine by examining the most significant bits of the instruction's source tag. For the example in Fig. 12, the most significant three bits of the source tag are compared with the two distance codes listed in Table I according to the allocated compact set. Note that the instruction entry number obtained from the issue window determines which compact set the instruction is assigned to. For a compact set, the two distance codes listed in Table I come from the three high-order bits of the two grant line groups (G_H and G_L), respectively. If the three high-order bits of the source tag match with one of the two distance codes of the allocated compact set, the wakeup distance of the instruction is in the range of this compact set. Then the least significant five bits of this source tag are inserted into the allocated entry. If the three high-order bits do not conform to any of the two codes of the allocated compact set, the wakeup distance is out of the range of this compact set and this tag is inserted into the backup unit.

A backup unit, which is shown at the lower half of Fig. 12, is employed to deal with the instructions that have the wakeup distances out of the limited range. This backup unit consists of a 16-entry 128-input STP design and a 16-entry 128-output bit-map RAM structure. Since this backup unit only takes care of the wakeup operations that occur infrequently, the design here uses 16 entries. Further discussions on the number of entries used for the backup unit will be elaborated in the result section. In this backup unit, the STP wakeup logic is used for the purpose of serving the out-of-range wakeup operations and the bit-map

TABLE II
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, 2048-entry bimodal 8-bit history, 2048-entry level 2 1024-entry chooser 4-way, 1024-entry BTB/ 16-entry RAS(return address stack)/ 8-cycle penalty.		

RAM structure is used to record the instruction entry numbers in the issue window for the instructions in the STP wakeup structure. In particular, the bit position in the rows of the bit-map RAM represents the entry number in the issue window for the instructions in the STP wakeup logic. The STP outputs, which are the match results, are used to READ the corresponding rows of the bit-map RAM. Each bit of the output of the bit-map RAM is connected to the corresponding ready bit.

Specifically, when an instruction exceeds the wakeup distance of its designated compact set, an entry with the same index in both of the STP wakeup logic and the bit-map RAM is allocated for this instruction. The source tag of the instruction is then inserted into the STP logic to wait for the corresponding grant line. In addition, of the same selected entry in the bit-map RAM, the bit that corresponds to the instruction's entry number in the issue window is set. On the contrary, if all the entries in the backup unit are allocated, instruction dispatching is stalled until the backup unit is available again. Once the source tag matches with a grant line, the match line from the matched entry of the STP logic is asserted as an index to READ the bit-map RAM. The asserted bits of the output from the bit-map RAM are used to set the ready bit through the OR gates.

With the backup unit, this design significantly reduces the used area compared to the STP wakeup logic without IPC loss. Since the wakeup range is limited, each entry of the compact set consists of only one basic wakeup unit regardless of the issue window size. Although the range-limited design may induce dispatch stalls that deteriorate the performance (IPC), these stalls can be avoided by using a 16-entry backup unit.

V. EXPERIMENTAL EVALUATION AND ANALYSIS

This section presents the evaluation methodology and discusses the latency, power, area, and performance of the proposed optimization schemes and the conventional wakeup designs.

A. Experimental Methodology

The power consumption and IPC results were obtained through architectural simulation, which was conducted by using Wattch [17] and SimpleScalar [18] toolsets. These execution-driven simulators simulate a superscalar processor with two-level caches, branch predictors, dynamic scheduler, and *et al.* by performing cycle by cycle instruction-level simulation, including execution down any speculative path until a branch misprediction is detected.

Table II lists the architectural parameters for the three different configured processors evaluated in this study. In Wattch, the basic wakeup unit was modeled in three components. First, the valid bit and the source tag were modeled as a 6-bit general RAM structure. Second, the decoder that predecodes the source tag was modeled as a 5-to-32 decoder that is composed of the NAND gates. Next, the 32 comparators that match the decoded lines with the grant lines were modeled as the match circuits shown in Fig. 9. Besides, the RAM cell of the bit-map RAM structure was modified from the cell in the conventional RAM. In particular, as shown in Fig. 7, the READ wordlines were modified to go vertically and the READ bitlines were modified to go horizontally. The other configurations for the Wattch include 1-GHz clock frequency, 1.8-V voltage, and 0.18-μm technology process.

The simulation results were collected from seven-integer and nine-floating point programs of the SPEC2000 benchmark suite. The test input set was used for the benchmark programs. Additionally, five programs of the Media-bench [19] were also employed for a more comprehensive evaluation of the wakeup designs. All of the selected benchmark programs were compiled with full optimization (-O4) and were run to completion.

To evaluate the wakeup delays and area sizes, the circuit characteristics of the different structures must be examined. The circuit models were extended from the one proposed by Ernst and Austin [20] and the timing results for the designs were extracted by using the Avant! Hspice tool. For the area sizes, only the structures of the evaluated designs were estimated, for example, the CAM structure, the bit-map RAM structure, the STP structure, the STP design constrained with wakeup locality (STP-WL), and the backup unit structure. The ready bits, request signals, and the interconnections outside the structures were not estimated into the area sizes. The area sizes of the wakeup designs were estimated based on the cells and wires in [21] and were scaled to 0.18-μm process. Finally, the parameters of CMOS transistors and wires were all conformed to the design rules of the TSMC 0.18-μm process.

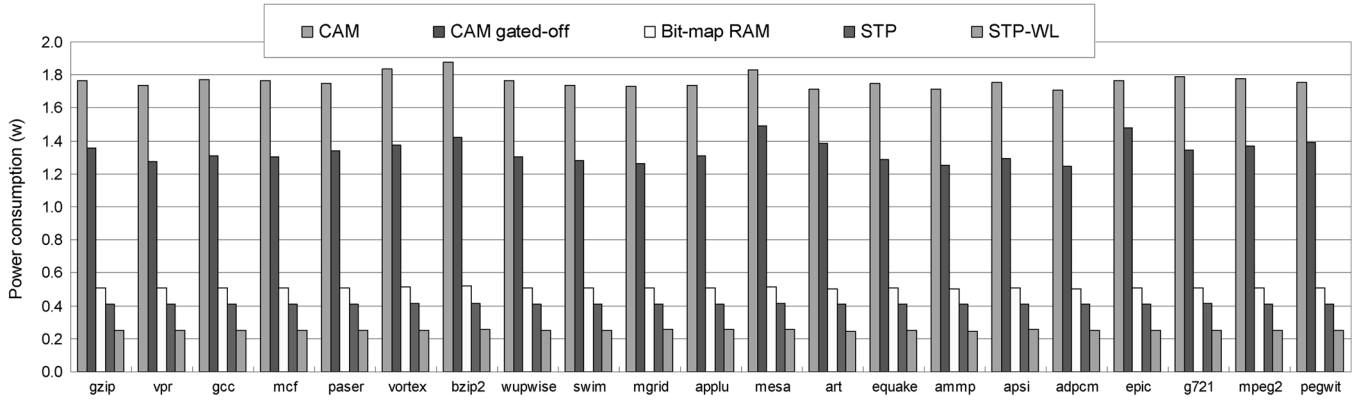

Fig. 13. Power consumption for an 8-issue 64-entry processor (w).

TABLE III
POWER CONSUMPTION STATISTICS (W)

	4-issue 32-entry processor					8-issue 64-entry processor					16-issue 128-entry processor				
	CAM	Gate-off design	Bit-map RAM	STP design	STP-WL design	CAM	Gate-off design	Bit-map RAM	STP design	STP-WL design	CAM	Gate-off design	Bit-map RAM	STP design	STP-WL design
Int avg.	0.460	0.365	0.126	0.130	0.109	1.79	1.34	0.511	0.412	0.251	10.17	7.48	2.58	1.68	0.568
FP avg.	0.441	0.350	0.124	0.129	0.108	1.75	1.32	0.508	0.410	0.252	10.12	7.48	2.57	1.68	0.575
Med avg.	0.459	0.378	0.126	0.130	0.108	1.76	1.37	0.509	0.411	0.252	10.10	7.64	2.57	1.68	0.568
Total avg.	0.452	0.362	0.126	0.130	0.108	1.76	1.34	0.509	0.411	0.252	10.13	7.52	2.57	1.68	0.571

B. Power Consumption Statistics

Fig. 13 presents the power consumptions for the five wakeup logics in an 8-issue processor with 64-entry issue window. The power consumption of the CAM-based wakeup design, shown at the left most bars, is found to be the most power hungry of all. This is due to the READ operation for the destination tags, the heavy load capacitance, and the surplus activities of the CAM structure. Although the gated-off CAM scheme reduces the power consumption by eliminating the redundant matches for the source tags in the ready and empty entries of the issue window, the power consumption is still high as shown in the second bars due to the inherent nature of the CAM structure.

The power consumed by the bit-map RAM scheme, shown in the middle bars, is about only 29% of that consumed in the CAM scheme. This significant reduction comes from the fewer amounts of activities incurred for each wakeup operation in the RAM structure.

Similarly, less grant and match lines are activated in the selective match wakeup scheme during the wakeup operation. For a 64- and 128-entry processor, the smaller area size (shown later) of the STP scheme leads to smaller load capacitance, which results in further power reduction, compared to the RAM-based scheme.

The final bars show that the STP-WL scheme achieves the best power saving among the five schemes. This advantage comes from the slight load capacitance of the small compact set. This design reduces the power consumption of the STP scheme by 39% on the average. Specifically, the power consumption of the STP-WL design is only 14% that of the CAM-based design and 50% that of the bit-map RAM design.

Additionally, the power consumption of these wakeup designs for different configurations is shown in Table III. Examining the 16-issue, 128-entry configuration, the conventional CAM design consumes about 10 W while the STP-WL design dissipates only about 0.57 W. The STP-WL design takes this noticeable advantage by using the compact set that is much smaller and less complex than the other designs. The size of the compact set is constant (16 entries) and does not grow with the issue window size. Importantly, the STP-WL design is scalable with the issue window size in terms of power consumption.

C. Circuit Delay Results

Since the gated-off CAM scheme only gates off the match lines in the empty and ready entries, this does not affect the critical path of the wakeup operation. Thus, the latency is the same as that in the CAM-based design. Hence, the wakeup delay of these two wakeup logics can be summarized as follows:

$$T_{\mathrm{CAM}} = T_{\mathrm{tagread}} + T_{\mathrm{tagdrive}} + T_{\mathrm{tagmatch}} + T_{\mathrm{matchOR}}$$

where T_{tagread} is the time for reading the destination tag from the tag RAM, T_{tagdrive} is the time for driving the tag into the CAM structure, T_{tagmatch} is the time spent by the match circuit in pulling the match line low, and T_{matchOR} is the time for performing a logical OR operation with the match lines.

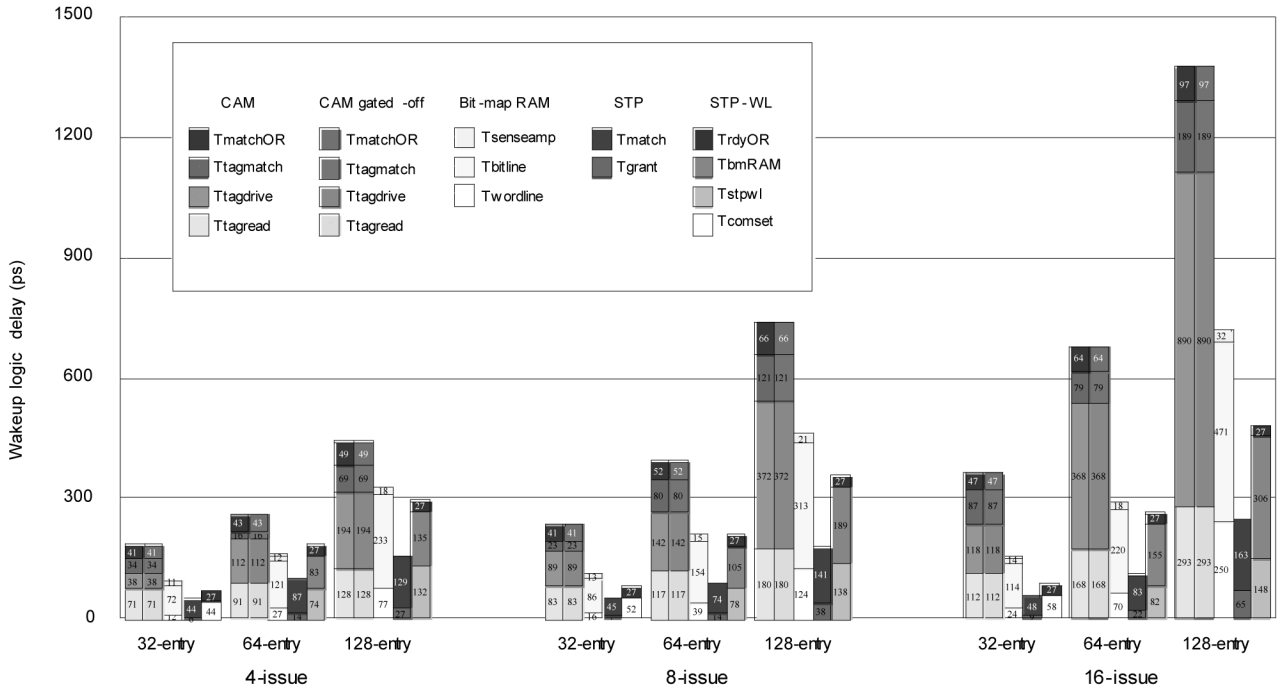

Fig. 14. Wakeup latencies of different designs (ps).

Analyzing the delay of the bit-map RAM scheduler is based on the circuit shown in Fig. 7. The wakeup latency of the bit-map RAM scheme can be represented as follows:

$$T_{\mathrm{bmRAM}} = T_{\mathrm{wordline}} + T_{\mathrm{bitline}} + T_{\mathrm{senseamp}}$$

where T_{wordline} is the delay as the word line driver drives the grant signal into the bit-map RAM, T_{bitline} is the time for activating the bit line, and T_{senseamp} is the time for amplifying the bit line.

The representation of latency for the selective match wakeup (STP) design is similar to that of the bit-map RAM wakeup logic and can be expressed as

$$T_{\mathrm{stp}} = T_{\mathrm{grant}} + T_{\mathrm{match}}$$

where the delay components T_{grant} and T_{match} are the time for driving the grant line and pulling the match line down, respectively.

For the STP-WL design, in the 32-entry configurations the wakeup delays come directly from the compact set because no backup unit is used in these configurations. In the other configurations that the issue window size is greater than 32 entries, the wakeup latency is the maximum latency of the compact set or the backup unit. After evaluation, we found that the maximum wakeup delay always comes from the backup unit that is given by

$$T_{\mathrm{backup}} = T_{\mathrm{stpwl}} + T_{\mathrm{bmRAM}} + T_{\mathrm{rdyOR}}$$

where T_{stpwl} and T_{bmRAM} are the delays of the 16-entry selective match unit and 16-entry bit-map RAM, respectively. T_{rdyOR} is the time for ORing the two ready lines from the compact set and the backup unit.

Fig. 14 shows the wakeup latencies of the five wakeup designs for various configurations. The wakeup latencies of the bit-map RAM scheme, selective match scheme (STP), and STP-WL are presented in the last three bars. Compared to the CAM-based schemes, these three designs have noticeable improvements that result from the lower capacitance on the wakeup path and equally from the fact that the READ of the destination tag is removed from the critical path.

The STP design achieves the best wakeup delay in all configurations. The wakeup delay of the STP design is more prominent compared to the conventional CAM and RAM schemes when the scheduler comes with a wider issue width and larger issue window. This result comes from that of the STP design separates the source tag from the match circuits in the basic wakeup unit to restrain the increase of the load capacitance on the wakeup path.

The last bars in Fig. 14 show the wakeup latency of the STP-WL design. The wakeup latencies come directly from the compact set for the 32-entry configurations because no backup unit is employed in these configurations. For the 64/128-entry configurations, the backup unit, which includes 16 entries of the selective match wakeup logic and 16 entries of the bit-map RAM, contributes to the total wakeup delay. As expected, the wakeup latencies of STP-WL are higher than that of the STP scheme. Despite this, the wakeup latencies of the STP-WL design are measured to be only 23%–71% of the CAM schemes in the evaluated configurations. Besides, the STP-WL design is 10%–44% faster than the RAM scheme for the 32-entry and 128-entry configurations.

The STP and STP-WL designs suit for sophisticated schedulers in terms of the latency. For the purpose of our study, we assume that the wakeup and select logic form the critical path of the pipeline stages. The STP-WL design has significantly improved the clock cycle time by 56% and 25% compared to the CAM and RAM schemes for a 16-issue 128-entry processor. Having achieved this, the critical path may migrate to other

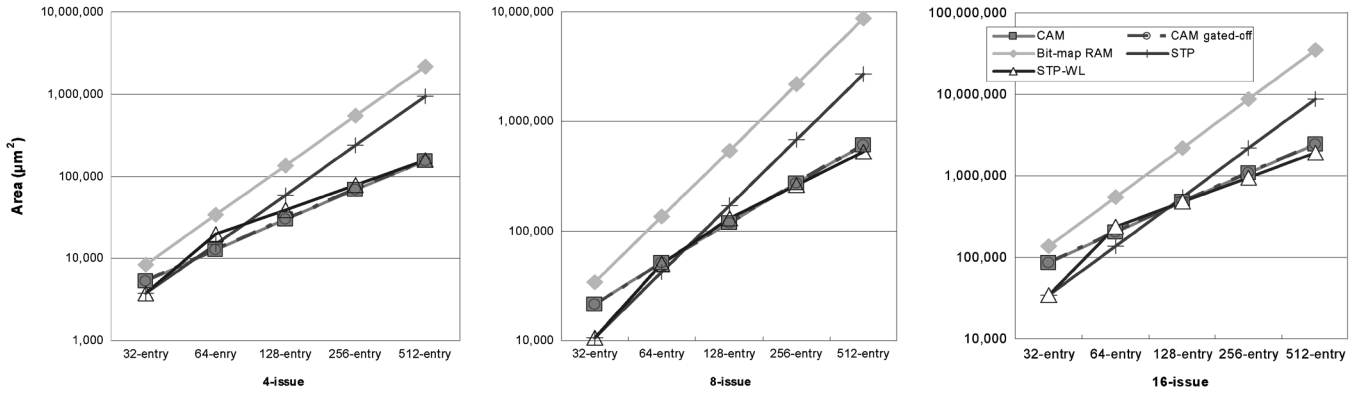

Fig. 15. Area size of the evaluated wakeup logics (μm^2).

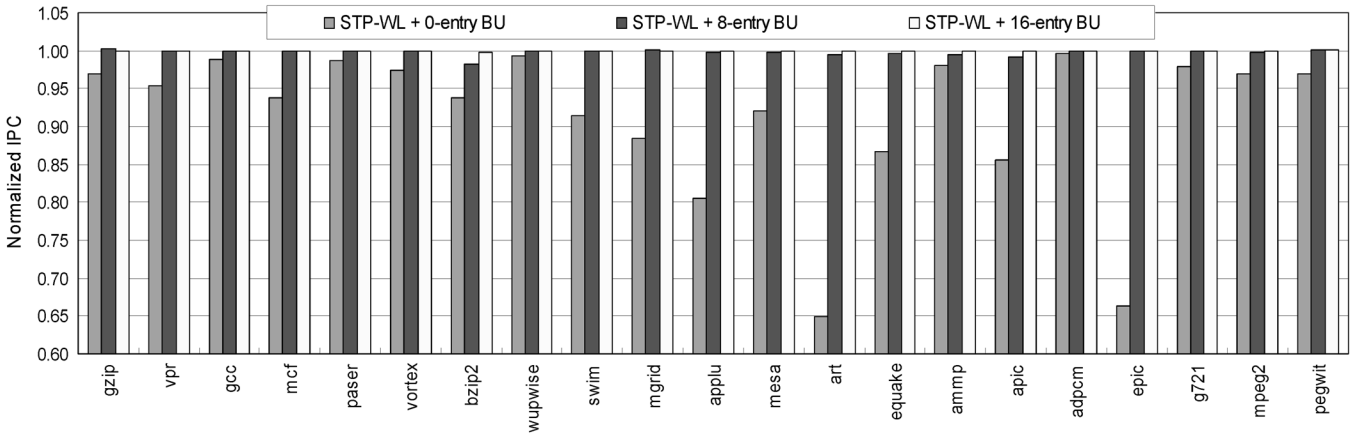

Fig. 16. IPC loss due to the limit of wakeup distance with three different configured backup units.

pipeline stage since the scheduler has been improved by the STP-WL design.

D. Area Size Tradeoff

Fig. 15 shows the area sizes of the evaluated wakeup designs for three different issue widths of various window sizes. Note that the scale is in logarithm. The area of the bit-map RAM scheme, shown by the first line from the top, is proportional to the square of the issue window size and becomes unacceptably large in a larger window scheduler.

Similar to the RAM scheme, the number of entries of the STP design grows with the issue window. However, the separation of the source tag from the match circuits effectively reduces about 70% of the area compared with the RAM scheme in 128-entry configurations.

For the CAM-based schemes, the entry number increases with the issue window size but the tag length only increases one bit as the issue window size doubles. Thus, the area of the CAM-based schemes grows only linearly with the issue window size. However, it is measured to be worse than the STP scheme in smaller window schedulers due to the larger size of the CAM cell.

In the wide-issue and large-window processors, the STP-WL design effectively reduces the area cost compared to the STP scheme. As shown, the area needed for the STP-WL scheme is similar to that required in the conventional CAM-based designs. The STP-WL design also scales well with the increase of the issue window size especially for a 16-issue machine.

To summarize, considering wakeup latency, power efficiency, area requirement, and scalability as a whole, the STP-WL design is the most attractive one. For the best speed performance, the STP scheme clearly is the winner for all the configurations.

E. Wakeup Distance and Performance

To reduce the area of the wakeup logic used for a large issue window, the STP-WL scheme takes the advantage of wakeup locality by limiting the wakeup range. This, however, may stall the insertion of the instructions that have wakeup distances out of the limited wakeup range. A backup unit that supports the wakeup operations with full wakeup distance is used to reduce the impact on IPC. We experiment on a 16-issue 128-entry processor to determine the number of entries required for this backup unit.

Fig. 16 shows the performances, which is normalized to the IPC of the STP-only scheme, for the STP-WL design with three different configurations of the backup unit. The IPC loss due to range-limited wakeup without a backup unit (STP-WL + zero-entry BU) is 11% on the average. This IPC drop comes from the stalling of instruction insertion into the issue window as the wakeup distance exceeds the limit. In contrast, the average IPC degradation is measured to be only 0.2% for the STP-WL design with an 8-entry backup unit. In this configuration, the worst IPC

degradation is 1.7% for the bzip2 program. When a 16-entry backup unit is used, there is almost no IPC degradation for all the benchmark programs.

The IPCs of the STP-WL schemes for some benchmark programs are slightly higher than the STP-only scheme. This is because some instructions on the miss-predicted paths may be blocked in the STP-WL design and fewer misprediction recoveries are performed than that in the STP scheme.

In summary, when a 16-entry backup unit is used, area reduction for the wakeup logic is achieved with no IPC loss.

VI. Related Works

Many previous researches have attempted to reduce the latency or energy consumption of the dynamic scheduler.

In addition to the gate-off technique mentioned in Section III, Folegnani and González presented another technique that dynamically manages the size of the issue window to reduce the power consumption [15]. However, this dynamically resizing technique needs an extra dynamic manger and results in a little IPC loss.

Hrishikesh *et al.* proposed a pipelined wakeup technique to improve the wakeup delay [22]. This design divides the issue window into multiple segments and wakes up the instructions in the segments in multiple sequential cycles. In this scheme, all the segments still need to be searched for the wakeup operation; besides the dependent instructions can be issued back to back only if they are in the first segment.

Brown *et al.* proposed a technique that removes the select logic from the critical path of the scheduler by speculatively selecting instruction for execution as soon as this instruction becomes ready [23]. This scheduler employs the wire-or style wakeup logic that is similar to the bit-map RAM wakeup design. This wakeup logic is efficient in terms of power consumption and wakeup delay. However, the area cost of this design is considerable for a large-window scheduler.

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 [20]. Ernst *et al.* also proposed a Cyclone scheduler that predicts the operand arrival time and schedules instructions in a countdown cyclic queue. This design boosts the clock frequency and, at the same time, reduces the power consumption and area cost of the scheduler [24]. However, the price for these two schedulers is a small IPC degradation.

Kim and Lipasti proposed a sequential wakeup mechanism to reduce the complexity of the scheduler [25]. In this design, the last-arrival operand is placed into the fast wakeup entry and two (left and right) source operands of the instructions are awakened in two sequential steps. Although the sequential wakeup logic enables a higher clock frequency by reducing the load capacitance for the tag driving, the two-cycle wakeup operation induces some IPC degradation.

Ponomarev *et al.* used three techniques to reduce the power dissipation of the issue window [26]. First, an efficient comparator is proposed to reduce the power dissipation due to tag match. Then, the 0-B encoding is employed to reduce the activities of the data path and bitlines. Third, the bitline segmentation is applied to the issue window in order to achieve a low-power issue window. But then, these techniques result in the cost of area and latency. The extra delays associate with the proposed comparators and 0-B encoding may slow down the clock cycle time of the processor.

Huang *et al.* proposed an index-based scheduler, which employs a producer instruction pointer and a consumer instruction pointer to index the instructions that should be awakened, to improve the energy efficiency of the scheduler [27]. This scheduler must work together with a conventional CAM structure. Canal and Gonzalez proposed two schemes to reduce the complexity of issue window [28]. One is the N-use scheme that replaces the associative search by the RAM-based index. The other scheme limits the associative search to a few entries of the issue window. The recovery from a branch miss prediction for these schemes becomes more complex. Henry *et al.* presented a cyclic segmented prefix (CSP) circuit to improve the performance of wakeup logic [29]. Other works [30]–[32] reduced the complexity of scheduler by scheduling dependent instructions into a data-flow based issue window.

VII. Conclusion

This paper presented two effective designs that improve wakeup delay, power requirement, and area cost of the wakeup logic without compromising the IPC. First, the proposed selective-match design removes needless tag matches by predecoding the source tags and activating only the selected match circuits during the wakeup operation. This design also eliminates the read of the destination tags from the critical wakeup path by matching the grant lines directly with the source tags. These optimizations significantly speedup 61%-84% of the wakeup operation and save 71%–83% power consumption compared to the conventional CAM scheme. Besides, by separating the match circuits from the tag RAM cells, this design uses only 25%–44% of the area required by the conventional RAM-based wakeup logic. Next, the STP-WL design explores the wakeup locality, that most wakeup distances between two dependent instructions are short in nature, to further minimize the area usage of the wakeup logic. Without IPC degradation, this design improves 78%–83% of the area cost in STP design for the 512-entry processors by limiting the wakeup range.

The experimental results also reveal that the major limiting factor for the scalability of the CAM scheme is the complexity of the CAM structure that induces large power consumption and considerable wakeup delay. As for the bit-map RAM scheme, the limiting factor is its large area requirement. In contrast to these two wakeup designs, the STP-WL design is attractive to use in terms of latency, power, and area cost for not only contemporary schedulers but also future sophisticated schedulers.

Acknowledgment

The authors would like to thank S. H. Gunther for his help with the power breakdown of the Pentium-4 processor. They would also like to thank all the reviewers and their colleagues for their insights and suggestions for strengthening this paper.

References

[1] S. Palacharla, N. P. Jouppi, and J. E. Smith, "Quantifying the complexity of superscalar processors," Univ. Wisconsin-Madison, Madison, Tech. Rep. CS-1328, 1997.

[2] S. H. Gunther, F. Binns, D. M. Carmean, and J. C. Hall, "Managing the impact of increasing microprocessor power consumption," *Intel Technol. J.*, vol. 5, no. 1, pp. 15–27, Feb. 2001.
[3] K. Wilcox and S. Manne, "Alpha processors: A history of power issues and a look to the future," in *Proc. Cool Chips Tutorial, 32nd Ann. Int. Symp. Microarch.*, 1999, pp. 16–37.
[4] A. Kumar, "The HP PA8000 RISC CPU," *IEEE Micro*, vol. 17, no. 2, pp. 27–32, Apr. 1997.
[5] G. Hinton, "The microarchitecture of the Pentium 4 processor," *Intel Technol. J.*, vol. 5, no. 1, pp. 36–44, Feb. 2001.
[6] K. C. Yeager, "The MIPS R10000 superscalar microprocessor," *IEEE Micro*, vol. 16, no. 2, pp. 28–40, Apr. 1996.
[7] R. E. Kessler, "The Alpha 21264 microprocessor," *IEEE Micro*, vol. 19, no. 2, pp. 24–36, Apr. 1999.
[8] L. Gwennap, "Intel's P6 uses decoupled superscalar design," *Microprocessor Report*, vol. 9, no. 2, pp. 1–7, Feb. 1995.
[9] S. P. Song, M. Denman, and J. Chang, "The PowerPC 604 RISC microprocessor," *IEEE Micro*, vol. 14, no. 5, pp. 8–17, Oct. 1994.
[10] L. Gwennap, "HAL reveals multichip SPARC processor," *Microprocessor Rep.*, vol. 9, no. 3, pp. 1–7, Mar. 1995.
[11] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 2nd ed. San Francisco, CA: Morgan Kaufmann, 1996.
[12] M. Butler and Y. N. Patt, "An investigation of the performance of various dynamic scheduling techniques," in *Proc. MICRO*, 1992, pp. 1–9.
[13] S. T. Srinivasan and A. R. Lebeck, "Load latency tolerance in dynamically scheduled processors," in *Proc. MICRO*, 1998, pp. 148–159.
[14] M. Goshima, K. Nishino, Y. Nakashima, S. Mori, T. Kitamura, and S. Tomita, "A High-speed dynamic instruction scheduling scheme for superscalar processors," in *Proc. MICRO*, 2001, pp. 225–236.
[15] D. Folegnani and A. González, "Energy-effective issue logic," in *Proc. Int. Symp. Comput. Arch.*, 2001, pp. 230–239.
[16] R. Ho, K. W. Mai, and M. A. Horowitz, "The future of wires," *Proc. IEEE*, vol. 89, no. 4, pp. 490–504, Apr. 2001.
[17] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proc. Int. Symp. Comput. Arch.*, 2000, pp. 83–94.
[18] D. Burger and T. M. Austin, "The SimpleScalar tool set," Univ. Wisconsin-Madison, Madison, Tech. Rep. CS-1342, 1997, version 2.0.
[19] C. Lee, M. Potkonjak, and W. Mangione-Smith, "MediaBench: A tool for evaluating multimedia and communications systems," in *Proc. MICRO*, 1997, pp. 330–335.
[20] D. Ernst and T. M. Austin, "Efficient dynamic scheduling through tag elimination," in *Proc. Int. Symp. Comput. Arch.*, 2002, pp. 37–46.
[21] G. Reinman and N. P. Jouppi, "CACTI 2.0: An integrated cache timing and power model," COMPAQ Western Research Lab, Palo Alto, CA, 2000.
[22] M. S. Hrishikesh, N. P. Jouppi, and K. I. Farkas, "The optimal useful logic depth per pipeline stages is 6–8 FO4," in *Proc. Int. Symp. Comput. Arch.*, 2002, pp. 14–24.
[23] M. D. Brown, J. Stark, and Y. N. Patt, "Select-free instruction scheduling logic," in *Proc. MICRO*, 2001, pp. 204–213.
[24] D. Ernst, A. Hamel, and T. Austin, "Cyclone: A broadcast-free dynamic instruction scheduler with selective replay," in *Proc. Int. Symp. Comput. Arch.*, 2003, pp. 253–262.
[25] I. Kim and M. H. Lipasti, "Half-price architecture," in *Proc. Int. Symp. Comput. Arch.*, 2003, pp. 28–38.
[26] D. V. Ponomarev, "Energy-efficient issue queue design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 5, pp. 789–800, Oct. 2003.
[27] M. Huang, J. Renau, and J. Torrellas, "Energy-efficient hybrid wakeup logic," in *Proc. Int. Symp. Low-Power Elect. Des.*, 2002, pp. 196–201.
[28] R. Canal and A. Gonzalez, "Reducing the complexity of the issue logic," in *Proc. Int. Conf. Supercomput.*, 2001, pp. 312–320.
[29] D. S. Henry, B. C. Kuszmaul, G. H. Loh, and R. Sami, "Circuits for wide-window superscalar processors," in *Proc. Int. Symp. Comput. Arch.*, 2000, pp. 236–247.
[30] S. Palacharla, N. P. Jouppi, and J. E. Smith, "Complexity-effective superscalar processors," in *Proc. Int. Symp. Comput. Arch.*, 1997, pp. 206–218.
[31] P. Michaud and A. Seznec, "Data-flow prescheduling for large instruction windows in out-of-order processors," in *Proc. High-Performance Comput. Arch.*, 2001, pp. 27–36.
[32] S. E. Raasch, N. L. Binkert, and S. K. Reinhardt, "A scalable instruction queue design using dependence chains," in *Proc. Int. Symp. Comput. Arch.*, 2002, pp. 318–329.

Kuo-Su Hsiao received the B.S. degree in computer and communication engineering from the National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan, in 1999, and the M.S. degree in electrical engineering from the National Cheng Kung University, Tainan, Taiwan, R.O.C., in 2001. He is currently pursuing the Ph.D. degree in electrical engineering at the same university.

His research interests include computer architecture, low-power processor, and VLSI design.

Chung-Ho Chen (M'06) received the M.S. degree in electrical engineering from the University of Missouri-Rolla, Rolla, in 1989, and the Ph.D. degree in electrical engineering from the University of Washington, Seattle, in 1993.

Since 1993, he was a Faculty Member of the Department of Electronic Engineering, National Yunlin University of Science and Technology, Yunlin, Taiwan. In 1999, he joined the Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, R.O.C., where he is currently an Associate Professor. His research interests include advanced computer architecture, video technology, and network storages. He holds a U.S. patent on a multicomputer cluster-based processing system and a R.O.C. patent on a multiple-protocol storage structure.

Dr. Chen was the Technical Program Chair of the 2002 VLSI Design/CAD Symposium held in Taiwan.