Computer Assisted Methods in Engineering and Science
30(1): 89–103, 2023, doi: 10.24423/cames.538

This article belongs to the *Special Issue on Information Theory with Artificial Intelligence in 5G/6G Networks* edited by Dr. A.K. Budati, Dr. M.K. Hasan, Dr. R. Arthi, Dr. K. Thangavel

Realization of Multiplexer Logic-Based 2-D Block FIR Filter Using Distributed Arithmetic

Ch. Pratyusha CHOWDARI[1)*], J. Beatrice SEVENTLINE[2)]

[1)] *Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad, India*

[2)] *Department of Electronics and Communication Engineering, GITAM University, Visakhapatnam, India; e-mail: seventline.joseph@gitam.edu*

[*] *Corresponding Author e-mail: pratyushachowdarich@gmail.com*

This paper presents a novel systolic two-dimensional (2D) block finite impulse response (FIR) filter architecture using a distributed arithmetic (DA)-based multiplexer look-up table (DA-MUX-LUT). The proposed DA-MUX-LUT architecture computes the instantaneous partial-product using the bit vector. The switching-based LUT replaces memory-based structures and reduces hardware complexity. Block processing allows memory reuse, which reduces the number of registers to store the previous input samples. Parallel adders are substituted by a modified carry look-ahead adder (MCLA), which minimizes the delay. Moreover, a resource-sharing concept is introduced to the DA-MUX-LUT block that drastically reduces the adder requirement. The application specific integrated circuit (ASIC) synthesis results show that the proposed DA-MUX-LUT-based 2-D block FIR filter for filter size 8×8 and block size 4 has 31.22% less delay, 28.66% less area-delay product (ADP), 37.70% less power-delay product (PDP), and occupies almost the same area than the existing architecture.

Keywords: 2-D FIR filter, switching-based LUT, distributed arithmetic, block processing.

1. Introduction

The 2-D filters are used in digital signal processing applications [4]. Critical applications include image restoration, compression, face recognition, edge detection, satellite imaging, target acquisition, tracking, microwave imaging, biometric applications, etc. Parhi [2] proposed a systolic architecture for 2-D FIR filters. Fewer computations used the symmetry property suggested in [5] and [7]. Alawad and Lin [8] presented a non-separable stochastic-based 2-D FIR filter architecture with low hardware complexity and high throughput using probabilistic convolution. The suggested work solves the problem within a predetermined ac-

curacy range. By exploiting the convolution theorem, the probability density function represents the 2-D input signal kernels. This well-known probabilistic convolution theorem replaces the expensive multipliers with simple adders. This work is more suitable for applications such as perception-based image processing, which can inherently tolerate some computing inaccuracy.

The non-separable and separable models for the 2-D FIR filters were suggested by Mohanty *et al.* [9]. The non-separable model has a shorter critical path but requires more computations, whereas the separable model has a longer critical path but efficiently reduces the computations. Kumar *et al.* [10] introduced a block-based high throughput and hardware-efficient systolic 2-D FIR filter architecture implementation method using the DA algorithm. In this, hardware-efficient DA-LUTs (HLUT) were developed for reconfigurable applications. The sharing occurs between the DA-LUTs, and a common DA-LUT is implemented and shared to reduce the hardware complexity at each systolic stage. This is the first article related to implementing a 2-D FIR filter architecture using DA, and this work resolved the shortcomings of previous works. The authors suggest block-based 2-D FIR filter banks decrease the memory footprint, area, and delay. Odugu *et al.* [14] proposed a generic filter bank architecture using block-based non-separable 2-D FIRAs with various symmetries. The complexity of the filter increases as the order of the filter and block size increase.

Due to their area-efficient and high-throughput abilities, multiplier-less approaches such as multiple constant multiplications (MCM) and DA approaches are popular in very large scale integration (VLSI) for designing multiply and accumulate (MAC) units. For the constant filter coefficients, MCM is used [15]. However, for efficient reconfigurable architectures, DA approaches are used [1, 3, 6, 10–13]. Some applications such as wireless communication networks, adaptive filtering, software designed ration (SDR)-based filtering, and multi-channel filtering systems have the necessity that the filter coefficients be modified dynamically, and reconfigurability must be considered in the filter architecture design. DA-based approaches allow for dynamic reconfigurability of filter coefficients, but MCM-based techniques are not.

This paper proposes a novel architecture for a DA-based 2-D non-separable block FIR filter using a multiplexer. Thus, the hardware complexity is reduced with a slight increment in critical path delay. Block processing helps in improving the throughput. Parallel adders are substituted by modified carry look-ahead adders (MCLAs), which reduces the critical path delay. Pipelining and parallelism reduce the proposed architecture's time and hardware complexities. Each LUT element is computed using a dedicated MUX-based architecture. The suggested DA-MUX-LUT architecture computes the partial-product instantaneous using the bit vector. Further, in each systolic stage, DA-MUX-LUTs are shared to minimize the number of adders. Though LUT sharing is used to reduce the

adder requirements, the proposed architecture requires large numbers of adders in the DA-MUX-LUTs design. The necessity of adders increases in great numbers with the size of the filter.

The paper is organized as follows: Sec. 2 explains the concept of block processing and memory sharing in a 2-D FIR filter. The mathematical formulation of the 2-D block FIR filter using DA-MUX-LUT is explained in Sec. 3. The proposed architecture for DA-MUX-LUT and inter-DA-MUX-LUT resource sharing is given in Sec. 4. Implementation and experimental results are explained in Sec. 5. In Sec. 6, the conclusion is summarized.

2. Block formulation of 2-D FIR filter

Consider $x\left(n_1, n_2 - m\right)$ is the $(m+1)$-th the input of the 2-D FIR filter, $w(p,q)$ represents filter coefficient matrix. Equation (1) represents $(m+1)$-th output of the 2-D FIR filter:

$$Y_m\left(n_1, n_2\right) = \left\{ \left\{ x\left(n_1 - p, n_2 - m - q\right) \cdot w\left(p, q\right) \right\}_{p=0}^{L-1} \right\}_{q=0}^{L-1}, \tag{1}$$

$w\left(p, q\right)$ is expressed as in Eq. (2):

$$w\left(p, q\right) = \begin{bmatrix} w(0,0) & w(0,1) & \cdots & w(0,L-1) \\ w(1,0) & w(1,1) & \cdots & w(1,L-1) \\ \vdots & \vdots & \cdots & \vdots \\ w(L-1,0) & w(L-1,1) & \cdots & w(L-1,L-1) \end{bmatrix}, \tag{2}$$

where $0 < m < N-1$. Equation (1) can be written using the direct form 1 (DF1) systolic architecture, as illustrated in Fig. 1. Here, we have considered that the input pixels are processed in a raster scanning format, i.e., row-wise scanning. So, in the architecture, to meet the requirements, shift registers of length $M-1$ are added at each systolic level to store the input pixels, where M is the width of the input image matrix.

Block processing enables fast and computationally efficient implementation of digital filters by increasing throughput. In Fig. 1, block of 4 input samples $\{x\left(n_1, n_2\right), x\left(n_1, n_2 - 1\right), x\left(n_1, n_2 - 2\right), x\left(n_1, n_2 - 3\right)\}$ is processed to obtain 4 output samples $\{y\left(n_1, n_2\right), y\left(n_1, n_2 - 1\right), y\left(n_1, n_2 - 2\right), y\left(n_1, n_2 - 3\right)\}$ in parallel for the 4th order filter. Each output of 4×4 filters is computed with the help of 16 samples corresponding to 4 rows in 1 column of 2-D input. Hence, for the computation of the outputs, a total of 64 input samples corresponding to 4 rows and 4 columns of 2-D filter inputs are needed. Out of 64 samples, 36 samples are redundant, and only 28 samples are different. The 36 samples

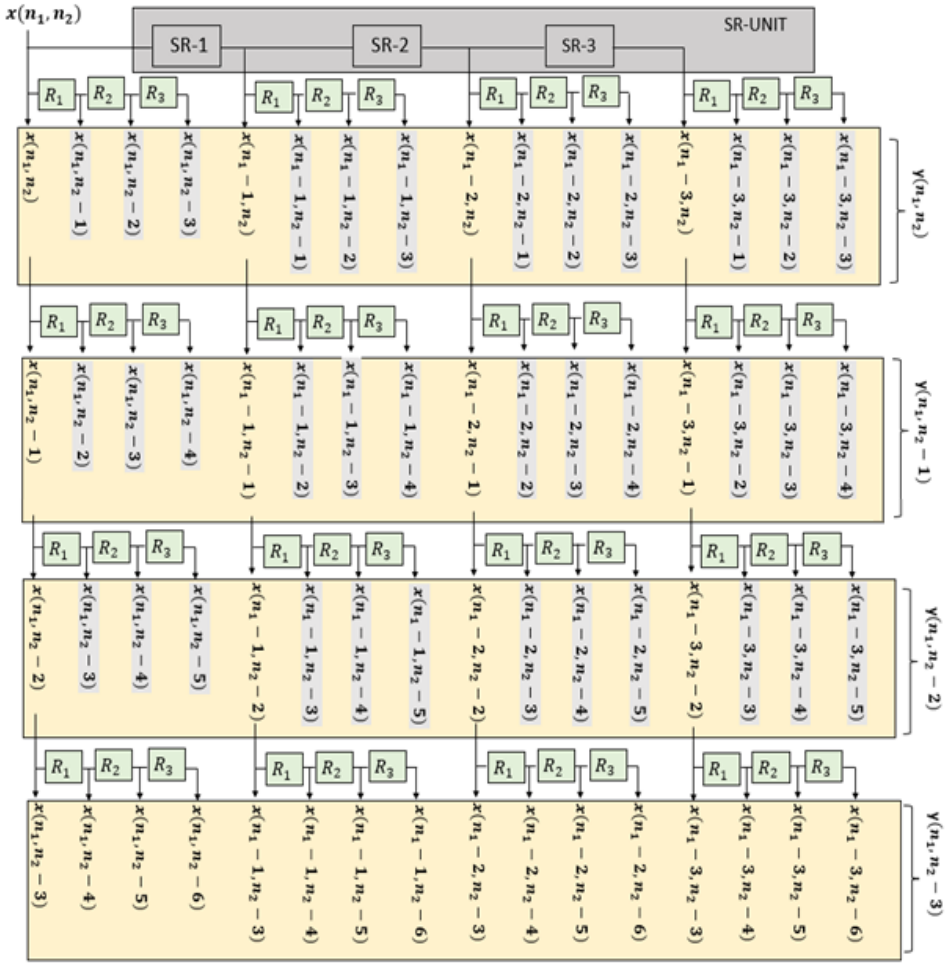

FIG. 1. Register unit dataflow of proposed FIR filter realized using direct form 1.

that are the same are shaded in Fig. 1. The memory can be saved by removing these redundancies and computing the filter outputs in parallel. The 28 different samples required to compute the outputs may be generated using an internal systolic structure. However, another close inspection reveals that out of these 28 samples, 4 are current input samples, and 21 are already present in SRUs (i.e., SRU-1, SRU-2, and SRU-3), which need to generate the remaining three samples. Therefore, only at the last systolic stage a dedicated structure comprising three delay elements are required to generate the remaining 3 samples: $x\,(n_1 - 3, n_2 - 4)$, $x\,(n_1 - 3, n_2 - 5)$ and $x\,(n_1 - 3, n_2 - 6)$. This is unlike the existing structures, where dedicated internal systolic structures are required to generate the delayed inputs. Therefore, the SRU is split into N equal parts of $\frac{M}{N}$ registers to obtain the required delayed samples.

2.1. Mathematical formulation

The input matrix $X(n_1, n_2)$ is needed at different systolic stages to generate 2-D filter output $Y(n_1, n_2)$ of the filter length L. Figure 1 presents the realization of the 2-D FIR filter architecture using DF1:

$$X(n_1, n_2) = \begin{bmatrix} x(n_1, n_2) & x(n_1, n_2 - 1) & \cdots & x(n_1, n_2 - L + 1) \\ x(n_1 - 1, n_2) & x(n_1 - 1, n_2 - 1) & \cdots & x(n_1 - 1, n_2 - L + 1) \\ \vdots & \vdots & \cdots & \vdots \\ x(n_1 - L + 1, n_2) & x(n_1 - L + 1, n_2 - 1) & \cdots & x(n_1 - L + 1, n_2 - L + 1) \end{bmatrix}. \tag{3}$$

Let us consider $x(n_1, n_2 - m)$ is the $(m+1)$-th input of the 2-D FIR filter, $w(p, q)$ represents a coefficient vector. The $(m+1)$-th output of the filter is expressed as:

$$Y_m(n_1, n_2) = \left\{ \{ x(n_1 - p, n_2 - m - q) \cdot w(p, q) \}_{p=0}^{L-1} \right\}_{q=0}^{L-1}, \tag{4}$$

In each iteration, a block of N input samples is processed. At each systolic stage, a set of $L-1$ delayed inputs is required for generating a block of input. The input pixels at $(p+1)$-th stage is represented by $G(n_1, n_2)$, which is given in matrix form as:

$$G(n_1, n_2) = \begin{bmatrix} x(n_1 - p, n_2) & x(n_1 - p, n_2 - 1) & \cdots & x(n_1 - p, n_2 - L + 1) \\ x(n_1 - p, n_2 - 1) & x(n_1 - p, n_2 - 2) & \cdots & x(n_1 - p, n_2 - L) \\ \vdots & \vdots & \cdots & \vdots \\ x(n_1 - p, n_2 - N + 1) & x(n_1 - p, n_2 - N) & \cdots & x(n_1 - p, n_2 - N - L + 2) \end{bmatrix}. \tag{5}$$

The filter coefficient vector required at each stage is expressed as:

$$w_p = [w(p, 0) \quad w(p, 1) \quad w(p, 2) \quad \cdots \quad w(p, L-1)]_{1 \times L}. \tag{6}$$

Thus, the 2-D FIR filter block output at each systolic stage is expressed using Eqs. (5) and (6) as:

$$Y = \left\{ G(n_1, n_2) \cdot w_p^T \right\}_{p=0}^{L-1}. \tag{7}$$

To facilitate parallelism, we further decompose the input pixel matrix $G(n_1, n_2)$ and weight vector by a factor of s. The input pixel matrix $G(n_1, n_2)$ is decomposed into $\frac{L}{S}$ sub-matrices represented as X_p^q of dimension $G(N \times S)$,

also the coefficient weight vector w_p^q of dimension $(1 \times S)$; $0 \leq q \leq \left(\frac{L}{S}\right) - 1$. Equation (9) is modified as:

$$[Y]_{N\times 1} = \left\{ \left\{ X_p^q (w_p^q)^T \right\}_{q=0}^{\left(\frac{L}{S}\right)-1} \right\}_{p=0}^{L-1}. \tag{8}$$

Equation (8) is re-written as

$$[Y]_{N\times 1} = \{y_p^q\}_{q=0}^{\left(\frac{L}{S}\right)-1}, \tag{9}$$

where y_p^q represents the inner block product of $X_p^q (w_p^q)^T$, i.e.,

$$y_p^q = \left\{ X_p^q (w_p^q)^T \right\}_{p=0}^{L-1}. \tag{10}$$

3. Mathematical formulations of proposed DA-MUX-LUT-based 2-D FIR filter

The $(i+1)$-th component of coefficient weight vector-matrix w_p^q is w_{pi}^q whose length is B bits, and $|w_{pi}^q| \leq 1$. Its two's complement representation is

$$w_{pi}^q = -w_{pi}^q(0) + \left\{ w_{pi}^q(l) \cdot 2^{-l} \right\}_{l=0}^{B-1},$$

where $w_{pi}^q(0)$ represents sign bit and $w_{pi}^q(B-1)$ represents the least significant bit (LSB):

$$w_{pi}^q = \left\{ w_{pi}^{q'}(l) \cdot 2^{-l} \right\}_{l=0}^{B-1}, \tag{11}$$

where

$$w_{pi}^{q'}(l) = \begin{cases} -w_{pi}^q(0) & \text{at } \ l = 0, \\ w_{pi}^q(l) & \text{otherwise.} \end{cases}$$

Let $(m+1)$-th element of y_p^q be $y_p^q(m)$, $0 \leq m \leq N-1$, and it is given as

$$y_p^q(m) = \left\{ \left\{ X_p^q(m,i) \cdot (w_{pi}^q)^T \right\}_{i=0}^{S-1} \right\}_{q=0}^{\left(\frac{L}{S}\right)-1} = \{R_m(p,q)\}_{q=0}^{\left(\frac{L}{S}\right)-1}, \tag{12}$$

where

$$R_m(p,q) = \left\{ X_p^q(m,i) \cdot (w_{pi}^q)^T \right\}_{i=0}^{S-1}. \tag{13}$$

Further, $R_m(p,q)$ in Eq. (13) is the vector product of the input bit vector $\{X_p^q(m,i)\}_{i=0}^{S-1}$ and the weight vector $\left\{\left(w_{pi}^q\right)^T\right\}_{i=0}^{S-1}$. LUT is used to save the precomputed partial products. The weight vector $\{w_{p1}^q, w_{p2}^q, w_{p3}^q, ..., w_{p(S-1)}^q\}$ is used as an address to fetch the partial products from LUT. Since the input bit vector has s bits, 2^s combinations are possible for $\left\{w_{pi}^q\right\}_{i=0}^{S-1}$.

Substituting Eqs. (9) and (11) in Eq. (13) results in

$$R_m(p,q) = \left\{\left\{x(u,v-m-i)\cdot w_{pi}^{q'}(l)\cdot 2^{-l}\right\}_{l=0}^{B-1}\right\}_{i=0}^{S-1}, \tag{14}$$

$$R_m(p,q) = \left\{R'_m(p,q)\cdot 2^{-l}\right\}_{l=0}^{B-1}, \tag{15}$$

where

$$R'_m(p,q) = \left\{x(u,v-m-i)\cdot w_{pi}^{q'}(l)\right\}_{i=0}^{S-1}. \tag{16}$$

It is observed that the input vector $\{X_p^q(m,i)\}_{i=0}^{S-1}$ varies from sample to sample. So, we are using an adder tree-based combinational circuit, as indicated in Fig. 2. It comprises $(L-1)$ adders of the $\log_2 L$ stage tree.

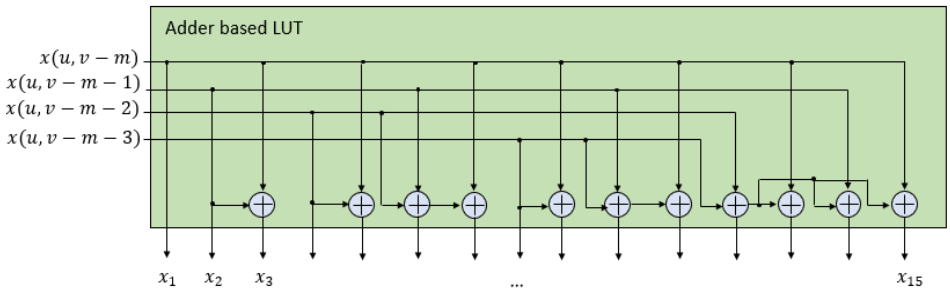

FIG. 2. Adder tree-based structure to compute partial sum terms.

Let $S = 8$

$$R'_m(p,q) = \left\{x(u,v-m-i)\cdot w_{pi}^{q'}(l)\right\}_{i=0}^{7}. \tag{17}$$

Equation (17) can be decomposed into two vector-products summations of order four as:

$$R'_m(p,q) = \left\{x(u,v-m-i)\cdot w_{pi}^{q'}(l)\right\}_{i=0}^{3} + \left\{x(u,v-m-i-4)\cdot w_{p(i+4)}^{q'}(l)\right\}_{i=0}^{3}. \tag{18}$$

Let us assume that the block size $S = 8$, then $2^8 = 256$ input vectors are needed to be computed. Instead of these many input vectors, we have considered small order decomposition to calculate the vector product, represented in Eq. (18). Using that equation, we require $2^{\frac{S}{2}+1}$ sum terms. Thus, it requires 32 sum terms. For simplicity, these sum terms are considered as $\{x_0, x_1, x_2, ..., x_{31}\}$:

$$R'_m(p,q) = R_1(p,q) + R_2(p,q), \tag{19}$$

where

$$R_1(p,q) = \left\{ x(u, v-m-i) \cdot w_{pi}^{q'}(l) \right\}_{i=0}^{3}, \tag{20}$$

$$R_2(p,q) = \left\{ x(u, v-m-i-4) \cdot w_{p(i+4)}^{q'}(l) \right\}_{i=0}^{3}. \tag{21}$$

4. Proposed architecture of 2-D block FIR filter

Figure 3 represents the block diagram of the DA-MUX-LUT-based 2-D block FIR filter. It consists of L (i) shift register units (SR-units), (ii) processing element block (PEB); the output of these PEBs is delayed and added using (iii) parallel-delay-adder (PDA) unit, and (iv) control unit (CU) is used to send required control signals and synchronize all the sub-blocks.

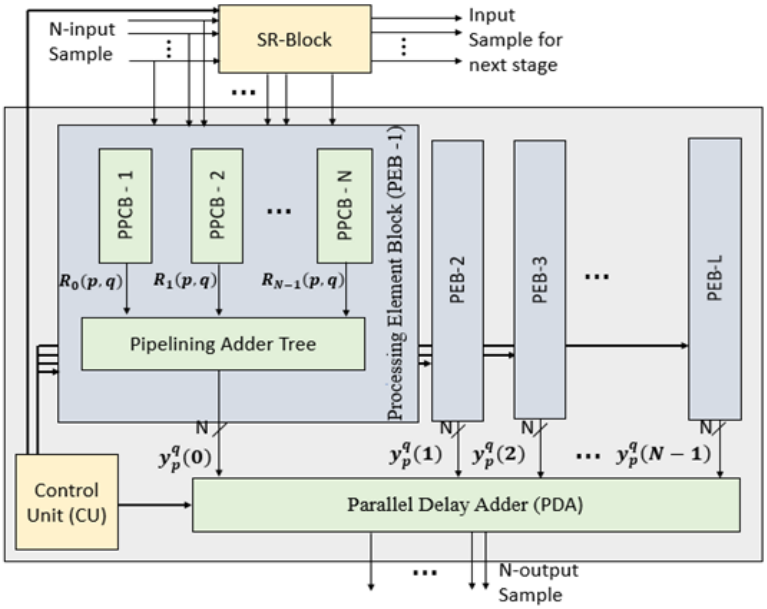

Fig. 3. Block diagram of the proposed reconfigurable 2-D block FIR filter.

To realize the 2-D FIR filter architecture for Eq. (7), we are required to design $G(n_1, n_2)$. It can be designed using the fully direct systolic structure, as shown in Fig. 1. The required delayed inputs are generated by using SR-unit. Thus, the input matrix $G(n_1, n_2)$ is given to the PEB, which computes the output vector y_p^q at each systolic stage. To accomplish block processing, the input matrix $G(n_1, n_2)$ is split into L/S parallel matrix, as presented in Eqs. (8) and (9). The PEB consists of N number of concurrent partial product computation units (PPCU) shown in Fig. 3. The parallel input vector-matrix X_p^q is also split into N parallel input vectors $X_p^q(m, i)$ and given to each PPCU.

Equations (20) and (21) represent partial product calculation of coefficient vector and input vector of order 4. Equation (20) is elaborated in Table 1, the weight vector elements $\{w_{p3}^{q'}(l), w_{p2}^{q'}(l), w_{p1}^{q'}(l), w_{p0}^{q'}(l)\}$ are used as an address to fetch the corresponding bits of the vector products. Table 1 contains 16 terms, these are given as input to the DA-MUX-LUT, and the weight vector elements are employed as selection lines. Thus, depending on the selection lines, output $R_1(p, q)$ gets the corresponding bits of the vector product. Similarly, when the weight vector elements $\{w_{p7}^{q'}(l), w_{p6}^{q'}(l), w_{p5}^{q'}(l), w_{p4}^{q'}(l)\}$ are used as an address, then we can get the corresponding bits of the output $R_2(p, q)$ according to Eq. (21). To get the vector product corresponding to Eq. (19), we need to per-

Table 1. DA-LUT for vector product $R_1(p, q)$.

$w_{p3}^{q'}(l)$	$w_{p2}^{q'}(l)$	$w_{p1}^{q'}(l)$	$w_{p0}^{q'}(l)$	$R_1(p, q)$	$R_1(p, q)$
0	0	0	0	0	x_0
0	0	0	1	$x(uv - m)$	x_1
0	0	1	0	$x(uv - m - 1)$	x_2
0	0	1	1	$x(u, v - m) + x(uv - m - 1)$	x_3
0	1	0	0	$x(uv - m - 2)$	x_4
0	1	0	1	$x(u, v - m - 2) + x(uv - m)$	x_5
0	1	1	0	$x(u, v - m - 2) + x(uv - m - 1)$	x_6
0	1	1	1	$x(u, v - m - 2) + x(u, v - m - 1) + x(uv - m)$	x_7
1	0	0	0	$x(uv - m - 3)$	x_8
1	0	0	1	$x(u, v - m - 3) + x(uv - m)$	x_9
1	0	1	0	$x(u, v - m - 3) + x(uv - m - 1)$	x_{10}
1	0	1	1	$x(u, v - m - 3) + x(u, v - m - 1) + x(uv - m)$	x_{11}
1	1	0	0	$x(u, v - m - 3) + x(uv - m - 2)$	x_{12}
1	1	0	1	$x(u, v - m - 3) + x(u, v - m - 2) + x(uv - m)$	x_{13}
1	1	1	0	$x(u, v - m - 3) + x(u, v - m - 2) + x(uv - m - 1)$	x_{14}
1	1	1	1	$x(u, v - m - 3) + x(u, v - m - 2)$ $+x(u, v - m - 1) + x(uv - m)$	x_{15}

form a summation of $R_1(p,q)$ and $R_2(p,q)$ using a MCLA, as shown in Fig. 4a, which reduces delay and slightly increases hardware complexity.

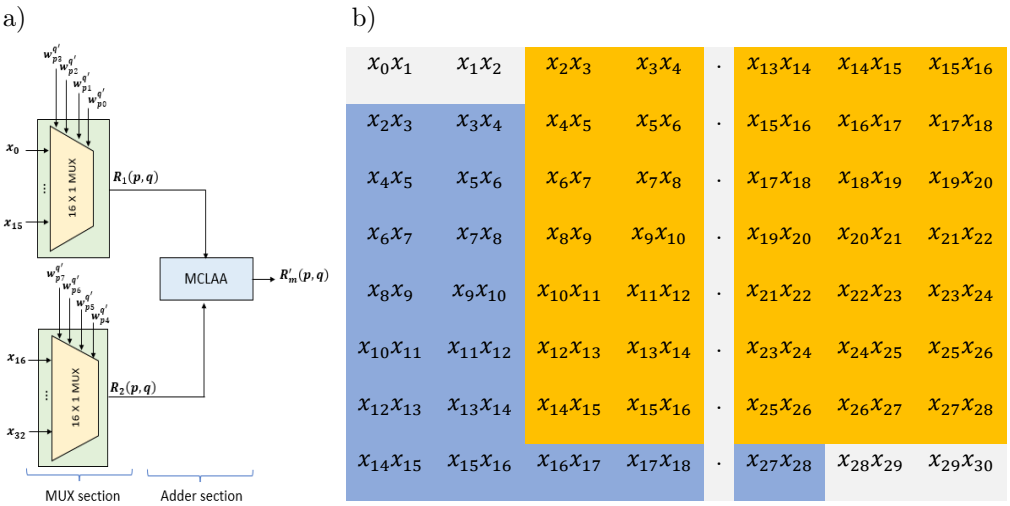

FIG. 4. (a) Structure of DA-based LUT using multiplexer (DA-MUX-LUT) using modified carry look-ahead adder (b) DA-MUX-LUT resource sharing.

It is noticed that the second summation terms of (22), (23), (24), and (25) are the same as the first summation terms of (26), (27), (28), and (29), respectively. Therefore, instead of 16 LUTs, we can use 12 LUTs, in which 4 LUTs are common. As the filter length increases, the requirement for adders and DA-MUX-LUTs also increases. Thus, using this resource-sharing concept, we can drastically reduce adder requirements.

The possible input vectors of $R'_m(p,q)$ for constant p are shown in Fig. 4b. Here, $x(n_1-p, n_2-q)$ is represented as x_q and they are written in vertical concerning m. The input vectors, highlighted by blue and orange colors, are the same as shown in Fig. 4b. It means the MUX section in Fig. 4a is needed to be designed only for first two columns and the last row. The designed MUX section can be shared among the DA-MUX-LUTs, with the same input vector. Using this sharing approach, the adder requirement is drastically reduced concerning constant p; for filter size $L = 16$ and $0 \leq p \leq 15$, almost 2950 adders can be reduced.

4.1. DA-MUX-LUT resource sharing

Let us consider filter order $L = 8$, block size $N = 4$, and decomposition factor $S = 8$. Thus $R'_m(p,q)$, for constant p, is written as:

$$R'_0(p,0) = \{x(u,v-i)\cdot w^0_{pi}(l)\}^3_{i=0} + \{x(u,v-i-4)\cdot w^0_{p(i+4)}(l)\}^3_{i=0}, \quad (22)$$

$$R'_1(p,0) = \{x(u,v-i-1)\cdot w^0_{pi}(l)\}^3_{i=0} + \{x(u,v-i-5)\cdot w^0_{p(i+4)}(l)\}^3_{i=0}, \quad (23)$$

$$R'_2(p,0) = \{x(u,v-i-2)\cdot w^0_{pi}(l)\}^3_{i=0} + \{x(u,v-i-6)\cdot w^0_{p(i+4)}(l)\}^3_{i=0}, \quad (24)$$

$$R'_3(p,0) = \{x(u,v-i-3)\cdot w^0_{pi}(l)\}^3_{i=0} + \{x(u,v-i-7)\cdot w^0_{p(i+4)}(l)\}^3_{i=0}, \quad (25)$$

$$R'_4(p,0) = \{x(u,v-i-4)\cdot w^0_{pi}(l)\}^3_{i=0} + \{x(u,v-i-8)\cdot w^0_{p(i+4)}(l)\}^3_{i=0}, \quad (26)$$

$$R'_5(p,0) = \{x(u,v-i-5)\cdot (l)\}^3_{i=0} + \{x(u,v-i-9)\cdot w^0_{p(i+4)}(l)\}^3_{i=0}, \quad (27)$$

$$R'_6(p,0) = \{x(u,v-i-6)\cdot w^0_{pi}(l)\}^3_{i=0} + \{x(u,v-i-10)\cdot w^0_{p(i+4)}(l)\}^3_{i=0}, \quad (28)$$

$$R'_7(p,0) = \{x(u,v-i-7)\cdot w^0_{pi}(l)\}^3_{i=0} + \{x(u,v-i-11)\cdot w^0_{p(i+4)}(l)\}^3_{i=0}. \quad (29)$$

In the proposed design, we considered the decomposition factor $S = 8$. Each $(m+1)$-th row of the input matrix $\{n_1 - p, n_2 - 8q - k\}^{L-1}_{k=0}$ is given to the corresponding PPCU. Each PPCU consists of DA-MUX-LUT and shift and accumulate unit (SAU), as shown in Fig. 5. We already mentioned that memory sharing is employed, so a common MUX section is used among various PPCU units. Further, the partial products are fed to SAU, where $R'_m(p,q)$ is shifted and accumulated to generate $R_m(p,q)$. According to Eq. (16), $R_m(p,q)$ has B bits, requiring B cycles to compute the output. Pipelining stage is used to limit the critical path delay. Each PPCU computes N-bit vector output $R_m(p,q)$. We have L such PPCUs in our design.

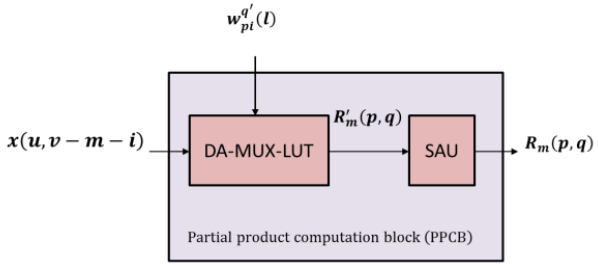

Fig. 5. Structure of partial product computation unit (PPCU).

These vector outputs are needed to be added according to Eq. (12). Therefore, an adder tree (AT) is used to compute the output $y_p^q(m) = \{y_p^q(0), y_p^q(1), ..., y_p^q(N-1)\}$. $\log_2\left(\frac{L}{S}\right)$ stage pipeline adder tree (PAT), as shown in Fig. 6. Further, the obtained output vector $y_p^q(m)$ is delayed and added symbolically using a parallel delay add (PDA) unit, shown in Fig. 7. The control unit (CU) generates the required control signals, and the flow of signals is synchronized.

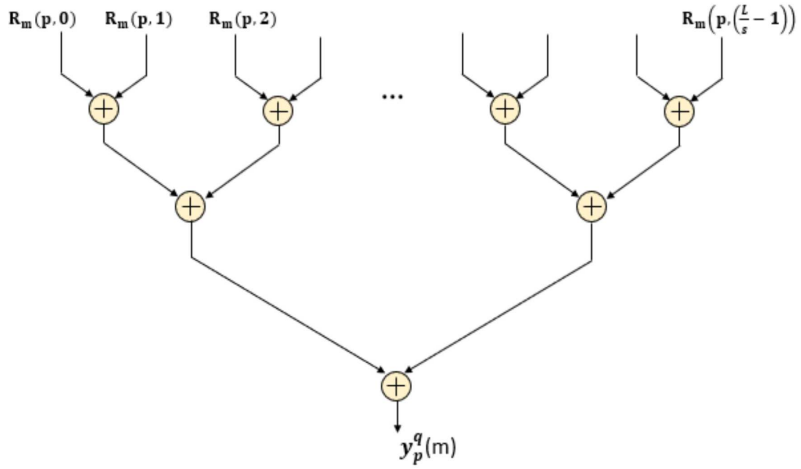

Fig. 6. Structure of pipeline adder tree (PAT).

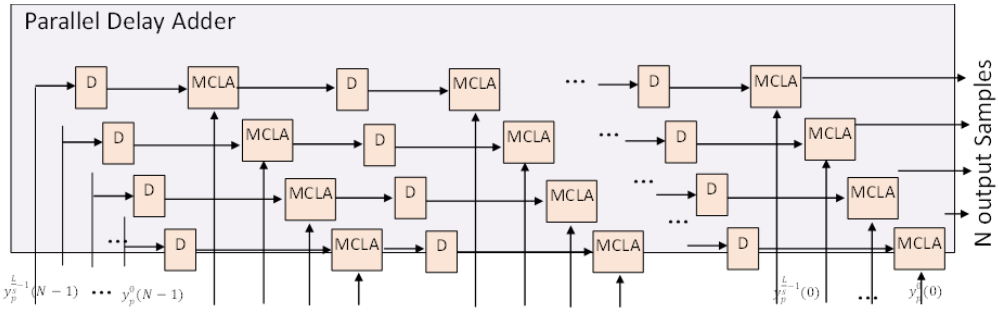

Fig. 7. Structure of parallel adder tree (PAT).

5. Implementation and experimental results

The proposed 2-D FIR filter architectures are coded by Verilog HDL and simulated by Cadence Incisive Enterprise Simulator-XL-15.2. The application-specific integrated circuit (ASIC)-based 2-D FIR filter architectures are realized in Cadence TSMC 45nm CMOS generic library using the advanced Genus synthesis tool. The synthesis tools generate the power, delay, and area reports.

The proposed block-based 2-D FIR filter using DA-MUX-LUT multipliers is compared with existing architectures. The performance metrics obtained from the synthesis tool for $L = 4$ and $N = 4$ are presented in Table 2.

Table 2. Comparison of proposed architecture with existing architectures for $L = 4$ and $N = 4$.

Architecture	Power [mW]	Delay [ns]	Area [μm^2]	ADP [$\mu m^2 \cdot Ms$]	PDP [$mW \cdot ns$]
Alawad and Lin [8]	3.05	12.82	286 153	3.66	39.101
Mohanty *et al.* [9]	7.093	15.02	591 361	8.88	106.53
Kumar *et al.* [10]	5.05	9.29	429 825	3.99	46.91
Proposed 2-D FIR filter	**5.21**	**6.46**	**480 455**	**3.1**	**33.65**

From Table 2, the ASIC synthesis results show that the proposed DA-MUX-LUT-based 2-D FIR filter architecture for $L = 4$ and $N = 4$ occupies 18.75% less area when compared to architectures in [9]. It has 49.61%, 56.9%, and 30.46% less delay when compared to architectures in Alawad and Lin [8], Mohanty *et al.* [9], and Kumar *et al.* [10], respectively. It has 26.54% less power consumption when compared to existing architectures [9]. It occupies 15.30%, 65.09%, and 22.30% less ADP when compared to architectures in [8–10], respectively, and it occupies 13.9%, 68.41%, and 28.26% less PDP when compared to architectures in [8–10], respectively.

From Table 3, the ASIC synthesis results show that the proposed DA-MUX-LUT-based 2-D FIR filter architecture for $L = 8$ and $N = 4$ occupies 27.55% less area when compared to [9]. It has 38.75%, 50.4%, and 31.22% less delay when compared to architectures in [8–10], respectively. It has 32.16%, 40.62%, and 8.7% less power consumption when compared to architectures in [8–10], respectively. It occupies 2.55%, 64.35%, and 28.66% less ADP when compared to existing architectures [8–10], respectively, and it occupies 58.73%, 70.75%, and 37.70% less PDP when compared to architectures in [8–10], respectively.

Table 3. Comparison of proposed architecture with existing architectures for $L = 8$ and $N = 4$.

Architecture	Power [mW]	Delay [ns]	Area [μm^2]	ADP [$\mu m^2 \cdot Ms$]	PDP [$mW \cdot ns$]
Alawad and Lin [8]	7.96	13.16	359 271	4.7	104.75
Mohanty *et al.* [9]	9.094	16.25	785 268	12.85	147.77
Kumar *et al.* [10]	5.92	11.72	547 803	6.42	69.38
Proposed 2-D FIR filter	**5.4**	**8.06**	**568 910**	**4.58**	**43.22**

6. Conclusion

This paper implements a novel 2-D FIR filter architecture using block processing using DA-MUX-based LUT multipliers. In this, considerable power saving is achieved due to the multiplier-less design by DA multiplication. The block processing increases the throughput of the 2-D FIR filter, and the multiplication process is implemented with the DA-MUX-LUT-based multiplication process to improve the performance metrics. Pipelining and parallelism reduce the proposed architecture's time and hardware complexities. A resource-sharing concept is introduced among the DA-MUX-LUT block that drastically reduces the adder requirements in the architecture. Parallel adders are substituted by the modified MCLAs, which minimizes the delay. The ASIC synthesis result shows that the proposed architecture for $L = 8$ and $N = 4$ has 31.22% less delay, 28.66% less ADP and 37.70% less PDP with a trade-off between area and power. It requires 3.78% more area and consumes 8.78% more power than the HLUT-based 2-D FIR filter. The proposed method is well suited for applications that require very high processing. Symmetry in the filter coefficients minimizes the number of multipliers in the architecture. Hence, symmetry will be introduced in the 2-D FIR filter architecture in the future to decrease the area and power consumption.

References

1. D.J. Allred, H. Yoo, V Krishnan, W. Huang, D.W. Anderson, LMS adaptive filters using distributed arithmetic for high throughput, *IEEE Transactions on Circuits and Systems I: Regular Papers*, **52**(7): 1327–1337, doi: 10.1109/TCSI.2005.851731.

2. K.K. Parhi, *VLSI digital signal processing systems: design and implementation*, John Wiley & Sons, 2007.

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*, **58**(9): 600–604, 2011, doi: 10.1109/TCSII.2011.2161168.

4. M.A. Sid-Ahmed, A systolic realization for 2-D digital filters, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **37**(4): 560–565, 1989, doi: 10.1109/29.17537.

5. I.-H. Khoo, H.C. Reddy, L.-D. Van, C.-T. Lin, Generalized formulation of 2-D filter structures without global broadcast for VLSI implementation, [in:] *2010 53rd IEEE International Midwest Symposium on Circuits and Systems*, pp. 426–429, IEEE, 2010, doi: 10.1109/MWSCAS.2010.5548755.

6. C.P. Chowdari, J.B. Seventline, An efficient FIR filter architecture implementation using distributed arithmetic (DA) for DSP applications, *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, **8**(9S3): 330–337, 2019, doi: 10.35940/ijitee.I3061.0789S319.

7. P.-Y. Chen, L.-D. Van, H.C. Reddy, I.-H. Khoo Area-efficient 2-D digital filter architectures possessing diagonal and four-fold rotational symmetries, [in:] *2013 9th International*

Conference on Information, Communications & Signal Processing, pp. 1–5, IEEE, 2013, doi: 10.1109/ICICS.2013.6782888.

8. M. Alawad, M. Lin, Memory-efficient probabilistic 2-D finite impulse response (FIR) filter, *IEEE Transactions on Multi-Scale Computing Systems*, **4**(1): 69–82, 2017, doi: 10.1109/TMSCS.2017.2695588.

9. B.K. Mohanty, P.K. Meher, S. Al-Maadeed, A. Amira, Memory footprint reduction for power-efficient realization of 2-D finite impulse response filters, *IEEE Transactions on Circuits and Systems I: Regular Papers*, **61**(1): 120–133, 2014, doi: 10.1109/TCSI.2013.2265953.

10. P. Kumar, P.C. Shrivastava, M. Tiwari, G.R. Mishra, High-throughput, area-efficient architecture of 2-D block FIR filter using DA algorithm, *Circuits, Systems, and Signal Processing*, **38**(3): 1099–1113, 2019, doi: 10.1007/s00034-018-0897-2.

11. C.P. Chowdari, J.B. Seventline, Systolic architecture for adaptive block FIR filter for throughput using distributed arithmetic, *International Journal of Speech Technology*, **23**(3): 549–557, 2020, doi: 10.1007/s10772-020-09745-4.

12. C.P. Chowdari, J.B. Seventline, VLSI implementation of distributed arithmetic based block adaptive finite impulse response filter, *Materials Today: Proceedings*, **33**(part 7): 3757–3762, 2020, doi: 10.1016/j.matpr.2020.06.206.

13. C.P. Chowdari, J.B. Seventline, Low power implementation of adaptive block FIR filter design using offset binary coding, *Materials Today: Proceedings*, 2021, doi: 10.1016/j.matpr.2020.12.869.

14. V.K. Odugu, C.V. Narasimhulu, K.S. Prasad, Design and implementation of low complexity circularly symmetric 2D FIR filter architectures, *Multidimensional Systems and Signal Processing*, **31**: 1385–1410, 2020, doi: 10.1007/s11045-020-00714-3.

15. X. Lou, Y.J. Yu, P.K. Meher, Lower bound analysis and perturbation of critical path for area-time efficient multiple constant multiplications, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **36**(2): 313–324, 2017, doi: 10.1109/TCAD.2016.2584181.

Received March 8, 2022; revised version June 21, 2022;
accepted July 28, 2022.