electronics

MDPI

Article

Scalable Hardware Efficient Architecture for Parallel FIR Filters with Symmetric Coefficients

Jinghao Ye [1], Masao Yanagisawa [2] and Youhua Shi [2,*]

[1] NVIDIA Semiconductor Technology (Shanghai) Co., Ltd., Shanghai 200001, China
[2] Faculty of Science and Engineering, Waseda University, Tokyo 169-8555, Japan
* Correspondence: shi@waseda.jp

Abstract: Symmetric convolutions can be utilized for potential hardware resource reduction. However, they have not been realized in state-of-the-art transposed block FIR designs. Therefore, we explore the feasibility of symmetric convolution in transposed parallel FIRs and propose a scalable hardware efficient parallel architecture. The proposed design inserts delay elements after multipliers for temporal reuse of intermediate tap products. By doing this, the number of required multipliers can be reduced by half. As a result, we can achieve up to 3.2× and 1.64× area efficiency improvements over the modern transposed block method on reconfigurable and fixed designs, respectively. These results confirm the effectiveness of the proposed STB-FIR architecture for hardware-efficient, high-speed signal processing.

Keywords: FIR filter; symmetric transposed FIR; hardware efficient; high-speed signal processing

Citation: Ye, J.; Yanagisawa, M.; Shi, Y. Scalable Hardware Efficient Architecture for Parallel FIR Filters with Symmetric Coefficients. *Electronics* **2022**, *11*, 3272. https://doi.org/10.3390/electronics11203272

Academic Editors: Fei Yu, José V Frances-Villora, Jun Mou and Young-Ho Seo

Received: 31 August 2022
Accepted: 9 October 2022
Published: 11 October 2022

1. Introduction

Finite impulse response (FIR) filter, one primary digital filter, has been widely used in signal processing due to its stability and linear phase characteristics. With the time domain input, x_n, and the filter coefficient, h_m, the corresponding output, y_n, of a T-tap FIR can be obtained as $y_n = \sum_{m=0}^{T-1} h_m \cdot x_{n-m}$, according to the discrete time convolution [1]. Due to the accuracy requirement in frequency domain, T is generally large [2] and consequently, incurs a large silicon area with significant power consumption. Therefore, over the past decades, much research effort has been toward hardware efficient FIR filter implementations.

Because multipliers are generally more expensive than adders in terms of area and power consumption, many previous works have focused on the design of FIR filters with area-efficient multipliers. In some application-specific FIRs, the coefficients can be pre-determined; thus, instead of using the costly general multipliers, several constant-multiplier-based designs (CM) have been proposed [3–8]. These CM-based FIRs, however, are application-specific and only work for a specific coefficient set; therefore, they are not suitable in reconfigurable systems with programmable coefficients for real-time applications, such as adaptive pulse shaping and signal equalization [9]. On the other hand, because FIR filters are widely used in high-throughput multimedia signal processing and cellular wireless communication systems, there are also several parallel FIR implementations, such as fast FIR algorithm (FFA) [10–12] and block FIRs [13–15]. The basic idea of FFA is to break up an FIR filter into several sub-filters using polyphase decomposition so that they can operate in parallel with reduced computation complexity. In an FFA-based FIR filter design, the required number of multipliers can be greatly reduced at the cost of an increase in adders for extra pre-processing and post-processing. Symmetric FFA-based designs have also been proposed in [11,12] with the consideration of symmetric convolutions. Although FFA-based methods can achieve a significant reduction in multipliers, they are only effective for parallel FIR filters with low parallelism. Otherwise, the increased adders will introduce significant area overhead with the increased design complexity. On the other

hand, block FIRs have also been proposed in [13–15] for high-throughput signal processing. Unlike FFA-based designs, block FIRs can be combined with CM-based methods for a specific coefficient set; however, the corresponding hardware resource increases linearly with the degree of parallelism. Therefore, area and power efficiency of the existing parallel FIRs are still the design challenges.

The symmetry of coefficients, which can lead to a significant saving in hardware cost, has not been taken into consideration in the existing block FIR designs yet. Therefore, we explore the feasibility of symmetric convolution in transposed parallel FIRs and propose a symmetric transposed block FIR filter (STB-FIR) architecture for area/power-efficient implementation of block FIR filters in which delay elements are inserted after multipliers for temporal reuse of intermediate tap products.

The remainder of this paper is organized as follows. The proposed STB-FIR architecture is illustrated in Section 2 with the corresponding generalized formulation. Evaluation results are provided in Section 3. Finally, the conclusion is given in Section 4.

2. Proposed STB-FIR

A T-tap digital FIR filter can be generally implemented either in the direct form or in the transposed form, as shown in Figure 1.

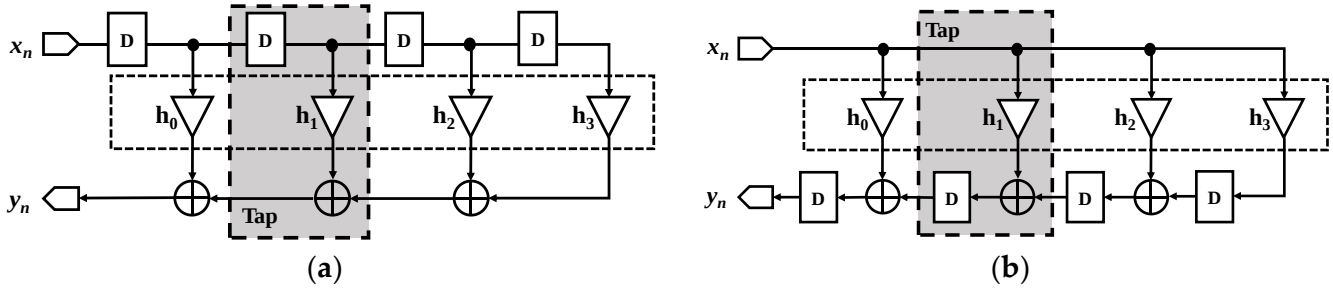

Figure 1. General FIR implementations: (**a**) direct form and (**b**) transposed form.

2.1. Generalized Mathematical Formulation for TB-Based FIRs

For L-parallel processing, the transposed block FIR proposed in [15] takes a block of L new input samples $\{x_n, x_{n-1}, \ldots, x_{n-L+1}\}$ and produces a block of L output samples $\{y_n, y_{n-1}, \ldots, y_{n-L+1}\}$ in each clock cycle. For a T-tap L-parallel transposed block FIR with $T = ML$ and L and M indicate the degree of processing parallelism and the total number of blocks, respectively, the operations can be expressed in matrix form as:

$$\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-L+1} \end{bmatrix} = \begin{bmatrix} x_n & \cdots & x_{n-L+1} & x_{n-L} & \cdots & x_{n-2L+1} & \cdots & x_{n-ML+1} \\ x_{n-1} & \cdots & x_{n-L} & x_{n-L-1} & \cdots & x_{n-2L} & \cdots & x_{n-ML} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ x_{n-j} & \cdots & x_{n-j-L+1} & x_{n-j-L} & \cdots & x_{n-j-2L+1} & \cdots & x_{n-j-ML+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ x_{n-L+1} & \cdots & x_{n-2L+2} & x_{n-2L+1} & \cdots & x_{n-3L+2} & \cdots & x_{n-(M+1)L+2} \end{bmatrix} \cdot \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{L-1} \\ \vdots \\ h_{ML-1} \end{bmatrix} \quad (1)$$

where $\{x_n, x_{n-1}, \ldots, x_{n-L+1}\}$ is the input of the current clock cycle, $\{x_{n-L}, x_{n-L-1}, \ldots, x_{n-2L+1}\}$ represents the input in the previous clock cycle, and $\{x_{n-mL}, x_{n-mL-1}, \ldots, x_{n-(m+1)L+1}\}$ represents the input m clocks before. It is obvious that the output of an FIR is a linear combination of the current input and some previous values.

According to (1), as an L-parallel FIR has L new inputs in each cycle, we define B_m as

$$B_m \equiv \begin{bmatrix} x_{n-mL} & x_{n-mL-1} & \cdots & x_{n-(m+1)L+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-mL-j} & x_{n-mL-(j+1)} & \cdots & x_{n-(m+1)L-(j-1)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-(m+1)L+1} & x_{n-(m+1)L} & \cdots & x_{n-(m+2)L+2} \end{bmatrix} \tag{2}$$

Hence, substituting it into (1), we can obtain

$$\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-L+1} \end{bmatrix} = \begin{bmatrix} B_0 & B_1 & \ldots & B_m & \ldots & B_{M-1} \end{bmatrix} \cdot \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{ML-1} \end{bmatrix} \tag{3}$$

Similarly, the coefficients can also be divided into M blocks as

$$H = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{ML-1} \end{bmatrix} \equiv \begin{bmatrix} H_0 \\ H_1 \\ \vdots \\ H_{M-1} \end{bmatrix} \text{ where } H_m = \begin{bmatrix} h_{mL} \\ h_{mL+1} \\ \vdots \\ h_{(m+1)L-1} \end{bmatrix} \tag{4}$$

Thus, (3) can be further rewritten as

$$\begin{aligned} \begin{bmatrix} y_n \\ \vdots \\ y_{n-j} \\ \vdots \\ y_{n-L+1} \end{bmatrix} &= \begin{bmatrix} B_0 & B_1 & \ldots & B_m & \ldots & B_{M-1} \end{bmatrix} \cdot \begin{bmatrix} H_0 \\ \vdots \\ H_m \\ \vdots \\ H_{M-1} \end{bmatrix} \\ &= B_0 \cdot H_0 + B_1 \cdot H_1 + \ldots + B_{M-1} \cdot H_{M-1} \\ &= \sum_{m=0}^{M-1} B_m \cdot H_m \end{aligned} \tag{5}$$

Here, let the calculation of $B_m \cdot H_m$ be TB_m, called time block in the following.

$$\begin{aligned} TB_m &\equiv B_m \cdot H_m \\ &= \begin{bmatrix} x_{n-mL} & x_{n-mL-1} & \cdots & x_{n-(m+1)L+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-mL-j} & x_{n-mL-(j+1)} & \cdots & x_{n-(m+1)L-(j-1)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-(m+1)L+1} & x_{n-(m+1)L} & \cdots & x_{n-(m+2)L+2} \end{bmatrix} \cdot \begin{bmatrix} h_{mL} \\ h_{mL+1} \\ \vdots \\ h_{(m+1)L-1} \end{bmatrix} \end{aligned} \tag{6}$$

According to (5), a TB-based scalable FIR architecture can be implemented in a transposed form, as shown in Figure 2, where a T-tap L-parallel block FIR filter has M (= T/L) time blocks and can process ML input samples in M clock cycles with all the TBs working simultaneously. It should be noted that the input samples B_m contains the current input and the previous input. Moreover, each time block (TB_m) has the corresponding coefficients, H_m, and the result is added to the output of the neighboring time block (TB'_{m+1}) to generate the output of TB_m and then is sent to the next time block (TB_{m-1}).

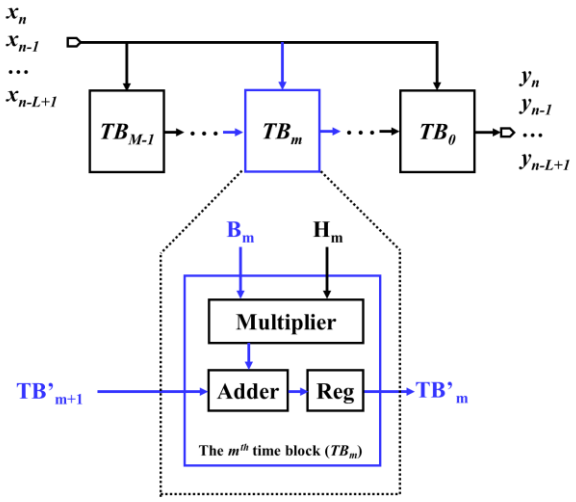

Figure 2. Time-block-based transposed block FIR architecture.

2.2. Proposed STB-FIR Design

For a linear-phase FIR filter, the impulse response can be symmetric ($h_i = h_{T-1-i}$) or anti-symmetric ($h_i = -h_{T-1-i}$). To simplify the explanation, only the symmetric realization with $h_i = h_{T-1-i}$ will be discussed in the following. It is worth noting that the proposed architecture is also able to be applied to anti-symmetric FIR implementations.

As shown in (2), there are $2L-1$ input samples that are multiplied with the corresponding coefficients in each time block. Among the $2L-1$ different input samples, L samples $\{x_{n-mL}, x_{n-mL-1}, \ldots, x_{n-(m+1)L+1}\}$ are the inputs in the current clock cycle while the other $L-1$ ones $\{x_{n-(m+1)L}, x_{n-(m+1)L-1}, \ldots, x_{n-(m+2)L+2}\}$ were obtained in the previous clock cycle. Therefore, totally L^2 different multipliers are required. In [15], delay elements are inserted on the input side to make it possible for the temporary storage of the $L-1$ samples for later calculation. Unfortunately, because $H_i \neq H_{M-1-i}$ in the transposed block form, the symmetry of coefficients cannot be easily realized.

To explore the feasibility of symmetric convolution in parallel block FIR filters, a hardware efficient symmetric transposed block FIR architecture (STB-FIR) is proposed. In STB-FIR, delay elements are inserted after multipliers; thus, temporal reuse of the intermediate tap products becomes possible. Consequently, half of the multipliers can be saved at the cost of increased registers.

For a T-tap L-parallel transposed FIR with symmetric coefficients where $T = ML$, there are two cases (i.e., M is odd or even) that should be considered in the proposed STB-FIR design method, as shown in Figure 3.

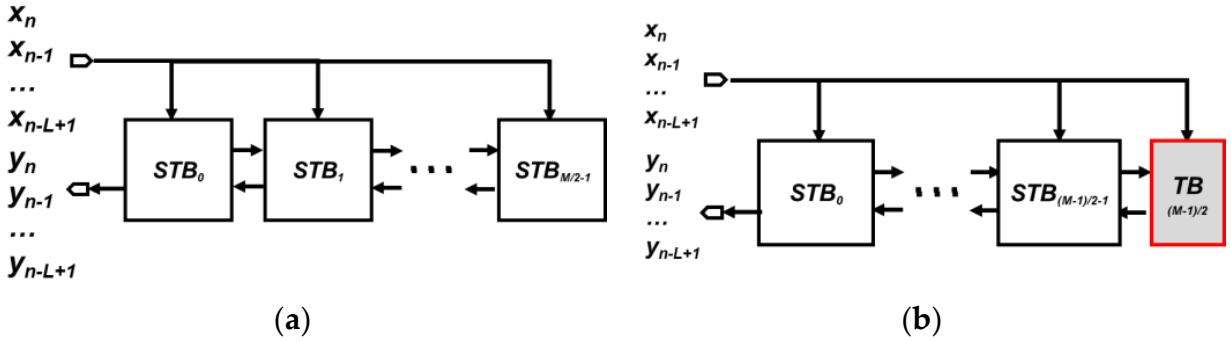

Figure 3. Proposed STB-FIR structure with two cases: (**a**) TB-based fully symmetric folded structure when M is even and (**b**) non-symmetric folded structure when M is odd.

2.2.1. Case 1: M Is Even

A linear FIR filter that falls into this category has an even tap (i.e., T is even), an even number of TBs (i.e., M is even), and symmetric coefficients (i.e., $h_i = h_{T-1-i}$). Without loss of generality, let us consider two symmetric TBs (TB_m and TB_{M-1-m}) in a TB-based symmetric FIR.

For the time block, TB_m, shown in (6), we can divide it into two terms as below, and each of them can be implemented, as shown in Figure 4a,b, respectively.

$$\begin{aligned} TB_m &= B_m \cdot H_m \\ &= \begin{bmatrix} x_{n-mL} & x_{n-mL-1} & \cdots & x_{n-(m+1)L+1} \\ x_{n-mL-1} & \vdots & \iddots & 0 \\ \vdots & x_{n-(m+1)L+1} & \iddots & \vdots \\ x_{n-(m+1)L+1} & 0 & \cdots & 0 \end{bmatrix} \cdot \begin{bmatrix} h_{mL} \\ h_{mL+1} \\ \vdots \\ h_{(m+1)L-1} \end{bmatrix} \\ &+ \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & \vdots & \iddots & x_{n-(m+1)L} \\ \vdots & 0 & \iddots & \vdots \\ 0 & x_{n-(m+1)L} & \cdots & x_{n-(m+2)L+2} \end{bmatrix} \cdot \begin{bmatrix} h_{mL} \\ h_{mL+1} \\ \vdots \\ h_{(m+1)L-1} \end{bmatrix} \end{aligned} \tag{7}$$

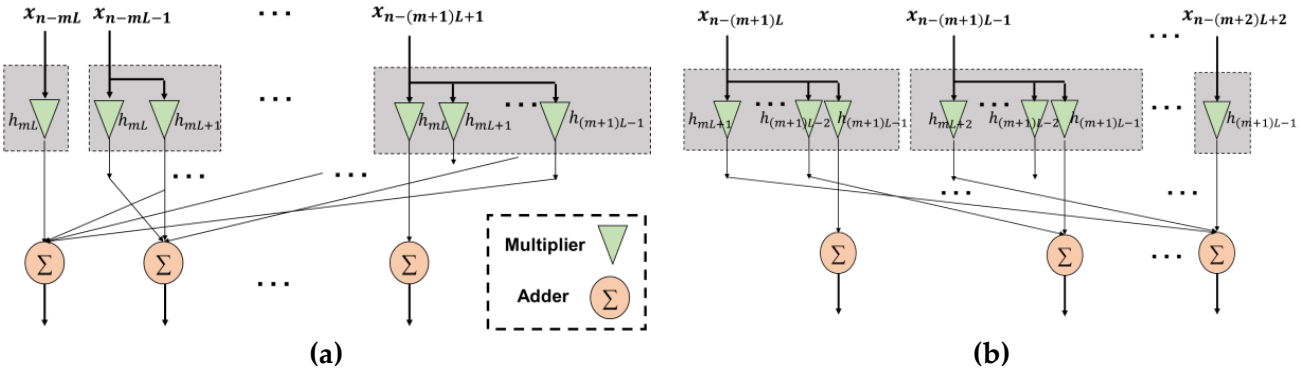

Figure 4. Implementation of the time block, TB_m, with the corresponding two terms in (7): (**a**) the form term and (**b**) the latter term.

Here, it should be mentioned that there are $2L-1$ data inputs that are multiplied with the corresponding coefficients (H_m) in each time block (TB_m). Among the $2L-1$ different input samples, if the L samples $\{x_{n-mL}, x_{n-mL-1}, \ldots, x_{n-(m+1)L+1}\}$ are the inputs in the current clock cycle, the other $L-1$ ones $\{x_{n-(m+1)L}, x_{n-(m+1)L-1}, \ldots, x_{n-(m+2)L+2}\}$ are obtained in the previous clock cycle. Since in transposed form, every input sample should be multiplied with all the coefficients to generate the intermediate tap products, we can conduct the multiplication firstly and then save the products in registers for later addition. By doing this, intermediate products can be reused at the cost of several additional registers, while the required multipliers are reduced by half. As a result, the two parts shown in Figure 4 can be combined into one circuit, as shown in Figure 5, where the current L input samples $\{x_{n-mL}, x_{n-mL-1}, \ldots, x_{n-(m+1)L+1}\}$ are multiplied with all the coefficients firstly in which some of the products will be directly delivered for addition, while some of them are firstly saved into the registers and then sent to the adder.

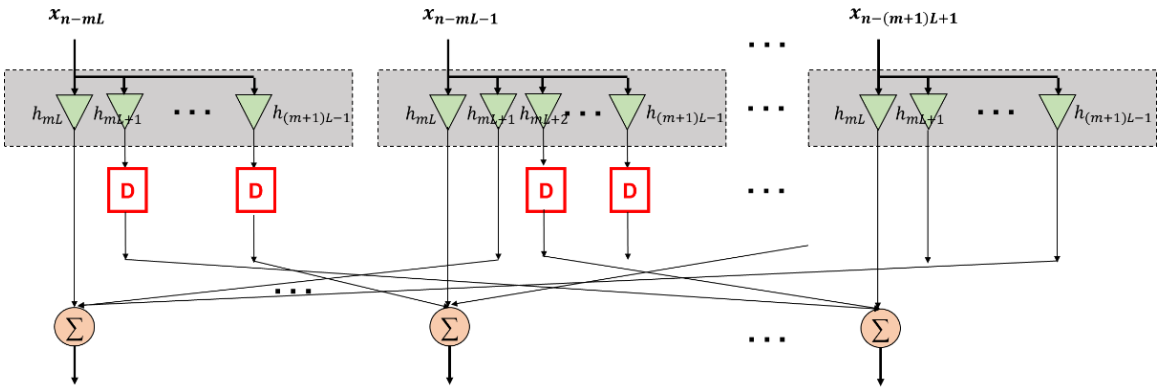

Figure 5. Combined implementation of the two circuits shown in Figure 4.

Furthermore, in a TB-based symmetric FIR (i.e., $h_i = h_{T-1-i}$), the coefficients in the two symmetric TBs (TB_m and TB_{M-1-m}) are H_m and H_{M-1-m}, respectively, and then we have

$$H_{M-1-m} = \begin{bmatrix} h_{(M-1-m)L} \\ h_{(M-1-m)L+1} \\ \vdots \\ h_{[(M-1-m)+1]L-2} \\ h_{[(M-1-m)+1]L-1} \end{bmatrix} = \begin{bmatrix} h_{(m+1)L-1} \\ h_{(m+1)L-2} \\ \vdots \\ h_{mL+1} \\ h_{mL} \end{bmatrix} = E^{-1} \cdot H_m \tag{8}$$

where $E^{-1} = \begin{bmatrix} 0 & \cdots & 1 \\ \vdots & \iddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix}$ providing the vector of coefficients in reverse order, and

$$\begin{aligned} TB_{M-1-m} &= \begin{bmatrix} x_{n-(M-1-m)L} & x_{n-(M-1-m)L-1} & \cdots & x_{n-(M-1-m)L-(L-1)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-(M-1-m)L-j} & x_{n-(M-1-m)L-(j+1)} & \cdots & x_{n-(M-1-m)L-(L+j-1)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-(M-1-m)L-(L-1)} & x_{n-(M-1-m)L-L} & \cdots & x_{n-(M-1-m)L-(2L-1)} \end{bmatrix} \cdot E^{-1} \cdot H_m \\ &= \begin{bmatrix} x_{n-(M-1-m)L} & x_{n-(M-1-m)L-1} & \cdots & x_{n-(M-1-m)L-(L-1)} \\ x_{n-(M-1-m)L-1} & \vdots & \iddots & 0 \\ \vdots & x_{n-(M-1-m)L-(L-1)} & \iddots & \vdots \\ x_{n-(M-1-m)L-(L-1)} & 0 & \cdots & 0 \end{bmatrix} \cdot E^{-1} \cdot H_m \\ &+ \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & \vdots & \iddots & x_{n-(M-1-m)L-L} \\ \vdots & 0 & \iddots & \vdots \\ 0 & x_{n-(M-1-m)L-L} & \cdots & x_{n-(M-1-m)L-(2L-1)} \end{bmatrix} \cdot E^{-1} \cdot H_m \end{aligned} \tag{9}$$

Although $H_m \neq H_{M-1-m}$, they consist of the same L separate coefficients as $\{h_{mL}, h_{mL+1}, \ldots, h_{(m+1)L-1}\}$. Since, in transposed form, every input sample should be multiplied with all the coefficients to generate the intermediate tap products, we can implement each pair of symmetric TBs (TB_m and TB_{M-1-m}) as one symmetric time block (STB_m) to take advantage of the same separate coefficients for tap product reuse.

The proposed STB-FIR structure with an even M is shown in Figure 3a, which only consists of the basic STB units, and the detailed STB design is shown in Figure 6. For each STB, it has L data inputs (i.e., x_n, x_{n-1}, x_{n-2}, $\ldots, x_{n-L+1}$) and L coefficients (H_k) where $H_k = \begin{bmatrix} h_{kL} & h_{kL+1} & \ldots & h_{(k+1)L-1} \end{bmatrix}^T$. It also accepts data from the neighboring STBs, performs the sum operations, and then sends the results to the corresponding two neighboring STBs.

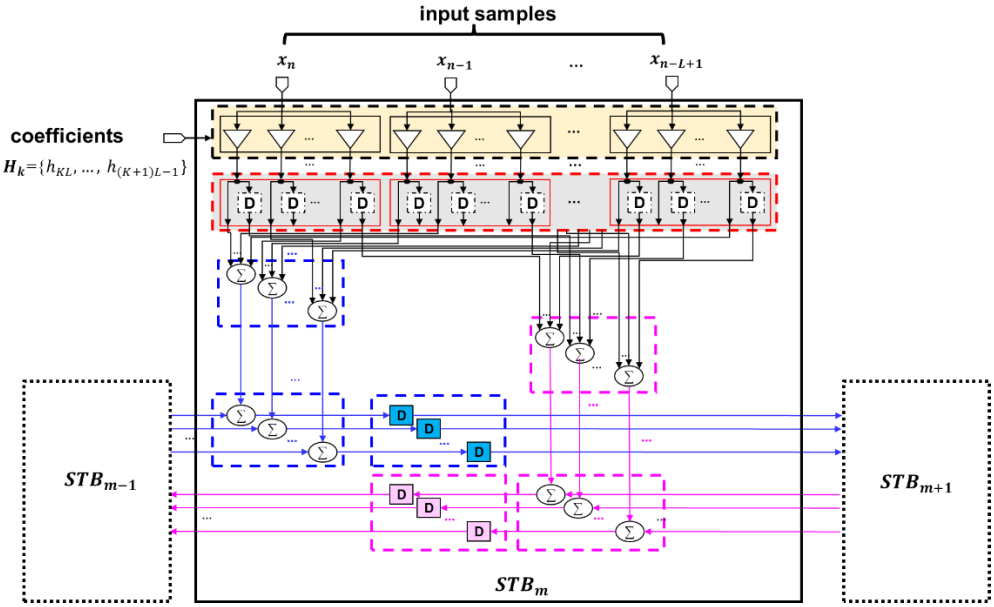

Figure 6. STB design in the proposed STB-FIR.

The proposed STB design can be viewed as the combination of two TBs, while the total number of multipliers is reduced by half when compared with the implementation of using two separate TBs. Meanwhile, the number of adders is kept the same as the existing transposed block FIR [15]. Therefore, the number of multipliers can be reduced at the cost of increased delay elements in STB-FIR. Since multipliers are much more expensive in area and power consumption than registers, STB-FIR can achieve significant area and power savings when compared with the existing transposed block FIRs.

2.2.2. Case 2: M Is Odd

When the number of time blocks (M) is odd, the FIR structure cannot be fully TB-based folded, and a half-STB unit ($TB_{(M-1)/2}$) is required, as shown in Figure 3b. Fortunately, due to the symmetry of coefficients (i.e., $h_i = h_{T-1-i}$), we have

$$H_{\frac{M-1}{2}} = \begin{bmatrix} h_{\frac{M-1}{2}\cdot L} \\ \vdots \\ h_{\frac{M-1}{2}\cdot L+j} \\ \vdots \\ h_{\frac{M-1}{2}\cdot L+L-1} \end{bmatrix} = \begin{bmatrix} h_{\frac{M-1}{2}\cdot L} \\ \vdots \\ \vdots \\ h_{\frac{M-1}{2}\cdot L+1} \\ h_{\frac{M-1}{2}\cdot L} \end{bmatrix} \tag{10}$$

Thus, $TB_{(M-1)/2}$ can be calculated as

$$TB_{\frac{M-1}{2}} = B_{\frac{M-1}{2}} \cdot H_{\frac{M-1}{2}} = B_{\frac{M-1}{2}} \cdot \begin{bmatrix} h_{\frac{M-1}{2}\cdot L} \\ \vdots \\ \vdots \\ h_{\frac{M-1}{2}\cdot L+1} \\ h_{\frac{M-1}{2}\cdot L} \end{bmatrix} \tag{11}$$

Due to the symmetry in $H_{M-1/2}$, $TB_{M-1/2}$ can be realized using the STB-like unit design only with the length changed from L to $\left\lceil \frac{L}{2} \right\rceil$.

For better illustration, Figure 7 gives the example designs of the proposed STB-FIR and the existing transposed block FIR [15] for two 6-tap transposed FIRs with different degrees

of parallelism (L). For the 6-tap 3-parallel transposed FIR (T = 6 and L = 3), because M equals to 2, it is TB-based fully symmetric folded, and the corresponding implementation of the proposed STB-FIR is shown in Figure 7a. Because $L = 3$, three data inputs (i.e., x_n, x_{n-1}, and x_{n-2}) are sent to the STB in each clock. When compared with the transposed block FIR [15], the number of multipliers is reduced from 18 to 9 with the same number of adders, while the number of registers is increased by 3. On the other hand, for the 6-tap 2-parallel transposed FIR (T = 6 and L = 2), as $M = 3$, it is not TB-based fully symmetric folded and then a half-STB unit is required, as shown in Figure 7b. Because $L = 2$, two input samples (i.e., x_n and x_{n-1}) are applied in each clock cycle. The STB and the half-STB units work in a similar way as obtaining the data from the neighboring units, performing the sum operations, and then sending the results to the corresponding neighboring units. Finally, two outputs (i.e., y_n and y_{n-1}) are generated in each clock cycle. When compared with that of [15], the number of multipliers is reduced from 12 to 6 with the same number of adders, and the number of registers is only increased by 2 in STB-FIR.

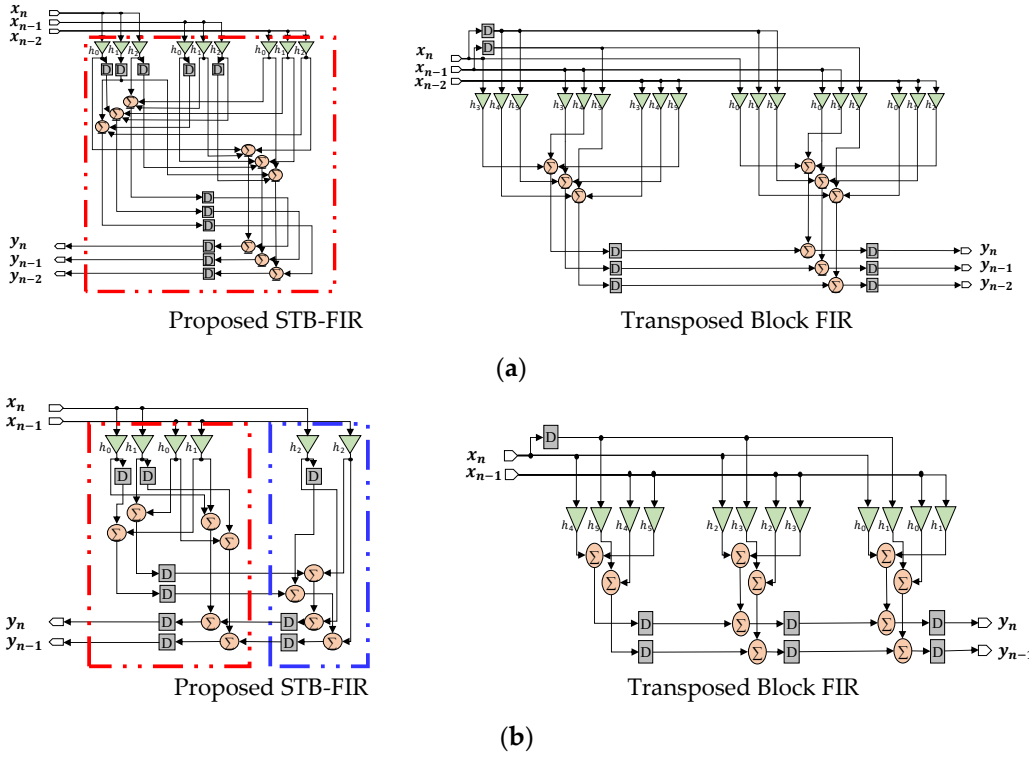

Figure 7. Comparisons of parallel FIR implementations (proposed STB-FIR vs. transposed block FIR). (**a**) Implementations of a 6-tap 3-parallel FIR (T = 6, L = 3 and M = 2) and (**b**) implementations of a 6-tap 2-parallel FIR (T = 6, L = 2 and M = 3).

From this simple example, it can be observed that, unlike the existing transposed block FIR structure [15] in which delay elements are inserted on the input side, the proposed STB-FIR inserts delay elements after multipliers for temporal reuse of tap products. By doing this, the costly multiplier can be reduced by half at the cost of slightly increased low-cost registers.

3. Evaluation Results and Comparisons

To examine the effectiveness of the proposed STB-FIR architecture, implementation results of various FIR filters are provided and compared with the existing works. In our work, all the designs are implemented in ROHM 180 nm CMOS process technology, and the post-synthesis simulation was conducted on the netlist for timing and power evaluation.

3.1. Comparison of Hardware Complexity

As the general FIR structure, the overall hardware complexity of the proposed STB-FIR is compared with the transposed-based block FIR [15], DA-based approach [13] and FFA-based L-parallel structure [1] in Table 1.

Table 1. Comparisons of various parallel FIR architectures.

Architecture.	No. of Multipliers	No. of Adders	No. of Registers
Transposed block	TL	$L(T-1)$	$T+L-1$
DA	TL	$L(T-1)$	$T+L-1$
L-Parallel FFA	$\frac{T}{\prod_{i=1}^{r} L_i} \prod_{i=1}^{r} M_i$	$A_1 \prod_{i=1}^{r} L_i + \sum_{i=2}^{r} \left(A_i \left(\prod_{j=i+1}^{r} L_j \right) \left(\prod_{k=1}^{i-1} M_k \right) \right) + \left(\prod_{i=1}^{r} M_i \right) \left(\frac{T}{\prod_{i=1}^{r} L_i} - 1 \right)$	$\frac{T}{L} \left(R_1 \prod_{i=2}^{r} M_i + R_r \right) + T$
Proposed STB-FIR	$T/2L$	$L(T-1)$	$\frac{T(3L-2)}{8} + T$

* In DA, the number of multipliers indicates the required LUT-based multiplier blocks. * In L-Parallel FFA, $L = \prod_{i=1}^{r} L_i$, and M_i, R_i , and A_i indicate the number of multipliers, registers, and adders of the *ith* parallel FFA basic unit (L_i), respectively. * For the registers in STB-FIR, $\frac{T(3L-2)}{8}$ is an approximate value depending on M and L.

A canonical T-tap FIR filter consists of T multipliers, T–1 adders, and T registers which depends on the tap number (T) for specific accuracy requirement. Therefore, a straightforward implementation of a T-tap L-parallel FIR design, L times hardware resources are required. The transposed block structure proposed in [15] involves TL multipliers, $L\ (T-1)$ adders and $T+L-1$ registers in which $L-1$ registers are inserted at the input side for the storage of $L-1$ samples for later calculation. The required hardware resource of the distributed arithmetic (DA)-based FIR structure [13] is estimated according to the results shown in [15] for comparison purpose. Here, it should be noted that DA [13] was implemented in the direct form, while transposed block [12] was in the transposed form. As for the FFA-based method, the required hardware resource is formulated according to the analysis, as shown in [1,11,12]. As illustrated above, the proposed STB-FIR in total involves $T/2{\cdot}L$ multipliers, $L(T-1)$ adders, and $\frac{T(3L-2)}{8} + T$ registers, among which $\frac{T(3L-2)}{8}$ registers are approximately used as the delay elements for intermediate product sharing, and the other T registers are required for the output storage of the adder trees.

When compared with DA-based FIR [13] and the transposed block FIR [15], the number of multipliers in STB-FIR can be reduced by half while with the same number of adders, which indicates the promising area savings in STB-FIR. The cost of this multiplier reduction is a slightly increased number of low-cost registers which are used to store the intermediate tap products for temporal reuse. According to our analysis, as L and T increase, the ratio of saved multipliers by the increased registers will get close to $4/3$, which indicates that four multipliers can be saved at the cost of three additional registers. Because the area and power of a general multiplier is much larger than the corresponding number of registers, great area and power savings can be achieved in the proposed STB-FIR.

3.2. Comparison of Reconfigurable FIR Implementations

For evaluation and comparison purposes, reconfigurable FIRs in 8, 16, 24, 32, 64, and 128 taps with various degrees of processing parallelism (i.e., $L = 2$, 4, and 8) are implemented by using STB-FIR and the existing transposed block FIR [15], and the corresponding synthesis results are shown in Table 2.

Table 2. Comparisons of various parallel FIR architectures.

Tap	FIR Structure		No. of Multipliers	No. of Adders	No. of Registers	Sampling Freq. (MHz)	Area (um^2)	Area Saving (%)	Power (mw)	Power Saving (%)
8	Parallel L = 2	Transposed block	16	14	9	207.04	203,380	39.35	12.47	34.40
		Proposed STB-FIR	8	14	10	206.83	123,353		8.18	
	Parallel L = 4	Transposed block	32	28	11	382.78	383,343	39.97	21.72	34.16
		Proposed STB-FIR	16	28	18	382.41	230,127		14.30	
	Parallel L = 8	Transposed block	64	56	15	735.97	731,514	37.39	42.17	35.83
		Proposed STB-FIR	32	56	30	733.27	457,993		27.06	
16	Parallel L = 2	Transposed block	32	30	17	207.04	410,745	39.28	26.29	37.31
		Proposed STB-FIR	16	30	24	206.83	249,384		16.48	
	Parallel L = 4	Transposed block	64	60	19	382.78	769,615	39.49	44.72	36.67
		Proposed STB-FIR	32	60	36	382.41	465,661		28.32	
	Parallel L = 8	Transposed block	128	120	23	717.49	1,474,409	39.03	88.39	41.23
		Proposed STB-FIR	64	120	60	732.60	898,994		51.95	
24	Parallel L = 2	Transposed block	48	46	25	207.04	618,293	39.66	34.96	36.13
		Proposed STB-FIR	24	46	36	206.83	373,099		22.33	
	Parallel L = 4	Transposed block	96	92	27	382.78	1,155,887	39.07	64.99	36.42
		Proposed STB-FIR	48	92	54	382.41	704,290		41.32	
	Parallel L = 8	Transposed block	192	184	31	733.27	2,217,483	38.81	120.65	32.95
		Proposed STB-FIR	96	184	90	732.60	1,356,819		80.90	
32	Parallel L = 2	Transposed block	64	62	33	207.04	827,379	39.62	50.40	32.34
		Proposed STB-FIR	32	62	48	206.83	499,542		34.10	
	Parallel L = 4	Transposed block	128	124	35	382.78	1,542,082	39.26	88.22	33.51
		Proposed STB-FIR	64	124	72	382.41	936,727		58.66	
	Parallel L = 8	Transposed block	256	248	39	733.27	2,963,939	39.00	159.96	33.09
		Proposed STB-FIR	128	248	120	732.60	1,808,023		107.03	

Table 2. *Cont.*

Tap	FIR Structure		No. of Multipliers	No. of Adders	No. of Registers	Sampling Freq. (MHz)	Area (um^2)	Area Saving (%)	Power (mw)	Power Saving (%)
64	Parallel L = 2	Transposed block	128	126	65	207.04	1,613,977	39.29	99.83	36.44
		Proposed STB-FIR	64	126	96	206.83	979,802		63.45	
	Parallel L = 4	Transposed block	256	252	67	382.78	3,087,828	39.30	177.32	34.97
		Proposed STB-FIR	128	252	144	382.41	1,874,254		115.32	
	Parallel L = 8	Transposed block	512	504	71	733.27	5,932,523	38.87	322.64	33.07
		Proposed STB-FIR	256	504	240	732.60	3,626,413		215.93	
128	Parallel L = 2	Transposed block	256	254	129	207.04	3,272,258	38.90	208.11	36.97
		Proposed STB-FIR	128	254	192	206.83	1,999,407		131.18	
	Parallel L = 4	Transposed block	512	508	131	382.78	6,178,507	39.12	356.05	34.70
		Proposed STB-FIR	256	508	288	382.41	3,761,321		232.5	
	Parallel L = 8	Transposed block	1024	1016	135	733.27	11,872,904	38.82	648.16	32.76
		Proposed STB-FIR	512	1016	480	732.60	7,263,638		435.81	

In [15], Mohanty et.al have shown that their transposed block FIR designs outperform the existing DA-based method [13]; therefore, we implemented their work as the state-of-the-art design, and the corresponding results are presented in Table 2 for comparison. The results are obtained using the logic synthesis tools, and the power consumption is extracted based on simulation of synthesis results with back-annotation of toggling activity where uniformly distributed sample input values are applied. In the table, the sample frequency is calculated as the degree of parallelism (L)/minimum clock period (MCP). Therefore, it is obvious that the *L*-parallel FIR designs, including both the proposed method and the existing method, can improve the sampling frequency over the baseline non-parallel FIR design at the cost of increased silicon area. On average, the proposed STB-FIR can achieve 39.12% area savings and 35.16% power consumption reduction for all the 18 FIRs when compared with the existing transposed block-based designs [15].

As the structure of reconfigurable FIR filters is very regular, the measurement of area efficiency per tap (*AE*) is introduced as

$$AE = \frac{Tap \times Parallelism}{Area \times MCP} \tag{12}$$

where, *Tap* is the tap number of the FIR, *Parallelism* indicates the degree of processing parallelism (i.e., *L* in the proposed STB-FIR), and *MCP* is the minimum clock period. The normalized *AE* results are given in Figure 8 in which the baseline design has the parallelism of 1, and STB-FIR can achieve up to 3.2× *AE* improvements.

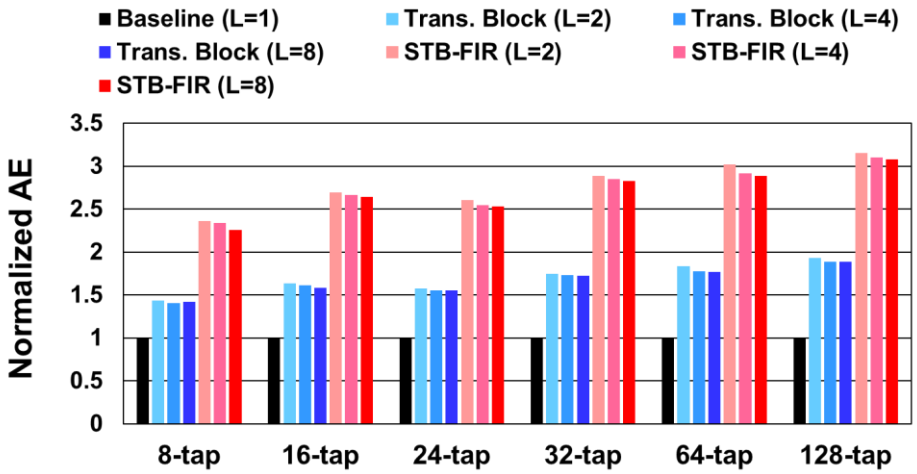

Figure 8. Normalized AE results of various reconfigurable FIR implementations.

3.3. Comparison of Fixed FIR Implementations

As for fixed FIR filters in which the coefficients are pre-determined, by referring to [16], a 105-tap FIR with $L = 3$ and a 60-tap FIR with $L = 2,\ 3$, and 4 are implemented in which the CSE multiplier [4] is adopted for area saving.

For the 105-tap 3-parallel FIR, STB-FIR can achieve 13.08% area saving and 23.05% power reduction when compared with the fixed transposed block FIR [15]. On the other hand, when compared with the fixed symmetric FFA designs [11,12], STB-FIR can achieve up to 31.5% and 40.5% sampling frequency improvement for the 105-tap 3-parallel FIR and the 60-tap 2-parallel one, respectively.

Without loss of generality, the normalized area efficiency comparison is presented in Figure 9 in which the baseline is the FFA-based design in [1]. For the 105-tap 3-parallel FIR, the proposed STB-FIR architecture can achieve 1.64× and 1.20× AE improvements over the transposed block FIR [15] and the fixed symmetric FFA design [11], respectively. Moreover, for the 60-tap FIR, the proposed STB-FIR filter design can achieve up to 1.29× AE improvement over the existing FFA designs.

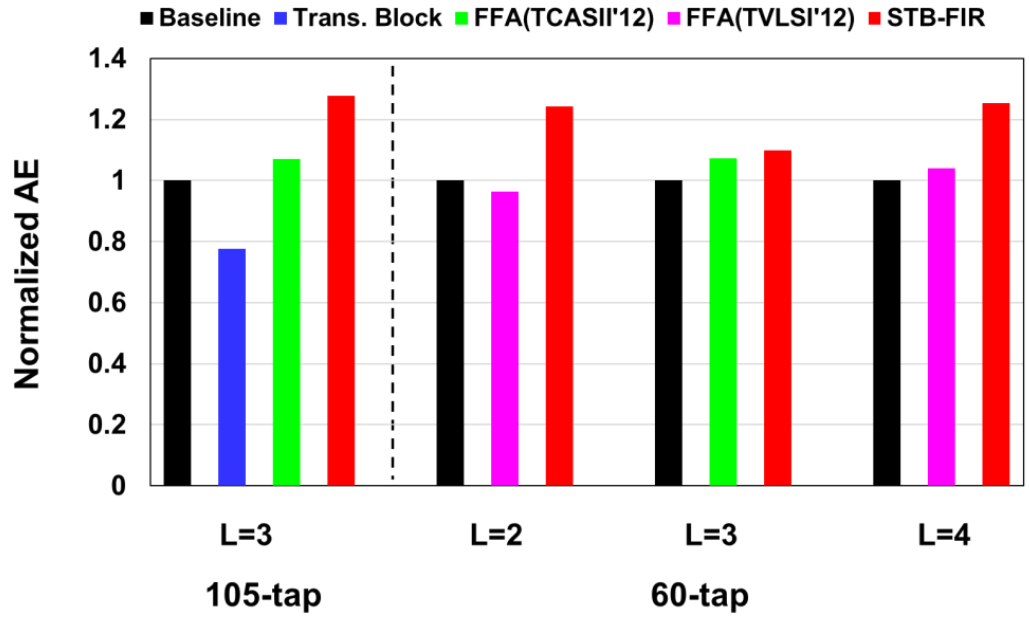

Figure 9. Normalized AE results of various fixed FIR implementations.

4. Conclusions

The feasibility of a symmetric transposed block FIR filter is explored in this paper by taking advantage of the symmetric coefficients for area-power efficient implementation. In the proposed STB-FIR architecture, using registers to save intermediate tap products for temporal reuse makes it possible to realize hardware-efficient symmetric transposed block FIR filters. The evaluation results show that compared with the state-of-the-art reconfigurable architecture [15], the proposed STB-FIR architecture can achieve up to 39.97% and 41.23% area saving and power reduction, respectively. On the other hand, compared with the existing symmetric FFA designs [11,12], the proposed STB-FIR architecture can achieve up to 31.5% and 40.5% sampling frequency improvement and 1.20× AE improvement as well for the fixed FIR implementations. These results clearly illustrate the efficiency of the proposed STB-FIR architecture and confirm that STB-FIR can be applicable to both reconfigurable and fixed FIR implementations for area-power efficient high-speed signal processing. On the other hand, the optimization of multipliers will result in an increased importance of adder trees, especially in fixed FIRs. Therefore, further optimization of adder tree implementation will be one of our future works.

Author Contributions: Conceptualization, J.Y. and Y.S.; methodology, J.Y.; validation, J.Y., M.Y. and Y.S.; data curation, J.Y.; writing—original draft preparation, Y.S.; writing—review and editing, J.Y., M.Y. and Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by Waseda University Grant for Special Research Projects (Project number: 2021C-147).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All the necessary data are included in the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Parhi, K.K. *VLSI Digital Signal Processing Systems: Design and Implementation*; Wiley: New York, NY, USA, 1999.
2. Mirchandani, G.; Zinser, R.L.; Evans, J.B. A new adaptive noise cancellation scheme in the presence of crosstalk [speech signals]. *IEEE Trans. Circuits Syst. II Analog. Digit. Signal Process.* **1995**, *39*, 681–694. [CrossRef]
3. Dempster, A.G.; Macleod, M.D. Use of minimum-adder multiplier blocks in FIR digital filters. *IEEE Trans. Circuits Syst. II Analog. Digit. Signal Process.* **1995**, *42*, 569–577. [CrossRef]
4. Mahesh, R.; Vinod, A.P. A new common subexpression elimination algorithm for realizing low-complexity higher order digital filters. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2008**, *27*, 217–229. [CrossRef]
5. Lou, X.; Yu, Y.J.; Meher, P.K. Fine-grained critical path analysis and optimization for area-time efficient realization of multiple constant multiplications. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2015**, *62*, 863–872. [CrossRef]
6. Meidani, M.; Mashoufi, B. Introducing new algorithms for realizing an FIR filter with less hardware in order to eliminate power line interference from the ECG signal. *IET J. Signal Process.* **2016**, *10*, 709–716. [CrossRef]
7. Ye, J.; Togawa, N.; Yanagisawa, M.; Shi, Y. A low cost and high speed CSD-based symmetric transpose block FIR implementation. In Proceedings of the IEEE International Conference on ASIC (ASICON), Guiyang, China, 25–28 October 2017.
8. Ye, J.; Togawa, N.; Yanagisawa, M.; Shi, Y. Static error analysis and optimization of faithfully truncated adders for area-power efficient FIR designs. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, 26–29 May 2019.
9. Park, J.; Jeong, W.; Meimand, H.M.; Wang, Y.; Choo, H.; Roy, K. Computation sharing programmable FIR filter for low-power and high-performance applications. *IEEE J. Solid-State Circuits* **2004**, *39*, 348–357. [CrossRef]
10. Parker, D.A.; Parhi, K.K. Low-area/power parallel FIR digital filter implementations. *J. VLSI Signal Process. Syst.* **1997**, *17*, 75–92. [CrossRef]
11. Tsao, Y.; Choi, K. Area-efficient parallel FIR digital filter structures for symmetric convolutions based on fast FIR algorithm. *IEEE Trans. Very Large Scale Integr. Syst.* **2012**, *20*, 366–371. [CrossRef]
12. Tsao, Y.; Choi, K. Area-efficient VLSI implementation for parallel linear-phase FIR digital filters of odd length based on fast FIR algorithm. *IEEE Trans. Circuits Syst. II Express Briefs.* **2012**, *59*, 371–375. [CrossRef]
13. Mohanty, B.K.; Meher, P.K. A high-performance energy-efficient architecture for FIR adaptive filter based on new distributed arithmetic formulation of block LMS algorithm. *IEEE Trans. Signal Process.* **2013**, *61*, 921–932. [CrossRef]

14. Mohanty, B.K.; Meher, P.K.; Al-Maadeed, S.; Amira, A. Memory footprint reduction for power-efficient realization of 2-D finite impulse response filters. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2014**, *61*, 120–133. [CrossRef]
15. Mohanty, B.K.; Meher, P.K. A high performance FIR filter architecture for fixed and reconfigurable applications. *IEEE Trans. Very Large Scale Integr. Syst.* **2016**, *24*, 444–452. [CrossRef]
16. Shahein, A.; Zhang, Q.; Lotze, N.; Manoli, Y. A novel hybrid monotonic local search algorithm for FIR filter coefficients optimization. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2011**, *59*, 616–627. [CrossRef]