A parallel arithmetic for hardware realization of digital filters

Chunxiao Fan, Fu Li, Xin Cao, Biao Qian, Peipei Song

PII: S0026-2692(18)30715-8

DOI: https://doi.org/10.1016/j.mejo.2018.11.012

Reference: MEJ 4451

To appear in: *Microelectronics Journal*

Received Date: 18 September 2018

Revised Date: 21 November 2018

Accepted Date: 27 November 2018

A Parallel Arithmetic for Hardware Realization of Digital Filters

Chunxiao Fan [1], Fu Li [2*], Xin Cao [3], Biao Qian [1], Peipei Song [1]
1 School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, Anhui 230601, China
2 School of Artificial Intelligence, Xidian University, Xi'an, Shaanxi 710071, China
3 School of Information Science and Technology, Northwest University, Xi'an, Shaanxi 710069, China

Abstract: Distributed Arithmetic (DA) is a classic technique for the hardware realization of digital filters. We present a novel parallel arithmetic operation to overcome two drawbacks in existing DA and DA-based methods: 1) the throughput is difficult to improve and 2) hardware resource consumption increases exponentially with the length of filter order. The fundamental difference between the proposed and existing methods is that the proposed method factors the filter coefficients to find several simple basic operations, which can circumvent the inherent bit-serial nature of DA methods and achieve the whole operation in one clock cycle. Additionally, the number of possible basic operations increases linearly with the length of filter order, which means we can relieve the exponentially increasing hardware resource consumption. The proposed method is evaluated through two experiments, and the results demonstrate that the proposed technique outperforms existing DA and DA-based methods in terms of throughput and resource consumption.

Index Terms: Distributed Arithmetic, digital filter, convolution, hardware realization, VLSI.

I. Introduction

Digital filtering is a basic operation in digital signal processing. Distributed Arithmetic (DA) was proposed in the 1970s [1, 2] for the hardware realization design of digital filters to reduce the cost and power consumption [3–12]. With DA, the total gate count in digital filters can be reduced by as much as 80% [13]. In addition, the acceleration of deep neural networks recently attracts much attention, DA can also be an efficient method to achieve the convolution operation in hardware acceleration.

The principle of DA is to factor inputs according to the binary representation, and the possible relevant operations are selected as intermediate arithmetic operations. In order to reduce hardware resource consumption, these operations are pre-computed offline and stored in memory structures. However, because of the inherent bit-serial nature of DA methods, the throughput is difficult to

This work was supported in part by the NSFC (No. 61802105, 61701403, 61672404, 61632019, 61751310, 61472301, 61875157 and 61572387), the Fundamental Research Funds of the Central Universities of China (No. 2192018JZ2018HGBZ0154, SA-ZD160203, JBG160228, JBG160213, K5051399020 and K5051202050), the China Post-doctoral Science Foundation (No. 2018M643719), the Scientific Research Program Funded by Shaanxi Provincial Education Department (No. 18JK0767), and Natural Science Basic Research Plan in Shaanxi Province of China (Program No. 2017JQ6006 and 2016ZDJC-08).
* Corresponding author: Fu Li (e-mail: fuli@mail.xidian.edu.cn).

improve and multiple clock cycles are needed to calculate the inner-product results. The memory requirements increase exponentially with the length of filter order, which results in the exponentially increasing of hardware resource consumption. In a complex digital filter, the memory structure consumes lots of hardware resources.

Many approaches have been proposed to overcome these drawbacks [13–18]. In order to improve the throughput, [14] and [15] attempted to partition the filter input into several sub-inputs and process them using several memory structures in parallel. [13] and [16–18] describe methods that can reduce memory size by using ingenious representations. Several intermediate arithmetic operations are combined and the intermediate results which stored in ROM are anti-symmetric. The memory size in these methods can be reduced by as much as 50%. However, none of these techniques can avoid these drawbacks.

In our previous work [19], we proposed an improved signed digit (ISD) representation to eliminate computational redundancy among different multipliers. In order to overcome the drawbacks of DA, the ISD representation is adopted and applied to calculate the product of several inputs and filter coefficients in parallel. In this paper, we present a parallel arithmetic operation for the hardware realization of digital filters. The fundamental difference between the proposed method and existing methods is that the proposed method factors filter coefficients to find basic operations. This means that the inherent bit-serial nature of DA and DA-based techniques can be circumvented to avoid time delay. The intermediate arithmetic operations after factorization increase linearly with the length of filter order, which means we can avoid exponentially increasing memory size and reduce hardware resource consumption.

The remainder of this paper is organized as follows. In Section II, the principles of DA and DA-based methods are reviewed. The mathematical model for our proposed method is described in Section III. Section IV presents two experiments to verify the proposed method. Our work is concluded in Section V.

II. Review of Distributed Arithmetic

DA is widely used in the hardware realization of digital filters. In DA, all possible intermediate arithmetic operations are stored in a memory structure. These stored values can be used to calculate the final results by using repeated additions and shifting operations. In the hardware realization of digital filters, there are no multiplications, but several memory structures are required.

A digital filter of length K is used to calculate the inner-product of an impulse response vector h_k ($k = 0,1,...,K-1$) and an input vector i_{n-k} ($k = 0,1,...,K-1$). It can be represented as:

$$y = \sum_{k=0}^{K-1} h_k i_{n-k} \tag{1}$$

For simplification, the time index n of i_{n-k} is removed and i_{n-k} is represented by its two's complement binary number x_k, as shown in (2). x_k consists of N bits of data and a 1-bit sign.

$$x_k = -b_{kN} 2^N + \sum_{n=0}^{N-1} b_{kn} 2^n \tag{2}$$

$$y = \sum_{k=0}^{K-1} h_k x_k$$

In the circuit, h_k is represented in two's complement form as C_k. Using (2), we can get (3).

$$\begin{aligned} y &= \sum_{k=0}^{K-1} C_k x_k \\ &= \sum_{k=0}^{K-1} C_k [-b_{kN} 2^N + \sum_{n=0}^{N-1} b_{kn} 2^n] \\ &= \sum_{k=0}^{K-1} C_k (-b_{kN}) 2^N + \sum_{n=0}^{N-1} [\sum_{k=0}^{K-1} C_k b_{kn}] 2^n \end{aligned} \tag{3}$$

Therefore, the inner-product is transformed to solve (4), which defines a distributed arithmetic computation.

$$\sum_{k=0}^{K-1} C_k b_{kn} \tag{4}$$

Because b_{kn} is the n-th bit in the binary representation of x_k, it can only take on values of 0 and 1. This means that (4) has 2^K possible results, which is a finite number. The values can be pre-computed offline and stored in ROM as intermediate arithmetic operations and the memory size is equal to 2^K. The inner-product can then be computed by using repeated additions and shifting operations based on (3). So, the hardware realization processes one bit at a time (1-BAAT) and completing the entire operation requires $N+1$ cycles.

For a more efficient hardware realization, many improved technologies are proposed to reduce the needed memory size and clock cycles. In order to reduce memory size, memory reduced DA-based (MR-DA) methods were proposed [13, 16–18]. In these methods, x_k is represented as:

$$x_k = \frac{1}{2}[x_k - (-x_k)] \tag{5}$$

$-x_k$ can be represented in two's-complement notation as:

$$-x_k = -\bar{b}_{kN} 2^N + \sum_{n=0}^{N-1} \bar{b}_{kn} 2^n + 1 \tag{6}$$

By substituting $-x_k$ into (6), (5) becomes (7).

$$x_k = \frac{1}{2}[-(b_{kN} - \bar{b}_{kN}) 2^N + \sum_{n=0}^{N-1} (b_{kn} - \bar{b}_{kn}) 2^n - 1)] \tag{7}$$

For convenience, a new variable a_{kn} is defined as:

$$a_{kn} = \begin{cases} -(b_{kN} - \bar{b}_{kN}), & n = N. \\ b_{kn} - \bar{b}_{kn}, & n \neq N. \end{cases} \tag{8}$$

Using this definition, we find that $a_{kn} \in \{1, -1\}$. Therefore, (3) can be transformed as:

$$\begin{aligned} y &= \sum_{k=0}^{K-1} C_k x_k \\ &= \sum_{k=0}^{K-1} C_k \{\frac{1}{2}[\sum_{n=0}^{N} a_{kn} 2^n - 1]\} \\ &= \frac{1}{2} \sum_{n=0}^{N} (\sum_{k=0}^{K-1} C_k a_{kn}) 2^n - \frac{1}{2} (\sum_{k=0}^{K-1} C_k) \end{aligned} \tag{9}$$

Thus, the intermediate arithmetic operations become:

$$\sum_{k=0}^{K-1} C_k a_{kn} \text{ and } \sum_{k=0}^{K-1} C_k \tag{10}$$

As $a_{kn} \in \{1, -1\}$, the possible values of $\frac{1}{2}\sum_{k=0}^{K-1} C_k a_{kn}$ are anti-symmetric. This means that the intermediate arithmetic operations have 2^{K-1} possible values, which reduces the required memory size by half.

To reduce the necessary clock cycles, several improved methods [14, 15] increase the memory size by L times and partition each input into L sub-inputs. These methods can process L bits at a time (L-BAAT) and the operation speed is increased by a factor of L. However, the additional memory structures used by these methods consume more hardware resources than traditional DA.

Therefore, the hardware realization of digital filters with DA requires memory structures and memory size increases exponentially as the length of the filter order K increases. Additionally, completing the entire operation requires $N+1$ cycles. Many improved methods have been proposed to reduce the needed memory size and clock cycles. However, in these DA-based methods, none can accomplish this goal efficiently, and it becomes a tradeoff between the throughput and memory size.

III. Principle of the Proposed Parallel Arithmetic Operation

In DA and DA-based methods, the operations for different inputs are factorized together and several intermediate arithmetic operations are selected to be pre-computed. These intermediate arithmetic operations are analyzed in a bit-serial manner, which causes the problem that inner-product is achieved in multiple clock cycles. Additionally, the number of intermediate arithmetic operations increases exponentially, which causes memory size to increase exponentially.

In this paper, we propose a novel parallel arithmetic operation for digital filters in which coefficients are factorized to find basic operations. In the proposed method, multiple bits of input data can be processed in parallel to improve throughput. The number of intermediate arithmetic operations is also much less than that in DA, which significantly reduces hardware resource consumption.

The proposed technique is based on the ISD representation presented in our previous work [19]. Using the ISD representation, we can prove that every constant can be represented as a sum of $\{2^m \pm 1, m \in Z\}$ and their shifts as:

$$n = \sum_{m=0}^{\infty} a_m \times 2^{p_m} \times (2^m \pm 1), \forall n \in Z, \exists p_m \in Z, a_m \in \{0,1\} \tag{11}$$

In a circuit, the bit width of every coefficient is finite. We use L to represent the bit width of C_k.

$$C_k = \sum_{m=0}^{L-1} a_{km} \times 2^{p_{km}} \times (2^m \pm 1), p_{km} \in Z, a_{km} \in \{0,1\} \tag{12}$$

Using (12), (3) can be transformed into (13).

$$\begin{aligned} y &= \sum_{k=0}^{K-1} C_k x_k \\ &= \sum_{k=0}^{K-1} [\sum_{m=0}^{L-1} a_{km} \times 2^{p_{km}} \times (2^m \pm 1)] x_k \\ &= \sum_{k=0}^{K-1} \sum_{m=0}^{L-1} [a_{km} \times 2^{p_{km}} \times (2^m \pm 1) \times x_k] \\ p_{km} &\in Z, a_{km} \in \{0,1\}, b_{kn} \in \{0,1\} \end{aligned} \tag{13}$$

Let $c_{km} = a_{km} \times 2^{p_{km}}$. Because $a_{km} \in \{0,1\}$, we can find $c_{km} \in \{0, 2^q\}, q \in Z$. Therefore, the inner-product can be transformed as follows:

$$y = \sum_{k=0}^{K-1} \sum_{m=0}^{L-1} [c_{km} \times (2^m \pm 1) \times x_k], c_{km} \in \{0, 2^q\}, q \in Z \tag{14}$$

For the sake of representation efficiency, (14) is represented in matrix form as follows:

$$\begin{aligned} y &= \mathbf{XCD} \\ &= \begin{bmatrix} x_0 & x_1 & \dots & x_{(K-1)} \end{bmatrix} \times \begin{bmatrix} c_{00} & c_{01} & \dots & c_{0(2L-1)} \\ c_{10} & c_{11} & \dots & c_{1(2L-1)} \\ \dots & \dots & \dots & \dots \\ c_{(K-1)0} & c_{(K-1)1} & \dots & c_{(K-1)(2L-1)} \end{bmatrix} \times \begin{bmatrix} d_0 \\ d_1 \\ \dots \\ d_{(2L-1)} \end{bmatrix} \end{aligned} \tag{15}$$

In (15), the elements in $\mathbf{D}$ are $2^m \pm 1, m \in Z$, which have $2L$ possible values and can be achieved by one adder. Therefore, they can be regarded as intermediate arithmetic operations, and achieved using the vector adders array as Figure 1 (a). $c_{km} \in \{0, 2^q\}, q \in Z$, and the exact value is related to C_k. They can be regarded as the own arithmetic operations in different productions, and achieved using the binary tree adders array as Figure 1 (b). Thus, the inner-product can be accomplished using a vector adders array and several binary tree adders arrays. Based on these structures and pipeline design method, the throughput of digital filter can be improved.

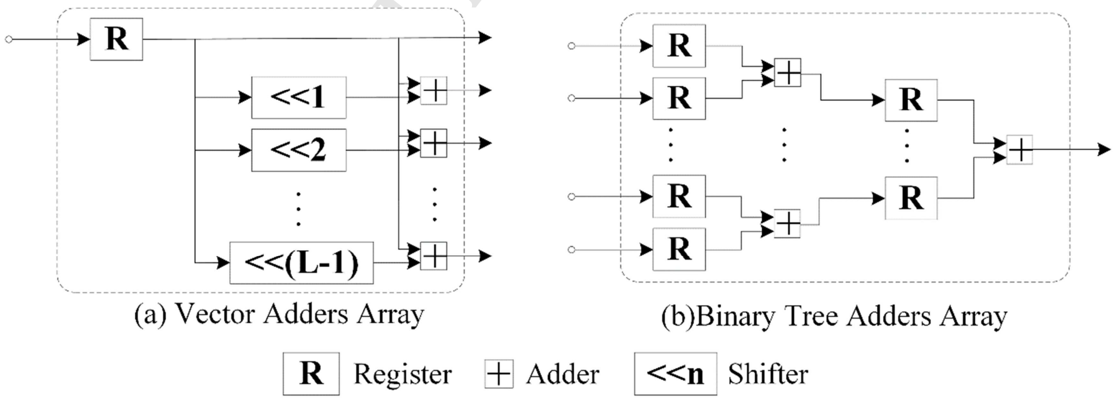

Figure 1. Adders Array Structure in Proposed Method

In the proposed method, different bits of x_k can be processed in parallel. Additionally, the intermediate arithmetic operations $2^m \pm 1$ only have $2L$ possible results. The intermediate arithmetic operations can be calculated efficiently using only addition and shifting operations. Therefore, the proposed method eliminates the need for complex memory structures and efficiently reduces hardware resource consumption. The whole operation needs up to $2LK-1$

additions, which means the proposed method can avoid exponentially increasing hardware resource consumption. And in most cases, the matrix of $\mathbf{C}$ is sparse, so the necessary additions can be further reduced.

IV. Performance Evaluation of the Proposed Approach

In this section, we evaluate the performance of proposed approach through two experiments: 1) a simple finite impulse response (FIR) digital filter and 2) random filters with variant orders. Experiment I is used to verify the performance of proposed method in improving throughput, and Experiment II is to show the improvement of hardware resource consumption reduction when the length of filter order increases.

A. Experiment I: A Common FIR Digital Filter

A common FIR digital filter with four orders is used as an example to demonstrate hardware realization using DA and the proposed method. The parameters are: $K = 4$, $N+1=8$ (7 bits of data and a 1-bit sign), $h_0 = 0.72$, $h_1 = -0.30$, $h_2 = 0.95$ and $h_3 = 0.11$. In this example, all filter coefficients can be left-shifted by 8 bits, meaning $C_0 = 184$, $C_1 = -76$, $C_2 = 243$ and $C_3 = 28$. The memory structure in the traditional DA method stores $2^K = 16$ possible intermediate arithmetic operations, as shown in Table I. The hardware realization is presented in Figure 2.

Table I Values in ROM for DA

Address	Input Code				Memory Contents
	b_{0n}	b_{1n}	b_{2n}	b_{3n}	
0x00	0	0	0	0	0
0x01	0	0	0	1	$C_3 = 28$
0x02	0	0	1	0	$C_2 = 243$
0x03	0	0	1	1	$C_2 + C_3 = 271$
0x04	0	1	0	0	$C_1 = -76$
0x05	0	1	0	1	$C_1 + C_3 = -48$
0x06	0	1	1	0	$C_1 + C_2 = 167$
0x07	0	1	1	1	$C_1 + C_2 + C_3 = 195$
0x08	1	0	0	0	$C_0 = 184$
0x09	1	0	0	1	$C_0 + C_3 = 212$
0x0A	1	0	1	0	$C_0 + C_2 = 427$
0x0B	1	0	1	1	$C_0 + C_2 + C_3 = 455$
0x0C	1	1	0	0	$C_0 + C_1 = 108$
0x0D	1	1	0	1	$C_0 + C_1 + C_3 = 136$
0x0E	1	1	1	0	$C_0 + C_1 + C_2 = 351$
0x0F	1	1	1	1	$C_0 + C_1 + C_2 + C_3 = 379$

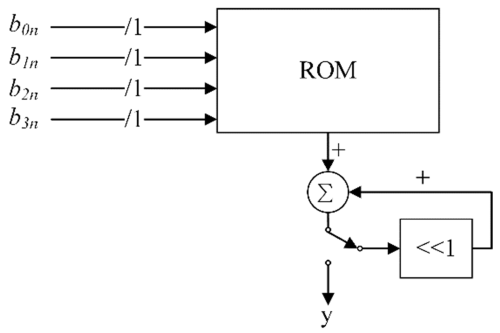

Figure 2. Hardware realization of traditional DA

In MR-DA, the intermediate arithmetic operations have $2^{K-1}=8$ possible values, as shown in Table II. Its hardware realization is presented in Figure 3. In this realization, the memory size is reduced by half, but the controlling and addressing units become more complex than those in direct DA.

Table II Values in ROM for MR-DA

Address	Input Code				Memory Contents
	b_{0n}	b_{1n}	b_{2n}	b_{3n}	
0x00	0	0	0	0	$C_0+C_1+C_2+C_3=379$
0x01	0	0	0	1	$C_0+C_1+C_2-C_3=323$
0x02	0	0	1	0	$C_0+C_1-C_2+C_3=-107$
0x03	0	0	1	1	$C_0+C_1-C_2-C_3=-163$
0x04	0	1	0	0	$C_0-C_1+C_2+C_3=531$
0x05	0	1	0	1	$C_0-C_1+C_2-C_3=475$
0x06	0	1	1	0	$C_0-C_1-C_2+C_3=45$
0x07	0	1	1	1	$C_0-C_1-C_2-C_3=-11$

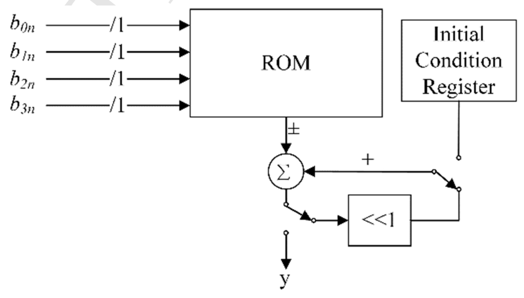

Figure 3. Hardware realization of MR-DA

The inner-product can also be calculated using the hardware realization of 8-BAAT DA with 8-times the memory, as shown in Figure 4.

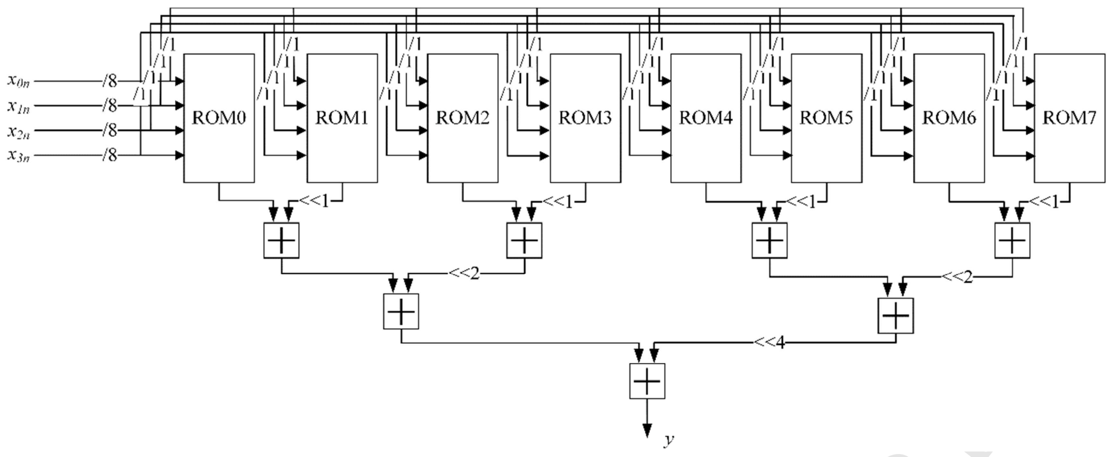

Figure 4. Hardware realization of 8-BAAT DA

In the proposed method, we need to find an efficient **CD** to simplify the circuit. Considering the computational complexity in finding an optimal **CD**, we use a greedy strategy and select the smallest number of intermediate operations first. We find that the set $\{2^1-1, 2^1+1, 2^2+1\}$ and their shifts can represent all the coefficients, so we use it as **C**. We then select the **D** that uses the smallest number of additions among all the possible results.

Thus, in this example, (15) is transformed into (16).

$$y = \mathbf{XCD} = [x_0 \quad x_1 \quad x_2 \quad x_3] \times \begin{bmatrix} -2^3 & 2^6 & 0 \\ -2^6 & -2^2 & 0 \\ 2^8 & -2^0 & -2^1 \\ 2^2 & 2^3 & 0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \qquad (16)$$

Based on (16), the inner-product can be calculated as shown in Figure 5. We find that the proposed method uses only 10 adders/subtractors to implement the digital filter.

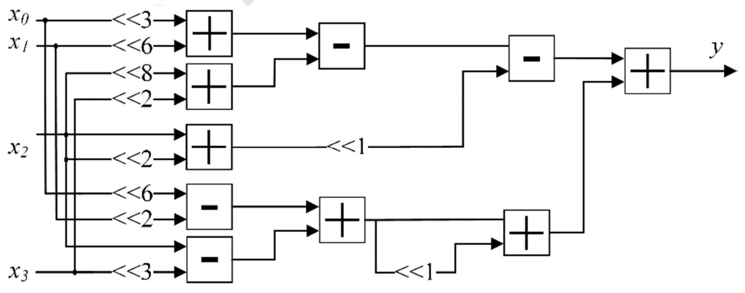

Figure 5. Hardware realization of the proposed method

In order to compare the performance of the various methods, all the four structures were written in a hardware description language and synthesized using the Synopsys Design Compiler for ASIC implementation with the CMOS 90-nm library. The statistics running at 1 GHz are listed in Table III, including necessary cells, operating area, bits processed at a time (BAAT), average clock cycles (ACC) to get one inner-product result and throughput.

Table III Performance Comparison of Four Hardware Realizations for Experiment I

Design	Cells	Area	BAAT	ACC	Throughput
Direct DA	728	4001	1	8	4 Gbps
MR-DA	467	4120	1	8	4 Gbps
8-BAAT DA	755	6888	8	1	32 Gbps
Proposed	514	5544	8	1	32 Gbps

In Table III, we can see that the proposed method for realization performs excellently. It requires the least cells among all the realizations except MR-DA. But both of Direct DA and MR-DA can only process 1 bit of each input at a time with the lowest throughput. The 8-BAAT DA and proposed realizations can circumvent the inherent bit-serial nature and process different bits of x_k in parallel. With pipeline structure, the average clock cycles to get one inner-product result can be reduced to 1 and the throughput increases to 32 Gbps efficiently. Compared with 8-BAAT DA, the proposed realization needs less cells and smaller area. Therefore, the proposed realization can improve the throughput of direct DA and MR-DA. In addition, the necessary cell is less and area is smaller than those in 8-BAAT DA. In this experiment, compared to the 8-BAAT DA, the proposed method reduces the necessary cells and area by approximately 32 and 20%, respectively.

B. Experiment II: Random Filters with Variant Orders

For a more thorough analysis, the performance of proposed method is evaluated on random filters. These filters perform 16-bit quantization with variant orders, which range from 1 to 255, and we test 10 different random filters in each order. The average number of necessary additions is plotted in Figure 6.

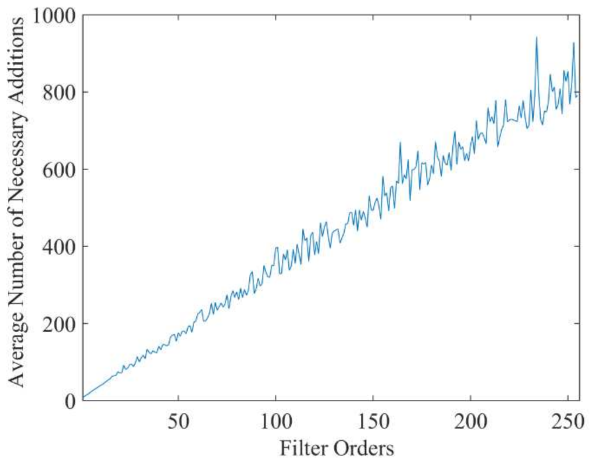

Figure 6. Average number of necessary additions for the proposed method

In Figure 6, we can find that the average number of necessary additions for the proposed method increases linearly. It needs 4 more additions on average when filter order increases by one. However, in the DA-based methods, memory size increases exponentially with variant order. With increasing filter order length, the hardware structure becomes increasingly difficult to synthesize because of the massive memory requirements. For example, when the length of the filter order is equal to 32, the memory size in MR-DA is $2^{(32-1)} = 2{,}147{,}483{,}648$, which is impossible to realize in circuitry. We only synthesize the realization with orders ranging from 1 to 14, which demonstrates the performance of the proposed method. The base-10 logarithms of the average cell numbers in different realizations are presented in Figure 7.

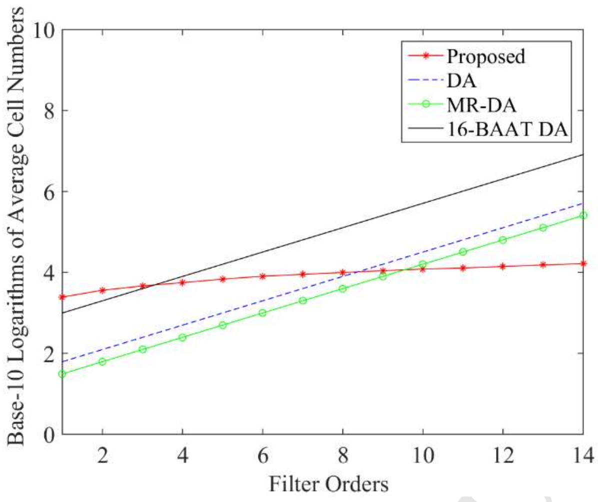

Figure 7. Base-10 logarithms of average cell numbers in different realizations

In Figure 7, one can see that the necessary cells of DA and DA-based realizations increased rapidly. When the length of the filter order is greater than 10, the proposed method requires the least cells among these hardware realizations and it increases linearly with the length of filter order. Therefore, the proposed method can reduce the hardware consumption in the realization of complex filters, and the reduction performance increases with filter length. This indicates that the proposed method can successfully overcome the problem of exponentially increasing hardware resource consumption with an increasing length of filter order.

V. Conclusion

We proposed a parallel arithmetic operation for the hardware realization of digital filters. In the proposed method, filter coefficients are factorized to find basic operations, meaning the inherent bit-serial nature of DA methods can be circumvented. Using these basic operations, the inner-product can be computed in parallel to improve throughput. The number of necessary additions increases linearly with the length of filter order, which allows us to avoid the problem of exponentially increasing hardware resource consumption. In order to verify the performance of the proposed method, the hardware realizations of different methods were evaluated through two experiments. The experimental results demonstrated that the proposed method can accomplish digital filtering in a single cycle on average and that its hardware resource consumption increases linearly, which is more efficient than existing DA-based methods. Therefore, the proposed method can overcome the drawbacks of existing DA and DA-based methods and is suitable for the hardware realization of digital filters.

REFERENCES

[1] A. Peled and B. Liu, “A new approach to the realization of nonrecursive digital filters,” IEEE Transactions on Audio & Electroacoustics, vol. 21, no. 6, pp. 477–484, 1973.

[2] A. Peled and B. Liu, “A new hardware realization of digital filters,” IEEE Transactions on Acoustics Speech & Signal Processing, vol. 22, no. 6, pp. 456–462, 1974.

[3] S. White, “Applications of digital signal processing to control systems,” in Proc. 8th Asilomar Conference on Circuits, Systems, and Computers, 1974, pp. 278–284.

[4] S. Zohar, “A vlsi implementation of a correlator/digital-filter based on distributed arithmetic,” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, no. 1, pp. 156–160, 1989.

[5] S. G. Smith and S. A. White, “Hardware approaches to vector plane rotation,” in International Conference on Acoustics, Speech, and Signal Processing, vol. 4, 1988, pp. 2128–2131.

[6] V. K. Sharma, K. K. Mahapatra, and U. C. Pati, “An efficient distributed arithmetic based vlsi architecture for dct,” in International Conference on Devices and Communications, 2011, pp. 1–5.

[7] K. Xu, “Monolithically integrated si gate-controlled light-emitting device: science and properties,” Journal of Optics, vol. 20, no. 2, 2018.

[8] B. Jalali and Y. Han, “Photonic time-stretched analog-to-digital converter: Fundamental concepts and practical considerations,” Journal of Lightwave Technology, vol. 21, no. 12, pp. 3085–3103, 2003.

[9] Y. H. Chen, T. Y. Chang, and C. Y. Li, “High throughput da-based dct with high accuracy error-compensated adder tree,” IEEE Transactions on Very Large Scale Integration Systems, vol. 19, no. 4, pp. 709–714, 2012.

[10] P. K. Meher, “Hardware-efficient systolization of da-based calculation of finite digital convolution,” IEEE Transactions on Circuits & Systems II Express Briefs, vol. 53, no. 8, pp. 707–711, 2006.

[11] Y. P. Sang and P. K. Meher, “Efficient fpga and asic realizations of a da-based reconfigurable fir digital filter,” IEEE Transactions on Circuits & Systems II Express Briefs, vol. 61, no. 7, pp. 511–515, 2014.

[12] C. S. Vinitha and R. K. Sharma, “Memory-based vlsi architectures for digital filters: A survey,” in IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics Engineering, 2017.

[13] S. A. White, “Applications of distributed arithmetic to digital signal processing: a tutorial review,” IEEE ASSP Magazine, vol. 6, no. 3, pp. 4–19, July 1989.

[14] H. Schroder, “High word-rate digital filters with programmable table look-up,” IEEE Transactions on Circuits and Systems, vol. 24, no. 5, pp. 277–279, 1977.

[15] C. Burrus, “Digital filter realization by distributed arithmetic,” in International Symposium on Circuits and Systems, Munich, 1976.

[16] M. Buttner and H. W. Schuessler, “On structures for the implementation of the distributed arithmetic,” NTZ Communication Journals, vol. 6, 1975.

[17] K. Kammeyer, “Digital filter realization in distributed arithmetic,” in Proc. European Conf. on Circuit Theory and Design, 1976.

[18] K.-P. Yiu, “On sign bit assignment for a vector multiplier,” Proceedings of the IEEE, vol. 64, no. 3, pp. 372–373, 1976.

[19] C. Fan, Y. Niu, G. Shi, F. Li, F. Qi, X. Xie, and D. Jiao, “An improved signed digit representation approach for constant vector multiplication,” IEEE Transactions on Circuits & Systems II Express Briefs, vol. 63, no. 10, pp. 999–1003, 2016.