論文 / 著書情報
Article / Book Information

Title	Design Method of Variable-Latency Circuit with Tunable Approximate Completion-Detection Mechanism
Authors	Yuta Ukon, Shimpei Sato, Atsushi Takahashi
Citation	IEICE Transactions on Electronics, Vol. E104-C, No. 7, pp. 309-318
Pub. date	2021, 7
Copyright	Copyright(C)2021 IEICE

PAPER Special Section on Solid-State Circuit Design — Architecture, Circuit, Device and Design Methodology

Design Method of Variable-Latency Circuit with Tunable Approximate Completion-Detection Mechanism

Yuta UKON[†a)], **Shimpei SATO**[†b)], ***Members***, ***and*** **Atsushi TAKAHASHI**[†c)], ***Fellow***

SUMMARY Advanced information-processing services such as computer vision require a high-performance digital circuit to perform high-load processing at high speed. To achieve high-speed processing, several image-processing applications use an approximate computing technique to reduce idle time of the circuit. However, it is difficult to design the high-speed image-processing circuit while controlling the error rate so as not to degrade service quality, and this technique is used for only a few applications. In this paper, we propose a method that achieves high-speed processing effectively in which processing time for each task is changed by roughly detecting its completion. Using this method, a high-speed processing circuit with a low error rate can be designed. The error rate is controllable, and a circuit design method to minimize the error rate is also presented in this paper. To confirm the effectiveness of our proposal, a ripple-carry adder (RCA), 2-dimensional discrete cosine transform (2D-DCT) circuit, and histogram of oriented gradients (HOG) feature calculation circuit are evaluated. Effective clock periods of these circuits obtained by our method with around 1% error rate are improved about 64%, 6%, and 12%, respectively, compared with circuits without error. Furthermore, the impact of the miscalculation on a video monitoring service using an object detection application is investigated. As a result, more than 99% of detection points required to be obtained are detected, and it is confirmed the miscalculation hardly degrades the service quality.

key words: *digital circuit design, variable-latency circuit, signal monitoring, approximate completion-detection mechanism*

1. Introduction

Advanced information-processing services such as computer vision require a high-performance digital circuit to perform high-load processing at high speed. For example, a video monitoring service demands an image-processing circuit to process a lot of images at high speed, in order to monitor video from multiple security cameras in real time. Several image-processing services do not necessarily require accurate results in primitive computations and can be tolerant of a calculation error that does not degrade service quality. In this paper, we focus on the throughput improvement of the applications that allow the calculation error.

The most popular digital circuit is a synchronous circuit that operates in synchronization with a clock signal. A process shrink has greatly contributed to improvement of the circuit performance, but in recent years Moore's law [1] is no longer applicable. This means that it is becoming more difficult to improve circuit performance using it. Moreover, in an advanced technology node, various types of delay variations have a negative effect on circuit performance. For example, a difference in the maximum path delay between flip-flops (FFs) and a difference in dynamic delay depending on an input pattern cause a long idle time in each primitive computation. These delays become a problem in a typical synchronous circuit that works with a single fixed clock cycle. Therefore, previous works have considered improving circuit performance by mitigating the effect of these delays.

The works [2] and [3] reduce the idle time of a primitive computation, which is caused by the difference in the maximum delay between FFs, by reducing the maximum delay as much as possible. However, it is difficult to make the maximum delay of all paths equal, and the difference remains even in the optimized circuit. To solve this problem, the works [4] and [5] propose general synchronous circuits. This type of circuits eliminates the difference by changing clock supply timing to each FF and achieves higher performance than that of typical synchronous circuits. The works [6] and [7] increase the flexibility of clock scheduling by increasing the minimum delay of paths and achieve further reduction of the clock period. Thanks to these works, the idle time due to the difference in the maximum delay between FFs is sufficiently reduced.

An approximate computing technique [8]–[10] enables to reduce idle time caused by the difference in dynamic delay by allowing a calculation error and enables to reduce signal propagation delay. This technique is effective, for example, in improving the throughput of an image-processing application that does not require high reliability and accuracy. However, it is difficult to design an approximate-computing circuit that satisfies target throughput while controlling an error rate so as not to degrade service quality, and this technique is used only for a few applications. Another problem with using this technique is that it would require a long circuit developing time for a use case. These problems are serious in providing various high-performance services promptly.

A timing-error detection and correction method (EDC) [11]–[14] enables circuit operation with a clock period of less than the maximum delay while guaranteeing the correct computations. An EDC-based circuit performs speculative execution using a short clock while monitoring a timing error. If the timing error is detected, the circuit takes several clock cycles to correct the output of the corresponding primitive computation. Therefore, if the circuit succeeds in

Manuscript received August 3, 2020.
Manuscript revised November 8, 2020.
Manuscript publicized December 21, 2020.

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

DOI: 10.1587/transele.2020CDP0007

the speculative execution in most of primitive computations, it achieves high-speed processing. A problem with the EDC method is that throughput improvement is restricted by the tight constraints [15], [16] for proper detection and correction of the timing error. The works [17] and [18] provide timing error prediction circuits. The former improves energy efficiency of the sub-threshold circuit by using "canary FF", and the latter achieves high-speed processing by monitoring part of a critical path and changing circuit behavior if a sign of the timing error is detected. Although these methods can reduce the effect of dynamic delay, they require a complex additional circuit and a strict constraint to avoid the timing error.

In this paper, we propose an approximate completion-detection method (ACD) that reduces idle time of primitive computations caused by the difference in dynamic delay by roughly detecting completion of processing and uses a variable different computation time for each task. Although an ACD-based circuit may make incorrect calculations due to false completion detection, it has a potential to improve throughput over the circuits that guarantee the correct computations. The error rate is controllable, and a circuit design method to minimize it also is presented in this paper. Furthermore, area increase due to an additional circuit is small because the ACD-based circuit uses a small completion detection mechanism. This method is effective, for example, in improving the throughput of an image-processing application that does not require high reliability and accuracy.

In the following, Sect. 2 describes ACD-based circuit operation and performance estimation. Section 3 illustrates configuration of the ACD-based circuit and a design method. Section 4 discusses the experimental results. Finally, Sect. 5 concludes the paper.

2. Approximate Completion-Detection Method

In this paper, we refer to a circuit that works with a single fixed clock cycle as a fixed-latency circuit. On the other hand, a circuit whose operation changes according to the dynamic delay is referred as a variable-latency circuit. The ACD-based circuit is the variable-latency circuit that has an approximate completion-detection mechanism (ACDM). The ACDM regards the processing as complete when output of the logic circuit does not change for a certain time. The ACDM monitors logic outputs using a sub clock that is faster than the main clock. If all outputs do not change for a while, the ACDM regards that the processing is completed and requests to start the next processing. By using the ACDM, the completion of processing is quickly detected according to the dynamic delay. The operation of a circuit using ACD method and the performance estimation are explained in this section.

2.1 Circuit Operation

Figure 1 shows the operation of the variable-latency circuit using the ACDM. In this circuit, the count value in the

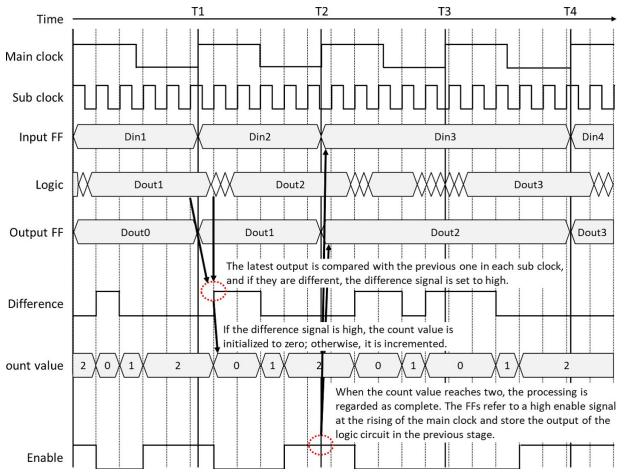

Fig. 1 Example of ACD-based circuit operation that detects the completion of the processing when the same logic output is obtained twice in a row.

ACDM is incremented for each sub clock if the output of the logic circuit is the same as the output at the previous sub clock. When the main clock is inputted to the input FF and the output FF, and if the count value is equal to a preset threshold, then, the next processing starts. This circuit uses a main clock period shorter than the maximum delay to utilize the distribution of dynamic delay effectively. In addition, a sub clock period shorter than the main clock period is used to judge the completion of the processing quickly.

The enable signal is used to notify the FFs of the judgment of the ACDM. If the enable signal is high, the FFs store the output of the logic circuit at the rising edge of the main clock; otherwise they do not. Therefore, if the processing time is shorter than the main clock period, the processing is completed in one cycle, otherwise the processing is completed after several cycles.

To detect the change in the output of the logic circuit, the output is stored in an FF every sub clock cycle, and the stored value is compared with the latest output of the logic circuit. If the output remains the same, a low difference signal is sent to the counter to increment the count value. In the ACDM, the counter counts up when it receives the low difference signal, but if the count value reaches a preset threshold value, the incrementation is stopped to prevent the malfunction caused by overflow. The threshold value is given in the circuit design process. The detail is discussed in Sect. 3.

On the other hand, if the output changes, a high difference signal is sent to the counter. When the counter receives the signal, it resets the count value to zero. This allows the ACDM to judge that the processing is completed only when the output of the logic circuit does not change continuously for a certain time.

Next, we explain the concrete behavior of the variable-latency circuit using Fig. 1. This circuit regards the process as complete if the same logic outputs are obtained twice or more in a row. At time T1, T2, and T4, the ACDM raises the enable signal because the count value is two. When the

input FF and the output FF receive the signal, they store the output of the logic circuit in the previous stage at the rising of the main clock. On the other hand, at time T3, the count value is smaller than the threshold value because it is reset several times when the change in the output is observed. In this case, the ACDM judges that the processing is not completed, and the FFs continue to hold current data Din3 and Dout2. By these operations, the ACD-based circuit starts the next processing quickly after the current processing is completed with a small additional time to detect the processing completion.

If the ACDM checks the temporarily stable logic output, it may mistakenly detect the completion of the processing and the miscalculation may be performed. On the other hand, if the ACDM takes a long time to judge the completion of processing, the ACD-based circuit may continue the processing even though it is finished. These undesirable situations need to be reduced in the design.

2.2 Effective Clock Period

The ACD-based circuit sometimes takes several main clock cycles for processing because it usually uses the main clock whose period is shorter than the maximum delay. That is to say, the processing time is different from the main clock period. For this reason, this paper considers an average processing time as a practical performance indicator. Hereinafter, the average processing time is referred to as an effective clock period. To efficiently design the ACD-based circuit that meets the target throughput, it is necessary to estimate the effective clock period. This part discusses equations for estimating the effective clock period.

Let us consider the effective clock period for N inputs. The ACD-based circuit does not start the next processing until all the ACDMs judge the completion of the processing. Hence, the number of clock cycles required for the processing depends on the worst dynamic delay. Let $w(i)$ be the worst dynamic delay of the circuit in the i-th processing ($1 \leq i \leq N$). The ACDM takes a certain time to judge after the completion of the processing. This time is represented by $C_{\text{th}} \times T_{\text{sub}}$, where C_{th} is the number of times to check the output of the logic circuit, and T_{sub} is a sub clock period. Therefore, the elapsed time from the start of the processing to the correct completion determination is represented by the following equation:

$$t_{\text{proc}}(i) = \left\lceil \frac{w(i)}{T_{\text{sub}}} \right\rceil \times T_{\text{sub}} + C_{\text{th}} \times T_{\text{sub}} \tag{1}$$

where $t_{\text{proc}}(i)$ is the elapsed time for the i-th processing. The first term of the right side in Eq. (1) is the time until the ACDM starts to check the stable output of the logic circuit. Since output is stored at the rising of the main clock, the number of clock cycles for the processing is represented by the following equation using $t_{\text{proc}}(i)$:

$$N_{\text{cycle}}(i) = \left\lceil \frac{t_{\text{proc}}(i)}{T_{\text{main}}} \right\rceil \tag{2}$$

where $N_{\text{cycle}}(i)$ is the number of clock cycles required to process the i-th input, and T_{main} is a main clock period. From the above, the effective clock period is estimated using the following equation:

$$T_{\text{eff}} = \frac{1}{N} \times \sum_{i=1}^{N} N_{\text{cycle}}(i) \times T_{\text{main}} \tag{3}$$

The ACD-based circuit operates by one main clock cycle if all the ACDMs judge the completion of the processing in less time than the main clock period. Otherwise, it takes several main clock cycles for the processing. If the circuit operates with two or more main clock cycles, the processing time may be longer than that of the fixed-latency circuit because it may exceed the maximum delay. However, the total processing time for N inputs will be reduced by processing most inputs in less than the maximum delay. To reduce the effective clock period, we should appropriately set C_{th}, T_{main}, and T_{sub}. The parameter setting is described in detail in Sect. 3.

3. Variable-Latency Circuit with ACDM

A variable-latency circuit using the ACD method is designed by adding the ACDM to a typical synchronous circuit. In this section, the configuration and the design method are explained.

3.1 Circuit Configuration

Figure 2 shows the configuration of the variable-latency circuit using the ACD method. This circuit consists of the logic circuits (Logic1, Logic2, and Logic3), flip-flops (FF1, FF2, FF3, and FF4), and ACDMs (ACDM1 and ACDM2). The ACDMs periodically check outputs of Logic1 and Logic2 and judge the completion if the outputs do not change for a certain time. Specifically, the ACDM compares the output of the logic circuit with the output of the compFF using the XOR gate to generate the difference signal. If the difference signal is low, the counter increments the count value for each sub clock; otherwise, it is reset. The comparator compares the count value with C_{th} which is a threshold. If the count value reaches C_{th}, the comparator rises the enable signal to notify the FFs of the completion of the processing. The enable signals are aggregated using the AND gate tree, and the aggregated enable signal is used to control the rising timing of the main clock. When the aggregated enable signal is low, the main clock is not sent to the FFs even it is high. Therefore, they continue to hold the current values. This enables the circuit to continue the current processing until all the ACDMs detect the completion of the processing.

The complex AND gate tree takes a long time to aggregate the enable signals. As a result, the aggregated enable signal may be delayed in reaching the AND gate, and the circuit may continue processing for the same input even

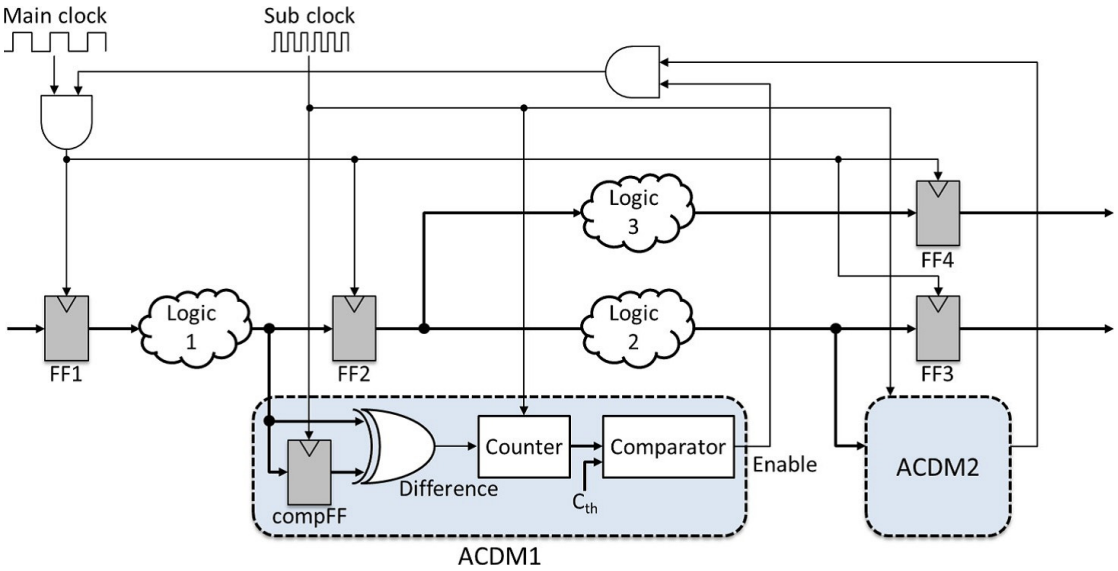

Fig. 2 Configuration of variable-latency circuit with ACDM.

though the processing is complete. Therefore, there is a possibility that the ACD-based circuit using a short main clock requires an extra main clock cycle due to the propagation delay of the aggregated enable signal, which increases the effective clock period. To avoid the extra main clock cycle, it is necessary to reduce the number of enable signals using minimum required ACDMs.

To minimize the increase of circuit area and power, it is desirable to use a small number of ACDMs by only monitoring logic circuits with long signal propagation paths. In addition, it is possible to reduce the size of the ACDM itself by reducing the number of signal lines to be monitored.

3.2 Design Method of Variable-Latency Circuit with ACDM

Figure 3 shows the overall design flow of the variable-latency circuit using the ACD method. In this design flow, a process of adding the ACDM is added to a conventional synchronous circuit design flow. The procedure is as follows: first of all, source code files (.vhd, .v, or .sv) are made using hardware description languages (HDLs) such as VHDL, Verilog, and SystemVerilog. After that, a netlist is generated by logic synthesis with the source code files. The ACDM addition process generates a netlist including the ACDM and suggests its optimal parameters. This process consists of three main steps: the step of selecting the FF section where adds the ACDM, the step of estimating T_{eff}, and the step of simulating the effective clock period and error rate. Finally, the layout process places the components on the target device and connects the components based on the modified netlist. Most of the processes allow the use of existing tools for synchronous circuit design since they are the same as conventional ones. By using the existing tools, a designer can efficiently develop the ACD-based circuit by utilizing their design skill and knowledge.

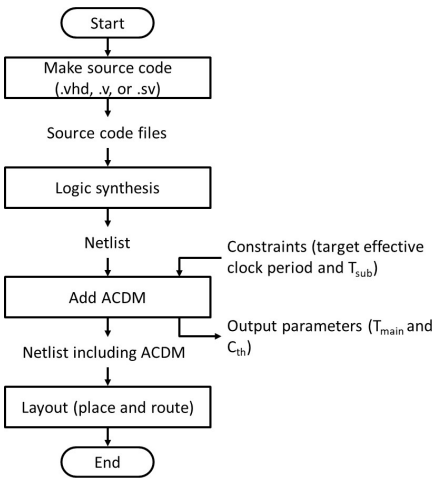

Fig. 3 Overall design flow.

The detail of the ACDM addition process is shown in Fig. 4. This process receives the netlist from the logic synthesis process and constraints (target effective clock period and T_{sub}) from the designer. After several steps, the netlist including the ACDM is generated and the optimal parameters T_{main} and C_{th} are suggested. Using a fast sub clock, the time until the ACDM judges the completion of the processing is reduced. However, if the judgment time is shorter than the minimum delay of the logic circuit, the ACDM makes an incorrect judgment by checking the stable output of the previous processing. Therefore, the sub clock period should be longer than the minimum delay.

The first step in this process determines where to add the ACDM by analyzing the dynamic delay in each FF section by using a technique such as [19] that estimates delay distribution between FFs in short time. Precisely, the step adds the ACDM to FF sections where the worst dynamic delay exceeds the target effective clock period. The second

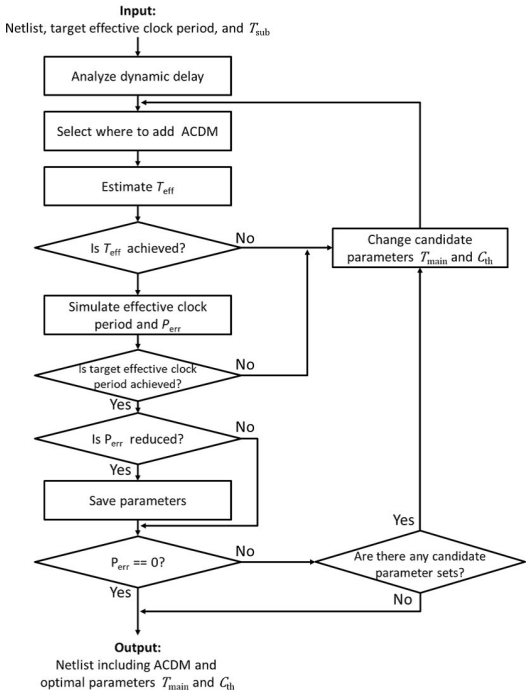

Fig. 4 Detail of ACDM addition process.

step estimates T_{eff} from Eqs. (1) to (3) using the delay distribution and gives candidate parameters T_{main} and C_{th}. If T_{eff} is longer than the target effective clock period, the estimation is repeated after changing the candidate parameters and reselecting FF sections where the ACDM should be added. By excluding the parameter set that does not satisfy the target effective clock period in step 2, the total simulation time in step 3 is reduced. In step 3, the effective clock period and error rate is evaluated by gate-level simulation with an electronic design automation (EDA) tool. In this paper, the error rate P_{err} is calculated using the following equation:

$$P_{\text{err}} = \frac{N_{\text{err}}}{N_{\text{sample}}} \times 100 \tag{4}$$

where N_{err} is the number of miscalculations, and N_{sample} is the total number of inputs.

As a result of the simulation, if the target effective clock period is not achieved, the process returns to step 1 after changing the parameters. This happens because T_{eff} is estimated with minimum conditions. Otherwise, the parameter set is saved if P_{err} is reduced. If P_{err} is zero, this process completes after outputting the netlist including the ACDM and the optimal parameters T_{main} and C_{th}. Otherwise, the process returns to step 1. If there is no candidate parameter set, the process completes after outputting the netlist including the ACDM and the second-best parameter set.

The effective clock period is reduced by using small T_{main} and C_{th}. On the other hand, the error rate increases by using these parameters because the ACD-based circuit using them is more likely to follow the incorrect instruction from the ACDM. From the above, the ACDM addition process should start with small parameters to achieve the target effective clock period.

In the ACDM addition process, it is not practical to evaluate every data path for every input vector pattern because it takes a long design time. Therefore, our design reduces the evaluation time by simulation using sample data. This method would not give the accurate effective clock period and error rate, but the reliability can be improved by increasing the number of sample data. It is assumed that random data or application data will be used for the simulation. The simulation using application data would be able to evaluate with high accuracy the effective clock period and the error rate of the circuit that processes biased data.

For example, image data often has pixels with similar colors. If the image-processing circuit processes similar pixels in succession, short dynamic delay would occur because the signal passes almost the same path. Therefore, the effective clock period would be shorter than that evaluated using random data because the ACD-based image-processing circuit operates at high speed for the input pattern. Furthermore, the error rate would be small because the number of erroneous judgments is reduced.

There is a trade-off between the evaluation accuracy and the design time. The simulation using a large amount of sample data improves the evaluation accuracy but increases the evaluation time. Therefore, iterating the simulation to obtain the minimum error rate takes a long design time. Fine-tuning the parameters may help you to find more suitable parameters, but this also increases your design time. Therefore, we need to limit the range and resolution of parameters in consideration of the required accuracy and time-to-market.

4. Experiment

To confirm the advantage of the ACD method, the effective clock period and error rate of an ACD-based ripple-carry adder (RCA), 2-dimensional discrete cosine transform (2D-DCT) circuit [20], and histogram of oriented gradients (HOG) feature calculation circuit [21] are evaluated. In addition, the area and power of the circuits are evaluated to confirm the overhead of adding the ACDM. Finally, object detection accuracy of the object detection application using the ACD-based HOG feature calculation circuit is evaluated. In this section, we discuss the evaluation results.

4.1 Preparation

The advantage of the ACD method is confirmed by evaluating several circuits. One is the RCA, which is a basic unit used in a wide range of applications. In addition, two more complex image-processing circuits, 2D-DCT circuit and HOG feature calculation circuit, are evaluated. The 2D-DCT circuit is used for image and video compression applications, and the HOG feature calculation circuit is used for object detection and image classification applications. We chose these circuits as examples where high throughput is required.

It will describe the ACD-based circuits that we designed. The ACD-based RCA adds two 32-bit inputs and outputs a 33-bit value. This circuit uses one ACDM to monitor the value.

The ACD-based 2D-DCT circuit calculates some 8×8 coefficient matrices from an image. For more information on the 2D-DCT circuit, please refer to [20]. The designed circuit has eight logic circuits in parallel, and each of which outputs a 40-bit coefficient. These outputs are monitored by eight ACDMs; if one or more ACDMs judge that the processing is not complete, the circuit continues the current processing.

The ACD-based HOG feature calculation circuit calculates some histograms from an image. This circuit contains a luminance calculation logic circuit, gradient intensity calculation logic circuit, and gradient direction calculation logic circuit. These logic circuits output a 24-bit value, 25-bit value, and 3-bit value.

The designed HOG feature calculation circuit is shown in Fig. 2. Logic1, Logic2, and Logic3 correspond to the luminance calculation logic circuit, gradient intensity calculation logic circuit, and gradient direction calculation logic circuit, respectively. FF2, FF3, and FF4 are a 24-bit FF, 25-bit FF, and 3-bit FF, respectively. As shown in this figure, two ACDMs are added to monitor Logic1 (luminance calculation logic circuit) and Logic2 (gradient intensity calculation logic circuit). On the other hand, Logic3 (gradient direction calculation logic circuit) does not monitor using the ACDM because the maximum delay is small.

In this paper, we focus on the difference in dynamic delay depending on an input pattern. Therefore, we designed the logic circuits using ROHM 0.18 μm standard cell library to obtain realistic signal propagation delays. On the other hand, to eliminate other influences, we assume that the clocks and the FFs are ideal, and the occurrence of metastable is not considered.

The designed ACD-based circuits use a main clock period that is shorter than the maximum delay shown in Table 1 to achieve high-speed processing. In addition, the sub clock period shorter than the main clock period is used to judge the completion of the processing quickly. In our experiments, we used the sub clock period of 0.20 nanosecond (ns) for the ACD-based RCA to quickly judge the completion of the processing. This sub clock period was also used for the ACD-based image-processing circuits. By using this sub clock period, the output of the logic circuits is checked multiple times within a cycle, even for a short main clock period. We believe that the fast sub clock is available by using state-of-the-art technology.

We also designed the EDC-based RCA for comparison of the effective clock period, area, and power. The EDC-based RCA uses 33-bit Speculative FF instead of the 33-bit FF, and an error detection circuit is added. In addition, considering the constraints proposed in [22], the clock timing difference, which is a parameter of Speculative FF, is set to 240 ns. This value is selected to reduce the effective clock period as much as possible within the range of normal operation.

Table 1 Worst maximum delay in designed circuit.

Circuit	Maximum delay (ns)
RCA	3.84
2D-DCT	17.92
HOG feature calculation	13.60

To compare the calculation accuracy, we designed two HOG feature calculation circuits using two types of approximate adders. One uses Lower-part OR adder (LOA) [9] and the other uses Segmented adder (SA) [10]. LOA consists of OR gates and full adders. This adder achieves high-speed processing by roughly calculating the lower digits using OR gates. SA has several full adder groups, and it achieves high-speed processing by restricting the path of signal propagation.

The above circuits include either 24-bit and 48-bit LOAs or 24-bit and 48-bit SAs. The designed 24-bit LOA and 48-bit LOA contain six and 17 OR gates, respectively. The designed 24-bit SA has two full adder groups, one is from the least significant bit (LSB) to the fourth bit and the other is from the fifth bit to the most significant bit (MSB). The 48-bit SA has three full adder groups, one is from the LSB to the eighth bit, another is from the ninth bit to the 28th bit, and the other is from the 29th bit to the MSB.

4.2 Evaluation

The effective clock periods and error rates of the designed ACD-based RCA were evaluated using simulation. Figures 5 and 6 show the evaluation results of the fixed-latency RCA, EDC-based RCA, and ACD-based RCA. Note that the ACD-based RCA uses the ACDM with C_{th} of one.

In these experiments, we used Synopsys VCS for gate-level simulation. The input is one million random data.

The effective clock periods of these circuits were reduced by using a small main clock period. However, the effective clock periods of the variable-latency circuits are larger than that of the fixed-latency circuit because they took multiple main clock cycles for some of the processing.

The ACD-based RCA operated without miscalculation even though the main clock period is 0.50 ns. These results indicate that the ACDM correctly detects the completion of processing. On the other hand, the fixed-latency RCA and EDC-based RCA using main clock periods of less than 3.75 ns and 3.52 ns, respectively, sometimes made a wrong calculation. The EDC-based RCA using a 3.53 ns or longer main clock period did not make incorrect calculations because it operated while properly detecting and correcting the timing error.

The minimum effective clock period of these RCAs when miscalculation did not occur is 3.75 ns, 3.53 ns, and 1.37 ns. Therefore, the effective clock period without miscalculation is reduced by about 64% and 61% compared with those of the fixed-latency RCA and the EDC-based RCA. However, note that these reduction rates are not

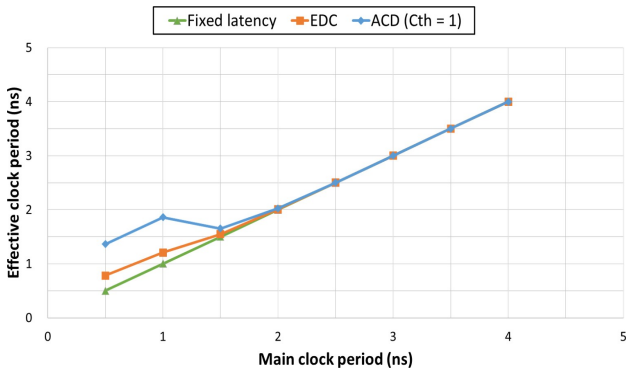

Fig. 5 Effective clock period of RCAs.

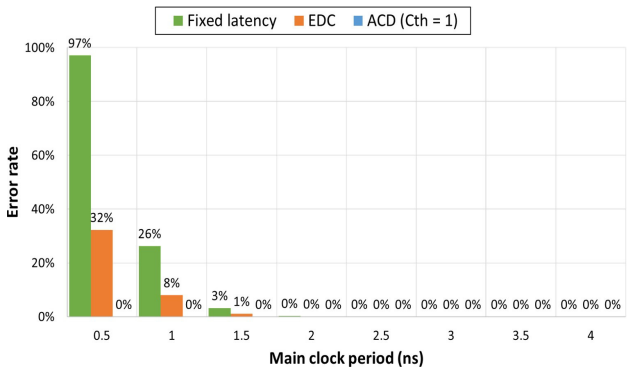

Fig. 6 Error rate of RCAs.

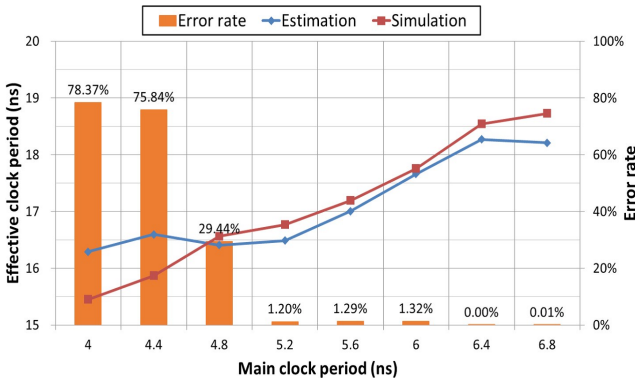

Fig. 7 Effective clock period and error rate of 2D-DCT circuit using ACDMs with C_{th} of one.

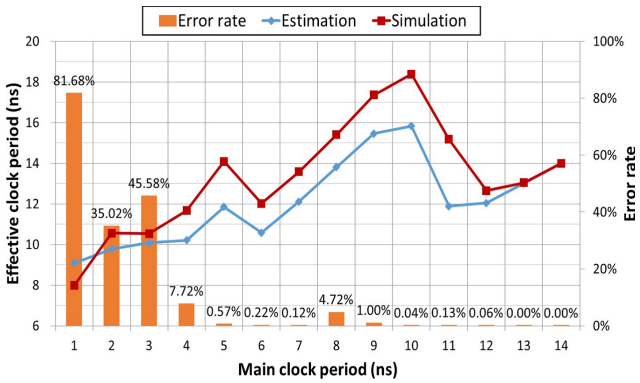

Fig. 8 Effective clock period and error rate of HOG feature calculation circuit using ACDMs with C_{th} of six.

achieved if a short main clock period is not available.

Then, the effective clock period and error rate of more complex image-processing circuits were evaluated. Figure 7 shows the results of the ACD-based 2D-DCT circuit using the ACDM with C_{th} of one. The estimated and the simulated effective clock periods were obtained from Eqs. (1) to (3) and from gate-level simulation, respectively. The error rates were obtained by comparing calculation results of the fixed latency circuit and ACD-based circuit. In the simulation, a VGA image (640×480 pixels) was used as sample data.

Figure 7 shows that the effective clock period is shortened by using a short main clock period. Although the estimated values and the simulated ones are similar for main clock periods of 4.8 ns or more, the simulated values are lower than the estimated values for shorter main clock periods. This is because the ACDM would mistakenly detect the completion of processing, and the circuit completed many calculations in an unexpectedly short time. Therefore, the error rate was higher when using a short main clock period.

Our design requires a target effective clock period. As an example, consider improving the effective clock period of the designed 2D-DCT circuit by 5% over that of the fixed-latency circuit. Although we can assume a shorter target effective clock period, it would lead a higher error rate. The ACD-based circuit uses C_{th} of one and T_{sub} of 0.20 ns. As shown in Table 1, the maximum delay of the designed 2D-DCT circuit is 17.92 ns. In this case, the effective clock period should be lower than 17.02 ns. Figure 7 shows that the target effective clock period is achieved by using a main clock period of 5.2 ns or less. The lowest error rate is obtained using the main clock period of 5.2 ns, and the effective clock period is reduced by about 6% with the error rate of about 1%.

Figure 8 shows the evaluation results of the ACD-based HOG feature calculation circuit using the ACDM with C_{th} of six. In these experiments, we used a VGA image as sample data.

The effective clock period of the ACD-based HOG feature calculation circuit was shortened by using a short main clock period. However, the simulated effective clock periods exceed the maximum delay when using the main clock period of 5 ns, 8 to 11 ns, and 14 ns. This is because the ACD-based circuit took multiple main clock cycles if the completion was not judged within a cycle. If using the ACDM with C_{th} of two or more can reduce false detection of processing completion, whereas it is more likely to take extra main clock cycles because the ACDM takes a long time to judge the completion.

As a case study to design a high-speed image-processing circuit, we consider improving the effective clock period of the designed HOG feature calculation circuit by 10% or more. To keep the error rate as small as possible,

C_{th} of six is used. T_{sub} is 0.20 ns. The maximum delay of the circuit is 12.24 ns, as shown in Table 1. The estimation results shown in Fig. 8 show that the target effective clock period is achieved using the main clock period of 12 ns, 11 ns, and seven ns or less. On the other hand, the simulation results show that it is achieved using the main clock period of six ns and four ns or less. This is because the simulation considers more complex conditions. These results suggest that using the main clock period of six ns to obtain the effective clock period of 12.02 ns with the minimum error rate of 0.22%. In other word, the designed ACD-based HOG feature calculation circuit can improve the effective clock period by about 12% with a small error rate.

Circuit area and power were evaluated to confirm the overhead of the additional circuit. The area of the designed circuits are shown in Tables 2 (a) to 2 (c), and the power of them are shown in Tables 3 (a) to 3 (c). These results were estimated using ROHM 0.18 μm standard cell library and Synopsys Design Compiler.

The area and power of the ACD-based RCA are about 17% and 4% smaller than those of the EDC-based RCA, respectively. Comparing with the fixed-latency RCA, the area and power are about 50% and 10% larger. This is because the RCA is a small circuit. On the other hand, the area of the ACD-based 2D-DCT circuit and HOG feature calculation circuit increases by 0.9% and 0.1%, respectively, and the power of them increases about 0.3% and 0.2%. Thus, these results confirmed that the overhead of the ACDM is small for complex image-processing circuits.

Tables 2 and 3 also show area-delay product (ADP) and power-delay product (PDP). This paper defines ADP as a product of area and effective clock period and PDP as a product of power and effective clock period. If ADP and PDP of the fixed-latency circuits are taken as one, those of the ACD-based RCA are 0.55 and 0.40, those of the ACD-based 2D-DCT circuit are 0.94 and 0.94, and those of the ACD-based HOG feature calculation circuit are 0.88 and 0.89. On the other hand, ADP and PDP of the EDC-based RCA are 1.71 and 1.08, respectively. Those results show that the designed ACD-based circuits improve ADP and PDP compared with the fixed-latency circuits and the EDC-based RCA.

Finally, the impact of the miscalculation on a video monitoring service using an object detection application was investigated. These experiments detected image blocks containing a moving object by comparing HOG features of three consecutive images of PETS 2009 benchmark data [23] using an inter-frame difference algorithm. The VGA images used in these experiments contains 4,800 image blocks consisting of 8×8 pixels. HOG features were calculated using the designed fixed-latency, LOA-based, SA-based, and ACD-based HOG feature calculation circuit. The ACD-based circuit used T_{main} of six ns, C_{th} of six, and T_{sub} of 0.20 ns.

Table 4 shows the effective clock periods and error rates of the HOG feature calculation circuits. The effective clock periods of the LOA-based, SA-based, and ACD-based circuits are almost the same, which is reduced by about 10% compared with that of the fixed-latency circuit. On the other hand, the error rate of the ACD-based circuit is low, whereas the error rates of the LOA-based and SA-based circuit are higher because they performed approximate calculations. However, the LOA-based and SA-based circuit were designed to minimize the error in the calculation results, so they would calculate high accurate HOG features.

The number of correct detection points and false-positive points is also shown in Table 4. The correct detection point is an image block that was judged to contain a moving object from the HOG features of the fixed-latency circuit. The false-positive point is an image block that was

Table 2 Area and normalized area-delay product (ADP) of (a) RCAs, (b) 2D-DCT circuits, and (c) HOG feature calculation circuits (using ROHM 0.18 μm standard cell library).

(a)

Method	Effective clock period (ns)	Area (μm^2) RCA	Area (μm^2) Extra	Normalized ADP
Fixed latency	3.75	6,675	-	1.00
EDC	3.53	-	5,478	1.71
ACD	1.37	-	3,366	0.55

(b)

Method	Effective clock period (ns)	Area (μm^2) 2D-DCT	Area (μm^2) Extra	Normalized ADP
Fixed latency	17.92	3,601,688	-	1.00
ACD	16.77	-	4,189	0.94

(c)

Method	Effective clock period (ns)	Area (μm^2) HOG	Area (μm^2) Extra	Normalized ADP
Fixed latency	13.60	6,412,564	-	1.00
ACD	12.02	-	5,877	0.88

Table 3 Power and normalized power-delay product (PDP) of (a) RCAs, (b) 2D-DCT circuits, and (c) HOG feature calculation circuits (using ROHM 0.18 μm standard cell library).

(a)

Method	Effective clock period (ns)	Power (mW) RCA	Power (mW) Extra	Normalized PDP
Fixed latency	3.75	6.06	-	1.00
EDC	3.53	-	0.89	1.08
ACD	1.37	-	0.60	0.40

(b)

Method	Effective clock period (ns)	Power (mW) 2D-DCT	Power (mW) Extra	Normalized PDP
Fixed latency	17.92	2286.40	-	1.00
ACD	16.77	-	0.77	0.94

(c)

Method	Effective clock period (ns)	Power (mW) HOG	Power (mW) Extra	Normalized PDP
Fixed latency	13.60	560.88	-	1.00
ACD	12.02	-	0.97	0.89

Table 4 Error rate of HOG feature calculation circuits and object detection accuracy in object detection application.

Method	HOG feature calculation circuit					Object detection application	
	T_{main} (ns)	C_{th}	T_{sub} (ns)	Effective clock period (ns)	P_{err} (%)	# of correct detection points	# of false-positive points
Fixed latency	13.60	-	-	13.60	-	120	-
LOA	12.34	-	-	12.34 (−9.3%)	99.96	10 (−91.7%)	17
SA	12.19	-	-	12.19 (−10.4%)	99.35	86 (−28.3%)	31
ACD	6.00	6	0.20	12.02 (−11.6%)	0.30	119 (−0.8%)	1

not detected from the HOG features of the fixed-latency circuit. Object detection accuracy was evaluated by comparing with the number of detection points obtained from the HOG features of the fixed-latency circuit. Using the LOA-based and SA-based circuits, the object detection accuracy decreased by about 92% and 28%. In addition, 17 and 31 false-positive points were detected. On the other hand, using the ACD-based circuit, the object detection accuracy was reduced by less than 1%, and only one false-positive point was detected. From these results we confirmed that a small number of low-quality HOG features from the ACD-based circuit hardly degrades the quality of the object detection application.

5. Conclusion

This paper proposed the method that changes the processing time by roughly detecting the completion of the processing. The proposed method enabled significantly reducing the idle time caused by the difference of the dynamic delay. Although the variable-latency circuit using the method occasionally make an incorrect calculation, the error rate was minimized by finding the optimal parameters with our design method. The experiments showed that the ACD-based RCA, 2D-DCT circuit, and HOG feature calculation circuit reduce the effective clock period by about 64%, 6%, and 12% with around 1% error rate compared with the fixed-latency circuits. In addition, it was confirmed that the object detection application using the ACD-based HOG feature calculation circuit improves the throughput with almost no decrease in object detection accuracy.

Acknowledgments

This work is supported by VLSI Design and Education Center (VDEC), the University of Tokyo with the collaboration with SYNOPSYS Corporation.

References

[1] G.E. Moore, "Cramming more components onto integrated circuits," Electronics, vol.38, no.8, 1965.
[2] J.P. Fishburn, "Clock skew optimization," IEEE Trans. Comput., vol.39, no.7, pp.945–951, 1990.
[3] E.G. Friedman, "Clock distribution networks in synchronous digital integrated circuits," Proc. IEEE, vol.89, no.5, pp.665–692, 2001.
[4] A. Takahashi and Y. Kajitani, "Performance and reliability driven clock scheduling of sequential logic circuits," Proc. ASP-DAC '97: Asia and South Pacific Design Automation Conference, pp.37–42, 1997.
[5] A. Takahashi, "Practical fast clock-schedule design algorithms," IEICE Trans. Fundamentals, vol.E89-A, no.4, pp.1005–1011, April 2006.
[6] 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.
[7] 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.
[8] H. Jiang, J. Han, and F. Lombardi, "A comparative review and evaluation of approximate adders," Proc. 25th edition on Great Lakes Symposium on VLSI, pp.343–348, 2015.
[9] H.R. Mahdiani, A. Ahmadi, S.M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," IEEE Trans. Circuits Syst. I, Reg. Papers, vol.57, no.4, pp.850–862, 2009.
[10] D. Mohapatra, V.K. Chippa, A. Raghunathan, and K. Roy, "Design of voltage-scalable meta-functions for approximate computing," 2011 Design, Automation & Test in Europe, pp.1–6, 2011.
[11] 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, 2003. MICRO-36, pp.7–18, 2003.
[12] D. Bull, S. Das, B.K. Shivshankar, G. Dasika, K. Flautner, and D. Blaauw, "A power-efficient 32b ARM ISA processor using timing-error detection and correction for transient-error tolerance and adaptation to PVT variation," Proc. 2010 IEEE International Solid-State Circuits Conference (ISSCC), pp.284–285, 2010.
[13] M. Kurimoto, H. Suzuki, R. Akiyama, T. Yamanaka, H. Ohkuma, H. Takata, and H. Shinohara, "Phase-adjustable error detection flip-flops with 2-stage hold driven optimization and slack based grouping scheme for dynamic voltage scaling," Proc. 45th Annual Design Automation Conference, pp.884–889, 2008.
[14] 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.
[15] S. Sato, E. Sassa, Y. Ukon, and A. Takahashi, "A low area overhead design method for high-performance general-synchronous circuits with speculative execution," IEICE Trans. Fundamentals, vol.E102-A, no.12, pp.1760–1769, Dec. 2019.
[16] S. Sato, H. Nakatsuka, and A. Takahashi, "Performance improvement of general-synchronous circuits by variable latency technique using dynamic timing-error detection," Proc. 20th Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI 2016), pp.60–65, 2016.
[17] H. Fuketa, M. Hashimoto, Y. Mitsuyama, and T. Onoye, "Adaptive performance compensation with in-situ timing error predictive sensors for subthreshold circuits," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.20, no.2, pp.333–343, 2012.
[18] Y. Shi, H. Igarashi, N. Togawa, and M. Yanagisawa, "Suspicious timing error prediction with in-cycle clock gating," Proc. International Symposium on Quality Electronic Design (ISQED), pp.335–340, 2013.

[19] D. Akita, K. Ando, and A. Takahashi, "Fast estimation of dynamic delay distribution," IEICE Technical Report, vol.112, no.246, pp.83–88, 2012 (in Japanese).
[20] L. Pillai, "Video compression using DCT," Application Note: Virtex-II Series, 2002.
[21] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," Proc. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05), vol.1, pp.886–893, 2005.
[22] Y. Ukon, K. Ando, and A. Takahashi, "Performance of the evaluation of a variable-latency-circuit on FPGA," IEICE Technical Report, vol.111, no.450, pp.127–132, 2012 (in Japanese).
[23] J. Ferryman, J. Crowley, and A. Shahrokni, "PETS 2009 benchmark data," http://www.cvg.reading.ac.uk/PETS2009/a.html, accessed June 13th, 2020.

Yuta Ukon received the B.E., and M.E. degrees in engineering from Osaka University, Osaka, Japan, in 2010, and 2012, respectively. In 2012 he joined NTT Microsystem Integration Laboratories, Kanagawa, Japan and has been engaged in research and development on a packet processing accelerator and packet processing circuit. He is currently with the NTT Device Innovation Center, Kanagawa, Japan. His research interests include hardware accelerator for SDN/NFV, hardware and software co-design technologies, and packet processing circuit. He is a member of IEICE.

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.

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.