# A Transduction-based Framework to Synthesize RSFQ Circuits

Shigeru Yamashita
Nara Institute of Sci. and Tech.
8916-5 Takayama
Ikoma 630-0192, Japan
Tel: +81-743-72-5301
Fax: +81-743-72-5309
e-mail: ger@is.naist.jp

Katsunori Tanaka
NEC Corporation
1753 Shimonumabe, Nakahara-ku,
Kawasaki 211-8666, Japan
Tel: +81-44-431-7541
Fax: +81-44-431-7589
e-mail k-tanaka@jm.jp.nec.com

Hideyuki Takada
Kyoto University
Yoshida-Honmachi, Sakyo
Kyoto 606-8501, Japan
Tel: +81-75-753-5375
Fax: +81-75-753-4970
e-mail: htakada@db.soc.i.kyoto-u.ac.jp

Koji Obata  Kazuyoshi Takagi
Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan
Tel: +81-52-789-4597
Fax: +81-52-789-3798
{obata, ktakagi}@takagi.nuie.nagoya-u.ac.jp

**Abstract— In this paper, we propose a new framework to synthesize rapid single flux quantum (RSFQ) logic circuits. In our framework, we construct a virtual cell, which we call "2-AND/XOR," from the RSFQ logic primitives. By using 2-AND/XOR cells, we can successfully adopt the conventional logic design techniques into our framework, and thus we can successfully generate RSFQ circuits in reasonable time even for large benchmark circuits that have not been reported in the existing researches.**

## I. Introduction

Rapid single flux quantum (RSFQ) integrated circuits composed of Josephson-junction devices have been intensively studied because of their potentially high performance with high clock frequency and extremely low power consumption [7]. RSFQ technology has the following features [7].

- Ultrafast digital signals can be passed along the chips ballistically with a propagation speed approaching that of light.
- Intrinsic switching time of the Josephson junction is also very short, typically a few picoseconds.
- The power dissipated by a Josephson junction is typically below one microwatt. Hence, the problem of removal of heat is quite solvable. Currently this is not essentially true since we need some cooling system for the whole RSFQ circuits themselves. Although special cooling systems cannot be used for a consumer PCs, we may be able to afford special cooling devices for high-end computing servers. Also, at this moment, we cannot deny the possibility that we can construct ultra low power systems by RSFQ technology in the future.
- The Josephson junction fabrication technologies are considerably simpler than those of the conventional semiconductor (both Si and GaAs) transistors with similar design rules.

Although it currently requires a refrigeration technique, such as liquid-helium cooling, the benefits of RSFQ technology would become huge in the near future.

As the current CMOS technology is approaching "Red Brick Wall" [1], the RSFQ technology is considered as one of the promising next generation technologies. Indeed, the RSFQ device is listed as one of the most promising ones in the "Emerging Research Devices" list by ITRS [1]. Actually RSFQ digital circuits containing several thousands of Josephson junctions have been successfully implemented and their high performance has been confirmed [12].

Although it still appears quite challenging to realize primitive RSFQ logic cells efficiently, it is also a challenge to establish systematic logic design methods for large RSFQ circuits. The reason is that the logic primitives in RSFQ circuits are very much different from those in the conventional CMOS technology, and thus we cannot directly utilize the conventional logic design techniques. Therefore, we also need to do researches for the logic design methodologies for RSFQ logic circuits.

Among the researches for logic design methods for RSFQ circuits, the cell-based methods [6, 16, 17] have been studied like conventional CMOS technology. In their methods, a logic primitive called "RSFQ $D_2$ flip-flop" is used to replace a node of BDD (Binary Decision Diagram) [2]. Recently a systematic design method [9]

0-7803-9451-8/06/$20.00 

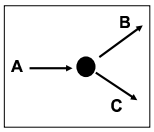


Fig. 1. SPL.

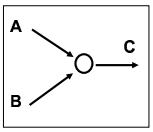


Fig. 2. CB.

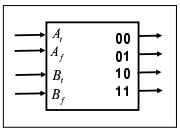


Fig. 3. 2x2-Join.

TABLE I
2X2-JOIN PULSE OPERATION

| Inputs | | | | Outputs | | | |
|---|---|---|---|---|---|---|---|
| $A_t$ | $A_f$ | $B_t$ | $B_f$ | 00 | 01 | 10 | 11 |
| Pulse | No | Pulse | No | No | No | No | Pulse |
| Pulse | No | No | Pulse | No | No | Pulse | No |
| No | Pulse | Pulse | No | No | Pulse | No | No |
| No | Pulse | No | Pulse | Pulse | No | No | No |

has been proposed to use another primitive called "2x2-Join" [5]. The method is also based on BDDs. For small circuits, BDD-based methods may generate good circuits. However, it is obvious that BDD-based methods such as [9] cannot be applied directly to large circuits since their manipulation of BDDs essentially takes a lot of time for the large functions. Thus it is desirable to have another approach that can be applied to large circuits.

For that purpose we propose a framework that is not based on the BDD manipulation. Our framework consists of two phases which are very much alike to the conventional logic design methodologies for AND/OR/NOT gates. At first phase, we generate initial circuits consisting of two-input nodes by using any conventional techniques. A two-input node is transformed to a virtual logic cell called *2-AND/XOR* cell which is introduced in this paper. Then, at the second phase, we optimize the circuit by using a transformation-based heuristic method like the conventional logic design. For that, we modify the original Transduction Method [8] to be suitable for the 2-AND/XOR cells.

A 2-AND/XOR cell exploits the property of a 2x2-Join fully, i.e., it can represent all the possible functions realized by a 2x2-Join. More importantly, by considering logic circuits consisting of 2-AND/XOR cells, we are able to adopt the conventional logic design techniques for RSFQ circuit design. Indeed our method can handle large circuits that have not been reported in existing researches [9]. Accordingly, our framework can be complementarity to the existing BDD-based methods. Also it would be possible to construct an efficient logic design system from the combination of our method and the above-mentioned BDD-based methods [6, 16, 17, 9].

This paper is organized as follows. In Sec. II, we provide minimum information to understand the contents of this paper. Then, Sec. III is devoted to explain our proposed logic design framework. We show some experimental results to demonstrate the effectiveness of our method in Sec. IV, and discuss the comparison between our method and related work in Sec. V. Finally, we conclude the paper in Sec. VI.

## II. PRELIMINARIES

### A. RSFQ Logic Primitives

In this paper, we consider a design framework of RSFQ logic circuits. In an RSFQ circuit, single flux quantum (SFQ) pulses are used for representing logic values. Pulses are generated and propagated by the combination of super-conducting rings with Josephson junctions. There has been proposed many logic primitives to manipulate pulses in RSFQ circuit in logic level. Among them, in this paper, we use the following three logic primitives [5, 7].

**SPL (Splitter)** This logic primitive generates two pulses from a single pulse. We express this primitive as a black circle as shown in Fig. 1.

**CB (Confluence Buffer)** This logic primitive generates a single pulse when two input pulses arrive. (The two input pulses are not allowed to arrive at the very same time.) We express this primitive as a white circle as shown in Fig. 2.

**2x2-Join** 2x2-Join has four inputs and four outputs as shown in Fig. 3, and it generates one pulse at one of the four outputs depending on the combination of input pulses. The relationship between inputs and outputs are described in Table I. For example, if it receives two pulses at $A_t$ and $B_t$, then it generates a pulse at the output 11. Note that the output pulse is not dependent on the arrival order of the input pulses.

### B. Dual-Rail Logic Design

At the early times when researches for RSFQ circuit started, a clock signal was used to translate a pulse on a data line in a "clock window" as logic "1" and no pulse as logic "0" like conventional synchronous circuit design. The reason is that we need to specify the exact time when a pulse comes, or otherwise we cannot distinguish between logic "0" and logic "1" while pulse has not arrived yet. Therefore, unlike the conventional technologies, for RSFQ circuit logic design, careful delay estimation and clock design are required [4] since an improper arrival order of data and clock pulses leads to erroneous data transfer. Facing the above-mentioned timing problems, the RSFQ technology has been paying an attention to an asynchronous approach. More precisely, dual-rail data encoding is used to enables clock free data transfer [3]. In the dual-rail scheme, a pair of (true- and false-) data lines carries 1-bit binary information. The propagation of a pulse on the true-line or the false-line represents logic "1" and "0," respectively. No race occurs because only one pulse propagates either true- or false-line during 1-bit data transfer. Therefore, for a large circuit, it is indispensable to adopt dual-rail scheme, and thus, in our framework we use dual-rail scheme; we use two lines (true- line and false- line) to propagate 1-bit information of an intermediate logic function as will be mentioned in the next section.

### C. Transformation-based Optimization Methods

Since it is totally impossible to generate an optimal logic circuit by one method (especially for a large problem), we iteratively apply optimization methods to transform an initial circuit generated by some methods into a smaller circuit. This strategy should also be important for RSFQ circuit design. Thus, we utilize the Transduction Method [8], which is one of such optimization methods, in our framework for the same purpose. the Transduction Method is based on the concept of **permissible functions (PFs)**. Intuitively, a permissible function at a gate (or a connection) is a set of functions any one of which can be used at the gate (or the connection) without changing the functionality of the circuit. A compatible set of permissible functions (CSPFs) is defined as a set of PFs that can be used independently on a set of gates (and/or connections) without changing functionality. Originally CSPFs were defined for NOR gates [8]. Later, they were generalized to arbitrary Boolean nodes and called compatible observability don't cares (CODCs) [10]. CODCs differ also in that they are expressed in terms of intermediate signals. CODCs are used in SIS [13], and transformation-based optimization methods are widely used in many conventional logic synthesis tools. We refer readers to [8] for the details of the calculation of CSPFs and the transformation procedures.

## III. The Proposed Framework to Design RSFQ Logic Circuits

### A. Overview of the Proposed Framework

As mentioned in the previous section, the logic primitives for RSFQ logic circuits are different from the conventional logic gates. Therefore, it is not possible to directly apply the conventional logic design techniques for designing RSFQ logic circuits. The difficulties are summarized as follows.

- A CB works in a very similar way to a conventional OR gate, but the two input pulses are not allowed to arrive at the same time for a CB. Thus two input functions, $h_1$ and $h_2$, to a CB should satisfy the condition $h_1 \cdot h_2 = 0$.
- A 2x2-Join works as generating a logic combination with respect to its input pulses as shown in Table I. However, the behavior of a 2x2-Join is not known for the combination of input pulses that are not described in the table (e.g., when a 2x2-Join receives pulses at both $A_t$ and $A_f$). Thus, $A_t$ and $A_f$ ($B_t$ and $B_f$ also) should be exactly the negation to each other; we should always generate $\overline{f}$ as well if we want to use $f$ as an intermediate function.

By considering the above issues, we restrict ourselves to design a logic circuit by using a virtual logic cell called 2-AND/XOR which will be mentioned below. As we will see soon, using 2-AND/XOR cells solves the above issues very naturally, and moreover, it utilizes almost all the abilities of 2x2-Joins.

### B. 2-AND/XOR Cell

Before introducing our 2-AND/XOR cells, let us consider how to use 2x2-Joins to make logic functions. Suppose we want to construct $f$ with respect to two intermediate functions $h_1$ and $h_2$. As we take dual-rail scheme, we also assume that there are $\overline{h_1}$ and $\overline{h_2}$. Then, we connect $h_1$, $\overline{h_1}$, $h_2$ and $\overline{h_2}$ to $A_t$, $A_f$, $B_t$ and $B_f$, respectively, of a 2x2-Join as shown in Fig. 4. Then the outputs of the 2x2-Join, 00, 01, 10 and 11 generate pulses corresponding to the logic functions of $(\overline{h_1} \cdot \overline{h_2})$, $(\overline{h_1} \cdot h_2)$, $(h_1 \cdot \overline{h_2})$ and $(h_1 \cdot h_2)$, respectively. They are the four minterms of $h_1$ and $h_2$, and thus, any function with respect to $h_1$ and $h_2$ can be constructed by merging some of four outputs, 00, 01, 10 and 11, with CBs. We can also construct $\overline{f}$ by merging the outputs of the 2x2-Join that are not used for $f$. An example where $f = h_1 \cdot h_2$ is shown in Fig. 4. To sum up, by using a 2x2-Join with two CBs, we can construct any dual-rail logic function $f$ (both $f$ and $\overline{f}$) of two intermediate functions. Therefore, from the conventional logic point of view, we can consider this 2-input sub-circuit as a 2-input LUT (look-up table).

There is another usage of a 2x2-Join, i.e, we consider making multiple functions from a single 2x2-Join. For example, we can make AND and XOR functions of two inputs by using a single 2x2-Join, three CBs and three SPLs as shown in Fig. 5. Indeed, we can make all the possible sixteen 2-input functions at the same time by a single 2x2-Joins with many CBs and SPLs. However, since we use dual-rail logic, we do not need to consider the polarity of the inputs and the outputs of the functions, and therefore, it is enough to consider the following five functions with respect to inputs $h_1$ and $h_2$: $f_1 = \overline{h_1} \cdot \overline{h_2}$, $f_2 = \overline{h_1} \cdot h_2$, $f_3 = h_1 \cdot \overline{h_2}$, $f_4 = h_1 \cdot h_2$ and $f_5 = h_1 \oplus h_2$. (Note that we may want to implement some of these five functions *at the same time*; we need to have four functions for AND type functions at the same time.)

With the above discussions in mind, we introduce a virtual logic cell called 2-AND/XOR that has two inputs, $h_1$ and $h_2$, and five outputs, $f_1 = \overline{h_1} \cdot \overline{h_2}$, $f_2 = \overline{h_1} \cdot h_2$, $f_3 = h_1 \cdot \overline{h_2}$, $f_4 = h_1 \cdot h_2$ and $f_5 = h_1 \oplus h_2$ as shown in Fig. 6. Obviously we can design a 2-AND/XOR cell by a single 2x2-Join, CBs and SPLs. This cell corresponds to all the possible output functions that can be implemented by a single 2x2-Join. Also we can always generate negations of $f_1$ to $f_5$. It should also be noted that the usage of CBs in a 2-AND/XOR cell satisfies the condition of CBs, i.e., two input pulses do not arrive at the same time since an output pulse is generated at only one of four outputs of a 2x2-Join. Thus the problems mentioned in Sec. III-A are naturally solved if we use only 2-AND/XOR cells to design logic circuits. Moreover, a 2-AND/XOR cell also naturally corresponds to a circuit consisting of the conventional AND, XOR and NOT gates; we can utilize the conventional logic design techniques.

### C. Initial Circuits Synthesis

As mentioned in the previous section, a sub-circuit realizing any 2-input logic function can be naturally mapped

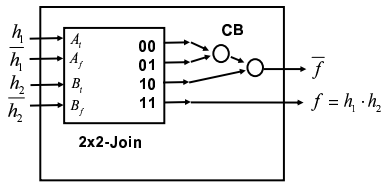


Fig. 4. 2x2-Join Usage: Single Function.

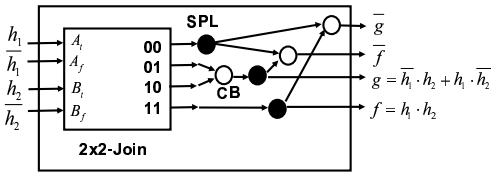


Fig. 5. 2x2-Join Usage: Multiple Functions.

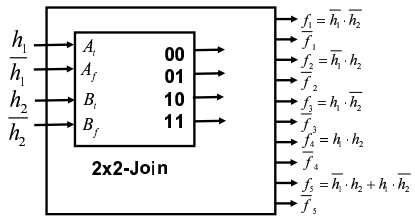


Fig. 6. 2-AND/XOR Cell.

to a 2-AND/XOR cell. Therefore, any logic circuit consisting of only 2-input nodes can be mapped to a circuit consisting of only 2-AND/XOR cells. Thus, our logic synthesis starts with a circuit consisting of only 2-input nodes. This initial circuit can be generated by conventional logic design tools, such as SIS (A System for Sequential Circuit Synthesis) [13]. For example, by using a standard script of SIS, we can obtain a circuit consisting of 2-input ANDs, 2-input XORs, and NOTs. Then, we can map this circuit naturally into one consisting of 2-AND/XOR cells. Note that four of five outputs of each 2-AND/XOR cell is not used (i.e., redundant) at this moment, however, the redundant outputs are very useful in our optimization procedure mentioned in the next section.

*D. Transduction Method for 2-AND/XOR Circuits*

After obtaining the initial circuit consisting of 2-AND/XOR cells, we apply the following procedure where we consider the whole circuit as just a conventional circuit consisting of 2-input XORs, 2-input ANDs and NOTs. However, while each AND or XOR gate is considered as a single gate in the conventional Transduction Method, the four AND gate and the XOR gate in a 2-AND/XOR cell should be handled together as we will mention.

**Step 1.** Calculate CSPFs of all connections and gates in the circuit.

**Step 2.** For an XOR gate, remove an input connection of the gate if the CSPF at the connection contains the constant-0 function. For an AND gate, remove an input connection of the gate if the CSPF at the connection contains the constant-1 function.

**Step 3.** If the function at a gate is included in the CSPF at a connection, replace the connection with a new connection from the gate.

We repeat the above transformation until there is no change.

This scheme is exactly the same as that of the conventional Transduction Method (See [8] for more details). However, note that we can remove a 2-AND/XOR cell only when all the four AND and the XOR outputs are removed. Therefore, unlike the case of conventional circuit optimization methods, it is not important to remove only one of XOR and AND gates in a 2-AND/XOR cell. Thus, we modify the conventional Transduction Method as follows. The modification is considered to be very natural for 2-AND/XOR cells.

- If there are multiple candidates for the alternative connection at Step 3, we choose the output of 2-AND/XOR whose total number of fanouts (i.e., the number of fanouts of all the four AND and the XOR outputs) is the largest. In contrast, in the conventional Transduction Method, we consider only the number of fanouts of the gate that is chosen as a replacement. This means that if we take the conventional strategy, we always consider only one of AND or XOR outputs in 2-AND/XOR cells, and therefore, we may miss a chance to remove a whole 2-AND/XOR cell even though we can remove one of its output gate.

- At Step 3, to select a candidate function, if there is a gate whose output function is $f$, we can also use $\overline{f}$ since we use dual-rail logic.

- At Step 2, we remove an AND (or XOR) gate only when all the four AND and the XOR gates in the 2-AND/XOR cell become redundant. By this modification, we can continue to have a possibility to use AND (or XOR) function to replace another connection even though it is currently not used.

It is very interesting to note that any functional redundancy cannot be obtained if we apply the CSPF calculation directly to circuits consisting of 2x2-Joins and CBs because of their difference from the conventional gates described in Sec. III-A. In other words, we can successfully utilize the Transduction Method by introducing 2-AND/XOR cells and considering the whole circuit as just a conventional circuit consisting of 2-input XORs, 2-input ANDs and NOTs. This enable us to optimize RSFQ circuits more as we will see in our experimental results in Sec. IV.

As we mentioned in Sec. III-B, we can consider the logic primitives as 2-input LUTs that can be constructed as shown in Fig. 4. For a logic circuit consisting of LUTs, there is an efficient optimization method that utilizes SPFD [14]. (SPFD is a generalization of CSPF to the case of LUTs where we can utilize the flexibility of LUTs, i.e., we can change the internal functions of LUTs.) Therefore, one might wonder if the following strategy is more natural and better than our strategy.

- Consider the logic primitives as 2-input LUTs which can be constructed as shown in Fig. 4.

- Apply SPFD-based optimization method [14].

It should be noted that the above strategy is essentially the same as the ours since it is sufficient to consider AND and XOR for 2-input functions when we use dual-rail logic. In other words, our strategy to use 2-AND/XOR cell essentially has the same power as the one that uses SPFDs. Therefore, for our problem, we do not need to calculate and manipulate SPFDs that are more complicated than CSPFs.

### *E. Optimality of Our Framework*

Our framework successfully utilizes the conventional logic design techniques by considering virtual cells called 2-AND/XOR cells in stead of considering directly 2x2-Joins, CBs and SPLs. Then, one may wonder how much we may miss the possibility to have a smaller circuit compared to the case where we design a circuit directly from 2x2-Joins, CBs and SPLs. As mentioned, as far as we consider a function realized by only a single 2x2-Join with SPLs and CBs, we do not miss the opportunity to have a small circuit, i.e., our 2-AND/XOR cell essentially represents all the possible functions realized by ORing any combination of the four outputs of a single 2x2-Join, i.e., any function with respect to two inputs of the cell.

However, we cannot generate a logic function realized by ORing two functions from two different 2x2-Joins. Suppose $h_1$ and $h_2$ be the functions realized at the outputs of two different 2x2-Joins. Then, if $h_1 \cdot h_2 = 0$ is satisfied we can make $f = h_1 + h_2$ by only a single CB. Obviously our framework cannot deal with the above usage of CBs. It should be noted that such a situation does not happen so frequently by the following reason. Since we take dual-rail scheme, if we generate $f = h_1 + h_2$, we also need to generate $\overline{f}$, i.e., we need to have $h_3$ and $h_4$ such that $h_3 \cdot h_4 = 0$ and $h_3 + h_4 = \overline{f}$. Thus, we need to find $h_3$ and $h_4$ with the above difficult conditions when we want to generate OR functions of $h_1$ and $h_2$ with a CB; there are not so many such situations. Therefore, missing the above usage of CBs may be allowed if we consider the advantage of our framework, i.e., we can adopt conventional logic design techniques.

One may also consider that our usage of 2x2-Joins seems to increase the number of CBs and SPLs since we use a lot of CBs and SPLs to generate multiple outputs from a single 2x2-Join in a 2-AND/XOR cell. (See Fig. 5 again.) However, we would like to stress that the number of CBs and SPLs are not increased, or even decreased in many cases. The reason is as follows. Consider a case when we want to generate the two functions, $f$ and $g$, which are shown in Fig. 5. If we do not use our 2-AND/XOR cells, i.e., just use two 2x2-Joins to implement two functions separately, the situation can be expressed as shown in Fig. 7. Note that we need four SPLs before two 2x2-Joins. Thus, by comparing Fig. 5 and Fig. 7, one can easily see that our usage of 2x2-Joins decreases the number of not only 2x2-Joins but also CBs and SPLs. (In this example, we can decrease the number of CBs since $\overline{f}$ and $g$ share a common minterm.)

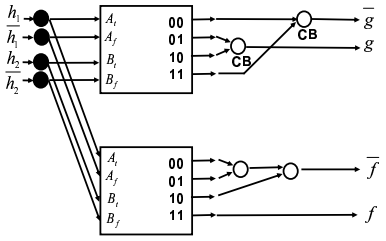


Fig. 7. Two 2x2-Joins without Sharing.

## IV. Experimental Results

We have implemented the methods presented in the previous sections and performed preliminary experiments on MCNC [15] benchmark circuits. To synthesize initial circuits we used the following recommended script of SIS [13].

(1) eliminate 2, (2) gkx -ac, (3) simplify -d,
(4) xl_part_coll -m -g 2 (5) xl_coll_ck,
(6) xl_partition -m, (7) simplify.

Table II shows the results of our optimization procedure described in Sec. III-D. In Table II, "Join," "Conn." and "Lev." show the number of 2x2-Joins, the number of connections between them and the level (depth) of the circuits, respectively, and "Time" shows optimization time our method took in these experiments. In the lowest row, with respect to the number of 2x2-Joins, the number of connections and the circuit level, we show the ratio to the initial circuits, and with respect to the optimization time, we show the average of those for the benchmark circuits. Here we consider only the number of 2x2-Joins like the existing researches by observing that the implementation cost of 2x2-Joins will be much larger than those of the other primitives although it is a little bit early to discuss the real implementation costs. It should also be noted that the numbers of CBs and SPLs are also decreased if we can decrease the number of 2x2-Joins as mentioned in Sec. III-E. In Table II, we can observe that our optimization method reduced the number of 2x2-Joins, the number of connections and the circuit level by 32.0%, 31.8% and 17.3%, respectively, while our optimization procedure took 6.08 seconds on average.

We consider that this large reduction is due to the change from the single-output LUT to the multi-output 2-AND/XOR composed of a 2x2-Join, SPLs and CBs. It is apparent that two 2-AND/XOR cells can be merged into one if their inputs are from the same 2-AND/XOR cells. Otherwise, we cannot determine whether they can be merged or not by only observing the circuit configuration. Our method can find more cases where we can merge 2-AND/XOR cells by using CSPFs. This feature makes our optimization method more powerful. The number of such mergers are reported in "Share" in Table II, and the ratio to the total number of the 2x2-Joins is in (%).

As the case of conventional logic synthesis, the logic optimization should be also very important for the RSFQ

TABLE II
EXPERIMENTAL RESULTS

| Circuits | PI | PO | Initial Circuits Join | Conn. | Lev. | Trans. Join | Conn. | Lev. | Time (s) | Shared | (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C1355 | 41 | 32 | 234 | 468 | 17 | 174 | 348 | 14 | 1.17 | 6 | 3.4 |
| C7552 | 207 | 108 | 1504 | 3059 | 32 | 994 | 2049 | 32 | 122.14 | 136 | 13.7 |
| alu2 | 10 | 6 | 386 | 773 | 42 | 236 | 473 | 28 | 3.35 | 41 | 17.3 |
| alu4 | 14 | 8 | 667 | 1334 | 43 | 479 | 958 | 36 | 11.26 | 84 | 17.5 |
| cmb | 16 | 4 | 44 | 88 | 6 | 25 | 51 | 5 | 0.01 | 4 | 16.0 |
| dalu | 75 | 16 | 1172 | 2344 | 36 | 809 | 1618 | 18 | 36.59 | 84 | 10.4 |
| f51m | 8 | 8 | 113 | 227 | 10 | 64 | 129 | 9 | 0.11 | 10 | 15.6 |
| i8 | 133 | 81 | 1260 | 2520 | 19 | 971 | 1942 | 15 | 155.37 | 127 | 13.1 |
| lal | 26 | 19 | 88 | 177 | 8 | 61 | 123 | 8 | 0.06 | 7 | 11.5 |
| my_adder | 33 | 17 | 96 | 192 | 48 | 64 | 128 | 48 | 0.08 | 16 | 25.0 |
| t481 | 16 | 1 | 1690 | 3380 | 20 | 1073 | 2146 | 19 | 110.66 | 57 | 5.3 |
| term1 | 34 | 10 | 259 | 519 | 16 | 116 | 234 | 11 | 0.68 | 15 | 12.9 |
| ttt2 | 24 | 21 | 182 | 364 | 10 | 127 | 254 | 10 | 0.41 | 19 | 15.0 |
| x3 | 135 | 99 | 724 | 1448 | 14 | 548 | 1096 | 12 | 14.19 | 41 | 7.5 |
| z4ml | 7 | 4 | 44 | 88 | 9 | 12 | 25 | 8 | 0.01 | 6 | 50.0 |
| Average | | | 100 | 100 | 100 | 68.0 | 68.2 | 82.7 | 6.08 | | 11.4 |

logic circuit design since it is difficult to design optimum RSFQ circuits directly from the specifications. From this viewpoint, our optimization method might be useful in the second step of any circuit synthesis method for RSFQ logic circuits. Moreover, although the conventional mapping tools mainly produce single-output LUT circuits, our method can transform them into multi-output 2-AND/XOR circuits. Therefore, by taking the advantage of the multi-output feature, our method can optimize the 2-input circuit mapped by SIS as the experimental results show.

## V. Comparison with Related Works

To the best of our knowledge, there are only two researches [16] and [9] for systematic methodology to design RSFQ logic circuits. Both of them are based on BDDs. Unlike our framework, they directly consider RSFQ logic primitives, Bina [16] or 2x2-Joins [9]. Although the paper [16] does not provide any results of benchmark circuits, it seems that the method proposed in [9] is more promising as they claim in [9].

Although we do not have comparison data, we can observe the disadvantages of the BDD-based methods as follows:

- BDD-based methods need relatively large time (compared with conventional logic synthesis techniques used in our framework) to manipulate the BDDs in a special manner.
- BDD-based methods need to consider BDD variable ordering and/or the division of functions if necessary, and the variable ordering should have a great influence on the resultant circuits.
- The level of the generated circuits is essentially the same as the number of inputs of the functions to be synthesized if we use the BDD-based methods.

Note that it is well-known that, for some functions, it is much better to use functional decomposition based on BDDs [11] than to use conventional logic design techniques, such as SIS. Thus, it might be true that the BDD-based methods should produce better circuits than our method especially for small circuits. However, it is also obvious that for some cases our method may be better. It is also true that our method can be applied for larger circuits to which BDD-based methods are not feasible. In conclusion, our method is complementarity to the existing BDD-based methods.

The obvious advantages of the method [9] over our method is the following. Our method cannot find the special usage of CBs as mentioned in Sec. III-E. On the other hand, the method [9] sometimes extracts the special usage of CBs from their BDD representation since the method constructs circuits directly from RSFQ logic primitives. As we mentioned, by ignoring this special usage of CBs, we can successfully utilize the conventional logic techniques in our framework. Note also that the optimization phase in our framework can handle this special CBs by treating them as OR gates with some special conditions. Thus, it is easy to modify our Transduction Method so that it can be applied to the circuit obtained by the method [9]. Such modifications and further comparison with the BDD-based methods should be done as our future work.

## VI. Conclusions

In this paper, we have presented a new framework to synthesize RSFQ logic circuits from 2x2-Joins. Our method can utilize conventional logic design techniques.

Indeed we show how to use logic synthesis tools such as SIS [13] and a transformation-based heuristic method based on the Transduction Method [8] in our framework.

Our contributions are summarized as follows.

- We propose a framework to use a 2x2-Join as a 2-AND/XOR cell for RSFQ logic circuit synthesis. By this we can utilize the conventional logic synthesis techniques including initial circuit design and optimization methods.
- We propose an optimization method by modifying the original Transduction Method so that it can utilize the property of 2-AND/XOR cells.
- By using our framework, we can successfully generate RSFQ circuits in reasonable time for large benchmark circuits that have not been reported in the existing researches.

The experimental results show that our Transduction Method reduces the number of 2x2-Joins by 32.0% on average.

There are other logic primitives for RSFQ circuits, such as RSFQ $D_2$ flip-flop. Our future work is to treat other logic primitives in our framework. Then, we would also like to combine our method with the existing methods [6, 16, 17] that use RSFQ $D_2$ flip-flops for their primitives.

## Acknowledgment

We are deeply grateful to the late Professor Yahiko Kambayashi for his initiation and encouragement of our research philosophy through the study of the Transduction Method. The first author is supported by MEXT.KAKENHI ((B) 16700067) and the Okawa Foundation Research Grant.

## References

[1] International Technology Roadmap for Semiconductors. Technical Report http://public.itrs.net/, 2004.

[2] R. E. Bryant. Graph-based algorithm for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):667–691, August 1986.

[3] J. Deng, S. Whiteley, and T. Van Duzer. Data-Driven Self-Timing of RSFQ Digital Integrated Circuits. In *Extended Abstracts of ISEC'95*, pages 189–191, September 1995.

[4] K. Gaj, E. G. Friedman, and M. J. Feldman. Timing of multi-gigahertz rapid single flux quantum digital circuits. *IEEE Journal of VLSI Signal Processing*, 16(2-3):247–276, 1997.

[5] Y. Kameda, S.V. Polonsky, M. Maezawa, and T. Nanya. Self-timed Parallel Adders based on DI RSFQ Primitives. *IEEE Trans. Appl. Superconductivity*, 9(2):4040–4045, June 1999.

[6] J. Koshiyama and N. Yoshikawa. A Cell-Based Design Approach for RSFQ Circuits Based on Binary Decision Diagram. *IEEE Trans. Appl. Superconductivity*, 11(1):263–266, March 2001.

[7] 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. Superconductivity*, 1(1):3–28, March 1991.

[8] S. Muroga, Y. Kambayashi, H. C. Lai, and J. N. Culliney. The Transduction Method - Design of Logic Networks Based on Permissible Functions. *IEEE Transactions on Computers*, 38(10):1404–1424, October 1989.

[9] K. Obata, K. Takagi, and N. Takagi. Design Method of Dual-Rail RSFQ Logic Circuits Using 2x2-Join. *IEICE Trans.*, J88-C(3):202–209, March 2005. (In Japanese)

[10] H. Savoj and R. K. Brayton. The Use of Observability and External Don't Cares for Simplification of Multi-Level Networks. pages 297–301, June 1990.

[11] H. Sawada, T. Suyama, and A. Nagoya. Logic Synthesis for Look-up Table Based FPGAs Using Functional Decomposition and Support Minimization. pages 353–358, November 1995.

[12] V. K. Semenov, Yu. A. Polyakov, and D. Schneider. Implementation of Oversampling Analog-to-Digital Converter Based on RSFQ Logic. In *Extended Abstracts of ISEC'97*, volume 1, pages 41–43, June 1997.

[13] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli. SIS: A System for Sequential Circuit Synthesis. Technical Report UCB/ERL M92/41, Univ. of California, Berkeley, May 1992.

[14] S. Yamashita, H. Sawada, and A. Nagoya. SPFD: A New Method to Express Functional Permissibilities. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 19(8):840–849, August 2000.

[15] S. Yang. Logic synthesis and optimization benchmarks user guide version 3.0. *MCNC*, January 1991.

[16] N. Yoshikawa and J. Koshiyama. Top-Down RSFQ Logic Design Based on a Binary Decision Diagram. *IEEE Trans. Appl. Superconductivity*, 11(1):1098–1101, March 2001.

[17] N. Yoshikawa, H. Tago, and K. Yoneyama. A New Design Approach for RSFQ Logic Circuits Based on the Binary Decision Diagram. *IEEE Trans. Appl. Superconductivity*, 9(2):3161–3164, June 1999.