社団法人　情報処理学会　研究報告
IPSJ SIG Technical Reports

2004－SLDM－117 (39)
2004／12／2

RSFQ 論理回路の回路変形による論理設計

山下　茂[†]　田中　克典[††]　高田　秀志[††]

† 奈良先端科学技術大学院大学 情報科学研究科
〒 630-0192 生駒市高山町 8916-5
†† 京都大学大学院 情報学研究科
〒 606-8501　京都市左京区吉田本町
E-mail: †ger@is.naist.jp, ††{ktanaka,htakada}@db.soc.i.kyoto-u.ac.jp

あらまし　本稿では，rapid single flux quantum (RSFQ) 論理回路向けの新しい論理設計手法を提案する．提案する手法は,「2x2-Join」と呼ばれるセルから「2x2-AND/XOR」と名付けた 2 入力 2 出力の論理セルを作成することを基本としている．具体的には，2x2-AND/XOR からなる初期回路を作成し，それをトランスダクション法に基づく回路変形により最適化を行なう．提案手法では、オリジナルのトランスダクション法を 2x2-AND/XOR セルの性質を効率良く利用できるように修正して使用している．SIS により作成した初期回路に提案手法の最適化を適用したところ，回路規模を 32.0%削減することができた．

キーワード　RSFQ，トランスダクション法，論理設計

Transformation-Based Logic Design for RSFQ Logic Circuits

Shigeru YAMASHITA[†], Katsunori TANAKA[††], and Hideyuki TAKADA[††]

† Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, JAPAN
†† Graduate School of Informatics, Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, JAPAN
E-mail: †ger@is.naist.jp, ††{ktanaka,htakada}@db.soc.i.kyoto-u.ac.jp

Abstract　In this paper, we propose a new method to synthesize rapid single flux quantum (RSFQ) logic circuits. We propose to use a primitive logic cell called "2x2-Join" to make a two-input and two-output logic cell, which we call "2x2-AND/XOR." Our method starts with the initial circuits consisting of 2x2-AND/XORs, and optimizes them by using a transformation based heuristic method based on the Transduction Method. In our transformation method, we modify the original Transduction Method so that we can fully utilize the property of 2x2-AND/XORs. Our experimental results show that we can achieve 32.0% reduction on average from the initial circuits generated by SIS.

Key words　RSFQ, Transduction Method, Logic Design

1. 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 [1]. RSFQ technology has the following features [1].

- 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. However, there is a 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" [2], the RSFQ technology is considered as one of the promising next generation technologies. Actually RSFQ digital circuits containing several thousands of Josephson junctions have been successfully implemented and their high performance has been confirmed [3].

Up to the present, for RSFQ circuits, there have been many researches for devices and design methods of some primitive logics cells, but few researches for logic design methods of large circuits. Among the researches for logic design methods for RSFQ circuits, the cell based methods [4]~[6] 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 Binary Decision Diagram [7]. In this paper, we use another primitive called "2x2-Join" [8] and propose a new method to synthesize RSFQ logic circuits from 2x2-Joins. It would be possible to construct an efficient logic design system from the combination of our method and the above-mentioned cell based methods [4]~[6].

To make an efficient logic design methods for larger circuits, it is very common to adopt *transformation based heuristic methods* like the Transduction Method [9] for conventional circuit design. One of the reasons is that the exact optimization of a large circuit is impossible, and therefore, we reduce the circuit size by transforming the initial circuits in a heuristic manner. The Transduction Method [9] simplifies a logic circuit based on the concept of **permissible functions (PFs)**. A permissible function expresses the condition that should be satisfied by the logic function to keep the output functions of the whole circuit. By using permissible functions, we can transform circuits consisting of the conventional AND/OR/NOT gates. The concept of PFs is applicable not only to the conventional logic gates but also to the other types of gates that are used in our method.

In this paper, we propose a transformation-based heuristic optimization method for RSFQ circuits like the conventional logic circuit design methods mentioned above. To do so, we carefully exploit the property of 2x2-Joins, and contrive how to construct logic circuits from them and how to apply a transformation method to them. It should be noted that the situation is a little bit different from that of the conventional AND/OR/NOT gates, and thus we need to modify some parts in a conventional transformation method, as we will mention in the rest of this paper.

This paper is organized as follows. In Section 2, we summarize the concept of CSPFs since it is useful to understand our method. In Section 3, we describe how logic primitives of RSFQ circuit work, and how to construct logic circuits from them. Then, Section 4 is devoted to explain our proposed logic design methods. We show some experimental results to demonstrate the effectiveness of our method in Section 5. Finally, we conclude the paper in Section 6.

2. Compatible Sets of Permissible Functions (CSPFs) and Their Utilization for Logic Optimization

The Transduction Method [9] simplifies a logic circuit based on the concept of **permissible functions (PFs)**. Intuitively, a completely specified function g is said to be a PF at a point (connection or gate's output), if no primary output function is undesirably changed even when the function realized at the point is changed to g. At each connection or gate, generally there exist two or more PFs, and so we can consider a set of PFs. There have been proposed two types of such sets, the **maximum set of PFs (MSPF)** and a **compatible set of PFs (CSPF)**. CSPFs can be used to transform a circuit at multiple points simultaneously unlike MSPFs, and therefore, we usually use CSPFs preferably to MSPFs in logic optimization.

CSPF can be expressed by an incompletely specified function whose values are 0, 1, or $*$ (*don't-care*). Let the number of primary inputs of a circuit be n. Then, the logic function realized at each point in the circuit can be specified by a truth table consisting of 2^n elements (0 or 1). Below, for simplicity, we consider 4 elements of them, and express a logic function and CSPF by using vectors, like (0110) and (011$*$), respectively. If one element of the CSPF at a point in a circuit is $*$, then the corresponding element of the function at the point can become either 0 or 1 without any undesirable change of the primary output functions. For example, if the CSPF at some point is (011$*$), both (0110) and (0111) are PFs for the point, i.e., both functions can be allowed to replace the function at the point. In other words, at each point, the CSPF expresses the condition for circuit transformation without any undesirable change of the primary output functions.

CSPF at each input connection of a gate is calculated from CSPF at its output and the functions at its input connections. Let us explain how CSPFs are calculated by using an example where an AND gate has two input connections. Let the CSPF at the output of the gate be (10*0), and let the input functions be (1100) and (1010). Let us see the first elements of the expressions. The CSPF at the output is 1. This means that both input function values must be 1 since the gate is AND. Namely, both of the input CSPF values

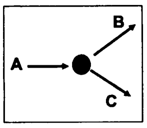

Fig. 1 SPL

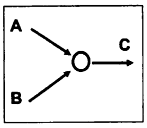

Fig. 2 CB

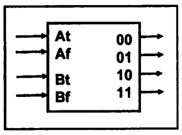

Fig. 3 2x2-Join

Tab. 1 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

are calculated as 1. With respect to the second values, since the output CSPF value is 0, at least one of the input CSPF values must be 0 and all the other values can be $*$. Since the element of the second function is 0, the element of the CSPF for the first input is calculated as $*$. With respect to the third elements, since the output CSPF value is $*$, both input CSPF elements are calculated as $*$. Finally, with respect to the fourth elements, the output CSPF and both input function values are 0. In this case, as mentioned above, at least one of the input CSPF values must be 0 and all the other values can be $*$. Now the both function values are 0, and hence, we assign priorities to input connections beforehand so that we can determine the element of which input connection should be 0 (and the other elements can be $*$). See [9] for more detail.

This calculation method can be easily generalized to any gate types, and therefore we can calculate CSPFs in a circuit consisting of AND, XOR and NOT, which will be used in our method.

By using CSPFs, we can determine whether a connection can be removed or replaced with another one in the following manner.

Removal When a gate is an OR, NOR or XOR gate, we can remove a input connection of the gate if the CSPF at the connection contains the constant-0 PF (all the elements of the expression are 0 or $*$). When a gate is an AND or NAND gate, we can remove a input connection of the gate if the CSPF at the connection contains the constant-1 PF.

Replacement If the function at a gate is included in the CSPF at a connection, then the connection can be replaced with a new connection from the gate.

We can optimize circuits by applying repeatedly the removal and replacement in a heuristic manner.

3. Our Circuit Model: RSFQ Logic Circuit

3.1 RSFQ Logic Primitives

We can generate and propagate single flux quantum (SFQ) pulses, as desired, by the combination of superconducting rings with Josephson junctions in RSFQ circuits. 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 [1], [8].

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

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

2x2-Join 2x2-Join has four inputs and four outputs as shown in Figure 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 1. For example, if it receives two pulses at A_t and B_t, then it generates a pulse at the output 11.

3.2 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 [10] 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 [11]. 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. In this paper, we also take dual-logic scheme, and therefore, we use two lines (true- line and false- line) to propagate 1-bit information of an intermediate logic function. We will mention our circuit model more precisely and formally in the next section.

3.3 Our Circuit Model and Problem Formulation

As mentioned in the previous section, we take dual-rail logic scheme, and therefore, our circuit model is specified as follows:

Dual-Rail RSFQ Logic Circuit

The inputs of RSFQ logic circuits are dual-rail, therefore, there are two lines (true- and false-) for each logical input. We denote the two lines as x_t and x_f for input x. The outputs of circuits are also dual-rail. Therefore, we have two outputs for each logical output. The outputs are also denoted as y_t and y_f for the logical output y. For the simplicity, we also restrict ourselves to construct a whole circuit from sub-circuits whose inputs and outputs are also dual-rail logic inputs. Then, our problem is described formally as follows:

Input m specified logic function $f_1, f_2, \cdots, f_m$.

Output A dual-rail logic circuit that realizes $f_1, f_2, \cdots, f_m$ and the negation of them. Our objective is to construct the desired circuit with as few logic primitives as possible. In particular, in this paper, we focus on the number of 2x2-Joins.

4. Transformation-Based Logic Design of RSFQ Circuits

In this section, we describe our logic design method of RSFQ circuits.

4. 1 2-Input Dual-Rail Logic Sub-Circuits

In our method to synthesize RSFQ logic circuits, we synthesize all sub-circuits as dual-rail logic circuits, as mentioned in Section 3. 3. More concretely, we generate 2-input dual-rail logic sub-circuits by our logic primitives, and then construct the entire circuit from the 2-input sub-circuits.

Our construction of 2-input sub-circuits is as follows. Suppose we want to construct f with respect to two intermediate inputs h_1 and h_2. In our construction, we always make any intermediate functions as dual-rail logic, and therefore, we also have h_1' and h_2'. We connect h_1, h_1', h_2 and h_2' to A_t, A_f, B_t and B_f, respectively, of a 2x2-Join as shown in Figure 4. Then the outputs of the 2x2-Join, 00, 01, 10 and 11 generate pulses corresponding to the logic functions of $(h_1' \cdot h_2')$, $(h_1' \cdot h_2)$, $(h_1 \cdot h_2')$ 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, by using CBs. We can construct 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 Figure 4. To sum up, by using a 2x2-Join with two CBs, we can construct any dual-rail logic function f of two intermediate functions. That is, we can consider this 2-input sub-circuit as a 2-input LUT (look-up table).

In this paper, we consider another utilization of 2x2-Join, that is, 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 Figure 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 output of the functions, and therefore, it is enough to implement AND and XOR for 2-input functions.

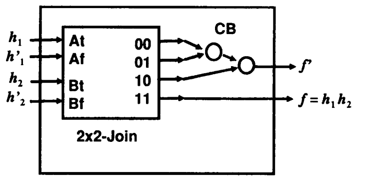

Fig. 4 2x2-Join Usage (1)

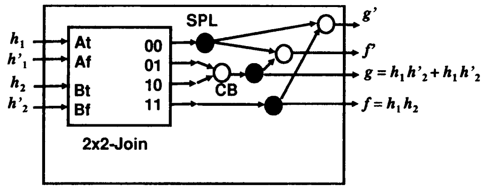

Fig. 5 2x2-Join Usage (2): 2x2-AND/XOR Cell

Considering the above discussions, our logic design method constructs a circuit by using the logic primitives that have two outputs of AND and XOR of its two inputs. We call this logic primitive "2x2-AND/XOR," and how to construct a circuit from 2x2-AND/XORs will be given in the following sections. Note that 2x2-AND/XOR cell is exactly the same as the cell shown in Figure 5.

4. 2 Initial Circuits Synthesis

As mentioned in the previous section, we can construct a sub-circuit that realizes any 2-input logic function. Therefore, any logic circuit consisting of only 2-input nodes can be naturally mapped to our RSFQ logic circuit. 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) [12]. 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 2x2-AND/XORs. Note that one of two outputs of each 2x2-AND/XOR 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.

4. 3 Transduction Method for 2x2-AND/XOR Circuits

After obtained the initial circuit consisting 2x2-AND/XOR circuits, we apply the following procedure where we consider the whole circuit as a conventional circuit consisting of just 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 AND gate and XOR gate in a 2x2-AND/XOR cell should be handled as a pair, as we will mention.

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

Step 2 Remove redundant connections and gates by using CSPFs.

Step 3 Replace a connection with another connection by using CSPFs.

We repeat the procedure until there is no change.

This scheme is almost the same as that of the conventional Transduction Method. However, note that we can remove a 2x2-AND/XOR cell only when the both AND and XOR outputs are removed. Therefore, unlike the case of the conventional circuit optimization, it is not important to remove only one of XOR or AND of a 2x2-AND/XOR cell. Therefore, the unique features of our strategy compared with the conventional Transduction Method are the followings:

- If there are multiple candidates for the alternative connection at Step 3, we choose the output of 2x2-AND/XOR such that whose total number of fanouts (i.e., the number of fanouts of the both AND and XOR outputs) are relatively large. 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 and XOR outputs of 2x2-AND/XOR cells, and therefore, we may miss a chance to remove a whole 2x2-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 f' since we use dual-rail logic.
- At Step 2, we remove an AND (or XOR) gate only when the corresponding XOR (or AND) gate of the same 2x2-AND/XOR cell is also 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. If the both gates becomes redundant, we remove the 2x2-AND/XOR cell.

As we mentioned in Section 4. 1, we can consider the logic primitives as 2-input LUTs that can be constructed as shown in Figure 4. For a logic circuit consisting of LUTs, there is an efficient optimization method that utilizes SPFD [13]. (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 primitive as 2-input LUTs which can be constructed as shown in Figure 4.
- Apply SPFD-based optimization method [13].

We would like to note 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 2x2-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. Moreover, to treat the possibility of having multiple outputs from single 2x2-Join cell as shown in Figure 5, we need to modify the original SPFD calculation. In contrast, we do not need to modify the way of calculation of CSPFs although we need to take a little different strategy to replace connections as mentioned above (however, apparently it is not difficult to implement our strategy compared with the original strategy).

5. Experimental Results

We have implemented the methods presented in the previous sections and performed preliminary experiments on MCNC [14] benchmark circuits. To synthesize initial circuits we used the following recommended script of SIS [12].

(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 2 shows the results of our optimization procedure described in Section 4. 3. In Table 2, "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 our method, several 2x2-Joins are shared to realize multiple functions. In Table 2, "Share" shows the number of shared 2x2Joins, and the percentage is the ratio to the number of the total 2x2-Joins. In the lowest row, with respect to the number of 2x2-Joins, the number of

Tab. 2 Experimental Results

			Initial Circuits			Trans.					
Circuits	PI	PO	Join	Conn.	Lev.	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

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. In Table 2, we can observe that our 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. This high performance is realized by an important our method's feature that 11.4% of 2x2-Joins can be used to realize multiple functions on average.

We consider that this large reduction is due to the change from the single-output LUT to the multi-output 2x2-AND/XOR composed of a 2x2-Join, SPLs and CBs. It is apparent that two 2x2-AND/XORs can be merged into one if their inputs are from the same 2x2-AND/XORs. Otherwise, we cannot determine only from the circuit configuration whether they can be merged or not. However, from the CSPFs and the functions at the two 2x2-AND/XORs, we can determine if they can be merged. Our method realizes this determination based on the CSPFs. This feature makes our method more powerful.

As the case of conventional logic synthesis, the logic optimization should be also very important for the RSFQ 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 for RSFQ logic circuits. Moreover, although the conventional mapping tools mainly produce single-output LUT circuits, our method can transform them into multi-output 2x2-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.

6. Conclusions and Future Work

In this paper, we have presented a new method to synthesize RSFQ logic circuits from 2x2-Joins. Our method adopts a transformation based heuristic method based on the Transduction Method [9]. Our contributions would be the followings:

- We propose to use a 2x2-Join as a 2x2-AND/XOR cell for RSFQ logic circuit synthesis.
- We propose an optimization method by modifying the original Transduction Method so that it can utilize the property of 2x2-AND/XOR cells.

The experimental results show that our method reduces the initial circuit size by 32.0% on average. In this paper we only consider the number of 2x2-Joins, but it is almost obvious that the number of CBs and SPLs are also decreased if we can decrease the number of 2x2-Joins.

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 method. Then, we would also like to combine our method with the existing methods [4]~[6] that use RSFQ D_2 flip-flops for their primitives.

Acknowledgement We would like to express our gratitude to Emeritus Prof. Saburo Muroga and the late Prof. Yahiko Kambayashi for gracious supervision and guidance.

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. Superconductivity*, **1**, *1, pp. 3–28 (1991).*

[2] "International Technology Roadmap for Semiconductors", Technical Report http://public.itrs.net/ (2003).

[3] V. K. Semenov, Y. A. Polyakov and D. Schneider: "Implementation of Oversampling Analog-to-Digital Converter Based on RSFQ Logic", Extended Abstracts of ISEC'97, Vol. 1, pp. 41–43 (1997).

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

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

[6] 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, pp. 3161–3164 (1999).*

[7] R. E. Bryant: "Graph-based algorithm for Boolean function manipulation", IEEE Trans. Comput., **C-35**, *8, pp. 667–691 (1986).*

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

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

[10] 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, pp. 247–276 (1997).*

[11] J. Deng, S. Whiteley and T. V. Duzer: "Data-Driven Self-Timing of RSFQ Digital Integrated Circuits", Extended Abstracts of ISEC'95, pp. 189–191 (1995).

[12] 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 (1992).

[13] S. Yamashita, H. Sawada and A. Nagoya: "SPFD: A New Method to Express Functional Permissibilities", IEEE Trans. Comput.-Aided Design Integrated Circuits, **19**, *8, pp. 840–849 (2000).*

[14] S. Yang: "Logic synthesis and optimization benchmarks user guide version 3.0", MCNC (1991).