BRIEF PAPER

Rapid Single-Flux-Quantum Truncated Multiplier Based on Bit-Level Processing

Nobutaka KITO[†a)], ***Member***, **Ryota ODAKA**[†], ***Nonmember***, ***and*** **Kazuyoshi TAKAGI**[††], ***Member***

SUMMARY A rapid single-flux-quantum (RSFQ) truncated multiplier based on bit-level processing is proposed. In the multiplier, two operands are transformed to two serialized patterns of bits (pulses), and the multiplication is carried out by processing those bits. The result is obtained by counting bits. By calculating in bit-level, the proposed multiplier can be implemented in small area. The gate level design of the multiplier is shown. The layout of the 4-bit multiplier was also designed.

key words: *rapid single flux quantum circuit, truncated multiplier, pulse logic*

1. Introduction

Superconducting computing devices have been considered as potentially alternative devices of mainstream semiconductor computing devices [1]. The superconducting rapid single-flux-quantum (RSFQ) circuit technology [2] is a promising digital circuit technology for high-speed and low-power operations.

In RSFQ logic circuit design, bit-serial or bit-slice processing has been used for arithmetic circuits than parallel processing which consumes larger circuit area. For example, a bit-serial adder and a bit-serial multiplier have been proposed [3], [4]. Designing layouts of large RSFQ logic circuits is a hard task because timing design of large circuits is tough. Simple and compact designs of arithmetic circuits are desired especially for multipliers which consume large area.

In this brief, we propose an RSFQ truncated multiplier based on bit-level processing. Generally, truncated multipliers, which discard lower part of partial product bits of a complete multiplier to save circuit area, are realized as parallel processing circuits. The proposed truncated multiplier processes in bit-level to realize small circuit area. In the multiplier, each operand is transformed to a serialized pattern of bits (pulses). The multiplication is done for those bits on two lines, and the multiplication result is obtained by counting bits of the same weight for $2^n - 1$ cycles in n-bit multiplication.

In bit-serial processing, each bit in n-bit operands fed serially has its corresponding weight. On the other hand, in the proposed multiplier, n-bit operands fed in parallel are converted into $(2^n - 1)$-bit bit-patterns and each bit in the patterns has the same weight. The proposed multiplier takes longer time for processing than bit-serial multipliers, however, the multiplier is simple and can be realized in compact area.

The proposed truncated multiplier is suitable for applications which tolerate small error. Recently, hardware accelerators for neural network processing such as [5] attract attention. It is well known that inference using neural networks can be carried out in low precision. 8-bit data type has been used in the accelerator in [5], and support for 4-bit data type has been added in NVIDIA Turing architecture GPUs [6] and AMD Vega architecture GPUs. The proposed multiplier will be suitable for such applications.

We designed a 4-bit layout of the proposed multiplier with the cell library developed for AIST advanced process (ADP2) [7]. Its functionality was evaluated by logic simulation. Its maximum absolute error in its multiplication result was also evaluated.

2. Preliminaries

2.1 Truncated Multiplication

We consider n-bit unsigned truncated multiplication of multiplicand X: $[0.x_1 \cdots x_n]_2$ and multiplier Y: $[0.y_1 \cdots y_n]_2$. We let the resultant product be Z: $[0.z_1 \cdots z_n]_2$. X, Y, and Z are fixed-point numbers $(0(= [0.0 \cdots 0]_2) \le X, Y, Z \le 1 - 2^{-n}(= [0.1 \cdots 1]_2))$, and each of x_i, y_i, and z_i is either 0 or 1.

In truncated multiplication, lower part of partial product bits is discarded. We show partial product bits in Fig. 1. The upper bits enclosed by the dashed lines, whose weights are larger than 2^{-n-1}, are summed up. The middle bits with weight 2^{-n-1} are used for compensating the result, and the

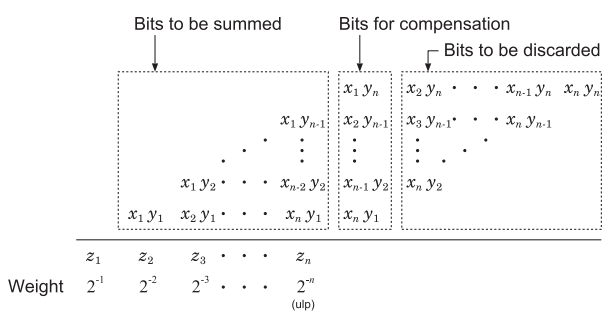

Fig. 1 Truncation of partial product bits in an multiplier.

Manuscript received August 24, 2018.
Manuscript revised November 24, 2018.

[†]The authors are with the School of Engineering, Chukyo University, Toyota-shi, 470–0393 Japan.
[††]The author is with the Graduate School of Informatics, Kyoto University, Kyoto-shi, 606–8501 Japan.
a) E-mail: nkito@sist.chukyo-u.ac.jp

DOI: 10.1587/transele.2018ECS6014

Copyright © 2019 The Institute of Electronics, Information and Communication Engineers

lower bits enclosed by dashed lines are discarded. As shown in the figure, the unit in the last place (*ulp*) of the result is 2^{-n}.

The result of the truncated multiplication is represented as follows:

$$Z = \sum_{i+j \le n} x_j y_i 2^{-(i+j)} + f(x_1 y_n, \ldots, x_n y_1) \cdot 2^{-n} \quad (1)$$

where f is an error-compensation function.

In this brief, $f(b_1, \ldots, b_n) = (b_1 + \cdots + b_n)$ is considered as the compensation function. The compensation using the function is equivalent to rounding each partial product $P_i = X \cdot y_i \cdot 2^{-i}$ to its nearest value.

2.2 RSFQ Circuits

In RSFQ circuits, voltage pulses are used to represent logic values. Each basic logic gate, such as AND, OR, and XOR, has a clock input terminal as shown in Fig. 2 (a) and works synchronized with clock pulses. 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" as shown in Fig. 2 (b). If no pulse arrives during the interval, the input value is "0". It is prohibited to feed plural pulses for a data input of a basic logic gate during an interval. The output of a gate is synchronized with the clock pulse.

In addition to basic logic gates, several special gates exist as shown in Fig. 3. In the figure, the symbol and the pulse-transferring finite state machine (PTFSMs) [8] of each gate are shown. The non-destructive read-out (NDRO) gate has two internal states, i.e., $ST0$ and $ST1$, as shown in Fig. 3 (a). It outputs a pulse at *dout* only when its internal state is $ST1$ and a pulse arrives at its *clk* terminal. The T1 gate in Fig. 3 (b) works like a counter of 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. The confluence buffer (CB) in Fig. 3 (c) merges pulses on its two inputs into its output.

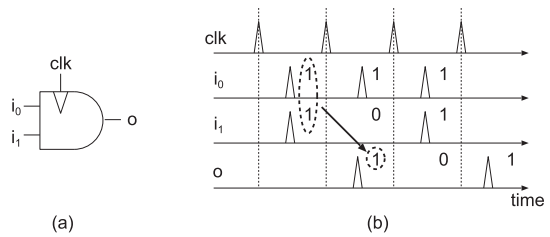

Fig. 2 RSFQ AND gate (a) and its behavior (b).

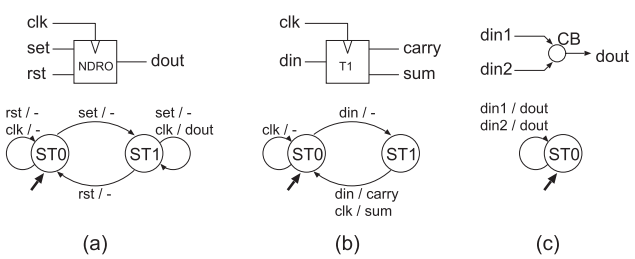

Fig. 3 Special gates in RSFQ circuits and their behaviors (Non-destructive read-out (NDRO) gate (a), T1 gate (b), and confluence buffer (CB) (c)).

3. RSFQ Truncated Multiplier Based on Bit-Level Processing

3.1 Structure

We propose an RSFQ truncated multiplier based on bit-level processing. We show its structure in Fig. 4. It consists of a pattern generator, two bit generators, an AND gate, and a pulse counter.

The pattern generator has n-bit pattern output S: $(s_1, s_2, \cdots, s_n)$. It outputs one of $2^n - 1$ patterns except the all-0 pattern (or 2^n patterns including the all-0 pattern) to S exhaustively without overlap in each clock cycle of a period of $2^n - 1$ (or 2^n) cycles. We let the number of clock cycles in a period be T. T is $2^n - 1$ or 2^n.

The structure of the bit generator is shown in Fig. 5. Its inputs are n-bit pattern R: $(r_1, r_2, \cdots, r_n)$, n-bit operand Q: $(q_1, q_2, \cdots, q_n)$, one-bit clock input *clk*, and one-bit reset input *rst*. Its output is one-bit signal b. An operand of the multiplier is fed to Q. b is calculated according to the operand value and an input pattern of R fed from the pattern generator. The output of the pattern generator S is connected to the two bit generators differently as shown in Fig. 4. s_i is connected to r_i of the bit generator for Y, and is connected to r_{n+1-i} of the bit generator for X. The input operands X and Y are treated as n-bit patterns $(x_1, x_2, \cdots, x_n)$ and $(y_1, y_2, \cdots, y_n)$, respectively. x_i is connected to q_i of the bit generator for X, and y_i is connected to q_i of the bit generator for Y. Once operand values of the multiplier are fed to the bit generators, those values are latched in the generators. *rst* of the generator is used for resetting the latches at the beginning of new calculation.

Each bit generator converts an operand into a serialized pattern of bits (pulses). The AND gate computes logical AND of bits on two lines. Weight of each bit (pulse) from the gate is 1 *ulp*. The pulse counter counts up pulses from the gate for T cycles, and outputs the counted result every T cycles. The counted result is the multiplication result.

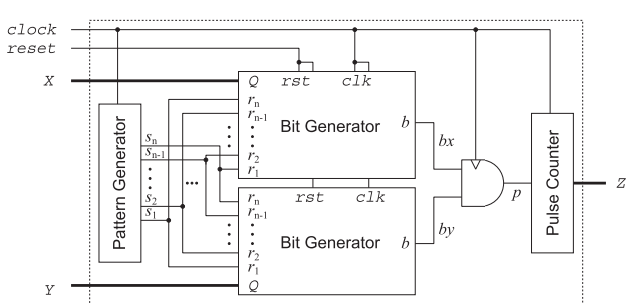

Fig. 4 Truncated multiplier based on bit-level processing.

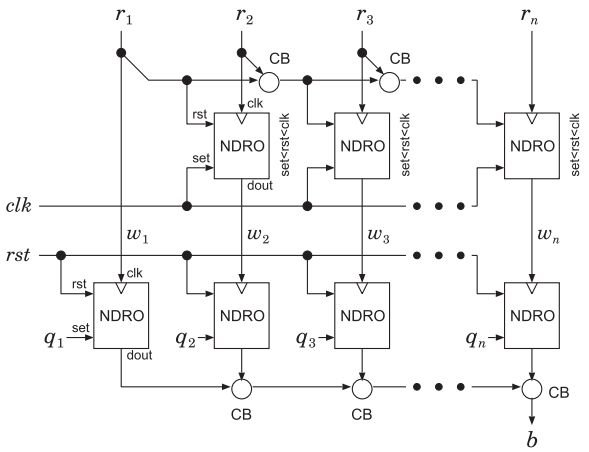

Fig. 5 Structure of the bit generator.

3.2 Calculation

We show the calculation of the multiplier. At first, the function of the bit generator is shown. Then, the calculation of the generators and the AND gate is shown.

In the bit generator of Fig. 5, NDRO gates in the first row are used to generate part of w_h signals ($1 < h \leq n$). For those NDRO gates, the orders of pulse arrivals are represented by inequalities [9]. Pulses distributed from *clk* terminal of the generator are fed into *set* terminals of those NDROs to set their internal state at first. Then, the NDRO for w_h receives $(r_1 + \cdots + r_{h-1})$ pulses at its *rst* terminal through CBs. Finally, the NDRO for w_h may receive a pulse fed to r_h at its *clk* terminal, and it outputs a pulse according to its internal state. The value of signal w_h is described as follows:

$$w_h = r_h \wedge \overline{r_{h-1}} \wedge \cdots \wedge \overline{r_1} = r_h \wedge \left(\bigwedge_{k<h} \overline{r_k} \right).$$

Note that, among all w_h signals, at most one signal takes "1", and the others are "0". In a period of T cycles, w_h takes "1" in $2^{(n-h)}$ cycles ($1 \leq h \leq n$).

The internal states of the NDROs in the second row of the bit generator are set by operand Q beforehand. Each NDRO in the second row outputs a pulse (bit) according to its internal state when it receives a pulse at its *clk* terminal. Among those NDROs, at most one NDRO outputs a pulse at each clock cycle. CBs are used for calculating logical OR of the outputs instead of OR gates to save circuit area. The output of the generator is represented as follow:

$$b = \bigvee_{h \leq n} \left(q_h \wedge r_h \wedge \left(\bigwedge_{k<h} \overline{r_k} \right) \right).$$

b takes "1" in $\sum_{h \leq n}(q_h \cdot 2^{(n-h)})$ cycles in a period of T cycles. For example, if $q_1 = \cdots = q_n = 1$, b takes "1" in $2^n - 1$ $(=q_1 \cdot 2^{n-1} + \cdots + q_n \cdot 2^0)$ cycles in a period of T cycles. By the bit generators, X and Y are converted to bit-patterns and the number of "1" in the bit-patterns corresponds to magnitude of them. Because the bit generator for X and the bit generator for Y are connected to the pattern generator differently, outputs of the bit generators are calculated differently.

The AND gate in Fig. 4 receives outputs of the two bit generators. The output of the AND gate p $(= by \wedge bx)$ is represented as follows:

$$\begin{aligned} p &= \left(\bigvee_{i \leq n} \left(y_i \wedge s_i \wedge \left(\bigwedge_{k<i} \overline{s_k} \right) \right) \right) \\ &\quad \wedge \left(\bigvee_{j \leq n} \left(x_j \wedge s_{n+1-j} \wedge \left(\bigwedge_{k<j} \overline{s_{n+1-k}} \right) \right) \right) \\ &= \bigvee_{i+j \leq n+1} \Bigg\{ x_j \wedge y_i \wedge \\ &\quad \left(\bigwedge_{k<i} \overline{s_k} \right) \wedge s_i \wedge s_{n+1-j} \wedge \left(\bigwedge_{n+1-j<k} \overline{s_k} \right) \Bigg\}. \end{aligned}$$

The above formula indicates that if $x_j = y_i = 1$ and $i + j \leq n + 1$, the AND gate outputs "1" when the output of the pattern generator S is as follows:

$$S_{j,i} = \begin{cases} (\underbrace{0, \cdots, 0}_{i-1}, 1, \underbrace{*, \cdots, *}_{n-(i+j)}, 1, \underbrace{0, \cdots, 0}_{j-1}) & (i + j \leq n) \\ (\underbrace{0, \cdots, 0}_{i-1}, 1, \underbrace{0, \cdots, 0}_{j-1}) & (i + j = n + 1) \end{cases}$$

where $*$ denotes "don't care". Among different pairs of j and i, $S_{j,i}$ does not overlap each other.

The pattern generator feeds all n-bit patterns other than the all-0 pattern. If $x_j = y_i = 1$ and $i + j \leq n$, the AND gate outputs at least $2^{n-(i+j)}$ pulses in a period of T cycles because there are $n - (i + j)$ bits of don't cares in $S_{j,i}$. Thus, the number of pulses the AND gate outputs in a period of T cycles is described as follows:

$$\sum_{i+j \leq n} x_j y_i 2^{n-(i+j)} + \sum_{i+j=n+1} x_j y_i$$

because any pair of different $S_{j,i}$ has no overlap. The former term corresponds to the former term in Formula (1), and the latter one corresponds to the compensation function. Therefore, the counted result of the pulse counter corresponds to the result of the truncated multiplication.

To show the operation of the truncated multiplier visually, we consider a 4-bit design as an example. The Karnaugh maps of the bit generators for bx and by with respect to the output of the pattern generator S are shown in Figs. 6 (a) and (b), respectively. In the maps, the operand value is transformed as the number of ones. 2^{4-i} grids corresponding to x_i or y_i are rounded by broken lines.

Figure 6 (c) is the Karnaugh map of the output of the AND gate when $X = [0.1111]_2$ and $Y = [0.1000]_2$. In the figure, the number of grids covered by vertical (orange) broken lines corresponding to $X = [0.1111]_2$ is 15, and those grids are divided by the horizontal (blue) broken lines corresponding to $Y = [0.1000]_2$. The pulse counter receives 8

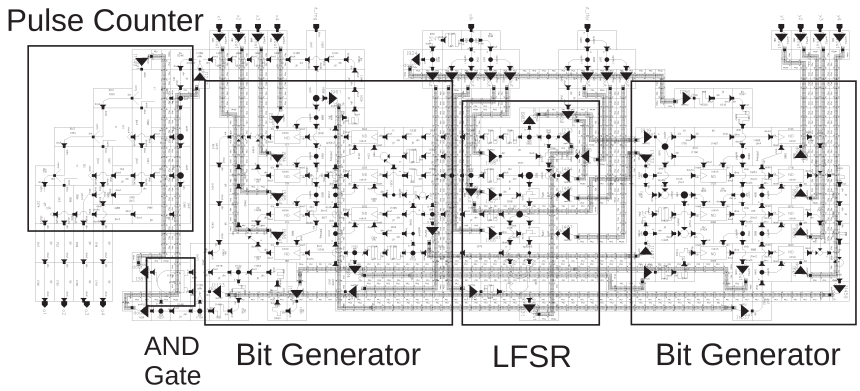

Fig. 8 Layout of a 4-bit design of the proposed truncated multiplier.

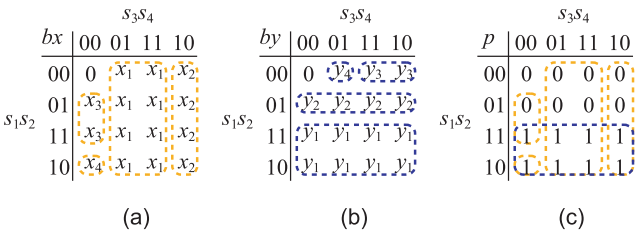

Fig. 6 Karnaugh maps of bx and by for $n = 4$ ((a) and (b), respectively), and Karnaugh map of p when $X = [0.1111]_2$ and $Y = [0.1000]_2$ (c).

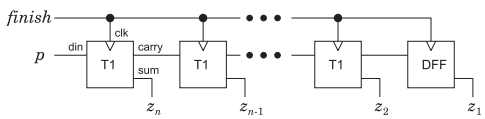

Fig. 7 Detailed design of the pulse counter.

(=$[1000]_2$) pulses in a period of T cycles because the number of ones in Fig. 6 (c) is 8. Therefore, it outputs $[0.1000]_2$ as the result.

3.3 Detailed Design

For the pattern generator, a linear feed-back shift register (LFSR) could be used. For given n, an LFSR whose period T is $2^n - 1$ can be designed.

We show a detailed design of the pulse counter in Fig. 7 as an example. Each T1 counts up its input pulses, and outputs a carry pulse every two input pulses. Once we feed a pulse for $finish$, the result preserved as the internal states of the T1s is obtained at circuit outputs $z_n, z_{n-1}, \ldots, z_1$. To obtain the result, w_n signal in the bit generator can be utilized to feed a pulse for $finish$ terminal because it feeds a pulse every T cycles.

4. Evaluation Results

We designed a layout of a 4-bit design of the proposed truncated multiplier. An LFSR was used as the pattern generator and the pulse counter shown in Sect. 3.3 was used in the layout. We used Cadence Virtuoso and the cell library designed for AIST advanced process (ADP2) [7].

The layout of the multiplier is shown in Fig. 8. Its circuit area is 0.57 mm^2 (0.45 × 1.26 mm^2), and the number of Josephson junctions (JJs) is 996. By the logic level simulation considering delay of gates using Cadence Verilog-XL, the functionality of the circuit was verified. It was estimated to work at high-frequency up to 40 GHz. It outputs a multiplication result every 15 (= $2^4 - 1$) cycles.

A bit-serial design of a multiplier was proposed in [4]. In [4], integer multiplication is carried out internally 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 of the proposed design is smaller than that of the bit-serial design.

The error in the multiplication result of the 4-bit design was evaluated by numerical simulation. Its maximum absolute error is 1.0625 ulp. When multiplication is carried out normally and its 4-bit result is derived by rounding toward 0, i.e., the result is derived by simply cutting the lower bits of the true multiplication result, the maximum absolute error is 0.9375 ulp. Therefore, the error of the proposed multiplier is not large.

4-bit data type is supported in the state-of-the-art semiconductor GPUs [6] now for machine-learning applications using neural networks. The proposed multiplier would be useful for such applications.

5. Conclusion

We proposed a truncated multiplier for RSFQ circuits. The multiplier transforms its binary operands into two serialized patterns of pulses, and its result is obtained by counting pulses of the same weight. By the bit-level processing, it can be implemented in small circuit area. The layout of a 4-bit design was shown, and was estimated to work at high-frequency up to 40 GHz.

Acknowledgements

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

References

[1] D.C. Brock, "The NSA's frozen dream," IEEE Spectrum, vol.53, no.3, pp.54–60, March 2016.
[2] 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, March 1991.
[3] M. Tanaka, H. Akaike, A. Fujimaki, Y. Yamanashi, N. Yoshikawa, S. Nagasawa, K. Takagi, and N. Takagi, "100-GHz single-flux-quantum bit-serial adder based on 10-kA/cm^2 niobium process," IEEE Trans. Appl. Supercond., vol.21, no.3, pp.792–796, June 2011.
[4] H. Hara, K. Obata, H. Park, Y. Yamanashi, K. Taketomi, N. Yoshikawa, M. Tanaka, A. Fujimaki, N. Takagi, K. Takagi, and S. Nagasawa, "Design, implementation and on-chip high-speed test of SFQ half-precision floating-point multiplier," IEEE Trans. Appl. Supercond., vol.19, no.3, pp.657–660, June 2009.
[5] N.P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," Proc. 44th International Symposium on Computer Architecture (ISCA), pp.1–12, 2017.
[6] NVIDIA, "Graphics Cards with Turing GPU Architecture," https://www.nvidia.com/en-us/geforce/turing/, Accessed on Oct. 28, 2018.
[7] 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. Electronics, vol.E97-C, no.3, pp.132–140, 2014.
[8] T. Kawaguchi, K. Takagi, and N. Takagi, "A verification method for single-flux-quantum circuits using delay-based time frame model," IEICE Trans. Fundamentals, vol.E98-A, no.12, pp.2556–2564, Dec. 2015.
[9] K. Takagi, N. Kito, and N. Takagi, "Circuit description and design flow of superconducting SFQ logic circuits," IEICE Trans. Electronics, vol.E97-C, no.3, pp.149–156, 2014.