Efficient Hardware Architecture for Taylor-Series Expansion Calculation Using Distributed Arithmetic with Term Division

Xaybandith Hemthavy[1,*], Jianglin Wei[1], Shogo Katayama[1], Anna Kuwana[1], Haruo Kobayashi[1], Kazuyoshi Kubo[2]

[1] Dept. of Electrical Engineering and Informatics, Gunma University, Gunma 376-8515, Japan
[2] Oyama National College of Technology, Tochigi 323-0806, Japan
* T221D074@gunma-u.ac.jp

Abstract - This paper describes the digital arithmetic that reduces the calculation and hardware (logic circuits and memory) for Taylor series expansion calculation by applying the distributed (bit-serial) arithmetic with the proposed term division method. The distributed arithmetic (DA) is a multiplier-less approach for calculating multiply-accumulate (MAC) operation, but its direct application to the Taylor-series expansion calculation still demands almost the same number of multiplications as the direct calculation and additionally large size Look Up Table (LUT); hence it is useless. Then we propose the term division method which can reduce the number of multiplications and the LUT size significantly. Further, we found that the optimal number of the term division is approximately the square root of the number of the Taylor series expansion terms.

I. Introduction

Digital electronic devices are getting smaller and low power in these days, and many multiply-accumulate (MAC) operations are often used there [1]. The distributed arithmetic (DA) is often used for MAC operations without multipliers, and hence it is small circuit and low power. It has been used in adaptive filter (ADF) using least-mean-square (LMS)[2]-[9].

Taylor series expansion is a popular approach to calculate various functions effectively in digital signal processing [10]-[13] and it also uses the MAC operations. In this paper we investigate the application of the DA to the Taylor-series expansion calculation. Direct application of the DA needs large memory but the number of the required multiplication is not reduced. Then we propose here the term division method for the DA application of the Tayler-series expansion calculation. Its principle and verification with some examples are shown.

II. Taylor Series Expansion

Taylor series expansion is defined as following:

$$f(x) = f(\alpha) + \frac{f'(\alpha)}{1!}(x-\alpha) + \frac{f''(\alpha)}{2!}(x-\alpha)^2 + \cdots \quad (1)$$

Suppose that $\alpha = 0$, $f(x)$ has 10 terms, and each coefficient is given by a_n, Eq. (1) yields to the following:

$$f(x) = a_0 + a_1x^1 + a_2x^2 + a_3x^3 + \cdots + a_9x^9 \quad (2)$$

Rearrange Eq. (2) as shown in Fig. 1 to multiply x sequentially, and then we see that there the number of multiplications is 9 and also the number of additions is 9. Notice that in many cases, the calculation and hardware bottleneck is the multiplication.

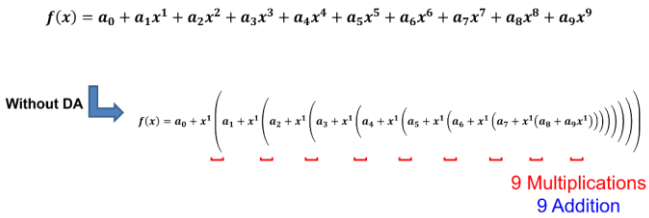


Fig. 1: Direct calculation of Taylor-series expansion.

III. Distributed Arithmetic

The DA performs the MAC calculation without multipliers and hence it can reduce the hardware amount. Fig. 2 shows the DA circuit composed of a Look-Up Table (LUT), a one-bit-right-shifter, an adder and registers. As an example, consider the following calculation:

$$y = h_0x_0 + h_1x_1 + h_2x_2 \quad (3)$$

Suppose that the inputs of x_0, x_1, x_2 in binary format are given as follows:

$$x_0 = 11.1,\ x_1 = 11.0,\ x_2 = 10.1.$$

In Fig. 2, these are provided as "address", and the calculation result y is obtained as "Out". Now prepare an LUT so that each sum comes from coefficients at each term where the address is "1" in Fig.3. As shown in Fig. 3, calculate sum of each address and multiply $\frac{1}{2}$ every time when doing one-bit-right-shift to next digit, repeatedly from LSB to MSB in a bit-serial manner. At the end, define b as the number of integer bits and (virtually) multiply 2^b or perform b-bit left-shift to correct the position of decimal point. Notice that in many actual implementation cases, there is no need for multiplication or bit-shift for the decimal point correction, but only interconnection change is enough

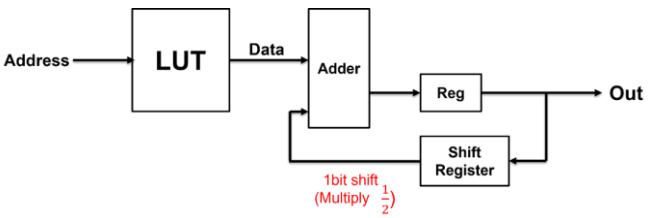


Fig. 2: Distributed arithmetic circuit.

$$y = h_0x_0 + h_1x_1 + h_2x_2$$

	x_0	x_1	x_2
MSB	1	0	1
	1	1	0
LSB	1	1	1

LUT			
x_0	x_1	x_2	Sum
0	0	0	0
1	0	0	h_0
0	1	0	h_1
0	0	1	h_2
1	1	0	$h_0 + h_1$
1	0	1	$h_0 + h_2$
0	1	1	$h_1 + h_2$
1	1	1	$h_0 + h_1 + h_2$

$$h_0 + h_1 + h_2$$

(a) MAC calculation of LSB.

$$y = h_0x_0 + h_1x_1 + h_2x_2$$

	x_0	x_1	x_2
MSB	1	0	1
	1	1	0
LSB	1	1	1

LUT			
x_0	x_1	x_2	Sum
0	0	0	0
1	0	0	h_0
0	1	0	h_1
0	0	1	h_2
1	1	0	$h_0 + h_1$
1	0	1	$h_0 + h_2$
0	1	1	$h_1 + h_2$
1	1	1	$h_0 + h_1 + h_2$

$$\left(\frac{1}{2}\times(h_0 + h_1 + h_2)\right) + (h_0 + h_1)$$

(b) MAC calculation of LSB + 1.

$$y = h_0x_0 + h_1x_1 + h_2x_2$$

	x_0	x_1	x_2
MSB	1	0	1
	1	1	0
LSB	1	1	1

LUT			
x_0	x_1	x_2	Sum
0	0	0	0
1	0	0	h_0
0	1	0	h_1
0	0	1	h_2
1	1	0	$h_0 + h_1$
1	0	1	$h_0 + h_2$
0	1	1	$h_1 + h_2$
1	1	1	$h_0 + h_1 + h_2$

$$\left(\frac{1}{2}\times\left(\frac{1}{2}\times(h_0 + h_1 + h_2)\right) + (h_0 + h_1)\right) + (h_0 + h_2)$$

(c) MAC calculation of MSB.

$$y = h_0x_0 + h_1x_1 + h_2x_2$$

	x_0	x_1	x_2
MSB	1	0	1
	1	1	0
LSB	1	1	1

2 bit (Integer part)

LUT			
x_0	x_1	x_2	Sum
0	0	0	0
1	0	0	h_0
0	1	0	h_1
0	0	1	h_2
1	1	0	$h_0 + h_1$
1	0	1	$h_0 + h_2$
0	1	1	$h_1 + h_2$
1	1	1	$h_0 + h_1 + h_2$

$$\left(\frac{1}{2}\times\left(\frac{1}{2}\times(h_0 + h_1 + h_2)\right) + (h_0 + h_1)\right) + (h_0 + h_2)\Big) \times 2^2$$

(d) Correction of decimal point position

Fig. 3: Distributed arithmetic algorithm for Eq. (3).

$$y = h_0x_0 + h_1x_1 + h_2x_2$$

	x^0	x^1	x^2
MSB	0	0	1
	1	1	0
	0	1	0
LSB	0	0	1

LUT			
x^0	x^1	x^2	Sum
0	0	0	0
1	0	0	a_0
0	1	0	a_1
0	0	1	a_2
1	1	0	$a_0 + a_1$
1	0	1	$a_0 + a_2$
0	1	1	$a_1 + a_2$
1	1	1	$a_0 + a_1 + a_2$

$$a_2$$

(a) MAC calculation of LSB

$$y = h_0x_0 + h_1x_1 + h_2x_2$$

	x^0	x^1	x^2
MSB	0	0	1
	1	1	0
	0	1	0
LSB	0	0	1

LUT			
x^0	x^1	x^2	Sum
0	0	0	0
1	0	0	a_0
0	1	0	a_1
0	0	1	a_2
1	1	0	$a_0 + a_1$
1	0	1	$a_0 + a_2$
0	1	1	$a_1 + a_2$
1	1	1	$a_0 + a_1 + a_2$

$$\left(\frac{1}{2}\times(a_2)\right) + (a_1)$$

(b) MAC calculation of LSB + 1

$$y = h_0x_0 + h_1x_1 + h_2x_2$$

	x^0	x^1	x^2
MSB	0	0	1
	1	1	0
	0	1	0
LSB	0	0	1

LUT			
x^0	x^1	x^2	Sum
0	0	0	0
1	0	0	a_0
0	1	0	a_1
0	0	1	a_2
1	1	0	$a_0 + a_1$
1	0	1	$a_0 + a_2$
0	1	1	$a_1 + a_2$
1	1	1	$a_0 + a_1 + a_2$

$$\left(\frac{1}{2}\times\left(\frac{1}{2}\times(a_2)\right) + (a_1)\right) + (a_0 + a_1)$$

(c) MAC calculation of LSB + 2

$$y = h_0x_0 + h_1x_1 + h_2x_2$$

	x^0	x^1	x^2
MSB	0	0	1
	1	1	0
	0	1	0
LSB	0	0	1

LUT			
x^0	x^1	x^2	Sum
0	0	0	0
1	0	0	a_0
0	1	0	a_1
0	0	1	a_2
1	1	0	$a_0 + a_1$
1	0	1	$a_0 + a_2$
0	1	1	$a_1 + a_2$
1	1	1	$a_0 + a_1 + a_2$

$$\left(\frac{1}{2}\times\left(\frac{1}{2}\times\left(\frac{1}{2}\times(a_2)\right) + (a_1)\right) + (a_0 + a_1)\right) + (a_2)$$

(d) MAC calculation of MSB

	x^0	x^1	x^2
x^0	0	1_0	0	0
x^1	0	1_0	1	0
x^2	1	0_0	0	1

	x^0	x^1	x^2
	0	0	1
	1	1	0
	0	1	0
	0	0	1

2bit (Integer part)

$$f(x) = \left(\frac{1}{2}\times\left(\frac{1}{2}\times\left(\frac{1}{2}\times(a_2)\right) + (a_1)\right) + (a_0 + a_1)\right) + (a_2)\Big) \times (2)^2$$

(e) Correction of the position of decimal point

Fig. 4: Distributed arithmetic algorithm for Tayler series expansion in Eq. (4).

IV. Distributed Arithmetic for Taylor Series Expansion

A. *How to DA for Taylor-Series Expansion Calculation.*

Now let us consider about Eq. (4) when decimal $x = 1.5$.

$$f(x) = a_0 + a_1x^1 + a_2x^2 \tag{4}$$

Notice that we need to calculate x^2 in binary format by multiplication of $x \times x$ to make a LUT and then do operations as shown in Fig. 4.

B. *Direct Application of DA*

Let us consider the direct application of the DA to the Taylor-series expansion with 10 terms (Eq. (10)). We need 8 multiplications for $x^2, x^3, x^4, x^5, x^6, x^7, x^8, x^9$ to make a LUT with 1024 words.

$$\begin{aligned} f(x) = {} & a_0 + a_1x^1 + a_2x^2 + a_3x^3 + a_4x^4 \\ & + a_5x^5 + a_6x^6 + a_7x^7 + a_8x^8 + a_9x^9. \end{aligned} \tag{5}$$

We see that since the number of multiplications is almost the same as the direct calculation in Fig. 1 and some additional hardware is required, the direct DA application is completely useless.

C. *Application of DA with Term Division Method*

Now, we propose the DA with the term division method. We rearranged Eq. (5) as follows:

$$\begin{aligned} f(x) = {} & a_0 + a_2x^2 + a_4x^4 + a_6x^6 + a_8x^8 + \\ & + x\,(a_1 + a_3x^2 + a_5x^4 + a_7x^6 + a_9x^8). \end{aligned} \tag{6}$$

We calculated x^2, x^4, x^6, and x^8 with 4 multiplications. Also, the followings are calculated with DA:

$$A = a_0 + a_2x^2 + a_4x^4 + a_6x^6 + a_8x^8. \tag{7}$$

$$B = a_1 + a_3x^2 + a_5x^4 + a_7x^6 + a_9x^8. \tag{8}$$

Then we calculated the following with 1 multiplication and 1 addition:

$$A + x\,B \tag{9}$$

Then 5 multiplications, 1 addition and 2 DA calculations are required. Also 2 LUTs with 2^5 words are used; the total LUT size is $2\text{x}2^5 (= 64)$ words.

Thus, the above investigated method can reduce the number of multiplications by almost half compared to the direct calculation method without DA, though 2 distributed arithmetic calculations are required.

Notice that if we calculate the followings with DA:

$$S = a_2x^2 + a_4x^4 + a_6x^6 + a_8x^8. \tag{10}$$

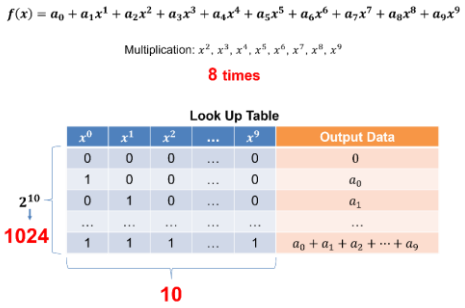
$$T = a_3x^2 + a_5x^4 + a_7x^6 + a_9x^8. \tag{11}$$

Then we obtain

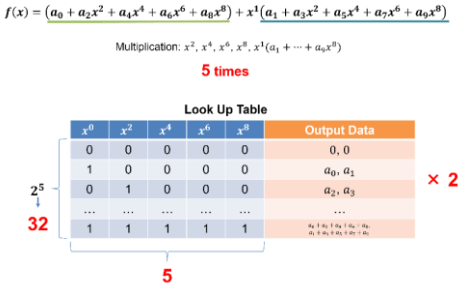
$$\text{A} = a_0 + S \tag{12}$$

$$\text{B} = a_1 + T \tag{13}$$

and the total LUT size is $2\text{x}2^4 (= 32)$ words, though two more additions are required.



(a) LUT for no term division



(b) LUTs for term division by 2

Fig. 5: LUTs in cases without term division and with term division by 2.

V. Verification of Term Division Method with Taylor Series Expansion with Many Terms

We investigated the cases of several different terms to find effective term division.

Let us consider the Taylor series expansion with 16 terms as follows:

$$\begin{aligned} f(x) = {} & a_0 + a_1x^1 + a_2x^2 + a_3x^3 \\ & + a_4x^4 + a_5x^5 + a_6x^6 + a_7x^7 \\ & + a_8x^8 + a_9x^9 + a_{10}x^{10} + a_{11}x^{11} \\ & + a_{12}x^{12} + a_{13}x^{13} + a_{14}x^{14} + a_{15}x^{15} \end{aligned} \tag{14}$$

(i) We considered the term division by 2:

$$\begin{aligned} f(x) = {} & a_0 + a_2x^2 + a_4x^4 + a_6x^6 \\ & + a_8x^8 + a_{10}x^{10} + a_{12}x^{12} + a_{14}x^{14} \\ & + x\,(a_1 + a_3x^2 + a_5x^4 + a_7x^6 \\ & + a_9x^8 + a_{11}x^{10} + a_{13}x^{12} + a_{15}x^{14}). \end{aligned} \tag{15}$$

We calculated x^2, x^4, x^6, x^8, x^{10}, x^{12} and x^{14} with 7 multiplications. Then the followings are calculated with DA:

$$\begin{aligned} C = {} & a_0 + a_2x^2 + a_4x^4 + a_6x^6 \\ & + a_8x^8 + a_{10}x^{10} + a_{12}x^{12} + a_{14}x^{14}. \end{aligned} \tag{16}$$

$$\begin{aligned} D = {} & a_1 + a_3x^2 + a_5x^4 + a_7x^6 \\ & + a_9x^8 + a_{11}x^{10} + a_{13}x^{12} + a_{15}x^{14}. \end{aligned} \tag{17}$$

Then we calculated the following with 1 multiplication and 1 addition:

$$C + x\,D \tag{18}$$

Then 8 multiplications, 1 addition and 2 DA calculations are required. Also 2 LUTs with 2^7 words are used: the total LUT size is $2\text{x}2^7 (= 512)$ words.

(ii) Next, we considered the term division by 4:

$$f(x) = a_0 + a_4x^4 + a_8x^8 + a_{12}x^{12} + x(a_1 + a_5x^4 + a_9x^8 + a_{13}x^{12}) + x^2(a_2 + a_6x^4 + a_{10}x^8 + a_{14}x^{12}) + x^3(a_3 + a_7x^4 + a_{11}x^8 + a_{15}x^{12}). \quad (19)$$

We calculated x^4, x^8 and x^{12} with 3 multiplications. Then the followings are calculated with DA:

$$E = a_0 + a_4x^4 + a_8x^8 + a_{12}x^{12}. \quad (20)$$
$$F = a_1 + a_5x^4 + a_9x^8 + a_{13}x^{12}. \quad (21)$$
$$G = a_2 + a_6x^4 + a_{10}x^8 + a_{14}x^{12}. \quad (22)$$
$$H = a_3 + a_7x^4 + a_{11}x^8 + a_{15}x^{12}. \quad (23)$$

Then we calculated the following with 3 multiplications and 3 additions:

$$E + x\left(F + x\left(G + x\,H\right)\right). \quad (24)$$

Then 6 multiplications, 3 additions and 4 DA calculations are required. Also 4 LUTs with 2^4 words are used: the total LUT size is $4\text{x}2^4 (= 64)$ words.

(iii) We considered the term division by 8.

$$f(x) = a_0 + a_8x^8 + x(a_1 + a_9x^8) + x^2(a_2 + a_{10}x^8) + x^3(a_3 + a_{11}x^8) + x^4(a_4 + a_{12}x^8) + x^5(a_5 + a_{13}x^8) + x^6(a_6 + a_{14}x^8) + x^7(a_7 + a_{15}x^8). \quad (25)$$

We calculated x^8 with 1 multiplication. Then the followings are calculated with DA:

$$I = a_0 + a_8x^8. \quad (26)$$
$$J = a_1 + a_9x^8. \quad (27)$$
$$K = a_2 + a_{10}x^8. \quad (28)$$
$$L = a_3 + a_{11}x^8. \quad (29)$$
$$M = a_4 + a_{12}x^8. \quad (30)$$
$$N = a_5 + a_{13}x^8. \quad (31)$$
$$P = a_6 + a_{14}x^8. \quad (32)$$
$$Q = a_7 + a_{15}x^8. \quad (33)$$

Then we calculated the following with 7 multiplications and 7 additions:

$$I + x\left(J + x\left(K + x\left(L + x\left(M + x(N + x(P + xQ))\right)\right)\right)\right) \quad (34)$$

Then 8 multiplications, 7 additions and 8 DA calculations are required. Also 8 LUTs with 4 words are used; the total LUT size is 32 words.

As shown in the above, the term division method results in reduction of LUT memory size and number of multiplications. We see that when the number of multiplications is dominant for hardware size and speed, the term division method is effective.

As Table 1 shows, the more you divide by, the smaller the LUT memory size. However, the number of multiplications begins to increase at certain point; the optimal point is the division by approximately $\sqrt{N}$ for N-term Taylor series expansion and the number of multiplications is approximately $2\sqrt{N}$- 2. Fig. 6 shows some explanations.

Due to the recent advancement of the VLSI technology, the large memory size LUT may not be a big issue in the cost viewpoint. However, the large size LUT requires some time for full data load, which would slow down the calculation speed.

Table 1: Number of Taylor series expansion terms, number of term divisions, number of multiplications, and LUT size.

	Division by	Number of multiplications	LUT size
16 terms	1	14	65536
	2	8	512
	4	6	64
	8	8	32
32 terms	1	30	4294967296
	2	16	131072
	4	10	1024
	8	10	128
	16	16	64
64 terms	1	62	1.84×10^{19}
	2	32	8589934592
	4	18	262144
	8	14	2048
	16	18	256
	32	32	128

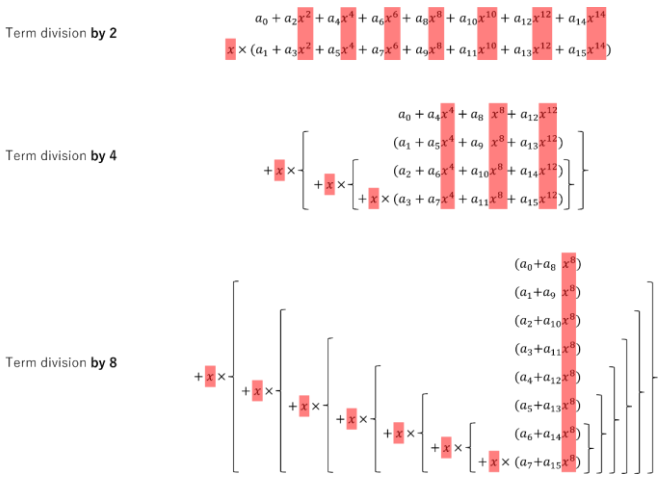


Fig. 6: Term division and multiplication parts

VI. Conclusion

We have investigated the application of DA to the Taylor-series expansion calculation and proposed the term division method. We have shown here that it can reduce the number of multiplications compared to the direct calculation, and reduce the memory size of LUT compared to case without the term division method. Notice that the multiplication is often dominant for hardware size and speed. The optimal division number to minimize the number of the multiplications is approximately $\sqrt{N}$ for N-term Taylor series expansion, and the LUT size becomes smaller as the number of the term divisions is increases. We have shown this statement when the

number of the Taylor series expansion terms is 10, 16, 32 and 64. The proposed method would make the Taylor-series expansion method for the digital calculation of various functions more effective, especially when the number of Taylor series expansion terms is large.

Finally, we conclude this paper by remaking the following: Today, much attention is being paid to Memory-in-Computing [15, 16]. There is a lot of memory close to digital arithmetic circuits is available and our proposed algorithm would be suitable to this architecture.

References

[1] M. T. Khan, R. A. Shaik, "High-performance VLSI architecture of DLMS adaptive filter for fast-convergence and low-MSE", *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 69, No. 4, pp. 2106—2110 (Apr. 2022).

[2] D. J. Allred, H. Yoo, V. Krishnan, W. Huang, D. V. Anderson, "LMS adaptive filters using distributed arithmetic for high throughput", *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 52, No. 7, pp. 1327–1337 (Jul. 2005).

[3] R. Guo, L. S. DeBrunner, "Two high-performance adaptive filter implementation schemes using distributed arithmetic", *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 58, No. 9, pp. 600–604 (Sept. 2011).

[4] M. S. Prakash, R. A. Shaik, "Low-area and high-throughput architecture for an adaptive filter using distributed arithmetic", *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 60, No. 11, pp. 781–785 (Nov. 2013).

[5] S. Y. Park, P. K. Meher, "Low-power, high-throughput, and low-area adaptive FIR filter based on distributed arithmetic", *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 60, No. 6, pp. 346–350 (Jun. 2013).

[6] M. T. Khan, R. A. Shaik, S. P. Matcha, "Improved convergent distributed arithmetic based low complexity pipelined least-mean-square filter", *IET Circuits, Devices & Systems*, Vol. 12, No. 6, pp. 792–801 (May. 2018).

[7] M. T. Khan, R. A. Shaik, "Optimal complexity architectures for pipelined distributed arithmetic-based LMS adaptive filter", *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 66, No. 2, pp. 630–642 (Feb. 2019).

[8] S. Ahmad, S. G. Khawaja, N. Amjad, M. Usman, "A novel multiplier-less LMS adaptive filter design based on offset binary coded distributed arithmetic", *IEEE Access*, Vol. 9, pp. 78138—78152 (May. 2021).

[9] R. Bala, S. Aktar, "Fast Fourier transformation realization with distributed arithmetic", *International Journal of Computer Applications*, Vol. 102, No. 15, pp. 22-25 (Sept. 2014).

[10] J. Wei, A. Kuwana, H. Kobayashi, K. Kubo, "Divide and conquer: floating-point exponential calculation based on Taylor-series expansion", *IEEE 14th International Conference on ASIC*, Kunming, China (Oct. 2021).

[11] J. Wei, A. Kuwana, H. Kobayashi, K. Kubo, "IEEE754 binary32 floating-point logarithmic algorithms based on Taylor-series expansion with mantissa region conversion and division", *IEICE Trans. Fundamentals*, Vol. E105-A, No.7 (Jul. 2022).

[12] J. Wei, A. Kuwana, H. Kobayashi, K. Kubo, Y. Tanaka, "Floating-point inverse square root algorithm based on Taylor-series expansion", *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 68, No. 7, pp. 2640-2644 (Jul. 2021).

[13] J. Wei, A. Kuwana, H. Kobayashi, K. Kubo, "Revisit to floating-point division algorithm based on Taylor-series expansion", 16th IEEE Asia Pacific Conference on Circuits and Systems, Ha Long Bay, Vietnam, (Dec. 2020).

[14] E. Özalevli, W. Huang, P. E. Hasler, D. V. Anderson, "A reconfigurable mixed-signal VLSI implementation of distributed arithmetic used for finite-impulse response filtering", *IEEE Trans. Circuits and Systems-I: Regular Papers*, Vol. 55, No. 2, pp. 510—521 (Mar. 2008).

[15] J. Chen, W. Zhao, Y. Ha, "Area-efficient distributed arithmetic optimization via heuristic decomposition and in-Memroy computing", IEEE 13th International Conference on ASIC, Kunming, China (Oct. 2019).

[16] V. Lakshmi, V. Pudi, J. Reuben, "Inner product computation in-Memory using distributed arithmetic", *IEEE Transactions on Circuits and Systems I: Regular Papers* (2022 Early Access)