Designs of Component Circuits for Stochastic Computing Using Rapid Single Flux Quantum Circuits

Nobutaka Kito  Yurie Koketsu  Kazuyoshi Takagi

School of Engineering  Graduate School of Informatics
Chukyo Univ.  Kyoto Univ.
Toyota 470-0393, Japan  Kyoto 606-8501, Japan
nkito@sist.chukyo-u.ac.jp  ktakagi@i.kyoto-u.ac.jp

Abstract—Designs of component circuits for stochastic computing using rapid single flux quantum (RSFQ) circuits are presented. A design of a stochastic number generator and a design of a stochastic-to-binary converter are proposed. In those designs, voltage pulses of RSFQ circuits are used for representing ones in a bit-stream of a signal in stochastic computing. By using special gates of RSFQ circuits, they are implemented as simple circuits. A layout for 4-bit multiplication using those proposed designs is shown. By comparison with circuits designed by various methods, it is suggested that low-precision arithmetic circuits can be implemented in smaller area by using stochastic computing.

I. Introduction

Superconducting computing devices have been considered as potentially alternative devices of mainstream semiconductor computing devices, for supercomputers [1]. The superconducting rapid single flux quantum (RSFQ) circuit technology [2] is a promising digital circuit technology for high-speed and low-power operations. RSFQ circuits work at up to 100 GHz [3, 4, 5]. In RSFQ circuits, voltage pulses are used for representing logic values.

The voltage pulses of RSFQ circuits are utilized for realizing stochastic computing [6, 7] in this paper. It is known that complex circuits such as an LDPC decoder can be implemented in small area by using stochastic computing [7]. In circuits using stochastic computing, ratio of ones in an observed bit-stream of a signal represents its signal value, and counting of ones in a bit-stream is necessary to observe its value. The voltage pulses of RSFQ circuits are countable, and they are suitable for representing ones in a bit-stream. Therefore, the voltage pulses are used to form a bit-stream in stochastic computing to represent a value.

In this paper, designs of basic modules for stochastic computing using RSFQ circuits, i.e., a stochastic number generator and a stochastic-to-binary converter, are proposed. Those modules utilize the pulse logic of RSFQ circuits. The proposed modules utilize special gates of RSFQ circuits, such as a non-destructive readout (NDRO) gate and a confluence buffer (CB) which converges pulses of its inputs into its output. For example, CBs are used to calculate logical OR in those modules for saving circuit area.

Circuits for multiplication and addition can be implemented in small area by using stochastic computing though long observation, i.e., counting of ones on the signal line, is necessary to determine a signal value. RSFQ circuits achieve high frequency operation, and RSFQ circuits using stochastic computing works with small latency. Stochastic computing will be useful for designing small low-precision RSFQ arithmetic circuits. In this paper, as an example, an RSFQ layout of multiplication using stochastic computing is designed, and it is showed that low-precision arithmetic circuits using stochastic computing can be implemented in smaller than circuits designed by traditional or previously proposed methods though error appears in some cases.

Systolic RSFQ arithmetic circuits, such as [8] and [9], have been studied. Because integration density of current fabrication technologies for RSFQ circuits are not high and circuit area of a systolic circuit is smaller than the area of its parallel implementation, systolic circuits have been used in RSFQ circuits. However, systolic circuits are difficult in designing their pipeline stages, and alternative methods for designing area efficient arithmetic circuits are desirable. Therefore, RSFQ stochastic circuits will be useful. For an arithmetic circuit of narrow bit-width, a ROM table is sometimes used in CMOS design. In RSFQ circuits, ROMs or RAMs consume large area. The state of the art design of a 2-bit ROM cell in [10] occupies $60 \times 70 \mu m^2$. For realizing a 4-bit two-operand operation, memory cells for the operation consume at least 2.1 mm$^2$. Because decoder circuits and routing for an RSFQ memory consume large area, ROM-based design is not suitable for arithmetic circuits.

Previously, RSFQ neurosystems using stochastic computing were researched [11, 12, 13]. Those researches aimed to realize RSFQ-based neural network circuits rather than arithmetic circuits. Designs for realizing a neuron of a neural network, such as an up/down counter.
and a sigmoid function generator, were proposed. Especially, in [11], two designs which can work as stochastic number generators were shown. One of the designs was a clock synchronous circuit using pipelined comparators. In this paper, a smaller design than that is proposed. The other design is an asynchronous circuit using T flip flops (TFFs) with smaller circuit size. In an RSFQ circuit, synchronization of a large asynchronous block with a clock signal is not easy in general because of variability of delay in each active device. Though the design proposed in this paper is not a clock synchronous circuit, asynchronous parts of the proposed design are smaller than that in the TFF-based design. As a result, we achieved both a small area and stable functionality of the circuit.

A layout for 4-bit multiplication containing the proposed component circuits was evaluated. It was designed with the cell library developed for AIST advanced process (ADP2) [3]. The functionality of the designed circuit was evaluated by simulation. It can work at high clock frequency up to 40 GHz. By comparison with a ROM-based design and a systolic design, it was suggested that low-precision arithmetic circuits can be implemented in smaller area by using stochastic computing.

This paper is organized as follows. In the next section, a brief review of RSFQ circuits and stochastic computing is shown. In Section III, a design of a stochastic number generator and a design of a stochastic-to-binary converter are proposed. In Section IV, experimental results are shown. In Section V, this paper is concluded.

II. Preliminaries

A. RSFQ Circuit

In an RSFQ circuit, voltage pulses for representing signals are transmitted on signal lines. Each basic gate has a clock input terminal and works synchronized with clock pulses. As an example, the symbol and the behavior of an RSFQ AND gate are shown in Figs. 1(a) and (b), respectively. The value of an input signal is evaluated with clock pulses as shown in Fig. 1 (b). When a pulse arrives at a data input of a gate during an interval between adjacent clock pulses, the input value corresponding to the interval is “1”. If no pulse arrives during the interval, the input value is “0”. The output of a gate is synchronized with the clock pulse.

In addition to the traditional logic gates, such as AND, OR, and XOR, several special gates exist as shown in Fig. 2. In the figure, the symbol of each gate and its pulse-transferring finite state machine (PTFSM) [14] are shown. A non-destructive read-out (NDRO) gate has two internal states, i.e. ST0 and ST1, as shown in Fig. 2(a). It outputs a pulse at dout only when its internal state is ST1 and a pulse arrives at its clk terminal. A resettable DFF (RDFF) gate also has two states as shown in Fig. 2(b). Its internal state returns to ST0 when a pulse arrives at either clk or rst terminal. A T1 gate in Fig. 2(c) works like a full adder, and is often used for counting pulses. When internal state of a T1 gate is ST1, it outputs a pulse at carry or sum terminal once a pulse arrives at din or clk terminal, respectively. A resettable TFF (RTFF) gate in Fig. 2(d) works like the carry part of T1 gate.

Splitters are used to feed a signal to plural gates, and confluence buffers (CBs) are used to merge two input signals into an output signal. In Fig. 3, behaviors of a splitter and a CB are shown. As shown in the figure, a splitter and a CB are depicted by symbols • and ○ in schematic, respectively. As in the figure, an arrow is used to clarify direction of a signal. For correct operation of a designed circuit, delay elements are inserted on signal lines to keep up order of pulse arrivals to the expected order at each gate.

B. Stochastic Computing

Stochastic computing uses a bit-stream to represent a number, and processes bit-streams for calculation among
numbers. In stochastic computing, a number is represented by the ratio of ones in a bit-stream. An example is shown in Fig. 4 (a). In this paper, unipolar encoding is considered, in other words, numbers between 0 to 1 are considered. In the figure, we observe four ones and four zeros on signal $s_1$, and its value is considered as $\frac{4}{8}$. The probability to output one at the AND gate can be represented by $p(s_1)p(s_2)$ where $p(x)$ represents probability of appearance of one at signal $x$ when the two signals have no correlation. Therefore, multiplication can be realized by an AND gate in stochastic computing. As well as multiplication, scaled addition can be realized by a 2:1 multiplexer.

In this paper, we represent bit-streams of signals along with a clock signal for synchronization purposes as shown in Fig. 4(b). Between adjacent clock pulses, we permit at most one data pulse.

III. COMPONENT CIRCUITS FOR STOCHASTIC COMPUTING USING RSFQ CIRCUITS

Conventional digital circuits mainly treat values in binary form. For stochastic computing, we need a stochastic number generator which converts a binary number to a stochastic bit-stream, and a stochastic-to-binary converter which converts a stochastic bit-stream to a conventional binary number. We show designs of those two component circuits utilizing pulse logic and special gates of RSFQ circuits.

A. RSFQ Stochastic Number Generator

We propose an RSFQ stochastic number generator. Usually, a stochastic number generator is designed with a random number generator and a comparator which compares a random number and the binary number to be converted. In a comparator, a path from the inputs to the outputs is not short. For an RSFQ comparator, we will need some DFFs to maintain pipeline stages for working because each basic logic gate works as if it contains a DFF in it. Therefore, designs of that type will consumes large circuit area.

We design an RSFQ-based generator based on the design shown in Fig. 5 [15]. It generates a stochastic bit-stream from a 4-bit binary number without a comparator. It consists of 4-bit linear feedback shift register (LFSR) as a random number generator and a weighted binary generator. The characteristic polynomial of the LFSR is $x^4 + x^3 + 1$, and the period of the 4-bit LFSR is 15 ($= 2^4 - 1$). In a period of 15 cycles, 8 pulses appear at each of $s_3, s_2, s_1,$ and $s_0$. By each AND gate in the first row, logical AND of the value of the corresponding position of the LFSR output and the inverted
values of its lower positions is calculated. We obtain the weighted bits $w_3, \ldots, w_0$. In a period of 15 cycles, 8, 4, 2, and 1 pulses appear at $w_3$, $w_2$, $w_1$, and $w_0$ terminals, respectively. We obtain the bit-sequence for binary number $x_3 x_2 x_1 x_0$ by the second row of AND gates and the final OR gate. Note that a modification is necessary to insert the all zero pattern in the clock. We set the internal states of the NDROs in the second row of AND gates and CBs for logical OR of the outputs instead of clocked-OR gates. The bit-sequence at the output of the generator corresponds to binary number $[0.x_3 x_2 x_1 x_0]_2$. We insert a circuit for the clock input of the generator for eliminating a clock pulse and the arrivals of other pulses, then pulses may arrive at $rst$ terminal. When at least one pulse exists in the lower LFSR positions of an NDRO, internal state of the NDRO is reset. After the arrival of $set$ pulse and the arrivals of $rst$ pulses, a pulse arrives at $clk$ terminal. The NDROs output pulses according to their internal states, and we obtain the weighted bits $w_3, \ldots, w_0$.

In the proposed design, AND gates in the original design are replaced by simple NDROs, and inverters for inputs of AND gates are replaced by CBs. By those replacements considering the feature of pulse logic, we can design the generator with a small number of gates.

We describe the operation of the upper part consisting of DFFs and RTFFs. When we feed a pulse for $rst$ terminal, the rightmost DFF receives a pulse. When 8 pulses arrive at clock terminal of the converter, a pulse is fed to the $clk$ terminal of the rightmost DFF, and it outputs a pulse toward the input of its left adjacent DFF. In the same way, when 15 ($=8+4+2+1$) pulses arrive at the clock terminal, the leftmost DFF outputs a pulse. The pulse is fed to the lower part of the converter. By the 16th pulse for the clock terminal, the rightmost RDFF generates a pulse and the rightmost DFF receives a pulse row by circuit inputs $x_3, \ldots, x_0$. Each NDRO outputs a pulse according to its internal state when a pulse is fed to its $clk$ terminal. At most one of the four NDROs outputs a pulse at their $dout$ terminals, and we use CBs for logical OR of the outputs instead of clocked-OR gates. The bit-sequence at the output of the generator corresponds to binary number $[0.x_3 x_2 x_1 x_0]_2$. We set the internal states of the T1s at circuit outputs $z_3, \ldots, z_0$, and the output binary value is $[0.z_3 z_2 z_1 z_0]_2$.

We describe the operation of the upper part consisting of DFFs and RTFFs. When we feed a pulse for $rst$ terminal, the rightmost DFF receives a pulse. When 8 pulses arrive at clock terminal of the converter, a pulse is fed to the $clk$ terminal of the rightmost DFF, and it outputs a pulse toward the input of its left adjacent DFF. In the same way, when 15 ($=8+4+2+1$) pulses arrive at the clock terminal, the leftmost DFF outputs a pulse. The pulse is fed to the lower part of the converter. By the 16th pulse for the clock terminal, the rightmost RDFF generates a pulse and the rightmost DFF receives a pulse.
though the CB. Therefore, the converter generates first pulse after 15 clock pulses, and then generates a pulse every 16 clock pulses. We can extend the converter for wider bit-width easily.

IV. Experimental Results

We show a layout for 4-bit multiplication. For the layout, we used the proposed stochastic number generator and the proposed stochastic-to-binary converter. We used Cadence Virtuoso and the cell library designed for AIST advanced process (ADP2) [3] for layout design. In the design flow using the cell library, a layout is composed by tiling cells in the schematic editor. We can easily convert a designed layout in schematic editor to a physical layout for fabrication.

We show the layout designed with schematic editor in Fig. 8. Its circuit area is 1.61 mm$^2$ (1.05 $\times$ 1.53 mm$^2$), and the number of Josephson junctions (JJs) in it is 1,820. It contains two stochastic number generators, a pulse elimination circuit, a stochastic-to-binary converter, and an AND gate for multiplication. The pulse elimination circuit eliminates a pulse in every 16 input pulses for clock inputs of generators, and the period of the generators with it is 16. The area of the AND gate for multiplication is 60 $\times$ 60 $\mu$m$^2$, and other circuits consume most of the layout area. By the logic level simulation considering delay of gates using Cadence Verilog-XL, we confirmed the functionality of the circuit. It was estimated that the circuit works at up to 40 GHz.

As we described in Section I, we can obtain a 4-bit multiplier with a 1k-bit (=2$^{(4+4)}$ $\times$ 4-bit) ROM. In [10], a memory cell of a mask ROM and a design of an 8-bit $\times$ 6 words (48 bits) ROM were shown. Though the target process technology in [10] is older than that considered in this paper, the 48-bit ROM consumes about 1 mm$^2$. Therefore, a multiplier realized with a ROM will be larger than the proposed design.

A systolic design of a multiplier was proposed in [8]. In [8], integer multiplication is carried out using systolic processing elements (PEs). Each PE consumes 639 JJs, and 2,556 JJs are necessary for a 4-bit multiplier. Therefore, the number of JJs in the stochastic realization is smaller than that in the systolic design.

An RSFQ 4-bit bit-slice 8 $\times$ 8-bit integer multiplier was proposed in [16]. It uses two 4-bit slices for 8 $\times$ 8-bit integer multiplication. It was composed of 11,488 JJs, and its layout area was 13.8 mm$^2$ (5.3 $\times$ 2.6 mm$^2$) with the same fabrication technology. Though the previous bit-slice multiplier computes a multiplication every four cycles and works at up to higher frequency (62.5 GHz), the designed layout with using the proposed modules carries out multiplication with very small area. For signal processing circuits which tolerate small error, stochastic computing will be useful for saving circuit area.

Fig. 8. Layout of an RSFQ circuit for 4-bit multiplication using stochastic computing.

V. Conclusion

We proposed key component circuits for stochastic computing using RSFQ circuits. We showed a design of stochastic number generator and a design of a stochastic to binary converter. In those designs, voltage pulses of RSFQ circuits are used for representing bit-stream of a signal in stochastic computing. By using special gates of RSFQ circuits, we realized them by small circuits. The layout for multiplication using those designs was shown, and it can work at high-frequency up to 40 GHz. By comparison with circuits designed by various methods, it is suggested that low-precision arithmetic circuits can be implemented in smaller area by using stochastic computing.

ACKNOWLEDGEMENTS

This work has been supported by VLSI Design and Education Center (VDEC), The University of Tokyo with the collaboration with Cadence Corporation.
REFERENCES


