FDP-64 / ISS 2004

Submitted 25 November 2004

Consideration of Logic Synthesis and Clock Distribution Networks for SFQ Logic Circuits

A. Akimoto¹, Y. Yamanashi¹, N. Yoshikawa¹, A. Fujimaki², S. Yorozu³, and H. Terai⁴

¹Department of Electrical and Computer Engineering, Yokohama National University, Yokohama 240-8501

²Department of Quantum Engineering, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8603

³SRL-ISTEC, 34 Miyukigaika, Tsukuba 305-8501

⁴NICT, 588-2 Iwaoka, Nishi-ku, Kobe 651-2492

Abstract

In this paper we have considered how to perform logic synthesis of the single-flux-quantum (SFQ) logic circuits by using logic synthesis tools for semiconductor logic circuits. Fundamental difference of the SFQ logic gates from the semiconductor logic gates in terms of logic synthesis is that typical SFQ gates are clocked gate, where gates have to be clocked by SFQ pulses. In our approach, we divided the procedure into two steps; logic synthesis and clock distribution. In the logic synthesis step, the logic function is optimized using a logic synthesis tool without considering the clock network. The Design Analyzer provided from Synopsys is used in this study. We have developed an SFQ cell library for the logic synthesis. In the clock distribution step, clock networks for every clocked SFQ gates are designed taking account of timing. We used a clock-followed-data clocking scheme in this study. We have classified every clocked gate into stages, where the clocked gates in the same stage are clocked simultaneously. To demonstrate the validity of our approach, we

have designed a controller of an 8-bit SFQ microprocessor.

PACS code: 85.25.Hv, 85.25.Na, 85.40.Bh

Keywords: logic synthesis, SFQ circuit, clock distribution, microprocessor, superconducting circuit, cell library

*Corresponding author.

Dr. Nobuyuki Yoshikawa

Postal address: Department of Electrical and Computer Engineering, Yokohama National University, Tokiwadai 79-5, Hodogaya, 240-8501, Japan

Phone: +81-45-339-4259

Fax: +81-44-338-1157

E-mail address: yoshi@yoshilab.dnj.ynu.ac.jp

1. Introduction

Recent progress in the design methodology of single-flux-quantum (SFQ) logic circuits [1] enables us to make large-scale SFQ circuits composed of several thousands of Josephson junctions [2]. However it is hard for circuit designers to design and optimize logic circuits including more than hundreds of thousands of logic gates without a logic synthesis tool. Our next target is, therefore, to establish a logic synthesis environment in our top-down design flow.

In this paper we have adapted a logic synthesis tool utilized in the semiconductor VLSI design for use in logic synthesis of SFQ logic circuits. One fundamental difference between SFQ and semiconductor logic circuits is that typical SFQ logic gates, such as AND, NOT and XOR, are clocked gates. This means that they need SFQ clock pulses to output their data, while the static CMOS logic gates do not need clock pulses. This prevents us from making a logic synthesis environment of SFQ logic circuits by directly mapping the technology library of the semiconductor logic synthesis tool. A proposed procedure of the SFQ logic synthesis consists of two steps, logic synthesis and clock distribution. The former step can be performed by using the semiconductor logic synthesis tools and by mapping the technology library into the SFQ circuits. We have developed an SFQ cell library for the logic synthesis for that purpose. In the latter step, a clock distribution network is generated taking account of timing between the clock and the data. In this study we have considered how to make the clock distribution network assuming clock-followed-data clocking in order to obtain the high throughput.

2. A Top-Down Design flow of SFQ Circuit Design

A proposed design flow of the SFQ logic circuits using a logic synthesis tool is shown in Fig. 1. At first we define a circuit function at the register transfer level using hardware description language (HDL). Then we simulate the behavior of the logic circuits to confirm their correct functionality using a logic simulator. If the function is correct, we perform logic synthesis using the semiconductor logic synthesis tool and the cell library for the SFQ logic synthesis. In this step, gate level net lists are generated from the HDL description. Because the semiconductor logic synthesis tool does not generate a clock distribution network for the SFQ clocked gates, we will move to the next step to make the clock distribution network. In this step, all SFQ clocked gates are divided into stages and clock trees are designed to distribute clock signals to each stage. In our approach, all SFQ clocked gates in the same stage are clocked simultaneously. Timing between clock signals and data signals are also adjusted in this step. Finally, physical placing and wiring of gates are performed according to the gate level net list. In this study, we assumed that all SFQ gates are connected by passive transmission lines whose area and propagation delay are negligible. In the following sections, we will describe the details of the two main steps in our design flow, the logic synthesis and the clock distribution.

3. Logic Synthesis

The logic synthesis is a part of the top-down design flow in semiconductor VLSI design. In this step, complex logic functions are reduced and optimized with some design constraints (area, delay, etc.) and gate level net lists are generated as an output. In this study, we have used a logic synthesis tool, Design Analyzer, provided from Synopsys Inc.

We have made a cell library for the SFQ logic synthesis based on the CONNECT

cell library [3]. The SFQ cell library for the logic synthesis consists of six gates as listed in Table. 1, where AND, NOT and XOR are clocked gates and CB, SPL2 and SPL3 are unclocked gates. CB in Table. 1 is an SFQ confluence buffer, whose function is the same to OR in the semiconductor logic gate. SPL2 and SPL3 are SFQ pulse splitters with two and three outputs, respectively. Area in Table. 1 shows physical area of the gates in the unit area of $40\text{ }\mu\text{m} \times 40\text{ }\mu\text{m}$. Delay of the clocked gate means a time interval between a clock input and a data output, and delay of the unclocked gates is a time interval between a data input and a data output. An example of the cell definition for the logic synthesis is shown in Fig. 2, which defines the CB cell in the SFQ cell library. In the cell description, we can define its function, number of input and output pins, fan-out of pins, a gate area, a delay between clock and output, and so on. In our SFQ cell library, fan-out of all pins is fixed to unity, because SPL2 or SPL3 gates have to be used to make fan-out in the SFQ logic circuits. We have confirmed that any logic function can be well synthesized by using six gates shown in Table. 1.

4. Clock Distribution

In this section, we will consider how to make a clock distribution network using the result of the logic synthesis. As shown in Fig. 3, all clocked SFQ gates in the logic circuits are divided into stages in our approach. The gates in the same stage are clocked simultaneously. Clock trees constructed with SPL2 distribute clock pulses to every clocked gate in the stages. Because we assume the clock-followed-data clocking scheme in order to get high throughput, delay elements are added between the clock trees to match the

traveling time of the data with that of the clock. The design of the clock distribution network consists of three steps: classification of clocked gates into stages, timing adjustment between stages, and timing adjustment of clock trees.

4.1 Classification of Clocked Gates into Stages

In this step, we check the order of the clocked gates in the logic circuits by tracing data paths from input ports to output ports, and classify them into the stages. Fig. 4 shows how to decide the stage number of the clocked gate. When input signals are applied to the circuit, the signals arrive at AND and NOT at first. Therefore they are classified into the first stage. Next the signal arrive at OR, but it is not classified into the stage because it is an unclocked gate. Finally data arrive at NOT, and it is classified into the second stage.

There is an exception in above procedure. An example is shown in Fig. 5 (a). When we trace a data path from the upper input port, XOR is classified into the first stage, while it is classified into the second stage if tracing from the lower two ports. In the case of this contradiction, we have to put D-FF into the data path as shown in Fig. 5 (b). Consequently, AND and D-FF are classified into the first stage and XOR is classified into the second stage.

We also have to adjust the stage number at the output ports if they are different each other. Fig. 6 (a) shows an example of the case. In the figure, the upper output port is connected to the gate in the fourth stage, while the lower output port is connected to the gate in the fifth stage. In this case, we have to add D-FF as shown in Fig. 6 (b).

4.2. Timing Adjustment between Stages

In our approach, all clocked gates in the same stage are clocked at the same time. In this case, propagation delays of the data from the previous stage to the next stage have to be equalized to increase the throughput of the logic circuit. In other words, all data have to arrive at the stages simultaneously. The propagation delays of signals in data paths between the stages t_1 are the sum of the following factors:

- (i) A gate delay of the clocked gate in the previous stage, t_p .
- (ii) A delay caused by the unclocked gates between the stages, t_u .
- (iii) A hold time of the clocked gate in the next stage, t_h .

A hold time is the minimum acceptable time interval from the data input to the clock input of the clocked gate. When the propagation delays of the signals in the data paths t_1 differ each other, delays have to be added to the data path with a shorter propagation delay in order to equalize the delay.

4.3. Timing Adjustments of Clock Trees

In this step, we design the clock trees to distribute clock pulses to every gate in the stages. In our design the clock trees are constructed with SPL2. The size of the clock tree is determined by the number of clocked gates in the stage. Assuming M is a number of clocked gates in one stage and L is a depth of the clock tree, the following relationship gives L .

$$2^{L-1} < M \leq 2^L$$

The delay t_d , which has to be added to the clock line between the clock trees (see Fig. 3), is calculated by

$$t_d = t_1 - t_2 - t_3$$

by using the following factors:

- (i) The propagation delay of the signal in the data paths, t_1 .
- (ii) Delay of the clock caused by the difference of the depth of the clock tree,

$$t_2 = t_2' - t_2''$$

- (iii) Delay caused by SPL2 in the clock line, t_3 .

5. A Case Study

In order to show the validity of our design approach, we have designed a controller of an 8-bit SFQ microprocessor, CORE1 β [4], using the SFQ cell library for the logic synthesis. Fig. 7 shows the schematic generated by using the logic synthesis and clock distribution methods proposed here. It has 11 input ports and 12 output ports, and consists of four stages. The total junction number and gate number are 1702 and 397, respectively. We have simulated its operation using a logic simulator [5] and confirmed its correct operation. It is found from the simulation that the delay from a clock input to a data output is 367 ps, and the maximum and typical clock frequencies are 42 GHz and 23 GHz, respectively, where we assume 0 ps and 20 ps timing margin for the evaluation of the maximum and typical frequencies.

6. Conclusion

We considered how to design large-scale SFQ logic circuits using a logic synthesis tool. Our approach is divided into two steps: logic synthesis and clock distribution. In the logic synthesis step, we have constructed an SFQ cell library for the logic synthesis by

mapping the technology library of the semiconductor logic synthesis tool into the SFQ circuits. In the clock distribution step, we have proposed the procedure to design a clock distribution network assuming the clock-followed-data clocking scheme. In order to show the validity of our design approach, we have designed a controller of the SFQ microprocessor, CORE1 β and confirmed its operation at 42GHz by logic simulations.

7. Acknowledgments

This work was supported by NEDO through ISTEK as Collaborative Research and Superconductors Network Device Project, and also supported by VDEC, the University of Tokyo in collaboration with Synopsys, Inc. We are grateful to all members in the CONNECT team for their help.

References

- [1] K. K. Likharev, V. K. Semenov, IEEE Trans. Appl. Supercond. 1 (1993) 3.
- [2] H. Hayakawa, N. Yoshikawa, S. Yorozu and A. Fujimaki, Proceedings of the IEEE. 92 (2004) 1549.
- [3] S. Yorozu, Y. Kameda, H. Terai, A. Fujimaki, T. Yamada, S. Tahara, Physica C 378 – 381 (2002) 1471.
- [4] M. Tanaka, T. Kondo, T. Kawamoto, Y. Kamiya, K. Fujiwara, Y. Yamanashi, A. Akimoto, A. Fujimaki, N. Yoshikawa, H. Terai, S. Yorozu, to appear in Physica C (2005).
- [5] F. Matsuzaki, N. Yoshikawa, M. Tanaka, A. Fujimaki, Y. Takai, Physica C 392 – 396 (2003) 1495.

Figure Captions

Fig. 1. A top-down design flow of large-scale SFQ logic circuits using a logic synthesis tool.

Fig. 2. An example of a cell definition in a cell library for logic synthesis. It describes the definition of the CB cell.

Fig. 3 Design of the clock distribution network based on the clock-followed-data clocking.

Fig. 4 Division of clocked gates into the stages.

Fig. 5 (a) An exception in dividing clocked gate into the stages.

(b) Adjustment of a stage number by putting a D-FF into a data path.

Fig. 6 (a) An example of the stage at the output port.

(b) Adjustment of a stage number at the output port by putting a D-FF into data path.

Fig. 7 A schematic of the controller of the SFQ microprocessor, CORE1 β .

Fig. 1

```

cell(or) {
  area : 1;
  pin(A) {
    direction : input;
    capacitance : 1;
    fanout_load : 1;
  }
  pin(B) {
    direction : input;
    capacitance : 1;
    fanout_load : 1;
  }
  pin(Z) {
    direction : output;
    max_fanout : 1;
    function : "A|B";
    timing() {
      intrinsic_rise : 14.3;
      intrinsic_fall : 18.1;
      related_pin : "A";
    }
    timing() {
      intrinsic_rise : 14.3;
      intrinsic_fall : 18.1;
      related_pin : "B";
    }
  }
}
}

```

Fig. 2

Fig. 3

Fig. 4

(a)

(b)

Fig. 5

(a)

(b)

Fig. 6

Fig. 7

Table. 1

Logic gates in the SFQ cell library for logic synthesis. Their symbol, area and delay.

Gate	Symbol	Area	Delay (ps)
AND		4	19.5
NOT		4	15.4
XOR		4	14.6
CB		1	18.8
SPL2		1	10.2
SPL3		1	8.7