Study on Single-Flux Quantum Floating-Point Divider Based on Goldschmidt's Algorithm

Akiyoshi Sanada, Yuki Yamanashi, *Member, IEEE,* Nobuyuki Yoshikawa, *Senior Member, IEEE*

***Abstract*—We investigated a Goldschmidt's single flux quantum (SFQ) floating-point divider that is suitable for the implementation using the bit-serial pipelined SFQ circuit. We designed the bit-serial SFQ divider that outputs the 11-bit quotient. The simulation results show correct operation at the frequency of 50 GHz and bias margin of 80-125%. We also estimated dependence of latency and the number of Josephson junctions on accuracy of outputs. According to the estimation, double-precision SFQ divider can be designed using 27000 Josephson junctions, which can be implemented on one chip using the current SFQ circuit fabrication technology. Because of the small circuit area and multiplier-based hardware architecture, the investigated divider can be applied to build an SFQ graphical processing unit.**

***Index Terms*—single-flux quantum logic, superconducting integrated circuits, Goldschmidt's division algorithm**

I. Introduction

A SINGLE-flux quantum (SFQ) logic circuit has the potential for high-speed and low power operation compared to the semi-conductor CMOS circuit [1]. The SFQ logic circuit is thought to be applied to the next generation high performance computing (HPC) systems because of the high-energy efficiency [2]. In the field of HPC, a general-purpose computing on graphical processing units (GPGPU) is becoming mainstream because of its high performance in terms of calculation power and energy efficiency [3]. In the graphics processing units (GPUs), floating-point arithmetic operation plays the important role compared to general-purpose processors. Among four arithmetic operations, division is the most complex and requires the longest calculation time. It is reported that floating-point division accounts for approximately 40% of overall calculation time, although division occupies only 3% of the total instructions to perform the SPECfp92 benchmark [4].

We have been studying the GPGPU-based SFQ processor to implement high-speed and energy efficient future computing system. To implement the GPGPU-based processor, implementation of the floating-point arithmetic unit that can efficiently perform four arithmetic operation, addition, subtraction, multiplication, and division, is important. So far, addition and multiplication based on the SFQ logic have been studied using various hardware architectures, such as a bit-serial architecture [5]–[8], a bit-slice architecture [9], [10], a bit-parallel architecture [11], [12].

In spite of the fact that divider plays an important role in the computing system as mentioned before, only one SFQ divider, which employs a systolic-array architecture, has been studied [13]. Though this divider has a good scalability, large circuit area is required to scale up the bit-length because the number of processing unit cells is proportional to the number of bits. In this study, we investigate a hardware algorithm of the SFQ divider, which also can be used as a multiplier and which is suitable for implementing SFQ-based GPU, based on the bit-serial approach that can be implemented with small circuit area.

Manuscript receipt and acceptance dates will be inserted here. This work was supported in part by JSPS KAKENHI under Grant JP26220904 and JP18K04280. A. Sanada, Y. Yamanashi, and N. Yoshikawa are with Department of Electrical and Computer Engineering, Yokohama National University, Yokohama, Kanagawa 240-8501, Japan (e-mail: sanada-akiyoshi-sk@ynu.jp).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier will be inserted here upon acceptance.

II. Goldschmidt's Algorithm

There are two major hardware architecture approaches to implement the divider, subtractive and multiplicative approaches [14]. The subtractive approach, which is similar to the manual division calculation, uses subtraction and shift operations to calculate quotient. The calculation time of the subtractive method is relatively slow because the subtraction and shift sequence has to be repeated many times, at least the bit-length of the dividend. Owing to simplicity of the calculation and implementation, the subtractive approach has been widely used in commercial microprocessors [15]. The multiplicative approach uses multiplication to calculate the quotient on the basis of mathematical algorithm. In this approach, the input dividend converges to the quotient quadratically by iterating multiplication. The multiplicative approach also has been implemented in various microprocessors such as the IBM RS/6000 and AMD K7 processor [16]. Because the multiplier in the divider based on the multiplicative approach can be also used as the floating-point multiplier, the GPU, which has to perform both multiplication and division, can be implemented efficiently. We employed the the multiplicative approach to implement SFQ floating-point divider. As the multiplicative approach, the Newton-Raphson algorithm [14] and the Goldschmidt's algorithm [17] are well-known. We employ the Goldschmidt's algorithm for the SFQ divider because this algorithm is suitable for the pipelined SFQ logic circuit as we discuss later. And the divider using Glodschmidt's algorithm can be used as a multiplier because this algorithm uses multiplication, and it bring efficient circuit area. We investigated the floating-point division of floating-point number defined by the IEEE standard 754 [18]. The floating-point number X is represented as $X = (-1)^{X_S} \times 2^{X_E} \times X_F$, where X_S, X_E and

Step 1

$d_0 \leftarrow d_F, z_0 \leftarrow z_F, x_0 \leftarrow approximation(1/d)$

Step 2

for $i = 0$ to k do

$z_i \leftarrow z_{i-1} x_{i-1}$

$d_i \leftarrow d_{i-1} x_{i-1}$

$x_i \leftarrow 2 - d_i$

end

Step 3

Return(z_i)

Fig. 1: Goldschmidt's division algorithm

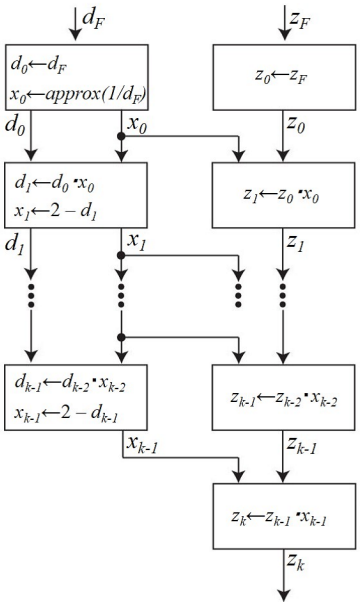


Fig. 2: Flow-chart of the iteration of Goldschmidt's division algorithm

X_F are sign, exponent, and significand bits respectively. The number of bits of the exponent and significand are represented as n_e and n_f, respectively. According to the IEEE 754, the standard format, (n_e, n_f) = (5,11), (8.24), (11,53), and (15, 113) are called half-, single-, double- and quadruple-precision floating-point numbers, respectively.

Let the dividend, the divisor, and the quotient be Z, D, and Q, respectively. When we represent Z, D, and Q as $Z = (-1)^{z_S} \times 2^{z_E} \times z_F$, $D = (-1)^{d_S} \times 2^{d_E} \times d_F$, and $Q = (-1)^{q_S} \times 2^{q_E} \times q_F$, respectively, where ($z_S$, d_S, q_S), (z_E, d_E, q_E), and (z_F, d_F, q_F) are sign, exponent, and significand of Z, D, and Q. Q can be calculated by

$$Q = \frac{Z}{D} = \frac{(-1)^{z_S} \times 2^{z_E} \times z_F}{(-1)^{d_S} \times 2^{d_E} \times d_F} = (-1)^{z_S \oplus d_S} \times 2^{z_E - d_E} \times \frac{z_F}{d_F}. \quad (1)$$

The sign, exponent, and the significand, q_S, q_E, and q_F of Q, can be represented as

$$q_S = z_S \oplus d_S, \quad (2)$$

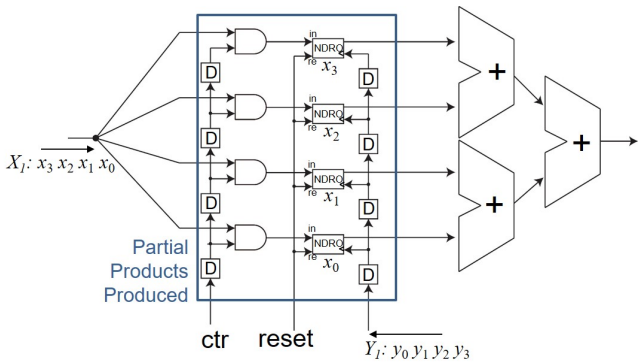


Fig. 3: Block diagram of bit-serial multiplier

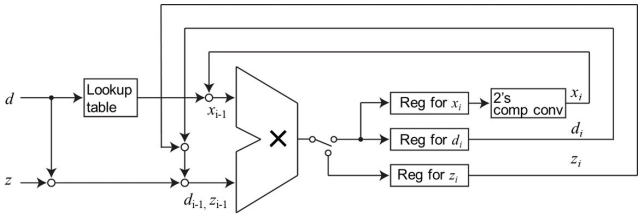


Fig. 4: The block diagram of the proposed bit-serial Goldschmidt's SFQ divider

$$q_E = z_E - d_E + n_{bias}, \quad (3)$$

$$q_F = z_F / d_F, \quad (4)$$

where n_{bias} is the bias for significand [18]. Bias is added to exponent to be an unsigned number, which is used to simplify a comparison of two exponents. To calculate q_F, we employ the Goldschmidt's division algorithm. The Goldschmidt's di-vision algorithm is based on a nature of fraction that the value does not change when the same value is multiplied to both the denominator and the numerator. In the Goldschmidt's division algorithm, the quotient q_F (= z_F/d_F) is calculated by iterating multiplications as

$$\frac{z_F}{d_F} = \frac{z_F}{d_F} \cdot \frac{x_0 \cdot x_1 \cdots x_{k-1}}{x_0 \cdot x_1 \cdots x_{k-1}}, \quad (5)$$

where k is the number of iterations of multiplication,

$$d_F \cdot x_0 \cdot x_1 \cdots x_{k-1} \rightarrow 1. \quad (6)$$

Let us define d_i and z_i as

$$d_i = d_{i-1} \cdot x_{i-1} = d_0 \cdot x_0 \cdot x_1 \cdots x_{i-1} \quad (7)$$

and

$$z_i = z_{i-1} \cdot x_{i-1} = z_0 \cdot x_0 \cdot x_1 \cdots x_{i-1}, \quad (8)$$

where d_0 and z_0 are denominator d_F and numerator z_F, respectively. By choosing x_i to satisfy the following:

$$x_i = 10_{(2)} - d_i \ (for\ i = 1, 2, \cdots k-1), \quad (9)$$

the quotient q_F converges to 1 [3]. To shorten the conversion time of q_F, x_0, the initial value of x_i, is set to be the approximation value of $1/d_F$ [14]. Fig. 1 shows a flow of

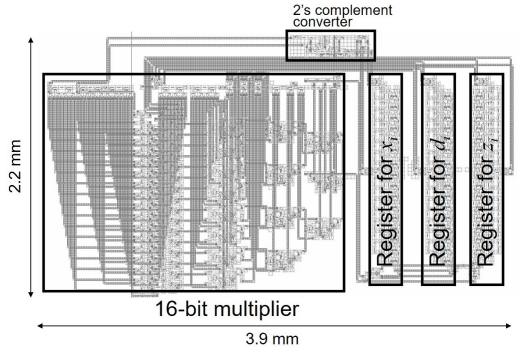


Fig. 5: The physical layout of 4-bit Goldschmidt's SFQ divider

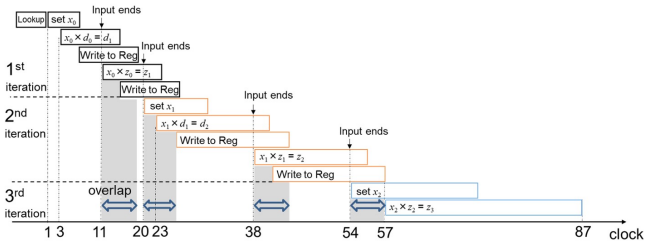


Fig. 6: Schedule of each component in the 4-bit input Goldschmidt's SFQ divider

the Goldschmidt's division algorithm. Fig. 2 shows a flowchart of the division based on Goldschmidt's algorithm. One can see that the two calculations represented by (7) and (8) can be performed by repeating simple multiplications. Moreover, multiplication represented by (7) and (8) can be performed independently. That means the data hazard does not occur in each multiplication branch, the multiplication can be performed efficiently by introducing pipelining. Therefore, Goldschmidt's division algorithm with iteration of simple multiplications is suitable for the SFQ circuit that has the latching function in logic gates and implementation of the pipelining without pipeline registers. Let assume x_0 to satisfy the following:

$$x_0 = \frac{1}{d_0} - \epsilon_0, \ 0 \leq \epsilon_0 < 2^{-p} \tag{10}$$

where ϵ_0 is the error between x_0 and $1/d_0$, and p is the ϵ_0 represented by using the unit of bit. Since Goldschmidt's division algorithm converges to accurate quotient quadratically, bit accuracy of q_F is improved to 7-bit, 15-bit, 31-bit, 63-bit and 127-bit after iteration.

III. Circuit Design and Peformance Evaluation

From equations (2), (3) and (4), floating-point number division can be calculated by using the exclusive-OR gate for sign, the adder/subtractor for exponent, and divider for significand. We designed a division circuit for the significand that calculates z_F/d_F based on the Goldschmidt's division algorithm as shown in Fig. 2. Circuit components in the division circuit for the significand are the lookup table for inputting x_0, the multiplier, 2's complement converter (2's comp conv.)

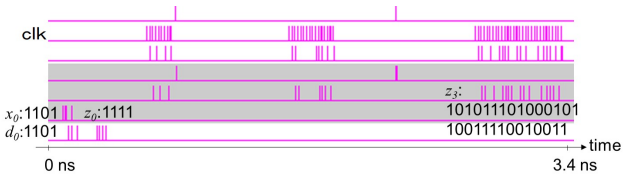


Fig. 7: A simulation result of the divider

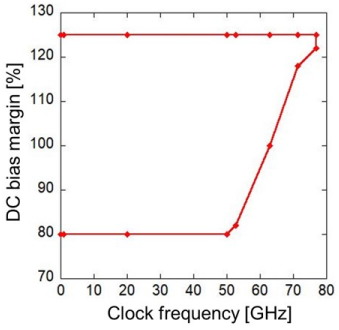


Fig. 8: Frequency dependence of the bias margin

for x_i, and registers (Reg) that store the multiplication results of each iteration stage. We employ the bit-serial multiplier proposed in [8] that has less latency and good scalability shown in Fig. 3.

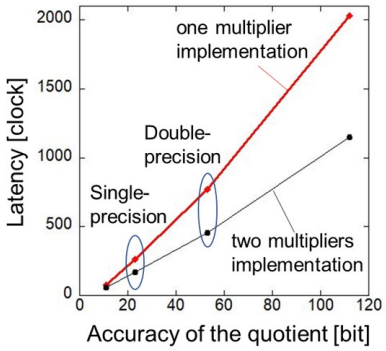
Fig. 4 shows the block diagram of the proposed bit-serial Goldschmidt's SFQ divider composed of circuit elements mentioned above. To evaluate the performance and the circuit scale of the SFQ divider, we designed the bit-serial divider using the cell library we developed [19] for the AIST 10 $\mathrm{kA/cm^2}$ Nb advanced process 2 [20]. In order to reduce the propagation delay of the data, passive transmission lines with the characteristics of 3.5 Ω are used for wiring between each circuit component [21], [22]. Because implementation of the large-scale SFQ lookup table using the current superconducting circuit fabrication process is difficult, use of the SFQ/CMOS hybrid lookup table [23] is assumed in the circuit design. Fig. 5 shows the layout of designed 4-bit SFQ divider composed of 8,091 JJs assuming the bit length of d and z are 4 and the accuracy of x_0 is 2^{-3} ($p = 3$). When these bit length and accuracy are assumed, the accuracy of quotient q_F would reach to 2^{-11} after iteration of multiplication three times. Since the accuracy of the 2^{-11} is acceptable for the practical application, we evaluated the latency of the divider assuming the 3 iterations. Fig. 6 shows the time scheduling of each calculation of the 4-bit input Goldschmidt's SFQ divider.

We have simulated the designed circuit with Cadence Verilog-XL simulator using the behavior model that defines setup time, hold time, and latency of each cell. Fig. 7 shows the digital simulation results of the designed half-precision SFQ divider. In this test sequence, $d_0 = 0.1101$, $z_0 = 1.111$, $x_0 = 1.101$, yielding the quotient $z_3 = 10.101110100010110011110010011$. The simulation results show that designed divider operates correctly up to the clock frequency of 76.9 GHz. The latency of the 4-bit divider is 87

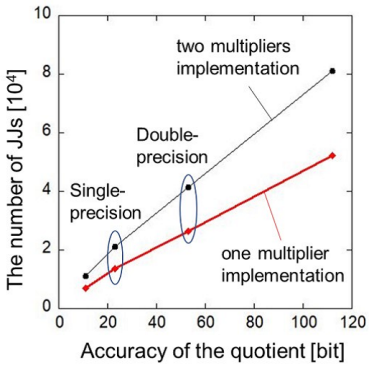
clocks as shown in Fig. 6. The dc bias margin of the 4-bit divider is 80-125%. The clock frequency dependence of the dc bias margin is shown in Fig. 8.

Based on the circuit divider design result, we estimated latency and the number of Josephson junctions (JJs) of the SFQ divider assuming p of 8 as a function of bit-length of the input data, which is the same as the accuracy of quotient. Fig. 9 shows the estimated latency and the number of JJs of the bit-serial SFQ divider as a function of accuracy of the quotient. The order of latency and the number of JJs are $O(nlogn)$ and $O(n)$ respectively, where n is the bit length of the input data. From this estimation, when the circuit operates at the frequency of 50 GHz, single- and double-precision division can be performed with the latency of 5.3, 15.1 ns, respectively. To improve the latency of the divider more, using two multipliers is effective. As discussed in section 2, the two multiplication branches can be performed independently. In the case we adopt expanded architecture, the latency of the divider can be reduced by approximately 40% by using two multipliers though the circuit area of the divider increases. It cannot be reduced by 50% because multiplication can not be parallelized at the last iteration.

We also estimated the number of JJs to as the function of accuracy of quotient. Fig. 9 (b) shows the dependence of the number of the JJs of the divider on the bit length of the input data. The order of the number of JJs are O(n) because the number of JJs of the multiplier is proportional of input bit length [8]. According to our estimation, the number of JJs required to implement single- and double-precision dividers are 13,594 and 26,518 respectively. These JJ number can be implemented using the current circuit fabrication process on one chip [24]. If we employ the parallel multiplication as mentioned above, the number of JJs of the double-precision divider is estimated to be 41,284.



(a)



(b)

Fig. 9: Dependences of (a) latency and (b) the number of Josephson junctions of the SFQ divider on the accuracy of the quotient

IV. Conclusion

We investigated the floating-point bit-serial divider for SFQ logic based on Goldschmidt's division algorithm. We designed the 4-bit SFQ divider composed of one pipelined multiplier using the AIST 10 $\mathrm{kA/cm^2}$ Nb advanced process. When the bit-length of the dividend and divisor are 4, accuracy of 2^{-11} can be obtained by iterating multiplication by three cycles. Estimated latency and the number of JJs of the designed 4-bit SFQ divider are 87 clocks and 8091, respectively. According to estimation based on the circuit design of the 4-bit divider, the double-precision divider can be implemented with 26,518 JJs by following this architecture, which can be implemented on one chip using the current SFQ circuit fabrication technology.

References

[1] K. K. Likharev and V. K. Semenov, "RSFQ logic/memory family: A new Josephson-junction technology for sub-terahertz-clock-frequency digital systems," *IEEE Trans. Appl. Supercond.*, vol. 1, no. 1, pp. 3–28, Mar. 1991.

[2] I. Soloviev, N. Klenov, S. Bakurskiy, M. Kupriyanov, A. Gudkov, and A. Sidorenko, "Beyond Moore's technologies: operation principles of a superconductor alternative," *Beilstein J. Nanotechnology*, vol. 8, pp. 2689–2710, Dec. 2017.

[3] S. Matsuoka, T. Aoki, and A. Nukada, "Rise of the Commodity GPGPU Vectors," *NVIDIA NVISION Conference*, San Jose, CA, Aug. 2008.

[4] S. Oberman and M. Flynn, "Implementing Division and Other Floating-Point Operations: A System Perspective," In Alefeld, Fromer and Lang, editors, *Scientific Computing and Validated Numbers*, pp. 18–24, 1996.

[5] 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/$\mathrm{cm^2}$ Niobium Process," *IEEE Trans. Appl. Supercond.*, vol. 21, no. 3, pp. 792–796, Jun. 2011.

[6] A. Akahori, M. Tanaka, A. Sekiya, A. Fujimaki, and H. Hayakawa, "Design and Demonstration of SFQ Pipelined Multiplier," *IEEE Trans. Appl. Supercond.*, vol. 13, no. 2, pp. 559–562, Jul. 2003.

[7] 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, Jun. 2015, Art. no. 1301106.

[8] H. Park, Y. Yamanashi, K. Taketomi, N. Yoshikawa, A. Fujimaki, and N. Takagi, "Fast Bit-serial Multipliers Using RSFQ Logic Circuits," *Supercond, Electron. Conf.*, O-S04, 2007.

[9] G. Tang, K.Takata, M. Tanaka, A. Fujimaki, K. Takagi, and N. Takagi, "4-bit Bit-Slice Arithmetic Logic Unit for 32-bit RSFQ Microprocessors," *IEEE Trans. Appl. Supercond.*, vol. 26, no. 1, Jan. 2016, Art. no. 1300106,

[10] G. Tang, K. Takagi, and N. Takagi, "32 × 32-Bit 4-Bit Bit-Slice Integer Multiplier for RSFQ Microprocessors," *IEEE Trans. Appl. Supercond.*, vol. 27, no. 3, Apr. 2017, Art. no. 1301005.

[11] M. Dorojevets, C. L. Ayala, N. Yoshikawa, and A. Fujimaki, "16-Bit Wave-Pipelined Sparse-Tree RSFQ Adder,"

$IEEE\ Trans.\ Appl.\ Supercond.$, vol. 23, no. 3, Jun. 2013, Art. no. 1700605.

[12] M. Dorojevets, A. Kasperek, N. Yoshikawa, and A. Fujimaki, "20-GHz 8 × 8-bit Pipelined RSFQ Multiplier," $IEEE\ Trans.\ Appl.\ Supercond.$, vol. 23, no. 3, Jun. 2013, Art. no. 1300104.

[13] M. Tanaka, K. Obata, K. Takagi, N. Takagi, A. Fujimaki, and N. Yoshikawa, "A High-Throughput Single-Flux Quantum Floating-Point Serial Divider Using the Signed-Digit Representation," $IEEE\ Trans.\ Appl.\ Supercond.$, vol. 19, no. 3, pp. 653–656, Jun. 2009.

[14] P. Soderquist and M. Leeser, "Area and Performance Trade-offs in Floating-Point Divide and Square Root Implementations," $ACM\ Computing\ Surveys$, vol. 28, pp. 518–564, 1996.

[15] J. L. Hennessy and D. A. Patterson, "Computer Architecture: A Quantitative Approach, 4th edit." $Morgan\ Kaufmann\ Publisher$, 2007.

[16] S. Oberman, "Floating-Point Division and Square Root Algorithms and Implementation in the AMD-K7 Microprocessor," Proc. $14th\ IEEE\ Symp.\ Computer\ Arithmetic$, Los Alamitos, CA, pp. 106–115, 1999.

[17] R. Goldschmidt, "Applications of division by convergence," $Master\ thesis$, MIT, 1964.

[18] ANSI/IEEE Std 754-2008, IEEE Standard for Floating-Point Arithmetic, 2008.

[19] H. Akaike, M. Tanaka, K. Takagi, I. Kataeva, R. Kasagi, A. Fujimaki, M. Igarashi, H. Park, Y. Yamanashi, N. Yoshikawa, K. Fujiwara,S. Nagasawa, M. Hidaka, and N. Takagi, "Design of single flux quantum cells for a 10-Nb-layer process," $Phys.\ C:\ Supercond.$, vol. 469, no. 15–20, pp. 1670–1673, Oct. 2009.

[20] M. Hidaka, S. Nagasawa, K. Hinode, and T. Satoh, "Improvements in Fabrication Process for Nb-Based Single Flux Quantum Circuits in Japan," $IEICE\ Trans.\ Electron.$, vol. E91-C, no. 3, pp. 318–324, Mar. 2008.

[21] T. Yamada, H. Ryoki, A. Fujimaki, and S. Yorozu, "Flexible Superconducting Passive Interconnects with 50-Gb/s Signal Transmissions in Single-Flux-Quantum Circuits," $Jpn.\ J.\ Appl.\ Phys.$, vol. 45, Part 1, no. 2A, pp. 752–757, 2006.

[22] T. Yamada, and A. Fujimaki, "A Novel Splitter with Four Fan-Outs for Ballistic Signal Distribution in Single-Flux Quantum Circuits up to 50Gb/s," $Jpn.\ J.\ Appl.\ Phys.$, vol. 45, pp. L262–L264, Mar. 2006.

[23] G. Konno, Y. Yamanashi, and N. Yoshikawa, "Fully Functional Operation of Low-Power 64-kb Josephson-CMOS Hybrid Memories," $IEEE\ Trans.\ Appl.\ Supercond.$, vol. 27, no. 4, Jun. 2017, Art. no. 1300607.

[24] M. Tanaka, Y. Yamanashi, N. Irie, H. 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, Oct. 2007.