Studies on Datapath Circuits for Superconductor Bit-Slice Microprocessors

Guang-Ming Tang

To

my children, wife and parents

Abstract

There have been considerable advances in superconductor digital circuit technology in recent years. Superconducting rapid single-flux-quantum (RSFQ) circuit technology is expected to become a next-generation circuit technology as an alternative technology to complementary metal-oxide-semiconductor (CMOS) with the physical limitations of scaling. RSFQ circuits have advantages of high-speed computation with low-power consumption. In the past, researchers focused on 8-bit RSFQ microprocessors because of the limitation of integration density of Josephson junctions (JJs). With the development of fabrication technology, it is becoming possible to build a 32-bit RSFQ microprocessor with comparable performance to a CMOS microprocessor operating at 2-3 GHz.

A 32-bit RSFQ bit-serial microprocessor cannot achieve comparable performance to a CMOS microprocessor operating at 2-3 GHz. On the other hand, a 32-bit parallel RSFQ microprocessor cannot be implemented on a couple of chips using today's fabrication technology. Bit-slice processing, in which 32-bit data are divided into several slices of several bits each and slices are processed in a pipelined fashion, may be a solution for a 32-bit RSFQ microprocessor.

The objective of the studies in this dissertation is to develop methods for designing datapath circuits for RSFQ bit-slice microprocessors. The challenges are: (1) Make each feedback loop include no clocked gate so that pipeline processing of slices is continuous; (2) Minimize the delay of each feedback loop, because the clock frequency is limited by the delay of feedback loops; (3) Make timing design easier, because timing design of bit-slice datapath circuits is more difficult than bit-serial datapath circuits; and (4) Make precise timing design in order to achieve high clock frequency.

The following approaches are adopted. (1) Algorithm level: Development of the algorithm for reducing the number of feedback loops without a clocked gate. (2) Logic level: minimization of the delay of feedback loops, reduction of the number of stages

with clocked gates. (3) Layout level: Precise timing design by calculating the delay of wirings.

An arithmetic logic unit (ALU) is the most fundamental component of microprocessors. As the first step towards developing a 32-bit superconductor microprocessor using RSFQ circuits, a 4-bit bit-slice RSFQ ALU has been proposed. The following methods for responding to the challenges have been developed. (1) Algorithm level: Sklansky adder with only one feedback loop without a clocked gate is used. (2) Logic level: The circuit for producing the carry from the most significant bit of a slice is duplicated. The calculations with several signals are executed before entering the feedback loop. (3) Layout level: The all clocked gates in each stage are aligned in a row. The wiring lengths between two adjacent stages are adjusted so that the data arrival time of all gates in each stage is approximately equal.

Using the National Institute of Advanced Industrial Science and Technology (AIST) Advanced Process 2 (ADP2) fabrication process, The 4-bit bit-slice RSFQ ALU has been fabricated and successfully demonstrated at 50 GHz. Testing results showed that the ALU has a throughput of 6.25×10^9 32-bit operations per second.

We have compared 2-/4-/8-/16-bit bit slice as well as bit-serial and parallel ALUs. The results show 4-bit bit-slice RSFQ ALU has the lowest latency and the highest operations per watt for processing 32-bit data at 50 GHz.

A Multiplier is one of the main components of microprocessors. A 32×32-bit 4-bit bit-slice RSFQ multiplier has been proposed. The following methods for responding to the challenges have been developed. (1) Algorithm level: Development of a new systolic-like algorithm with unidirectional data signal flow, carry save addition, and Sklansky adder for final addition. There is only one feedback loop without a clocked gate in the whole multiplier. (2) Logic level: The calculations with the start signal are executed before entering the feedback loop. (3) Layout level: The adjacent blocks are aligned in two stages off. The all clocked gates in each stage are aligned in a row. The wiring lengths are adjusted as in the bit-slice ALU.

We have designed and simulated the multiplier with the target frequency of 50 GHz using the AIST ADP2. It has a throughput of 3.125×10^9 32×32-bit multiplications per second.

We hope that the approaches and the developed methods are useful for designing datapath circuits for RSFQ bit-slice microprocessors.

Acknowledgments

First of all, I would like to thank my advisor Professor Naofumi Takagi of Kyoto University for his patience and guidance. I came to Kyoto University for my PhD degree in October, 2012, in which I knew nothing about the knowledge of rapid single-flux-quantum (RSFQ). Professor Naofumi Takagi took great patience to explain the skills about the design of RSFQ logic circuit again and again when I was a research student in the Computer Engineering during the semesters of Fall 2012 and Spring 2013. In the following three years of my PhD program, we discussed about my design, and revised my papers. His unrelenting conviction and strive for perfection has improve my skills, and affect my life in the future. I appreciate Professor Naofumi Takagi, who gave me freedom of working at unusual hours and days as I had to coordinate between my works and family.

Of course, this dissertation would not be possible without the help of all of members from the Takagi Lab, Graduate School of Informatics, Kyoto University. Especially, thanks to Professor Kazuyoshi Takagi for his continuous guidance, helpful comments during my research. To Professor Hideki Takase, Mr. Takahiro Kawaguchi, Mr. Sho Nishimura, Mr. Yukio Ohmomo, Ms. Yuki Ando, Mr. Masaya Ohata, Mr. Masao Moriya, Mr. Takuya Hatayama, and Professor Nobutaka Kito from Chukyo University for being a great friend during my study.

I would like to thank the people from Fujimaki Lab at Nagoya University, especially to Professor Akira Fujimaki and Professor Masamitsu Tanaka for discussing and helping me to test my chip. I am also grateful to Mr. Kensuke Takata and Mr. Ryo Sato for helping me with testing.

I would like to thank Professor Hidetoshi Onodera and Professor Takashi Sato from the Graduate School of Informatics, Kyoto University. Their comments are all valuable and very helpful for revising and improving my dissertation.

I would like to thank Chinese Government Scholarship, Japanese Government Scholarship, Advance Low CArbon technology research development program (ALCA) of Japan Science and Technology agency (JST), and National Institute of Advanced Industrial Science and Technology (AIST).

And finally, I would like to thank my family for love.

Contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Background

In 1971, the world's first single-chip microprocessor, Intel 4004, was released [1]. It was with clock frequency of 0.74 MHz and power consumption of 0.45 W. It was fabricated by 10 μm process. With the development of complementary metal oxide semiconductor (CMOS) circuit technology, Intel Core m7 has been released in 2015 [2]. It is with clock frequency of 3.1 GHz and power consumption of 4.5 W. It is fabricated by 14 nm process. From 1971 to 2015, the clock frequency and power consumption of Intel CPUs were raised more than four thousand times and 10 times, respectively.

Although power consumption of a large scale CMOS semiconductor system is relatively low, significant amount of power consumption is required for data centers and supercomputers where a large number of cores for computing are required. For example, Chinese Sunway TaihuLight, which is the No.1 of the TOP500 list of the world's top supercomputers in June 2016 [3], requires 15.37 Mega Watts of power consumption on average. The power consumption of data centers and supercomputers is hard to reduce because a minimum switching energy of a semiconductor transistor is required to ensure reliability and speed [4]. Although Intel will release products fabricated by 10 nm process in 2017 [5], CMOS fabrication process has a limitation of 5 nm because the radius of a single atom is about 0.2-0.3 nm. The scaling limit of CMOS circuits will be reached around 2020 forecasted in the International Roadmap for Semiconductors (ITRS) [6]. Therefore, CMOS circuit technology is facing unprecedented challenges.

In order to reduce power consumption and to raise clock frequency, alternative digital circuit technologies with high speed and low power consumption have been

studied. Superconductor digital circuit technology is one of the alternative candidate technologies of CMOS circuits.

In 1966-1983, IBM tried to build a computer using superconductor digital circuit technology. The project was not successful because of choosing the logic similar to CMOS level logic and adopting lead alloy as ultrathin oxide layer with the limitation of 1 GHz clock frequency. As a revolutionary step forward in superconductor digital circuits, rapid single-flux-quantum (RSFQ) circuit technology was developed in 1991 [7]. The clock frequency of RSFQ circuits can achieve several tens of gigahertz with 1/1000 power consumption of CMOS circuits [8]. It is expected to become a next-generation circuit technology that enables high speed computation with low power consumption.

RSFQ large scale integrated (LSI) circuit technology has been developed in order to achieve high clock frequency and to increase circuit functionality. In 2000, RSFQ LSIs including 10^4 Josephson junctions (JJs)/cm^2 were implemented using the fabrication technologies of Japanese NEC standard process (STP) [9] and US TRW's LTS process [10]. Microprocessors using RSFQ LSI circuit technology have been designed, including a 8-bit bit-serial microprocessor (FLUX-1) with eight arithmetic logic units (ALUs) [11], 8-bit bit-serial microprocessors (CORE1 series) [12], [13], [14], [15], and a 8-bit bit-serial asynchronous microprocessor (SCRAM2) [16]. Several CORE1 microprocessors were successfully demonstrated using high speed clock signals [12], [14]. Previous works in RSFQ microprocessors are summarized in Table 1.1.

With the development of RSFQ fabrication process technology featuring 1.0-μm Nb/AlOx/Nb junctions (AIST ADP2) [17], it has become possible to realize RSFQ LSIs including 10^5 JJs/cm^2 in 2014 [18]. The increase in the wiring layers combined with the multi-layer passive transmission line (PTL) technology [19] further increases the circuit integration density. Consequently, it is becoming possible to build a 32-bit RSFQ microprocessor with comparable performance to a CMOS microprocessor operating at 2-3 GHz.

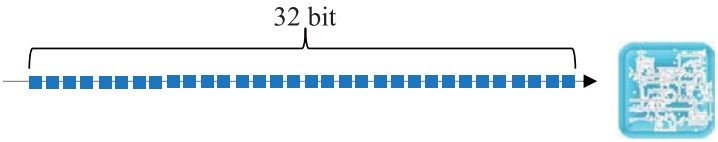
Although bit-serial processing shown in Figure 1.1(a) entails lower complexity, it involves greater latency in processing 32-bit data. This leads to low throughput and degrades microprocessor performance. It is very difficult to implement more than 2 GHz system clock frequency using bit-serial processing. On the other hand, parallel processing shown in Figure 1.1(c) will lead to high throughput performance, but it enta-

Table 1.1. The pervious works in RSFQ microprocessors

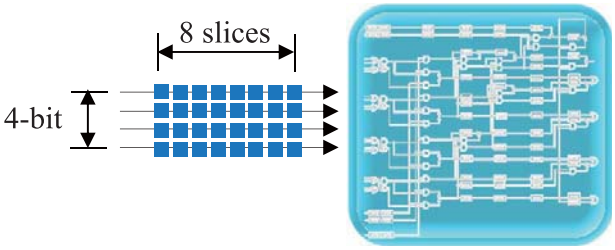
Time	Work	System clock (GHz)	Local clock (GHz)	Data length (bits)	Architecture	Josephson Junctions (JJs)	Power (mW)	Status
2000-2002	FLUX-1 [11]	not clear	20	8	bit-serial	63,107	9.5	designed
2002-2007	CORE1α (ver. 5) [12]	1	15.2	8	bit-serial	4,999	1.6	designed fabricated demonstrated
	CORE1α (ver. 10) [13]	1	18	8	bit-serial	7,222	2.3	designed fabricated
	CORE1β [14]	1	20	8	bit-serial	10,995	3.4	designed fabricated demonstrated
	CORE1γ [15]	1	25	8	bit-serial	22,302	6.56	designed fabricated
2007	SCRAM [16]	not clear	20	8	bit-serial	8,197	2.6	designed fabricated demonstrated

ils high hardware cost. It is difficult to realize a 32-bit bit-parallel RSFQ microprocessor on a couple of RSFQ LSI chips using today's fabrication technology.

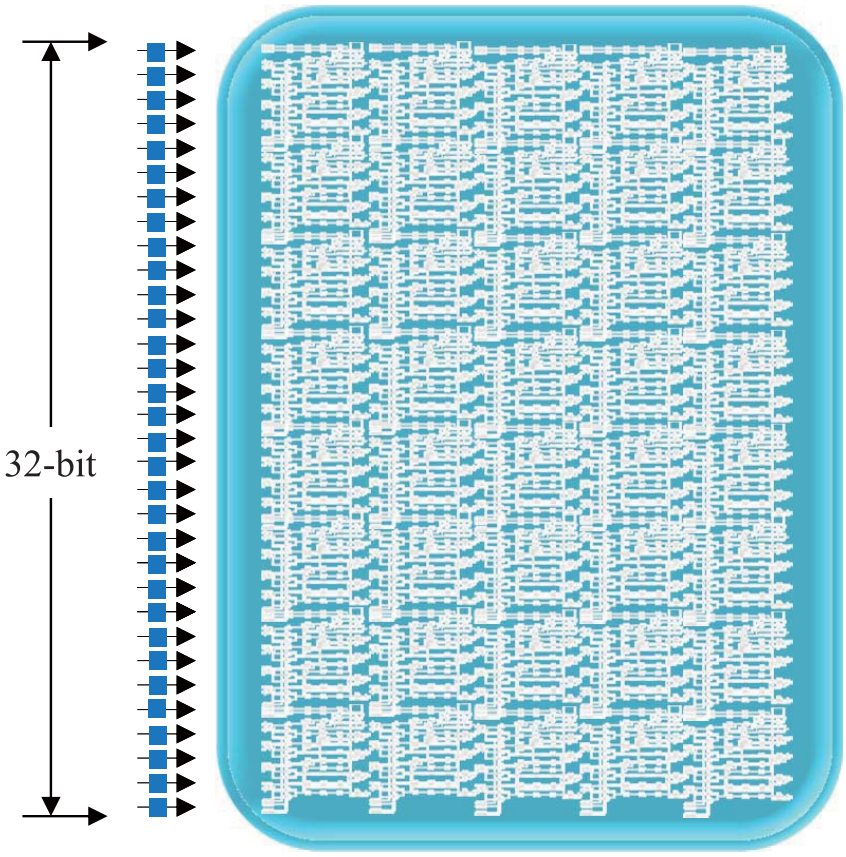
Figure 1.1(b) shows 4-bit bit-slice processing. Because it offers higher speed than bit-serial processing and entails lower hardware cost than parallel processing, bit-slice processing, whereby data are divided into several slices (each of several bits), may be a promising solution for practical 32-bit superconductor microprocessors based on today's fabrication technology. By pipeline processing of slices with 50 GHz clock, 4-bit bit-slice processing achieves a system clock frequency of 6.25 GHz, and can realize comparable performance to a 2-3 GHz CMOS microprocessor.



(a) Bit-serial processing



(b) 4-bit bit-slice processing



(c) Parallel processing

Figure 1.1. The approaches of processing 32-bit data.

An arithmetic logic unit (ALU) is the most fundamental datapath component of microprocessors. Several bit-serial ALUs have been fabricated and successfully

demonstrated in bit-serial microprocessors [12], [14]. A 4-bit parallel ALU has been fabricated and successfully demonstrated at 5 GHz [20]. An 8-bit parallel ALU has been fabricated and successfully passed high-speed test at 20 GHz [21], [22]. Another 8-bit parallel ALU has been designed and fabricated [23].

A Multiplier is one of the main datapath components of microprocessors. An RSFQ 4×4-bit and an 8×8-bit parallel multiplier have been designed and fabricated [24], [25]. An RSFQ 24×24-bit bit-serial multiplier based on the systolic algorithm proposed in [26] has been designed and fabricated as a component of a single-precision floating-point multiplier [27].

Previous works in RSFQ ALUs and multipliers are summarized in Table 1.2. No bit-slice ALUs and multipliers have been demonstrated so far.

Table 1.2. The pervious works in RSFQ ALUs and multipliers

Datapath	Work	Time	Clock (GHz)	Data length (bits)	Architecture	Josephson Junctions (JJs)	Power (mW)	Status
ALUs	[12] [14]	2002-2007	18	8	bit-serial	not clear	not clear	designed fabricated demonstrated
	[20]	2005	5	4	parallel	8,197	2.6	designed fabricated demonstrated
	[21] [22]	2012	20	8	parallel	6,973	2.8	designed fabricated demonstrated
	[23]	2013	30	8	parallel	8,832	2.8	designed fabricated
multipliers	[24]	2006	24	4×4	parallel	2,800	1.2	designed fabricated
	[25]	2013	20	8×8	parallel	5,948	1.7	designed fabricated
	[27]	2015	50	24×24	bit-serial	not clear	not clear	designed fabricated

1.2 Purpose of the studies

The objective of the studies in this dissertation is to develop methods for designing datapath circuits for superconductor bit-slice microprocessors.

As main datapath components of an RSFQ bit-slice microprocessor, a 4-bit bit-slice ALU and a 4-bit bit-slice multiplier operating at 50 GHz clock are studied. There are the following challenges:

(1) Make each feedback loop include no clocked gate so that pipeline processing of slices is continuous. Unlike parallel datapath circuits, feedback loops are required in bit-slice datapath circuits for carry signal.

(2) Minimize the delay of each feedback loop. The clock frequency is limited by the delay of each feedback loop (without a clocked gate).

(3) Make timing design easier. Bit-slice datapath circuits need more complex signal flow and interconnection than bit-serial datapath circuits. Timing design of bit-slice datapath circuits is more difficult than bit-serial datapath circuits.

(4) Make precise timing design in order to achieve high clock frequency.

1.3 Overview of this dissertation

In Chapter 2, first, the principle, logic gates, representation of logic value, and clocking method of RSFQ circuits are introduced. Then, the fabrication technology for the chips of this study is described. The environment of design, simulation, and testing for this study are also described. The challenges in developing RSFQ bit-slice datapath circuits are summarized.

In Chapter 3, a 4-bit bit-slice ALU is designed, fabricated and demonstrated. First, the underlying algorithm is designed. An algorithm based on Sklansky adder is developed, so that the ALU has only one feedback loop without a clocked gate. Then, the RSFQ logic design of the ALU is shown. The feedback loop is minimized. The number of pipeline stages is reduced. Then, the physical layout is shown. Precise timing design is made. The ALU is fabricated and successfully demonstrated at 50 GHz. In addition, a bit-serial, several bit-slice and a 32-bit parallel ALUs are compared.

In Chapter 4, a 32×32-bit 4-bit bit-slice multiplier is designed and simulated. First, a systolic-like algorithm is developed. Carry save addition is adopted and Sklansky adder is used for the final addition. Data signal flow is unidirectional and there is only one feedback loop in the multiplier. Then, the RSFQ logic design of the multiplier is

shown. The feedback loop is minimized. Then, the physical layout is shown. Precise timing design is made. The multiplier is simulated and correct operation at 50 GHz is verified. In order to show the correctness of the algorithm and logic design, an 8×8-bit 4-bit bit-slice multiplier is fabricated.

Finally, Chapter 5 summarizes the results of the studies, and concludes the dissertation.

Chapter 2

Preliminaries

RSFQ circuit is one family of superconductor digital circuits. The principle, logic gates, representation of logic value, and clocking method of RSFQ circuits are introduced in Section 2.1. The fabrication technology used in this study is described in Section 2.2. The environment of design, simulation, and testing for this study are described in Section 2.3.

2.1 RSFQ circuits

2.1.1 Principle

RSFQ circuit consists of superconductor inductance loop with JJs as shown in Figure 2.1. The basic active components for RSFQ circuits are two-terminal JJs. Unlike CMOS circuits, the basic passive components for RSFQ circuits are inductances, not capacitances. The magnetic flux in the superconductor inductive loop is quantized as $\Phi = n\Phi_0$ (n: integer) where $\Phi_0 = 2.07\times10^{-15}$ Wb. Information is stored in the form of magnetic flux quantum and transferred in the form of single-flux-quantum (SFQ) voltage pulse.

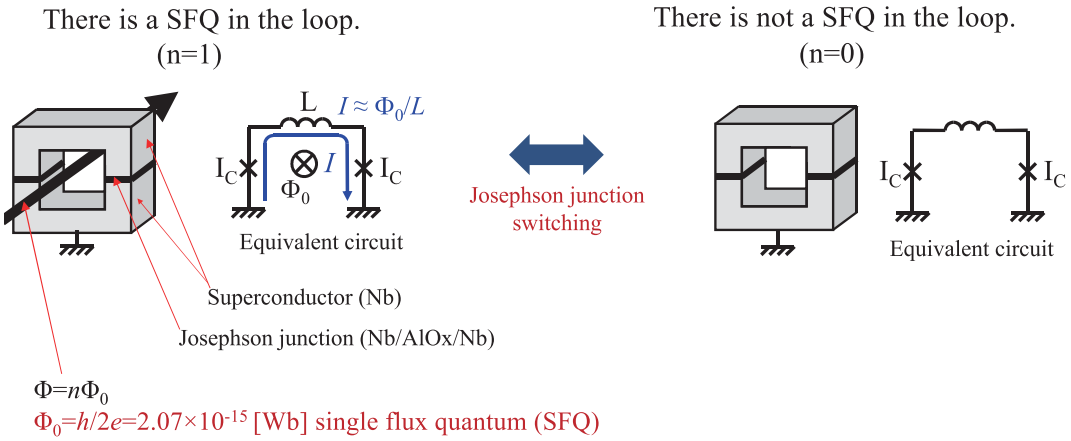


Figure 2.1. A superconductor inductive loop [28].

Figure 2.2 shows equivalent circuit of Josephson transmission line (JTL). DC bias current I_B is applied to each JJ. When an SFQ pulse is input from port *in*, the sum of induced currents $I \approx \Phi_0/L$ and I_B exceeds the critical current of J1. Then, J1 switches from superconductor state to voltage state and an SFQ pulse is generated. The SFQ voltage pulse generates current about Φ_0/L, which goes into J2 with I_B. Then, J2 switches and generates an SFQ pulse, and so on. Finally, an SFQ pulse is output from *out* port.

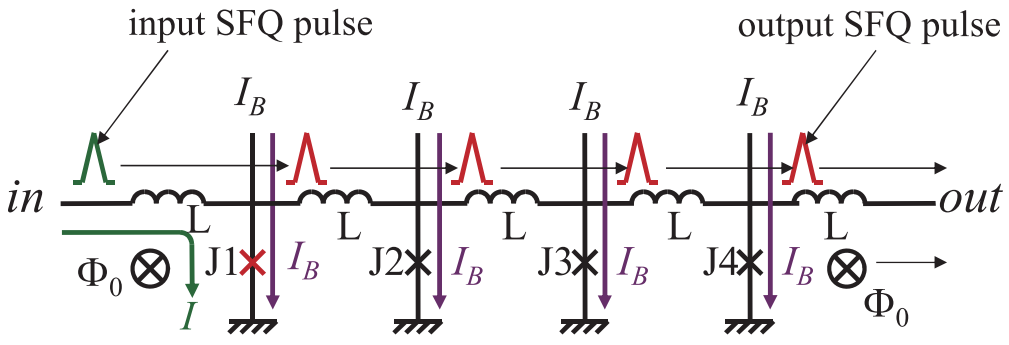
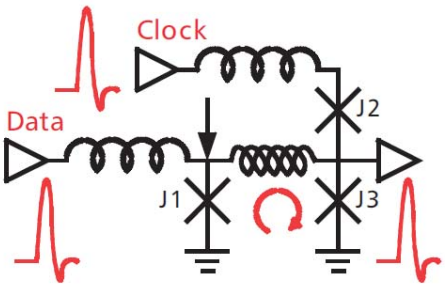


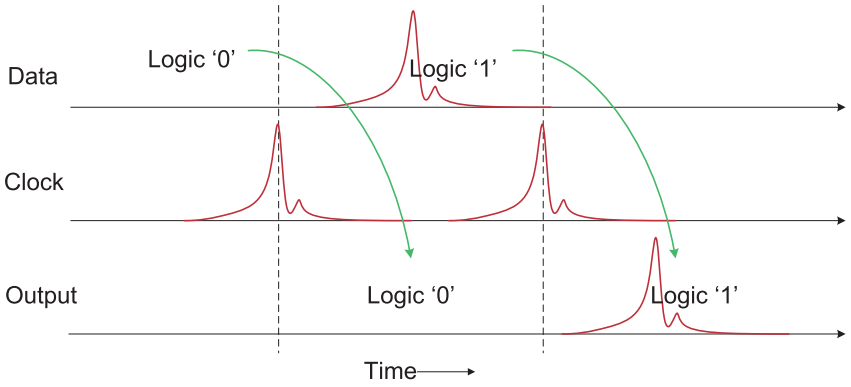
Figure 2.2. JTL [28].

2.1.2 Logic gates and representation of logic value

RSFQ gates are built with multiple JJs, inductors and bias resistors at the circuit level. An equivalent circuit of a D flip-flop (DFF) is shown in Figure 2.3 (a). It includes an input junction J1, a decision-making comparator composed of junctions J2 and J3, and an inductor between J1 and the J2/J3 pair to store an SFQ. When an SFQ voltage pulse arrives at the input port of data, J1 switches and the inductor loop stores an SFQ. This stored SFQ adds amount of current to J3. When a clock pulse arrives at this state, J3 switches, an SFQ pulse is output, and then the DFF is set back to the zero-state. If there was no SFQ stored in the DFF when a clock pulse arrives, then the current in J3 is not enough to switch. Thus no SFQ pulse is output. The logic value and timing of DFF are shown in Figure 2.3(b). "1" is represented by the presence of an SFQ pulse between two adjacent clock pulses, and "0" is represented by the absence of an SFQ pulse between two adjacent clock pulses. Bias resistors are used as current distributing elements to supply current to the JJs.



(a) Equivalent circuit [29]



(b) Logic value and timing

Figure 2.3. DFF gate.

Figure 2.4 shows an equivalent circuit of an RSFQ AND gate. It is based on two DFFs. In the figure, J is a JJ, L is an inductor, and I is bias current. A clock input is necessary in order to meet the two input pulses. A logic gate with a clock input is called a clocked gate.

An SFQ is stored in the loop J_1-L_1-J_2 when an SFQ pulse is input from IN1. Another SFQ is stored in the loop J_5-L_2-J_6 when an SFQ pulse is input from IN2. Then, a clock pulse arrives, the stored SFQs are moved into loops, J_2-L_3-J_9-J_{11} and J_6-L_4-J_{10}-J_{11}, respectively. Large amount of current flows through J_{11} because two SFQs affect it. Thus, J_{11} switches before J_9 or J_{10} switches, and an output pulse is generated. If there is only one input pulse, only one SFQ is stored in one of the loops J_1-L_1-J_2 or J_5-L_2-J_6. Then, a clock pulse arrives, the stored SFQ is moved into one of loops, J_2-L_3-J_9–J_{11} or J_6-L_4-J_{10}–J_{11}. The amount of current flows through J_{11} is not enough for switching. Therefore, J_9 or J_{10} switches before J_{11} switches. Transmission of the SFQ to the output port is blocked. Thus, there is no output pulse.

A synchronous clocking method is commonly used for representing the logic values. The logic value on a data line in a period between adjacent clock pulses is 1 or 0 according to the presence or absence of an SFQ pulse on the line.

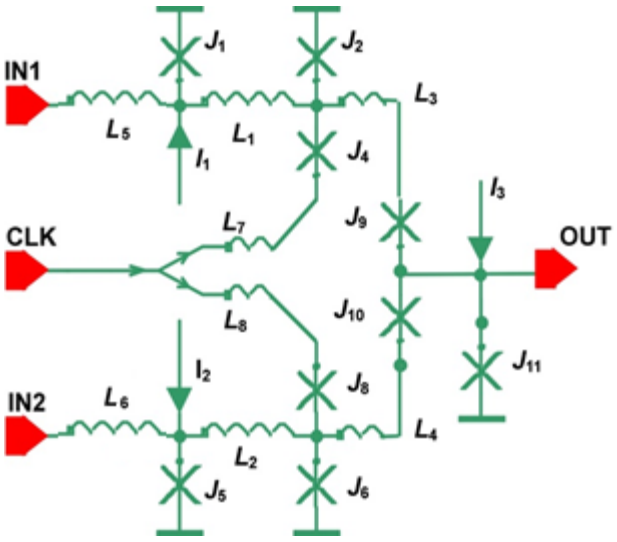


Figure 2.4. Equivalent circuit of an RSFQ AND gate [30].

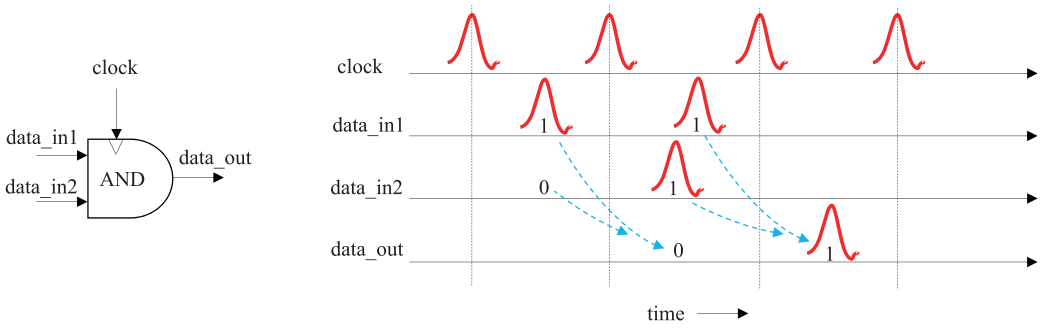


Figure 2.5. Logic value and timing of an AND gate.

Figure 2.5 shows the logic value and timing of an AND gate. When data_in1=1 and data_in2=0, data_out=0 after a clock arriving at the AND gate. If data_in1=1 and data_in2=1, data_out=1 after a clock arriving at the AND gate.

2.1.3 Clocking method

The delays of wiring elements cannot be ignored because RSFQ circuits work at several tens of gigahertz. In order to implement high speed RSFQ circuits, the clocking used in RSFQ circuits is different from the clocking used in CMOS circuits. Concurrent-flow clocking is used in RSFQ circuits [31].

Figure 2.6 shows two clocked gates (gate1 and gate2) in an RSFQ circuit. A clock (clk) must be provided for the gates. Let the time when a clock pulse passes the base point of the figure as zero. Then, the time when the pulse arrives at gate2 $Time_{clk}$ is sum of the delays of line2, splitter and line3. The time when the data pulse produced by the clock pulse at gate1 arrives at gate2 $Time_{data}$ is sum of the delays of line1, gate1 and line4. In concurrent-flow clocking, the clock pulse that produced a data pulse at gate1 arrives gate2 earlier than this data pulse (i.e., $Time_{clk} < Time_{data}$). This data pulse is processed at gate2 by the next clock pulse. N clock cycles are needed to carry the data through an RSFQ circuit comprised of N gate stages. We design a circuit so that slices of a datum are processed successively in a pipelined fashion.

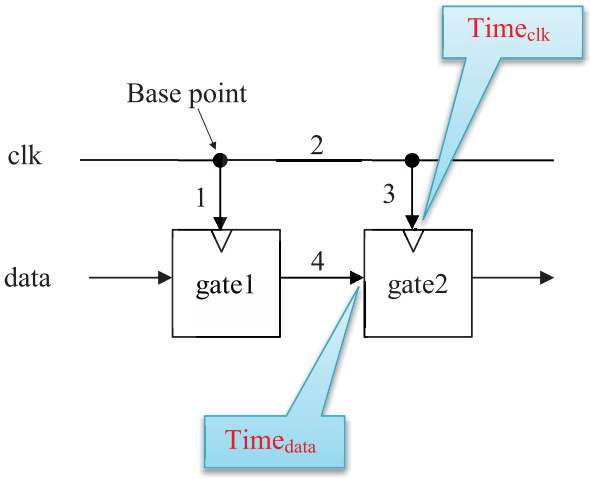


Figure 2.6. The arrival timing of RSFQ pulse.

Figure 2.7 shows constraints of timing. Here, t_c is the arrival time of a clock pulse to gate2, the clock cycle is t_{cycle}, the hold time of gate2 is t_{hold}, the setup time of gate2 is t_{setup}. $t_c + t_{hold} < t_{data} < t_c + t_{cycle} - t_{setup}$ must be satisfied.

Figures 2.8-2.10 show how a 3-stage RSFQ circuit using concurrent-flow clocking operates. The numbers in the figure are the arrival time of pulses to the clocked gates (unit: picosecond). We assume that t_{cycle} = 20 ps (50 GHz clock) and t_{hold} = 2 ps, t_{setup} = 3 ps for the AND gates.

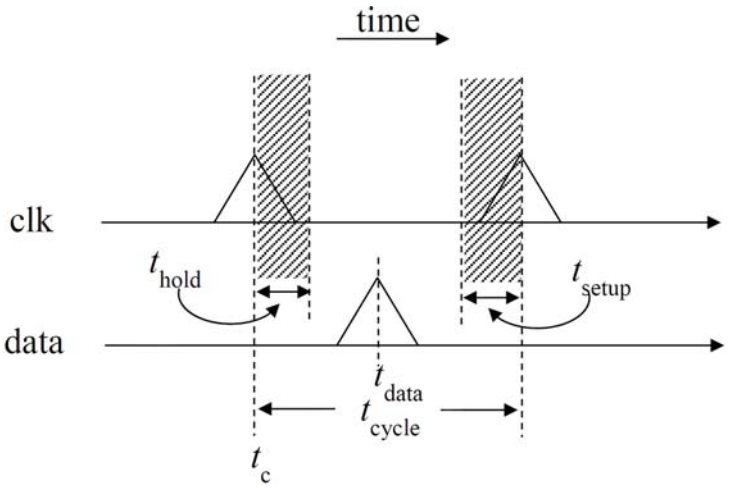


Figure 2.7. Constraints of timing for an RSFQ gate.

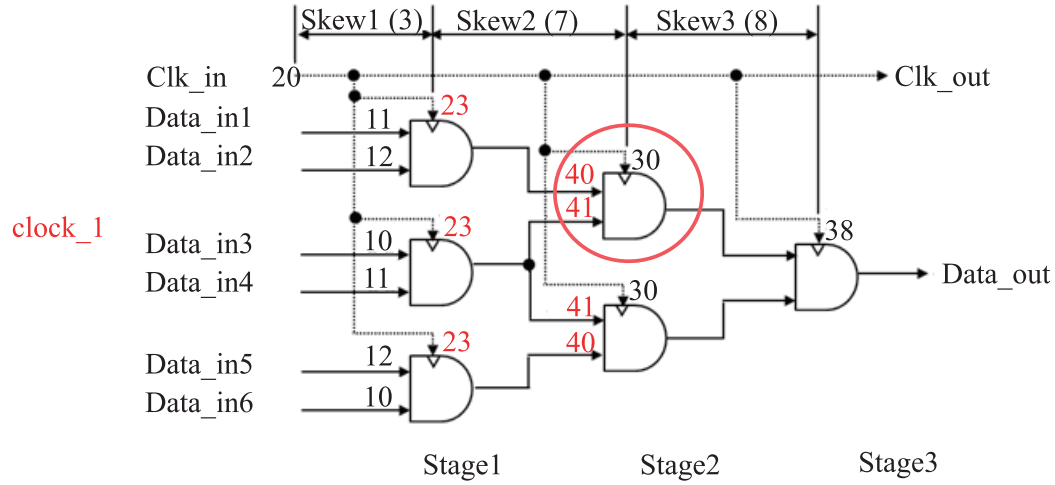


Figure 2.8. Arriving time of pulses at clock_1.

Figure 2.8 shows that the clock pulse clock_1 arrives the gates in Stage2 before the data pulses produced by it at the gates in Stage1. These data pulses are processed by clock_2 as shown in Figure 2.9. At the AND gate in the red circle, the constraints of timing (i.e., $t_c + t_{hold} = 30 + 2 = 32 < t_{data} = 40$, $41 < t_c + t_{cycle} - t_{setup} = 30 + 20 - 3 = 47$) is satisfied.

Figure 2.9 shows that clock_2 arrives the gate in Stage3 before the data pulses produced by it at the gates in Stage2. These data pulses are processed by clock_3 as

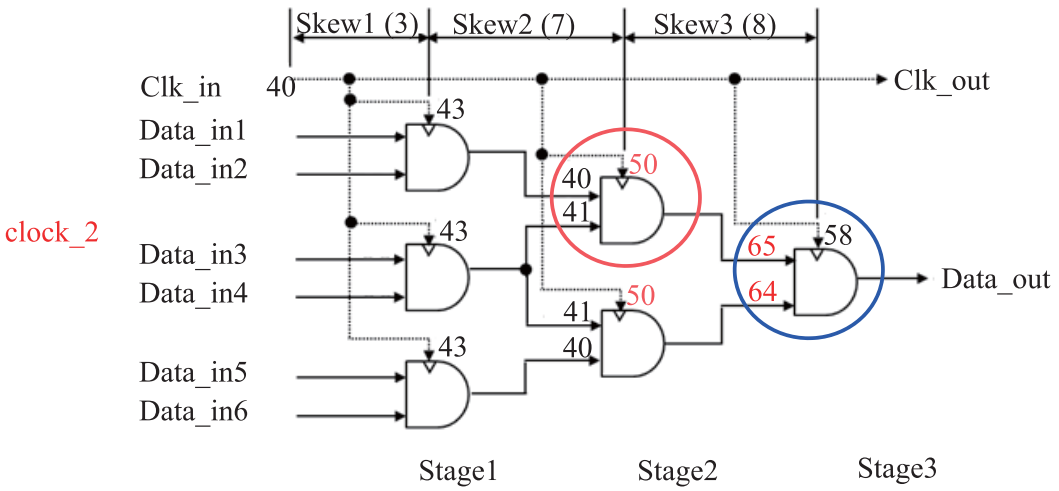


Figure 2.9. Arriving time of pulses at clock_2.

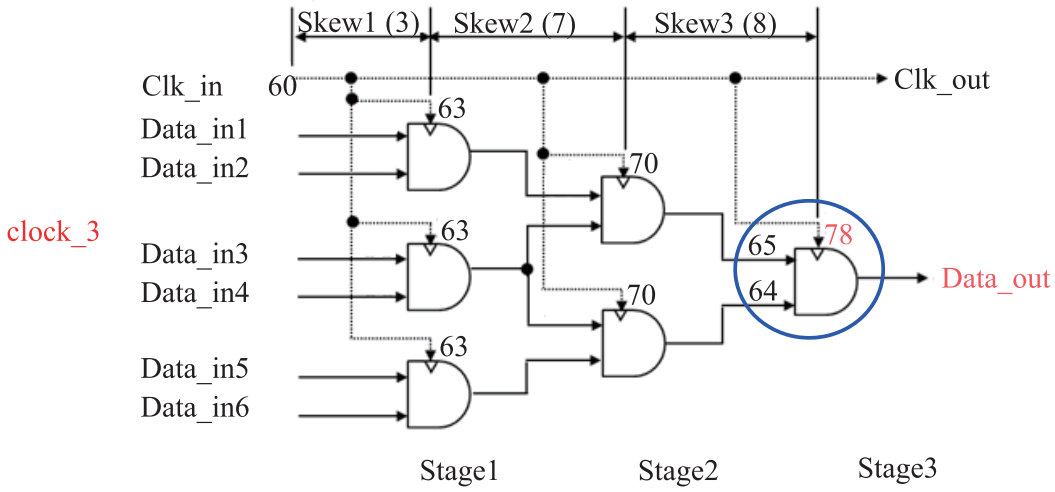


Figure 2.10. Arriving time of pulses at clock_3.

shown in Figure 2.10. At the AND gate in the blue circle, the constraints of timing (i.e., $t_c + t_{hold} = 58 + 2 = 60 < t_{data} = 65$, $64 < t_c + t_{cycle} - t_{setup} = 58 + 20 - 3 = 75$) is satisfied. The total of clock skews (i.e., Skew1 + Skew2 + Skew3 = 3 + 7 + 8 = 18) from clock input to clock output is called '*clock delay*' for the RSFQ circuit.

Pipeline processing is possible in an RSFQ circuit with concurrent-flow clocking. One stage of gates forms one pipeline stage.

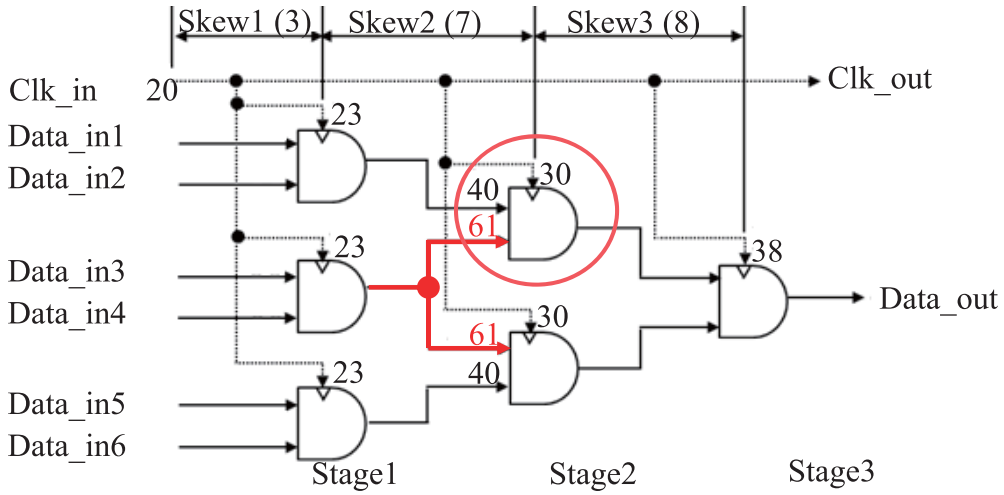


Figure 2.11. Timing for an RSFQ gate with timing error.

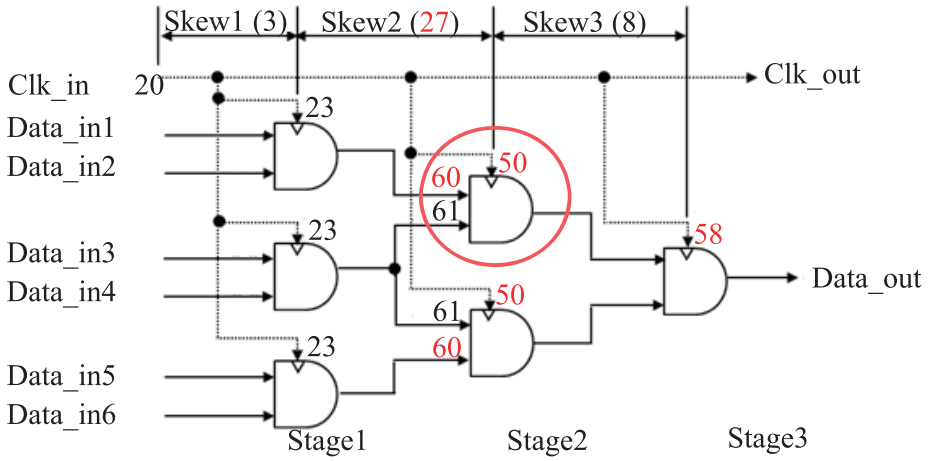


Figure 2.12. Adjustment wiring lengths to solve timing error.

In RSFQ circuits, precise timing design is necessary for achieving high clock frequency. Figure 2.11 shows an example of an RSFQ circuit with a timing error, where the constraints of timing is not satisfied at the AND gate in the red circle (i.e., $t_c + t_{hold} = 30 + 2 = 32 < t_{data} = 61 \not< t_c + t_{cycle} - t_{setup} = 30 + 20 - 3 = 47$). t_c must be increased by increasing Skew2 in order to maintain 50 GHz clock frequency and satisfy $t_c + t_{hold} < t_{data} = 61 < t_c + t_{cycle} - t_{setup}$. When t_c is increased, the other data arrival time 40 must be increased to be approximately equal to 61.

Figure 2.12 shows timing adjustment for satisfying the constraints of timing (i.e., $t_c + t_{hold} = 50 + 2 = 52 < t_{data} = 60$, $61 < t_c + t_{cycle} - t_{setup} = 50 + 20 - 3 = 67$). Skew2 is increased from 7 to 27. Therefore, the clock delay is increased to 38 (= 3 + 27 + 8).

Figure 2.13 shows an RSFQ circuit with a feedback loop including clocked gates. There is a feedback from Stage3 to Stage1. When slices of data are processed in this circuit in pipeline fashion, slices must be input to the circuit every three cycles, namely not continuously. For continuous pipeline processing of slices, a bit-slice datapath circuit must not include feedback loops with clocked gates.

Figure 2.14 shows a feedback loop without clocked gates. The clock frequency of an RSFQ circuit is limited by the delay of the feedback loop in it. In order to achieve high clock frequency, the delay must be minimized because the constrains of timing (i.e., $t_c + t_{hold} < t_{feedback} + t_c < t_c + t_{cycle} - t_{setup}$) must be satisfied.

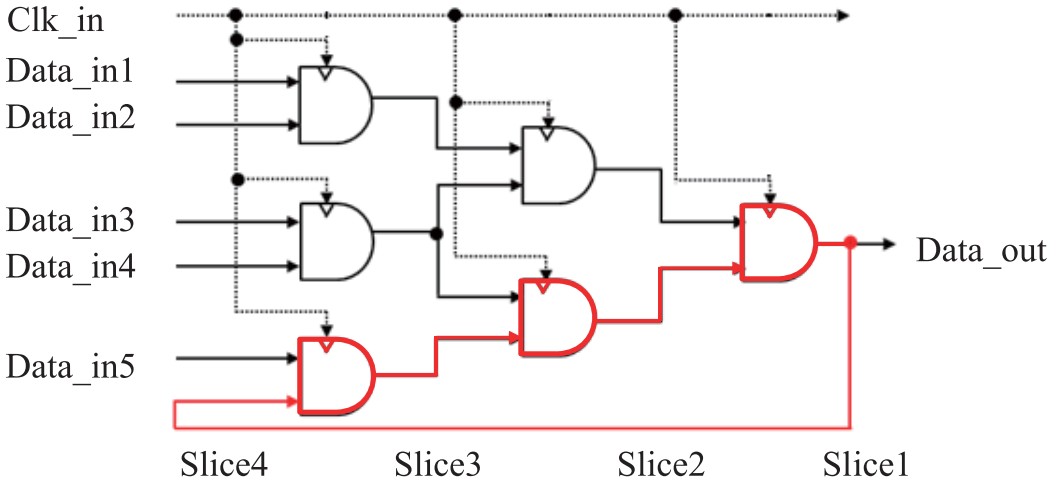


Figure 2.13. A feedback loop with clocked gates.

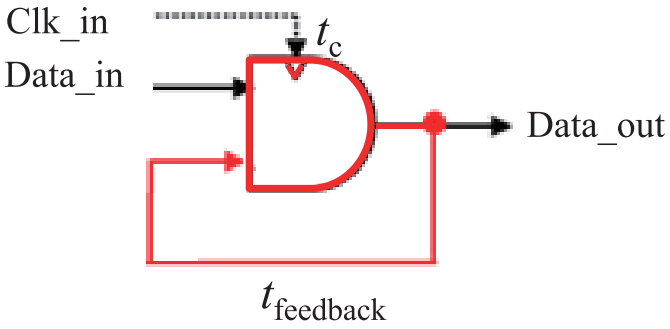


Figure 2.14. A feedback loop without clocked gates.

2.2 Fabrication technology of RSFQ circuits

The RSFQ circuits designed in this dissertation are fabricated using the AIST ADP2 process in the Clean Room for Analog-digital superconductiVITY (CRAVITY) of the AIST.

Two layers of passive transmission line (PTL) are attached in the AIST ADP2 fabrication technology. The increase in the wiring layers combined with the multi-layer PTL technology further increases the circuit integration density.

PTLs are the basic passive transmission lines. The delay of PTLs is shorter than JTLs. PTLs are used to connect adjacent RSFQ gates with long distance. The lengths of PTLs must be carefully adjusted when designing layout of high speed RSFQ circuits.

Figure 2.15 [32] shows the structure of the device fabricated in the AIST ADP2 fabrication technology. It consists of an active layer including JJ (Nb/AlOx/Nb junction) (size: 1 μm) and resistor (RES) layer, main ground plane (GP) and complemented planarization layer (CPL), passive transmission line (PTL) layers, and DC power layer. M1-M9 are Nb layers; M2, M4, M6, and M7 are ground planes GND1, GND2, GND3, and GP, respectively. C1-C6, GC, RC, BC, and JC are used to contact between metal layers. Interlayer insulator is SiO_2. Resistor RES is made up with molybdenum (Mo), whose thickness is 45nm. A micrograph of the device fabricated in AIST ADP2 process is shown in Figure 2.16.

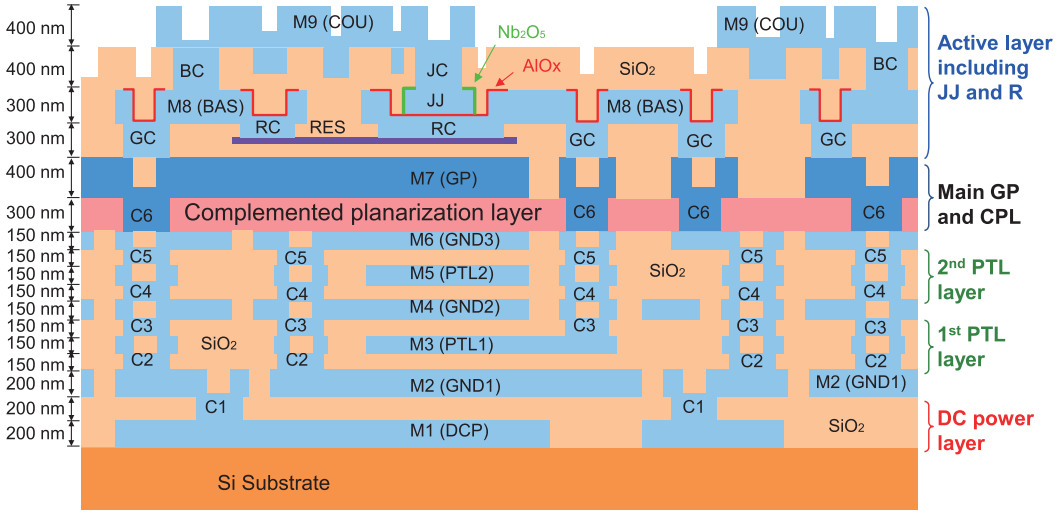


Figure 2.15. The structure of the device fabricated in AIST ADP2 process [32].

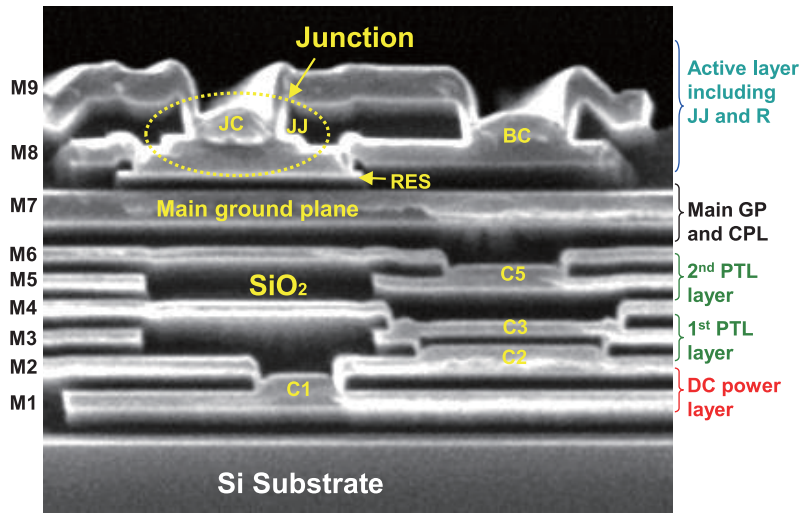


Figure 2.16. The micrograph of the device fabricated in AIST ADP2 process [32].

The integration density is 10^5 JJs/cm^2. There are 10^5 JJs in one chip with the area of 10×10 mm^2.

2.3 Environments of design, simulation, and test

2.3.1 Environments of design and simulation

For design and simulation, RSFQ computer-aid-design (CAD) software is necessary. In this dissertation, all RSFQ circuits are designed and simulated using the RSFQ CAD tools [33], [34] in Takagi Lab. Environments of design and simulation for RSFQ digital circuits are shown in Figure 2.17.

Cell-based design methodology is used for designing RSFQ digital circuits based on basic RSFQ clocked logic gates (AND, OR, exclusive OR (XOR), NOT), flip-flop (D flip-flip (DFF) and non-destructive read-out (NDRO)), wiring elements (JTL, PTL, and splitter (SPL)), and confluence buffer (CB) in the RSFQ cell library. In this dissertation, the RSFQ cell library for the AIST ADP2 developed by Nagoya University is used. The RSFQ cells used in this dissertation are described in Table 2.1.

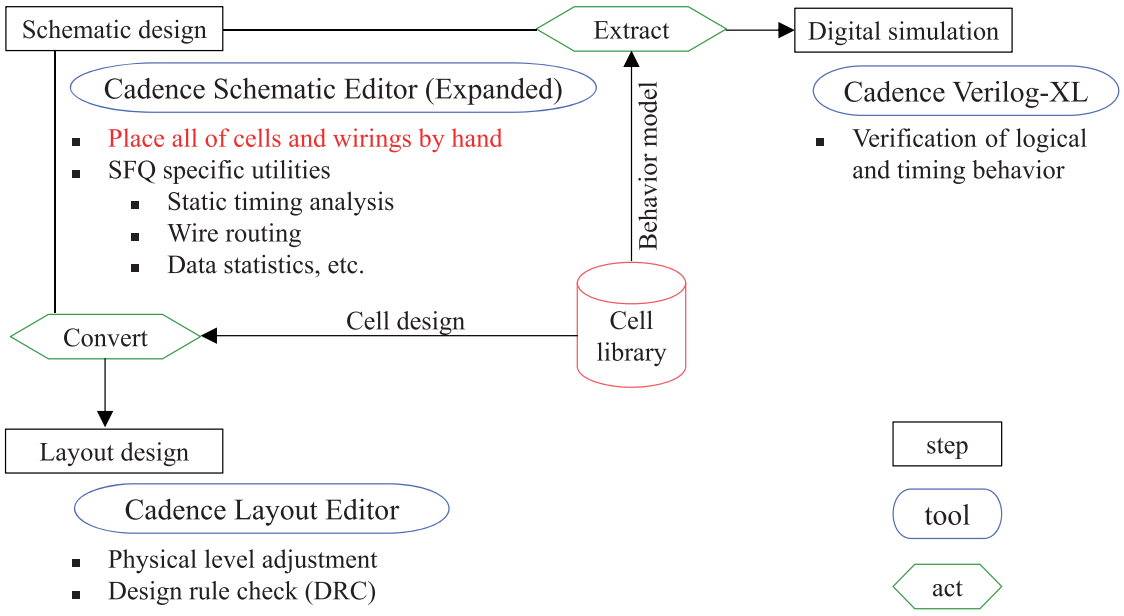
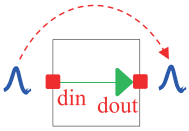
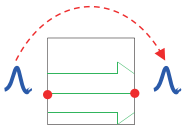
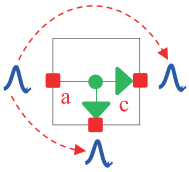
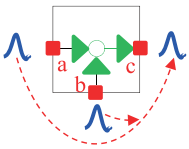
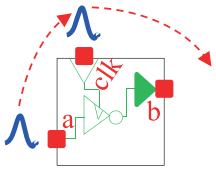
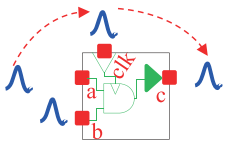
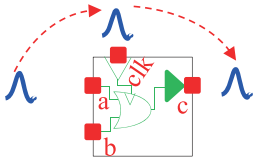


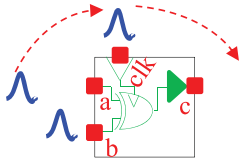
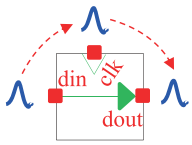
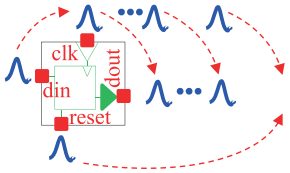
Figure 2.17. Environments of design and simulation of RSFQ digital circuits.

In schematic design step, Cadence schematic editor (expanded) is used. All of cells and wirings are placed and connected by hand. The tool provides static time analysis, wire routing, and data statistics. After routing and static timing analysis, we perform digital simulation of the physical layout with Cadence Verilog-XL software using the behavior model (cells' delay, setup time, and hold time with DC bias are described using Verilog HDL language) of cells provided with the cell library. In layout design step, we use Cadence layout editor to adjust the wiring lengths by hand in order to achieve high clock frequency.

After design rule check (DRC) of the physical layout, the source file is sent to the Fujimaki Lab of Nagoya University for assigning bias current and connecting pins. Finally, the layout file (i.e., a GSDII type file) is generated and sent to the AIST for fabrication. In general, the same four chips (with different position in a wafer) are fabricated after three months.

Table 2.1. Listing of RSFQ cells used in this dissertation

Cell	RSFQ pulse operation and symbol	JJs	Bias current	Function description
JTL		2	0.3 mA	Josephson transmission line (JTL) is wiring element used for shorter distance between two cells. JTL can be also used to add delay.
PTL		0	0	Passive transmission line (PTL) is wiring element used for longer distance between two cells. PTL can be also used to add delay.
SPL		3	0.3 mA	Splitter (SPL) is wiring element used to increase a fanout (1-to-2, or 1-to-3) of a element.
CB		7	0.9 mA	Confluence buffer (CB) is used to combine two RSFQ pulse path into a single path. Its function is similar to OR logic gate but it does not require clock.
NOT		11	0.8 mA	NOT, AND, OR, and XOR gates are two-terminal RSFQ clocked logic gates. If, and only if, RSFQ pulse(s) arrive at inputs before the arrival of clock, a SFQ output pulse is generated after the clock pulse arrives. Their logic function is same as NOT, AND, OR, and XOR gates of CMOS, respectively. There are not three-terminal gates in RSFQ clocked logic gates.
AND		15	1.4 mA	
OR		12	1.2 mA	

XOR		11	1.1 mA	
DFF		6	0.8 mA	Like DFF gate of CMOS, D flip-flop (DFF) receives and stores a RSFQ pulse, which is read out by the clock.
NDRO		11	1.1 mA	Non-destructive read-out (NDRO): If an RSFQ pulse arrived at the input port before the clock, an RSFQ pulse is generated at the output port after the clock pulse arrives. This state is not changed until the reset pulse arrives.

2.3.2 Environment of test

In order to verify the functionality of superconducting RSFQ circuits designed in this study, the chips fabricated by AIST are tested after bonding aluminum wires between the RSFQ chip and chip holder at the Fujimaki Lab of Nagoya University. Table 2.2 lists the detail of the test equipment used in this dissertation.

Table 2.2 List of testing equipment used in this dissertation

Equipment Name (Label)	Manufacturer (Type)	Function description
Electromagnetic shield room (A)	Custom made (metal Fe)	Shields electromagnetic wave
Geomagnetic shield box (K)	Custom made (metal Mu)	Shields geomagnetic field
Chip holder (B)	Custom made	Wire-bonds the RSFQ digital chip onto a chip holder that is placed into the chip probe
Liquid Helium Dewar (M)	Custom made	Cools down the RSFQ digital chip tested to T=4.2 K
RSFQ digital chip probe (L)	Custom made	Holds the RSFQ digital chip into the liquid Helium Dewar, and links the connection box
Connection box (E)	Custom made	Connects attenuators, DC power supplies, output differential amplifiers, and RSFQ digital chip probe
Attenuator (F)	TAMAGAWA (UBA-761A)	Lowers the voltage of the input active filter to millivolt level for RSFQ digital chip
Active filter (D)	NF Corporation (P-82)	Minimizes noise of the input data generated by data generator before putting them into attenuators
Data generator (I)	SONY Tektronix (DG2020A)	Generates input data for the RSFQ digital chip after filtering and attenuating
Variable data output pod (H)	SONY Tektronix (P3420)	Provides variable digital signals output levels to the RSFQ digital chip under test
DC power supply (J)	KIKUSUI (PMR24-1QU)	Supplies DC bias current for the RSFQ digital chip
Differential amplifier (C)	NF Corporation (P-61)	Amplifies the output wave of the RSFQ digital chip for easily viewing on oscilloscope
Oscilloscope (G)	SONY Tektronix (TLS216)	Displays the output wave of the RSFQ digital chip

Figure 2.18 shows the equipment providing a test condition where the RSFQ chip works in a temperature of ～4.2 K (～−270℃) (provided by liquid Helium in the Dewar (M)) and an environment without electromagnetic and geomagnetic fields. Figure 2.19 shows a block diagram of the testing environment. The labels in Table 2.2, Figure 2.18, and Figure 2.19 represent the same equipment.

Because the RSFQ digital chip is very sensitive to electromagnetic and geomagnetic fields, the testing equipment must be placed in a room with electromagnetic shield (A) and the RSFQ digital chip must be placed in an environment without geomagnetic filed. Fujimaki Laboratory provides a box made of double-layer Mu-metal to shield geomagnetic filed (K). DC-to-SFQ bias, SFQ-to-DC bias and main RSFQ circuit bias for the testing chip are supplied by DC power supply (J) via the

Figure 2.18. Equipment used for testing RSFQ digital chips.

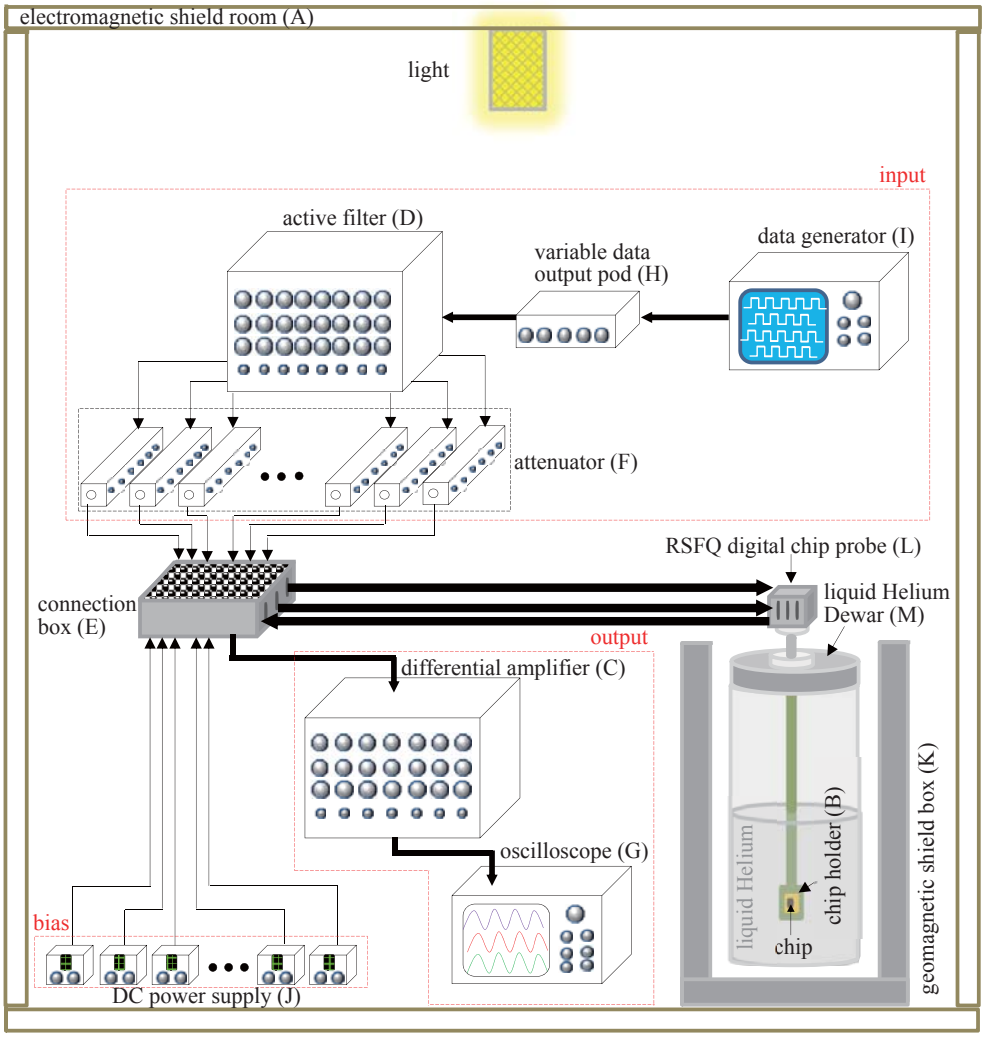


Figure 2.19. Block diagram of the testing environment.

connection box (E). The input data are generated by data generator (I), changed by variable data output pod (H), filtered by active filter (D), and attenuated by attenuator (F). Then they are linked to the connection box (E), and input into the testing chip held on the chip holder (B) placed in the RSFQ digital chip probe (L). The output data generated by the testing chip are displayed on the oscilloscope (G) after amplifying by the differential amplifier (C).

In this study, a method of the on-chip testing [35] is used. Figure 2.20 shows a block diagram of a chip with circuits for testing.

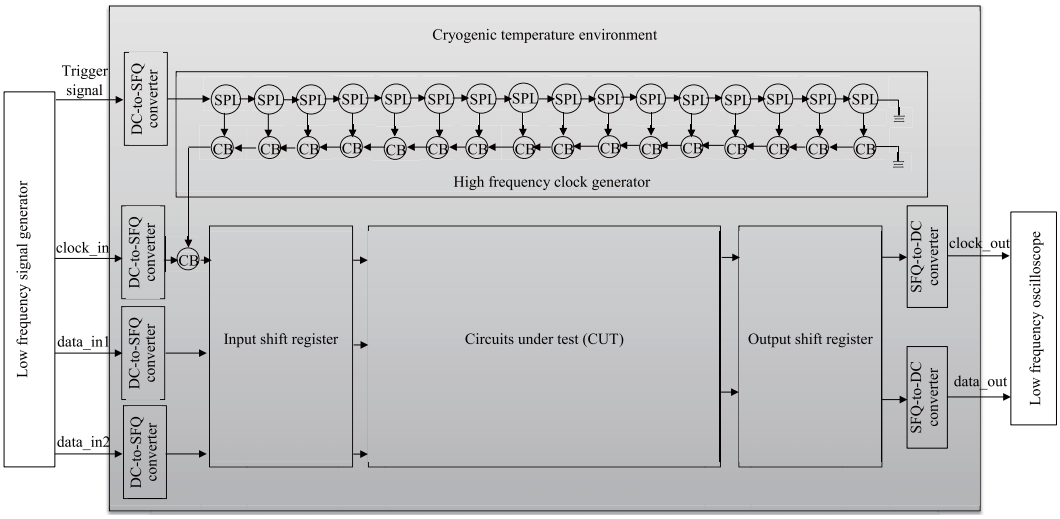


Figure 2.20. Block diagram of the on-chip testing system.

A special circuit component called “DC-to-SFQ converter” to input SFQ signals from room-temperature electronics. The DC-to-SFQ converter generates an SFQ pulse when its input voltage rises. Because it is very difficult to capture rapid and weak SFQ signals directly using room-temperature electronics, a special circuit component called “SFQ-to-DC converter” is also used. The operation of SFQ-to-DC converter is based on toggle flip-flop. The output voltage is inverted when an SFQ pulse arrives at the SFQ-to-DC converter.

Because it is impossible to test circuits at several tens of gigahertz clock signals using normal measurement setup, we use a high frequency clock generator (CG) based on RSFQ circuits, and the high frequency operations of the circuit under test (CUT) is performed using the high frequency clock signals on the chip. A block diagram of a 16-bit ladder high frequency clock generator is shown in Figure 2.20. Each stage consists of an SPL, a CB and JTLs represented by the arrows. The first high frequency clock SFQ pulse is generated after the low frequency trigger pulse traveling through the first SPL, the first rung of the ladder and the first CB. The second high frequency clock SFQ pulse comes out through the first two SPLs, the second rung of the ladder and the first two CBs. The total number of high frequency clock SFQ pulses generated from a single low frequency trigger pulse is controlled by the number of stages in the high frequency clock generator. The high frequency clock SFQ pulse interval is roughly the delay of one stage which can be adjusted by the number of JTLs, and also depends on the DC

bias. In the last stage, the unconnected SPL output and CB input are each terminated to ground.

In order to satisfy on-chip high frequency testing, input shift registers is required to load input data with low frequency clock and output shift registers are also required to stored output data with an on-chip high frequency clock. During the testing, (1) test vectors are written into the input shift registers using low frequency signal generator in room-temperature environment; (2) high frequency operations are operated in CUT using the on-chip high frequency clock generated by CG in cryogenic temperature environment; (3) results are read from the output shift registers and displayed on oscilloscope using low frequency signals in room-temperature environment.

2.4 Summary

In this chapter, the basic knowledge of RSFQ circuits, the fabrication technology used to fabricate our chips, and environments of designing, simulating and testing our RSFQ circuits in this dissertation are described.

Unlike CMOS circuits, pulse logic is adopted in RSFQ circuits. Synchronous clocking is used to represent the logic value of RSFQ circuits. The wiring elements are JTLs, SPLs, and PTLs. The delay of wiring elements cannot be ignored because RSFQ circuits work at several tens of gigahertz. Concurrent-flow clocking, which is different from CMOS circuits are used in RSFQ circuits. The AIST ADP2 fabrication technology and cell-based design methodology are used. The physical layout is designed and simulated with RSFQ CAD tools. The chip is fabricated at AIST and tested at Nagoya University.

There are four challenges for designing datapath circuits for RSFQ bit-slice microprocessors as follows:

(1) Make each feedback loop include no clocked gate so that pipeline processing of slices is continuous.

Unlike parallel datapath circuits, feedback loops are required in bit-slice datapath circuits for carry signal. During algorithm design step, we must ensure each feedback loop without a clocked gate.

(2) Minimize the delay of each feedback loop.

The clock frequency is limited by the delay of each feedback loop (without a clocked gate).

(3) Make timing design easier.

Bit-slice datapath circuits need more complex signal flow and interconnection than bit-serial datapath circuits. Timing design of bit-slice datapath circuits is more difficult than bit-serial datapath circuits. During the logic design step, we must simplify circuit structure.

(4) Make precise timing design in order to achieve high clock frequency.

The timing of each gate in RSFQ circuits must be satisfied the constrains of timing $t_c + t_{hold} < t_{data} < t_c + t_{cycle} - t_{setup}$.

Chapter 3

Bit-slice arithmetic logic unit

3.1 Introduction

An ALU is the most fundamental component of microprocessors. It is responsible for executing the arithmetic and logic operations. Several bit-serial ALUs in bit-serial microprocessors [12], [14] and several 4-bit or 8-bit parallel ALUs [20], [21], [22], [23] were designed, fabricated, and demonstrated. However, no bit-slice ALU for 32-bit practical RSFQ microprocessors has been demonstrated so far.

In this chapter, our study is focused on developing a 50 GHz 4-bit bit-slice ALU. For this purpose, we adopted the following design approaches: (1) Algorithm level: Development of the algorithm for reducing the number of feedback loops without a clocked gate, (2) Logic level: Minimization of the delay of feedback loops, reduction of the number of stages with clocked gates, and (3) Layout level: Precise timing design by calculating the delay of wirings.

Each of the two 32-bit operands is divided into eight slices of 4 bits each. The eight pairs of operand slices are input one by one starting from the least significant one. All the arithmetic and logic ALU operations, namely, addition (ADD), subtraction (SUB), set on less than (SLT), AND, OR, exclusive OR (XOR), and NOR, as well as comparison of equality of operands (EQ) for conditional branches, are carried out. An operation to be executed is selected by a combination of control signals. The result of each operation is expressed as eight 4-bit slices that are output one by one (except for SLT and EQ, which give 1-bit results).

The ALU has eight input ports for a pair of operand slices, X_0, X_1, X_2, X_3, Y_0, Y_1, Y_2, and Y_3, and seven ports for control signals, including *Carry_in*. Further, it has four output ports for a resultant slice, Z_0, Z_1, Z_2, and Z_3, and a port called *Carry_out*.

The designed ALU entails low hardware cost and offers high-speed implementation of 32-bit data calculations. It is based on a Sklansky adder [36]. The ALU was fabricated using the AIST ADP2. All the arithmetic and logic operations were successfully demonstrated at 50 GHz.

The algorithm, logic design, physical layout design and simulation, and fabrication and test are described in Sections 3.2-3.5. In Section 3.6, we compare 2-/4-/8-/16-bit bit-slice as well as bit-serial and parallel RSFQ ALUs for 32-bit data.

3.2 Algorithm

An adder is the basic component of an ALU. Feedback loops are required in a bit-slice adder because the carry from the most significant position of a slice must be fed into the least significant position of the next slice. For continuous pipeline processing of slices, a bit-slice ALU must not include feedback loops with clocked gates.

Implementations of hardware algorithms of adders using synchronous clocking RSFQ circuits are compared in [37]. Ripple carry adder has low throughput and large hardware cost with a large feedback loop if it is used to build bit-slice RSFQ ALUs. Thus, parallel prefix adders seem to be most suitable for implementation bit-slice ALUs. An RSFQ 4-bit bit-slice adder based on the Kogge-Stone adder has been developed [38]; it features a complex carry look-ahead circuit with four feedback signals divided into a 4-stage pipeline. Therefore, we adopt the Sklansky adder which is also one of parallel prefix adders. A bit-slice adder based on the Sklansky adder has a simple carry look-ahead circuit with only one feedback signal. This circuit structure is advantageous for constructing an ALU. Although its largest fan-out is larger than that of the Kogge-Stone adder in general, it is not so large in a 4-bit slice.

Hereafter, $\wedge$, $\vee$, $\oplus$, and $\bar{}$ denote logical AND, OR, Exclusive-OR, and NOT, respectively.

Skalansky adder is adopted in this study. The N-bit Sklansky adder consists of four parts as follows:

1. Initialization part: The carry generate signal $g_i = x_i \wedge y_i$ and carry propagate signal $p_i = x_i \oplus y_i$ are calculated at each position i (for 0 to N-1).
2. Carry look-ahead part (PG network): The carry generate signal $G_{i:0}$ and carry propagate signal $P_{i:0}$ from the LSB to each position i (for 0 to N-1) are calculated. Figure 3.1(a) shows the PG network of an 8-bit Sklansky adder.

The calculations represented by black boxes, and gray boxes are shown in Figure 3.1(b), where i and j are from 0 to $N-1$, $j < k \leq i$, respectively. The height of the prefix tree is $\log_2 N$. The maximum fanout is $N/2$.

3. Carry calculation part: The $c_i = G_{i:0} \vee (P_{i:0} \wedge c_{in})$ is calculated at each position i (for 0 to N-1).
4. Sum calculation part: The final sum $z_i = p_i \oplus c_i$ is calculated at each position i (for 0 to N-1).

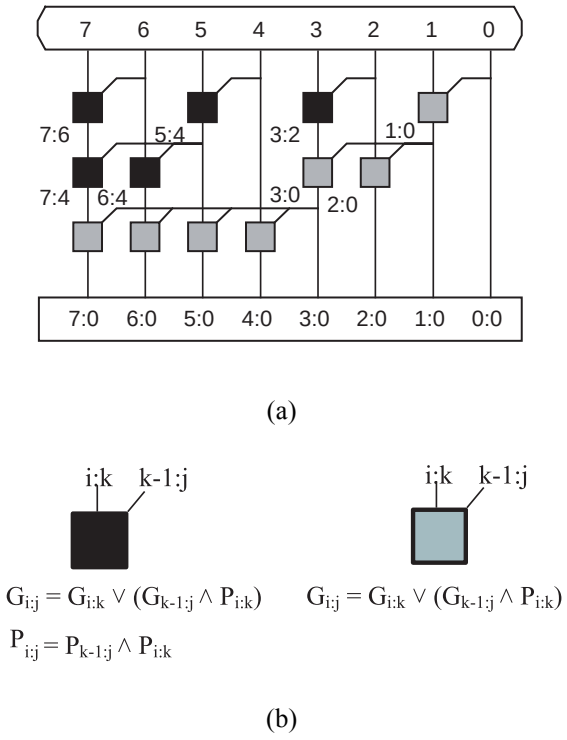


Figure 3.1. The PG network of an 8-bit Sklansky adder [36].

Figure 3.2 shows the PG network of a 4-bit bit-slice Sklansky adder. The carry from the MSB of the slice is fed back to the succeeding slice. There is only one feedback loop.

Figure 3.3 shows the PG network of the 4-bit bit-slice Sklansky adder using RSFQ circuits. In order to ensure the synchronization of each slice, D flip-flops (DFFs) are required because the clocked logic gates have latch function in RSFQ circuits. It forms a 5-stage pipeline.

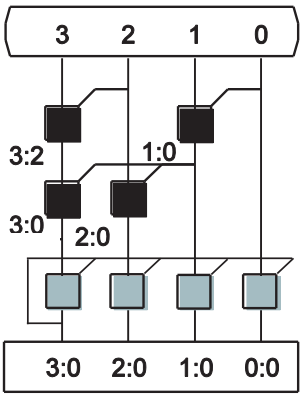


Figure 3.2. The PG network of a 4-bit bit-slice Sklansky adder.

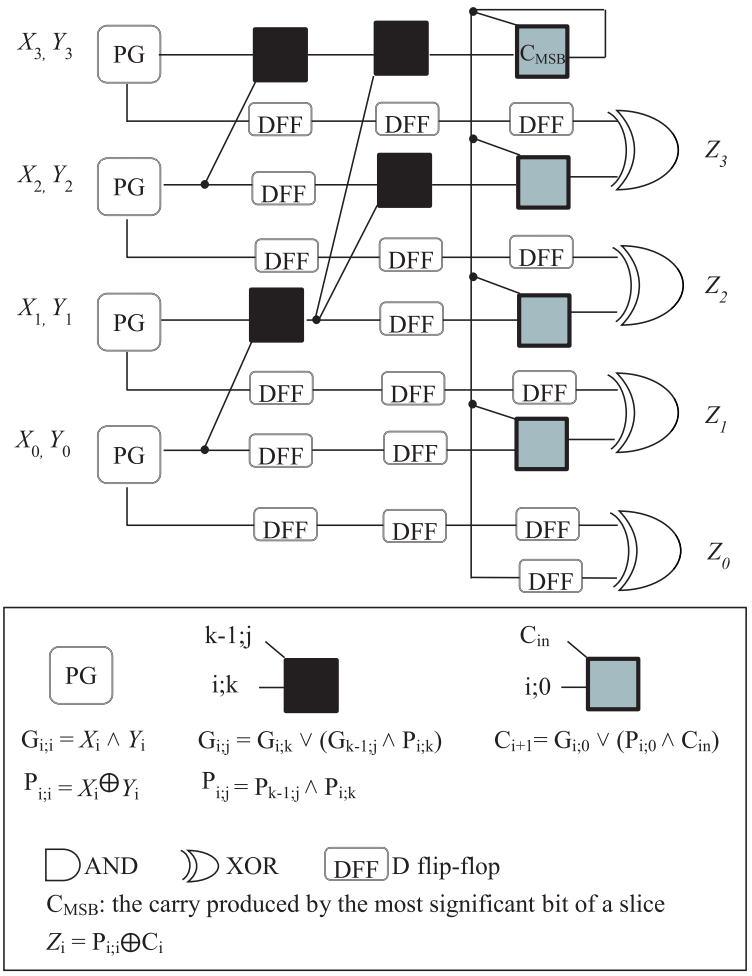


Figure 3.3. The PG network of a 4-bit bit-slice Sklansky adder using RSFQ circuits.

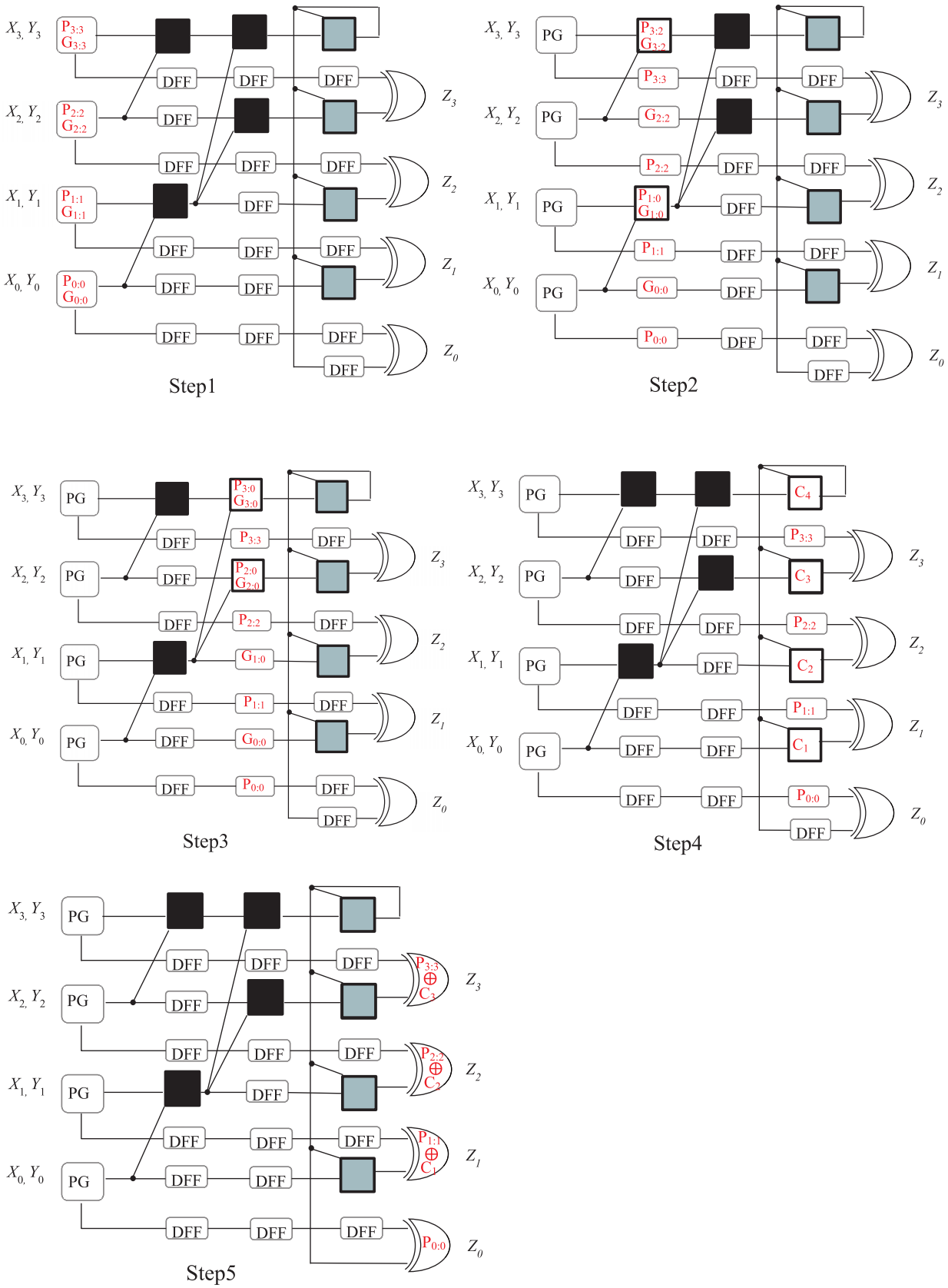


Figure 3.4. Step-by-step processing of a slice pair in the 4-bit bit-slice Sklansky adder.

Figure 3.4 shows the processing of a slice pair in the 4-bit bit-slice Sklansky adder.

Step 1

The slice pair is input to the adder from the input ports (i.e., X_0-X_3 and Y_0-Y_3) and the PG signals (i.e., $P_{0:0}$, $P_{1:1}$, $P_{2:2}$, $P_{3:3}$, $G_{0:0}$, $G_{1:1}$, $G_{2:2}$, $G_{3:3}$) of this slice pair are calculated in the first stage.

Step 2

$P_{1:0}$, $G_{1:0}$, $P_{3:2}$ and $G_{3:2}$ signals of the slice pair are calculated and $P_{0:0}$, $P_{1:1}$, $P_{2:2}$, $P_{3:3}$, $G_{0:0}$, and $G_{2:2}$ signals of the slice pair are moved into the DFFs of the second stage.

Step 3

$P_{2:0}$, $G_{2:0}$, $P_{3:0}$ and $G_{3:0}$ signals of the slice pair are calculated and $P_{0:0}$, $P_{1:1}$, $P_{2:2}$, $P_{3:3}$, $P_{1:0}$, $G_{1:0}$ and $G_{0:0}$ signals of the slice pair are moved into the DFFs of the third stage.

Step 4

The carries (i.e., C_1, C_2, C_3, and C_4) are calculated using the carry from the proceeding slice, and $P_{0:0}$, $P_{1:1}$, $P_{2:2}$, and $P_{3:3}$ signals of the slice pair are moved into the DFFs of the 4-th stage. The carry C_4 will be fed back to this stage for the next slice.

Step 5

The sum bits (i.e., Z_0, Z_1, Z_2, and Z_3, here, $Z_i = P_{i:i} \oplus c_i$) are calculated in the 5-th stage.

Subtraction can be implemented by attaching a path of inverting operand. The logical operations can be implemented by reusing PG signals without extra paths.

3.3 RSFQ Logic design

Figure 3.5 shows an RSFQ logic design of the proposed 4-bit bit-slice ALU for processing 32-bit data in an 8-stage pipeline. It is based on the Sklansky adder described in Section 3.2. Concurrent-flow clocking is employed.

In order to simplify the feedback loop, the circuit for producing the carry from the most significant bit of a slice is duplicated. Computations with *Carry_in* and *End_bar* are performed in advance, before entering the feedback loop.

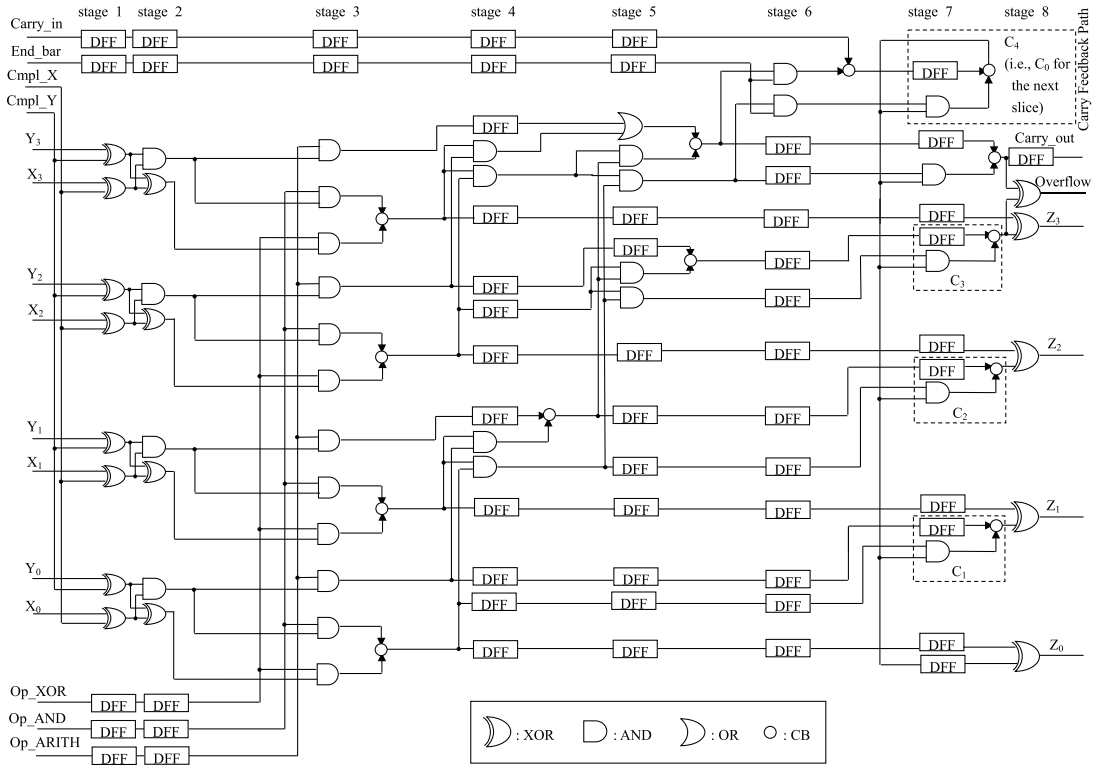


Figure 3.5. RSFQ logic design of 4-bit bit-slice ALU.

For a 4-bit bit-slice Sklansky adder, first, the carry generate signal $g_i = X_i \wedge Y_i$ and carry propagate signal $p_i = X_i \oplus Y_i$ are calculated at each position i (0–3). Next, for the block of conjunctive positions 0 and 1, the carry propagate signal $G_{1;0} = g_1 \vee (p_1 \wedge g_0)$ and carry propagate signal $P_{1;0} = p_1 \wedge p_0$ are calculated. Concurrently, $G_{3;2}$ and $P_{3;2}$ are calculated in a similar manner. Next, $G_{2;0} = g_2 \vee (p_2 \wedge G_{1;0})$ and $P_{2;0} = p_2 \wedge P_{1;0}$, as well as $G_{3;0} = G_{3;2} \vee (P_{3;2} \wedge G_{1;0})$ and $P_{3;0} = P_{3;2} \wedge P_{1;0}$, are calculated. Then, the carry $c_{i+1} = G_{i;0} \vee (P_{i;0} \wedge c_0)$ is calculated at each position i (0–3), where $G_{0;0} = g_0$ and $P_{0;0} = p_0$. Here, c_0 from previous slice is used. Finally, $Z_i = p_i \oplus c_i$ is calculated at each position i (0–3). Only c_4 is fed back as c_0 for the calculation of the next slice.

Subtraction is implemented as $X + Y + 1$, where Y is the bit-wise complement of Y. Therefore, we provide control signal *Cmpl_Y* for inverting Y_i's and *Carry_in* for "+1".

The result of SLT is equal to 1 if and only if $X - Y < 0$. Therefore, subtraction is carried out. The result is the MSB of the result of subtraction, i.e., Z_3 of the last slice, for signed SLT, and the inverse of the carry out, i.e., c_4 of the last slice, for unsigned SLT. We provide output port *Carry_out* for c_4. This output can also be used to check for *Overflow*.

We reuse g_i and p_i for AND and XOR operations, respectively, and obtain OR as $(X_i \wedge Y_i) \vee (X_i \oplus Y_i)$ as in [21], [22]. Since $X_i \wedge Y_i$ and $X_i \oplus Y_i$ are mutually exclusive, we can use a confluence buffer (CB) for the $\vee$ operation. We realize NOR as $X_i \wedge Y_i$. Therefore, we provide control signal *Cmpl_X* for inverting X_i's. For propagating the result of a logic operation to the output port, we use the path for bringing p_i to the Z_i calculation stage in the adder at each position in order to reduce the number of D flip-flops and wirings. To activate AND and XOR to the path, we provide control signals *Op_AND* and *Op_XOR*, respectively.

The result of EQ is equal to 1 if and only if $X = Y$. We can obtain the result as the carry out, i.e., c_4 of the last slice, by setting $g_i = 0$ and $p_i = X_i \oplus Y_i$ for all i's (0–3) of all slices and *Carry_in* (c_0 for the first slice) = 1, in the adder. Note that when $X = Y$, all p_i's are equal to 1.

In order to distinguish arithmetic operations from logic operations, we provide control signal *Op_ARITH*. Then, we encode the operations using control signals, as shown in Table 3.1.

Table 3.1 ALU operations

ALU Operation	***Op_ARITH***	***Op_AND***	***Op_XOR***	***Cmpl_X***	***Cmpl_Y***	***Carry_in***
ADD	1	0	1	0	0	0
SUB	1	0	1	0	1	1
SLT	1	0	1	0	1	1
EQ	0	0	1	0	1	1
AND	0	1	0	0	0	0
OR	0	1	1	0	0	0
XOR	0	0	1	0	0	0
NOR	0	1	0	1	1	0

Pairs of operand slices are input one by one from the first clock cycle to the eighth clock cycle. Another control signal, *End_bar*, is fed in order to distinguish the last slice pair, which is equal to 0 for the eighth cycle and 1 for the first to seventh cycle. Five control signals, *Op_ARITH*, *Op_AND*, *Op_XOR*, *Cmpl_X*, and *Cmpl_Y*, maintain the same values during the eight cycles. *Carry_in* signal is fed in the first cycle. The next ALU operation can be started just after the input of the most significant slice of the previous operation. In other words, the ALU performs 32-bit operations every eight clock cycles, and the maximum throughput in this bit-slice processing is achieved.

As shown in Figure 3.5, the ALU consists of basic RSFQ clocked logic gates (AND, XOR, OR, and D flip-flop (DFF)) and wiring elements (Josephson transmission line (JTL), splitter, and confluence buffer (CB)) from the RSFQ cell library for AIST ADP2.

We reduce the number of pipeline stages in the ALU by using CBs for implementing the ∨ operation in the calculations of $G_{1;0}$, $G_{3;2}$, $G_{2;0}$, $G_{3;0}$, and c_{i+1}. As the result, the number of pipeline stages is eight. Therefore, the first, i.e., the least significant, slice of the result is output at the 9th cycle, and the last slice of the result is output at the 16th cycle.

3.4 Physical layout design and simulation

A physical layout of the 4-bit bit-slice ALU has been designed with the cell library for the AIST ADP2. We set the target clock frequency to 50 GHz. Figure 3.6

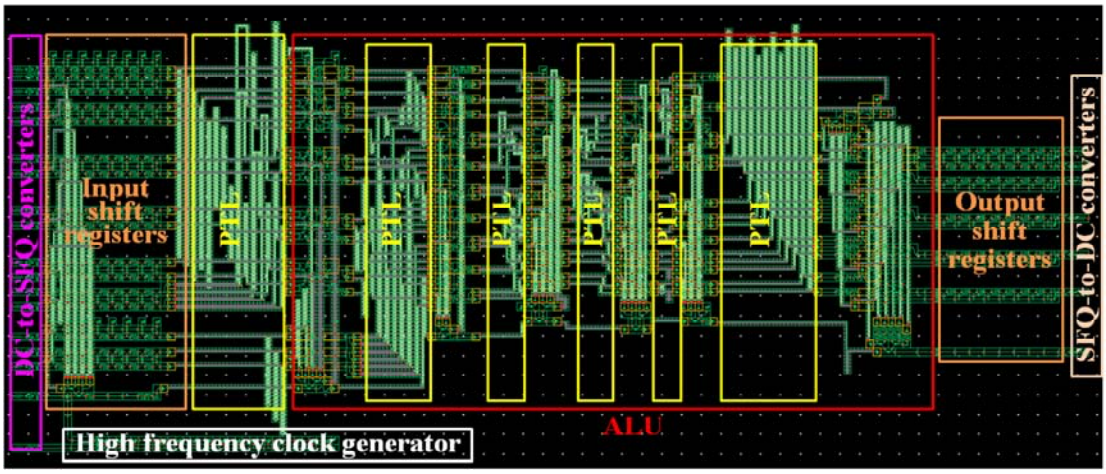


Figure 3.6. The physical layout of the 4-bit bit-slice ALU.

shows the physical layout of the ALU with the DC-to-SFQ converters, input shift registers, output shift registers, SFQ-to-DC converters, and high frequency clock generator.

For precise timing design, all the clocked gates in each stage are aligned in a row. For clock distribution, because most data signals are connected to either the same or the next upper bit position of a slice as they travel in pipeline stages, we placed the root of the clock distribution tree for each stage at the LSB side (i.e., the underside of each stage). This clock distribution strategy eased the difficulty in timing design. For the data paths, we adjusted the total time of flight in all the PTLs between the pipeline stages to be nearly equal by adjusting the wiring lengths between two adjacent stages. Using the RSFQ CAD tools developed by our group, the lengths of PTLs can be calculated automatically. The data arrival time of all clocked gates in each stage is approximately equal.

In order to verify the logic design and the physical layout design, we simulated the designed circuit.

In order to check whether each path of the physical layout works, several simulation vectors are used.

Because OR operation is implemented by $(X_i \wedge Y_i) \vee (X_i \oplus Y_i)$, an OR operation is simulated in order to check the paths for generating the carry generate signal g_i and carry propagate signal p_i, operating the $\vee$ operation with CBs in the third stage, bringing

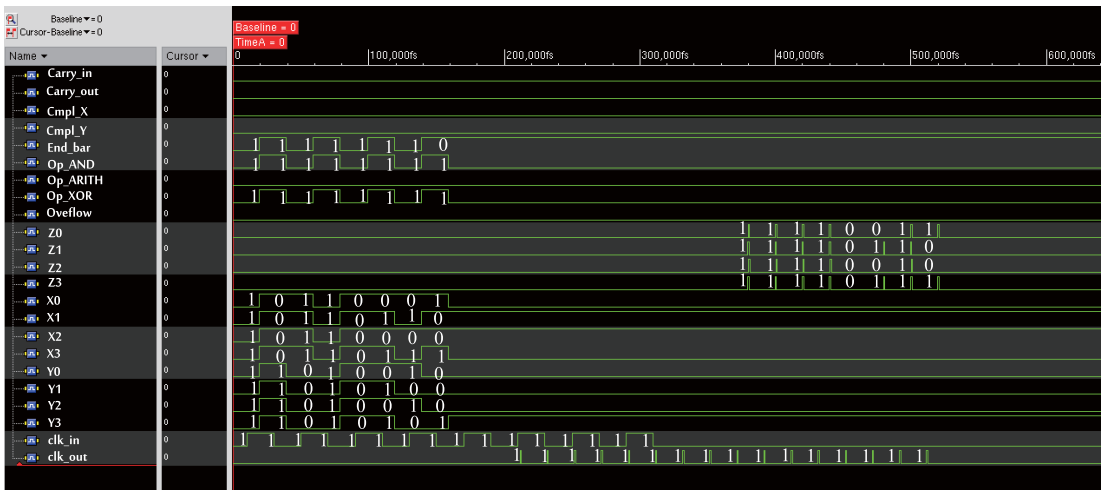


Figure 3.7. Simulation waveforms showing OR operation at 50 GHz.

p_i to Z_i. Figure 3.7 shows the simulation result of $Z_i = X_i \vee Y_i$ operation, 1001 1010 1010 0000 1111 1111 0000 1111 ∨ 1000 0101 1010 0000 1111 0000 1111 1111 = 1001 1111 1010 0000 1111 1111 1111 1111. The control signals (i.e., *Op_AND* and *Op_XOR*) for selecting g_i and p_i are equal to 1 from the first to the eighth clock cycle. The control signal (i.e., *End_bar*) for distinguishing the last slice pair is equal to 0 for the eighth cycle and 1 for the first to seventh cycle. The simulation result shows that the paths and the control signals described in this paragraph work correctly.

In order to check the path for inverting X_i and Y_i (i.e., the first stage), an NOR operation is simulated because NOR operation is represented by $X_i \wedge Y_i$. Figure 3.8 shows a correct simulation result of $Z_i = X_i \wedge Y_i$ operation, 0000 0000 0000 0000 0000 0000 0000 0000 ↓ 0000 0000 0000 0000 0000 0000 0000 0000 = 1111 1111 1111 1111 1111 1111 1111 1111. Here, ↓ denotes logical NOR. The control signals for inverting X_i and Y_i (i.e., *Cmpl_X* and *Cmpl_Y*), and selecting g_i (i.e., *Op_AND*) are equal to 1 from the first to the eighth clock cycle. *End_bar* is the same as OR operation. The simulation result shows that the path for inverting X_i and Y_i, and the control signals of *Cmpl_X* and *Cmpl_Y* work correctly.

SUB operation is simulated in order to check the carry network of the ALU. Figure 3.9 shows the simulation result of $Z_i = X_i - Y_i$ operation, 1000 1000 1000 1000 1000 1000 1000 1000 − 0111 0111 0111 0111 0111 0111 0111 0111 = 0001 0001 0001 0001 0001 0001 0001 0001, with *Carry_out* equal to 1. The control signals for inverting Y_i (i.e., *Cmpl_Y*), and distinguishing arithmetic operations from logic operations (i.e., *Op_ARITH*), and selecting p_i (i.e., *Op_XOR*) are equal to 1 from the

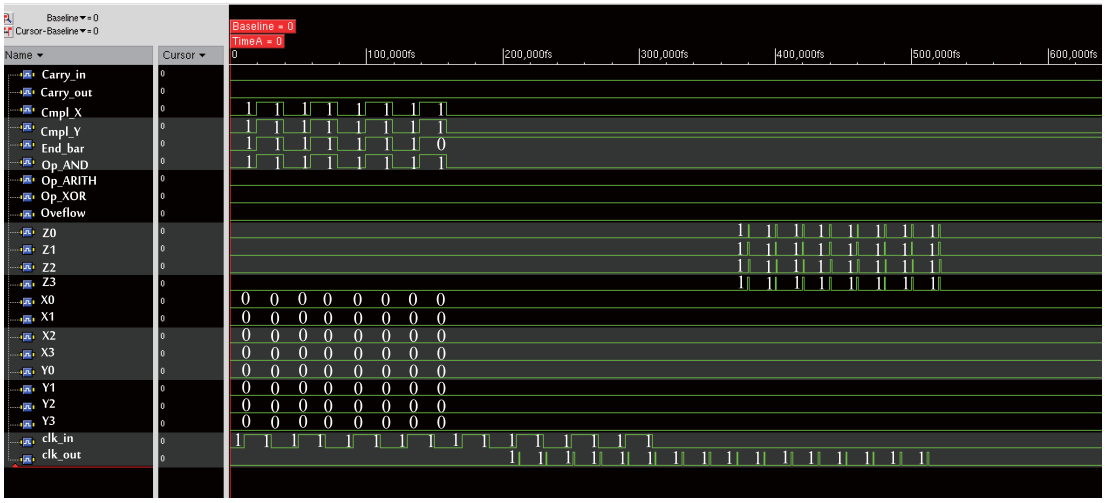


Figure 3.8. Simulation waveforms showing NOR operation at 50 GHz.

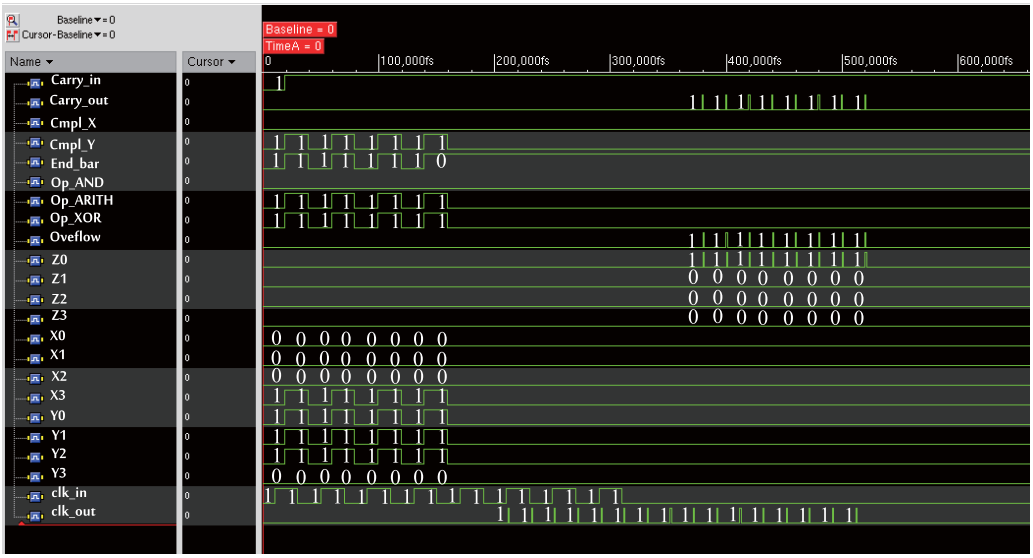


Figure 3.9. Simulation waveforms showing SUB operation at 50 GHz.

first to the eighth clock cycle. *End_bar* is the same as OR operation. *Carry_in* is equal to 1 at the first clock cycle. The simulation result shows that the path of carry network of the ALU and the control signals of *Op_ARITH* and *Carry_in* work correctly.

In addition, ADD, SLT, EQ, AND, and XOR operations are also successfully simulated at 50 GHz. The simulation results show that the proposed ALU can correctly work and perform all arithmetic/logic operations.

Then, in order to satisfy on-chip high frequency testing, high frequency clock generator (CG), DC-to-SFQ converters, 8-bit input shift registers (SR_IN), 8-bit output shift registers (SR_OUT), and SFQ-to-DC converters are attached to the physical layout of the 4-bit bit-slice ALU. The CG consists of 16-bit ladder SFQ pulse generator. Because the number of stages in the high frequency clock generator is sixteen, the total number of high frequency clock SFQ pulses generated from a single low frequency trigger pulse is sixteen. The high frequency clock is 50GHz because the SFQ pulse interval is 20ps, which is equal to the delay of one stage with the DC bias of 2.5 mV.

3.5 Fabrication and test

We fabricated the ALU with the attached circuits for high speed testing.

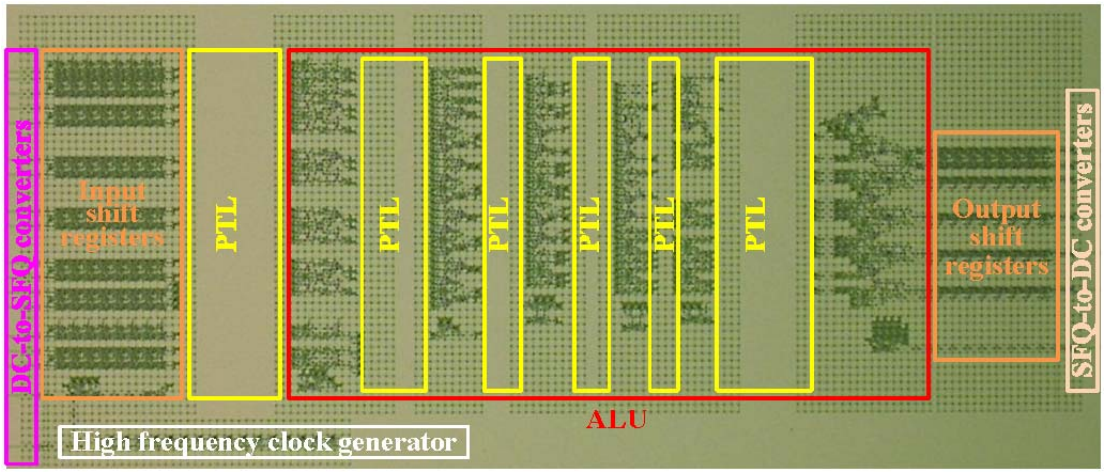


Figure 3.10. Micrograph of the test circuit of the 4-bit bit-slice ALU.

Figure 3.10 shows a micrograph of the fabricated chip. The ALU without the attached circuits consists of 3,481 Josephson junctions and occupies an area of 3.09 × 1.66 mm^2. The bias current is 404 mA and the latency for a 32-bit operation is 524 ps (20 ps × 15 cycles for calculating eight slices in 8-stage pipeline + 224 ps delay of the clock) with the bias voltage of 2.5 mV.

We tested the fabricated ALU at Fujimaki lab of Nagoya University. Figure 3.11 shows the chip mounted in a chip holder.

When an operation for 32-bit data is tested, firstly, eight low frequency clock

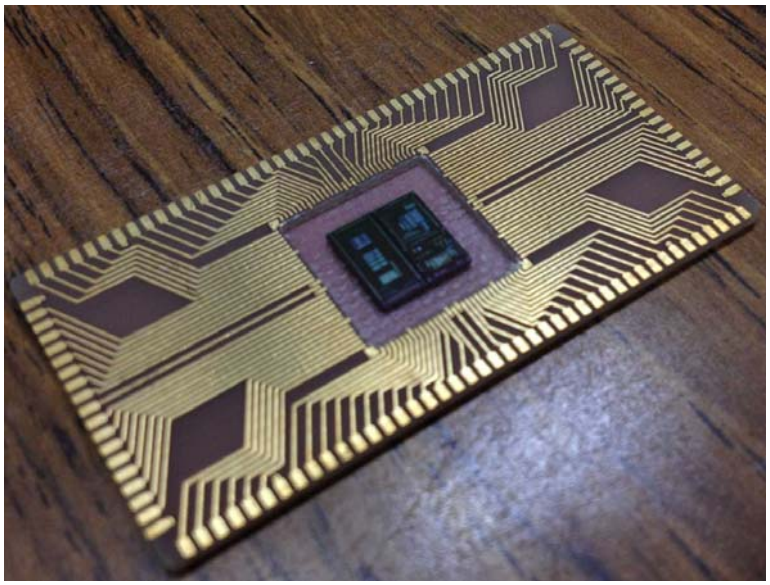


Figure 3.11. The chip of the 4-bit bit-slice ALU mounted in a chip holder.

pulses are used to load the operands divided into eight slices of four bits each into 8-bit input shift registers in room-temperature environment. Then one low frequency trigger signal is input into CG to generate sixteen high frequency clock pulses of 50 GHz. The 32-bit operation is operated in the ALU and the results of the eight slices of four bits each are stored into 8-bit shift registers using the sixteen on-chip high frequency clock pulses generated by CG in cryogenic temperature environment. Finally, the results of the eight slices of four bits each are read from the output shift registers and display on oscilloscope using eight low frequency clock pulses in room-temperature environment.

We successfully obtained correct results for all the ALU operations at high-speed clock for several test vectors, such as (1111 1111 1111 1111 1111 1111 1111 1111, 0000 0000 0000 0000 0000 0000 0000 0001) and pairs of random numbers. Figure 3.12 shows the test result of an ADD operation. Here, two 32-bit integers, operand *X* and operand *Y*, were successfully added at high speed: 1111 0111 1001 0110 1111 0111 1001 0110 + 1010 0011 0100 1101 1010 0011 0100 1101 = 1001 1010 1110 0100 1001

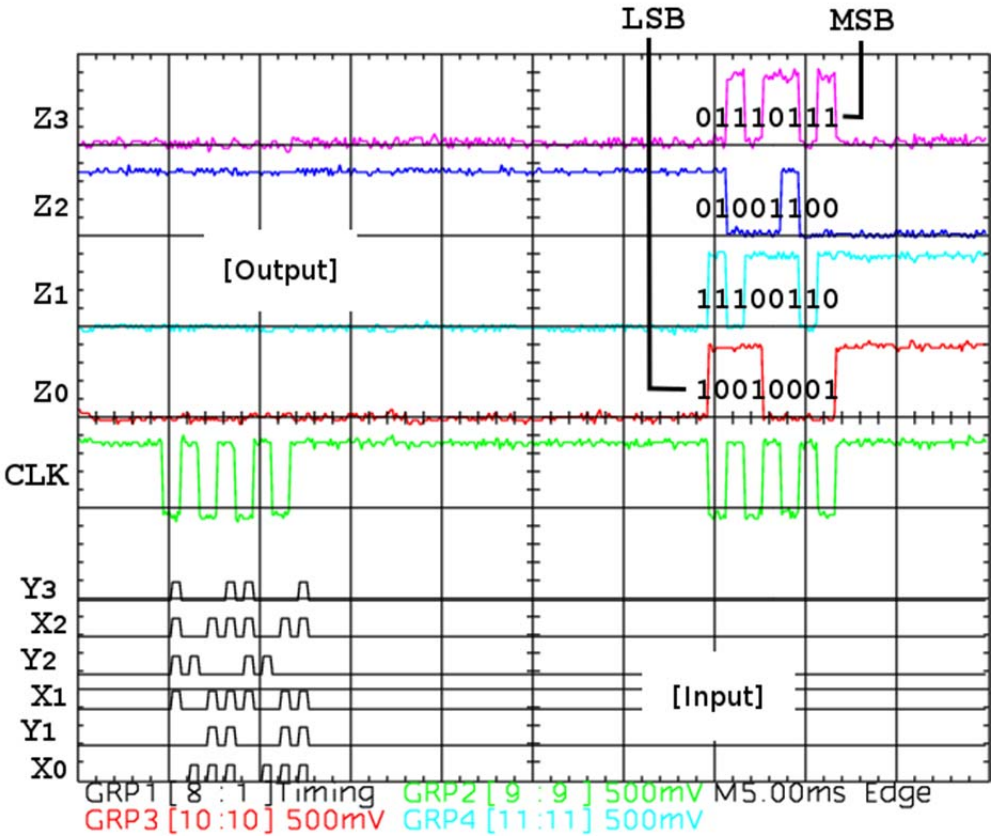


Figure 3.12. Output oscilloscope waveforms showing ADD operation.

1010 1110 0011, with *Carry_out* equal to 1 in the 15th cycle.

Figure 3.13 shows the test result of a SUB operation. Here, the values of operand *X* and operand *Y* are the same as those in the ADD operation described above. We successfully obtained *Z* = 0101 0100 0100 1001 0101 0100 0100 1001 and confirmed that *Carry_out* was equal to 1 in the 15th cycle.

In Figures 3.12 and 3.13, the six waveforms at the bottom correspond to the input signals for *X* and *Y*, where the input SFQ pulses are generated at rising edge of each input using DC-to-SFQ converters. The upper five waveforms represent the outputs of clock (CLK) and Z_0–Z_3, whose signal is inverted when output SFQ pulses arrive at the SFQ-to-DC converters.

We can only observe the low frequency signals of the input and output on the oscilloscope because the high frequency clock signals are too high to capture. Firstly, the test vectors are loaded in the 8-bit input shift registers using eight low frequency clock cycles (There is a SFQ pulse at the falling/rising edge of CLK because the output

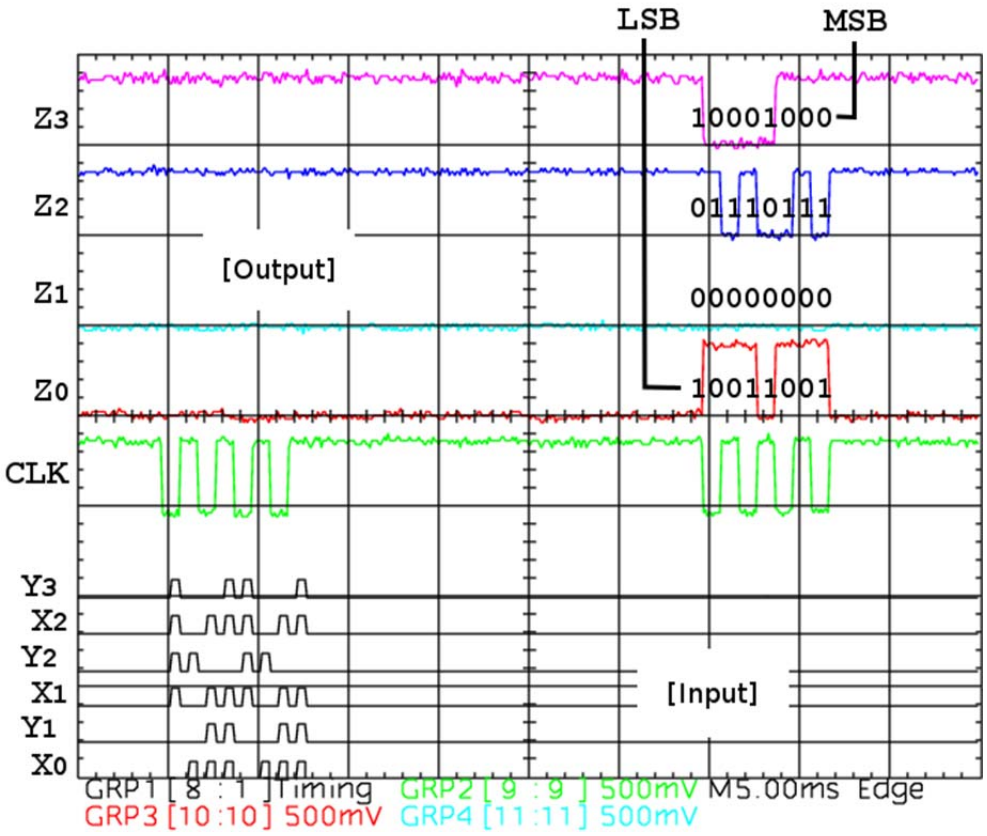


Figure 3.13. Output oscilloscope waveforms showing SUB operation.

voltage is inverted when a SFQ pulse arrives at SFQ-to-DC converter.) under room-temperature environment. Then, high frequency operations are performed and stored the results into the 8-bit output shift registers using sixteen on-chip high frequency clock cycles under cryogenic temperature environment. Finally, we can use eight low frequency clock cycles (about 1 kHz) to read the results from the 8-bit output shift registers under room-temperature environment. Based on the above reasons, the output signal of CLK is discontinuous.

The input signals for *X* and *Y* are only partly displayed due to the limitations of the oscilloscope. The number of waveforms we can display on the oscilloscope screen is limited due to the number of available channels of the oscilloscope and the setup of our measurement equipment. In addition, we sometimes reduce the number of displayed waveforms to avoid overlap in order for easy verification. The current configuration allows us to display up to eight input waveforms at the same time. In Figures 3.12 and 3.13, we selected X_0, Y_1, X_1, Y_2, X_2, and Y_3 signals within this limitation because we cannot display all the other input signals i.e., clk_in, ALU control signals, Y_0, and X_3 on the same screen. The *Carry_out* signal was not displayed in Figures 3.12 and Figure

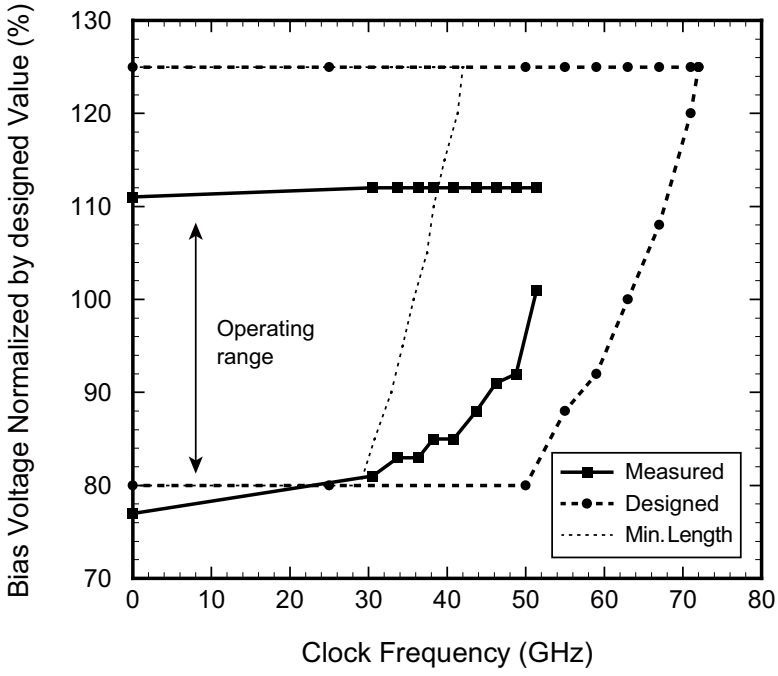


Figure 3.14. Dependence of DC bias margin on clock frequency: estimation assuming that all the PTLs are routed with Manhattan distance, simulation results from final design after adjustment of PTL lengths, and measured result.

3.13. We checked its correctness separately.

As shown at the bottom of Figures 3.12 and 3.13, the scale of horizontal axis is 5 ms/div. For the vertical axis, the inputs (X_0, Y_1, X_1, Y_2, X_2, and Y_3) are digital signals without scale (threshold voltage is approximately 0.7 mV), and the scale of outputs (CLK, Z_0–Z_3) is 500 mV/div. Please note that output signals of RSFQ circuits are amplified by 1000 (60 dB) before entering the oscilloscope.

The dependence of the measured DC bias margin on the clock frequency is shown in Figure 3.14, with the logic simulation results. After adjustment of PTL lengths, the operating frequency was estimated to increase by 43% ((50-35)/35) at 100% bias voltage, as compared to the design in which all the PTLs were routed with minimum length, i.e., Manhattan distance. (Recall that we employed ‘grid routing’.) Although the measured operating margin was slightly narrowed, we successfully achieved the maximum operating frequency of the designed 4-bit bit-slice ALU above the target clock frequency of 50 GHz. Figure 3.14 shows frequency dependence of bias margin. In general, switching speed of Josephson junction depends on bias voltage, and it becomes slower as bias voltage is lowered. That is why the lower margin of operating area

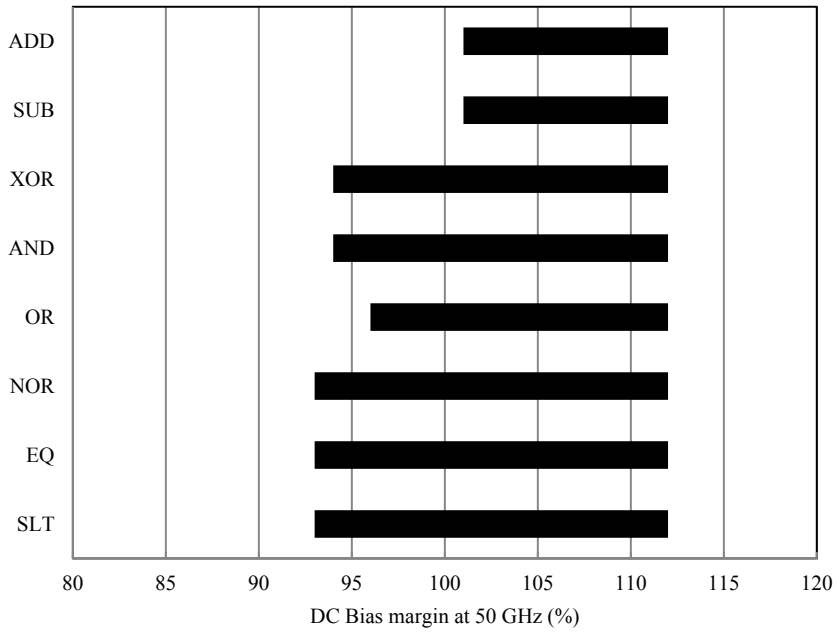


Figure 3.15. DC bias margins of all the operations of the proposed ALU.

shrinks for higher clock frequencies.

Figure 3.15 summarizes the DC bias margins of all the ALU operations (ADD, SUB, XOR, AND, OR, NOR, EQ, and SLT) of the fabricated 4-bit bit-slice ALU at 50 GHz, measured using the same test vector as the ADD operation described above. We obtained the measured DC bias margins with overlaps in the range of 101%–112%. Because the carry network and feedback loop are used for the ADD and SUB operations, the DC bias margins of the two operations have smaller ranges than the other operations.

3.6 Comparison of bit-slice, bit-serial and parallel RSFQ ALUs

We have designed RSFQ logic circuits and physical layouts of 2-/8-/16-bit bit-slice as well as bit-serial and 32-bit parallel ALUs with target clock frequency of 50 GHz, and have compared them.

(1) Bit-serial ALU

Figure 3.16 shows an RSFQ logic design of a bit-serial ALU. Its physical layout is shown in Figure 3.17. The ALU consists of 6 pipeline stages. It consists of 1182 Josephson junctions and occupies an area of 1.55 × 0.53 mm^2. The circuit delay is 135 ps. The bias current is 144.02 mA.

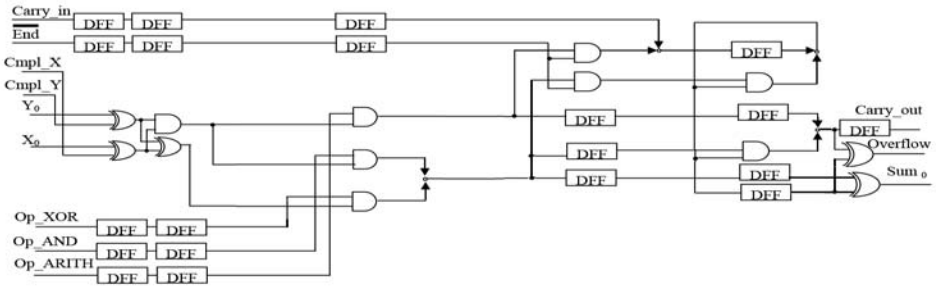


Figure 3.16. RSFQ logic design of a bit-serial ALU.

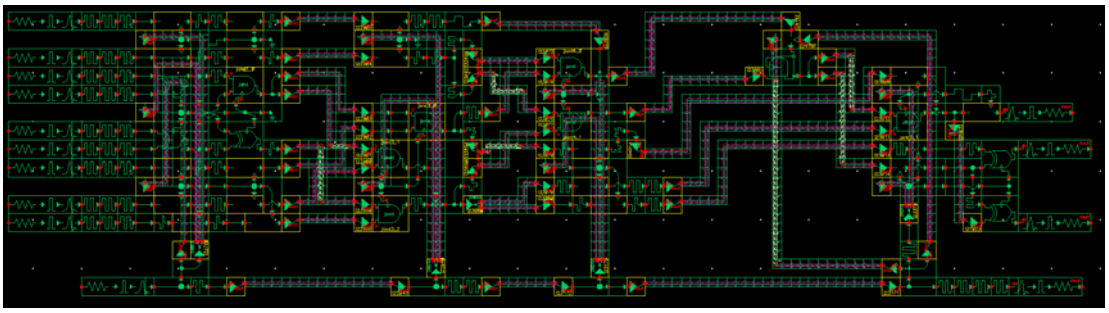


Figure 3.17. The physical layout of the bit-serial ALU.

(2) 2-bit bit-slice ALU

Figure 3.18 shows an RSFQ logic design of a 2-bit bit-slice ALU. Its physical layout is shown in Figure 3.19. The ALU consists of 7 pipeline stages. It consists of 2001 Josephson junctions and occupies an area of 2.84 × 0.90 mm^2. The circuit delay is 170 ps. The bias current is 238.68 mA.

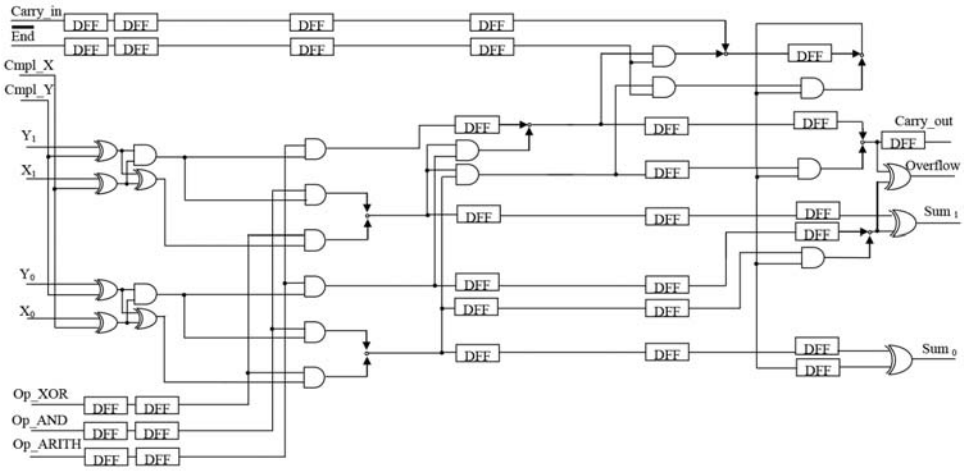


Figure 3.18. RSFQ logic design of a 2-bit bit-slice ALU.

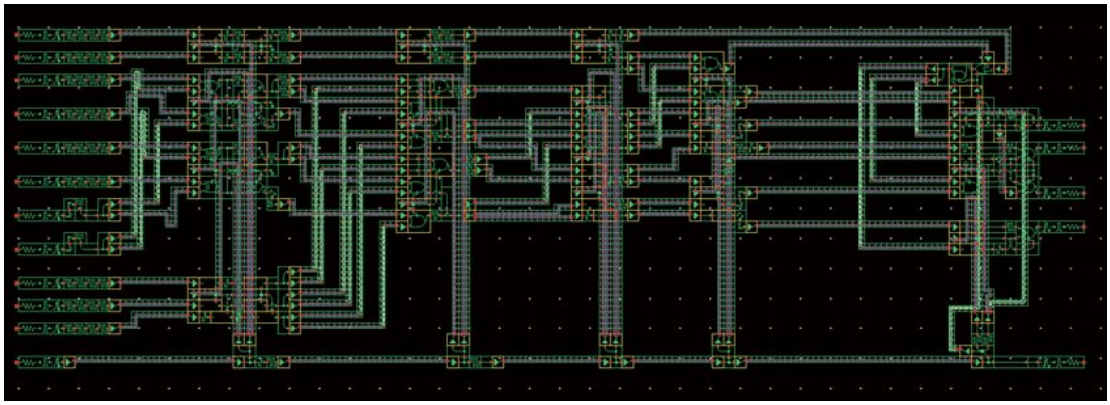


Figure 3.19. The physical layout of the 2-bit bit-slice ALU.

(3) 8-bit bit-slice ALU

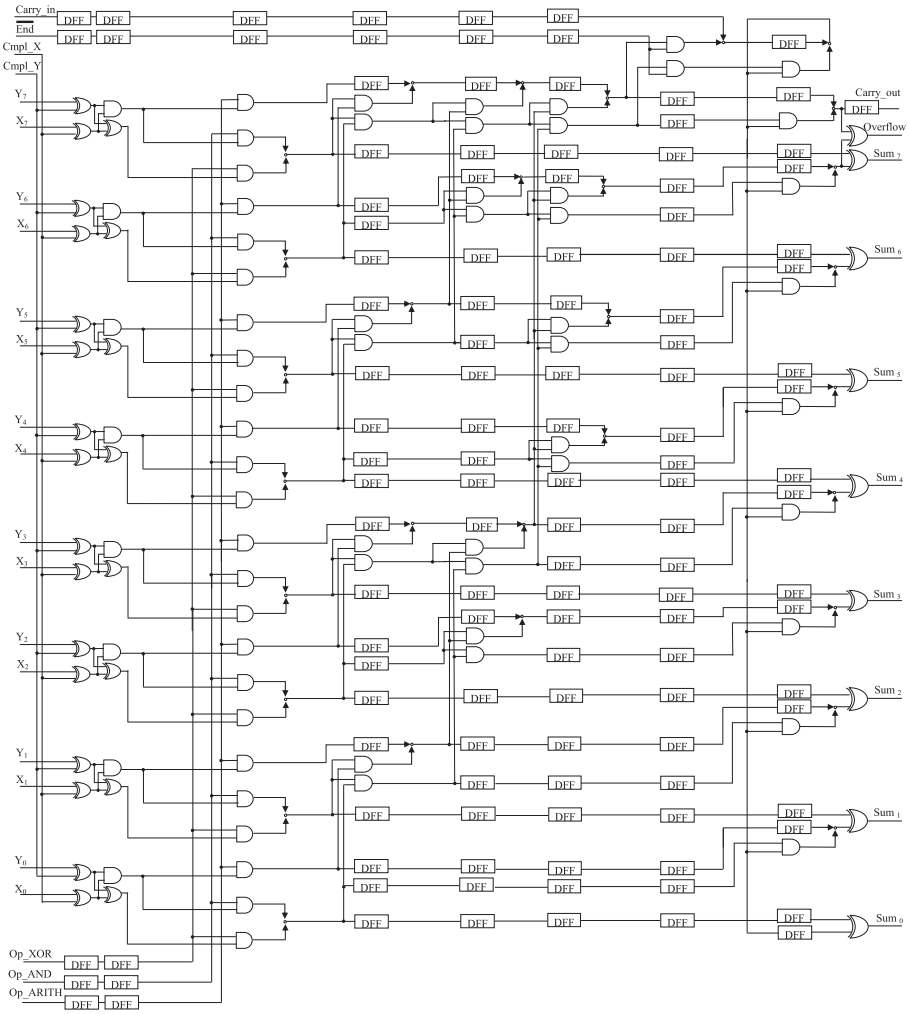


Figure 3.20. RSFQ logic design of an 8-bit bit-slice ALU.

Figure 3.20 shows an RSFQ logic design of an 8-bit bit-slice ALU. Its physical layout is shown in Figure 3.21. The ALU consists of 9 pipeline stages. It consists of 7812 Josephson junctions and occupies an area of 5.73 × 2.22 mm^2. The circuit delay is 297 ps. The bias current is 929.51 mA.

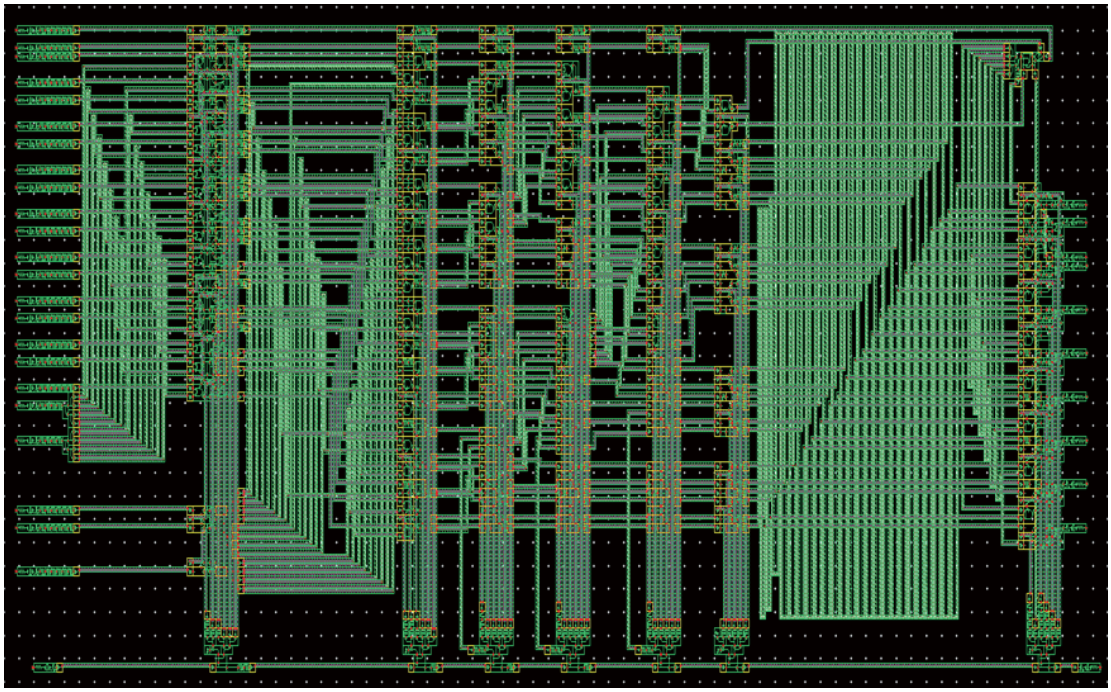


Figure 3.21. The physical layout of the 8-bit bit-slice ALU.

(4) 16-bit bit-slice ALU

Figure 3.22 shows an RSFQ logic design of a 16-bit bit-slice ALU. Its physical layout is shown in Figure 3.23. The ALU consists of 10 pipeline stages. It consists of 19,649 Josephson junctions and occupies an area of 8.97 × 4.13 mm^2. The circuit delay is 471 ps. The bias current is 2,429.25 mA.

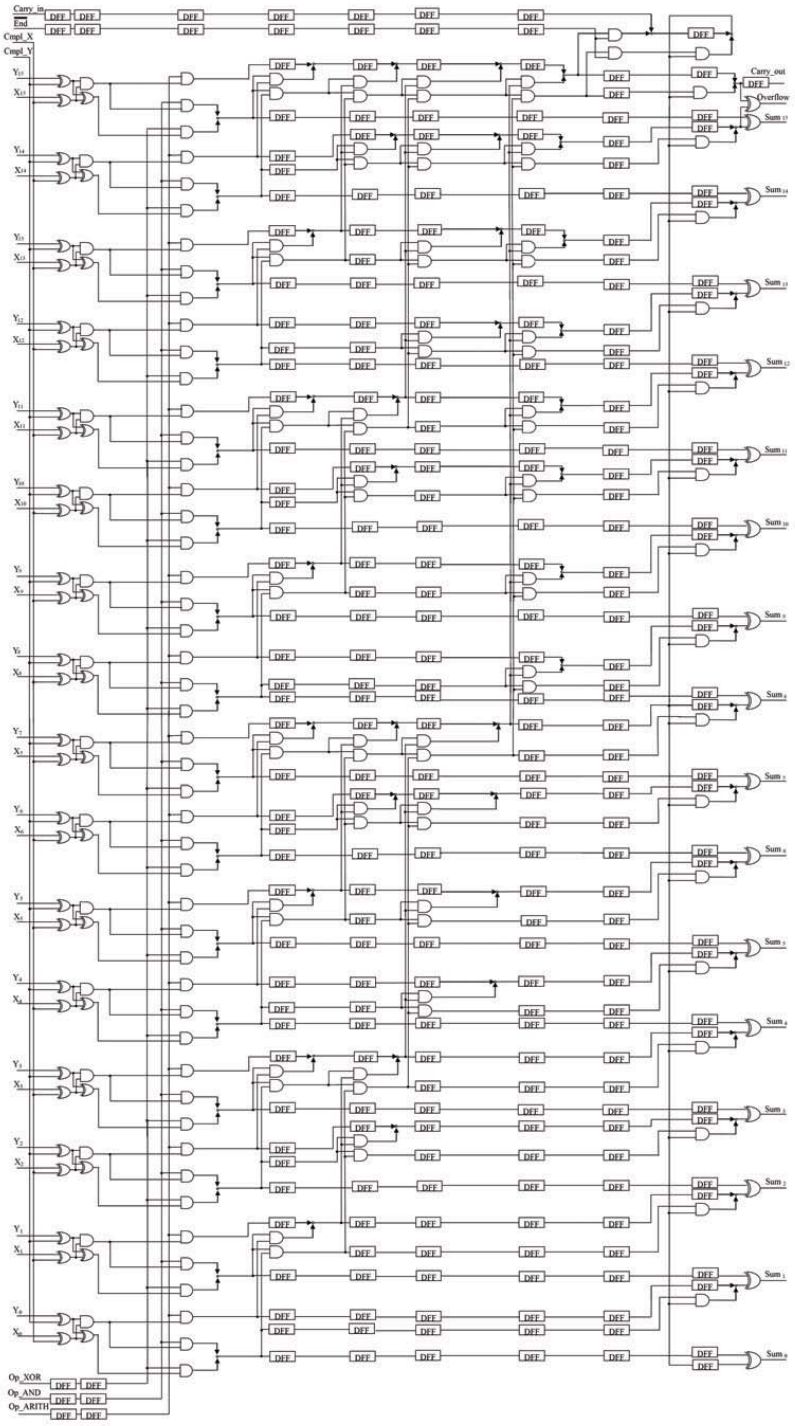


Figure 3.22. RSFQ logic design of a 16-bit bit-slice ALU.

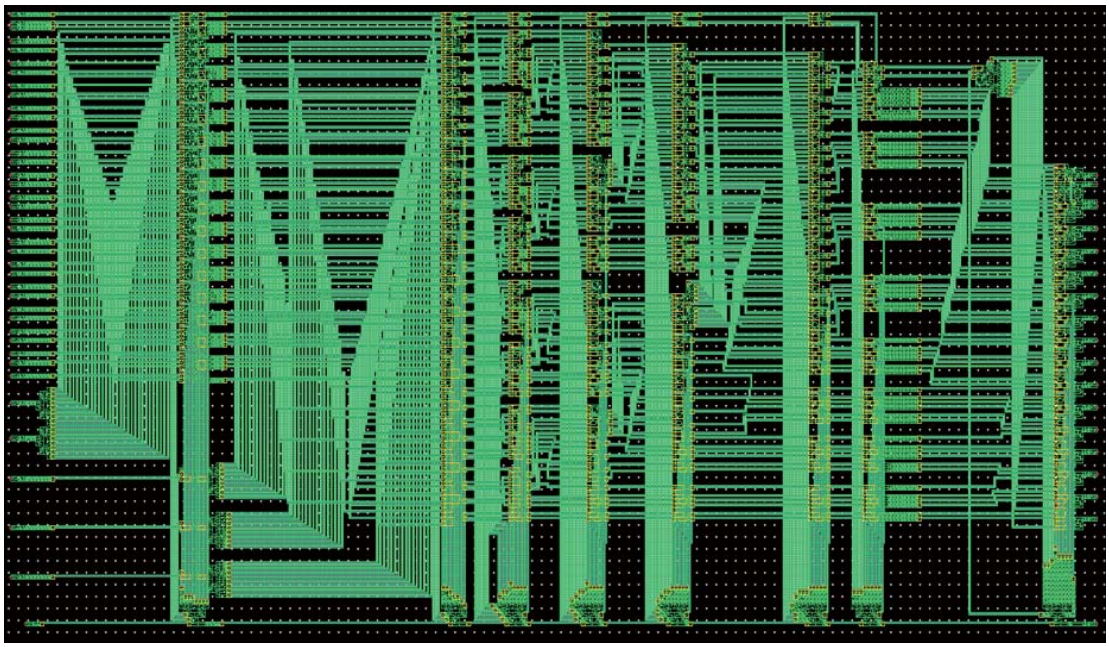


Figure 3.23. The physical layout of the 16-bit bit-slice ALU.

(5) 32-bit parallel ALU

Figure 3.24 shows an RSFQ logic design of a 32-bit parallel ALU. Its physical layout is shown in Figure 3.25. The ALU consists of 10 pipeline stages. It consists of 36,424 Josephson junctions and occupies an area of 11.16 × 9.14 mm^2. The circuit delay is 599 ps. The bias current is 4,410.69 mA.

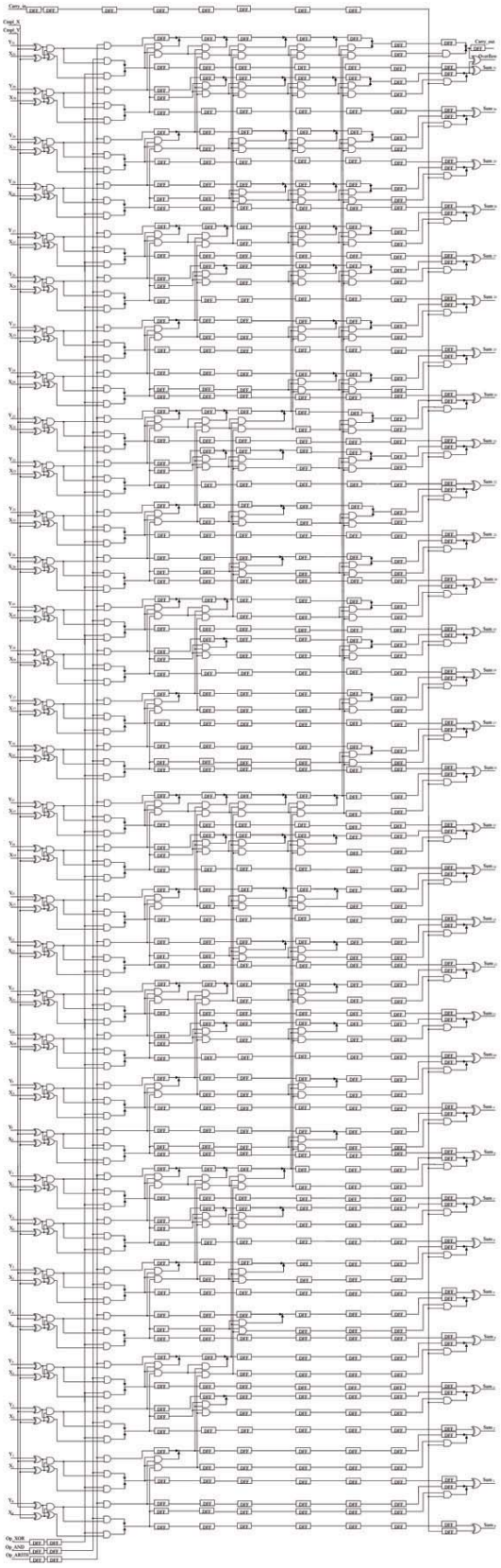


Figure 3.24. RSFQ logic design of a 32-bit parallel ALU.

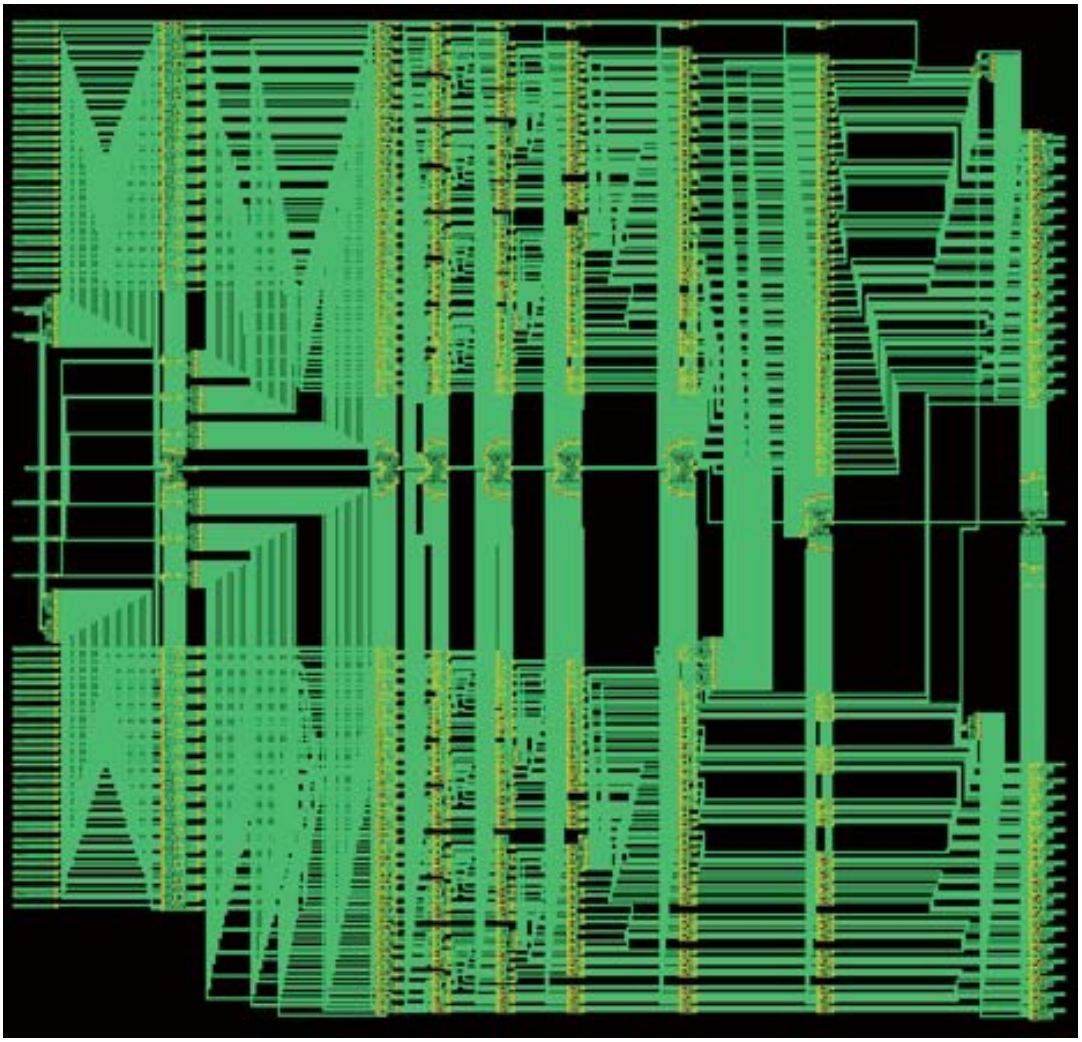


Figure 3.25. The physical layout of the 32-bit parallel ALU.

The number of cells used in the ALUs is shown in Table 3.2. The number of cells (i.e., and, or, xor, dff, cb, SPL, sink, DC-to-SFQ, SFQ-to-DC, and PTL Drivers and Receivers) increases linearly with the slice width. However, the number of JTL and PTL increases sharply from 8-bit to 16-bit bit-slice ALU because a large number of wiring cells (i.e., JTL and PTL) are used in 16-bit bit-slice and 32-bit parallel ALUs.

Table 3.2 The numbers of cells used in ALUs

Cell	bit-serial	2-bit	4-bit	8-bit	16-bit	parallel
and	8	15	31	67	147	320
or	1	1	2	3	4	8
xor	5	9	17	33	65	129
dff	19	31	57	118	259	478
cb	2	5	11	26	62	136
SPL	30	55	92	211	436	888
JTL	116	166	252	670	2154	3461
PTL	581	2601	9346	27650	80716	199033
sink	4	6	8	21	39	72
DC-to-SFQ	10	12	16	24	40	71
SFQ-to-DC	4	5	7	11	19	35
PTL Drivers and Receivers	69	158	300	625	1341	2693
Total	849	3064	10139	29459	85282	207324

Comparison of the ALUs is summarized in Table 3.3. As mentioned in Chapter 2, JLT and PTL have delay and JTL consists of Josephson junctions with bias current consumption. The number of JJs, bias current, and clock delay increases sharply in16-bit bit-slice and 32-bit parallel ALUs because the number of JTL and PTL increases sharply.

Table 3.3 Comparison of the main parameters in ALUs

Bit-slice size	JJs	Area (mm^2)	Bias current (mA)	Pipeline stages	Clock delay (ps)
bit-serial	1182	0.811	144.022	6	135
2-bit	2001	2.552	238.676	7	170
4-bit	3481	5.129	404.000	8	224
8-bit	7812	12.721	929.508	9	297
16-bit	19649	37.001	2429.245	10	471
parallel	36424	96.012	4410.687	10	599

The performance of the ALUs in processing 32-bit data is shown in Table 3.4.

Table 3.4 Performance of the ALUs in processing 32-bit data

Slice size of ALUs	Bit-serial	2-bit	4-bit	8-bit	16-bit	parallel
Clock frequency (GHz)	50	50	50	50	50	50
Throughput (32-bit operation per second) (GOPS)	1.56	3.13	6.25	12.5	25	50
Latency (ps)	875	610	524	537	691	779
Power consumption with the bias voltage of 2.5 mV (mW)	0.36	0.60	1.01	2.32	6.07	11.03
Operations per watt (Energy efficiency) (GOPS/W)	4333	5217	6188	5388	4119	4533

The latency of the ALUs is represented by the following formula:

$$latency = clock\ delay + \left(pipeline\ stages + \frac{word\ length}{slice\ width} - 1\right) \times clock\ period.$$

The 4-bit bit-slice ALU offers higher throughput than the bit-serial ALU and entails lower hardware cost than the 32-bit parallel ALU. It has the lowest latency in processing 32-bit data at 50 GHz. Furthermore, its energy efficiency is the highest.

3.7 Summary

In the design of bit-slice ALUs, there are four challenges:

(1) Make each feedback loop include no clocked gate so that pipeline processing of slices is continuous.

(2) Minimize the delay of each feedback loop.

(3) Make timing design easier.

(4) Make precise timing design.

The following approaches for responding to the challenges are adopted:

(1) Algorithm level: Development of the algorithm for reducing the number of feedback loops without a clocked gate.

(2) Logic level: Minimization of the delay of feedback loops, reduction of the number of stages with clocked gates.

(3) Layout level: Precise timing design by calculating the delay of wirings.

In this chapter, the following methods for responding to the challenges have been developed:

(1) Algorithm level: Sklansky adder with only one feedback loop without a clocked gate is used.

(2) Logic level: The circuit for producing the carry from the most significant bit of a slice is duplicated. The calculations with several signals are executed before entering the feedback loop. CBs are used to implement the OR operations in the PG network in order to reduce pipeline stages.

(3) Layout level: The all clocked gates in each stage are aligned in a row. The wiring lengths between two adjacent stages are adjusted so that the data arrival time of all gates in each stage is approximately equal.

We fabricated and successfully demonstrated high-frequency operation of the 4-bit bit-slice ALU. The proposed ALU consists of 3,481 Josephson junctions with an area of 3.09×1.66 mm^2. Testing results showed that the ALU has a latency of 524 ps and a throughput of 6.25×10^9 32-bit operations per second at 50 GHz. The proposed ALU can be used for any 4n-bit processing.

We have designed and compared 2-/4-/8-/16-bit bit slice as well as bit-serial and parallel ALUs. The 4-bit bit-slice RSFQ ALU has the lowest latency and the highest energy efficiency for processing 32-bit data at 50 GHz.

Chapter 4

Bit-slice multiplier

4.1 Introduction

A multiplier is one of the main components of microprocessors. In previous researches, an RSFQ 4×4-bit and an 8×8-bit parallel multiplier have been designed and fabricated [24], [25]. An RSFQ 24×24-bit bit-serial multiplier based on the systolic algorithm proposed in [26] has been designed and fabricated [27].

In general, an $n \times n$-bit multiplication requires n^2 bit-multiplications (ANDs) for generating n n-bit partial products and n-1 n-bit additions for summing up them. Therefore, an $n \times n$-bit parallel multiplier would include n^2 AND gates and $n(n\text{-}1)$ full adders. If ripple-carry adder is used to calculate the last results of multiplication, the number of full adders is also n^2. A logic design (without JTLs, PTLs, SPLs, PTL Drivers, and PTL Receivers) of an RSFQ 32×32-bit parallel multiplier would consist of more than 83,591 JJs using the AIST ADP2 fabrication process [17]. On the other hand, an RSFQ $n \times n$-bit bit-serial multiplier would have unacceptably high latency.

In this chapter, our study is focused on developing a 50 GHz 32×32-bit 4-bit bit-slice integer multiplier. For this purpose, we adopted the following design approaches: (1) Algorithm level: Development of the algorithm for reducing the number of feedback loops without a clocked gate, (2) Logic level: Minimization of the delay of feedback loops, reduction of the number of stages with clocked gates, and (3) Layout level: Precise timing design by calculating the delay of wirings.

We consider a 32×32-bit unsigned or signed integer multiplication, $Z = X \times Y$, where X (=[$x_{31}x_{30}\ldots x_1x_0$]) is a multiplicand, Y (=[$y_{31}y_{30}\ldots y_1y_0$]) is a multiplier, and Z (=[$z_{63}z_{62}\ldots z_1z_0$]) is the product. For unsigned integer multiplication, X is multiplied by

each bit of *Y* to generate the partial products, which are added to get *Z* represented by the following formula [26]:

$$Z = \sum_{l=0}^{4n-1} \sum_{k=0}^{4n-1} x_l y_k 2^{l+k}$$

Unlike unsigned integer multiplication, in which *X* is multiplied by each bit of *Y* to generate the partial products added to get *Z*, signed integer multiplication requires to invert the partial product bits multiplied by the sign bits of *X* and *Y* according to the following formula [26]:

$$Z = -2^{8n-1} + x_{4n-1} y_{4n-1} 2^{8n-2} + \sum_{k=0}^{4n-2} \overline{x_{4n-1} y_k} 2^{4n+k-1} + \sum_{l=0}^{4n-2} \overline{y_{4n-1} x_l} 2^{4n+l-1} + \sum_{l=0}^{4n-2} \sum_{k=0}^{4n-2} x_l y_k 2^{l+k} + 2^{4n}$$

Therefore, we need to design a control signal to generate the different partial products from unsigned integer multiplication.

Each of the 32-bit multiplicand *X* (=[$x_{31}x_{30}\ldots x_1x_0$]) and multiplier *Y* (=[$y_{31}y_{30}\ldots y_1y_0$]) is divided into eight slices of 4 bits each. The eight pairs of operand slices (i.e., *X*0-*X*7, *Y*0-*Y*7) are input one by one starting from the least significant one. The 64-bit product *Z* (=[$z_{63}z_{62}\ldots z_1z_0$]) is in sixteen 4-bit slices (i.e., *Z*0-*Z*15) which are output one by one from the least significant one.

The proposed multiplier is based on a newly developed systolic-like multiplication algorithm. A 32×32-bit 4-bit bit-slice multiplier is mainly composed of 8 almost identical systolic blocks.

The 32×32-bit 4-bit bit-slice multiplier was designed and simulated. For verifying the proposed algorithm and the logic design, a physical layout of the 8×8-bit 4-bit bit-slice multiplier have been fabricated.

The algorithm, logic design, physical layout design and simulation, and fabrication are described in Sections 4.2-4.5.

4.2 Algorithm

We develop a systolic-like multiplication algorithm.

Figure 4.1 illustrates a dot diagram of 32×32-bit multiplication. A dot in the figure represents a partial product bit or a bit of intermediate results or a final product bit. The 1024 partial product bits are divided into 72 groups as shown in Figure 4.2. Each group is labeled by a pair coordinate (j, i) (j = 0 to 8, i = 0 to 7). The positions of the sixteen 4-bit slices of final product $Z0$ to $Z15$ are labeled by '(0)' to '(15)'.

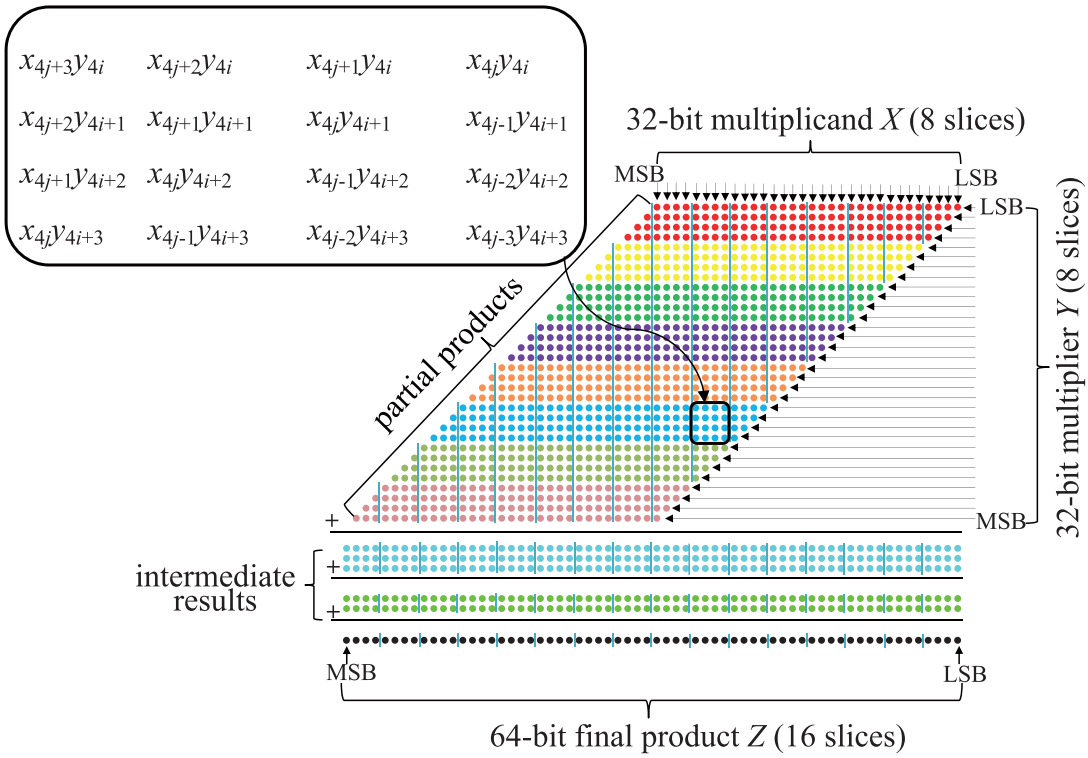


Figure 4.1. Dots diagram of 32×32-bit 4-bit bit-slice multiplication.

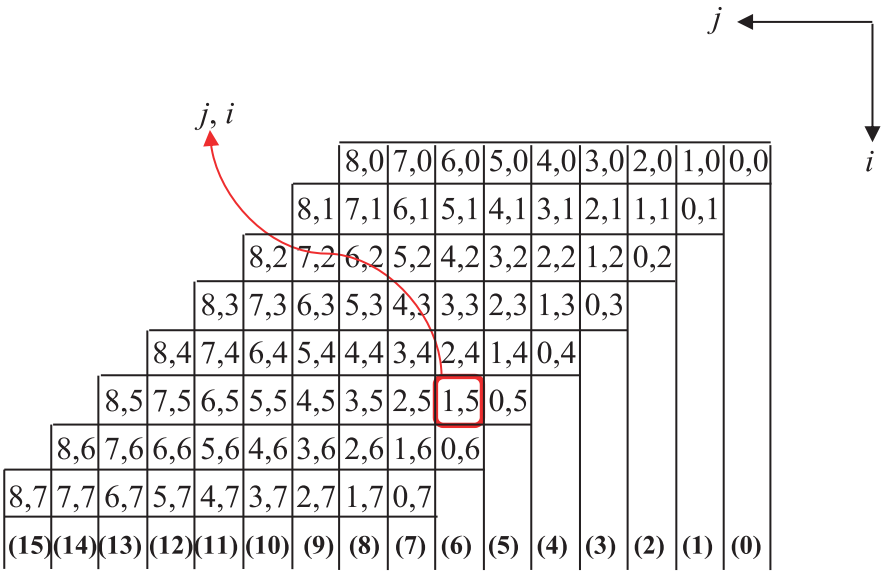


Figure 4.2. Block diagram of Figure 4.1.

Figure 4.3 shows systolic array for 32×32-bit 4-bit bit-slice multiplication. Slice Xi (=[x4i+3x4i+2x4i+1x4i]) and slice Yi (=[y4i+3y4i+2y4i+1y4i]), are input to the multiplier at step i (i=0 to 7). Slice Xi is fed to block0 and forwarded to the next block in every two steps. Slice Y_i is fed to block$_i$ and is kept there. The partial product generator (PPG) of block$_i$ generates four 4-bit slices of partial products (group 'j, i'. i.e., $[(x_{4j+3}y_{4i})\ (x_{4j+2}y_{4i})\ (x_{4j+1}y_{4i})\ (x_{4j}y_{4i})]$, $[(x_{4j+2}y_{4i+1})\ (x_{4j+1}y_{4i+1})\ (x_{4j}y_{4i+1})\ (x_{4j-1}y_{4i+1})]$, $[(x_{4j+1}y_{4i+2})\ (x_{4j}y_{4i+2})\ (x_{4j-1}y_{4i+2})\ (x_{4j-2}y_{4i+2})]$, and $[(x_{4j}y_{4i+3})\ (x_{4j-1}y_{4i+3})\ (x_{4j-2}y_{4i+3})\ (x_{4j-3}y_{4i+3})]$) at step $2i+j$ (j=0 to 8). ($x_l = 0$ for l<0 and l>31.). The accumulator of block$_i$ sums up the slices from PPG and the slices from block$_{i-1}$, and produces slices to block$_{i+1}$ at step i to $2i$+8.

Figures 4.4-4.8 show the 32×32-bit 4-bit bit-slice multiplication step-by-step.

At step 0, slices X_0 and Y_0 is fed to block$_0$. Y_0 is kept in block$_0$. The partial product '0, 0' is generated, and the addition for the position (0) is done. In the figure, bold (0) '**(0)**' denotes the final result of position (0).

At step 1, slice X_1 is fed to block$_0$. Y_1 is fed to block$_1$ and kept here. Slice X_0 stays in block$_0$. **(0)** is moved into block$_1$. The partial product '1, 0' is generated, and the addition for position (1) is carried out in block$_0$. In the figure, non-bold (1) '**(1)**' denotes an intermediate result of position (1).

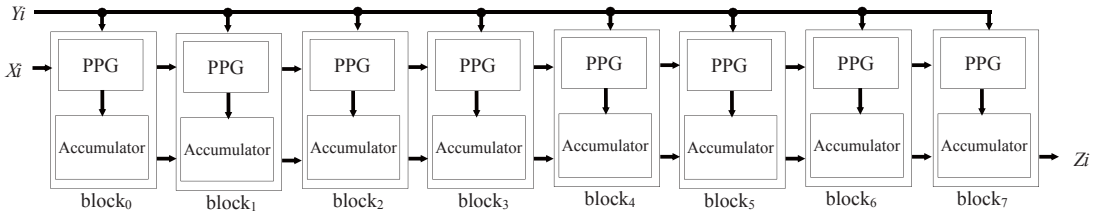


Figure 4.3. Block diagram of 32×32-bit 4-bit bit-slice multiplication.

At step 2, slice X_2 is fed to block$_0$. Y_2 is fed to block$_2$ and kept here. Slice X_0 is moved into block$_1$. Slice X_1 stays in block$_0$. **(0)** is moved into block$_2$. (1) is moved into block$_1$. In block$_0$, the partial product '2, 0' is generated, and the addition for position (2) is carried out. In block$_1$, the partial product '0, 1' is generated, and the addition for position (1) is carried out.

Then, at step 8, the result for position (0) is output.

Final, at step 23, the result for position (15) is output.

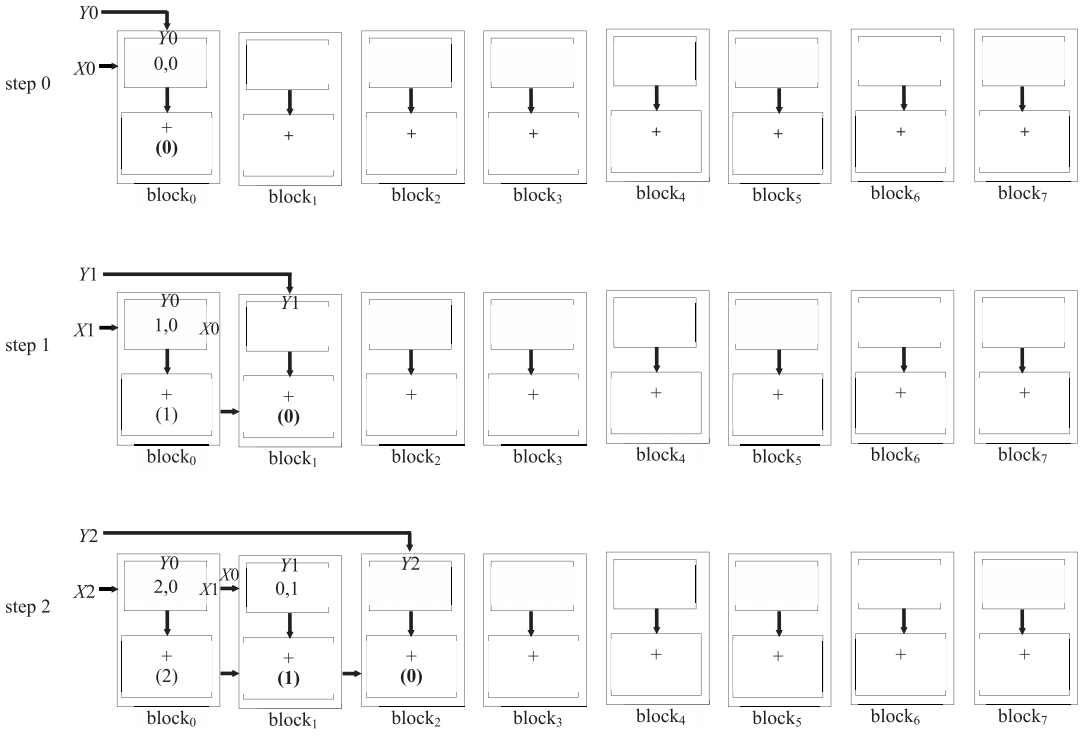


Figure 4.4. Step-by-step (step0-2) of 32×32-bit 4-bit bit-slice multiplication.

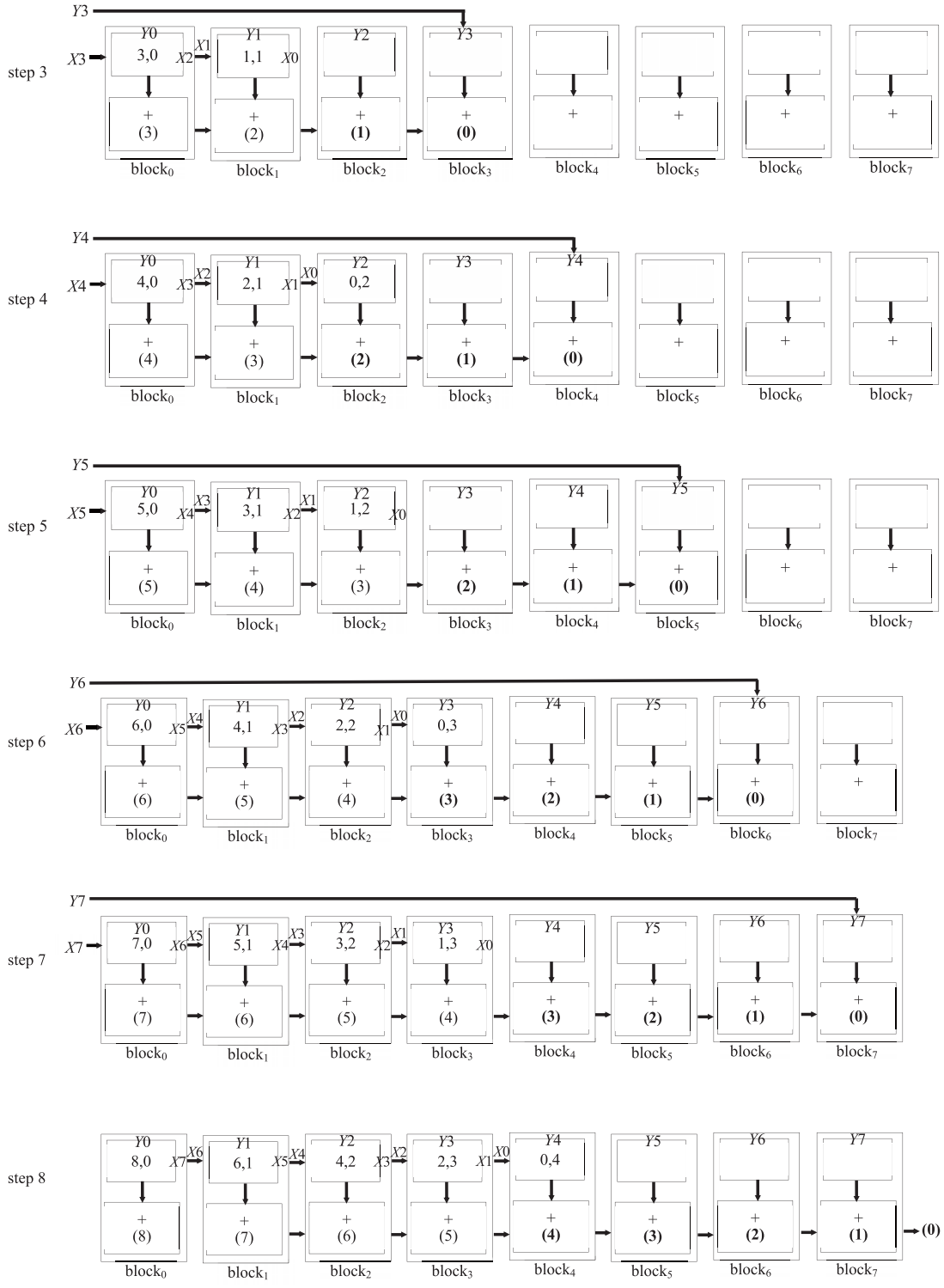


Figure 4.5. Step-by-step (step3-8) of 32×32-bit 4-bit bit-slice multiplication.

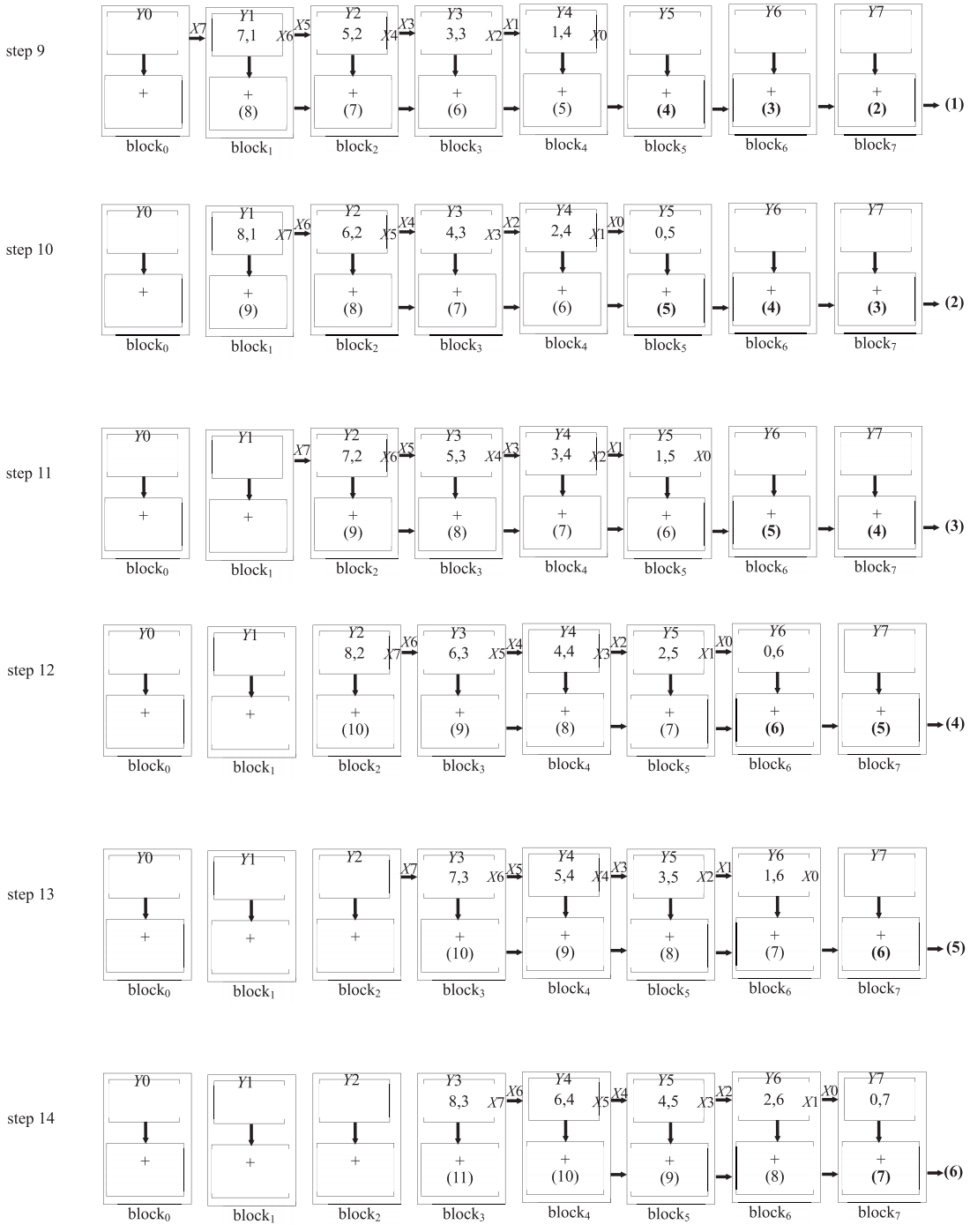


Figure 4.6. Step-by-step (step9-14) of 32×32-bit 4-bit bit-slice multiplication.

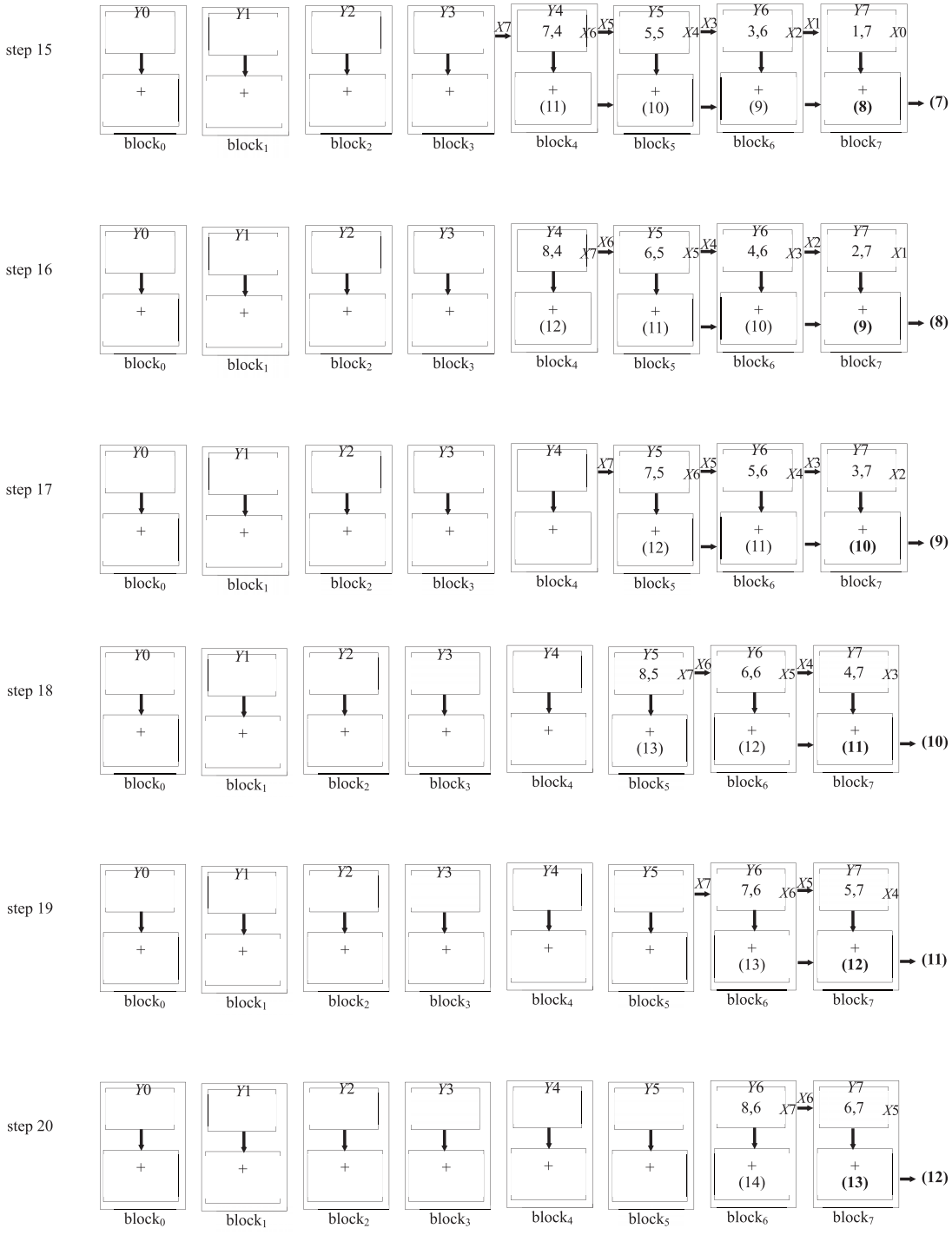


Figure 4.7. Step-by-step (step15-20) of 32×32-bit 4-bit bit-slice multiplication.

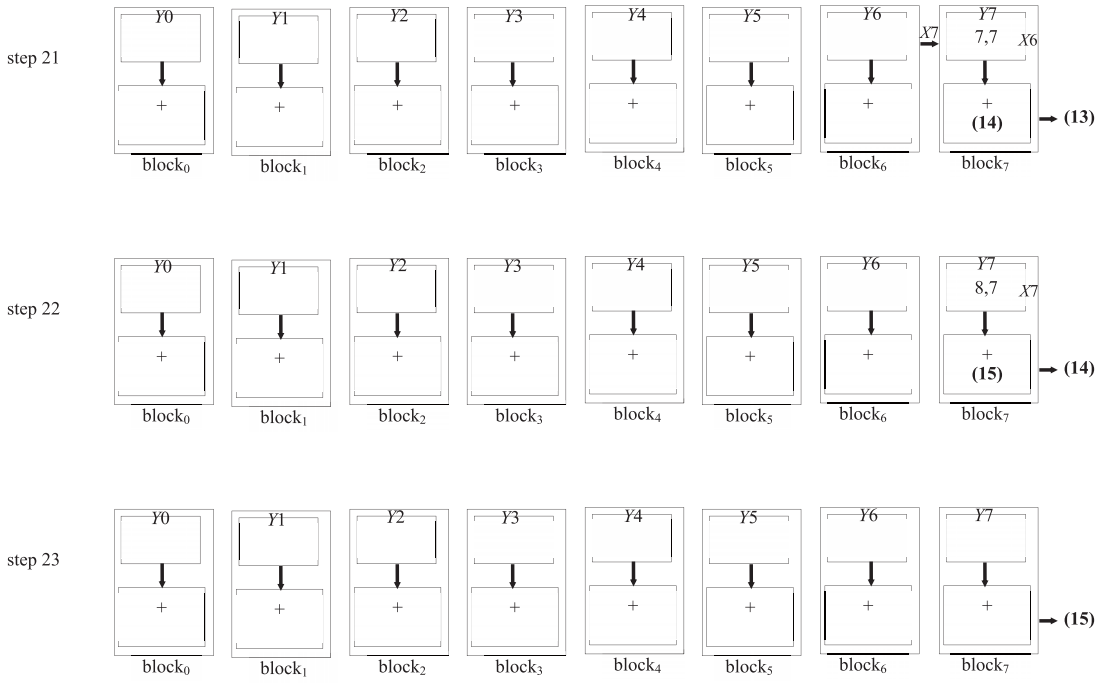
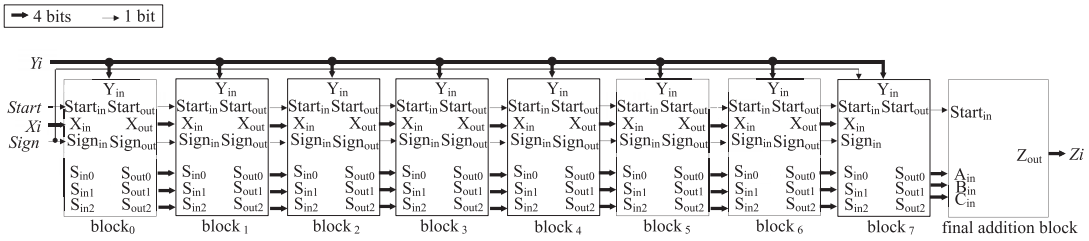


Figure 4.8. Step-by-step (step21-23) of 32×32-bit 4-bit bit-slice multiplication.

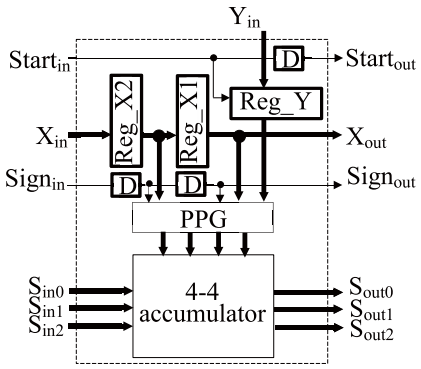
We adopt carry save addition for calculation in the accumulators. Then, we need carry propagate addition to the output of the systolic array.

Figure 4.9(a) shows a block diagram of a 32×32-bit 4-bit bit-slice multiplier based on the algorithm. It is composed of eight main blocks and a final addition block. As shown in Figure 4.9(b), a main block consists of a PPG and a '4-4 accumulator', as well as registers, 'Reg_X1', 'Reg_X2', and 'Reg_Y', for keeping two slices of *X* and a slice of *Y*, and D flip-flops for keeping control signals. The 4-4 accumulator sums up the four 4-bit partial product slices from the PPG and three 4-bit slices from the preceding block through carry save additions, and produces three 4-bit slices to the succeeding block.

The PPG for the most significant $block_7$ is slightly different from the $block_{0\text{-}6}$ for handling the sign in signed multiplication. The $block_7$ also has an additional register for signal *Sign* which indicates signed multiplication.



(a) The bolck diagram including eight main systolic blocks and a final additon block



(b) A main systolic block

Figure 4.9. 32×32-bit 4-bit bit-slice multiplier structural block diagram.

The final addition block consists of a 4-bit bit-slice 3-to-2 compressor and a 4-bit bit-slice adder. We use a bit-slice adder based on Sklansky adder as in the ALU in Chapter 3. There is only one feedback loop at this bit-slice adder in the whole multiplier.

A 32×32-bit multiplication is carried out through 25 steps. At the step 0, signal *Start* is input to the multiplier with the least significant operand slices *X*0 and *Y*0. It is fed to block$_0$ and forwarded to the next block every step. Therefore, it reaches block$_i$ at step i (i=0 to 7). The i-th pair of the operand slices, i.e., *Xi* (=[$x_{4i+3}x_{4i+2}x_{4i+1}x_{4i}$]) and *Yi* (=[$y_{4i+3}y_{4i+2}y_{4i+1}y_{4i}$]), is input to the multiplier at step i. *Xi* is fed to block$_0$ and forwarded to the next block in every two steps. *Yi* is set to Reg_Y of block$_i$ by signal *Start* which

reaches block$_i$ then, and is kept there. Signal *Sign* is input to the multiplier at step 7, and is immediately split into two. One is fed to block$_0$ and forwarded to the next block every two steps along with *X*7. The other is sent to block$_7$ directly, and is set into the additional register by signal *Start*. Signal *Sign* is used because partial product bits $x_{31}y_k$ (k=0 to 30) and $x_l y_{31}$ (l=0 to 30) must be inverted, and the correction terms (i.e., -2^{63} and $+2^{32}$) must be added in signed multiplication.

In the final addition block, the 4-bit bit-slice 3-to-2 compressor sums up the three 4-bit slices from block$_7$ and produces two 4-bit slices, and then, the 4-bit bit-slice adder sums up these two 4-bit slices and produces a 4-bit slice of the final product at step 9 to 24.

4.3 RSFQ Logic design

A block diagram of 4-4 accumulator is shown in Figure 4.10. A 4-4 accumulator, in which a carry save type addition is performed, consists of four 4-bit carry save adders each of which is a row of four FAs. In the figure, the upper side is the LSB side and the lower side is the MSB side. The three 4-bit slices from the preceding block are input from S_{in0}, S_{in1} and S_{in2}. The produced three 4-bit slices are output to S_{out0}, S_{out1} and S_{out2}. The DFFs are required in order to keep the carries from the most significant position of the corresponding slices and add them to the least significant position of the succeeding slices.

A fully pipelined synchronous RSFQ logic design of the proposed multiplier using concurrent flow clocking is considered. Namely, each pipeline stage consists of a row of RSFQ clocked logic gates. The basic RSFQ logic gates and flip-flops: AND, XOR, NOT, DFF, and NDRO (non- destructive read-out), and wiring elements: JTL (Josephson transmission line), SPL (splitter), and CB (confluence buffer) in the cell library for the AIST ADP2 are used.

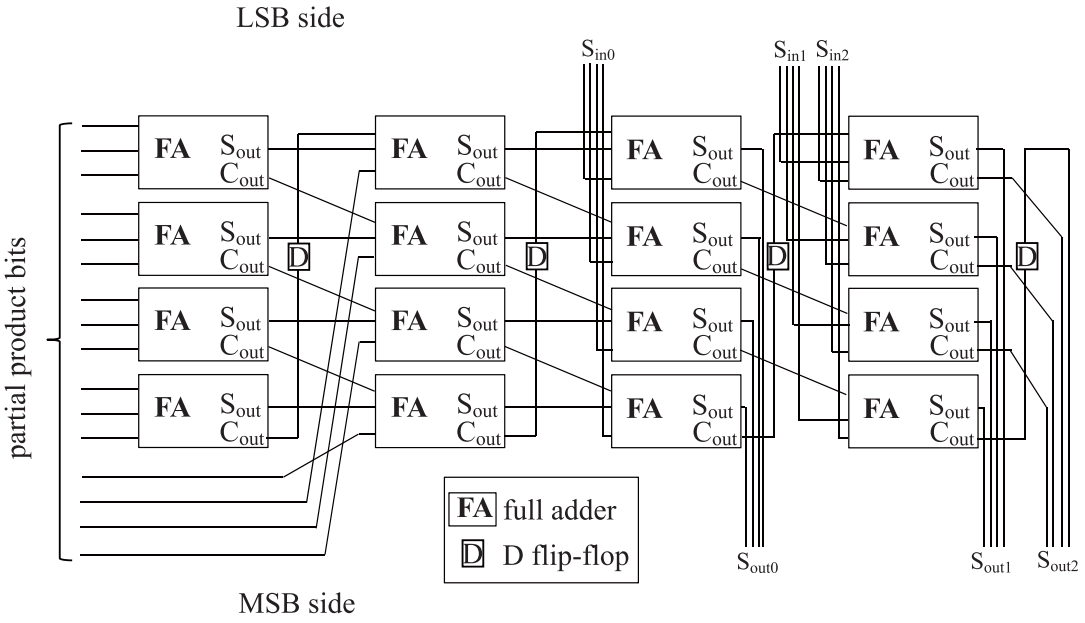
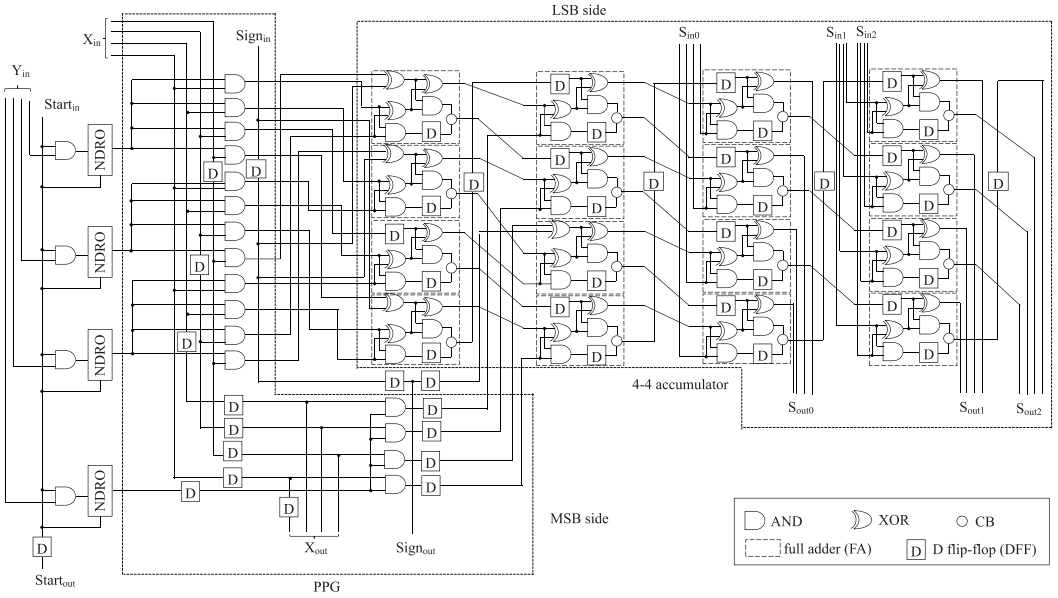


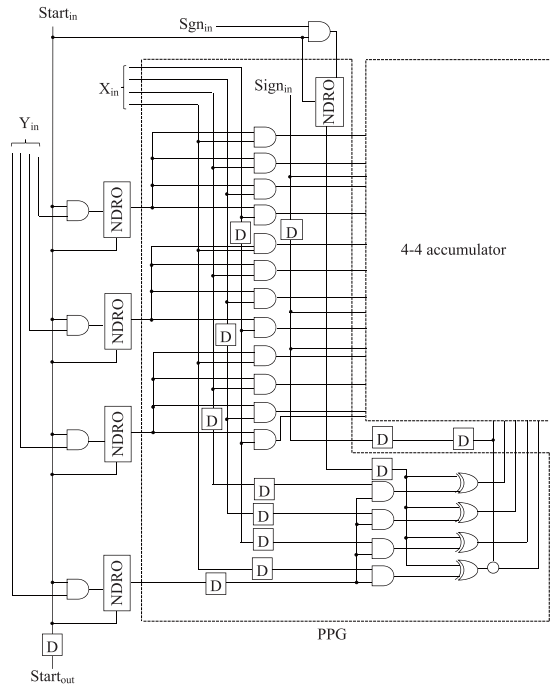
Figure 4.10. Block diagram of a 4-4 accumulator.

Figure 4.11 (a) shows an RSFQ logic design of a normal main block. (In the figure, the upper side is the LSB side and the lower side is the MSB side.) Reg_Y is implemented using four NDROs. Each of Reg_X1 and Reg_X2 is implemented using four DFFs. PPG is implemented using 16 AND gates and 4 XOR gates for inverting $x_{31}y_{4i}$, $x_{31}y_{4i+1}$, $x_{31}y_{4i+2}$ and $x_{31}y_{4i+3}$ in signed multiplication. These XOR gates are combined with full adders of the 4-4 accumulator. 4-4 accumulator consists of four 4-bit carry save adders each of which is a row of four full adders. In order to keep the carries from the most significant position of the corresponding slices and add them to the least significant position of the succeeding slices, the DFFs are required. Each full adder is implemented using two AND gates, two XOR gates, two DFFs, and a CB. In four of the full adders, one of the DFFs is replaced by the XOR gate for inverting a partial product bit. In the case of signed multiplication, one of the correction terms $+2^{32}$ is generated from signal *Sign* and is fed to $block_0$ through S_{in0}.

Figure 4.11 (b) shows an RSFQ logic design of the final main block, $block_7$, except a 4-4 accumulator which is the same as that in the normal main blocks. The additional register for keeping signal *Sign* is implemented using an NDRO. In signed



(a) An RSFQ logic design of a normal main block ($block_{0\text{-}6}$).



(b) An RSFQ logic design of the final main block ($block_7$).

Figure 4.11. The RSFQ logic designs of the main blocks.

multiplication, $x_{4j}y_{31}$, $x_{4j-1}y_{31}$, $x_{4j-2}y_{31}$, and $x_{4j-3}y_{31}$ are inverted by XORs. The other correction term -2^{63} is generated from signal *Sign*.

Each main block consists of 9 stages. Adjacent main blocks are aligned in 2 stages off. Namely, the clock cycles for the latter 7 stages are overlapped with those for the former 7 stages of the succeeding block.

An RSFQ logic design of the final addition block is shown in Figure 4.12. The 3-to-2 compressor is a carry save adder, i.e., a row of four full adders. A type of parallel prefix adder called Sklansky adder is used for the bit-slice adder designed in Chapter 3. The block consists of 8 stages.

The 32×32-bit multiplier consists of 33 stages in total, where two stages for setting Y_0 to block$_0$, 23 for the eight main blocks and 8 for the final addition block. Eight pairs of operand slices are fed at the first to the 8-th clock cycles, and sixteen slices of the

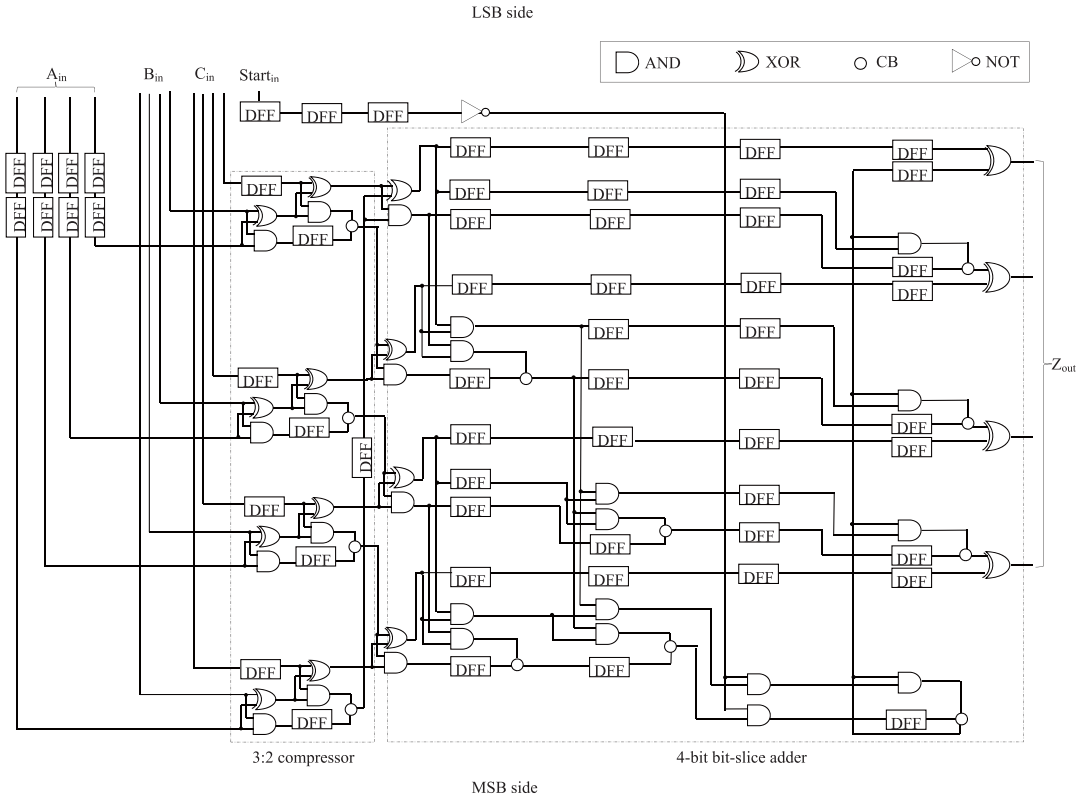


Figure 4.12. A logic gate level circuit of the final addition block.

product is output at the 34-th to 49-th clock cycles. The latency for a 32×32-bit multiplication is 49 clock cycles (plus clock delay for the circuits). The multiplier can

carry out a 32×32-bit multiplication every 16 clock cycles. The logic design (without JTLs, PTLs, SPLs, PTL Drivers, and PTL Receivers) consists of 17,951JJs.

4.4 Physical layout design and simulation

A physical layout of a 32×32-bit 4-bit bit-slice multiplier was designed using the AIST ADP2. In order to reduce the latency of the multiplier, PTLs are used for the wiring between adjacent pipeline stages.

Figure 4.13 shows a physical layout design of the multiplier. The clock signal propagates on the path running in the bottom side of the physical layout. Clock wires for each pipeline stage branch from a point on the main clock path and run from the bottom to the top of the layout. A 4-4 accumulator is designed in a shape of parallelogram so as to align the logic gates of adjacent blocks in the same pipeline stage. Thus, the data signal flow from the left to the right, except for the final addition block where they flow from the top to the bottom. These considerations in physical design are advantageous for the robustness of the circuit functionality.

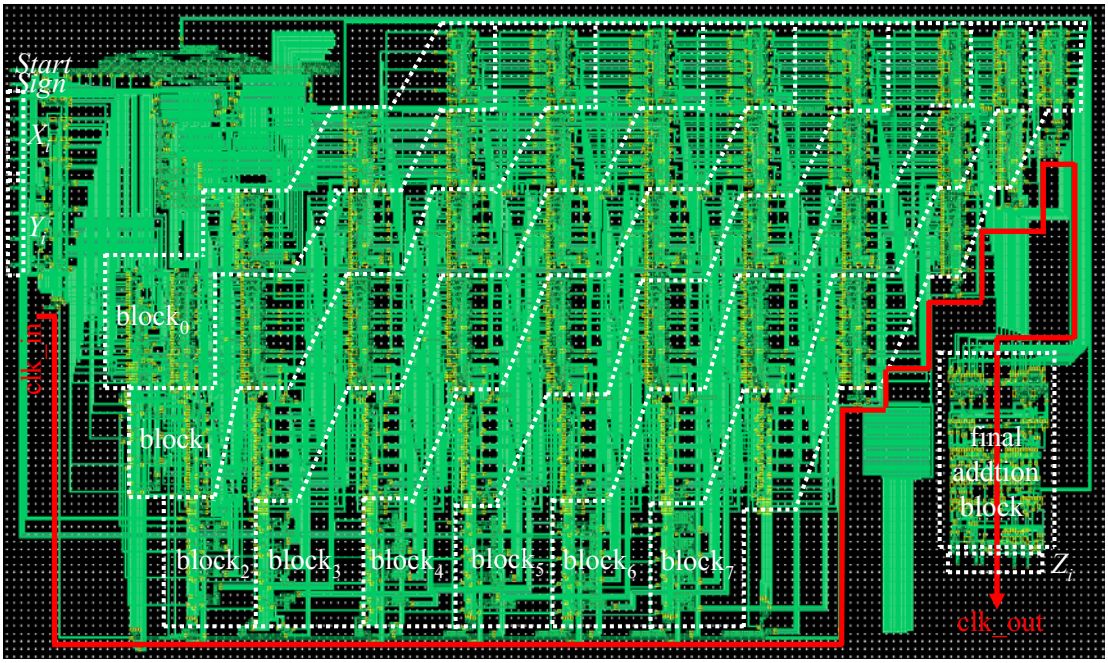


Figure 4.13. The physical layout of a 32×32-bit 4-bit bit-slice multiplier.

As the target frequency is 50 GHz, data signals are adjusted to arrive at each gate about 10 ps later than the clock signal. The critical point which limits the operation frequency is the feedback loop of the last stage in the final addition block, in which the carry required to be fed to the next slice. In order to minimize the feedback loop, we adopted the technique developed in the 4-bit bit-slice ALU designed in Chapter 3. In order to achieve the clock frequency of 50 GHz, we adjusted the PTLs lengths between two adjacent stages to implement the data arrival time of all gates in each stage is the approximate same value.

The physical layout (including JTLs, PTLs, SPLs, PTL Drivers, and PTL Receivers) of the multiplier consists of 56,885 Josephson junctions and occupies an area of 12.00×6.65 mm^2. The bias current is 7,211 mA.

In order to verify the logic design and the physical layout design, we simulated the designed circuit. We successfully obtained correct results for 32×32-bit operations for several special simulation vectors and random vectors at 50 GHz.

The clock delay amounts 1,251 ps; 50 ps at the input of $block_0$, 1,038 ps for the main blocks, and 163 ps for the final addition block. The latency for a 32×32-bit multiplication is 2,231 ps (20 ps × 49 cycles + 1,251 ps delay of the clock for the circuit) with the bias voltage of 2.5 mV. The throughput is 3.125×10^9 32×32-bit multiplications per second.

Figure 4.14 shows the simulation result of a 32×32-bit unsigned multiplication:

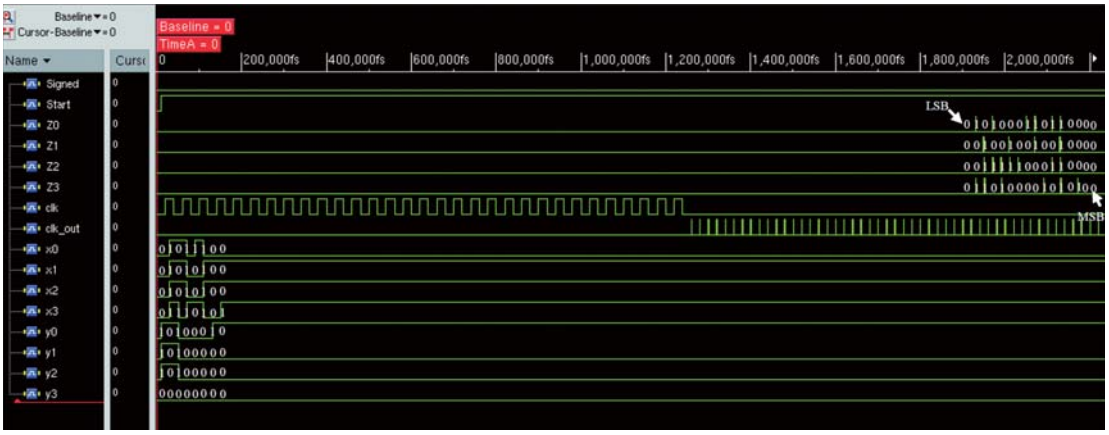


Figure 4.14. A simulation result of a 32×32-bit 4-bit bit-slice multiplier at 50 GHz.

1000 0000 1111 0001 1111 1000 1111 0000 × 0000 0001 0000 0000 0000 0111 0000 0111 = 0000 0000 1000 0000 1111 0101 1000 0011 0001 0100 0110 1100 0101 1110 1001 0000. In the figure, the eight waveforms at the bottom correspond to the input signals for *X* and *Y*, where the input pulse is generated at each rising/falling edge. The four waveforms at the top right represent the outputs of *Z*.

4.5 Fabrication

An 8×8-bit 4-bit bit-slice multiplier was fabricated for verifying our algorithm and logic design. It includes all components (the normal main block (i.e., block0), the final main block (i.e., block7), and the final addition block) of the 32×32-bit 4-bit bit-slice multiplier.

We fabricated 8×8-bit 4-bit bit-slice multiplier with shift-registers for high frequency input and output (SR_IN and SR_OUT), and we employed a ladder-type built-in high frequency clock generator (CG) for on-chip testing [35].

A micrograph of the multiplier is shown in Figure 4.15. The 8×8-bit 4-bit bit-slice multiplier (without DC-to-SFQ converters, input shift registers, output shift registers,

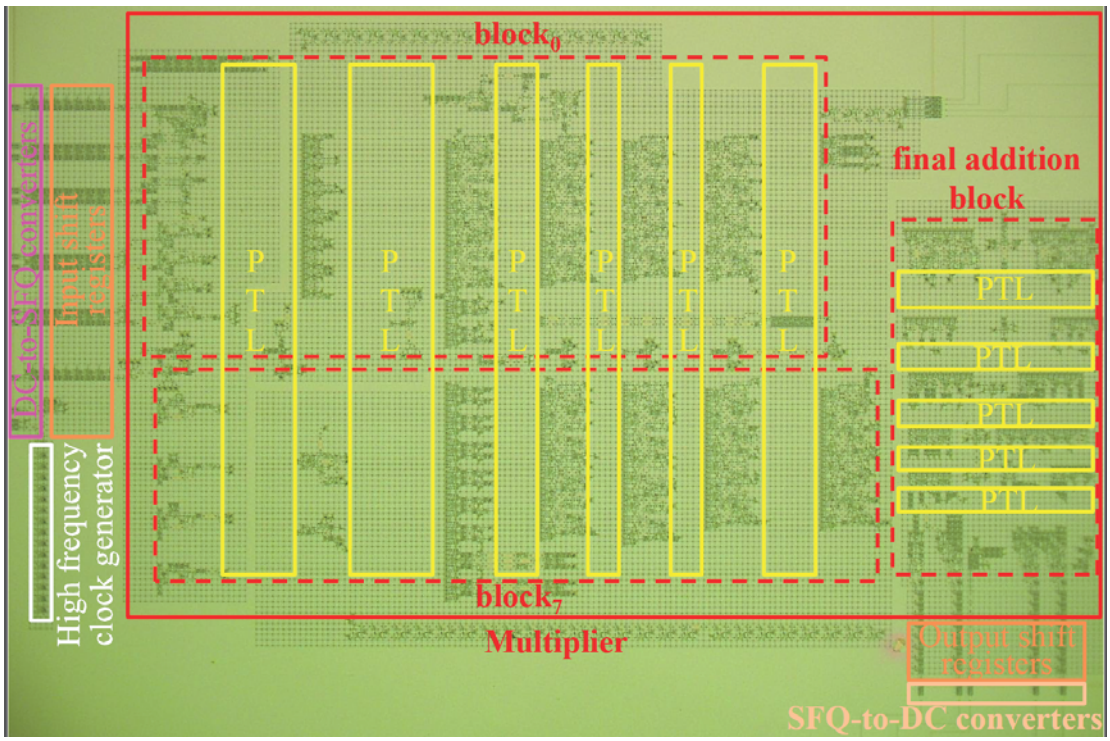


Figure 4.15. Micrograph of the test circuit of the 8×8-bit 4-bit bit-slice multiplier.

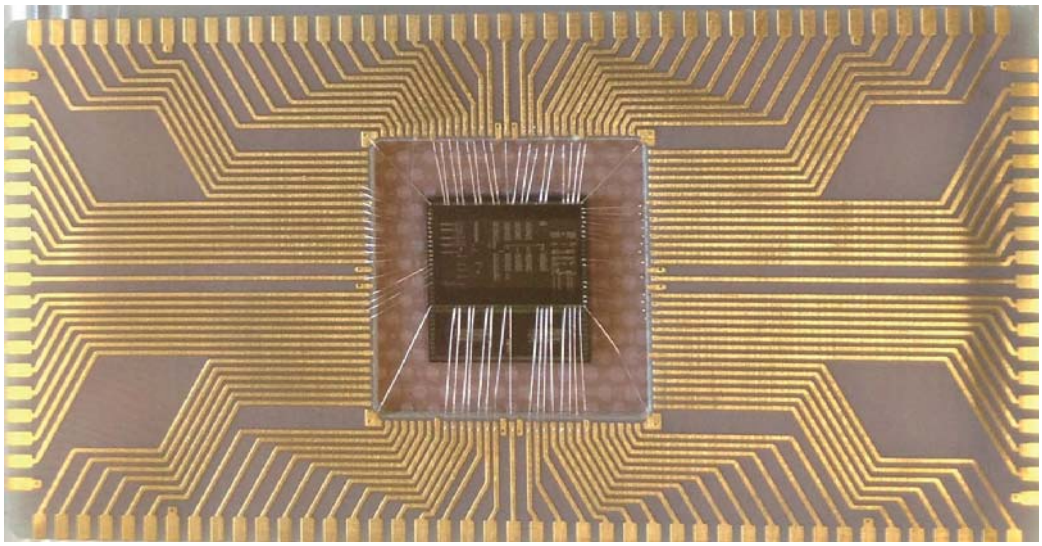


Figure 4.16. The chip of 8×8-bit 4-bit bit-slice multiplier mounted in a chip holder.

and SFQ-to-DC converters) consists of 21 stages and 11,488 JJs. It occupies the area of 5.3×2.6 mm^2. It has the bias current of 1,302 mA and the *colck delay* of 460 ps with the bias voltage of 2.5 mV. The latency is 20 ps/cycle × 25 cycles + 460 ps = 960 ps at 50 GHz.

The chip of the 4multiplier mounted in a chip holder is shown in Figure 4.16.

4.6 Summary

There are four challenges in the design of bit-slice multipliers:

(1) Make each feedback loop include no clocked gate.

(2) Minimize the delay of each feedback loop.

(3) Make timing design easier.

(4) Make precise timing design.

The following approaches for responding to the challenges are adopted:

(1) Algorithm level: Development of the algorithm for reducing the number of feedback loops without a clocked gate.

(2) Logic level: minimization of the delay of feedback loops, reduction of the number of stages with clocked gates.

(3) Layout level: Precise timing design by calculating the delay of wirings.

In this chapter, the following methods for responding to the challenges have been developed:

(1) Algorithm level: Development of a new systolic-like algorithm with unidirectional data signal flow, carry save addition, and Sklansky adder for final addition. There is only one feedback loop without a clocked gate in the whole multiplier.

(2) Logic level: The calculations with the start signal are executed before entering the feedback loop. CBs are used in full adders to reduce the number of stages.

(3) Layout level: The adjacent blocks are aligned in two stages off. The all clocked gates in each stage are aligned in a row. The wiring lengths are adjusted.

We have proposed a 32×32-bit 4-bit bit-slice RSFQ multiplier based on a new systolic-like algorithm. The physical layout of the multiplier has been designed and simulated. The simulation results show that the 32×32-bit 4-bit bit-slice multiplier achieves the target frequency of 50 GHz and a throughput of 3.125×10^9 multiplications per second.

Comparing with the parallel architecture, the bit-slice approach simplifies the circuit complexity and reduces the hardware cost. The bit-slice processing is a practical solution for a multiplication with longer operand length. Although we have let the length of a bit-slice be four in this paper, we can design a bit-slice multiplier with any slice length in the same way. When we design an $m \times m$-bit k-bit bit-slice multiplier, it will mainly consist of m/k blocks and the amount of hardware of each block is proportional to k^2. Therefore, the amount of hardware of the multiplier will be $O(km)$ instead of $O(m^2)$.

Chapter 5

Conclusion

CMOS circuit technology is facing unprecedented challenges of high power consumption and the process limitation. Superconducting rapid single-flux-quantum (RSFQ) circuit technology with high speed computation and low power consumption is expected to become a next-generation circuit technology. With the development of fabrication technology, it is becoming possible to develop practical 32-bit superconductor microprocessors with a system clock above 5 GHz in order to achieve comparable performance to a CMOS microprocessor operating at 2-3 GHz.

For the computation of 32-bit data, bit-serial processing cannot achieve a system clock above 2 GHz and parallel processing cannot be implemented on a couple of chips using today's fabrication technology. Bit-slice processing may be a solution for 32-bit superconductor microprocessors using today's fabrication technology.

The objective in this study is to develop methods for designing datapath circuits for superconductor bit-slice microprocessors. As the main components of superconductor bit-slice microprocessors, bit-slice ALU and bit-slice multiplier are studied.

To achieve comparable performance to a CMOS microprocessor operating at several gigahertzes, bit-slice processing should be performed with a clock frequency of several tens of gigahertz. Unlike parallel processing, the feedback loops are required for bit-slice processing for carry signal. In order to ensure the continuous calculation of sliced data, each feedback loop must include no clocked gate. Because the delay of feedback loops limits the clock frequency, each feedback loop must be minimized. On the other hand, the signal flow and interconnection of bit-slice processing are more complex than bit-serial processing. The timing error of wiring also limits the clock frequency. Thus, the circuits must be simplified and the delay of wirings needs to be precisely calculated.

This study indicates that the challenges of designing bit-slice datapath circuits can be handled using the approaches as follows:

(1) Algorithm level: Development of the algorithm for reducing the number of feedback loops without a clocked gate.

(2) Logic level: minimization of the delay of feedback loops, reduction of the number of stages with clocked gates.

(3) Layout level: Precise timing design by calculating the delay of wirings.

For a bit-slice ALU, the following methods for responding to the challenges have been developed:

(1) Algorithm level: Sklansky adder with only one feedback loop without a clocked gate is used.

(2) Logic level: The circuit for producing the carry from the most significant bit of a slice is duplicated. The calculations with several signals are executed before entering the feedback loop. CBs are used to implement the OR operations in the PG network in order to reduce pipeline stages.

(3) Layout level: The all clocked gates in each stage are aligned in a row. The wiring lengths between two adjacent stages are adjusted so that the data arrival time of all gates in each stage is approximately equal.

We fabricated and successfully demonstrated high-frequency operation of the 4-bit bit-slice ALU. The proposed ALU consists of 3,481 Josephson junctions with an area of 3.09×1.66 mm^2. Testing results showed that the ALU has a latency of 524 ps and a throughput of 6.25×10^9 32-bit operations per second at 50 GHz. The proposed ALU can be used for any 4n-bit processing.

We have designed and compared 2-/4-/8-/16-bit bit slice as well as bit-serial and parallel ALUs. The 4-bit bit-slice RSFQ ALU has the lowest latency and the highest energy efficiency for processing 32-bit data at 50 GHz.

For a bit-slice multiplier, the following methods for responding to the challenges have been developed:

(1) Algorithm level: Development of a new systolic-like algorithm with unidirectional data signal flow, carry save addition, and Sklansky adder for final addition. There is only one feedback loop without a clocked gate in the whole multiplier.

(2) Logic level: The calculations with the start signal are executed before entering the feedback loop. CBs are used in full adders to reduce the number of stages.

(3) Layout level: The adjacent blocks are aligned in two stages off. The all clocked gates in each stage are aligned in a row. The wiring lengths are adjusted.

We have proposed a 32×32-bit 4-bit bit-slice RSFQ multiplier based on a new systolic-like algorithm. The physical layout of the multiplier has been designed and simulated. The simulation results show that the 32×32-bit 4-bit bit-slice multiplier achieves the target frequency of 50 GHz and a throughput of 3.125×10^9 multiplications per second.

We hope that the approaches and the developed methods are useful for designing datapath circuits for RSFQ bit-slice microprocessors.

Bibliography

[1] [Online] Available: https://en.wikipedia.org/wiki/Intel_4004.

[2] [Online] Available: http://ark.intel.com/products/88199/Intel-Core-m7-6Y75-Processor-4M-Cache-up-to-3_10-GHz.

[3] [Online] Available: https://www.top500.org/lists/2016/06/.

[4] [Online] Available:http://nanohub.org/local/ipod/Mukhopadhyay_NANO501_Switching_Energy_in_CMOS_Logic(PDF).pdf

[5] [Online] Available: https://en.wikipedia.org/wiki/10_nanometer.

[6] 2015 International Technology Roadmap for Semiconductors (2015 ITRS). [Online]. Available: http://www.itrs2.net/

[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. Supercond.*, vol. 1, no. 1, pp. 3-28, Mar. 1991.

[8] A. Fujimaki, "Technology trends of superconductor digital devices," *Superconductivity Web21*, pp. 1, Oct. 2012.

[9] S. Nagasawa, Y. Hashimoto, H. Numata, S. Tahara, "A 380 ps, 9.5 mW Josephson 4-Kbit RAM operated at a high bit yield," *IEEE Trans. Appl. Supercond.*, vol. 5, no. 2, pp. 2447–2452, Jun. 1995.

[10] Ballistic missile defense organization, Technology applications report, 1994.

[11] M. Dorojevets, P. Bunyk, and D. Zinoviev, "FLUX chip: design of a 20-GHz 16-bit ultrapipelined RSFQ processor prototype based on 1.75-μm LTS technology," *IEEE Trans. Appl. Supercond.*, vol. 11, no. 1, pp. 326 –332, Mar. 2001.

[12] M. Tanaka, F. Matsuzaki, T. Kondo, N. Nakajima, Y. Yamanashi, A. Fujimaki, H. Hayakawa, N. Yoshikawa, H. Terai, and S. Yorozu, "A single-flux-quantum logic prototype microprocessor," in *Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International*, Feb. 2004, vol.1, pp. 298 – 529.

[13] Y. Yamanashi, M. Tanaka, A. Akimoto, H. Park, Y. Kamiya, N. Irie, N. Yoshikawa, A. Fujimaki, H. Terai, and Y. Hashimoto, "Design and implementation of a pipelined bit-serial SFQ microprocessor, CORE1*β*," *IEEE Trans. Appl. Supercond.*, vol. 17, no. 2, pp. 474 –477, Jun. 2007.

[14] A. Fujimaki, M. Tanaka, T. Yamada, Y. Yamanashi, H. Park, and N. Yoshikawa, "Bit-serial single flux quantum microprocessor CORE," *IEICE Trans. Electron.*, vol. E91-C, pp. 342–349, Mar. 2008

[15] M. Tanaka, Y. Yamanashi, N. Irie, H. J. Park, S. Iwasaki, K. Takagi, K. Taketomi, A. Fujimaki, N. Yoshikawa, H. Terai, and S. Yorozu, "Design and implementation of a pipelined 8 bit-serial single-flux-quantum microprocessor with cache memories," *Supercond. Sci. Technol.*, vol. 20, no. 11, pp. S305-S309, 2007.

[16] Y. Nobumori, T. Nishigai, K. Nakamiya, N. Yoshikawa, A. Fujimaki, H. Terai, and S.Yorozu, "Design and implementation of a fully asynchronous SFQ microprocessor: SCRAM2," *IEEE Trans. Appl. Supercond.*, vol. 17, no. 2, pp. 478-481, Jun. 2007.

[17] S. Nagasawa, K. Hinode, T. Satoh, M. Hidaka, H. Akaike, A. Fujimaki, N. Yoshikawa, K. Takagi, and N. Takagi, "Nb 9-layer fabrication process for superconducting large-scale SFQ circuits and its process evaluation," IEICE Trans. Electron., vol. E97-C, no. 3, pp. 132–140, Mar. 2014.

[18] A. Fujimaki, M. Tanaka, R. Kasagi, K. Takagi, M. Okada, Y. Hayakawa, K. Takata, H. Akaike, N. Yoshikawa, S. Nagasawa, K. Takagi, and N. Takagi, "Large-scale integrated circuit design based on a Nb nine-layer structure for reconfigurable data-path processors," *IEICE Trans. Electron.*, vol. E97-C, no. 3, pp. 157–165, Mar. 2014.

[19] K. Takagi, M. Tanaka, S. Iwasaki, R. Kasagi, I. Kataeva, S. Nagasawa, T. Satoh, H. Akaike, and A. Fujimaki, "SFQ propagation properties in passive transmission lines based on a 10-Nb-layer structure," *IEEE Trans. Appl. Supercond.*, vol. 19, no. 3, pp. 617-620, Jun. 2009.

[20] J. Y. Kim, S. Kim, and J. Kang, "Construction of an RSFQ 4-bit ALU with half adder cells," *IEEE Trans. Appl. Supercond.*, vol. 15, no. 2, pp. 308–311, Jun. 2005.

[21] T. Filippov, M. Dorojevets, A. Sahu, A. Kirichenko, C. Ayala, and O. Mukhanov, "8-bit asynchronous wave-pipelined RSFQ arithmetic-logic unit," *IEEE Trans. Appl. Supercond.*, vol. 21, no. 3, pp. 847-851, Jun. 2011.

[22] T. Filippov, A. Sahu, A. Kirichenko, I. Vernik, M. Dorojevets, C. Ayala, and O. Mukhanov, "20 GHz operation of an asynchronous wave-pipelined RSFQ arithmetic-logic unit," *Phys. Procedia*, vol. 36, pp. 59-65, 2012.

[23] M. Dorojevets, C. Ayala, N. Yoshikawa, A. Fujimaki, "8-bit asynchronous sparse-tree superconductor RSFQ arithmetic-logic unit with a rich set of operations," *IEEE Trans. Appl. Supercond.*, vol. 23, no. 3, Art. ID. 1700104, Jun. 2013.

[24] I. Kataeva, H. Engseth, and A. Kidiyarova-Shevchenko, "New design of an RSFQ parallel multiply accumulate unit," *Supercond. Sci. Technol.*, vol. 19, pp. S381–S386, Mar. 2006.

[25] M. Dorojevets, A. K. Kasperek, N. Yoshikawa, and A. Fujimaki, "20-GHz 8 × 8-bit Parallel Carry-Save Pipelined RSFQ Multiplier," *IEEE Trans. Appl. Supercond.*, vol. 23, no. 3, Art. ID. 1300104, Jun. 2013.

[26] K. Obata, M. Tanaka, Y. Tashiro, Y. Kamiya, N. Irie, K. Takagi, N. Takagi, A. Fujimaki, N. Yoshikawa, H. Terai, S. Yorozu, "Single-flux-quantum integer multiplier with systolic array structure," *Physica C*, vol. 445-448, pp. 1014-1019, Jul. 2006.

[27] X. Peng, Q. Xu, T. Kato, Y. Yamanashi, N. Yoshikawa, A. Fujimaki, N. Takagi, K. Takagi, and M. Hidaka, "High-speed demonstration of bit-serial floating-point adders and multipliers using single-flux-quantum circuits," *IEEE Trans. Appl. Supercond.*, vol. 25, no. 3, Art. ID. 1301106, Jun. 2015.

[28] Y. Hashimoto, "Studies on high speed system technology of superconducting SFQ," PhD dissertation, Nagoya University, 2009.

[29] Superconducting Technology Assessment (STA). National Security Agency Office of Corporate Assessments, Aug. 2005. [Online] Available: http://www.nitrd.gov/pubs/nsa/sta.pdf

[30] [Online] Available: http://www.physics.sunysb.edu/Physics/RSFQ/Lib/KL/and.html.

[31] K. Gaj, E.G. Friedman, and M.J. Feldman, "Timing of multi-gigahertz rapid single flux quantum digital circuits," *Journal of VLSI Signal Processing*, vol. 16, pp. 247–276, Jun. 1997.

[32] [Online] http://www.ieice-hbkb.org/files/09/09gun_02hen_02.pdf#page=2.

[33] K. Takagi, Y. Ito, S. Takeshima, M. Tanaka, N. Takagi, “Layout-driven skewed clock tree synthesis for superconducting SFQ circuits,” *IEICE Trans. Electronics*, E94-C, no. 3, pp. 288-295, Mar. 2011.

[34] M. Tanaka, K. Obata, Y. Ito, S. Takeshima, M. Sato, K. Takagi, N. Takagi, H. Akaike, A. Fujimaki, “Automated passive-transmission-line routing tool for single-flux- quantum circuits based on A* algorithm,” *IEICE Trans. Electronics.*, E93-C, no. 4, pp. 435-439, Apr . 2010.

[35] A. Kirichenko, O. Mukhanov, and A. Ryzhikh, “Advanced on-chip test technology for RSFQ circuits,” *IEEE Trans. Appl. Supercond.*, vol. 7, pp. 3438–3441, Jun. 1997.

[36] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 2011, Addison Wesley.

[37] N. Takagi and M. Tanaka, “comparisons of synchronous-clocking SFQ adders,” *IEICE Trans. Electron.*, vol. E93-C, no. 4, pp. 429–434, Apr. 2010.

[38] H. Park, Y. Yamanashi, N. Yoshikawa, M. Tanaka, and A. Fujimaki, “Design of fast digit-serial adders using SFQ logic circuits,” *IEICE Electronics Express*, vol. 6, no. 19, pp. 1408-1413, Oct. 2009.

List of publications

Journal

1. **Guang-Ming Tang**, Kazuyoshi Takagi, and Naofumi Takagi, "RSFQ 4-bit bit-slice integer multiplier", IEICE Transactions on Electronics, vol.E99-C, no.6, pp.697-702, Jun. 2016.
2. **Guang-Ming Tang**, Kensuke Takata, Masamitsu Tanaka, Akira Fujimaki, Kazuyoshi Takagi, and Naofumi Takagi, "4-bit bit-slice arithmetic logic unit for 32-bit RSFQ microprocessors", IEEE Transactions on Applied Superconductivity, vol.26, no.1, Art. ID.1300106, Jan. 2016.

International Conference with Peer Review

1. **Guang-Ming Tang**, Kazuyoshi Takagi, Naofumi Takagi, “Conceptual design of a 4-bit bit-slice 32-bit RSFQ microprocessor", Proc. of 2016 Applied Superconductivity Conference (ASC’16), 4EOr2B-03, Denver, U. S., Sep. 4th − 9th, 2016. (To appear)
2. **Guang-Ming Tang**, Kazuyoshi Takagi, Naofumi Takagi, "A 4-bit bit-slice multiplier for a 32-bit RSFQ Microprocessor", Proc. of the 15th International Superconductive Electronics Conference (ISEC 2015), DS-P05, Nagoya, Japan, July 6th − 9th, 2015.

Workshop without Peer Review

1. **Guang-Ming Tang**, Kazuyoshi Takagi, Naofumi Takagi, "Comparison of Bit-Slice Arithmetic Logic Units for 32-bit RSFQ Microprocessors", Proc. of the 7th Superconducting SFQ VLSI Workshop (SSV2014), pp.39-44, Kobe, Japan, December 1st − 2nd, 2014.

2. Kensuke Takata, Masamitsu Tanaka, Akira Fujimaki, **Guang-Ming Tang**, Kazuyoshi Takagi, Naofumi Takagi, "Demonstration of 4-bit-Parallel Bit-Slice ALU", Proc. of the 7th Superconducting SFQ VLSI Workshop (SSV2014), pp.84-89, Kobe, Japan, December 1st − 2nd, 2014.
3. **Guang-Ming Tang**, Kazuyoshi Takagi, Naofumi Takagi, "Logic Design of a 4-bit Bit-Slice Arithmetic Logic Unit for 32-bit RSFQ Microprocessors", Proc. of Superconducting SFQ VLSI Workshop for Young Scientists (SSV2014-YS), pp.11-14, Nagoya, Japan, March 5th − 7th, 2014.
4. Kensuke Takata, Yuhi Hayakawa, Masamitsu Tanaka, Akira Fujimaki, **Guang-Ming Tang**, Kazuyoshi Takagi, Naofumi Takagi, "Implementation of 32-bit ALU Based on 4-bit-Parallel Bit-Slice Processing", Proc. of Superconducting SFQ VLSI Workshop for Young Scientists (SSV2014-YS), pp.57-60, Nagoya, Japan, March 5th − 7th, 2014.
5. Kensuke Takata, Masamitsu Tanaka, Akira Fujimaki, **Guang-Ming Tang**, Kazuyoshi Takagi, Naofumi Takagi, "Design and Evaluation of the 4-Bit Parallel Bit-Slice ALU", Technical report of IEICE. SCE 114(147), pp.7-12, Tokyo, Japan, July 23rd, 2014. (in Japanese)