PAPER

Delay Evaluation of Issue Queue in Superscalar Processors with Banking Tag RAM and Correct Critical Path Identification

Kyohei YAMAGUCHI[†], **Yuya KORA**[††*], *Nonmembers*, *and* **Hideki ANDO**[†a)], *Member*

SUMMARY This paper evaluates the delay of the issue queue in a superscalar processor to aid microarchitectural design, where quick quantification of the complexity of the issue queue is needed to consider the tradeoff between clock cycle time and instructions per cycle. Our study covers two aspects. First, we introduce banking tag RAM, which comprises the issue queue, to reduce the delay. Unlike normal RAM, this is not straightforward, because of the uniqueness of the issue queue organization. Second, we explore and identify the correct critical path in the issue queue. In a previous study, the critical path of each component in the issue queue was summed to obtain the issue queue delay, but this does not give the correct delay of the issue queue, because the critical paths of the components are not connected logically. In the evaluation assuming 32-nm LSI technology, we obtained the delays of issue queues with eight to 128 entries. The process of banking tag RAM and identifying the correct critical path reduces the delay by up to 20% and 23% for 4- and 8-issue widths, respectively, compared with not banking tag RAM and simply summing the critical path delay of each component.

key words: *microprocessor, superscalar processor, issue queue, delay, complexity*

1. Introduction

The tradeoff between complexity and attained instructions per cycle (IPC) must be considered in the microarchitectural design phase of microprocessors. To quantify complexity precisely, the delay of structures must be known. Although a quick estimation is required in the microarchitectural design phase, this is a time-consuming task in reality, since a circuit-level simulation that considers the layout must be carried out. It would thus be useful if evaluated delays for structures were readily available.

The issue queue is one of the superscalar processor structures whose delay is difficult to estimate. To achieve increased IPC, the issue queue needs to be enlarged. In fact, the recently released Intel Sandy Bridge has a 54-entry issue queue, whereas the previous generation, the Nehalem, only has 36 entries [1]. However, as the issue queue comprises one of the critical paths in a processor, its enlargement can degrade the clock cycle time. In this study, we evaluate the delay of issue queues of varying sizes and issue widths through circuit-level simulations. Despite previous evaluations of the delay of parts of or the whole issue queue [2], [3], the contributions of this paper, as given below, are quite different.

- To reduce the delay, a banked tag RAM, which is one of the components comprising the issue queue, is designed. Although it has intuitively been found that banking is effective in large issue queues in current and future technology generations where wire delay is serious, there have been no studies on banking tag RAM. In normal RAM, multiple banks are accessed in parallel using part of the address, and then one of the outputs from the banks is selected by the other part of the address. However, this method is unusable in tag RAM, since this is not accessed via an address but instead through direct activation of a wordline connected to the selection logic, another component in the issue queue. Thus, a novel banked tag RAM, accommodating the selection logic, is designed.
- In a previous study [3], the issue queue delay was obtained by summing the delay of the critical path of each component comprising the issue queue. Unfortunately, the delay obtained in this way is overestimated, especially in a large issue queue, because the critical paths of these components are not connected logically. In this paper, we identify the correct critical path of the issue queue through exhaustive simulation and give its delay.

This paper is an extension of our previous conference paper [4]–[6] providing more detailed descriptions and additional evaluation results.

The remainder of this paper is organized as follows. Section 2 presents related work. Section 3 describes the processor organization we assumed to narrow down the organization of the issue queue in this study. Section 4 explains the circuit organization of the issue queue, while Sect. 5 discusses its critical paths. Evaluation results are presented in Sect. 7, and finally our conclusions are stated in Sect. 8.

2. Related Work

Although many studies have focused on the organization of the issue queue (e.g., [7], [8]), there are few that cover circuit design. In this section, we discuss relevant studies that address circuit designs and evaluate the delay. Note that the critical path of an issue queue traverses three components,

Manuscript received February 21, 2012.
Manuscript revised April 20, 2012.

[†]The authors are with the Department of Electrical Engineering and Computer Science, Nagoya University, Nagoya-shi, 464–8603 Japan.
[††]The author is with the Department of Computational Science and Engineering, Nagoya University, Nagoya-shi, 464–8603 Japan.
[*]Presently, with Rohm Co., Ltd.
a) E-mail: ando@nuee.nagoya-u.ac.jp

DOI: 10.1587/transinf.E95.D.2235

namely, the wakeup logic, selection logic, and tag RAM, as described in Sect. 4.

Having evaluated the delay of critical structures in a processor using 800 nm to 180 nm technology [2], [9], Palacharla et al. suggested that the issue queue, which is a target for enlargement, is one of the key structures affecting the clock cycle time. By assuming a CAM organization of the wakeup logic, the authors evaluated the delay, pointing out that the tag drive delay is the most serious in a large issue queue under deep submicron technology, because the wire delay is not scaled.

Palacharla et al. also evaluated selection logic implemented using arbiter circuits. Generally, selection takes a long time if many requests are being arbitrated at the same time. Therefore, a logic circuit was implemented by serially connecting small arbiters that grant the request with the highest priority out of four requests. Since the resulting selection logic circuit grants only one of the requests, it is necessary to connect the logic serially to extend it so as to be able to grant multiple requests. Unfortunately, this extension has the drawback of linearly increasing the delay according to the number of requests granted, and thus it is not suitable for the unified issue queue used in modern processors (e.g., [1], [10]).

Although Palacharla et al. evaluated the delays of the wakeup and selection logic, they did not evaluate the tag RAM delay, and thus the total delay of the issue queue was not obtained.

Goshima et al. proposed a scheme that organizes the wakeup logic using RAM instead of CAM [3]. In the RAM organization, comparators are not necessary, unlike in the CAM organization. In addition, since RAM is more compact than CAM, the length of the wires stretching across the logic is reduced. Although this in turn reduces the delay, the RAM organization has the drawback that compaction of the issue queue is difficult to implement, because dependencies are represented by the entry positions in the issue queue. Goshima et al. compared the delay of the RAM organization with that of the CAM organization. However, it seems that they calculated the delay of the issue queue of the CAM organization by simply summing the delay of the critical path of each component. Unfortunately, using this simple calculation to obtain the issue queue delay is incorrect, because the critical paths of all the components are not connected logically.

Regarding tag RAM, Goshima et al. assumed a monolithic RAM in their evaluation, but did not consider banking it[†]. For a large queue, however, the bitline delay is significant, and thus banking is effective.

Regarding the selection logic, Goshima proposed a circuit using prefix-sum logic [11]. This logic uses adders to count the number of requests with a higher priority than a specific request. If the calculated number for a particular request is smaller than the issue width, the request is granted. Unlike Palacharla's arbiter logic circuit, this logic circuit does not significantly increase the delay depending on the number of requests granted. The drawback, however, is that the delay of the adders is large. Goshima attempted to reduce this delay by improving the adders through elaborative encoding of their input and output.

3. Processor Organization

In this section, we explain the processor organization assumed to narrow down the organization of the issue queue discussed in this paper. Figure 1 shows a block diagram of the organization of the processor. Multiple instructions are fetched from the L1 instruction cache, and are decoded. The logical registers of the decoded instructions are then renamed to appropriate physical registers, with the physical register numbers used as tags identifying dependences. The renamed instructions are inserted into the issue queue, where they await the resolution of their dependences. Instructions with all dependences resolved are issued and executed in a function unit with their source operands fetched from either the register file or the bypass logic. Completing instructions broadcast their destination tags to the issue queue making it known that their results are ready to use. The results are written into the register file. The L1 data cache provides memory accesses of load and store instructions.

In general, there are two main processor organizations, with the first of these shown in Fig. 1. In this type of organization, the register file contains both committed and temporary values for completed instructions that have not yet committed, and provides source operand values to the issued instructions. It should be noted that the issue queue does not hold source operand values. Processors with this organization include the MIPS R10000 [12], Digital Equipment Alpha 21264 [13], Intel Sandy Bridge [1], and AMD Bulldozer [10].

The other major organization is the one introduced in the Intel P6 family of processors [14]. Unlike the former organization, temporary values for completed instructions that have not yet committed are held in a structure called the reorder buffer, which controls the commit of values, while the register file holds only committed values. Instructions inserting into the issue queue read source operand values from the register file or the reorder buffer, and write them to the issue queue. This type of issue queue is called a reservation station. The point to be noted is that, unlike the issue queue in the former organization, the reservation station holds source operand values. Thus, for instructions waiting in the reservation station to obtain the results of completing instructions, execution results, together with the destination tags, are broadcast to the reservation station and the entry with a tag match acquires a result. Owing to holding source operand values and the necessity for broadcasting the result,

[†]They evaluated issue queues with up to 32 entries. In such small queues, banking is not very effective, and the issue queue delay obtained by simple summation is not much different to the delay of the correct critical path, based on our evaluation described in Sect. 7.4. Therefore, the evaluation results obtained under their assumptions and experimental setup are correct.

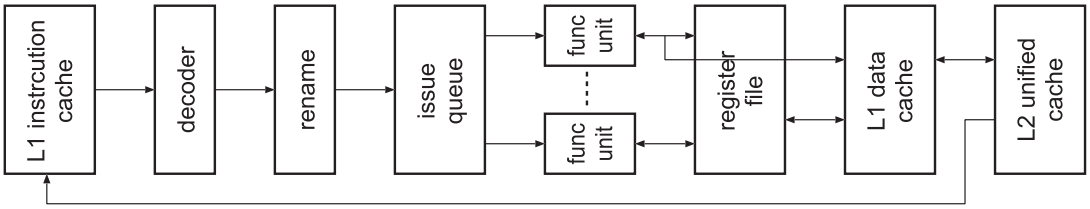


Fig. 1 Processor organization.

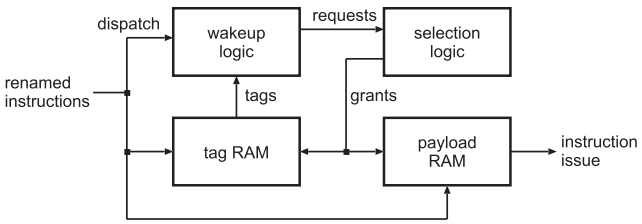


Fig. 2 Organization of issue queue.

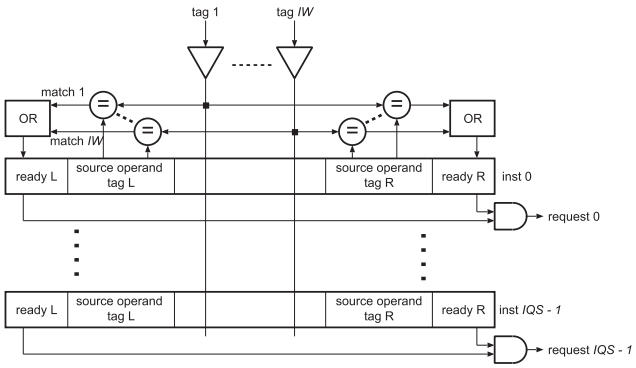


Fig. 3 Wakeup logic.

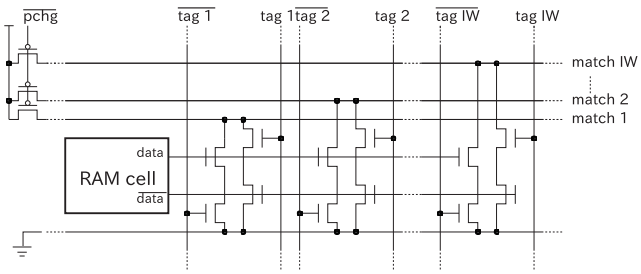


Fig. 4 CAM cell circuit for tag comparison.

the reservation station is larger and less power-efficient than the issue queue. Thus, many recent processors have adopted the former organization, including the Intel Sandy Bridge and AMD Bulldozer.

4. Issue Queue Organization and Circuit

As described in Sect. 3, the issue queue holds renamed instructions and determines instructions to be issued. It comprises the wakeup logic, selection logic, tag RAM, and payload RAM, as illustrated in Fig. 2. The wakeup logic is a one-dimensional array, with each entry holding the tags of two source operands attached at renaming, and ready flags indicating the data dependence state (resolved or not) for the corresponding instruction. If both data dependences are resolved, an issue request (simply denoted by request in the figure) is sent to the selection logic, which grants some requests by considering resource constraints. The grant signals are sent to the payload RAM, which outputs information regarding the issued instructions. The signals are also sent to tag RAM, and the destination tags are read. These tags are broadcast to the wakeup logic to update the ready flags.

The critical path of the issue queue begins at the wakeup logic, goes via the selection logic to the tag RAM, and finally returns to the wakeup logic. This paper evaluates the delay of this critical path.

As stated in Sect. 2, various circuits have been proposed for the wakeup and selection logic. Here, we assume a circuit comprising CAM for the wakeup logic, since the RAM organization has the serious drawback that it is difficult to compact the issue queue (A simple way to avoid this drawback is to implement the issue queue using a circular buffer, although because of wrap-around this gives incorrect priority information of the issue requests to the selection logic. A more elaborate solution is to use the age matrix [15] for the selection logic; it is, however, difficult to extend this circuit to grant multiple requests). For the selection logic, we assume a circuit composed of prefix-sum logic, because contrary to what happened with arbiter logic, the delay in this circuit does not increase much as the number of granted requests increases.

4.1 Wakeup Logic

Figure 3 illustrates the wakeup logic comprising CAM. *IW* destination tags (depending on the context, we refer to a destination tag simply as a tag) read from the tag RAM are broadcast to all entries of *IQS* in the wakeup logic, driven by tag drivers. Here, *IW* and *IQS* denote the issue width and issue queue size, respectively. Each entry has two source operand tags, which are compared with the destination tags broadcast. If there is a match, the ready flag is set. If both ready flags are set, an issue request is output.

Figure 4 shows the circuit of a CAM cell that performs tag comparison. A single entry of the wakeup logic comprises a row of CAM cells, depending on the number of tag bits. The SRAM cell on the left of the figure holds a single bit of the source operand tag. The horizontal lines, called match lines, indicate that the tags are matched. The two

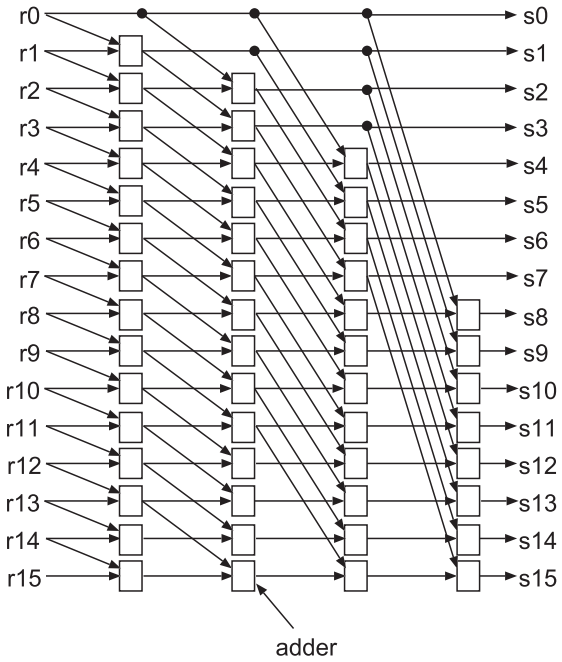


Fig. 5 prefix-sum circuit with $N = 16$.

stacked transistors are pull-down transistors that discharge the match lines, depending on the result of the tag comparison.

The circuit operates as follows. First, the match lines are pre-charged, and then the destination tags are driven. If any bit in the pair of source and destination tags is unmatched, the match line corresponding to the pair is pulled down by the stacked transistors. If all bits are matched, the match line remains high.

4.2 Selection Logic

We assume that the selection logic is implemented by prefix-sum logic, as described at the beginning of Sect. 4. In general, prefix-sum logic has N inputs and N outputs, where the i-th $(0 \leq i \leq N-1)$ output is the sum of the values of the 0-th to i-th input. Figure 5 shows a circuit diagram of the prefix-sum logic [11] with $N = 16$, as an example. Note that the number of adders on the critical path is $\log_2 N$.

When using this logic as the selection logic, we assume each input is the Boolean value of the issue request, and the logic adds the input values arithmetically. If the $(i-1)$-th output is less than the issue width and the i-th issue request is true, the request is granted. Figure 6 shows the grant signal output circuit, using the one-hot encoding adder described below. Note that signal granti_u is connected to the u-th $(0 \leq u \leq IW-1)$ wordline for the cell of the i-th entry in the tag RAM.

A new adder circuit was proposed by Goshima to lessen the delay [11]. In the selection logic, it is sufficient that the input and output of the adder represent only five values, that is, "0", "1", "2", "3", and "≥ 4" when $IW = 4$ for example. Goshima represented the input and output using four-bit one-hot encoding, as shown in Table 1, and organized the circuit as a domino circuit, as shown in Fig. 7. Here, a four-bit a is added to b, and the four-bit sum c is output. As intuitively found, this circuit is faster than a conventional adder, and thus we used it in our evaluation.

Figure 8 shows the top-level diagram of the selection logic.

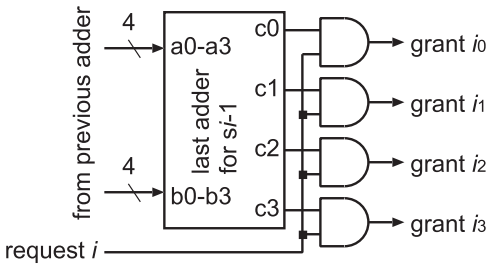


Fig. 6 Grant output circuit with $IW = 4$.

Table 1 One-hot encoding for adder input and output ($IW = 4$).

value	0	1	2	3	≥ 4
encoding	1000	0100	0010	0001	0000

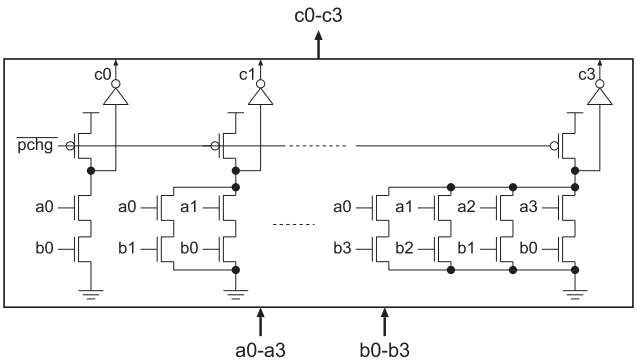


Fig. 7 Adder for selection logic ($IW = 4$).

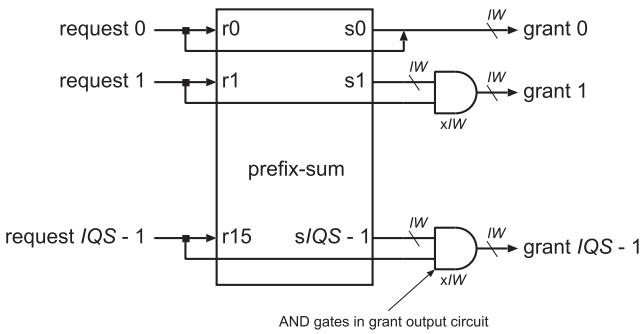


Fig. 8 Top-level diagram of selection logic.

4.3 Tag RAM

Tag RAM consists of SRAM without the address decoder. It has IW ports, with IW-bit grant lines per entry (see Fig. 6) from the selection logic directly connected to the IW wordlines per entry. In a monolithic organization, up to IW destination tags held in the cells connected to the asserted word-

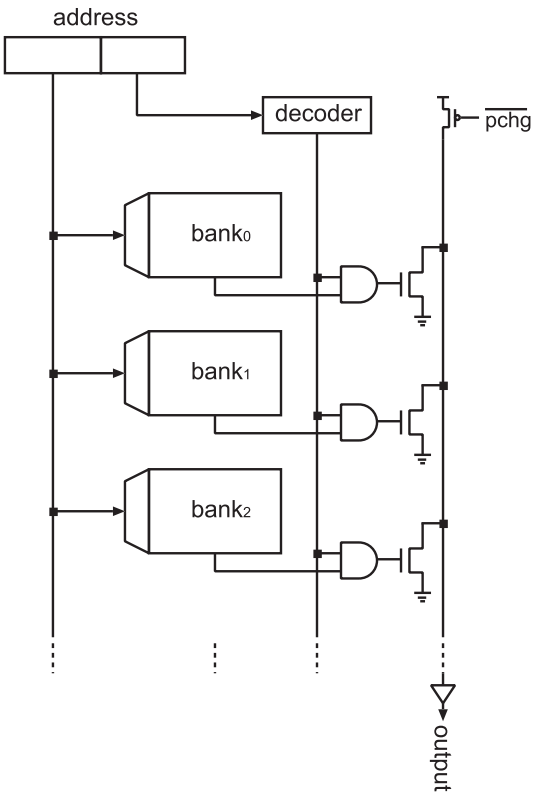


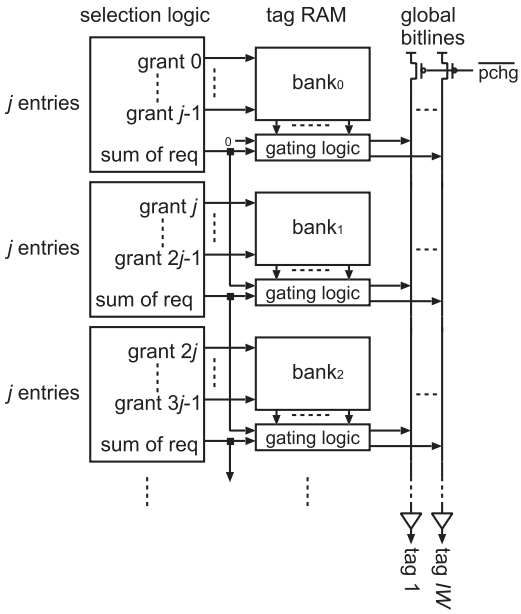
Fig. 9 Banked normal RAM.

lines are read to bitlines, and output after being amplified with senseamps.

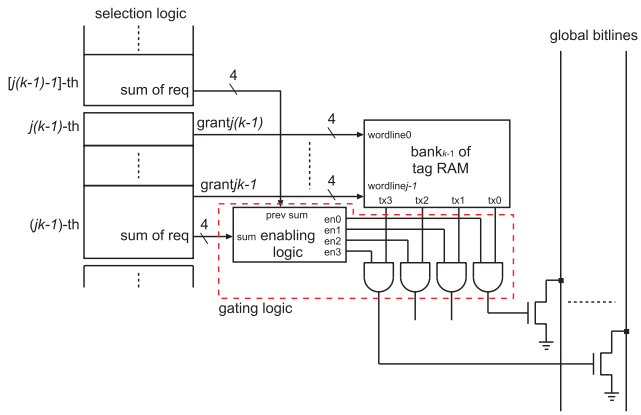
As in a normal RAM, banking can reduce access time. However, the organization of tag RAM is not straightforward, because it is not accessed via an address.

We first review banking normal RAM accessed by an address. Figure 9 illustrates the organization of banked normal RAM. The locations of the RAM are interleaved for the sake of explanation. Accessing this banked RAM is carried out as follows. All banks are accessed simultaneously using the upper portion of the address. Simultaneously, the lower portion of the address is decoded, and the decoded signals are used to select a target bank to be accessed. The output of the selected bank is allowed to drive the global bitline through the AND gate that enables the pull-down transistor.

In the case of tag RAM, selection of a bank is difficult because there is no address identifying the target bank to be accessed. Instead, we use the sums of the issue requests (the outputs of the prefix-sum). Figure 10 ((a) overview and (b) details of a bank) shows the organization of tag RAM with banks of j entries and $IW = 4$. We add the *gating logic*, which enables the outputs of a bank, tx_0–tx_3. The *enabling logic* in the gating logic identifies which port of the corresponding bank is valid by observing the total sum of issue requests from the first entry of the queue to the last entry of the corresponding bank and that of the previous bank. For example, if $\sum_{i=0}^{7} \text{request}i = 1$ and $\sum_{i=0}^{15} \text{request}i = 3$, with



(a) Overview



(b) Details of a bank ($IW = 4$)

Fig. 10 Organization of banked tag RAM.

$j = 8$, the logic identifies that the 1st and 2nd ports of the 0th–3rd ports of bank$_1$ are valid, and thus only tx_1 and tx_2 are allowed to be output to the global bitlines. Table 2 gives the complete truth table for the enabling logic. Note that c0–c3 are the outputs of the last adder associated with the sum output, which is one-hot encoded (see Table 1 and Fig. 7). The third row in the second section in the table depicts the case in the previous example.

Unlike banking in normal RAM, there are two critical paths in accessing a bank; one is the grant signal to the tag output via a banked RAM, while the other is the sum of requests to the tag output via the gating logic.

4.4 Layout

The assumed layout of the wakeup logic, selection logic, and tag RAM is shown in Fig. 11. This circuit layout was manually drawn, assuming MOSIS design rules [16]. Since the height of a single entry in each component is nearly

Table 2 Truth table of enabling logic ($IW = 4$).

prev sum c0–c3	sum c0–c3	enable en0–en3
1 0 0 0	1 0 0 0	0 0 0 0
1 0 0 0	0 1 0 0	1 0 0 0
1 0 0 0	0 0 1 0	1 1 0 0
1 0 0 0	0 0 0 1	1 1 1 0
1 0 0 0	0 0 0 0	1 1 1 1
0 1 0 0	0 1 0 0	0 0 0 0
0 1 0 0	0 0 1 0	0 1 0 0
0 1 0 0	0 0 0 1	0 1 1 0
0 1 0 0	0 0 0 0	0 1 1 1
0 0 1 0	0 0 1 0	0 0 0 0
0 0 1 0	0 0 0 1	0 0 1 0
0 0 1 0	0 0 0 0	0 0 1 1
0 0 0 1	0 0 0 1	0 0 0 0
0 0 0 1	0 0 0 0	0 0 0 1
0 0 0 0	0 0 0 0	0 0 0 0

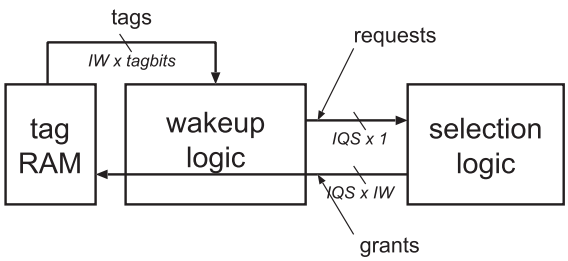


Fig. 11 Layout of wakeup logic, selection logic, and tag RAM.

equal, we determined the height of a single entry of the issue queue from the largest value. Thus, the issue request and grant signal lines are laid out horizontally without jogs.

5. Correct Critical Path

The path obtained by simply connecting the critical path of each component is not logically connected. For example, consider the configuration where the output of the tag RAM and the tag driver in the wakeup logic are connected at the last entry, as shown in Fig. 12. Suppose that a signal starts from mark (1) and goes via marks (2) to (5) before returning to (1), traversing the critical paths of the wakeup logic and selection logic. In this case, the signal does not go through the critical path of the tag RAM (marks (6) to (5), bitline traversing).

Possibly correct critical paths are shown in Fig. 13, with different configurations, where the output of the tag RAM and the tag driver in the wakeup logic are connected (a) at the last entry and (b) at the first entry. Two different configurations arise from the up-down asymmetry of the selection logic. In configuration A, the path turns at the n-th entry in the selection logic, whereas in configuration B, it turns at the m-th entry in the wakeup logic ($0 \leq n, m \leq IQS - 1$). Based on an evaluation, we first find the longest path (i.e., find n and m) in configurations A and B, respectively. Then, we determine that the configuration with the shorter path is the better configuration, and the associated path is the correct critical path.

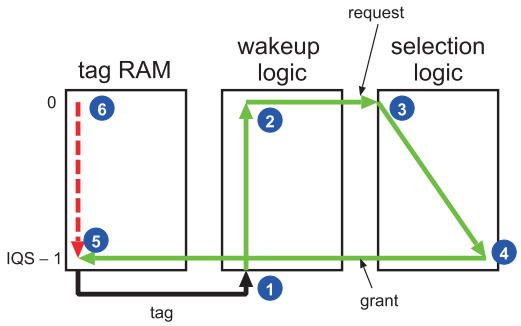
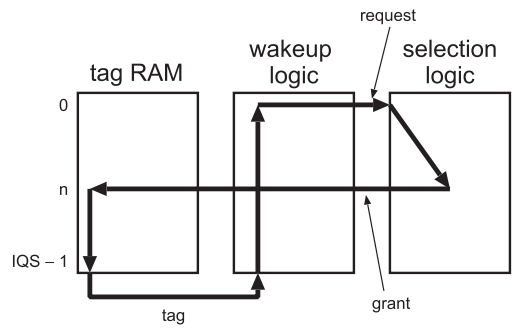
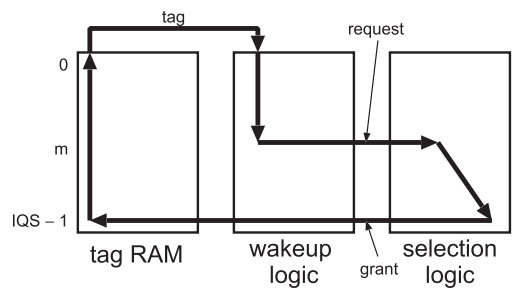


Fig. 12 Path simply connecting the critical path of each component.



(a) Configuration A



(b) Configuration B

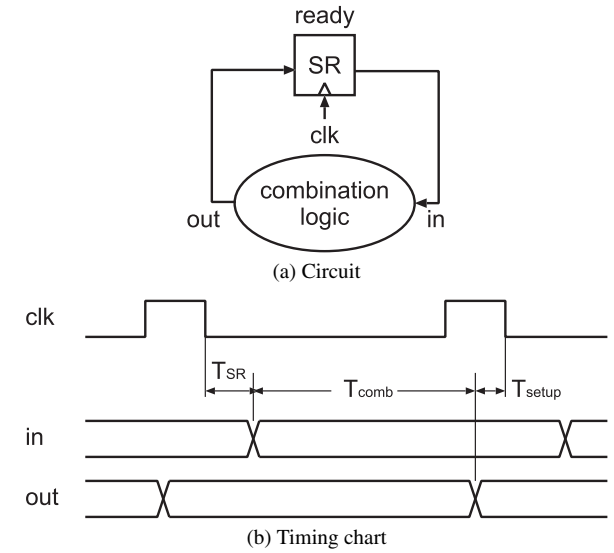
Fig. 13 Possibly correct critical paths.

6. Timing Assumptions

Before presenting the evaluation results, we describe the assumptions with respect to timing.

6.1 Basic Assumption

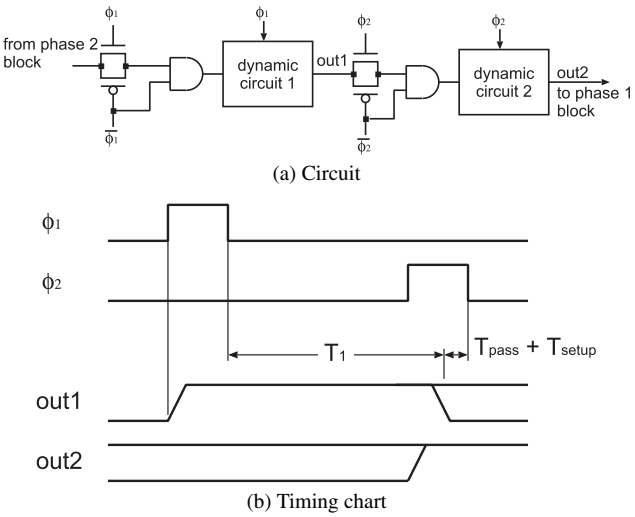
Figure 14 (a) shows a simplified block diagram of the issue queue to explain the basic assumption of timing. The signal starts at the synchronous SR-latches for the ready flags, proceeds through the combination logic (the wakeup and selection logic, and tag RAM), and returns to the ready flag latches. Figure 14 (b) shows the timing chart for this block diagram, where T_{SR} and T_{comb} are the delays of the SR-latch and combination logic, respectively, and T_{setup} is the setup time for the SR-latch. The delay of the issue queue is the sum of these three values. However, for simplicity, we assume that the setup time is 0 in our evaluation. Thus, the issue queue delay is given by $T_{SR} + T_{comb}$.



(a) Circuit

(b) Timing chart

Fig. 14 Simplified block diagram of the issue queue.



(a) Circuit

(b) Timing chart

Fig. 15 Two-phase dynamic circuit.

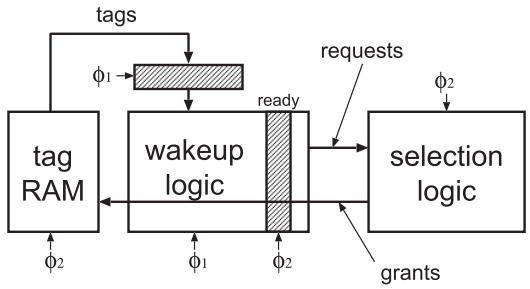
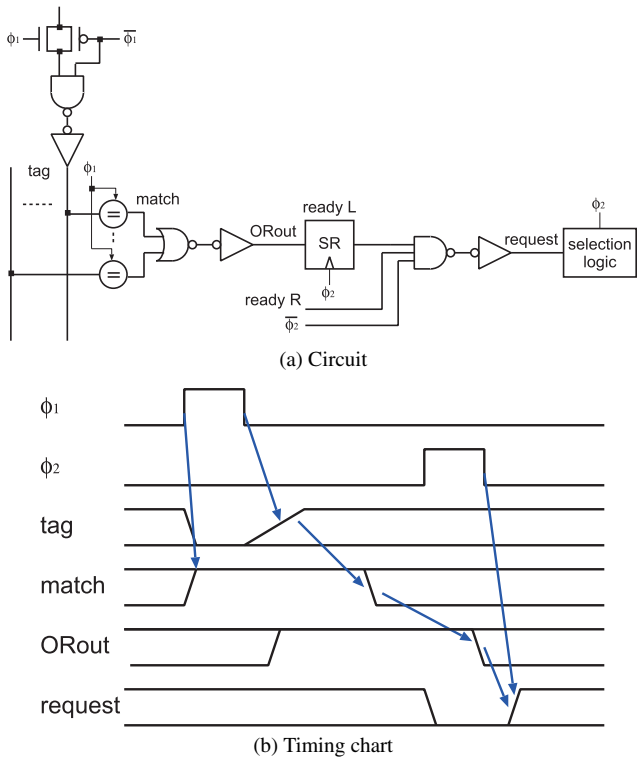


Fig. 16 Insertion of clocks and latches.



(a) Circuit

(b) Timing chart

Fig. 17 Circuit from tag drive to selection logic.

6.2 Assumptions in Dynamic Circuits

As explained in Sect. 4, the circuits for the wakeup and selection logic, and the tag RAM are basically composed of dynamic circuits. In this section, we describe the timing assumptions in dynamic circuits.

We introduce the two-phase clocking scheme shown in Fig. 15 (a). This figure shows a general organization in this scheme; a specific organization in the issue queue is described in Sect. 6.3. In the figure, the box labeled "dynamic circuit" represents a single-stage dynamic circuit or cascaded domino circuits.

Dynamic circuits 1 and 2 are precharged by a clock with different phases, ϕ_1 and ϕ_2, respectively, while the pass-transistors dynamically latch the value evaluated when the clock falls. Two-phase clocking allows the precharge time of a circuit to be hidden by the evaluation time of another circuit driven by the other phase clock.

Figure 15 (b) shows the timing chart, where T_1 is the delay of dynamic circuit 1 including the AND gate with input $\overline{\phi_1}$, T_{pass} is the delay of the pass-transistors after circuit 1, and T_{setup} is the setup time against the falling edge of ϕ_2. The substantive delay of dynamic circuit 1 is the sum of these three values. However, for simplicity, we assume that the setup time is 0 in our evaluation, as in Sect. 6.1, and thus the delay of dynamic circuit 1 is $T_1 + T_{pass}$. We also assume that the clock edge falls at the exact time of the arrival of a signal on a critical path, and rises before the arrival of a signal. Under these assumptions, the delay we need to evaluate is where a signal goes through the circuit without taking clock timing into account. Note again that the precharge time is hidden by the evaluation time of the dynamic circuit in the other phase, and thus we do not need to consider this.

The delay under these assumptions gives the lower

bound of the delay. If a reader wishes to consider the timing margin, add it to the delay given in this paper.

6.3 Inserting Two-Phase Clocks and Latches into the Issue Queue

Figure 16 illustrates our assumptions underlying the insertion of two-phase clocks and latches into the issue queue circuit, where the shaded boxes represent the latches. The latches clocked by ϕ_1 are explicitly inserted, whereas the latches clocked by ϕ_2 are the already-existing ready flag latches.

Figure 17 extracts the circuit from the tag drive to the selection logic (a) and shows its timing chart (b). While clock ϕ_1 is HIGH, the comparators are precharged. When clock ϕ_1 falls, the tags are driven and the match line is determined. After proceeding through the NOR and inverter gates, the signal is stored into the ready latch while clock ϕ_2 is HIGH. During this period, the selection logic is precharged. When clock ϕ_2 falls, the request signal is sent to the selection logic, and the logic operates.

7. Evaluation

By means of a SPICE simulation, we evaluated the issue queue delay, varying the size of the issue queue with $IW = 4$ and 8, respectively. We fixed the tag width at 8. The delay variation with respect to tag width is small, and thus we omit these evaluation results.

Assuming 32-nm technology, we used the predictive transistor model [17], [18] developed by the Nanoscale Integration and Modeling group of Arizona State University as the transistor model for SPICE. We used the resistance and capacitance per unit length of the wire predicted by the International Technology Roadmap for Semiconductors (ITRS) [19]. Repeaters were inserted in long wires to reduce the delay, with the optimum interval of insertion selected by experimentation.

7.1 Delay of Wakeup Logic

Figure 18 shows the evaluated delay of the wakeup logic for various issue queue sizes. Each bar is divided into five sections: delays of the tag drive, tag match (comparison of tags), OR (ORing all comparison results), ready (going through synchronous SR-latch for a ready flag), and AND (ANDing two matching ORs). As shown in the figure, with both $IW = 4$ and 8, the delays of the tag match and ready latch are equivalently the largest part in small queues, but the delay of the tag drive increases significantly as the issue queue size increases.

If IW increases from 4 to 8, the delay increases significantly. Since the CAM cell size is increased both horizontally and vertically, the length of the tag and match line wires increases. Thus the delays of the tag drive and tag match are increased. Also, since the fanin of the OR gate is increased, the delay of ORing all comparison results increases.

Figure 19 summarizes which parameter (IQS, IW, or BS (bank size)) has an impact on the delay of the associated paths. The notation for the attributes on each path is as follows.

$$attribute = param\langle type, direction\rangle$$
$$param = \{IQS \mid IW \mid BS\}$$
$$type = \{gate \mid wire\}$$
$$direction = \{+ \mid -\}$$

Attribute denotes which *type* of delay of the associated path is positively or negatively (*direction*) impacted, if *param* is increased.

7.2 Delay of Selection Logic

Figure 20 shows the evaluated delay of the selection logic for various issue queue sizes. Each bar is divided into three sections: the sum of the adders' delay, the sum of the wires' delay, and the AND delay of the issue request and output of the prefix-sum logic. As shown in the figure, the gate delay of the adders is dominant with both $IW = 4$ and 8. The delay increases by $O(\log_2 IQS)$, because the number of adders on the critical path is $\log_2 IQS$, as discussed in Sect. 4.2.

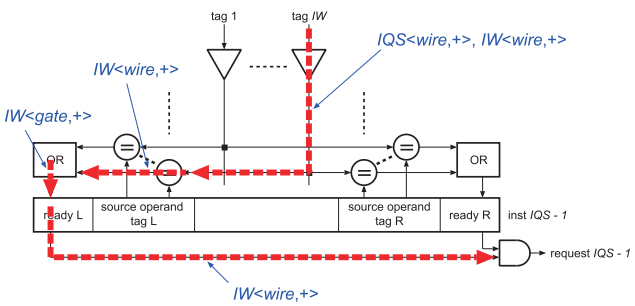
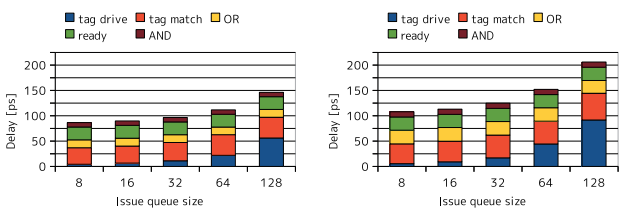
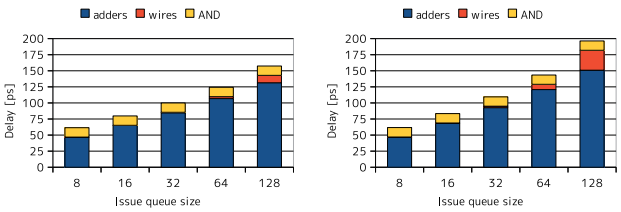


Fig. 19 Parameters that impact delays in the wakeup logic.



(a) $IW = 4$ (b) $IW = 8$

Fig. 18 Delay of wakeup logic.



(a) $IW = 4$ (b) $IW = 8$

Fig. 20 Delay of selection logic.

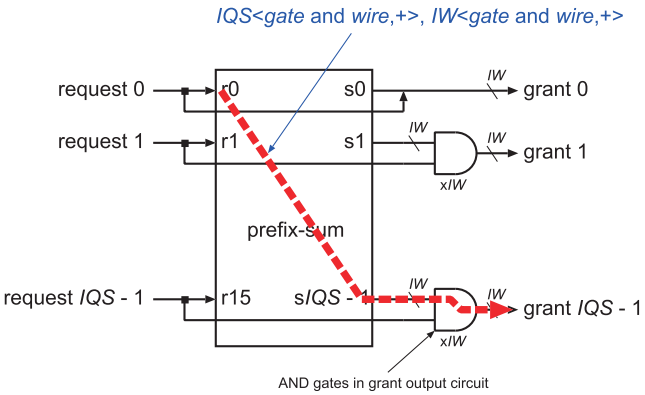
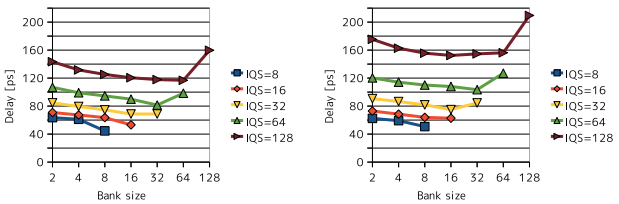


Fig. 21 Parameters that impact the delay in the selection logic.



(a) $IW = 4$ (b) $IW = 8$

Fig. 22 Tag RAM delay for various bank sizes.

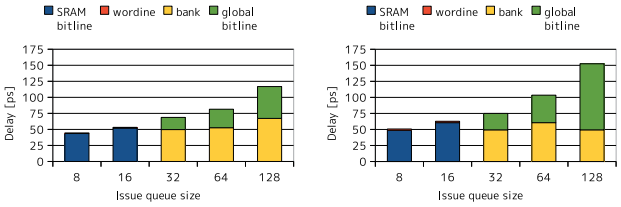
If IW increases from 4 to 8, the delay of adders increases, because the drain capacitance that is parasitized to the input node of the inverter outputting $c7$ (seventh-bit of the output of the adder) with $IW = 8$ is larger than that outputting $c3$ with $IW = 4$ (see Fig. 7). Note that eight and four stacked transistors are connected to the output inverter with $IW = 8$ and 4, respectively. The delay of the wires also increases because the height of the entry in the issue queue increases.

Figure 21 summarizes which parameter has an impact on the delay of the associated paths.

7.3 Delay of Tag RAM

Figure 22 shows the evaluated delay of the tag RAM for various issue queue and bank sizes. Note that the point where the bank size is equal to the issue queue size (IQS) represents the case in which banking is not carried out. With $IW = 4$ and 8, for small queues ($IQS \leq 16$), banking is ineffective, because the length of the bitline is not long, and the number of SRAM cells connected to the bitline is small. On the other hand, banking is effective in large queues for the opposite reason.

If IW increases from 4 to 8, the delay of tag RAM increases, because the entry size of the issue queue increases and thus the wire length increases. The optimal bank sizes are the same (16 and 32) for both IWs with $IQS = 32$ and 64. On the other hand, they are different (64 in $IW = 4$ and 16 in $IW = 8$) with $IQS = 128$. This is because the bitline is longer with $IW = 8$, and the tag RAM is more aggressively divided into banks to reduce the delay.



(a) $IW = 4$ (b) $IW = 8$

Fig. 23 Delay of tag RAM with optimal bank configuration.

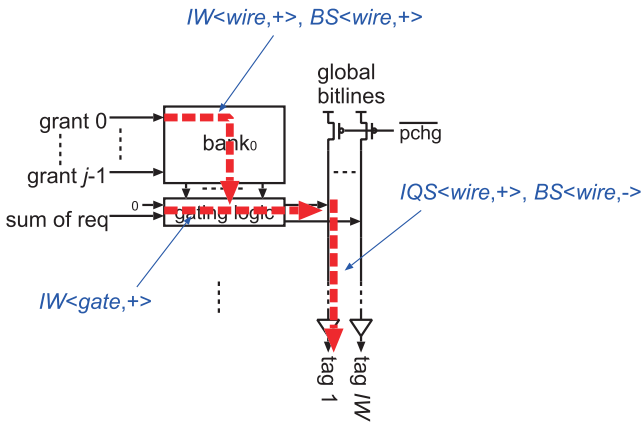


Fig. 24 Parameters that impact the delay in the tag RAM.

As described in Sect. 4.3, there are two critical paths in accessing the bank. From the experiment, the longer path goes via the gating logic only in the case of $IQS = 32$ with $IW = 4$, whereas in the other banked cases for both IWs it goes via a banked RAM.

Figure 23 shows the evaluated delay of the tag RAM for various issue queue sizes with the optimal banking configuration. Each bar is divided into two sections, containing the appropriate values from the following four categories: the SRAM bitline delay (no banking case, including the senseamp delay), wordline delay (no banking case), bank delay (banking case, including the gating logic delay if it is on the critical path), and global bitline delay (banking case). Note that, the tag RAM was not banked for $IQS \leq 16$ with both IWs, whereas it was banked for $IQS \geq 32$. As shown in the figure, the bitline delay dominates in small queues (wordline delay is too small to be visible with $IW = 4$), whereas the bank delay dominates in large queues, although the delay of the global bitline also contributes a significant part. Note that the bank delay for $IQS = 128$ with $IW = 8$ is smaller than that for $IQS = 64$ because the tag RAM is divided into smaller banks in the former case.

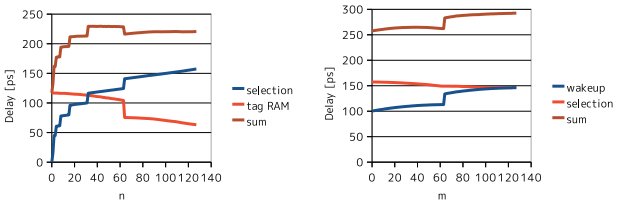
Figure 24 summarizes which parameter has an impact on the delay of the associated paths.

7.4 Critical Path

To find the critical path, we evaluated the delays in the two configurations described in Sect. 5, varying the turning points n and m. Figure 25 shows the evaluated delays of

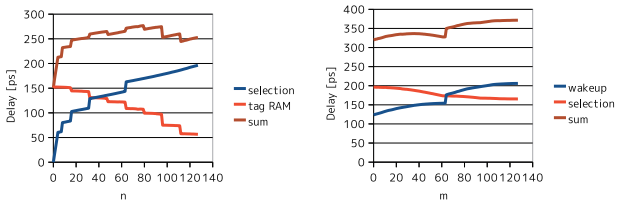
Table 3 Delay of slowest paths in configurations A and B.

issue queue size	4-issue				8-issue			
	configuration A		configuration B		configuration A		configuration B	
	n	delay [ps]	m	delay [ps]	n	delay [ps]	m	delay [ps]
8	7	247	7	247	7	289	7	289
16	15	276	15	277	15	326	15	328
32	31	309	28	319	31	360	31	376
64	63	349	61	369	31	427	63	460
128	43	430	127	464	79	552	127	593



(a) Configuration A (b) Configuration B

Fig. 25 Delay of components for various turning points with $IQS = 128$ and $IW = 4$.



(a) Configuration A (b) Configuration B

Fig. 26 Delay of components for various turning points with $IQS = 128$ and $IW = 8$.

the components in a 128-entry issue queue with $IW = 4$, for example, in configurations A and B, varying n and m, respectively. The graph for configuration A shows the selection logic delay, tag RAM delay, and their sum; the delay of the wakeup logic is constant for n and is not shown. The graph for configuration B shows the wakeup logic delay, selection logic delay, and their sum; the delay of the tag RAM is constant for m and is not shown.

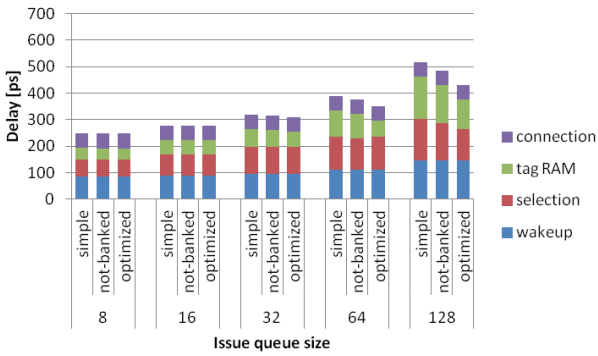
In configuration A, the delay of the selection logic increases by $O(\log_2 n)$, because the number of adders on the signal path increases at this rate. On the other hand, the delay of the tag RAM is roughly constant without a discontinuous drop, which is caused by which bank the signal traverses; the signal through the bank for lower entries must traverse the global bitline completely. The slowest path is found when the path turns at $n = 43$.

In configuration B, the delay of the wakeup logic increases slowly as m increases (the discontinuous rise is caused by a repeater inserted in the tag lines; the evaluated signal does not go through the repeater when $n \leq 63$). On the other hand, the delay of the selection logic is almost constant, because the number of adders to the output in the last entry is constant and independent of the input position (see Fig. 5); the slant is caused by wire delays. The slowest path occurs when the path turns at $m = 127$.

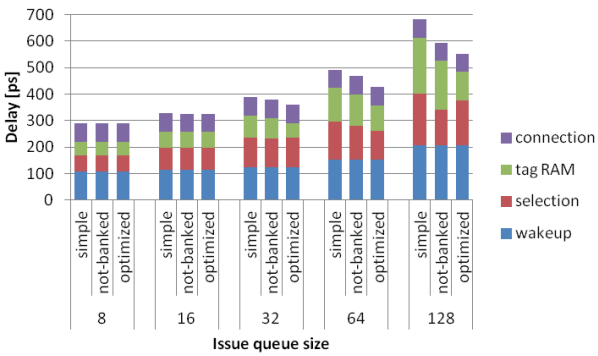
Since the faster of the slowest paths in configurations A and B is the one in configuration A, configuration A is the better configuration, and the path in configuration A is the critical path.

Figure 26 shows an 8-issue case. Characteristics similar to the 4-issue case are observed. Note that the discontinuous drops in the tag RAM delay in Fig. 26 (a) appear more frequently than in the 4-issue case, because the tag RAM is divided into smaller banks (16 entries).

Table 3 lists the evaluated delays of the slowest paths in configurations A and B, together with the n and m values



(a) $IW = 4$



(b) $IW = 8$

Fig. 27 Total delay of issue queue with design optimizations.

of the paths, with $IW = 4$ and 8. The delays of the slowest paths in configurations A and B are close, especially in small queues, but those for configuration A are slightly better, with either issue width.

7.5 Summary of Evaluation

Figure 27 summarizes the total delay of various sizes of the issue queue with different design optimization. There are

three design optimization classes. "Simple" does not divide the tag RAM into banks, and the issue queue delay is obtained by simply summing the critical path of each component. "Not-banked" does not divide the tag RAM, but a correct critical path is measured. "Optimized" divides the tag RAM into banks if beneficial, and a correct critical path is measured (the proposal in this paper). Each bar is divided into the delays of the wakeup logic, the selection logic, tag RAM, and the connection. The connection delay is the sum of the delays of the connection of the wakeup logic, selection logic, and tag RAM. As the figure shows, "optimized" reduces the issue queue delay more as the issue queue grows. The reduction rates compared with "not-banked" and "simple," are 13% and 20%, respectively, for a 128-entry queue with $IW = 4$. With $IW = 8$, the values are 8% and 23%, respectively.

8. Conclusion

This paper showed the issue queue delay for various queue sizes and issue widths of four and eight. Our evaluation results are useful as a quick reference in the microarchitectural design of a processor. In the evaluation, we designed a banked tag RAM, and identified the correct critical path of the issue queue. With these optimizations, the issue queue delay is reduced by up to 20% and 23% for 4- and 8-issue widths, respectively, for the sizes we explored, compared with a simple design using a monolithic tag RAM and with a simple calculation of the delay.

Acknowledgment

This work is supported by the Ministry of Education, Culture, Sports, Science and Technology Grant-in-Aid for Scientific Research (C) (No. 22500045). This work is also supported by VLSI Design and Education Center (VDEC), the University of Tokyo with the collaboration with Synopsys Inc.

References

[1] L. Gwennap, "Sandy Bridge spans generations," Microprocessor Report, 9/27/10-01, 2010.
[2] S. Palacharla, N.P. Jouppi, and J.E. Smith, "Quantifying the complexity of superscalar processors," Tech. Rep. CS-TR-1996-1328, University Wisconsin, Nov. 1996.
[3] M. Goshima, K. Nishino, Y. Nakashima, S. Mori, T. Kitamura, and S. Tomita, "A high-speed dynamic instruction scheduling scheme for superscalar processors," Proc. 34th Annual International Symposium on Microarchitecture, pp.225–236, Dec. 2001.
[4] Y. Kora and H. Ando, "Evaluation of issue queue delay," Proc. 2010 Symposium on Advanced Computing Systems and Infrastructures, pp.45–52, May 2010.
[5] K. Yamaguchi, Y. Kora, and H. Ando, "Banking tag RAM of issue queue and evaluation of correct critical path delay," IPSJ SIG Technical Report, vol.2011-ARC-196, no.17, July 2011.
[6] K. Yamaguchi, Y. Kora, and H. Ando, "Evaluation of issue queue delay: Banking tag RAM and identifying correct critical path," Proc. 29th International Conference on Computer Design, pp.313–319, Oct. 2011.
[7] M.D. Brown, J. Stark, and Y.N. Patt, "Select-free instruction scheduling logic," Proc. 34th Annual International Symposium on Microarchitecture, pp.204–213, Dec. 2001.
[8] E. Brekelbaum, J. Rupley, C. Wilkerson, and B. Black, "Hierarchical scheduling windows," Proc. 35th Annual International Symposium on Microarchitecture, pp.27–36, Nov. 2002.
[9] S. Palacharla, N.P. Jouppi, and J.E. Smith, "Complexity-effective superscalar processors," Proc. 24th Annual International Symposium on Computer Architecture, pp.206–218, June 1997.
[10] L. Gwennap, "AMD Bulldozer plows new ground," Microprocessor Report, 8/30/10-01, 2010.
[11] M. Goshima, Research on high-speed instruction scheduling logic for out-of-order ILP processors, Ph.D. thesis, Kyoto University, 2004.
[12] K.C. Yeager, "The MIPS R10000 superscalar microprocessor," IEEE Micro, vol.16, no.2, pp.28–40, April 1996.
[13] R.E. Kessler, "The Alpha 21264 microprocessor," IEEE Micro, vol.19, no.2, pp.24–36, March 1999.
[14] Intel, P6 Family of Processors - Hardware Developer's Manual, Sept. 1998.
[15] P.G. Sassone, J.R. II, E. Brekelbaum, G.H. Loh, and B. Black, "Matrix scheduler reloaded," Proc. 34th Annual International Symposium on Computer Architecture, pp.335–346, June 2007.
[16] http://www.mosis.com/.
[17] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45nm design exploration," Proc. 7th International Symposium on Quality Electronic Design, pp.585–590, March 2006.
[18] http://www.eas.asu.edu/˜ptm/.
[19] International Technology Roadmap for Semiconductors, 2010 update (http://www.itrs.net/).

Appendix: Delay Data

Table A· 1 gives the raw data of the evaluated delay of the optimized class in Fig. 27.

Table A· 1 Breakdown of issue queue delay.

(a) $IW = 4$

issue queue size	wakeup delay [ps]	selection delay [ps]	tag RAM delay [ps]	connection delay [ps]	total delay [ps]
8	86	61	44	55	247
16	90	80	52	54	276
32	97	100	58	54	309
64	112	124	59	54	349
128	146	119	110	54	430

(b) $IW = 8$

issue queue size	wakeup delay [ps]	selection delay [ps]	tag RAM delay [ps]	connection delay [ps]	total delay [ps]
8	108	62	50	69	289
16	113	83	61	69	326
32	125	109	56	69	360
64	152	109	96	69	427
128	206	170	107	69	552

Kyohei Yamaguchi received his B.E. from Nagoya University, Nagoya, Japan in 2011. He is currently a graduate student in the department of electrical engineering and computer science of Nagoya University.

Yuya Kora received his B.E. and M.E. degrees from Nagoya University, Nagoya, Japan in 2009 and 2011, respectively. Since 2011, he has been with Rohm Co., Ltd.

Hideki Ando received his B.S. and M.S. degrees in electronic engineering from Osaka University, Suita, Japan in 1981 and 1983, respectively. He received his Ph.D. degree in information science from Kyoto University, Kyoto, Japan in 1996. From 1983 to 1997 he was with Mitsubishi Electric Corporation, Itami, Japan. From 1991 to 1992 he was a visiting scholar at Stanford University. In 1997 he joined the faculty of Nagoya University, Nagoya, Japan, where he is currently a professor in the department of electrical engineering and computer science. In 1998 and 2002, he received the IPSJ best paper awards. His research interests include computer architecture and compilers.