Title	A Low Area Overhead Design Method for High-Performance General-Synchronous Circuits with Speculative Execution
Authors	Shimpei Sato, Eijiro Sassa, Yuta Ukon, Atsushi Takahashi
Citation	IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E102-A, No. 12, pp. 1760-1769
Pub. date	2019, 12
Copyright	Copyright(C)2019 IEICE

PAPER *Special Section on VLSI Design and CAD Algorithms*

A Low Area Overhead Design Method for High-Performance General-Synchronous Circuits with Speculative Execution

Shimpei SATO[†a)], ***Member***, **Eijiro SASSA**[†b)], ***Nonmember***, **Yuta UKON**[†c)], ***Member***,
and **Atsushi TAKAHASHI**[†d)], ***Fellow***

SUMMARY In order to obtain high-performance circuits in advanced technology nodes, design methodology has to take the existence of large delay variations into account. Clock scheduling and speculative execution have overheads to realize them, but have potential to improve the performance by averaging the imbalance of maximum delay among paths and by utilizing valid data available earlier than worst-case scenarios, respectively. In this paper, we propose a high-performance digital circuit design method with speculative executions with less overhead by utilizing clock scheduling with delay insertions effectively. The necessity of speculations that cause overheads is effectively reduced by clock scheduling with delay insertion. Experiments show that a generated circuit achieves 26% performance improvement with 1.3% area overhead compared to a circuit without clock scheduling and without speculative execution.

key words: *circuit design, variable-latency circuit, speculative execution, general-synchronous circuit, timing-error detection*

1. Introduction

In advanced technology node, the impact of various kinds of delay variations on circuit performance becomes very large. Therefore, various kinds of approaches to ease the influence of delay variability have been proposed. In the synchronous circuit, reducing waiting time during circuit operation caused by these delay variations leads to performance improvement. In this paper, we adopt an approach using a speculative execution based variable-latency circuit [1] to reduce the waiting time aggressively.

There are various types of delay variations. The difference of delays among flip-flop paths exists even though the maximum delay between flip-flops is tried to be reduced as much as possible in conventional design methodology. In addition, the delay between flip-flops varies depending on manufacturing, environments, aging, and etc. as well as input vectors because of false paths. In this paper, we focus on the following two types of delay differences in a circuit that cause waiting time during operation. One is the delay difference of maximum delay among paths of different flip-flops and the other is the delay difference on a primitive computation that varies depending on inputs. The waiting time caused by the former one is the time elapsed beyond the worst-case delay before valid data is used in subsequent primitive computations. The waiting time caused by the latter one is the time elapsed until the worst-case delay even if valid data is obtained earlier.

In this paper, we propose a low area overhead design method of general-synchronous circuit with the speculative execution. Our method generates a high-performance variable-latency circuit from a netlist of a fixed-latency circuit synthesized by general synthesis tools. The generated circuit outputs the correct result faster than the inputted circuit. However, the timing to output the result becomes variable since speculative execution has some delay penalty when the speculation missed. Our design method introduces the speculative execution only to reduce the waiting time of the latter one which cannot be reduced by the optimization methods of general-synchronous circuit, and achieve performance improvement of a circuit by an effective combination of the methods. Note that, this paper is based on our previous works [2], [3].

The waiting time of the former one is typically reduced during performance optimization in which maximum delay is reduced as much as possible in a typical clock synchronous circuit assuming zero-clock skew (hereinafter, we refer this as complete-synchronous circuit or C-circuit) [4], [5]. However, it is difficult to equalize the maximum delay of all paths in a circuit, and the delay difference remains in optimized circuits. Clock period minimization methods [6]–[9] on general-synchronous circuit completely reduce the waiting time of the former one and achieve higher performance than C-circuits. General-synchronous circuit (hereinafter, we simply call this as “circuit”) is a synchronous circuit in which the clock signals have the same period but the timing they ticks is different. The clock scheduling [6], [7] finds a feasible clock scheduling to assign an appropriate delay to each path. The delay insertion [8], [9] expands the feasibility of the clock scheduling by increasing the minimum delay of paths and achieves the minimum clock period of a circuit assuming fixed-latency.

Speculative execution enables to improve the circuit performance [1], [2], [10]–[12] by utilizing the waiting time of the latter one. In the speculative execution, some primitive computations are executed without waiting for the completion of the prior computations of the worst-case delay. For example, when a self-loop in a circuit has different delay depending on its input, it is required to assign a clock period of its worst-case delay to guarantee the correct com-

Manuscript received March 12, 2019.
Manuscript revised July 10, 2019.

[†]The authors are with Tokyo Institute of Technology, Tokyo, 152-8550 Japan.
a) E-mail: satos@ict.e.titech.ac.jp
b) E-mail: sassa@eda.ict.e.titech.ac.jp
c) E-mail: ukon@eda.ict.e.titech.ac.jp
d) E-mail: atsushi@ict.e.titech.ac.jp

DOI: 10.1587/transfun.E102.A.1760

putation in a circuit without speculative execution. Even if the circuit has some room for performance improvement in other parts, the worst-case delay of such self-loop limits the performance improvement. Speculative execution enables to run the subsequent primitive computations earlier than the worst-case delay and achieves performance improvement.

Speculative execution is a technique to reduce the waiting time both on the delay difference among paths and the delay difference on a primitive computation. In our proposed method, since the waiting time on the delay difference among paths is reduced by the existing methods, the speculative execution can be introduced only to reduce the waiting time on the delay difference on a primitive computation. Therefore, the introduction of the speculative execution will be kept as small as possible and realizing low area overhead and high-performance circuit is expected.

In the experiments, we confirm that a circuit designed by our method achieves the performance improvement compared to a circuit without speculation and a C-circuit from gate-level simulations. And, we confirm that the performance improvement realizes with reasonable circuit size.

Main contributions of this paper are the following:

- Proposal of a design method that realizes better performance of a circuit with speculative execution while keeping the circuit size as small as possible. Clock scheduling and speculative execution have overheads to realize them, but have potential to improve the performance by averaging the imbalance of maximum delay among paths and by utilizing valid data available earlier than worst-case scenarios, respectively. We combine them effectively to achieve performance improvement while keeping the increase of circuit size in small.
- Performance improvement of digital circuits by introducing variable-latency realized with speculative execution.

2. Preliminaries

Here, the elemental techniques used in the proposed method is explained using an example circuit. Figure 1 and Fig. 2 are an example of input and output of the proposed method, respectively.

We consider a circuit consisting of flip-flops and some combinational circuits between them. Gray rectangles from r_0 to r_3 are flip-flops. A pair of values in a square between flip-flops represents the delay of the primitive computation where the upper number is the maximum delay and the lower number is the minimum delay. The maximum delay and the minimum delay include the setup time and the hold time of the flip-flop, respectively. A value in a circle on the clock signal denotes the clock signal delay.

The clock scheduling [6], [7], the delay insertion [8], [9] and the introduction of speculative execution [1] are adopted to the input circuit G and the output circuit is generated. The generated circuit works correctly when it is operated with the clock period 6. To guarantee the correct output, the input circuit is modified in the following points: the minimum delay between r_0 and r_1 is increased from 6 to 12 by the delay insertion, the flip-flop r_1 is replaced to the speculative FF, and the clock signals to each flip-flop is scheduled as in the Fig. 2.

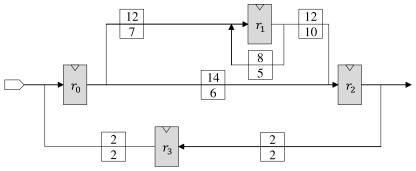


Fig. 1 An example of input circuit G of the proposed method.

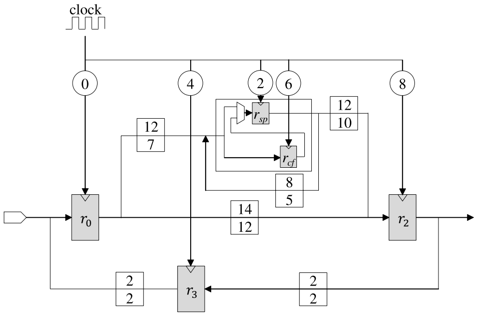


Fig. 2 An example of output circuit G_{sp} of the proposed method generated from the circuit G.

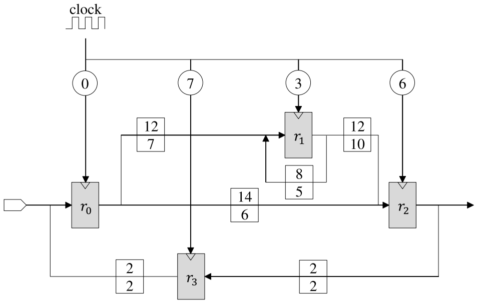


Fig. 3 An example of a clock scheduling of the circuit G.

In the following subsections, the elemental techniques to guarantee the correct operation of generated circuits by the proposed method are explained in detail.

2.1 Clock Scheduling

Here, the mechanism of the clock scheduling method reduces the waiting time that remains in a C-circuit is explained. For an easy understanding, we consider the circuit under the following conditions: enable clock scheduling, disable delay insertion, and fixed-latency.

A circuit works correctly if hold constraints and setup constraints are satisfied for signal propagation between every flip-flop pair [4]. Constraint graph [6], [13] $H(G) = (V, E)$ for the circuit G is given in Fig. 4. Where vertex set V corresponds to flip-flops in G and directed edge set E corresponds to hold constraints and setup constraints [6], [13]. In

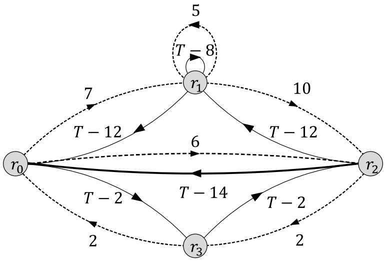


Fig. 4 The constraint graph $H(G)$ of the circuit G.

constraint graph, a flip-flop pair is connected by two edges. One is an edge corresponds to a hold constraint (dashed arrows in the graph) and the other is an edge corresponds to a setup constraint (solid arrows in the graph). The valuable T is a clock period.

The algorithm in the paper [7] finds the lower bound of a clock period and a feasible clock scheduling under that clock period on a circuit. Also, it finds a feasible clock scheduling under a given target clock period. The lower bound of the clock period T is the smallest T that satisfies the sums of weights of each cycle in a constraint graph be not less than zero. In the case of the circuit G and the constraint graph $H(G)$, the lower bound of the clock period T is 9. The cycle (r_0, r_2, r_1, r_0) is the cycle that gives the lower bound of the clock period of this circuit. The sum of weights of this cycle is $6 + (T - 12) + (T - 12) = 2T - 18$, and the sum of weights be equal to 0 when T is 9, For other cycles in the $H(G)$, the sums of weights are larger than 0 when T is 9. Thus, a feasible clock scheduling exists when T is 9.

The clock timing of r_0, r_1, r_2, and r_3 are set as 0, 3, 6, and 7, respectively as in Fig. 3. Seeing the path (r_0, r_1), the scheduling $S(r_1) - S(r_0)$ has to be smaller than the minimum delay of the path, which is 7, and $S(r_1) - S(r_0) + T$ has to be larger than the maximum delay of the path, which is 12. When $S(r_0)$ be 0 and T be 9, the range of $S(r_1)$ is from 3 to 7. A feasible clock scheduling is a clock timing of each flip-flop that satisfies all of the constraints between flip-flops.

Seeing the circuit G as a C-circuit, the lowest clock period is the same as the largest maximum delay and it is 14. Thus, a circuit which consisted of the same flip-flops and primitive computations to a C-circuit works correctly with a smaller clock period than the C-circuit.

In cases of variable-latency circuits, the clock scheduling works right since a constraint graph is given even if the speculative execution is introduced to a circuit.

2.2 Delay Insertion

Here, the mechanism of the delay insertion that expands the feasibility of the clock scheduling and achieves the minimum clock period is explained. The minimum clock period is the smallest clock period of a circuit assuming fixed-latency. For an easy understanding, we consider a circuit under the following conditions: enable clock scheduling, enable delay insertion, and, fixed-latency.

Reducing the difference between the maximum delay and the minimum delay of a primitive computation between the appropriate pair of flip-flops relaxes to find a feasible clock scheduling and leads to the clock period minimization [9]. The minimum delay is able to increase by some techniques such as inserting buffers or replacing logic cells to smaller ones.

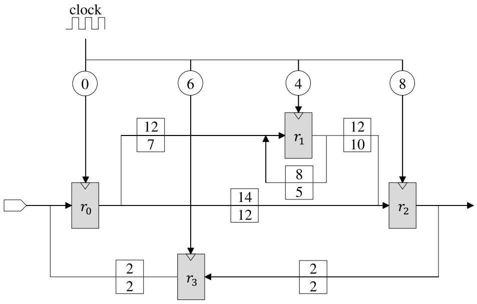


Fig. 5 An example of a circuit G_{ins}. The minimum delay between r_0 and r_2 is increased from 6 to 12.

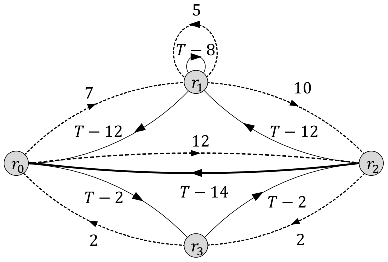


Fig. 6 The constraint graph $H(G_{ins})$ of the circuit G_{ins}.

Figure 5 shows a circuit G_{ins} in which, compared to the circuit G, the minimum delay between r_0 and r_2 is increased from 6 to 12. Figure 6 shows the constraint graph of the circuit G_{ins}.

In Fig. 4, the cycle (r_0, r_2, r_1, r_0), which gives the lower bound of the clock cycle includes a hold constraint edge (r_0, r_2). The delay insertion to the minimum delay of this path increases the weight of this edge and the limitation of the clock period is relaxed. When the minimum delay of the path (r_0, r_2) is 12 and T is larger than 6, the sum of weights of this cycle $2T - 12$ be larger than 0 and the constraints of this cycle are satisfied. In this case, this cycle does not give the lower bound of the clock period of this circuit.

In the case of Fig. 6, the lower bound of the clock period T of the circuit G_{ins} is 8. The clock timing of r_0, r_1, r_2, and r_3 are set as 0, 4, 8, and 6, respectively. The circuit G_{ins} works correctly with a smaller clock period than the circuit G. In this case, the lower bound of the clock period T is determined by the self-loop of r_1. The maximum delay of self-loop limits the minimization of a clock period by the delay insertion and the clock scheduling.

Clock period minimization by the delay insertion is effective if the lower bound of the clock period is determined by the constraints including the hold constraints. If not, the lower bound of the clock period is determined only by the setup constraints. That is, the maximum delay of a self-loop

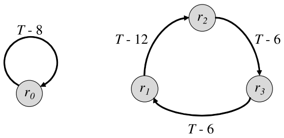


Fig. 7 Examples of a cycle consisting of setup constraint edges.

or the sum of maximum delays of a directed cycle across multiple flip-flops determine the lower bound of the clock period. Figure 7 shows examples of these cases. In these examples, the clock period T should be larger than 8 to satisfy the sum of weights of these cycles be not less than zero. In these cases, the clock scheduling with the delay insertion has no effect on the clock period minimization and the speculation is one of the solutions for further clock period reduction.

The delay insertion has the effect of expanding the feasibility of the clock scheduling also in variable-latency circuits with speculative execution. When speculative execution is introduced, the lower bound of the clock period will be determined by the other parts in the circuit. By applying the delay insertion to that part, the limitation will be relaxed and the circuit can be operated with a smaller clock cycle.

2.3 Variable-Latency Circuit by Speculative Execution

Here, we consider a circuit under the following conditions: enable clock scheduling, enable delay insertion, and variable-latency.

The variable-latency circuit by speculative execution is realized by a dynamic timing-error detection/correction (EDC) mechanism [1]. A circuit G_{sp} shown in Fig. 8 is a circuit with speculative execution. The circuit G_{sp} is implemented by introducing the EDC mechanism to the circuit G_{ins} shown in Fig. 5. In this implementation, a conventional deterministic flip-flop[†] (r_1 in the circuit G_{ins}) is replaced to the Speculative FF which consists of two flip-flops called the *spFF* and the *cfFF*.

In this circuit, timing-errors at the spFF is allowed, while no timing-error is allowed at the cfFF. The value stored in the spFF is used for an output signal of Speculative FF, and the following primitive computations start earlier. On the other hand, the value stored in cfFF is error-less since the clock signal to the cfFF is scheduled appropriately to satisfy the constraints for storing the correct output.

The error detection is done by comparing the output values of spFF and cfFF. When an error is detected, clock signals to all flip-flops except spFF are stopped by gating and the value in cfFF is copied to the spFF to correct the value in spFF at that cycle. Thus, the EDC mechanism requires one clock cycle for the recovery process.

There is no computation between cfFF and spFF, and the delays to be considered are only the wire delay, the setup time, and the hold time. Without loss of generality, we assume that cfFF and spFF are placed enough close to set the wire delay as 0 and that the maximum delay and the minimum delay between cfFF and spFF are 0. The setup time and the hold time will be enough smaller than the clock timing difference between cfFF and spFF.

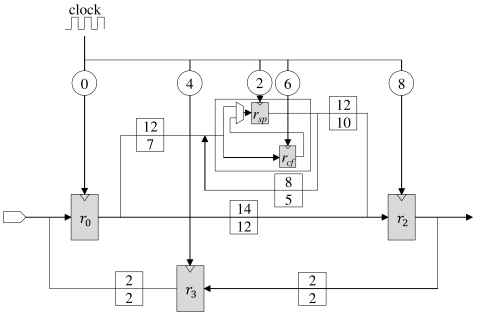


Fig. 8 An example of a circuit G_{sp} with the speculative execution. The flip-flop r_1 is replaced to the Speculative FF. This figure is the same as Fig. 2.

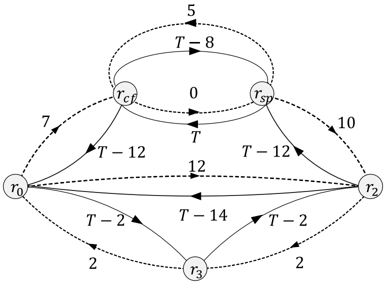


Fig. 9 The constraint graph $H(G_{sp})$ of the circuit G_{sp}.

Figure 9 shows the constraint graph of the circuit G_{sp}. The spFF is required only satisfying the constraints between following flip-flops, and the cfFF is required only satisfying the constraints between previous flip-flops. In this case, the cfFF is also required to satisfy the constraints between sfFF because the original flip-flop r1 has a self-loop. By introducing the Speculative FF as the alternative of the flip-flop r_1 in G_{ins}, the constraints are relaxed to be able to set a smaller clock period than the minimum clock period of the circuit G_{ins}. For example, the clock period 6 satisfies the constraints in the circuit G_{sp}, because the sums of weights of each cycle in G_{sp} is not less than 0. Thus, when T is 6, the circuit works correctly and it is smaller clock period than the lower bound of the clock period of the circuit G_{ins}. However, the performance improvement of this circuit relies on the error rate of the speculative FF. The clock timing of r_0, r_{sp}, r_{cf}, r_2, and r_3 are set as 0, 2, 6, 8 and 4, respectively.

Figure 10 shows the detailed implementation of the EDC mechanism. A timing-error at spFF is allowed, while no timing-error is allowed at cfFF. Thus, the value stored in spFF is erroneous but is available earlier, and the value stored in cfFF is error-less but is available later. The value stored in spFF is used for an output signal of Speculative FF, and the following primitive computations start earlier.

Figure 11 shows the timing chart of a timing-error de-

[†] A conventional deterministic flip-flop means a typical flip-flop used in fixed-latency circuit

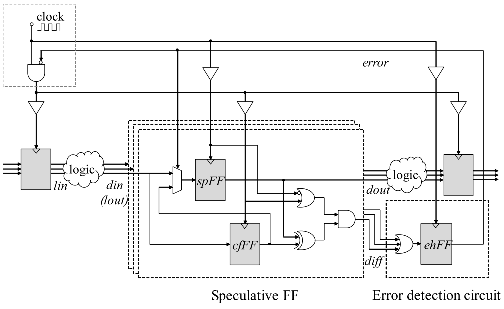
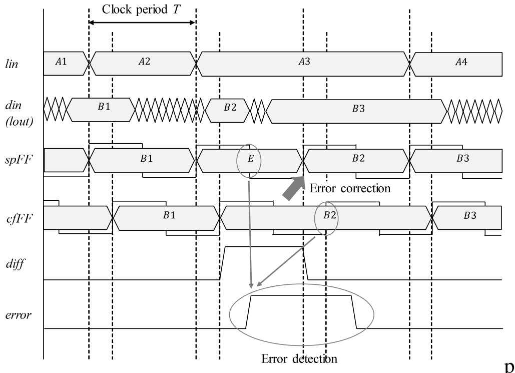


Fig. 10 Implementation of the dynamic timing-error detection/correction mechanism.



pp

Fig. 11 A timing chart in the case of timing-error.

tection and value correction of the mechanism. A timing-error is detected by comparing the values stored in spFF and cfFF by the error detection circuit. The result of comparison is stored to *ehFF* and it controls the error correction process. When ehFF outputs an error signal, all conventional deterministic flip-flops stop updating their state by clock signal gating for one clock cycle. In Speculative FFs, the value in the spFF is replaced to the value in the cfFF while stopping deterministic flip-flops. Then, the circuit returns to execute its primitive computations from the next clock cycle.

Timing constraints

Two new constraints for the EDC mechanism are required in addition to the setup constraint and the hold constraint. They are *Error signal constraint* and *Clock gating constraint*. Error signal constraint is the constraint required to set the result of comparison between spFFs and cfFFs into a flip-flop (we call this as ehFF). Clock gating constraint is the constraint required to handle the clock supply to guarantee the correctness of computations.

A circuit with speculative execution works correctly with a clock period T if the following constraints are satisfied for every signal propagation between pairs of flip-flops. Let F_{nr} be the set of deterministic flip-flops, F_{sp} be the set of spFFs, and F_{cf} is the set of cfFF. $D_{\max}(u, v)$ is the maximum delay which includes the setup time and $D_{min}(u, v)$ is the minimum delay which includes the hold time of adjacent flip-flops u and v. $S(.)$ is the clock scheduling of a flip-flop.

Normal constraints

$$P_{normal} = \{(u, v) | u \in F_{nr} \cup F_{sp}, v \in F_{nr} \cup F_{cf}\}$$
$$\forall (u, v) \in P_{normal},$$
$$S(u) - S(v) \leq T - D_{\max}(u, v)$$
$$S(v) - S(u) \leq D_{\min}(u, v)$$

Error signal constraints

$$\forall w \in F_{sp} \cup F_{cf},$$
$$S(w) - S(ehFF) \leq -D_{\max}(w, ehFF)$$
$$S(ehFF) - S(w) \leq T + D_{\min}(w, ehFF)$$

Clock gating constraint

$$\forall x \in F_{nr} \cup F_{cf},$$
$$S(x) \leq S(ehFF) \leq T + S(x)$$

Normal constraints are a setup constraint and a hold constraint when spFF and cfFF are introduced. In order to work the circuit with speculative execution correctly, spFF and cfFF have to satisfy these constraints with following flip-flops and previous flip-flops, respectively.

Error signal constraints are a setup constraint and a hold constraint between ehFF, and spFF and cfFF. In order to detect a timing-error within one clock cycle, the result of comparison between spFF and cfFF has to be transferred to the following ehFF during that clock cycle.

Clock gating constraint is a constraint to realize clock gating for one clock cycle of the error correction. For the error correction after a timing-error, a signal from ehFF has to stop the clock ticks to keep values in F_{nr} and F_{cf} for one clock cycle.

3. Design Method of General-Synchronous Circuit with Speculative Execution

We propose a low area overhead design method of a variable-latency circuit using speculative execution. The generated circuit by the proposed method achieves performance improvement by reducing the waiting time caused by the delay difference of maximum delay among paths of different flip-flops and the delay difference on a primitive computation that varies depending on inputs. The clock scheduling and the delay insertion reduce the waiting time caused by the former delay difference, and the speculative execution reduces the waiting time caused by the latter delay difference. The effective combination of these technique achieves a low area overhead design of a circuit.

Figure 12 is the proposed design flow. A variable-latency circuit is designed for a given target clock period. The inputs of the flow are a netlist of a fixed-latency circuit and a target clock period. The outputs of the flow are an optimized netlist of the inputted circuit and a clock scheduling for it. The flow processes some steps mainly including the clock scheduling, the delay insertion, and introducing the speculative execution.

As the first step of the design flow, an inputted netlist is synthesized to obtain its delay information by a standard synthesis tool. The delay information is outputted in a standard delay format (SDF).

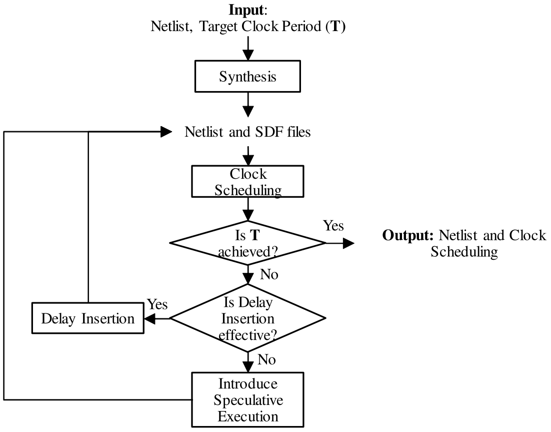

Input:
Netlist, Target Clock Period (T)
Synthesis
Netlist and SDF files
Clock Scheduling
Is T achieved?
Yes
Output: Netlist and Clock Scheduling
No
Is Delay Insertion effective?
Yes
Delay Insertion
No
Introduce Speculative Execution

Fig. 12 Design flow of circuits with speculative execution.

Next, the clock scheduling step is processed and then the feasible clock scheduling is checked whether the target clock period is achieved. Clock scheduling is processed by using the method in the paper [7] for all flip-flops. If a clock scheduling which satisfies the constraints exists, the design flow outputs the netlist of the circuit and the result of the clock scheduling and finishes.

The tool for the clock scheduling step provides some candidate ranges for a clock scheduling. In this paper, we select a schedule that the number of clock signal phases is minimum.

If a feasible clock scheduling is not found, the flow tries to relax the limitation of a clock period by the delay insertion. The delay insertion is processed by the method in the paper [9]. The clock scheduling and the delay insertion steps are iteratively executed and delay elements are implemented to the circuit as much as possible to realize the target clock period.

If the target clock period is not achieved by delay insertion, the speculative execution is introduced to the circuit. A target flip-flop to replace to a Speculative FF is appropriately selected from flip-flops of a series of primitive computations which determines the minimum clock period. If the clock period is limited by a self-loop in the circuit, the flip-flop is replaced to a Speculative FF. Otherwise, a flip-flop which comes after a path having the largest difference between the maximum delay and the minimum delay is chosen to replace, because a large amount of the delay difference leads less speculative misses when assuming the distribution of the delay as the uniform. After replacing a flip-flop, the flow returns to the clock scheduling step.

The additional circuit element for the speculative execution is larger than that of the optimization methods of a circuit because it requires additional flip-flops for the EDC mechanism. On the other hand, the delay insertion method is realized by inserting buffers or replacing gate cells to small ones and the additional circuit element for it is considered less than that for the speculative execution. Based on the above considerations, our design flow applies the delay insertion method as much as possible before introducing the speculative execution, and achieves to generate a low area overhead high-performance circuit.

4. Experimental Results

4.1 Environment

Here, we show a performance comparison between fixed-latency circuit and variable-latency circuit. In a variable-latency circuit, clock cycles required to finish some task is variable and execution time-based comparison is required for such comparison. Additionally, a certain scale of a circuit will be appropriate for these experiments. For the above reasons, we design a MIPS processor and measure the execution time of applications on it as experiments.

For the evaluations, we implement a 5-stage pipelined processor of MIPS I instruction set [14]. The performance of the circuit is measured by gate-level simulation of executing some applications on the processor. For the gate-level simulation, we use Synopsys VCS version I-2014.03-SP1-5. Note that the wire delay is not considered in this simulation.

The processor is implemented in Verilog HDL and is synthesized by Synopsys Design Compiler version I-2013.12-SP5 using ROHM 0.18 μm standard cell library. As the circuit area, we refer to the synthesis report of Design Compiler. The synthesis result of the base circuit of the processor is that the clock period as that C-circuit is 6.28 ns and the estimated area is 0.228 mm^2. The proposed method generates variable-latency circuits using the netlist of this circuit and a target clock period as inputs.

The clock schedule for the circuit is obtained from our in-house tool. The algorithms for the clock scheduling and the delay insertion are from the paper [7] and the paper [9], respectively. As a delay element, multiple pairs of NOT gates are used. The speculative execution is realized by the method in the paper [1]. Delay insertion and introducing the speculative execution are manually implemented to a netlist of the circuit. Also, we do not introduce the speculative execution to the memory access part of the processor to keep the consistency of memory access.

The applications executed on the processor are "Bubble Sort", "Quick Sort", "Eight Queens", "Towers of Hanoi", "Puzzle", and "Permutations" from Stanford Integer Benchmark [15]. The compiler used for the application is GCC 4.3.3 with optimization option O2.

The performance of circuits which is represented as *Effective clock period* is calculated by the following equation.

$$\textit{Effective clock period} = T_{op} \times C_{op}/C_{normal} \qquad (1)$$

Where T_{op} is an operate clock period, C_{op} is clock cycles to finish an application execution including the penalty of speculation misses. and C_{normal} is clock cycles to finish the application without speculation miss.

4.2 Performance and Timing-Error Rate

We designed 14 circuits while giving different target clock

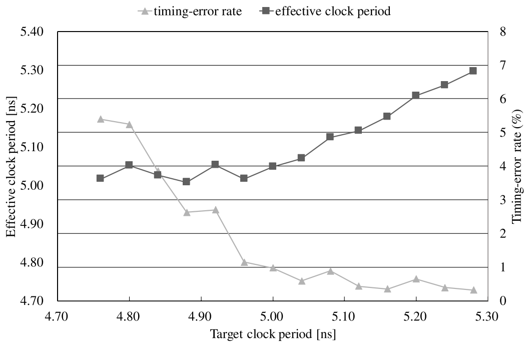
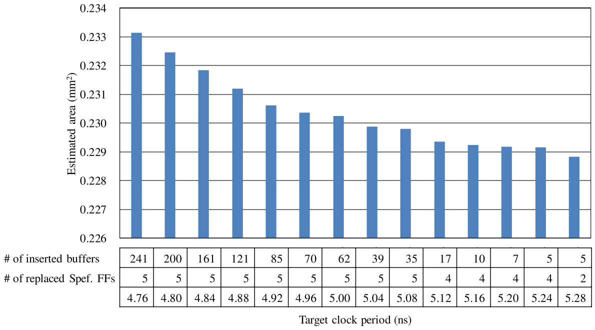


Fig. 13 Performance and timing-error rate.



# of inserted buffers	241	200	161	121	85	70	62	39	35	17	10	7	5	5
# of replaced Spef. FFs	5	5	5	5	5	5	5	5	5	4	4	4	4	2
	4.76	4.80	4.84	4.88	4.92	4.96	5.00	5.04	5.08	5.12	5.16	5.20	5.24	5.28

Fig. 14 Relative performance and area.

periods using our proposed method. Figure 13 shows the performance and timing-error rate of each circuit. The x-axis is the target clock period given for each circuit, and each circuit is operated with that clock period. The left y-axis is the effective clock period, and the right y-axis is the timing-error rate. The effective clock period is the average for all applications.

We can see that the effective clock period becomes faster for the circuits given faster target clock period even if the timing-error rate increases. However, the effective clock period saturates near 5.00 *ns*. The circuit given the target clock period 4.88 *ns* shows the best performance among 14 circuits, and the effective clock period is 5.01 *ns*.

4.3 Area

Figure 14 shows area of each circuit. The x-axis is the target clock period given for each circuit, and also the number of inserted buffer elements and the number of replaced flip-flops for speculative execution are shown in the table. The y-axis is the estimated area, and it is represented from 0.226 mm^2.

We can see that the area increases for the circuits given faster target clock period. However, the amount seems reasonable for the performance gain. The number of replaced flip-flops for speculative execution is the same in some circuits and it is confirmed that our method effectively uses the delay insertion to realize a faster circuit. The estimated area of the best performance circuit is 0.231 mm^2.

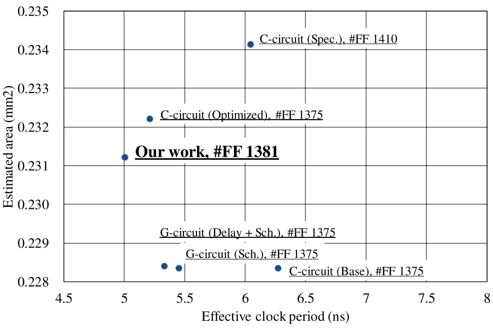


Fig. 15 Comparison of area and effective performance.

4.4 Comparison with Other Designs

4.4.1 Area and Effective Performance

Area and effective performance comparison with other designs is shown in Fig. 15. In this experiment, 6 designs are compared. All designs are obtained by modification of the design “C-circuit (base)”. The total number of flip-flops used in each design is denoted with each label.

The design “C-circuit (Base)” is a circuit synthesized with target clock period 6 *ns*. From the static timing analysis using the typical value of gates, the operating clock period of it is 6.28 *ns*. Its total number of flip-flops is 1,375 and the estimated area is about 0.228 mm^2.

The design “G-circuit (Sch.)” is a circuit that only the clock scheduling is applied to the base design. This design works correctly with the operating clock period 5.46 *ns*. The circuit element of this design is the same as the base design and the number of flip-flops and the area is the same.

The design “G-circuit (Delay + Sch.)” is a circuit that the delay insertion and the clock scheduling is applied to the base design. The delay insertion relaxes constraints for the clock scheduling and achieves clock period reduction. This design works correctly with the operating clock period 5.34 *ns*. In this design, the number of flip-flops is the same as the base design. However, some buffers are introduced by the delay insertion. Thus, the estimated area is about 0.228 mm^2, but slightly large compared to the base design.

The design “Our work” is the performance of the circuit with the speculative execution generated by our method which shows the best performance in the experiments of Sect. 4.2 and Sect. 4.3. This design is a variable-latency circuit, and its effective clock period is 5.01 *ns* when it operated with the clock period 4.88 *ns*. In this design, the number of flip-flops is 1,381, because 6 flip-flops are introduced by the speculative execution. 5 flip-flops are for the speculative FF and 1 flip-flop is for the error-correction mechanism. The estimated area is about 0.231 mm^2.

The design “C-circuit (Optimized)” is a circuit synthesized with target clock period 5 *ns*. From the static timing analysis using the typical value of gates, the operating clock period of it is 5.22 *ns*. In this design, the number of flip-flops is the same as the base design. However, the estimated area is about 0.232 mm^2, which is larger than the base design, because large logic gates are used in some part of this design to realize a small clock period.

The design "C-circuit (Spec.)" is a circuit that only the speculative execution is applied to the base design. We applied the speculative execution as much as possible to this circuit, and further application is impossible due to the hold constraints on cfFF. It is impossible to apply more speculative execution to this design. This design is a variable-latency circuit, and its effective clock period is 6.05 ns when it operated with the clock period 5.99 ns. The number of flip-flops of this design is 1,410, because 35 flip-flops are introduced by the speculative execution. 34 flip-flops are for the speculative FF and 1 flip-flop is for the error-correction mechanism. The estimated area is about 0.234 mm^2.

The circuit "Our work" achieves 26% performance improvement with 1.3% area overhead compared to the based "C-circuit (Base)". Also, it achieves 6.6% performance improvement with 1.2% area overhead compared to the "G-circuit (Delay + Sch.)" which is a circuit without the speculative execution. Compared to the "C-circuit (Optimized)", "Our work" shows better performance with less area overhead. Even if the "C-circuit (Optimized)" is further optimized to be the same performance as "Our work", its area will be larger. Compared to the "C-circuit (Spec.)", "Our work" shows better performance with less introduction of the speculative execution.

4.4.2 Power and Effective Performance

Power and effective performance comparison with other designs is shown in Fig. 16. The power is estimated by Power Compiler included in Design Compiler using SAIF file obtained from gate-level simulation using each benchmark application. Its value is the average of 6 applications.

We can see that the power increases almost linearly with the effective clock period improves. From this result, it is confirmed that the power of "Our work" is reasonable compared to the other designs.

4.5 Discussion on the Proposed Design Flow

In the design "Our work", 5 flip-flops out of 1,375 flip-flops are replaced to the speculative FF. That is the introduction of speculative execution in the flow is applied for 5 times. In the design "C-circuit (Base)", there are 2,399 paths whose maximum delay is larger than 4.88 ns and the number of endpoint flip-flops is 257 for these paths. The delay insertion and the clock scheduling reduces the candidate flip-flops to be replaced significantly.

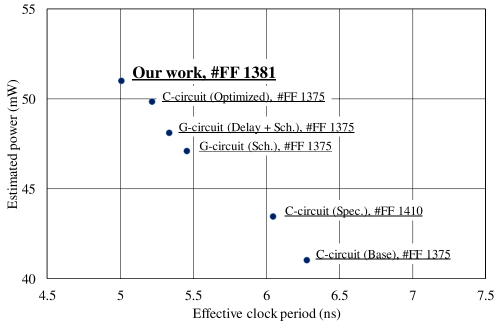


Fig. 16 Comparison of power and effective performance.

In the design "C-circuit (Spec.), which is a design only speculative execution is introduced, 34 flip-flops are replaced to the speculative FF. Many flip-flops compared to the design "Our work" have to be replaced even though the target clock period is 5.99 ns. Therefore, applying the delay insertion as much as possible before introducing the speculative execution is effective.

Furthermore, in the design "C-circuit (Spec.)", we cannot introduce the speculative execution because we cannot use a smaller clock period for it due to the hold constraint on cfFF. Delay insertion relaxes the hold constraint that limits the clock period reduction. Thus, applying the delay insertion after introducing the speculative execution is also effective for further performance improvement.

5. Related Works

Telescopic units [16], [17] is a variable-latency circuit. Based on static analysis of inputs, they assign different clock cycles to a primitive computation. For example, a primitive computation outputs a result in 1 clock cycle for some inputs and outputs a result in 2 clock cycles for other inputs. This work reduces the waiting time caused by both of the delay difference of maximum delay among paths of different flip-flops and the other is the delay difference on a primitive computation that varies dynamically depending on inputs by the variable-latency technique. Our work uses the variable-latency technique only to reduce the waiting time caused by the delay difference on a primitive computation that varies dynamically depending on inputs. Additionally, the point to introduce the variable-latency is selectable in our work while Telescopic units have to introduce it to the longest path.

6. Conclusion

In order to obtain high-performance circuits in advanced technology nodes, design methodology has to take the existence of large delay variations into account. This paper focuses on utilizing the delay difference among paths and the delay difference on a primitive computation that varies depending on inputs. We propose a high-performance digital circuit design method with speculative executions with less overhead by utilizing clock scheduling with delay insertions effectively. The necessity of speculations that cause overheads is effectively reduced by clock scheduling with delay insertion. From the evaluation using a 5-stage pipelined processor, we confirmed that the variable-latency circuit generated by our design method achieved 26% performance improvement with about 1.3% area overhead compared to a C-circuit without speculative execution which is a fixed-latency circuit. It also achieved about 6.6% performance improvement with while less than 1.2% area overhead compared to a general-synchronous circuit without speculative execution which is a fixed-latency circuit.

Acknowledgments

This work is supported by VLSI Design and Education Center (VDEC), the University of Tokyo with the collaboration with SYNOPSYS Corporation. This work is supported by JSPS KAKENHI Grant Number 17K12659.

References

[1] K. Ando and A. Takahashi, "Performance evaluation of various configuration of adder in variable latency circuits with error detection/correction mechanism," SASIMI 2012 Proceedings, pp.549–554, 2012.
[2] S. Sato, H. Nakatsuka, and A. Takahashi, "Performance improvement of general-synchronous circuits by variable latency technique using dynamic timing-error detection," SASIMI 2016 Proceedings, pp.1–4, 2016.
[3] S. Sato, E. Sassa, Y. Ukon, and A. Takahashi, "A low area overhead design for high-performance general-synchronous circuits with speculative execution," Proc. 2019 IEEE International Symposium on Circuits and Systems (ISCAS 2019), pp.1–5, 2019.
[4] J.P. Fishburn, "Clock skew optimization," IEEE Trans. Comput., vol.39, no.7, pp.945–951, July 1990.
[5] E. Friedman, "Clock distribution networks in synchronous digital integrated circuits," Proc. IEEE, vol.89, no.5, pp.665–692, 2001.
[6] A. Takahashi and Y. Kajitani, "Performance and reliability driven clock scheduling of sequential logic circuits," Proc. Asia and South Pacific Design Automation Conference, ASP-DAC'97, pp.37–42, 1997.
[7] A. Takahashi, "Practical fast clock-schedule design algorithms," IEICE Trans. Fundamentals, vol.E89-A, no.4, pp.1005–1011, April 2006.
[8] T. Yoda and A. Takahashi, "Clock period minimization of semi-synchronous circuits by gate-level delay insertion," IEICE Trans. Fundamentals, vol.E82-A, no.11, pp.2383–2389, Nov. 1999.
[9] Y. Kohira and A. Takahashi, "Clock period minimization method of semi-synchronous circuits by delay insertion," IEICE Trans. Fundamentals, vol.E88-A, no.4, pp.892–898, April 2005.
[10] D. Ernst, N.S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," Proc. 36th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-36, pp.7–18, 2003.
[11] T. Sato and Y. Kunitake, "A simple flip-flop circuit for typical-case designs for DFM," Proc. 8th International Symposium on Quality Electronic Design, ISQED'07, pp.539–544, 2007.
[12] D. Bull, S. Das, K. Shivashankar, G. Dasika, K. Flautner, and D. Blaauw, "A power-efficient 32 bit ARM processor using timing-error detection and correction for transient-error tolerance and adaptation to PVT variation," IEEE J. Solid-State Circuits, vol.46, no.1, pp.18–31, 2011.
[13] R.B. Deokar and S.S. Sapatnekar, "A graph-theoretic approach to clock skew optimization," Proc. IEEE International Symposium on Circuits and Systems, ISCAS'94, vol.1, pp.407–410, May 1994.
[14] J.L. Hennessy and D.A. Patterson, Computer Architecture, Fourth Edition: A Quantitative Approach, Morgan Kaufmann Publishers, San Francisco, CA, USA, 2006.
[15] J. Hennessy and P. Nye, "Stanford integer benchmark suite," Stanford University, 1988.
[16] L. Benini, E. Macii, M. Poncino, and G.D. Micheli, "Telescopic units: A new paradigm for performance optimization of VLSI designs," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol.17, no.3, pp.220–232, March 1998.
[17] Y.S. Su, D.C. Wang, S.C. Chang, and M. Marek-Sadowska, "An efficient mechanism for performance optimization of variable-latency designs," 2007 44th ACM/IEEE Design Automation Conference, pp.976–981, June 2007.

Shimpei Sato received the B.E., M.E., and D.E. degrees in engineering from Tokyo Institute of Technology, Tokyo, Japan, in 2007, 2009, and 2014, respectively. He is currently an Assistant Professor with the Department of Information and Communications Engineering of Tokyo Institute of Technology. From 2014 to 2016, he worked in High performance computing area as a post doctoral researcher, where he investigated an application performance analysis/tuning method. His current research interests include approximate computing realization by architecture design and circuit design and their applications. He is a member of IEEE, ACM, and IPSJ.

Eijiro Sassa received the B.E. degree from National Institute of Technology, Kumamoto College, Kumamoto, Japan, in 2018. He is currently a master course student of Department of Information and Communications Engineering in Tokyo Institute of Technology. His current research interests include VLSI layout design and combinational algorithms.

Yuta Ukon received the B.E., and M.E. degrees in electrical and electronic engineering from Osaka University, Osaka, Japan, in 2010, and 2012, respectively. Currently, he takes a doctor's course of Information and Communications Engineering, School of Engineering, Tokyo Institute of Technology. His research interests include digital circuit design, SW/HW co-design, and image processing. He is a member of IEICE.

Atsushi Takahashi received the B.E., M.E., and D.E. degrees in electrical and electronic engineering from Tokyo Institute of Technology, Tokyo, Japan, in 1989, 1991, and 1996, respectively. He was at the Tokyo Institute of Technology as a Research Associate from 1991 to 1997, and as an Associate Professor from 1997 to 2009 and from 2012 to 2015. From 2009 to 2012, he was at Osaka University, Suita, Japan, as an Associate Professor. He visited University of California, Los Angeles, U.S.A., as a Visiting Scholar from 2002 to 2003. He is currently a Professor with Department of Information and Communications Engineering, School of Engineering, Tokyo Institute of Technology. His current research interests include VLSI layout design and combinational algorithms. He is a senior member of IEEE and IPSJ, and a member of ACM.