Reducing Energy Consumption of Wakeup Logic through Double-Stage Tag Comparison

Yasutaka MATSUDA∗†, Nonmember, Ryota SHIOYA∗†, and Hideki ANDO††, Members

SUMMARY The high energy consumption of current processors causes several problems, including a limited clock frequency, short battery lifetime, and reduced device reliability. It is therefore important to reduce the energy consumption of the processor. Among resources in a processor, the issue queue (IQ) is a large consumer of energy, much of which is consumed by the wakeup logic. Within the wakeup logic, the tag comparison that checks source operand readiness consumes a significant amount of energy. This paper proposes an energy reduction scheme for tag comparison, called double-stage tag comparison. This scheme first compares the lower bits of the tag and then, only if these match, compares the higher bits. Because the energy consumption of tag comparison is roughly proportional to the total number of bits compared, energy is saved by reducing this number. However, this sequential comparison increases the delay of the IQ, thereby increasing the clock cycle time. Although this can be avoided by allocating an extra cycle to the issue operation, this in turn degrades the IPC. To avoid IPC degradation, we reconfigure a small number of entries in the IQ, where several oldest instructions that are likely to have an adverse effect on performance reside, to a single stage for tag comparison. Our evaluation results for SPEC2017 benchmark programs show that the double-stage tag comparison achieves on average a 21% reduction in the energy consumed by the wakeup logic (15% when including the overhead) with only 3.0% performance degradation.

key words: microprocessor, superscalar processor, instruction-level parallelism, issue queue, power consumption

1. Introduction

The high energy consumption of a microprocessor causes several problems. First, increasing the clock frequency, which is the most effective way to improve the single-thread performance, becomes difficult because it linearly increases the power/energy consumption, while the resulting generated heat has already reached a limit. Second, the battery life of mobile devices is reduced as the microprocessor is operated, thus reducing dynamic energy. Increased by high energy consumption. An increase of only 10–15°C halves the lifetime of the device [4].

The issue queue (IQ) is a major energy consumer in a microprocessor [5], and consumes 18–40% of the core energy [6], [7]. An effective way of improving performance is to increase the IQ size, and it has in fact steadily increased as the generation of processors has advanced [8]–[10]. However, an increase in the IQ size increases the IQ energy consumption. The IQ comprises the wakeup logic, select logic, tag RAM, and payload RAM. Among these components, the wakeup logic consumes the most energy. The wakeup logic comprises content-addressable memory (CAM) and includes many tag comparators that check the readiness of instructions. The number of comparators is tremendous in the large IQ of a current processor (e.g., 1,536 in 128-entry, 6-issue IQ). The tag comparator is configured using dynamic logic [11] because a short delay is essential. The wakeup logic is on the critical path of the IQ and lies on one of the critical paths in the processor. Unfortunately, the tag comparator is highly inefficient in terms of its energy consumption because it consumes dynamic energy if tags are not matched. Generally, the number of consumers of the result of a single instruction is very small (often only one), and thus, only a very small number of tag comparisons result in a match, with the majority of comparisons not matched. Consequently, tag comparators consume much dynamic energy every cycle.

This paper proposes a scheme called double-stage tag comparison (DSC) to reduce the energy consumed by tag comparison in the wakeup logic. The scheme first compares the lower bits of a tag and then, only if these bits match, compares the remaining higher bits. Thus, if the lower bits are not matched, the comparator of the higher bits is not operated, thus reducing dynamic energy.

The DSC circuit is an application of the pipelined CAM circuit, which was proposed in a general-purpose large-scale CAM [12]–[14]. The DSC is the first application of the pipelined CAM to wakeup logic. Unfortunately, a straightforward application lengthens the clock cycle time owing to the two sequential tag comparisons. To avoid this, we allocate two clock cycles for the issue operation. This, however, degrades IPC because dependent instructions cannot be issued back-to-back. To alleviate this IPC degradation, we propose using an additional queue, which is called the program order queue (PQ) [15]. The PQ holds the pointers to instructions in the IQ in a program order. Every cycle, in a small number of entries of the IQ (two in our config-
uration described in Sect. 4) pointed by the pointers stored in the several head entries of the PQ, the tag comparators are reconfigured to an unpipelined single stage, while the tags are compared over two stages in all other entries. Because the PQ holds pointers in program order, this scheme attempts to issue the several oldest instructions in a single cycle as per normal. Because tag comparison occurs in a single stage for only a fraction of the entries, the deterioration in terms of energy reduction, compared with full double-stage tag comparison, is only slight, whereas IPC degradation is dramatically reduced compared with the case that the PQ is not added. The idea behind this is that only a small number of oldest instructions in the IQ reside on the head of the critical path of dataflow [15]–[17], and IPC degradation is thus significantly alleviated if only these oldest instructions are issued in a single cycle. Our evaluation results obtained using SPEC2017 benchmark programs show that the DSC achieves on average 21% reduction in the energy consumed by the wakeup logic (15% when including the energy consumed by the PQ), with only 3.0% performance degradation.

The remainder of this paper is organized as follows. Section 2 gives an overview of the IQ and wakeup logic circuits including the tag comparators. The DSC scheme is explained in Sect. 3, while the evaluation results are presented in Sect. 4. Related work is discussed in Sect. 5, and finally, our conclusions are given in Sect. 6.

2. Organization of the IQ and Tag Comparator

We first describe the organization of the IQ in Sect. 2.1 and then explain the circuits and energy consumption of the wakeup logic and tag comparator in Sect. 2.2.

2.1 Organization of IQ

There are two types of IQ organization depending on the type of wakeup logic circuit: CAM and RAM (MS: matrix scheduler) [18]. The DSC is applied to the CAM type. We explain the organization of CAM type IQ in this section. We compare the energy consumption and area of the DSC with the MS in Sect. 4.8.

The IQ fundamentally comprises the wakeup logic, select logic, tag RAM, and payload RAM [19], [20], as illustrated in Fig. 1. The wakeup logic is a one-dimensional array in which each entry holds the tags for two source operands and ready flags that indicate the data dependence state (resolved or not) for the corresponding instruction. If both data dependences are resolved, an issue request is sent to the select logic, which grants some requests by considering the resource constraints. The grant signals are sent to the payload RAM that holds instructions. Instructions are read (issued) from the payload RAM and sent to the function units. The grant signals are also sent to the tag RAM, the wordlines of the tag RAM are directly connected to the grant signals (i.e., there is no address decoder), and the destination tags are read by responding to the grant signals. These tags are broadcast to the wakeup logic to update the ready flags.

The issue operation described above is pipelined. The wakeup, select, and tag RAM read are performed in the first cycle whereas the payload RAM is read in the second cycle. The first cycle operations are not generally pipelined; if pipelined, dependent instructions cannot be issued back-to-back and the IPC is thus degraded. The path comprising these three operations is the critical path of the IQ, and a critical path of the processor [11].

We next describe dispatch policy for the benefit of the reader. In an old era of superscalar processors, the IQ comprised a FIFO buffer, where instructions are dispatched to the tail of the buffer. In the first era of superscalar processors, the FIFO buffer was implemented in a strictly physical manner, using a shift mechanism. The shifting operation, called compaction, filled the “holes” created by issued instructions with younger instructions [21], making the IQ capacity-efficient.

After this era, the FIFO buffer was implemented as a circular buffer, where the shift mechanism was eliminated because it was complicated and consumed much power. This organization was simple, but it was inefficient in terms of capacity, because “holes” between the head and tail of the queue were not filled by newly dispatched instructions.

In current processors, the IQ is implemented by not a FIFO buffer but a non-ordered simple buffer, where instructions are simply dispatched into the “holes.” This type of IQ is called a random IQ [22]. Capacity inefficiency is resolved, while the circuit simplicity is unchanged. However, the instruction order in the IQ becomes random in terms of age, because “holes” arise randomly in the long term. This degrades IPC, because the select logic is comprised, for circuit simplicity, such that it assigns the issue priority on the basis of the entry order in the IQ (where the bottom of the queue is the highest, and the top the lowest). As a result, the instruction age order does not match the priority order that the select logic considers. It is widely known that high IPC is obtained using the age-ordered issue priority [23]. Therefore, the random IQ alone suffers from low IPC.

To mitigate IPC degradation, an additional circuit called the age matrix [16], [24] is typically used in parallel with the select logic [16], [25], [26]. The age matrix receives issue requests and grants the request from the oldest ready instruction; other grant signals are output by the select logic. Although the age matrix selects only the oldest ready instructions, it is effective [17], [22].

![Fig. 1 Basic organization of the IQ.](image)
2.2 Wakeup Logic and Energy Consumption of the Tag Comparator

This section first explains the circuits for the wakeup logic and tag comparator therein and then discusses the energy consumption of the tag comparator.

2.2.1 Wakeup Logic

Figure 2 depicts the wakeup logic [19]. Here, $IW$ and $IQS$ denote the issue width and IQ size, respectively. Driven by tag drivers, $IW$ destination tags read from the tag RAM are broadcast to all entries of the $IQS$ in the wakeup logic. Each entry has two source operand tags, which are compared with the broadcast destination tags. If there is a match, the ready flag is set. If both ready flags are set, an issue request is output.

Figure 3 shows the circuit for 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 stacked transistors are pull-down transistors that discharge the match lines according to the result of the tag comparison.

The circuit operates as follows. First, the match lines are precharged, and the destination tags are driven simultaneously. After precharging, the comparison begins. 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 H.

2.2.2 Energy Consumption of Tag Comparator

As intuitively found from the wakeup logic circuit described in Sect. 2.2.1, significant energy is consumed by the tag comparison. In this section, we explain that this energy consumption is proportional to the number of tag bits compared. Our DSC scheme reduces the energy consumed by the wakeup logic by reducing the total number of tag bits compared.

In general, operation of a CMOS device consumes the following dynamic energy [1]:

$$E = C_L V_{DD}^2,$$

where $C_L$ and $V_{DD}$ are the capacitance of the device and supply voltage, respectively.

Using this equation, the dynamic energy consumption of the tag comparator is calculated as

$$E_{cmp}(n_{tagbits}) = \left((C_{pdwn} + C_{mline}) \times n_{tagbits} + C_{pchg}\right) V_{DD}^2,$$

where $C_{pdwn}$, $C_{mline}$, $C_{pchg}$, and $n_{tagbits}$ denote the junction capacitance of two stacked pull-down transistors per bit, the wire capacitance of the match line per bit, the junction capacitance of the precharge transistor, and the number of tag bits, respectively.

Note that dynamic energy is consumed if the comparison result is unmatched, because the precharged match line is pulled down. If matched, the precharged state of the match line is retained, consuming no dynamic energy. This characteristic leads to high energy consumption in the IQ. In general, the number of consumers using the result of an instruction is small (often only one). Thus, only a few comparators result in a match, while the results of all the others are unmatched, causing high energy consumption.

It is seen from Eq. (2) that $E_{cmp}$ is roughly proportional to the number of tag bits, because $(C_{pdwn} + C_{mline}) \times n_{tagbits} \gg C_{pchg}$. Thus, the DSC scheme reduces the energy consumed by the wakeup logic by reducing the total number of tag bits to be compared.

3. Double-Stage Tag Comparison Scheme

We first describe the circuit and timing for the basic DSC scheme and then extend the scheme with the PQ to suppress IPC degradation.

3.1 Basic Scheme

To reduce the energy consumption of the wakeup logic, we propose the double-stage tag comparison (DSC) scheme, which compares first the lower bits of a tag and then the remaining higher bits only if the lower bits have
been matched. From the circuit view, the scheme always precharges the comparator for the lower bits but precharges the comparator for the higher bits only if the lower bits have been matched. This scheme reduces energy consumption for the following reasons.

- The bit width of a comparator that is always operational is reduced. Note that the dynamic energy consumption of a comparator is roughly proportional to the number of tag bits as shown in Eq. (2).
- The comparator for the higher bits operates only if the lower bits have been matched. Note that the higher-bit comparator is not precharged if the lower bits have not been matched, and the dynamic energy thus is not consumed.

So as not to increase the clock cycle time, we allocate two cycles for the operation of the IQ. Figure 4 (a) shows the conventional operation timing, while (b) shows the modified timing for the DSC. In the conventional operation, wakeup (WU), select (SL), and tag RAM read (TR) are carried out in a single cycle. Each operation is carried out in three phases, $\phi_0$ to $\phi_2$. In contrast, in the DSC, a single cycle is exclusively allocated to the wakeup (including no-operation (nop) in the third phase), and the subsequent cycle is allocated to the select and tag RAM read (including nop in the first phase). In the first cycle, the destination tags are broadcast and the lower bits are then compared in $\phi_0$ (WU1). The comparator for the higher bits is precharged at the beginning of $\phi_1$ only if the comparison of the lower bits has resulted in a match. In this case, the higher bits are compared during the remaining time for $\phi_1$ (WU2). Then, at the beginning of $\phi_2$, the ready bit is set if both the lower- and higher-bit comparators have resulted in a match. In the next cycle, select and tag RAM read are carried out as in normal operation.

Figure 5 shows the wakeup logic circuit for the DSC scheme. The destination tag read from the tag RAM is latched at $\phi_0$ (1 in Fig. 5 (a)) and broadcast to all entries in the wakeup logic. In each entry, while $\phi_0$ is H, the lower-bit comparator is precharged. If $\phi_0$ falls to L, the lower bits of the destination tag are compared with those of the source tag (2). In this phase, the higher bits of the destination tag are simply fed to the higher-bit comparator (3). The comparison result of the lower bits gates the precharge signal $\phi_1$ for the higher-bit comparator (4). In other words, it is only if the lower-bit comparison results in a match, the higher-bit comparator is precharged. After $\phi_1$ falls, the higher bits are compared (5). Then, the results of the lower- and higher-bit comparisons are ANDed (6), and the ready bit is updated at $\phi_2$.

Note that the area overhead is merely two AND-gates (4 and 6). However, gate 6 is not entirely an overhead. Figure 6 shows the decomposed match-OR circuit, which is the OR gate connected with the match lines. As shown in Fig. 6 (a), the match-OR gate is implemented with inverters and a NAND gate in conventional wakeup logic. In the DSC, the gate 6 can be implemented by replacing the inverter in the match-OR circuit with a NAND gate, as shown
in Fig. 6 (b). Therefore, the overhead is only the increase in the fanin of the first gate in the match-OR circuit.

3.2 Reconfiguration of the Tag Comparison to a Single Stage

As described in Sect. 3.1, two cycles are allocated to the issue operation. This degrades IPC because dependent instructions cannot be issued back-to-back. To suppress this degradation, we reconfigure the tag comparison circuit of the wakeup entries with several oldest instructions to be single-staged, where the lower- and higher-bit comparators are operated in parallel. The single-staged tag comparison completes the wakeup–select loop in a single cycle as per normal. The reason that we make the entries with oldest instructions single-staged is as follows. If an instruction is on the critical dataflow path, there should be many precedent instructions on its dependency chain. Consequently, such an instruction remains in the IQ for a long time. As a result, oldest instructions are likely on the critical dataflow path, and they affect the IPC adversely if two cycles are consumed for issue. This heuristic motivates the IQ with a shift mechanism in old era processors [21], and the use of the age matrix in current processors [16], [25], [26]. Although this reconfiguration does not reduce energy consumption in a few single-stage entries, a large amount of energy is saved by the double-stage comparison in numerous other entries.

To identify the several oldest instructions in the IQ, we introduce an additional queue called the program order queue (PQ) [15]. Figure 7 shows the organization of the DSC IQ with the PQ. The PQ is organized using a circular buffer, and the tail entries are allocated to the dispatched instructions in the program order. (This allocation policy is modified later in Sect. 3.3 to reduce the PQ size.) If an instruction is dispatched to the n-th entry of the IQ, n is written into the tail entry of the PQ. In other words, an entry of the PQ holds the pointer to an IQ entry, in which the associated instruction is held. Logically, every cycle, the following number of pointers are read from the head of the PQ, and the wakeup entries in the IQ to which the pointers point are then single-staged:

\[
\text{min}(M - k, P), \quad (3)
\]

where \(M\) is the predetermined allowable maximum number of single-staged entries in the IQ, \(k\) is the current number of single-staged entries in the IQ, and \(P\) is the number of the read ports of the PQ. The term \((M - k)\) is included in the above equation for the number of single-staged entries not to become excessive and be kept constant.

The process of the wakeup logic reconfiguration is pipelined as shown in Fig. 8. The PQ read and reconfiguration setting are carried out in one cycle. In the next cycle, the setting is made valid, and the wakeup logic works in the single-stage mode. Because the PQ is very small (56 bytes in the evaluation of Sect. 4) and the setting of the reconfiguration is merely to change the \(\phi\) clock using 1-bit 2-to-1 MUXes, as described later, the delay for reconfiguration is obviously less than the conventional IQ delay (i.e., clock cycle time).

We slightly change the tag comparator circuit shown in Fig. 5 to that shown in Fig. 9, so that it can switch between double- and single-stage modes. The modification is only the operation timing of the higher-bit comparator and ready latch. Specifically, the modifications involve inserting 1) a 1-bit 2-to-1 MUX (7⃣ in Fig. 9) into the input of the precharge signal of the higher-bit comparator so that it can be precharged with \(\phi_0\) in the single-stage mode, instead of the output of the AND gate 4⃣ (this allows lower- and higher-bit comparators to operate in parallel in the single-stage mode), and 2) a 1-bit 2-to-1 MUX (8⃣) into the clock input of the
ready latch so that it can be updated with $\phi_1$ in the single-stage mode. As described previously, these MUX control signals are set in the cycle before the tag comparator circuit is reconfigured to be single-staged. Therefore, the MUXes are not on the critical path of the IQ.

Finally, we explain that although the AND gate $\oplus$ resides on the critical path of the wakeup logic, it does not increase the wakeup logic delay in the single-stage mode. As described in Sect. 3.1, the overhead owing to the AND gate $\oplus$ is converted to only the increase in the fanin of the first gate in the match-OR circuit shown in Fig. 6. Because of this, the delay increase is only slight intuitively; however, the actual delay of the DSC wakeup logic in the single-stage mode is 3% shorter according to our HSPICE simulation, because the number of input bits for each lower- and higher-bit comparator in the DSC is smaller than that of the monolithic comparator in conventional wakeup, and the delay of the comparators of the DSC is thus shorter.

### 3.3 Reducing Energy Consumption of the PQ

As described in Sect. 3.2, the PQ is organized as a circular queue, where the pointers to the IQ of dispatching instructions are written only to the tail entries. Because the pointers are not written into the entries between the head and tail entries, this organization suffers from low capacity efficiency. To avoid dispatch stalls due to the shortage of PQ entries, the PQ is required to have a size similar to that of the re-order buffer (ROB). This large PQ consumes much energy.

To reduce the PQ size, we introduce an alternative policy in PQ writes, where PQ writes occur only for empty entries at the tail. This means that only $m(< DW)$ instructions write their IQ pointers to the PQ if there are only $m$ empty entries at the tail (where $DW$ denotes the dispatch width); the remaining $DW - m$ instructions do not write their pointers to the PQ, and there is thus no dispatch stall. We call this PQ write policy the no-stall policy. This policy has the advantage that a small PQ is allowed without dispatch stalls, reducing the energy consumption. However, it has the disadvantage that the opportunities of single-staging the wakeup logic are partially lost, degrading the performance. However, according to our evaluation results in Sect. 4.4, which discusses this tradeoff, the PQ size can be significantly reduced with only modest performance degradation.

A possible optimization appropriately selects $m$ instructions from $DW$ instructions. We evaluated several heuristics that include age order (i.e., program order), integer instruction first, and branch instruction first. However, little change was observed among the heuristics we evaluated. We decided to use the heuristic of floating-point instruction first as the first key and age order as the second key. This heuristic improves the performance by 0.1% points on average relative to using the simplest age order according to our evaluation.

### 4. Evaluation Results

We first describe the evaluation methodology in Sect. 4.1. We then evaluate the performance of the DSC in Sect. 4.2. In Sects. 4.3 and 4.4, the sensitivities to the allowable maximum number of single-stage entries and PQ size are evaluated, respectively. In Sects. 4.5 and 4.6, the reduction of the energy consumption of the wakeup logic and the energy consumption overhead of the PQ are then evaluated, respectively. Section 4.8 compares the energy consumption and area of the DSC with those of the matrix scheduler.

#### 4.1 Methodology

We built a simulator based on the SimpleScalar Tool Set version 3.0a [27] to evaluate the IPC. The instruction set used was Alpha ISA. We used all the programs from the SPEC2017 benchmark suite except gcc and wrf; these programs were excluded because they do not run correctly on our simulator at present. The configuration of the base processor used for the evaluation is summarized in Table 1. The optimized default parameters of the DSC are listed in Table 2. We simulated 100M instructions after the first 16B instructions were skipped using the noref speed inputs.

In addition to the IPC evaluation, we evaluated the delay and energy consumption. An existing IQ power simulator, McPAT [29], incorporates CAM power models for the wakeup logic. However, it assumes the traditional CAM circuit, whereas the DSC uses a different circuit. Hence, rigorous low-level reliable simulations must be performed to obtain useful energy consumption and delay data. We

| Pipeline width | 6-instruction wide for each of fetch, decode, issue, and commit |
|---------------|---------------------------------------------------------------|
| Reorder buffer | 256 entries                                                   |
| IQ            | 128 entries, w/ age matrix                                    |
| Load/store queue | 128 entries                                                |
| Physical registers | 256(int) + 256(fp)                                 |
| Branch prediction | 36-bit history                                 |
| Function unit | 512-entry weight table perceptron [28], 2K-set 4-way BTB, 10-cycle misprediction penalty |
| L1 I-cache | 32KB, 8-way, 64B line                                         |
| L1 D-cache | 32KB, 8-way, 64B line, 2 ports                              |
| L2 cache | 2-cycle hit latency, non-blocking                            |
| Main memory | 1MB, 16-way, 64B line, 1-cycle hit latency, 300-cycle min. latency, 8B/cycle bandwidth |
| Data prefetch | stream-based: 32-stream tracked, 16-line distance, 2-line degree, preffect to L2 cache |

| Comparators bits | lower 3-bit, higher 6-bit                                    |
|-------------------|---------------------------------------------------------------|
| PQ configuration  | 64 entries, 2-read/6-write ports                              |
| PQ write policy   | no-stall                                                      |
| Allowable maximum number of single-stage entries | 4 entries |
4.2 Performance of the DSC

We evaluated the performance (i.e., IPC) of the following two DSC models.

- **Always two-stage**: The tag comparators are always configured as double stage.
- **Reconf**: The tag comparators are reconfigured between double and single stage using the PQ.

Figure 10 shows the evaluation results. The Y-axis represents the performance degradation relative to that of the baseline processor, where the wakeup is carried out with single-stage comparators. Higher bars correspond to worse performance, and thus lower bars are better.

As shown in the figure, the use of the **always two-stage** model significantly degrades the performance; the degradation has an average value of 7.0% and maximum value of 21.8%. In contrast, the **reconf** model successfully suppresses the performance degradation; the average degradation is only 3.0%. Thus, the DSC without reconfiguration is not acceptable, and the reconfiguration to the single stage is effective, despite that the maximum number of single-stage entries of the IQ is only four.

In general, performance degradation for the **reconf** model is small in integer (INT) programs, whereas there are several floating-point (FP) programs for which there is not small performance degradation. This means that the probability that only oldest instructions are on a critical path is smaller for FP programs than for INT programs, because the IPC is higher for FP programs, and we thus infer that multiple critical paths disperse independently of age. However, the average performance degradation in FP programs is still small (i.e., 4.5%).

Although the IPC is degraded by 3%, the clock cycle time is possibly reduced because the wakeup logic delay of the DSC is 3% shorter than the conventional wakeup logic delay, as described in Sect. 3.2. Evaluating the IQ circuits other than the wakeup logic, we found that the overall IQ delay is reduced by 1%. The IQ critical path is one critical path of the processor and the net performance degradation is thus possibly 2%.

4.3 Sensitivity to the Allowable Maximum Number of Single-Stage Entries in the IQ

Figure 11 shows the evaluated results of the performance sensitivity to the allowable maximum number of single-stage entries in the IQ. The Y-axis represents the average performance degradation relative to that of the baseline processor, while the X-axis represents the allowable maximum number of single-stage entries in the IQ. Zero on the X-axis means that reconfiguration is disabled. The performance improves as the allowable maximum number of single-stage entries increases, but the energy consumption increases. Therefore, the balance is important.

As shown in the figure, having only four entries (our choice) for single staging significantly reduces the performance degradation, and the performance improvement is only slight from four to eight. This saturation does not arise from the saturation of the number of instructions issued from single-staged entries. Figure 12 shows the ratio of instructions issued from single-staged entries to the total number of issued instructions.
number of issued instructions. As shown in the figure, the ratio clearly increases as the allowable maximum number of single-stage entries increases even when the number is beyond four. This implies that only a small number of oldest instructions are on a critical path.

4.4 Sensitivity to the PQ Size

Figure 13 shows the evaluation results of the performance sensitivity to the PQ size. The Y-axis represents the average performance degradation relative to that of the baseline processor, while the X-axis represents the PQ size. Two-hundred and fifty-six entries of the PQ is the full size because the ROB size is 256 entries. Two lines represent the performance degradation when the PQ write policy is stall and no-stall, respectively. In stall policy, the dispatch stalls if the PQ becomes full, whereas no-stall does not stall by writing the pointers only for empty entries of the PQ (see details in Sect. 3.3). A larger PQ size improves the performance but increases the energy consumption. Therefore, the balance is again important.

As shown in the figure, the performance is sensitive to the PQ size in stall policy. The dispatch stalls significantly degrade the performance when the PQ size is small, and the performance recovers when the PQ size is increased. However, even with 128 entries, which is the same size as the IQ, the stall policy is inferior to the no-stall policy. In contrast, the performance is less sensitive to the PQ size in no-stall policy. Having only 64 entries (our choice) sufficiently suppresses the performance degradation, compared with the full size (256 entries) case.

4.5 Reduction in the Energy Consumption of Wakeup Logic

This section evaluates the energy consumption of the wakeup logic of the DSC, comparing with that of the conventional IQ. The evaluated energy consumption includes the energy consumed in the entire circuit in the wakeup logic, including comparators, tag drivers, match-OR gates, and the ready latch. We assume that no dynamic energy is consumed in invalid entries and in the comparator for an already-ready operand by disabling precharge for the comparators.

Figure 14 shows the energy consumption of the wakeup logic when varying the allocation of comparator bits. The Y-axis represents the energy consumption relative to that of the conventional wakeup logic. Conv on the X-axis denotes the conventional wakeup logic, and remaining columns on the X-axis denote the DSC wakeup logic, where each number on the X-axis denotes the number of bits allocated to the lower-bit comparator. If the value on the X-axis is n, then the number of bits allocated to the higher-bit comparator is 9 - n. Each bar is divided into dynamic and static energy consumption.

A greater number of lower-bit comparator bits corresponds to a greater number of unmatched lower-bit comparators and thus more dynamic energy consumed by the lower-bit comparators. However, as the number of operating higher-bit comparators decreases, less dynamic energy is consumed by the higher-bit comparators. Therefore, an optimal bit allocation exists.

The figure shows that the energy consumption is least when there are three lower-bit comparator bits. The reduction in the total energy consumption is 21%. The dynamic energy consumption significantly reduces by as much as 36%. However, the static energy consumption increases by 34% because there are two sets of comparators (lower-bit and higher-bit comparators) in a single entry. The leakage current path is the charged match line to the stacked N-channel transistors in the comparator (see Fig. 3). The number of leakage paths is double in the DSC, compared with the conventional wakeup logic. However, the overall energy reduction is still considerable.

4.6 Energy Overhead of the PQ

Figure 15 shows the magnitude of the energy overhead of the PQ. The Y-axis represents the energy consumption relative to that of the conventional wakeup logic. The left and right bars respectively represent the energy consumption of the conventional and DSC wakeup (WU) logic including that of the PQ. In the figure, “st” and “dyn” represent static and dynamic energy, respectively.

As shown in the figure, the energy consumption of the
PQ is small, being only 5.6% for the dynamic and static energy combined, compared with the energy consumption of the conventional wakeup logic. The energy consumption reduction of the DSC is still considerable (15%).

4.7 Delay

The delay of the DSC wakeup logic is 3% shorter than that of the conventional wakeup logic according to our HSPICE simulation. See the last paragraph of Sect. 3.1 for the detailed explanation.

4.8 Comparison with the MS

This section compares the energy consumption and area of the DSC with those of the MS [18].

The MS performs the wakeup operation not with CAM but with two SRAM matrices, each corresponding to two source operands. Each row and column of the matrix corresponds to an instruction in the IQ, and a cell in a row holds a value of 1 if the instruction associated with the column is the producer of the operand of the instruction associated with the row; it otherwise holds a value of zero. The grant signal from the select logic is connected to the wordline of the SRAM matrices, which traverses the column. The values of the cells are read via the bitlines, which traverse the rows. If the read value is 1, the ready latch, which is located at the right side of the matrix, is set. If both ready latches of the rows of the two matrices are set, the instruction associated with the row becomes ready to execute and sends the issue request to the select logic.

The MS is efficient compared with the CAM-type wakeup logic, in terms of dynamic energy consumption, for two reasons. 1) SRAM is more physically dynamic-energy-efficient than CAM. 2) Only the minimum required circuits operate in the MS, whereas all comparators operate in the CAM-type wakeup logic. It is thus widely supposed that the MS consumes less energy than the CAM-type IQ.

The downside of the MS is that the size of the matrix increases quadratically with the IQ size, because the MS comprises square matrices of order IQS. Therefore, the size of the MS grows large for the large IQ used in modern processors, which increases the cost, delay, and energy consumption.

Figure 16 compares the energy consumption of the DSC with that of the MS. The Y-axis represents the energy consumption relative to that of the conventional CAM-type wakeup logic plus tag RAM. Because the MS replaces the tag RAM as well as the wakeup logic, we include the energy consumption of the tag RAM in the graph. We also include that of PQ. In contrast, the energy consumption of the MS includes the energy consumed by the table called the producer table (PT) [18], which translates the source register number of a dispatching instruction to the IQ entry number of its producer instruction. Each bar is divided into static (“st” in the figure) and dynamic (“dyn”) energy consumption of each circuit. The circuit of the MS is configured so that the delay becomes nearly equal to the delay of the DSC wakeup logic and tag RAM. The MS delay is dominated by the long wire delays of grant wordlines and ready bitlines. Therefore, we insert the minimum number of repeaters so that the delay becomes equal to the DSC wakeup delay. Although we can make the delay shorter if we insert more repeaters, the energy consumption considerably increases owing to the leakage of the repeaters; we could make the delay at most 9% shorter, but the energy consumption was increased by 4.0 times. Figure 16 thus shows the results of a delay-neutral comparison.

As shown in Fig. 16, the energy consumption of the MS is 39% greater than that of the conventional CAM-type wakeup logic and tag RAM. As a result, the MS consumes 57% more energy than the DSC.

As expected, the dynamic energy consumption of the MS matrices is small. However, the static energy is large. Much of the static energy is consumed by the ready bitline leakage. There are several reasons for this. First, eight-transistor (8T) cells rather than conventional six-transistor (6T) cells are used for the matrix SRAM. In current SRAM designs of nano-LSI technology, 8T cells are used to ensure read margins under low supply voltages [1]. (In fact, Intel switched from 6T to 8T cells for its 45nm line of Core processors [33].) When using 8T cells, each bitline is single, and pull-down transistors are connected to read by pulling the precharged bitline to L. These pull-down transistors are leaky because high-performance transistors are used to reduce the bitline delay, and charge on the bitline
flows through these leaky pull-down transistors. Note that the cross-coupled inverters of the SRAM do not consume much static energy because we use low-power transistors, which are less leaky. Second, we use the hierarchical bitline structure [1] to reduce the long bitline delay. The bitlines are long in the MS (see the circuit size of the MS in Fig. 17) and thus slow. The hierarchical bitline structure doubles the number of bitlines and thus increases the number of pull-down transistors, increasing the leakage current. Note that we could not reduce the delay of the MS to the delay-neutral level without the hierarchical bitline structure. Third, repeaters are inserted to reduce the delay of the slow bitlines. They also increase the leakage current.

Figure 17 compares the area of the wakeup logic of the DSC (along with the tag RAM and PQ) with that of the MS. The size of each circuit is scaled in proportion to the actual size. As shown in the figure, the area of the MS is 6 times that of the DSC. Note that our layout of the MS is reasonable because the transistor density of the MS with 1R/6W ports is sparser than that of a single-ported cache (i.e., one of the densest structures in processors) but denser than that of the 54-bit multiplier (dense logic array) and that of the entire Intel Skylake processor chip, as listed in Table 3.

5. Related Work

The IQ was extensively studied around 2000 and a comprehensive survey was performed by Abella et al[5].

Pipelining the comparator in a large-scale CAM was proposed in [12]–[14]. The DSC is the first application of the pipelined CAM to the wakeup logic, where simply pipelining the wakeup logic to multiple stages is unacceptable in terms of the performance. We successfully suppress the performance degradation by unpipelining the wakeup logic for the small number of IQ entries at a small expense of energy reduction.

Ernst et al. proposed decomposing the IQ into three queues, with zero, one, or two sets of comparators in each entry corresponding to the number of non-ready operands of an instruction [36]. An instruction is inserted into the appropriate IQ according to the number of non-ready operands. Energy consumption is reduced by not performing a useless tag comparison for an already-ready operand. However, decomposing the IQ reduces the capacity efficiency, because the ratio of instructions based on the number of non-ready operands does not necessarily match the ratio of the capacity of each decomposed queue.

Folegnani et al. and Ponomarev et al. independently proposed resizing the IQ according to demand [37], [38]. If the size of the IQ is reduced, the comparators in the inactivated entries are disabled, thereby saving energy. However, these schemes cannot reduce energy consumption if the demand is high, whereas our scheme can.

Michaud et al. proposed a scheme that preschedules instructions by calculating their expected issue time according to dependency and latency and places them in a simple FIFO buffer [39]. The latency of several instructions is variable, and a small conventional IQ is thus placed after the FIFO buffer. This organization replaces a large portion of the conventional IQ with a simple FIFO buffer, thereby reducing energy consumption. The drawback of this scheme is that the small conventional IQ after the FIFO buffer is easily clogged by instructions that depend on long-latency operations (e.g., cache misses), and, as a result, the entire IQ is blocked.

Palacharla et al. proposed constructing an IQ with several FIFO buffers, in which instructions are placed according to their dependencies [11]. By taking advantage of the dependency ordering in each FIFO buffer, readiness to issue can be checked by reading the scoreboard that holds the operand availability of each register, considering only the head instructions in the FIFO buffers. In this scheme, the scoreboard must have twice as many ports as the number of FIFO buffers. According to this study, a small number of FIFO buffers is sufficient to achieve high IPC, and thus, the complexity of the readiness check logic is acceptable. Although this is true for compute-intensive programs, the FIFO buffers are filled by long-latency cache missed loads and their consumers in memory-intensive programs, and a new independent dispatch instruction may be blocked owing to the lack of an empty FIFO buffer. This can be avoided by increasing the number of FIFO buffers, which in turn requires more ports for the scoreboard, causing unacceptable complications.

| Design   | Circuit                      | Tr. density (×10^{-3}/λ^2) |
|----------|------------------------------|-----------------------------|
| Authors  | wakeup logic                 | 1.586                       |
|          | matrix of MS                 | 1.168                       |
| Sun Micro [34] | 512KB L2 cache               | 3.957                       |
| Fujitsu [35] | 54-bit multiplier for FP     | 0.726                       |
| Intel    | processor (Skylake)          | 0.701                       |

Fig. 17 Area comparison of the DSC with the MS. The size of each circuit is scaled in proportion to the actual size.
Stark et al. proposed breaking the wakeup–select loop into different pipeline stages [40]. Brown et al. proposed issuing instructions without selection [41], where the select operation is eliminated from the critical path. These schemes are similar to the DSC in terms of pipelining the issue operation. However, these schemes pipeline the issue operation for all instructions to reduce the clock cycle time, whereas our scheme only pipelines the issue operation for latency-tolerant instructions. Additionally, these schemes are speculative, and misspeculation occurs frequently. This significantly increases energy consumption.

The MS comprises square matrices of order IQS, growing the area and static energy significantly in a large IQ, as described in Sect. 4.8. The MS is inefficient in terms of these matrices, because few (at most only one for each row of a single matrix) SRAM cells hold a value of 1. The authors of the MS study [18] proposed the reduction of the size of the matrices by taking advantage of the fact that the producer and consumer instructions often reside closely in a program. The proposal modifies the matrix to include only the several columns close to the diagonal cells. Although this can reduce the number of SRAM cells for the matrix, the instructions in the IQ must be ordered by program order to enable the modification of the matrix. This means that the IQ must be a circular IQ, which is inefficient in terms of capacity because “holes” created by instruction issues between the head and tail of the queue are not filled by newly dispatched instructions, as described in Sect. 2.1. This significantly decreases the performance [17].

Sakai et al. proposed an IQ scheme that assigns high priority to multiple oldest instructions [15]. The scheme divides the IQ into a large main queue and small old queue and assigns a higher priority to the instructions in the old queue than those in the main queue. The scheme moves multiple oldest instructions in the main queue to the old queue each cycle and consequently assigns higher priority to multiple oldest instructions. The PQ was first proposed in this study to identify the multiple oldest instructions in the main queue. However, a scheme that reduces the PQ size was not proposed. Such a scheme is important because the full-size PQ consumes/will consume much energy in modern and future processors having large windows. In contrast, this study proposed a scheme of reducing the PQ size at little expense of performance.

6. Conclusion

The high energy consumption of modern processors causes several problems, such as a limited clock frequency, short battery lifetime, and reduced device reliability. It is thus necessary to reduce the energy consumption of the processor. One large energy consumer in the processor is the wakeup logic in the IQ.

To reduce the energy consumption of the wakeup logic, we proposed the DSC scheme, which first compares the lower bits of the tag and then, only if these match, compares the higher bits. The DSC reduces the energy consumption by reducing the total number of bits compared. The DSC basically pipelines the IQ operation, but unpipelines the small number of entries where the critical oldest instructions are held, reducing IPC degradation through pipelining.

Our evaluation results using SPEC2017 benchmark programs show that the DSC achieves on average a 21% reduction in the energy consumed by the wakeup logic (15% when including the overhead), with only 3.0% performance degradation.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Numbers JP20K11732 and JP20H04153. This work is also supported by VLSI Design and Education Center (VDEC), the University of Tokyo with the collaboration with Synopsys Inc.

References

[1] N.H.E. Weste and D.M. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, fourth edition, Addison Wesley, 2010.
[2] “IEEE international roadmap for devices and systems, 2020 update, more Moore.” https://irds.ieee.org/, 2020.
[3] F. Monsieur, E. Vincent, D. Roy, S. Bruyre, G. Pananakakis, and G. Ghibaudo, “Time to breakdown and voltage to breakdown modeling for ultra-thin oxides (Tox<32Å),” Proceedings of the 2001 IEEE International Integrated Reliability Workshop, pp.20–25, Oct. 2001.
[4] R. Viswanath, V. Wakharkar, A. Watve, and V. Lebonheur, “Thermal performance challenges from silicon to systems,” Intel Technology Journal, vol.4, no.3, pp.1–16, Aug. 2000.
[5] J. Abella, R. Canal, and A. Gonzalez, “Power- and complexity-aware issue queue designs,” IEEE Micro, vol.23, no.5, pp.50–58, Sept.-Oct. 2003.
[6] M.K. Gowan, L.L. Biro, and D.B. Jackson, “Power considerations in the design of the Alpha 21264 microprocessor,” Proceedings of the 35th Annual Design Automation Conference, pp.726–731, May 1998.
[7] K.-S. Hsiao and C.-H. Chen, “An efficient wakeup design for energy reduction in high-performance superscalar processors,” Proceedings of the 2nd Conference on Computing frontiers, pp.353–360, May 2005.
[8] K. Krewell, “Intel’s Haswell cuts core power,” Microprocessor Report, Sept. 2012.
[9] L. Gwennap, “Skylake speed shifts to next gear,” Microprocessor Report, Sept. 2015.
[10] D. Kanter, “Intel’s Sunny Cove sits on an icy Lake,” Microprocessor Report, Feb. 2019.
[11] S. Palacharla, N.P. Jouppi, and J.E. Smith, “Complexity-effective superscalar processors,” Proceedings of the 24th Annual International Symposium on Computer Architecture, pp.206–218, June 1997.
[12] K. Pagiamtzis and A. Sheikholeslami, “Content-addressable memory (CAM) circuits and architectures: A tutorial and survey,” IEEE Journal of Solid-State Circuits, vol.41, no.3, pp.712–727, March 2006.
[13] K. Pagiamtzis and A. Sheikholeslami, “Pipelined match-lines and hierarchical search-lines for low-power content-addressable memories,” Proceedings of the IEEE 2003 Custom Integrated Circuits Conference, pp.383–386, Sept. 2003.
[14] C.A. Zukowski and S.-Y. Wang, “Use of selective precharge for low-power content-addressable memories,” Proceedings of 1997 IEEE International Symposium on Circuits and Systems, pp.1788–1791, June 1997.
[15] S. Šákai, T. Suenaga, R. Shiyoa, and H. Ando, “Rearranging random
issue queue with high IPC and short delay,” Proceedings of the 36th IEEE International Conference on Computer Design, pp.123–131, Oct. 2018.

[16] R.P. Preston, R.W. Badeau, D.W. Bailey, S.L. Bell, L.L. Biro, W.J. Bowhill, D.E. Dever, S. Felix, R. Gammack, V. Germini, M.K. Gowan, P. Gronowski, D.B. Jackson, S. Mehta, S.V. Morton, J.D. Pickholtz, M.H. Reilly, and M.J. Smith, “Design of an 8-wide superscalar RISC microprocessor with simultaneous multithreading,” 2002 IEEE International Solid-State Circuits Conference, Digest of Technical Papers, pp.334–472, Feb. 2002.

[17] H. Ando, “SWQUE: A mode switching issue queue with priority-correcting circular queue,” Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, pp.506–518, Oct. 2019.

[18] M. Goshima, K. Nishino, Y. Nakashima, S. Mori, T. Kitamura, and S. Tomita, “A high-speed dynamic instruction scheduling scheme for superscalar processors,” Proceedings of the 34th Annual IEEE/ACM International Symposium on Microarchitecture, pp.225–236, Dec. 2001.

[19] S. Palacharla, N.P. Jouppi, and J.E. Smith, “Quantifying the complexity of superscalar processors,” Tech. Rep. CS-TR-1996-1328, University of Wisconsin-Madison, Nov. 1996.

[20] K. Yamaguchi, Y. Kora, and H. Ando, “Evaluation of issue queue delay: Banking tag RAM and identifying correct critical path,” Proceedings of the 29th International Conference on Computer Design, pp.313–319, Oct. 2011.

[21] J.A. Farrell and T.C. Fischer, “Issue logic for a 600-MHz out-of-order execution microprocessor,” Journal of Solid-State Circuits, vol.33, no.5, pp.707–712, May 1998.

[22] H. Ando, “Performance improvement by prioritizing the issue of the instructions in unconfident branch slices,” Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture, pp.82–94, Oct. 2018.

[23] M. Butler and Y. Patt, “An investigation of the performance of various dynamic scheduling techniques,” Proceedings of the 25th Annual IEEE/ACM International Symposium on Microarchitecture, pp.1–9, Dec. 1992.

[24] P.G. Sassone, J. Rupley, E. Brekelbaum, G.H. Loh, and B. Black, “Matrix scheduler reloaded,” Proceedings of the 34th Annual International Symposium on Computer Architecture, pp.335–346, June 2007.

[25] M. Golden, S. Arekapudi, and J. Vinh, “40-entry unified out-of-order scheduler and integer execution unit for the AMD Bulldozer x86–64 core,” 2011 IEEE International Solid-State Circuits Conference, Digest of Technical Papers, pp.80–82, Feb. 2011.

[26] B. Sinharoy, J.A.V. Norstrand, R.J. Eickemeyer, H.Q. Le, J. Leenstra, D.Q. Nguyen, B. Konigsburg, K. Ward, M.D. Brown, J.E. Moreira, D. Levitan, S. Tung, D. Husecky, J.W. Bishop, M. Gschwind, M. Boersma, M. Kroener, M. Kaltenbacha, T. Karkhanis, and K.M. Fernsler, “IBM POWER8 processor core microarchitecture,” IBM Journal of Research and Development, vol.59, no.1, pp.2:1–2:21, Jan.-Feb. 2015.

[27] http://www.simplescalar.com/.

[28] D.A. Jiménez, “Dynamic branch prediction with perceptrons,” Proceedings of the 7th International Symposium on High-Performance Computer Architecture, pp.197–206, Jan. 2001.

[29] S. Li, J.H. Ahn, R.D. Strong, J.B. Brockman, D.M. Tullsen, and N.P. Jouppi, “McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures,” Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, pp.469–480, Dec. 2009.

[30] https://www.mosis.com/.

[31] http://ptm.asu.edu/.

[32] International Technology Roadmap for Semiconductors, (http://www.itrs2.net/).

[33] R. Kumar and G. Hinton, “A family of 45nm IA processors,” 2009 IEEE International Solid-State Circuits Conference, Digest of Technical Papers, pp.58–59, Feb. 2009.

[34] J.L. Shin, B. Petrick, M. Singh, and A.S. Leon, “Design and implementation of an embedded 512-KB level-2 cache subsystem,” IEEE Journal of Solid-State Circuits, vol.40, no.9, pp.1815–1820, Sept. 2005.

[35] G. Goto, A. Inoue, R. Ohe, S. Kashiwakura, S. Mitarai, T. Tsuru, and T. Izawa, “A 4.1-ns compact 54 × 54-b multiplier utilizing sign-select Booth encoders,” IEEE Journal of Solid-State Circuits, vol.32, no.11, pp.1676–1682, Dec. 1997.

[36] D. Ernst and T. Austin, “Efficient dynamic scheduling through tag elimination,” Proceedings of the 29th Annual International Symposium on Computer Architecture, pp.37–46, May 2002.

[37] D. Folegnani and A. González, “Energy-effective issue logic,” Proceedings of the 28th Annual International Symposium on Computer Architecture, pp.230–239, June 2001.

[38] D. Ponomarev, G. Kucuk, and K. Ghose, “Reducing power requirements of instruction scheduling through dynamic allocation of multiple datapath resources,” Proceedings of the 34th Annual IEEE/ACM International Symposium on Microarchitecture, pp.90–101, Dec. 2001.

[39] P. Michaud and A. Seznec, “Data-flow prescheduling for large instruction windows in out-of-order processors,” Proceedings of the 7th International Symposium on High-Performance Computer Architecture, pp.27–36, Jan. 2001.

[40] J. Stark, M.D. Brown, and Y.N. Patt, “On pipelining dynamic instruction scheduling logic,” Proceedings of the 33rd Annual IEEE/ACM International Symposium on Microarchitecture, pp.57–66, Dec. 2000.

[41] M.D. Brown, J. Stark, and Y.N. Patt, “Select-free instruction scheduling logic,” Proceedings of the 34th Annual IEEE/ACM International Symposium on Microarchitecture, pp.204–213, Dec. 2001.

Yasutaka Matsuda received his B.E. and M.E. degrees from Nagoya University, Nagoya, Japan in 2018 and 2020, respectively. Since then, he has been with AISIN Corporation.

Ryota Shioya received his Ph.D. in Information and Communication Engineering from the University of Tokyo in 2011. From 2011 to 2016, he was an assistant professor at the Graduate School of Engineering, Nagoya University. From 2016 to 2018, he was an associate professor at the Graduate School of Engineering, Nagoya University. Since 2018, he has been an associate professor at the Graduate School of Information Science and Technology, the University of Tokyo. His current research interests include computer systems and microprocessors. He received IEEE Computer Society Japan Chapter Young Author Award (2011), IPSJ Yamashita Award (2019), and IPSJ/IEEE-CS Young Computer Researcher Award (2020).
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 Information and Communication Engineering. He received IPSJ best paper awards in 1998 and 2002, and a best paper award at the Symposium on Advanced Computing Systems and Infrastructures in 2013. His research interests include computer architecture and compilers.