Timing-aware description methods and gate-level simulation of single flux quantum logic circuits

Nobutaka Kito Kazuyoshi Takagi Naofumi Takagi

Graduate School of Informatics
Kyoto University
Kyoto, 606–8501, Japan
{nkito, ktakagi, ntakagi}@lab3.kuis.kyoto-u.ac.jp

Abstract— Single Flux Quantum (SFQ) circuits are high-speed and ultra low-power circuits using superconductive devices. In SFQ circuits, skew of signals is not negligible and basic gates (such as AND and OR) are clocked because SFQ circuits are very fast and use pulse logic. Therefore, SFQ circuits of the same topology may have different functions depending on skew of signals. Thus, we need to be aware of timing issues for SFQ circuit design. We propose two timing-aware description methods for SFQ circuits to describe function of circuits precisely. One is a circuit schematic with a note about the order of pulse arrivals. The other is a timing-aware circuit description language. As an example application of the description language, we show a logic simulation algorithm for SFQ logic circuits.

I. Introduction

Single Flux Quantum (SFQ) circuits[1] are high-speed and ultra low-power circuits using superconductive devices. Technologies for supercomputers using SFQ circuits have been studied intensively [2, 3, 4, 5]. SFQ circuits use pulse logic. In other words, “0” and “1” are represented by pulses. Gates in SFQ circuits are synchronized with a clock signal. An SFQ circuit works at high frequency up to several hundreds GHz [6, 7] and skew of signals is not negligible.

Because functions of gates in SFQ circuits vary with timing of pulse arrivals and SFQ circuits works at high frequency, designers of SFQ circuits adjust skew between gates carefully by inserting delay elements manually. In the existing design flow, precise pulse timing at each gate cannot be determined until the circuit layout is obtained. This means that logical behavior of a circuit cannot be described in a circuit schematic by itself. Logic description methods for SFQ circuits independent from fabrication processes and circuit layouts are necessary to design logic circuits efficiently.

In this paper, we propose two timing-aware description methods for SFQ logic circuits to describe the function of them. One description method is a circuit schematic with a note about the order of pulse arrivals. When we draw a schematic, for each gate, we attach a note about the order of pulse arrivals to clarify the function of the circuit. It is useful for designers to represent circuit schematic. The other description method is a timing-aware circuit description language. We describe the order of pulse arrivals for each gate in a description written in the language. It is designed for future design automation of SFQ logic circuits.

We can describe SFQ circuits clearly by the proposed language. Thus, we can simulate a circuit described by the language. We show a simulation algorithm for SFQ logic circuits as an example application. By the algorithm, we can simulate a circuit with a description written in the language.

As an example, we show a schematic of a 4-bit multiplier and a description written in the proposed language. We also show simulation result of the description.

This paper is organized as follows. In the next section, we briefly review timing design of SFQ logic circuits. In Section 2, we propose a schematic for SFQ logic circuits with a note about the order of pulse arrivals. In Section 3, we propose a description language for SFQ logic circuits considering the order of pulse arrivals. In section 4, we show a logic simulation algorithm for SFQ circuits. In Section 5, we show a schematic and a description of a 4-bit multiplier as an example. We show simulation result of the circuit. In Section 6, we conclude this paper.

II. Timing design of SFQ circuits

In an SFQ circuit, signals are transmitted using voltage pulses on wires. Each gate in a circuit has a clock input terminal and each gate is synchronized with clock pulses. As an example, we show a 2-input AND gate and its behavior in Figs. 1(a) and (b), respectively. In Fig. 1(b), the logic value of a data signal is determined with reference to the clock input. If a pulse arrives at an input terminal during interval between two adjacent clock pulses, the input value for the terminal is “1”. If no pulse arrives, the input value is “0”. The output pulse of a gate is synchronized with the clock input.

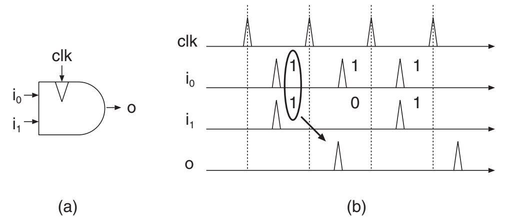

Fig. 1. AND gate of SFQ circuits and its behavior

In an SFQ circuit, the order of pulse arrivals at each gate is important for logical behavior. We show two example circuits in Figs. 2(a) and (b). All gates in the figures are D flip-flops. Fig. 2(a) is a circuit in which each gate receives a data pulse before a clock pulse. We call this clocking scheme clock-follow-data clocking[8]. Fig. 2(b) is a circuit in which each gate receives a data pulse after a clock pulse. We call this clocking scheme concurrent-flow clocking[8]. Though the circuit in Fig. 2(b) works as a shift register, that in Fig. 2(a) works like a wire. There is a circuit containing both clocking schemes. Therefore, we need to represent timing of pulse arrivals to describe the behavior of a circuit. When we design an SFQ circuit, we need to manage skew of gate input signals carefully to realize the expected behavior, i.e., expected timing.

In many cases, SFQ digital circuits are designed using cell libraries such as [9, 10]. In a design flow using a cell library, a circuit is composed of cells. Not only gates in a circuit but also wires are realized using cells. Inputs and outputs of a cell are located on its edges. Designers tile cells manually to form a circuit layout. For a circuit layout using CONNECT cell library[9], logic simulation using a commercial simulator is possible. However, unlike traditional logic circuits, logic simulation of SFQ circuits is process-dependent because logical behavior depends on timing of pulse arrivals and timing information is available only from physical cell library. This situation is very tough for designing large scale logic circuits, and methods to describe timing of pulse arrivals independent from the fabrication process are desired. In this paper, we propose two methods for this aims.

III. SFQ circuit schematics with a note about the order of pulse arrivals

We propose a timing-aware SFQ circuit schematic. Because function of each gate in a circuit depends on timing of pulse arrivals, we attach a note about the order of pulse arrivals to each gate in a schematic to describe function precisely. We represent the order with inequality between input terminals. We consider the order during a time frame which is a period for processing a group of pulses from the circuit inputs. Generally, a time frame corresponds to a clock cycle. In this paper, we consider

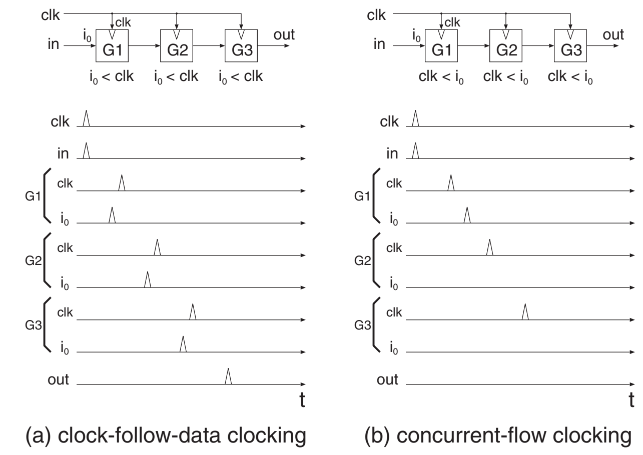

Fig. 2. Two circuits with different clocking schemes

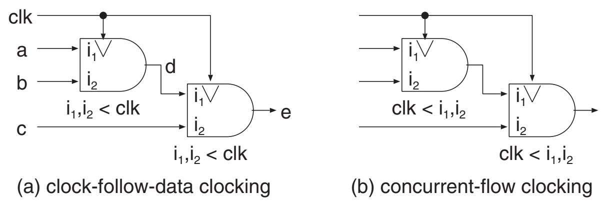

Fig. 3. Examples of SFQ logic circuit schematics with a note about the order of pulse arrivals

circuits which satisfy the following conditions.

Condition 1: Each gate in a circuit has the order of pulse arrivals independent of concrete values such as clock frequency and gate delay.

Condition 2: The order of pulse arrivals at each gate can be represented by inequalities between input terminals.

By these conditions, we do not consider circuits containing multi-cycle paths and oscillator circuits.

SFQ circuits of the same topology may have different functions depending on timing of pulse arrivals. We need representation methods to represent a function of an SFQ circuit uniquely. The following proposition holds under above conditions.

Proposition 1: Circuits of different functions have different schematics with a note.

Proof: When we feed input data to a circuit, internal states of gates after current time frame are determined by current internal states and the current input pulses(Condition 1). We assume two circuits have the same schematic with a note. When we feed the same input to these circuits, if each pair of gates in the same

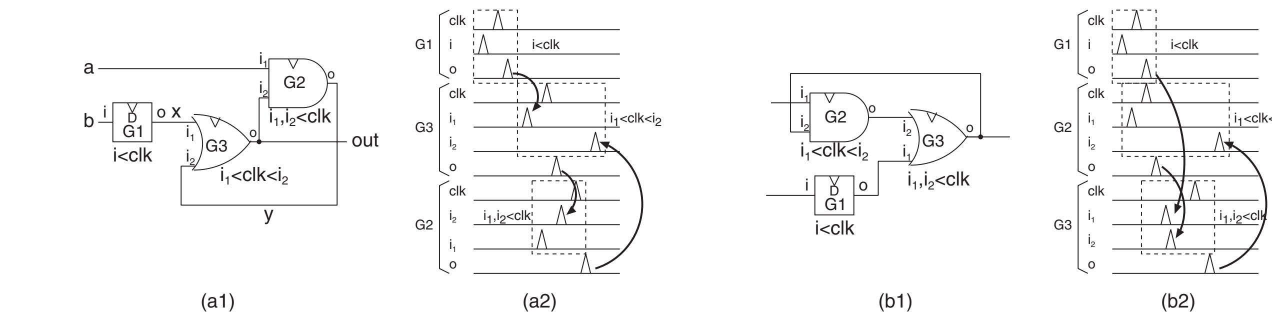

Fig. 4. Examples of SFQ logic circuits with feedback loops

position have the same internal state, inputs and outputs of each pair are the same. It is because the behavior of gates is described by inequalities(Condition 2) and each pair of gates has the same inequalities. Therefore, circuits having the same schematic must have the same function. □

Figs. 3(a) and (b) are example schematics. Fig. 3(a) is a circuit using clock-follow-data clocking and Fig. 3(b) is a circuit using concurrent-flow clocking. In a circuit of clock-follow-data clocking, for each gate in the circuit, data pulses arrive at the gate before a clock pulse arrives. Therefore, we give an inequality "$i_1, i_2 < clk$" for all gates in Fig. 3(a). In a circuit of concurrent-flow clocking, at each gate in the circuit, a clock pulse arrives before data pulses arrive. Therefore, we give an inequality "$clk < i_1, i_2$" for all gates in Fig. 3(b).

As other examples, we show schematics of SFQ circuits with feedback loops in Figs. 4(a1) and (b1). In these figures, we do not show clock wiring. Figs. 4(a2) and (b2) show behavior of the circuits in Figs. 4(a1) and (b1), respectively. In Fig. 4(a1), gate $G3$ receives a pulse though the feedback loop. The pulse through the loop arrives at $G3$ after a clock pulse arrives, and the pulse is used for the next time frame. In Fig. 4(b1), $G2$ receives a pulse through the feedback loop.

IV. Timing-aware circuit description language for SFQ logic circuits

We propose a circuit description language for SFQ circuits considering timing of pulse arrivals. This description method are designed for future design automation for SFQ logic circuits. In the new description language, we represent a gate as follows.

$$o = GATE\ \ G\ (w_1@p_1, w_2@p_2, \ldots, w_k@p_k);$$

Note that, $GATE$ represents a primitive, such as AND, OR and XOR. G represents the name of the gate. $w_1, w_2, \ldots, w_k$ represent inputs of the gate G. $p_1, p_2, \ldots, p_k$ are integers to describe the order of pulse arrivals. o is the output of gate G. This description represents one equality for each gate. Thus, description ability is different between the schematic in previous section and this description language because we can use multiple equalities for a gate in the schematic.

```
1: x   = D   G1( b@0, clk@1 );
2: out = OR  G3( x@0,   y@2, clk@1 );
3: y   = AND G2( a@0, out@0, clk@1 );
```

Fig. 5. Circuit description for Fig. 4(a1)

A description of a logic circuit consists of descriptions of gates. In a description of a logic circuit, we describe all gates. We treat SFQ circuits which satisfy Conditions 1 and 2 in previous section. For example, we can describe the circuit shown in Fig. 3(a) as follows.

$$\begin{aligned} d &= AND\ \ G_1\ (a@1, b@1, clk@2); \\ e &= AND\ \ G_2\ (c@1, d@1, clk@2); \end{aligned}$$

Because a clock pulse arrives after data inputs, a greater value is attached to "clk" than values for "a", "b", "c", and "d". We attach the same value 1 for "a" and "b", and "c" and "d". This means we give no order between these inputs. For this description language, Proposition 1 in the previous section holds.

As an example, we show a description of a circuit with a feedback loop in Fig. 5. This description corresponds to the circuit in Fig. 4(a1).

By a description of an SFQ logic circuit, logic simulation of the circuit is possible. In other words, we need no circuit layout for logic simulation.

V. Logic simulation with a circuit description written in the proposed language

We show a simulation algorithm for SFQ circuits in Fig. 6 as an example application of the proposed description language.

We use a description written in the proposed language for simulation. Simulation for each input data is done by

```
// Input: input sequences $I_0, I_1, \ldots$ (length of each sequence: $T$), circuit description(composed of $N$ gates from $G_1$ to $G_N$)
// Output: output sequences
//
// Definition of $G_l$: $o_l = gate_l \;\; G_l(w_{l,1}@p_{l,1}, w_{l,2}@p_{l,2}, \ldots, w_{l,k_l}@p_{l,k_l})$;
// $p_l^{clk}$: the order for "clk" input of $G_l$ if it exists, otherwise $\infty$.
// $PH_l$: set of pairs of input and order $\{w_{l,k}@p_{l,k} | p_{l,k} > p_l^{clk}\}$.
// $PL_l$: set of pairs of input and order $\{w_{l,k}@p_{l,k} | p_{l,k} \leq p_l^{clk}\}$.
// $o_l$ corresponds to sequence $O_l$.
// $w_{l,1}, \ldots, w_{l,k_l}$ correspond to sequences $W_{l,1}, \ldots, W_{l,k_l}$, respectively.

Prepare sequences of length $T$ corresponding to wires in the description.
Initialize all sequences except input sequences( Invalidate all elements of sequences ).

for $t$ from 1 to $T$ do
    $S \leftarrow \{1, 2, \ldots, N\}$
    while $S$ is not empty do
        Select $l$ from $S$ such that all sequences corresponds to $PL_l$ have valid $t$-th elements.
        Remove $l$ from $S$.
        Calculate new internal state of $G_l$ using $(t-1)$-th elements of sequences corresponding to $PH_l$ according to the order.
        Calculate new internal state of $G_l$ using $t$-th elements of sequences corresponding to $PL_l$ according to the order.
        $O_l[t] \leftarrow$ output of $G_l$.
    end while
end for
print output sequences
```

Fig. 6. Logic simulation algorithm

calculating outputs of gates having input data, iteratively. For each gate, we calculate the internal state according to the order of pulse arrivals.

We consider time complexity of the algorithm. At first time frame in simulation, we need to determine sequence of calculation of gate outputs. The sequence of calculation can be determined by topological sort. Therefore, we need $O(N)$ time to determine the sequence, where N denotes the number of gates, when we consider two or three input gates. Sequence of calculation after the first time frame is the same to that of the first time frame. We need constant time for calculating a gate output. Thus, the time complexity of the algorithm is $O(NT)$, where T denotes the number of time frames to simulate.

Note that, of course, we can translate timing-aware descriptions into Verilog descriptions for logic simulation. In the translation, a function of each gate is evaluated according to the order of pulse arrivals represented in the proposed description. Gates with concurrent-flow clocking should be translated to gates with latches, and gates with clock-follow-data clocking should be translated to gates without latches.

VI. Timing-aware descriptions and simulation result of a 4-bit multiplier

We show a circuit schematic and a circuit description of a processing element(PE) for the floating-point multiplier proposed in [11] in Figs. 7 and 8, respectively. We do not show clock wiring in Fig. 7. A CB(confluence buffer)

```
y  = ND G1( load_in@0, y_in@1, load_in@2 );
pp = ND G2( load_in@0,    y@1,    x_in@2 );

s0    = XOR G3( pp@0, s1@0, clk@1 );
c0    = AND G4( pp@0, s1@0, clk@1 );

s_out = XOR G5( s0@0, c3@2, clk@1 );
c2    = AND G6( s0@0, c3@2, clk@1 );

c1    = D   G7( c0@0, clk@1 );
c3    = CB  G8( c1@0, c2@0 );

load_out  = D G9 ( load_in@1, clk@0 );
x_out     = D G10(    x_in@1, clk@0 );
reset_out = D G11(reset_in@1, clk@0 );
```

Fig. 8. Circuit description of a processing element for the multiplier

gate behaves as an asynchronous OR gate. ND represents a non-destructive read-out flip-flop gate. In Fig. 7, for any input for the circuit, the CB gate does not receive pluses from both $c1$ and $c2$ at a time frame. We give an inequality "i_1, i_2" for this CB gate because we do not need to give order between input terminals. Gates G5 and G6 receive pulses from the feedback loop. Gates G1 and G2 realize a partial product generator in the multiplier. Gates from G3 to G8 realize a partial product adder in the multiplier.

We simulated a 4-bit integer multiplier shown in Fig. 9

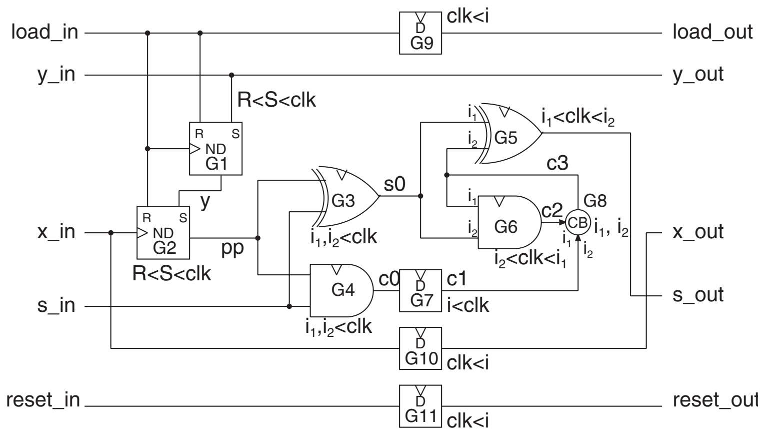

Fig. 7. Schematic of the processing element for the multiplier

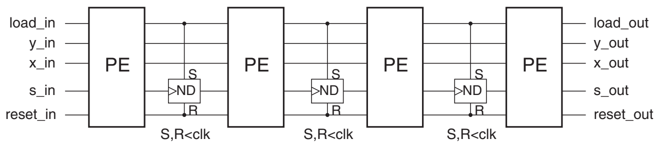

Fig. 9. 4-bit multiplier composed of the PEs

composed of the PEs. We obtain the upper 5-bit of a result. We show simulation result of the operation in Fig. 10. In Fig. 10, "1" means that a pulse exists during a time frame, "0" means that no pulse exists, and "x" means uncertain. The leftmost column corresponds to the 1st time frame, and the rightmost column corresponds to the last time frame in the simulation. Because a time frame corresponds to a clock cycle, we apply a pulse to the circuit input "clk_in" for each time frame.

In Fig. 10, the multiplier computes $(1111)_2 \times (1101)_2$ $(= (11000011)_2)$ and $(1010)_2 \times (1110)_2$ $(= (10001100)_2)$ as shown by rectangles with broken lines. Inputs for "x_in" and "y_in" are fed from the LSB bit to the MSB bit. As shown in the figure, we obtained the correct outputs $(11000)_2$ and $(10001)_2$ at the circuit output "s_out".

```
Input
reset_in :[ 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 ]
load_in  :[ 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 ]
y_in     :[ 0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 1 0 0 0 0 0 0 0 ]
x_in     :[ 0 0 0 0 0 0 0 1 0 1 1 0 0 1 1 1 0 0 0 0 0 0 0 ]
s_in     :[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
clk_in   :[ 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ]

Output
reset_out:[ x x x x 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 ]
load_out :[ x x x x 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 ]
y_out    :[ 0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 1 0 0 0 0 0 0 0 ]
x_out    :[ x x x x 0 0 0 0 0 0 0 1 0 1 1 0 0 1 1 1 0 0 0 ]
s_out    :[ x x x x x 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 ]
```

Fig. 10. Simulation result of the 4 bit multiplier

The proposed description methods are useful for communication between designers and for expressions of ideas. Logic simulation based on the description language is useful for exploring an architecture of circuit before detailed design.

VII. Conclusion

We have proposed two timing-aware description methods for SFQ circuits. One method is a timing-aware circuit schematic. We attach a note about the order of pulse arrivals for each gate in a schematic to describe function of circuit precisely. The other is a timing-aware description language for SFQ logic circuits. As an example application of the language, we have shown a logic simulation algorithm. By the simulation algorithm, we can verify the logic function of the circuit before designing a circuit layout.

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, pp. 3–28, Mar. 1991.

[2] National Security Agency, "Superconducting Technology Assessment," Aug. 2005. `http://www.nitrd.gov/pubs/nsa/sta.pdf`

[3] A. H. Silver, "Superconductor Technology for High-End Computing System Issues and Technology Roadmap," *Proc. ACM/IEEE Conference on Supercomputing (SC'05)*, pp. 64, Nov. 2005.

[4] M. Dorojevets, "Opportunities, Challenges, and Projections for Superconductor RSFQ Microprocessors," *Proc. ACM/IEEE Conference on Supercomputing (SC'05)*, pp. 65, Nov. 2005.

[5] N. Takagi, K. Murakami, A. Fujimaki, N. Yoshikawa, K. Inoue, and H. Honda, "Proposal of a Desk-Side Supercomputer with Reconfigurable Data-Paths Using Rapid Single-Flux-Quantum Circuits," *IEICE Trans. Electronics*, vol.E91-C, no.3, pp.350–355, Mar., 2008.

[6] W. Chen, A.V. Rylyakov, V. Patel, J.E. Lukens, and K.K. Likharev, "Superconductor Digital Frequency Divider Operating up to 750 GHz," *Appl. Phys. Lett.*, vol.73, pp.2817–2819, 1998.

[7] Y. Yamanashi, T. Kainuma, N. Yoshikawa, I. Kataeva, H. Akaike, A. Fujimaki, M. Tanaka, N. Takagi, S. Nagasawa, and M. Hidaka, "100 GHz Demonstrations Based on the Single-Flux-Quantum Cell Library for the 10 kA/cm^2 Nb Multi-Layer Process," *IEICE Trans. Electronics*, vol.E93-C, no.4, pp.440–444, Apr., 2010.

[8] K. Gaj, E.G. Friedman,and M.J. Feldman, "Timing of Multi-Gigahertz Rapid Single Flux Quantum Digital Circuits," *J. VLSI Signal Process. Syst.*, vol. 16, pp. 247–276, July, 1997.

[9] S. Yorozu, Y. Kameda, H. Terai, A. Fujimaki, T. Yamada, and S. Tahara, "A Single Flux Quantum Standard Logic Cell Library," *Physica C*, vol.378-381, pp. 1471-1474, 2002.

[10] SUNY RSFQ Cell Library, `http://pavel.physics.sunysb.edu/RSFQ/Lib/`

[11] 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, pp. 657–660, June 2009.