Retiming of Single Flux Quantum Logic Circuits for Flip-flop Reduction

Nobutaka Kito Kazuyoshi Takagi Naofumi Takagi

School of Engineering
Chukyo University
Toyota, 470–0393, Japan
nkito@sist.chukyo-u.ac.jp

Graduate School of Informatics
Kyoto University
Kyoto, 606–8501, Japan

Abstract— We propose a retiming method of superconductive Single Flux Quantum (SFQ) logic circuits for flip-flop reduction. Because SFQ logic circuits use pulse logic, each input of logic gates has latching function. The number of flip-flops in SFQ circuits can be reduced by utilizing the latching function. We formulate retiming for flip-flop reduction as an instance of integer linear program considering the latching function. Experimental results show that most of flip-flops in SFQ circuit realizations of ISCAS'89 benchmark circuits can be eliminated by the proposed method.

I. Introduction

Superconducting devices have been considered as potentially alternative devices of mainstream semiconductor devices for long years[1]. Single Flux Quantum (SFQ) circuits[2] are high-speed and ultra low-power circuits using superconductive devices. Recently, difficulties in process shrinking of semiconductor circuits emerge, and large scale computer systems need extremely high performance with reduced power consumption. Technologies for computers using SFQ circuits have been studied [3, 4], and several research programs of superconducting digital circuits including SFQ circuits are in progress[5, 6]. SFQ circuits use pulse logic. In other words, logical values "0" and "1" are represented with using pulses. In the current SFQ circuit design technology, each gate in SFQ circuits is synchronized with clock pulses. SFQ circuits work at high frequency up to tens of GHz, and some basic circuits are reported to work up to hundreds GHz[7, 8].

Each gate in SFQ circuits has a clock input, and is synchronized with clock pulses. A logic gate of SFQ circuits can work as if it has flip-flops inside the gate. In other words, each gate has latching function. In SFQ circuit design, two major clocking schemes are known for driving gates. One is concurrent-flow clocking and the other is clock-follow-data clocking[9], and only order of pulse arrivals for the clock input and data inputs of each gate is different between the two clocking schemes. In a circuit consisting of gates using concurrent-flow clocking, each gate works as if each data input has a flip-flop. Thus, a circuit consisting of gates using concurrent-flow clocking works as a pipelined circuit even if only logic gates, such as AND gates and OR gates, are used in the circuit. In a circuit consisting of gates using clock-follow-data clocking, each gate works like a traditional logic gate without the latching function. Thus, a circuit consisting of gates using clock-follow-data clocking works like a traditional combinational circuit. Though we could use both the schemes in a circuit widely, mixture of clocking schemes is used only for adjusting latency of circuits because current design flow of SFQ circuits is semi-automatic and designers of SFQ circuits adjust skew between gates carefully by inserting delay elements manually.

As design automation for SFQ circuits progresses, precise timing management in SFQ circuit design will be automated. Extensive use of the two clocking schemes in an SFQ circuit, by introducing an idea of retiming, may reduce the number of flip-flops. In this paper, we propose a retiming method for flip-flop reduction of SFQ circuits considering the latching function of gates. In the method, the problem is formulated as an instance of integer linear program (ILP), and the optimal solution can be obtained systematically by solving it.

In SFQ circuits, an active device, called a splitter, is required to split a signal. In addition, a short wiring is realized by an active device, called a josephson transmission line, and a long wiring is realized by a high-speed point-to-point passive device, called a passive transmission line, which requires a transceiver and a receiver at the both ends. The flip-flop reduction will reduce design effort to synthesize a clock tree, and circuit area for clock distribution. Thus, reduction of flip-flops is important for SFQ circuit design. Though retiming methods for register reduction have been proposed for traditional sequential circuits[10], new formulation and solution for SFQ circuits are necessary for flip-flop reduction with considering the latching function of gates.

Currently logic designs of SFQ circuits are mainly performed by hands, and using logic synthesis tools for conventional VLSI circuits is considered as an option for designing random logic circuits. The proposed method is useful for refining handmade logic designs and designs ob-

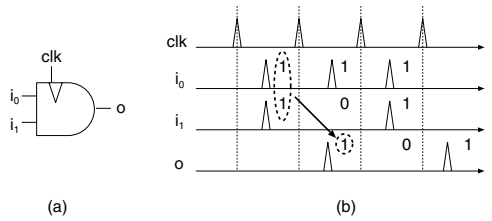

Fig. 1. SFQ AND gate and its behavior

tained by conventional logic synthesis tools.

This paper is organized as follows. In the next section, we briefly review timing design of SFQ logic circuits and retiming of traditional sequential circuits. In Section 3, we propose a retiming method for reducing flip-flops of SFQ circuits. We show a problem formulation using integer linear programming and show an example. In Section 4, we show evaluation results. In Section 5, we conclude this paper.

II. Preliminaries

A. Clocking and timing design of SFQ logic circuits

In an SFQ circuit, signals are transmitted using voltage pulses on signal lines. Each gate of an SFQ circuit has a clock input terminal and works synchronized with clock pulses. As an example, the symbol and the behavior of an AND gate are shown in Figs. 1(a) and (b), respectively. In Fig. 1(a), the gate has a clock input terminal as described above. In Fig. 1(b), the value of a data input signal is evaluated with clock pulses. When a pulse arrives at a data input of a gate during an interval between adjacent clock pulses, the input value corresponding to the interval is "1". If no pulse arrives during the interval, the input value is "0". The output of a gate is synchronized with the clock pulse. In other words, each data input holds an input value till the arrival of a clock pulse, and the gate outputs the result after the arrival of the clock pulse.

In an SFQ circuit, the order of pulse arrivals at each gate affects its functional behavior as shown in Figs. 2(a) and (b). Both circuits consist of D flip-flops. In Fig. 2(a), each gate receives a data pulse before a clock pulse. This clocking scheme is known as clock-follow-data clocking[9]. In Fig. 2(b), each gate receives a data pulse after a clock pulse. This clocking scheme is known as concurrent-follow clocking[9]. Each gate in Fig. 2(b) works as a traditional D flip-flop, and the circuit in Fig. 2(b) works as a shift register. On the other hand, the circuit in Fig. 2(a) works as a wire. The two clocking schemes can be used in a circuit simultaneously. In this paper, we use the description method shown in [11] to describe the clocking scheme of each gate. An inequality is attached for each gate to represent the order of pulse arrivals. In Fig. 2(a), the inequality "$i_0 < clk$" is attached to represent that a data

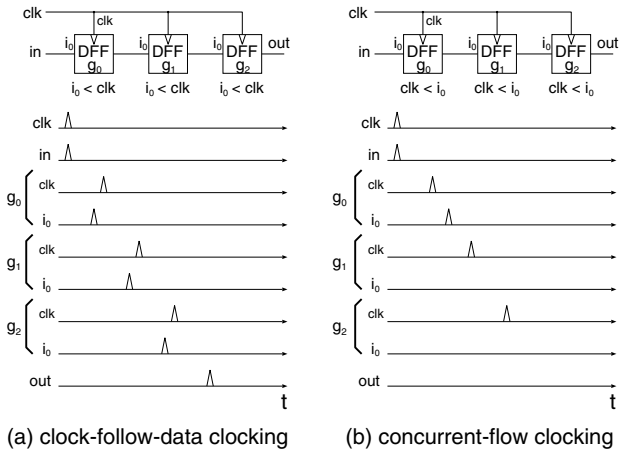

Fig. 2. Two circuits with different clocking schemes

pulse for input i_0 arrives before the corresponding clock pulse.

We present another example in Fig. 3. In Fig. 3(a), all gates except g_4 use concurrent-flow clocking, and g_4 uses clock-follow-data clocking. In Fig. 3(b), the order of pulse arrivals is changed for g_4, and D flip-flop g_3 is removed because an input pulse for input i_1 of gate g_4 arrives after the corresponding clock pulse and the data input has the latching function. For gate g_4, we use both clock-follow-data clocking and concurrent-flow clocking simultaneously. Note that, though we have considered clocking of each gate so far, we can consider clocking for each data input of a gate. As a result of flip-flop reduction, we can also reduce a splitter for clock distribution. Hereafter, we will not show clock wiring in the figures, because the clock signal is fed to all gates and topology of clock distribution is not concerned in this paper.

B. Retiming of traditional sequential circuits

Retiming is a transformation of a circuit with preserving its functional behavior by inserting and deleting flip-flops. In this paper, functional behaviors of circuits are considered to be the same if their outputs from the primary outputs are the same at each clock cycle when the same sequence of input values is fed. Although functional behavior of a circuit is preserved by retiming, functional behavior of each gate may be changed. As an example, the circuit in Fig. 4(a) is transformed to the circuit in Fig. 4(b) by retiming. In Fig. 4(a), gate g_2 receives inputs from primary inputs a and b through flip-flops g_0 and g_1,respectively, and g_4 receives an input from primary input c through flip-flop g_3. On the other hand, in Fig. 4(b), gate g_2 receives inputs from primary inputs a and b directly and gate g_4 receives an input from primary input c directly. Although these two circuits have different structures, their functional behaviors are the same, i.e., their

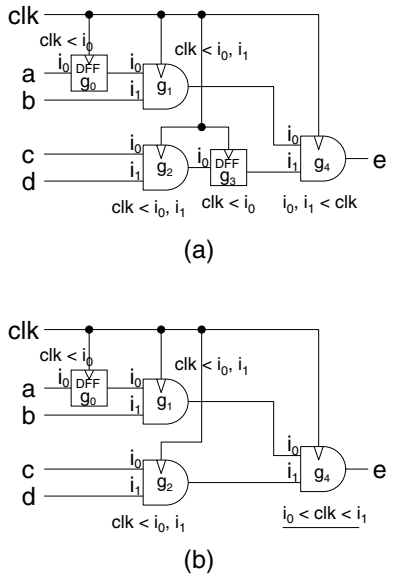

Fig. 3. Flip-flop reduction by changing the order of pulse arrivals

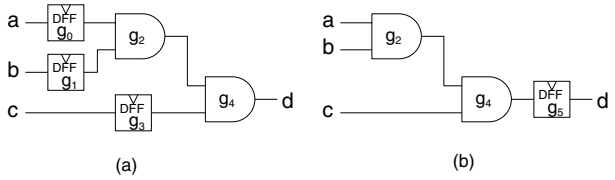

Fig. 4. Retiming of a circuit and reduction of flip-flops

outputs are the logical AND of inputs a, b, and c of the previous clock cycle. While the circuit in Fig. 4(a) has three flip-flops, the circuit in Fig. 4(b) has only one flip-flop. Thus, the number of flip-flops can be reduced by retiming.

For dealing with retiming of circuits, it is useful to consider a circuit as a graph (hereafter referred to as a circuit graph)[10]. In a circuit graph, logic gates, primary inputs, and primary outputs are treated as nodes of the graph, and each net is treated as a set of edges of the graph. Flip-flops are not treated as logic gates, and, to each edge, the number of flip-flops between the two gates at its both ends is attached as a label. We show the circuit graphs in Figs. 5 (a) and (b) corresponding to the circuits in Figs. 4 (a) and (b), respectively. We let a circuit graph be $G = (V, E, w)$ where V and E denote a set of nodes and a set of edges in a circuit, respectively, and function $w : E \to \mathbb{Z}$ represents the label of each edge.

We let a retiming of a circuit be function $r : V \to \mathbb{Z}$, and the circuit graph after retiming r be $G_r = (V, E, w_r)$. In the circuit graph after retiming r, the sum of edge labels on each path from one of nodes corresponding to the primary inputs to node v is incremented by $r(v)$. For

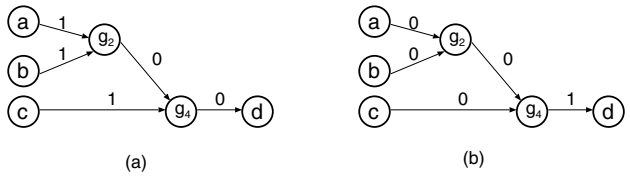

Fig. 5. Example of circuit graphs ((a) and (b) correspond to Figs. 4(a) and (b), respectively)

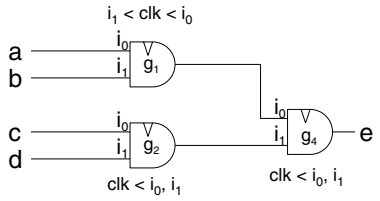

Fig. 6. Flip-flop reduction for the circuits in Fig 3 by retiming and changing the order of pulse arrivals

the circuit graph after retiming r, label $w_r(e)$ of edge e is defined as follows:

$$w_r(e) = w(e) + r(dest(e)) - r(src(e)) \geq 0 \qquad (1)$$

where $src(e)$ and $dest(e)$ represent the source node and the destination node of edge e, respectively. For example, the retiming shown in Fig. 4 is represented by $r(a) = r(b) = r(c) = r(d) = 0$, and $r(g_2) = r(g_4) = -1$. Because $r(g4) = -1$, the sum of edge labels on each path to g_4 is 0 in Fig. 5(b), while the sum of edge labels on each path to g_4 is 1 in Fig. 5(a). Because $r(d) = 0$, the sum of edge labels on each path to d is 1 both in Figs. 5(a) and (b). Note that this formulation of retiming is also applicable for circuits with feedback loops [10].

III. Retiming of SFQ logic circuits for flip-flop reduction

We propose a method to reduce the number of flip-flops in SFQ circuits by retiming and optimization of the order of pulse arrivals. As an example of the reduction of the number of flip-flops, the circuit shown in Fig. 3(a) is transformed to the circuit without flip-flops shown in Fig. 6 by considering both the retiming and the order of pulse arrivals of the gates. In SFQ circuits, because splitters are needed to split a signal and wires occupy relatively large circuit area, reduction of flip-flops is important for reduction of circuit area.

In SFQ circuits, each input of a logic gate can obtain the latching function by adjusting the order of pulse arrivals of the gate. We introduce a graph to consider retiming of SFQ circuits by extending the circuit graph. Hereafter, we

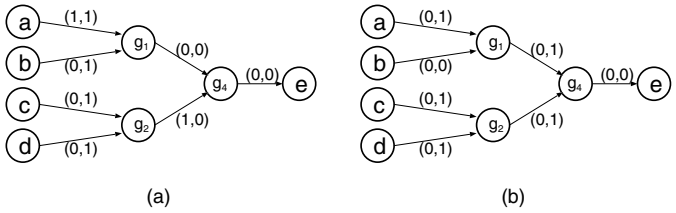

Fig. 7. Example of SFQ circuit graphs ((a) and (b) correspond to Fig. 3(a) and Fig. 6, respectively)

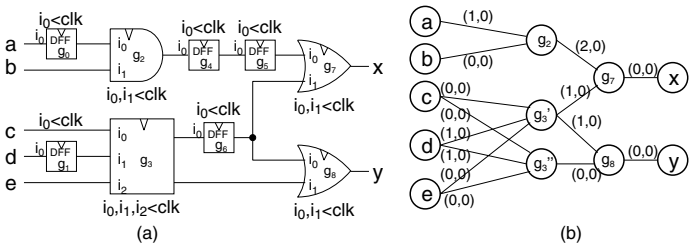

Fig. 8. Example SFQ circuit (a), and its SFQ circuit graph (b)

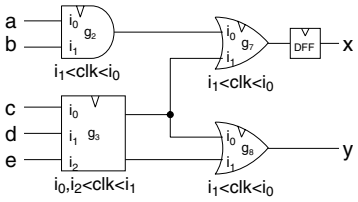

Fig. 9. Example of flip-flop reduction for the circuit in Fig. 8

call the extended graph an SFQ circuit graph. SFQ circuit graph $S = (V, E, wt)$ can treat the latching function of data inputs of gates. Although SFQ circuit graph S is similar to circuit graph $G = (V, E, w)$, we change edge labeling to represent the latching function of each data input, and use a pair of values as a label of an edge. The function $wt : E \to \mathbb{Z} \times \{0, 1\}$ represents the assignment of edge labels. The first element of the pair represents the number of flip-flips between the two gates at the both ends of the edge. The second element of the pair represents the latching function of the data input, namely, values "1" and "0" represent that the data input has the latching function or not, respectively. We show an example of SFQ circuit graphs in Fig. 7. When the following equation holds between circuit graph G and SFQ circuit graph S which has the same V and E as G, the functional behavior of S is equivalent to that of G.

$$w(e) = wt^d(e) + wt^t(e)$$

where $wt^d(e)$ and $wt^t(e)$ represent the first element and the second element of $wt(e)$, respectively.

We can introduce retiming for SFQ circuits. When retiming r is applied to SFQ circuit graph S, edge label wt_r has the following relation similar to formula (1).

$$wt_r^d(e)+wt_r^t(e) = wt^d(e)+wt^t(e)+r(dest(e))-r(src(e)) \geq 0 \quad (2)$$

where $wt_r^d(e)$ and $wt_r^t(e)$ represent the first element and the second element of the label of edge e after retiming r, respectively. Note that, while we can determine edge labeling w_r by a pair of circuit graph G and retiming r, we need to determine second element of edge labels wt_r^t with retiming r for SFQ circuit graph S because we may not determine whether $wt_r^t(e)$ is one or zero by a pair of retiming r and S.

When a fanout point exists in a circuit, flip-flops may be shared between gates connected by branched wiring. For example, one of the outputs of gate g_3 is fed to two gates g_7 and g_8 in Fig. 8(a), and flip-flop g_6 is shared. When a node is the source of several edges in a circuit graph, for implementation, we prepare a series of flip-flops for the furthermost gate, i.e., the gate connected by the edge with largest wt_r^d, and we feed signals to the other gates by splitting the signal from the series. We let the maximum number of flip-flops between the gate corresponding to node u and the furthermost gate of u be $w_r^{max}(u)$ when retiming r is considered. $w_r^{max}(u)$ can be represented as follows:

$$w_r^{max}(u) = \max_{src(e)=u} \left(wt_r^d(e)\right). \quad (3)$$

Note that, we consider $w_r^{max}(u) = 0$ when node u is not a source of any edge. The number of flip-flops for implementing SFQ circuit graph $S = (V, E, wt)$ is calculated as follows:

$$\sum_{u \in V} w_r^{max}(u). \quad (4)$$

We can reduce the number of flip-flops by searching a pair of retiming r and assignment of the latch function wt_r^t which reduces the value of the above formula. We use formula (4) as the objective function of an instance of ILP, and use equation (2) as the constraints. By solving the instance, we can obtain labels for all edges. In equation (2), $wt^d(e)$ and $wt^t(e)$ can be extracted from an original circuit.

To preserve the circuit function in retiming, $r(v) = 0$ is given as a constraint for each input node and output node v. For each edge e whose destination is an output node, a constraint $wt_r^t(e) = 0$ is also given because the node does not correspond to a logic gate. When a gate with multiple output terminals exists, we consider each output terminal of the gate as a node of a circuit graph. Nodes g_3' and g_3'' in Fig. 8(b) correspond to gate g_3 in Fig. 8(a). In the formulation of flip-flop reduction, nodes corresponding to the same gate must have the same retiming value.

We can specify the latching function of data inputs as design options by appending constraints $wt_r^t(e) = 0$ or

minimize: $w_r^{max}(a)+w_r^{max}(b)+w_r^{max}(c)+w_r^{max}(d)+w_r^{max}(e)+w_r^{max}(g_2)+w_r^{max}(g_3')+w_r^{max}(g_3'')+w_r^{max}(g_7)+w_r^{max}(g_8)$

Subject to

$$
\begin{aligned}
wt^d((a,g_2)) - w_r^{max}(a) &\le 0\\
wt^d((b,g_2)) - w_r^{max}(b) &\le 0\\
wt^d((c,g_3')) - w_r^{max}(c) &\le 0\\
wt^d((c,g_3'')) - w_r^{max}(c) &\le 0\\
wt^d((d,g_3')) - w_r^{max}(d) &\le 0\\
wt^d((d,g_3'')) - w_r^{max}(d) &\le 0\\
wt^d((e,g_3')) - w_r^{max}(e) &\le 0\\
wt^d((e,g_3'')) - w_r^{max}(e) &\le 0\\
wt^d((g_2,g_7)) - w_r^{max}(g_2) &\le 0\\
wt^d((g_3',g_7)) - w_r^{max}(g_3') &\le 0\\
wt^d((g_3',g_8)) - w_r^{max}(g_3') &\le 0\\
wt^d((g_3'',g_8)) - w_r^{max}(g_3'') &\le 0\\
wt^d((g_7,x)) - w_r^{max}(g_7) &\le 0\\
wt^d((g_8,y)) - w_r^{max}(g_8) &\le 0\\
wt_r^d((a,g_2)) + wt_r^t((a,g_2)) - r(g_2) + r(a) &= 1\\
wt_r^d((b,g_2)) + wt_r^t((b,g_2)) - r(g_2) + r(b) &= 0\\
wt_r^d((c,g_3')) + wt_r^t((c,g_3')) - r(g_3') + r(d) &= 0\\
wt_r^d((d,g_3')) + wt_r^t((d,g_3')) - r(g_3') + r(d) &= 1\\
wt_r^d((e,g_3')) + wt_r^t((e,g_3')) - r(g_3') + r(e) &= 0\\
wt_r^d((c,g_3'')) + wt_r^t((c,g_3'')) - r(g_3'') + r(d) &= 0\\
wt_r^d((d,g_3'')) + wt_r^t((d,g_3'')) - r(g_3'') + r(d) &= 1\\
wt_r^d((e,g_3'')) + wt_r^t((e,g_3'')) - r(g_3'') + r(e) &= 0\\
wt_r^d((g_2,g_7)) + wt_r^t((g_2,g_7)) - r(g_7) + r(g_2) &= 2\\
wt_r^d((g_3',g_7)) + wt_r^t((g_3',g_7)) - r(g_7) + r(g_3') &= 1\\
wt_r^d((g_3',g_8)) + wt_r^t((g_3',g_8)) - r(g_8) + r(g_3') &= 1\\
wt_r^d((g_3'',g_8)) + wt_r^t((g_3'',g_8)) - r(g_8) + r(g_3'') &= 0\\
wt_r^d((g_7,x)) + wt_r^t((g_7,x)) - r(x) + r(g_7) &= 0\\
wt_r^d((g_8,y)) + wt_r^t((g_8,y)) - r(y) + r(g_8) &= 0\\
r(a) = r(b) = r(c) = r(d) = r(e) = r(x) = r(y) &= 0\\
wt_r^t((g_7,x)) &= 0\\
wt_r^t((g_8,y)) &= 0\\
wt_r^d(e) &\ge 0\\
0 \le wt_r^t(e) &\le 1\\
r(g_3') - r(g_3'') &= 0
\end{aligned}
$$

Fig. 10. ILP formulation for the circuit in Fig. 8

$wt_r^t(e) = 1$ for each edge e whose destination is one of those data inputs. When appended optional conditions are unreasonable, a contradiction occurs in constraints and an ILP solver will report infeasibility. Note that the proposed method can treat circuits with feedback loops because retiming in Sect. II.B can treat such circuits.

As an example of the formulation of flip-flop reduction, we show an instance of ILP in Fig. 10 for the circuit shown in Fig. 8. The objective function corresponds to formula (4). The first 14 constraints correspond to equation (3), and the next 14 constraints correspond to formula (2). The circuit contains a gate with multiple output terminals. The last constraint is required to treat gate g_3 having multiple outputs. SFQ circuit after reduction is shown in Fig. 9. In this example, the number of flip-flops is reduced from five to one.

IV. Experiments

We evaluated the proposed method using ISCAS'89 benchmark circuits. We generated instances of ILP from the benchmark circuits. We used lp_solve 5.5.0.13 as the ILP solver, and a workstation with Intel Xeon E3-1240 processor for solving instances of ILP. Evaluation results are shown in Table I. Each logic gate, such as an AND gate and an OR gate, in the original benchmark circuits is regarded as a gate of clock-follow-data clocking, i.e., each data input does not have the latching function, and each flip-flop in the original benchmark circuits is regarded as a flip-flops of concurrent-flow clocking. Thus, each starting point realization of the original benchmark circuits which is considered in this experiment has the same amount of logic gates and flip-flops and the same functional behavior as the original one has.

The first column in "# of FFs" represents the number of flip-flops in the original benchmark circuits (or the starting point realizations of SFQ circuits). The second column represents the result of the traditional retiming [10] without considering the latching function of data inputs. As shown in Fig 3, the number of flip-flops can be reduced by changing the order of pulse arrivals. The third column represents the result of optimization of the order of pulse arrivals without retiming. The fourth column represents the result of the proposed method. The last column of the table shows the runtime of the ILP solver for the proposed method. As shown in the table, we can reduce the number of flip-flops drastically by the proposed method. The runtime of the solver for the proposed method is short and reasonable for the circuits. In the current SFQ circuit design flow, circuits composed of several hundred gates are treated as a circuit block. The proposed method is useful and effective for designing such circuit blocks.

V. Conclusion

We have proposed a retiming method for reduction of flip-flops in SFQ circuits. Because SFQ circuits use pulse logic system, each data input of a gate may have the latching function depending on the order of pulse arrivals at the gate. The proposed method utilizes the latching function of data inputs of gates to reduce the number of flip-flops. The method formulates the reduction as an instance of integer linear program. We have evaluated the proposed method for ISCAS'89 benchmark circuits. Experimental results show that most of flip-flops in the circuits are eliminated by the method.

TABLE I
EXPERIMENTAL RESULTS FOR ISCAS'89 BENCHMARK CIRCUITS

		# of FFs				
			after reduction			
circuits	# of gates	original	retiming	pulse ordering	proposed	time(s)
s27	10	3	3	0	0	0.00
s208.1	104	8	8	0	0	0.01
s298	119	14	14	0	0	0.01
s344	160	15	15	0	0	0.01
s349	161	15	15	0	0	0.00
s382	158	21	18	0	0	0.02
s386	159	6	6	0	0	0.01
s400	164	21	18	0	0	0.02
s420.1	218	16	16	0	0	0.01
s444	181	21	18	0	0	0.03
s510	211	6	6	0	0	0.02
s526	193	21	21	0	0	0.03
s526n	194	21	21	0	0	0.04
s641	379	19	19	0	0	0.02
s713	393	19	19	0	0	0.02
s820	289	5	5	0	0	0.06
s832	287	5	5	0	0	0.02
s838.1	446	32	32	0	0	0.06
s953	395	29	22	23	0	0.10
s1196	529	18	18	1	0	0.05
s1238	508	18	18	1	0	0.06
s1423	657	74	74	0	0	0.26
s1488	653	6	6	0	0	0.35
s1494	647	6	6	0	0	0.11
s5378	2779	179	143	0	0	1.67
s9234	5597	228	194	0	0	4.01
s9234.1	5597	211	191	0	0	3.08
s13207	7951	669	464	76	10	16.58
s13207.1	7951	638	455	74	13	12.66
s15850	9772	597	567	14	0	22.73
s15850.1	9772	534	515	14	0	16.41
s35932	16065	1728	1728	288	0	309.35
s38417	22179	1636	1390	218	0	201.15
s38584	19253	1452	1451	154	98	151.13
s38584.1	19253	1426	1425	154	98	170.10

ACKNOWLEDGEMENT

This work has been supported in part by ALCA JST.

REFERENCES

[1] K.K. Likharev, "Superconductor digital electronics," *Physica C: Superconductivity*, vol.482, pp.6-18, Nov. 2012.

[2] 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.

[3] A.H. Silver, "Superconductor technology for high-end computing system issues and technology roadmap," *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference*, p.64, Nov. 2005.

[4] 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.

[5] ALCA research projects, http://www.jst.go.jp/alca/en/kadai.html

[6] Cryogenic Computing Complexity (C3) Program, http://www.iarpa.gov/Programs/sso/C3/c3.html

[7] W. Chen, A.V. Rylyakov, V. Patel, J.E. Lukens, and K.K. Likharev, "Rapid single flux quantum T-flip flop operating up to 770 GHz," *IEEE Trans. Applied Superconductivity*, vol.9, no.2, pp.3212-3215, June 1999.

[8] 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. Applied Superconductivity*, vol.21, no.3, pp.792-796, June 2011.

[9] K. Gaj, E. Friedman, and M. Feldman, "Timing of multi-gigahertz rapid single flux quantum digital circuits," *J. VLSI Signal Process. Syst.*, vol.16, pp.247-276, July 1997.

[10] C.E. Leiserson, and J.B. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol.6, no.1, pp.5-35, 1991.

[11] N. Kito, K. Takagi, and N. Takagi, "Timing-aware description methods and gate-level simulation of single flux quantum logic circuits," *Proceedings of the 17th Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI2012)*, pp.319-324, Mar. 2012.