PAPER *Special Section on VLSI Design and CAD Algorithms*

A Verification Method for Single-Flux-Quantum Circuits Using Delay-Based Time Frame Model

Takahiro KAWAGUCHI[†,††a)], **Kazuyoshi TAKAGI**[†,††], ***Members, and*** **Naofumi TAKAGI**[†,††], ***Senior Member***

SUMMARY Superconducting single-flux-quantum (SFQ) device is an emerging device which can realize digital circuits with high switching speed and low power consumption. In SFQ digital circuits, voltage pulses are used for carrier of information, and the representation of logic values is different from that of CMOS circuits. Design methods exclusive to SFQ circuits have been developed. In this paper, we present timing analysis and functional verification methods for SFQ circuits based on new timing model which we call delay-based time frame model. Assuming that possible pulse arrival is periodic, the model defines comprehensive time frames and representation of logic values. In static timing analysis, expected pulse arrival time is checked based on the model, and the order among pulse arrival times is calculated for each logic gate. In functional verification, the circuit behavior is abstracted in a form similar to a synchronous sequential circuit using the order of pulse arrival times, and then the behavior is verified using formal verification tools. Using our proposed methods, we can verify the functional behavior of SFQ circuits with complex clocking scheme, which appear often in practical design but cannot be dealt with in existing verification method. Experimental results show that our method can be applied to practical designs.

key words: *single-flux-quantum circuit, static timing analysis, formal verification*

1. Introduction

A superconducting single-flux-quantum (SFQ) circuit is an emerging circuit with high switching speed and low energy consumption [1]. An SFQ logic gate consists of superconducting loops with Josephson junctions and inductances. Fabrication of an SFQ circuit with more than 10,000 Josephson junctions has become possible [2]–[4]. An SFQ logic circuit uses a voltage pulse and a magnetic flux quantum as a carrier of information and for state representation, respectively. Cell-based design is adopted in designing SFQ circuits. A set of logic gates for SFQ circuits is predefined as a cell library [5]. Behavior of each gate which is represented by pulse transferring and state transition is defined in the cell library.

To design large scale SFQ circuits, computer-aided design (CAD) tools are indispensable. Because SFQ circuits use voltage pulses and behave differently from CMOS circuits, CAD tools for CMOS circuits may not be used in SFQ circuits as they are without change. Tools for clock tree synthesis [6], placement and wire routing [7], [8], circuit description [9], [10], static timing analysis [11] and formal verification [12] are developed for SFQ circuits.

SFQ circuits are driven by voltage pulses. Representation of logic values using voltage pulse is different from that using voltage level. In general, the logic values “1” and “0” are represented by the presence and the absence of a pulse, respectively. Such representation needs to distinguish the logic value “0” from the state of “no signal”. In most common logic representation using a synchronizing clock, a time frame for a gate is defined as a time section partitioned by clock pulses and the absence of a pulse in the time frame represents logic value “0” at the time frame. Therefore, A gate with clock supply, which is called a clocked gate, is commonly used in SFQ circuits. Because the clocked gates store the data, SFQ circuits is often designed with gate-level pipelines.

Switching of SFQ circuit is faster than that of CMOS circuits. To achieve multi-giga-hertz circuit, skewed clocking scheme is often chosen in SFQ circuits. There exist several clocking schemes with different behavior of a gate. In the most common clocking scheme in SFQ circuits, called concurrent-flow clocking, a data pulse arrives after the corresponding clock pulse and the gate behaves like as a logic gate combined with a delay flip-flop. In the clocking scheme called clock-follow-data clocking, a data pulse arrives before the corresponding clock pulse and the gate behaves like as a logic gate in a combinational circuit.

In general, the time frame of a gate is defined by synchronizing clock distributed to the gate. However, there often exist SFQ circuits partly or completely composed of gates without clock supply. In addition, the skewed clocking scheme is used in almost of SFQ circuits. Because of gates without clock supply and skewed clocking scheme, the time frames of these gates cannot be defined in the existing time frame model. When the time frames are not defined for a gate explicitly, the logic values also cannot be defined for the gate.

In SFQ circuits, the order of pulse arrival times of inputs of a gate effects the behavior of the gate. The simple example of the effect of the order is a clocked gate in the concurrent-flow clocking and the clock-follow-data clocking. A clocked gate in the concurrent-flow clocking has the different order of pulse arrival times of inputs from that in the clock-follow-data clocking and even if these gates are the same type, the behavior of a gate are different from that of the other gate.

The CAD algorithms and tools proposed in [11] and

Manuscript received March 13, 2015.
Manuscript revised July 13, 2015.

[†]The authors are with the Graduate School of Informatics, Kyoto University, Kyoto-shi, 606-8501 Japan.
[††]The authors are with the ALCA-JST, Kyoto-shi, 606-8501 Japan.
a) E-mail: kawaguti@lab3.kuis.kyoto-u.ac.jp

DOI: 10.1587/transfun.E98.A.2556

[12] assume that a synchronizing clock is distributed to all gates in a circuit and only the concurrent-flow clocking scheme is employed because the behavior of SFQ gates, which is not clocked or using the other clocking, could not be represented as a logic function and state transition. Static timing analysis and formal verification for SFQ circuits employing other clocking schemes such as clock-follow-data clocking scheme and/or using gates without clock supply has not been proposed. Such circuits are verified by only logic simulation. Logic simulation is hard to verify a large SFQ circuit on exhaustive input patterns. Omission of verification on exhaustive input patterns can overlook errors in corner cases. Formal verification is effective to detect such errors.

In this paper, we propose a verification method of SFQ circuits using delay-based time frame model. The proposed method aims to verify an SFQ circuit employing not only concurrent-flow clocking scheme but also other clocking schemes and composed of not only gates with clock supply but also gates without clock supply by representing behavior of the circuit as a logic function and state transition. The model assumes the unique clock period of the SFQ circuit under verification and the periodicity of pulse arrival time. In the model, time frames for each gate are defined using the assumptions and the path delay from the primary inputs to the inputs of the gate. The proposed verification method detects errors of an SFQ circuit by static timing analysis and formal verification. There are two main purposes of the static timing analysis. They are timing verification and calculation of the order of pulse arrival times of the inputs of a gate in a time frame. The proposed verification method abstracts the behavior of an SFQ circuit along the order of pulse arrival times. The abstracted behavior is similar to that of a synchronous sequential circuit and can be verified by formal verification tools for synchronous sequential circuits.

The rest of the paper is organized as follows. In Sect. 2, we describe SFQ gates and circuits. In Sect. 3, we propose delay-based time frame model. In Sect. 4, we propose a verification method using the delay-based time frame model. In Sect. 5, we show experimental results. In Sect. 6, we conclude this paper.

2. SFQ Gates and Circuits

An SFQ logic gate consists of some superconducting loops with Josephson junctions and inductances. When a voltage pulse arrives at a loop, a flux quantum, i.e. an SFQ, can be trapped or released. When an SFQ is released, a short voltage pulse is produced. A voltage pulse and an SFQ are used as a carrier of information and for state representation, respectively.

Cell-based design is adopted in designing SFQ circuits [5]. Timing parameters, i.e. delay and timing constraints, and behavior of each gate are defined in a cell library for SFQ circuits. Details of the timing constraints will be described in Sect. 4.2.2. While an SFQ circuit is working, bias current is supplied to all gates in the circuit. The timing parameters depend on the amount of bias current to a gate. A gate with clock supply, which is called a clocked gate, is commonly used in SFQ circuits. There are clocked AND, OR, EXOR, NOT, etc. A delay flip-flop (DFF) is also a clocked gate. Some gates do not have clock supply.

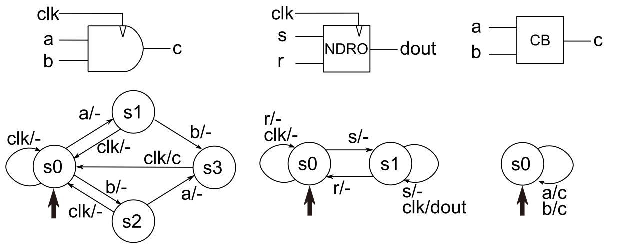

Fig. 1 PTFSMs of a clocked AND, an NDRO and a CB.

The behavior of a gate is represented by a finite state machine. We call it a *pulse-transferring finite state machine (PTFSM)*. The PTFSM of gate g is represented with $M = (S, I, O, \delta, \lambda, q)$. M is composed of a set of states S, a set of pulse arrivals at inputs I, a set of pulse arrivals at outputs O, the transition function $\delta : S \times I \rightarrow S$, the output function $\lambda : S \times I \rightarrow O$ and an initial state q. We suppose that $se \in S$ represents an error state and $- \in O$ represents no pulse arrivals at all outputs. If a pulse arrival at input i on state s is forbidden for the gate, $\delta(i, s)$ and $\sigma(i, s)$ is se and $-$, respectively. We call such an input a *forbidden input*. For example, M of a clocked AND gate includes $S = \{s0, s1, s2, s3, se\}, I = \{a, b, clk\}, O = \{c, -\}$, $\delta(s3, clk) = s0$, $\delta(s1, a) = se$, $\lambda(s3, clk) = c$, $\lambda(s1, a) = -$, and $q = s0$. Figure 1 shows the PTFSMs of a clocked AND, a Non-Destructive Read Out (NDRO) and a Confluence Buffer (CB). In this figure, transition and output functions for forbidden inputs of the clocked AND gate are omitted. NDRO and CB gates do not have forbidden inputs.

Clocking and timing designs are important factors affecting the performance of an SFQ circuit because gates switch with high speed and wiring delay is not negligible. Zero skew clocking, which is commonly used in CMOS circuits, is not used in most SFQ circuits. To achieve multi-gigahertz circuit, flow-clocking is often chosen [13]. In flow-clocking, a skewed clock pulse is distributed to clocked gates along data path and an interval between a clock pulse and a data pulse is shorter than that in zero skew clocking. Several flow-clocking schemes are used in SFQ circuits. One scheme is called concurrent-flow clocking. In the concurrent-flow clocking scheme, a data pulse arrives after the corresponding clock pulse and a gate behaves like as a logic gate combined with a DFF. There is another flow-clocking scheme, called clock-follow-data clocking. In the clock-follow-data clocking scheme, a data pulse arrives before the corresponding clock pulse and a gate behaves like as a gate in a combinational circuit. Concurrent-flow clocking and clock-follow-data clocking have different order of pulse arrival times and cause different behavior of a gate.

In most SFQ circuits, a synchronizing clock is used for

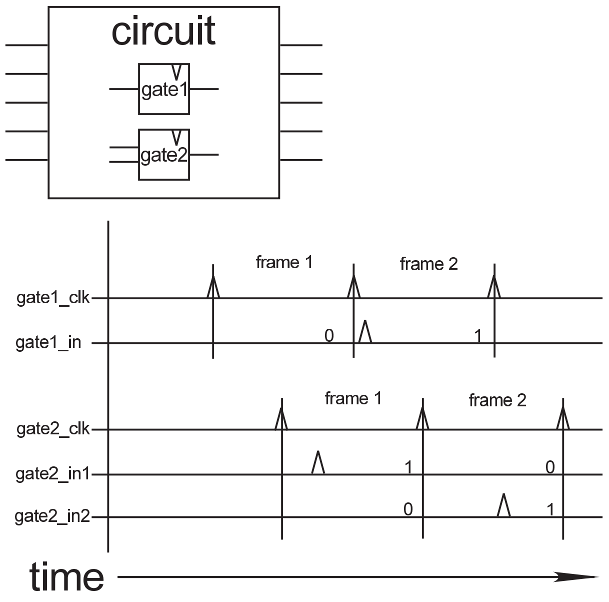

Fig. 2 Time frames and logic values of gates with clock supply.

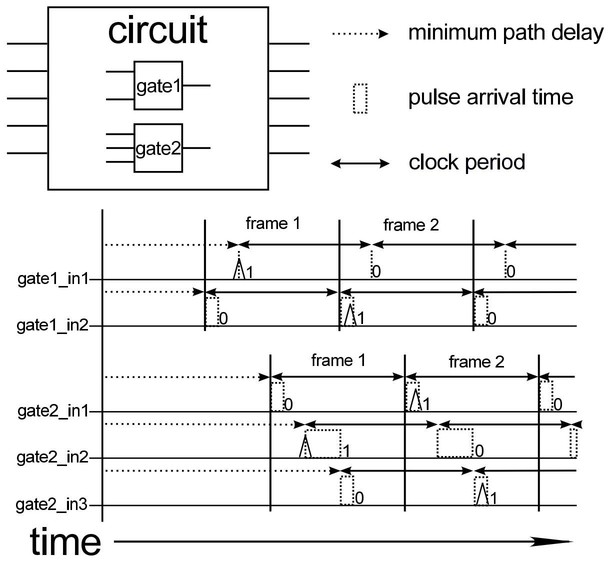

Fig. 3 Time frames and logic values of gates in the delay-based time frame model.

the definition of time frames and the representation of logic values. Figure 2 shows time frames and logic values of the gates with clock supply in an SFQ circuit. In this figure, a synchronizing clock is skewed on gate1 and gate2. A time frame for each gate is defined as the time section divided by clock pulses. The logic values “1” and “0” at a time frame are represented by the presence and the absence of a pulse in the time frame, respectively.

There often exists an SFQ circuit using logic gates without clock supply. Typical examples of such gates are CB and NDRO. A CB has no clock input and is used as an asynchronous OR gate. Although an NDRO has a clock input *clk*, the clock input sometimes connects to a data signal line rather than a clock signal line [3], [4], [14]. For these gates, the order of pulse arrival times also affects behavior of the gate.

3. Delay-Based Time Frame Model

The existing time frame model defines time frames, which are also called clock cycles, as time sections divided by synchronizing clock pulses. It is not suited to an SFQ circuit using gates without clock supply because of clock skew and partial lack of clock supply. If the time frames are not defined for a gate, the representation of logic values is also not defined for the gate.

We propose delay-based time frame model to define time frames for SFQ circuits composed of gates with and without clock supply and/or employing various clocking schemes. The model assumes a unique clock period of a circuit and the periodicity of pulse arrival time. The unique clock period decides breadth of all time frames. The periodicity means that pulse arrival time on a line appears periodically and is skewed along path delay from primary inputs to the line. In addition, the model assumes that there do not exist multi-cycle paths nor multiple pulses on a line in a time frame. In the model, a clock signal line is treated without distinction from the other data signal lines.

For the sake of simplicity, the following discussion of the time frame model assumes that pulses on primary inputs, if they exist, arrive at the same time. However, the time frame model can be easily applied to a circuit not based on the assumption by giving pulse arrival times to primary inputs. In the model, a time frame is skewed on each gate. The amount of the skew of the first time frame of a gate is the minimum path delay from primary inputs to inputs of the gate. The time frame on a gate proceeds periodically to the next time frame when the unique clock period of the circuit elapses. Therefore, the amount of skew of time frames on a gate is the same and equal to the amount of the skew of the first time frame on the gate. By this definition, the amount of the skew of all time frames on all gates with and without clock supply in a circuit can be decided.

In the model, the logic values “1” and “0” at a time frame are represented by the presence and the absence of a pulse in the frame, respectively. On a clock signal line, one pulse appears in every time frame. Therefore, a clock signal line is dealt with as a data signal line with constant value “1”.

Figure 3 shows the time frames and the representation of logic values in the delay-based time frame model. The amount of the skew on a gate is the minimum path delay from primary inputs to inputs of the gate. Clock inputs of all gates are not distinguished from data inputs unlike those in Fig. 2. In other words, inputs of a gate include data inputs and a clock input. A pulse arrival time on a line is represented by a dotted line box and appears periodically because of the periodicity of pulse arrival time. Left and right edges of the box represent the earliest and the latest pulse arrival times, respectively. Skewed time frames on a gate also appear periodically. Note that the starting point of each time frame on a gate equals to the earliest pulse arrival time of all inputs of the gate. For example, time frames of gate1 in Fig. 3 start at the earliest pulse arrival time of gate1_in2. When a line has the logic value "1" at a time frame, a pulse exists on the line in the time frame.

This model does not assume multiple pulses on a line in a time frame nor a large difference between the earliest and the latest pulse arrival times on a line. In some cases, we need to treat them. For example, if a circuit has multiple clock signals and timing of each input of a CB gate depends on that of the different clock signal, a difference between the pulse arrival times of the output of the CB will be large. To treat it, we duplicate the output line. Each of the duplicated lines represents a path from the different input line to the same output line. By the duplication of the output line, the multiple pulses on a line can also be treated in this model. It is important to duplicate a small part of lines in a circuit because the duplication complicates the circuit. Therefore, if the duplication is necessary for an output of a gate, which is a CB gate in particular, the gate should be flagged or renamed.

4. Verification Method Using the Delay-Based Time Frame Model

4.1 Overview

We propose a verification method using the delay-based time frame model. The proposed method detects errors of an SFQ circuit using static timing analysis and formal verification. The verification flow is shown in Fig. 4.

The static timing analysis calculates the path delay, the timing slacks, the minimum clock period and the order of pulse arrival times. To use formal verification tools such as model checker and equivalence checker, PTFSMs of all gate in the SFQ circuit are abstracted. The proposed method detects two types of errors. First, the occurrence of the forbidden input is detected by model checking. We call it *forbidden input checking*. Second, a discrepancy between the implementation of a circuit and the specification for the circuit is detected by equivalence checking.

A negative timing slack is usually dealt with as a timing violation. However, if a negative timing slack is calculated from a pair of inputs whose pulses do not appear simultaneously in a time frame, a timing violation owing to the negative timing slack does not occur. In forbidden input checking, the negative timing slack is checked whether it leads to the timing violation. Thus, not only formal verification but also static timing analysis includes forbidden input checking. The abstracted behavior is similar to that of a synchronous sequential circuit. Therefore, verification tools for synchronous sequential circuits can be used for the forbidden input checking and the equivalence checking. If the forbidden input checking detects an error of a gate, the gate can have a forbidden input which is caused by a negative timing slack or an unexpected input stimulus of the

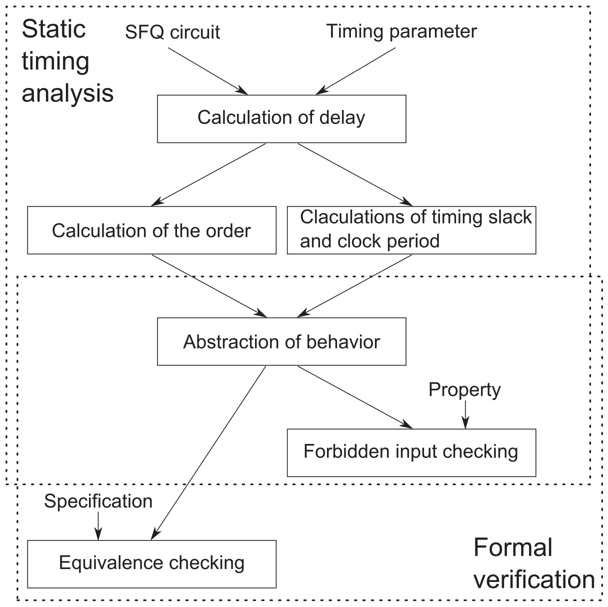

Fig. 4 The proposed verification flow.

gate. Then, we should change the timing or the connection of inputs of the gate. If the equivalence checking detects the error, there exists a discrepancy between the implementation and the specification. Then, we should change the implementation or the specification.

4.2 Static Timing Analysis

4.2.1 Calculation of Delay

Calculation of the path delay of an SFQ circuit is slightly different from that of a CMOS circuit. In an SFQ circuit, memory elements such as delay flip-flop are not distinguished from other logic gates, because all logic gates have memory functions. Timing parameters such as delay and timing constraint of a gate are affected by bias current, timing jitter and fabrication variations.

In this paper, we refer to an input pin of a gate as an input of a gate. Similarly, an output pin of a gate is referred to as an output of a gate. Timing dependency between an input and an output of each gate is described in the cell library. When a pulse on an output of a gate can be produced by a pulse arrival on an input of the gate, pulse arrival time of the output depends on that of the input. A pulse arrival time of an output often depends on those of several inputs.

The earliest pulse arrival time to output o, $T_e(o)$, and the latest pulse arrival time to o, $T_l(o)$, are calculated by the followings,

$$T_l(o) = \max_{i \in ID(o)} (T_l(i) + D(i, o)), \quad (1)$$

$$T_e(o) = \min_{i \in ID(o)} (T_e(i) + D(i, o)), \quad (2)$$

where the set of inputs whose pulse arrival times affect that of o is $ID(o)$ and the delay from i to o is $D(i, o)$. The pulse

Algorithm 1: Calculation of delay

```
   Input: an SFQ circuit
   Output: pulse arrival times of primary outputs and inputs of all
           gates
   // $FO(o)$ is the set of inputs which are fan-outs
      of output $o$
   // $ID(o)$ is the set of inputs whose pulse arrival
      times affect that of output $o$
   // $OD(i)$ is the set of outputs whose pulse arrival
      times depend on that of input $i$
1  $I \leftarrow$ the set of inputs connecting to primary inputs;
2  while $I \neq \emptyset$ do
3      Pick an input $i \in I$;
4      $I \leftarrow I - \{i\}$;
5      foreach $od \in OD(i)$ do
6          if $\forall id \in ID(od), T_e(id)$ and $T_l(id)$ are calculated then
7              Calculate $T_e(od)$ and $T_l(od)$ by Eq. 1 and Eq. 2;
               foreach $fo \in FO(od)$ do
8                  Calculate $T_e(fo)$ and $T_l(fo)$;
9              $I \leftarrow I \cup FO(od)$;
```

arrival times to input i, $T_e(i)$ and $T_l(i)$ are calculated by the followings,

$$T_l(i) = (T_l(fi) + D(fi, i)), \quad (3)$$

$$T_e(i) = (T_e(fi) + D(fi, i)), \quad (4)$$

where the fanin of i is fi which is an output of a gate.

Algorithm 1 shows the calculation process of the path delay in an SFQ circuit. This algorithm assumes that the arrival time of an output of a gate in the circuit does not depend on itself. If there exists such output, pulse arrival time of it cannot be defined.

4.2.2 Calculation of Timing Slack and Clock Period

In general, timing constraints of an SFQ gate are given as setup and hold time constraints [15]. The setup and hold time constraints represent the relationship between a data input and a clock input of a gate. The concurrent-flow clocking and the clock-follow-data clocking have different definitions of setup and hold constraints. If an SFQ circuit employs multiple clocking schemes with the different definitions, the clocking schemes need to be declared on each gate to use setup and hold constraints. In addition, timing constraints of an SFQ gate may also exist between two data inputs. For example, data inputs s and r of an NDRO gate have an interval time constraint. If pulses on the inputs arrive closely within the minimum interval time defined in the constraint, the gate can perform incorrectly. On the other hand, inputs a and b of a CB gate have another interval time constraint. When a pulse on each of the inputs arrives closely within the maximum interval time defined in the constraint, only one pulse appears on an output c of the CB. When these two pulses are farther than the maximum interval time defined in the constraint, two pulses appear on the output c. Because the presence of two pulses on a line in a time frame is not assumed in the model, a large interval time between a and b of CB can lead to a timing violation.

We classify these interval time constraints of a gate into two types. They are the minimum and the maximum interval time constraints. Each ordered pair of inputs of a gate has one of these constraints or does not have an interval time constraint. We consider an ordered pair of inputs (x, y) such that a pulse of y arrives after a pulse arrival at x. When (x, y) has the minimum interval time constraint, a pulse on y should arrive after the elapse of *the minimum interval time* from a pulse arrival time of x at the earliest. When (x, y) has the maximum interval time constraint, a pulse on y should arrive by the elapse of *the maximum interval time* from a pulse arrival time of x at the latest. The interval time constraints do not depend on clocking schemes. Therefore, the declaration of a clocking scheme on each gate is not necessary in the proposed verification method.

For an ordered pair of inputs (x, y) of gate g, the timing slack $TS(g, x, y)$ is defined in three cases (a), (b) and (c): (a) $T_l(y)$ is larger than $T_e(x)$ and (x, y) has the minimum interval time constraint (b) $T_l(y)$ is larger than $T_e(x)$ and (x, y) has the maximum interval time constraint (c) otherwise, namely, $T_l(y)$ is not larger than $T_e(x)$ or (x, y) does not have the minimum or the maximum interval time constraint. Figure 5(a) shows the timing slack calculated by the minimum interval time constraint. Figure 5(b) shows the timing slack calculated by the maximum interval time constraint. $TS(g, x, y)$ is calculated by the following,

$$TS(g, x, y) = \begin{cases} T_e(y) - T_l(x) - IT_{min}(x, y) & \text{(a)} \\ T_e(x) + IT_{max}(x, y) - T_l(y) & \text{(b)} \\ \text{no value} & \text{(c)} \end{cases} \quad (5)$$

where the minimum and the maximum interval times of (x, y) are $IT_{min}(x, y)$ and $IT_{max}(x, y)$, respectively. If $TS(g, x, y)$ or $TS(g, y, x)$ is negative, a timing violation may occur between x and y.

A clocked AND gate has the minimum interval time constraints on (a, clk), (clk, a), (b, clk), (clk, b), (a, a), (b, b) and (clk, clk). This gate does not have the interval time constraints on (a, b) and (b, a). A CB gate has the minimum interval time constraints on (a, a) and (b, b) and the maximum interval time constraints on (a, b) and (b, a).

The calculation of timing slack can produce a negative timing slack not causing a timing violation. For example, when a CB gate has a negative timing slack on (a, b) and pulses do not arrive at both of a and b in a time frame, the negative timing slack can be ignored. It is checked whether a negative timing slack causes a timing violation by forbidden input checking in Sect. 4.3.2.

To guarantee the interval time constraints between inputs in a time frame and the next time frame, we calculate the minimum clock period. Figure 5 shows the minimum clock period of each gate. The minimum clock period of an ordered pair of inputs (x, y) of gate g, $CP_{min}(g, x, y)$, and the minimum clock period of gate g, $CP_{min}(g)$, are calculated by the followings,

$$CP_{min}(g, x, y) = \{T_l(y) - T_e(x) + IT(y, x)\}, \quad (6)$$

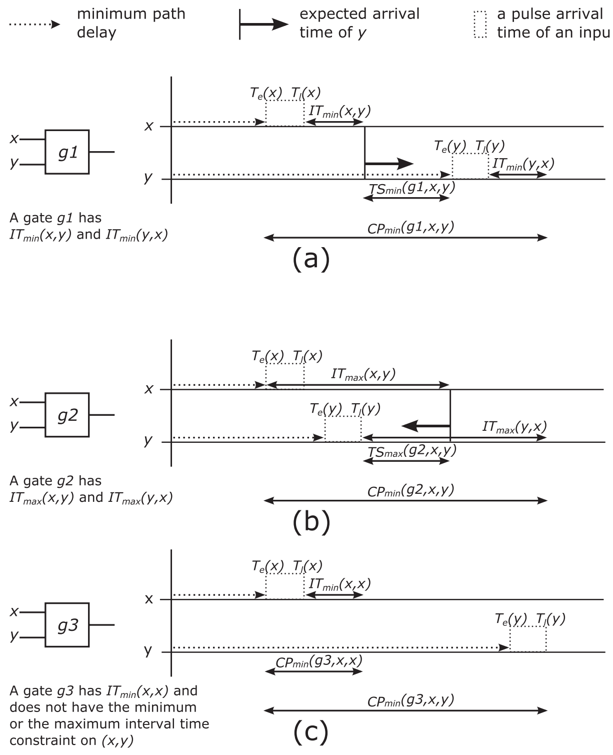
minimum path delay
expected arrival time of y
a pulse arrival time of an inpu
A gate g1 has $IT_{min}(x,y)$ and $IT_{min}(y,x)$
$T_e(x)$ $T_l(x)$
$IT_{min}(x,y)$
$T_e(y)$ $T_l(y)$
$IT_{min}(y,x)$
$TS_{min}(g1,x,y)$
$CP_{min}(g1,x,y)$
(a)
A gate g2 has $IT_{max}(x,y)$ and $IT_{max}(y,x)$
$IT_{max}(x,y)$
$IT_{max}(y,x)$
$TS_{max}(g2,x,y)$
$CP_{min}(g2,x,y)$
(b)
A gate g3 has $IT_{min}(x,x)$ and does not have the minimum or the maximum interval time constraint on (x,y)
$IT_{min}(x,x)$
$CP_{min}(g3,x,x)$
$CP_{min}(g3,x,y)$
(c)

Fig. 5 Calculated timing slack and clock period (a) $TS_{min}(g1,x,y) = T_e(y) - T_l(x) - IT_{min}(x,y)$ and $CP_{min}(g1,x,y) = T_l(y) - T_e(x) + IT_{min}(y,x)$ (b) $TS_{max}(g2,x,y) = T_e(x) + IT_{max}(x,y) - T_l(y)$ and $CP_{min}(g2,x,y) = T_l(y) - T_e(x) + IT_{max}(y,x)$ (c) $CP_{min}(g3,x,x) = T_l(x) - T_e(x) + IT_{min}(x,x)$ and $CP_{min}(g3,x,y) = T_l(y) - T_e(x)$.

$$CP_{min}(g) = \max_{x,y \in I(g)} (CP(g,x,y)), \tag{7}$$

where $I(g)$ is the set of inputs of gate g and $IT(y,x)$ is $IT_{min}(y,x)$ in Fig. 5(a), $IT_{max}(y,x)$ in Fig. 5(b) or 0 in Fig. 5(c). When an interval time constraint does not exist on (y,x), $IT(y,x)$ is 0. $IT(y,x)$ includes $IT_{max}(y,x)$, because $IT_{max}(y,x)$ is the interval time constraint in a time frame. If the clock period is smaller than $T_l(y) - T_e(x) + IT_{max}(y,x)$, a pulse on x in a time frame is dealt with as a pulse in the previous frame. The minimum clock period of the circuit is

$$CP_{min} = \max_{\forall g}(CP_{min}(g)). \tag{8}$$

4.2.3 Calculation of the Order

To define the behavior of a gate in a time frame, the order of pulse arrival times of the inputs is necessary. A gate can have the different behavior under the different order. A simple example of the different behavior of a gate owing to the different order is a clocked gate employing concurrent-flow clocking or clock-follow-data clocking. Even if these gates are the same type, the different clockings cause the different order and the different behavior of the gates.

In an SFQ circuit, the order of pulse arrival times of inputs of a gate is often multiple. For example, when a clocked AND gate holds on timing relations of $T_e(clk) < T_l(a)$, $T_e(clk) < T_l(b)$, $T_e(a) < T_l(b)$ and $T_e(b) < T_l(a)$, the order of pulse arrival times can be (clk, a, b) or (clk, b, a).

Multiple order can result in multiple behavior. Although the proposed verification method uses the one of the order, which is the order of the earliest pulse arrival times of the inputs. We consider a pair of inputs leading to multiple order $\{x, y\}$ of a gate g. In the case where $TS(g,x,y)$ or $TS(g,y,x)$ is negative, pulse arrivals at both of $\{x,y\}$ in a time frame is dealt as a forbidden input in the following abstraction of the behavior of a gate. In the other case where both of $TS(g,x,y)$ and $TS(g,y,x)$ are not negative, no negative timing slacks and the multiple order on $\{x,y\}$ represent that multiple order must result in the same behavior on all SFQ gates. An example of the case is $\{a,b\}$ of a clocked AND gate g. $TS(g,a,b)$ and $TS(g,b,a)$ cannot be negative because there do not exist interval time constraints between a and b. Both of pulse arrival sequences (a,b) and (b,a) on any state in a time frame result in the same behavior of the gate. In both of the cases, the order results in the same behavior in the following formal verification. Therefore, we use the order of the earliest pulse arrival times as the order.

The order of the earliest pulse arrival times of inputs of gate g, $OI(g)$ is defined by the following,

$$OI(g) = \left(i_1, i_2, \cdots, i_n \mid \bigwedge_{1 \le j \le n-1} T_e(i_j) \le T_e(i_{j+1}) \right) \tag{9}$$

where n is the number of inputs of g and $i_1, i_2, \cdots, i_n$ are the inputs.

4.3 Formal Verification

4.3.1 Abstraction of the Behavior of a Gate and a Circuit

We discretize the temporal behavior of a PTFSM using delay-based time frame model. Using the PTFSM, it is hard to describe a specification and to accomplish formal verification, because the PTFSM is driven by a pulse arrival which is independent of a time frame. In the proposed verification method, we abstract the behavior represented by the PTFSM to that represented by a *synchronous finite state machine (SFSM)*. An SFSM of a gate is defined by the followings.

- The next state is a state at the next time frame
- The input alphabet is a set of bit vectors of all input lines of the gate
- The output alphabet is a set of bit vectors of all output lines of the gate
- An input and an output of an SFSM appear once in every time frame

Abstraction of a PTFSM uses the order of the earliest pulse arrival times of inputs of the gate. An input bit vector of an SFSM with the order corresponds to a pulse arrival sequence. Thus, the output bit vector and the next state from an input vector at a present state of an SFSM can be decided by the corresponding PTFSM. For example, we consider an

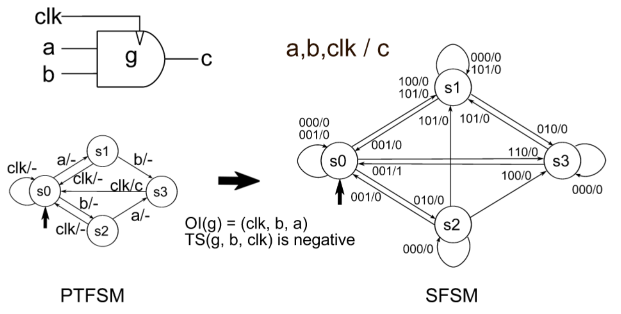

Fig. 6 Example of abstraction the PTFSM to the SFSM.

input bit vector 101 at a, b and clk on a present state $s3$ with order (clk, b, a) of a clocked AND gate g in Fig. 6. The input bit vector 101 corresponds to the pulse arrival sequence (clk, a). On the PTFSM of the clocked AND gate, when pulses arrive at clk and then a on state $s3$, the state results in $s1$ and a pulse is produced on c. Therefore, on the SFSM of g with order (clk, b, a), the next state and the output bit vector from the input bit vector 101 at a, b and clk on the present state $s3$ are $s2$ and 1 at c, respectively.

An SFSM has three types of forbidden input bit vectors which are an input bit vector with the values "1" on both of a pair of input lines whose timing slack is negative, an input bit vector including a forbidden input of PTFSM and an input bit vector producing more than one pulse on an output line in a time frame. For example, when a clocked AND gate g has a negative timing slack $TS(g, a, clk)$ or $TS(g, clk, a)$, input bit vectors 101 and 111 at (a, b, clk) on any state are forbidden. An input bit vector 100 at (a, b, clk) on state $s1$ of g is forbidden, because the input bit vector includes a pulse arrival at a on $s1$ which is forbidden on the PTFSM of g.

The SFSM of gate g, M' is abstracted from the PTFSM of g, $M = (S, I, O, \delta, \lambda, q)$. $M' = (S, I', O', \delta', \lambda', q)$ is composed of a set of states S, a set of input bit vectors I', a set of output bit vectors O', the transition function $\delta' : S \times I' \rightarrow S$, the output function $\lambda' : S \times I' \rightarrow O'$ and an initial state q.

I' is a set of all bit vectors at input lines of g. O' is a set of all bit vectors at output lines of g. When $OI(g)$ is decided, an input pulse arrival sequence on M, $(i_0, i_1, \ldots, i_{m-1})$ is defined uniquely from an input bit vector $i' \in I'$ where m is the number of values "1" on i'. $\delta'(i', s \in S)$ and $\lambda'(i', s)$ are decided uniquely by $(i_0, i_1, \ldots, i_{m-1})$ on M corresponding to i' on M'. $\delta'(i', s)$ is the resulting state of $\delta(i_{m-1}, \ldots \delta(i_1, \delta(i_0, s)) \ldots)$. We consider a state sequence $(s_0, s_1 \ldots, s_m)$ and an output pulse arrival sequence $(o_0, o_1 \ldots, o_{m-1})$ where $s_0 = s$, $s_{j+1} = \delta(i_j, s_j)$ and $o_j = \lambda(i_j, s_j)$ for $0 \leq j \leq m-1$. Then, $\delta'(i', s)$ can also be represented as s_m. When $o \in O$ does not appear in $(o_0, o_1 \ldots, o_{m-1})$, an output bit vector $\lambda'(i', s)$ has value "0" at an output line of o. When o appears once in $(o_0, o_1 \ldots, o_{m-1})$, an output bit vector $\lambda'(i', s)$ has value "1" at an output line of o. When o appears more than one time in $(o_0, o_1 \ldots, o_{m-1})$, $\lambda'(i', s)$is the indefinite bit vector x. In addition, whenever (i', s) is a forbidden input of M', $\delta'(i', s)$ and $\lambda'(i', s)$ are dealt as the error state se and the indefinite bit vector x, respectively. If i' has values "1" on both of input lines a and b with negative $TS(g, a, b)$ or $TS(g, b, a)$, $\exists j \delta(i_j, s_j) = se$ or $\lambda'(i', s) = x$, (i', s') is a forbidden input.

For example, M' of a clocked AND gate includes $S = \{s0, s1, s2, s3, se\}$, $I' = \{000, 001, ..., 111, x\}$ at a, b and clk, $O' = \{0, 1, x\}$ at c, $\delta'(s0, 110) = s3, \lambda'(s3, 001) = 1, q = s0$. The next state and the output bit vector from the forbidden input bit vectors at the corresponding state, any input bit vector at state se or input bit vector x at any state are always se and x, respectively.

Figure 6 shows an example of abstraction the PTFSM to the SFSM of a clocked AND gate g with $OI(g) = (clk, b, a)$ and negative timing slack $TS(g, b, clk)$. Input bit vectors 111 and 011 at (a, b, clk) on any state are forbidden because $TS(g, b, clk)$ is negative. Other forbidden input bit vectors at (a, b, clk) are 100 and 110 on $s1$, 010 and 110 on $s2$ and 010, 100 and 110 on $s3$ because the forbidden input of the PTFSM.

The product machine of SFSMs of all gates in the circuit corresponds to the SFSM of the circuit. By abstracting the behavior of all gates in the circuit, the SFSM of the circuit are given.

4.3.2 Forbidden Input Checking and Equivalence Checking

When the SFSM of an SFQ circuit is abstracted from the PTFSM of all gates in the circuit, the following verification can be accomplished in a similar way to synchronous sequential circuits. Model checking and equivalence checking, which are called formal verification, are used for verification of synchronous sequential circuit. Some tools for the formal verification are developed [16]–[18]. We can use them for verification of an SFQ circuit.

In the proposed verification method, a forbidden input of an SFSM is checked by the forbidden input checking. The forbidden input checking is accomplished by using properties representing no forbidden input and checking whether the properties are satisfied or not by model checking. When a negative timing slack exists on a gate and forbidden input checking is passed without errors on the gate, the negative timing slacks can be ignored because negative timing slacks also effect the SFSM. In the equivalence checking, the specification corresponding to behavior of the SFQ circuit needs to be prepared. The specification and the abstracted SFSM of the SFQ circuit are compared to detect an error of the design.

5. Experimental Results

We have implemented the proposed static timing analysis tool in SKILL language and the abstraction tool of behavior in SKILL and C++ languages. The implemented static timing analysis tool obtains gate delay and timing constraints from the cell library and calculates the path delay, timing slacks including the minimum timing slack, the minimum clock period and the order of pulse arrival times at each gate

Table 1 Verification results.

circuit	#gate	#JJ	#state	#NTS	min TS (ps)	min CP (ps)	STA time (s)	FC time (s)	EC time (s)
full adder	9	292	5832	0	3.6	23.5	0.3	< 0.01	< 0.01
8 bit CLA	158	6380	3.09×10^{86}	9	−5.3	33.0	3.27	12.3	0.07
4-bit SS	190	3829	1.10×10^{81}	50	−3.9	42.4	1.87	27.0	0.06

of an SFQ circuit. The abstraction tool translates PTFSM of an SFQ circuit to the SFSM described in BLIF-MV which is used for verification of synchronous sequential circuit. Simultaneously, the tool produces properties which represent no occurrence of forbidden inputs in LTL language.

We verified several circuits using these tools. The SFSM of the circuits was verified by forbidden input checking and equivalence checking using vis [17] and ABC [18], respectively. The forbidden input checking uses the properties produced by the abstraction tool. We verified three circuits, a full adder using NDROs and CBs without clock supply, an 8-bit carry look-ahead adder employing concurrent-flow clocking (8-bit CLA) [6] and a 4-bit slice 32 bit shifter using gates with and without clock supply (4-bit SS).

The experiments were conducted on a Linux platform (Debian 6.0) with an Intel Xeon X5470 (3.33 GHz) and 32 GB of RAM. Table 1 shows the number of logic gates (#gate), the number of Josephson junctions (#JJ), the number of states (#state), the number of negative timing slacks (#NTS), the minimum timing slack (min TS), the minimum clock period (min CP), CPU time for static timing analysis (STA time), CPU time for forbidden input checking (FC time) and CPU time for equivalence checking (EC time). The forbidden input checking was accomplished by unbounded model checking. The equivalence checking was accomplished by unbounded sequential equivalence checking.

In the forbidden input checking of the 8-bit CLA, we detected a failed property caused by a connection error on a clocked AND gate. The forbidden input of the gate, which is pulse arrival at a on $s1$ specifically, could occur because of this error. This error was also detected in the equivalence checking.

50 negative timing slacks were detected by the static timing analysis of the 4-bit SS. They were 29 in CBs, 8 in RDFFs (resettable DFFs) and 13 in D2FF (DFFs with two clock inputs and two corresponding outputs). The negative timing slacks in RDFFs were caused by predecessors of CBs and the output line of the CB had two pulses in a time frame. In order to avoid the two pulses on a line in a time frame, we duplicate lines from CBs to RDFFs as shown in Fig. 7. All gates in the redesigned version of the 4-bit SS passed through forbidden input checking and the negative timing slacks on D2FFs and CBs could be ignored because pulses at two inputs causing the negative timing slacks did not appear simultaneously in a time frame.

The experimental results show that the proposed verification method can handle circuits employing various clocking and composed of gates with and without clock supply.

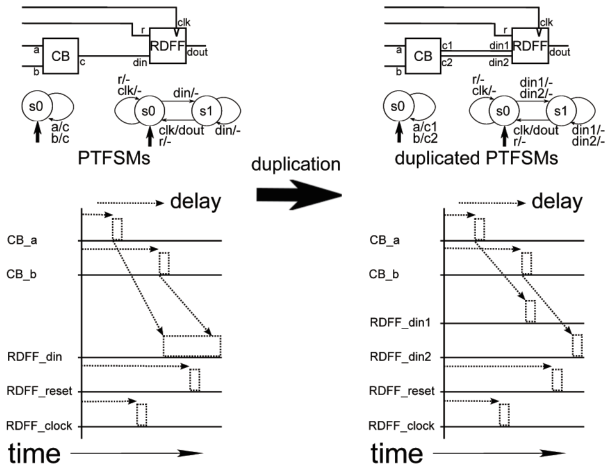

Fig. 7 Duplication of a line from CB to RDFF in 4-bit SS.

6. Conclusion

We have proposed a verification method of SFQ circuits. To accomplish verification, we introduced a new time frame model and definition of timing constraints to cope properly with various clocking schemes and gates with and without clock supply. The proposed method abstracts the behavior of an SFQ circuit to a form similar to a synchronous sequential circuit. The abstracted behavior of an SFQ circuit can be verified by an existing formal verification tool for synchronous sequential circuits. When the formal verification is applied to SFQ circuits using our method, quality of verification will be increased as compared to the verification using only logic simulation.

We have shown the verification results for several SFQ circuits. The results show that our method can be applied to practical SFQ circuits. Other SFQ circuits such as a dual-rail circuit can be verified by the proposed method as long as the number of pulses on a line in a time frame is not more than one.

Acknowledgments

The authors thank Prof. M. Tanaka of Nagoya University and Prof. K. Hamaguchi of Shimane University. This work was supported in part by CREST JST, ALCA JST and VLSI Design and Education Center (VDEC), the University of Tokyo in Collaboration with Cadence Design Systems, Inc.

References

[1] K.K. Likharev and V.K. Semenov, “RSFQ logic/memory family:

A new Josephson-junction technology for sub-terahertz-clock-frequency digital systems," IEEE Trans. Appl. Supercond., vol.1, no.1, pp.3–28, 1991.

[2] Y. Yamanashi, M. Tanaka, A. Akimoto, H. Park, Y. Kamiya, N. Irie, N. Yoshikawa, A. Fujimaki, H. Terai, and Y. Hashimoto, "Design and Implementation of a Pipelined Bit-Serial SFQ Microprocessor, CORE1β," IEEE Trans. Appl. Supercond., vol.17, no.2, pp.474–477, 2007.

[3] H. Park, Y. Yamanashi, K. Taketomi, N. Yoshikawa, M. Tanaka, K. Obata, Y. Ito, A. Fujimaki, N. Takagi, K. Takagi, and S. Nagasawa, "Design and implementation and on-chip high-speed test of SFQ half-precision floating-point adders," IEEE Trans. Appl. Supercond., vol.19, no.3, pp.634–639, 2009.

[4] H. Hara, K. Obata, H. Park, Y. Yamanashi, K. Taketomi, N. Yoshikawa, M. Tanaka, A. Fujimaki, N. Takagi, K. Takagi, and S. Nagasawa, "Design, implementation and on-chip high-speed test of SFQ half-precision floating-point multiplier," IEEE Trans. Appl. Supercond., vol.19, no.3, pp.657–660, 2009.

[5] S. Yorozu, Y. Kameda, H. Terai, A. Fujimaki, T. Yamada, and S. Tahara, "A single flux quantum standard logic cell library," Physica C: Superconductivity, vol.378-381, pp.1471–1474, 2002.

[6] K. Takagi, Y. Ito, S. Takeshima, M. Tanaka, and N. Takagi, "Layout-driven skewed clock tree synthesis for superconducting SFQ circuits," IEICE Trans. Electron., vol.E94-C, no.3, pp.288–295, 2011.

[7] Y. Kameda and S. Yorozu, "Automatic Josephson-transmission-line routing for single-flux-quantum cell-based logic circuits," IEEE Trans. Appl. Supercond., vol.13, no.2, pp.519–522, 2003.

[8] M. Tanaka, K. Obata, Y. Ito, S. Takeshima, M. Sato, K. Takagi, N. Takagi, H. Akaike, and A. Fujimaki, "Automated passive-transmission-line routing tool for single-flux-quantum circuits based on A* algorithm," IEICE Trans. Electron., vol.E93-C, no.4, pp.435–439, 2010.

[9] F. Matsuzaki, N. Yoshikawa, M. Tanaka, A. Fujimaki, and Y. Takai, "A behavioral-level HDL description of SFQ logic circuits for quantitative performance analysis of large-scale SFQ digital systems," Physica C: Superconductivity, vol.392-396, pp.1495–1500, 2003.

[10] K. Takagi, N. Kito, and N. Takagi, "Circuit description and design flow of superconducting SFQ logic circuits," IEICE Trans. Electron., vol.E97-C, no.3, pp.149–156, 2014.

[11] Y. Kameda, S. Yorozu, Y. Hashimoto, H. Terai, A. Fujimaki, and N. Yoshikawa, "High-speed demonstration of single-flux-quantum cross-bar switch up to 50 GHz," IEEE Trans. Appl. Supercond., vol.15, no.1, pp.6–10, 2005.

[12] K. Takagi, M. Sato, M. Tanaka, and N. Takagi, "A verification method of pipeline processing behavior of superconducting singl-flux-quantum pulse logic circuits," 16th Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI2010), R2-17, pp.208–231, 2011.

[13] O.A. Mukhanov, P.D. Bradley, S.B. Kaplan, S.V. Rylov, and A. Kirichenko, "Design and operation of RSFQ circuits for digital signal processing," Proc. 5th Int. Supercond. Electron. Conf, pp.27–30, 1995.

[14] M. Tanaka, H. Akaike, A. Fujimaki, Y. Yamanashi, N. Yoshikawa, S. Nagasawa, K. Takagi, and N. Takagi, "100-GHz single-flux-quantum bit-serial adder based on 10-kA/cm^2 niobium process," IEEE Trans. Appl. Supercond., vol.21, no.3, pp.792–796, 2011.

[15] K. Gaj, E.G. Friedman, and M.J. Feldman, "Timing of multi-gigahertz rapid single flux quantum digital circuits," in High Performance Clock Distribution Networks, pp.135–164, Springer, 1997.

[16] A. Aziz, F. Balarin, S.-T. Cheng, R. Hojati, T. Kam, S.C. Krishnan, R.K. Ranjan, T.R. Shiple, V. Singhal, S. Tasiran, H.-Y. Wang, R.K. Brayton, and A.L. Sangiovanni-Vincentelli, "HSIS: A BDD-based environment for formal verification," Proc. 31st Annual Conference on Design Automation Conference. DAC'94, pp.454–459, 1994.

[17] R.K. Brayton, G.D. Hachtel, A. Sangiovanni-Vincentelli, F. Somenzi, A. Aziz, S.-T. Cheng, S. Edwards, S. Khatri, Y. Kukimoto, A. Pardo, S. Qadeer, R.K. Ranjan, S. Sarwary, T.R. Staple, G. Swamy, and T. Villa, "VIS: A system for verification and synthesis," Computer Aided Verification, Lecture Notes in Computer Science, vol.1102, pp.428–432, 1996.

[18] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," Computer Aided Verification, Lecture Notes in Computer Science, vol.6174, pp.24–40, 2010.

Takahiro Kawaguchi received the B.E., and M.I.S. degrees in information engineering from Nagoya University, Nagoya, Japan, in 2010 and 2012, respectively. He is currently working toward Dr. degree at Kyoto University. His current research interests include algorithms for computer-aided design of SFQ integrated circuits.

Kazuyoshi Takagi received the B.E., M.E. and Dr. of Engineering degrees in information science from Kyoto University, Kyoto, Japan, in 1991, 1993 and 1999 respectively. From 1995 to 1999, he was a Research Associate at Nara Institute of Science and Technology. He had been Assistant Professor since 1999 and promoted to an Associate Professor in 2006, at the Department of Information Engineering, Nagoya University, Nagoya, Japan. He moved to Department of Communications and Computer Engineering, Kyoto University in 2011. His current interests include system LSI design and design algorithm.

Naofumi Takagi received the B.E., M.E. and Ph.D. degrees in information science from Kyoto University, Kyoto, Japan, in 1981, 1983 and 1988 respectively. He joined Kyoto University as an instructor in 1984 and was promoted to an associate professor in 1991. He moved to Nagoya University, Nagoya, Japan, in 1994, and promoted to a professor in 1998. He returned to Kyoto University in 2010. His current interests include computer arithmetic, hardware algorithms, and logic design. He received Japan IBM science Award and Sakai Memorial Award of the Information Processing Society of Japan in 1995, and The Commendation for Science and Technology by the Minister of Education, Culture, Sports, Science and Technology of Japan in 2005.