Circuits, Systems, and Signal Processing (2019) 38:1099–1113
https://doi.org/10.1007/s00034-018-0897-2

High-Throughput, Area-Efficient Architecture of 2-D Block FIR Filter Using Distributed Arithmetic Algorithm

Prashant Kumar[1] · **Prabhat Chandra Shrivastava**[1] · **Manish Tiwari**[1] · **Ganga Ram Mishra**[2]

Received: 18 December 2017 / Revised: 6 July 2018 / Accepted: 9 July 2018 / Published online: 19 July 2018

Abstract

This paper proposes a new architecture of 2-D block FIR filter using distributed arithmetic (DA) algorithm, which is known for the efficient design of multiply and accumulate block. Hardware-based architecture is proposed for DA lookup table (DA-LUT) that makes the architecture of 2-D FIR filter reconfigurable. Further, due to block processing, sharing takes place among DA-LUTs at various stages. Thus, a common DA-LUT may be designed for block inputs which reduce the hardware complexity for DA-LUT. Furthermore, memory overlapping is used to reduce the systolic architectures in proposed design over existing designs. For higher-order 2-D FIR filter, the complexity of DA-LUT is reduced by dividing the internal block into parallel and small blocks. With the help of ASIC synthesis results, a comparative analysis of proposed design with the earlier reported designs is presented, and it is shown that the proposed design leads to significant improvements in various performance parameters.

Keywords 2-D block FIR filtering · Distributed arithmetic · Multiplier-less filter · VLSI design

✉ Prashant Kumar
pkumar.mnnit@gmail.com

Prabhat Chandra Shrivastava
prabhatphd@gmail.com

Manish Tiwari
mtiwari@mnnit.ac.in

Ganga Ram Mishra
gr_mishra@rediffmail.com

[1] Department of Electronics and Communication Engineering, MNNIT, Allahabad 211004, India

[2] Amity University (Lucknow Campus), Lucknow, India

1 Introduction

Two-dimensional (2-D) digital filters are widely used in the variety of image and video processing applications, such as image restoration, image enhancement, template matching, and video communication [11, 13]. The 2-D FIR filters are preferred over IIR filters due to their numerical stability and simplicity in design. Moreover, an FIR filter makes an attractive choice when the applications require preserving linear phase.

An efficient structure design of 2-D FIR is always challenging for researchers due to a large number of computations. In [11], Parhi has suggested a systolic structure for 2-D FIR filters. However, a large number of computations required in [11] is reduced in [2, 4] by using symmetry properties. Recently, Mohanty has suggested the separable and non-separable structure for the 2-D block FIR filters in [9]. The non-separable structure requires a large number of computations but has the shorter critical path over the separable structure, whereas separable structure reduces the computations but has the longer critical path.

In the recent past, multiplier-less multiple constant multiplications (MCMs) and distributed arithmetic (DA) technique have gained substantial popularity in VLSI design due to its high-throughput and area-efficient capabilities in MAC (multiply and accumulate) design. However, MCM is used when the filter coefficients are constant [6], whereas DA techniques are generally used for the efficient reconfigurable architecture design [1, 3, 5, 7, 8, 12]. In the recent trends, reconfigurability needs to be taken into account in the filter architecture design since applications such as adaptive filtering, multichannel filtering, multifiltering (different types of filtering operations using a single chip) and software-defined radio (SDR)-based filtering system require the filter coefficients to be changed dynamically. The use of DA-based techniques provides dynamic reconfigurability of filter coefficients in such applications, but the same may not be suitable for the MCM-based techniques. Further, based on DA concept, many researchers have proposed architectures for 1-D digital filters, for example, see [1, 3, 7, 8, 12] and references therein. However, to the best of authors' knowledge, the architecture for 2-D block FIR filters using DA algorithm has not been reported in the literature. This motivates us to carry out the present work.

This paper, therefore, proposes a novel DA-based systolic architecture for 2-D block FIR filters. In the proposed architecture, hardware-based DA-LUT (DA lookup table) and pipelining and parallelism are used to reduce the time and hardware complexities. Further, large numbers of DA-LUTs are shared at each systolic stage which reduces the count of processing elements in the architecture. Moreover, only a few parts of architecture work on the higher clock, which results in the improvement in power efficiency.

In the next section, we briefly discuss the background of block processing for 2-D FIR filters, followed by its DA formulation in Sect. 3. In Sect. 4, a design of proposed architecture is presented. Finally, a discussion on the results and comparative analysis is given in Sect. 5.

2 Block Input Processing For 2-D FIR Filter

For 2-D block FIR filter, output can be given as:

$$y^k(m,n) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x(m-i, n-k-j)h_{ij}, \tag{1a}$$

where $y^k(m,n)$ represents the $(k+1)$th output of 2-D block FIR filter, N is the size of the filter, h_{ij} is the coefficient matrix and $x(m, n-k)$ represents the $(k+1)$th input of the block inputs.

In z domain, the output can be expressed as:

$$Y^k(z_2, z_1) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} X(z_{2,}z_1)z_2^{-i}z_1^{-(k+j)}h_{ij}. \tag{1b}$$

In [11], a systolic structure to realize (1b) (for single input processing, i.e., $k=0$) has been proposed. In this structure, first a systolic architecture with respect to z_2 is designed and then systolic delayed input at each systolic level is fed to the internal systolic architecture of z_1. We now assume that an image of size $M_1 \times M_2$ is fed to the structure in [11], in row-wise scanning mode, so that for the required delayed inputs, a shift register of length $M_2 - 1$ is needed at each systolic stage with respect to z_2, where M_2 is the width of the image. Hence, z_2^{-1} in (1b) can be represented as z^{-M_2}. Further, to maintain the generality of the structure $M_2 \geq N$. To facilitate block processing, every register in M_2 shift registers should shift the input by L units, where L is the number of input samples in each iteration. Therefore, at each systolic stage, M_2 shift registers are required to be split into M_2/L equal parts as shown in Fig. 1.

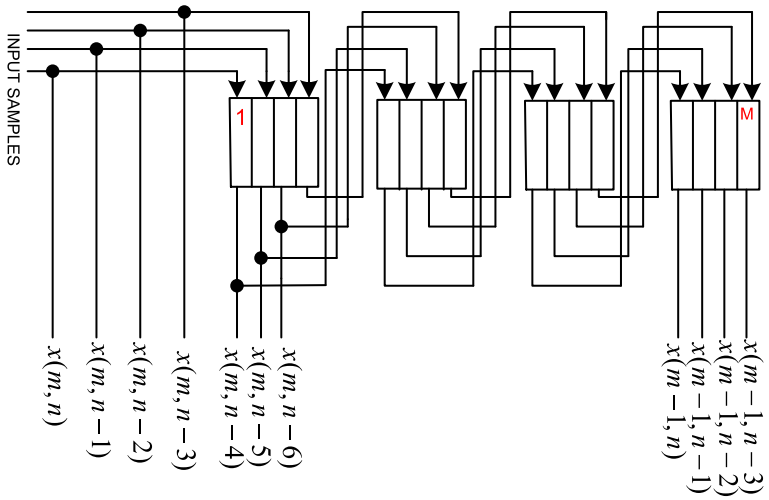

Fig. 1 Shift register block (SRB) (for $L=4$ and image width $M_2=M$)

For each input processing, an N-1 set of delayed inputs are required at each systolic stage. So for L input block processing, set of inputs require at each systolic stage can be represented in the matrix form as,

$$S_i(m,n) = \begin{bmatrix} x(m-i,n)\,x(m-i,n-1)\ldots\ldots\ldots\ldots\ldots x(m-i,n-N+1) \\ x(m-i,n-1)\,x(m-i,n-2)\ldots\ldots\ldots\ldots\ldots x(m-i,n-N) \\ x(m-i,n-2)\,x(m-i,n-3)\ldots\ldots\ldots\ldots x(m-i,n-N-1) \\ . \\ . \\ x(m-i,n-L+1)\,x(m-i,n-L)\ldots..x(m-i,n-L-N+2) \end{bmatrix},$$

where $S_i(m, n)$ represents the input-pixels required at $(i+1)$th systolic stage.

It may be noted that (1b) is not a systolic representation. So, we reorganize (1b) in the systolic form as

$$\begin{aligned} Y^k(z_2,z_1) = &\left(\sum_{j=0}^{N-1} X(z_2,z_1)z_1^{-(k+j)}h_{0j} + z^{-M_2}\left(\sum_{j=0}^{N-1} X(z_2,z_1)z_1^{-(k+j)}h_{1j}\right.\right. \\ &+\cdots z^{-M_2}\left(..+z^{-M_2}\left(\sum_{j=0}^{N-1} X(z_2,z_1)z_1^{-(k+j)}h_{N-2j}\right.\right. \\ &\left.\left.\left.\left.+z^{-M_2}\left(\sum_{j=0}^{N-1} X(z_2,z_1)z_1^{-(k+j)}h_{N-1j}\cdots\right)\right)\right)\right) \end{aligned} \tag{1c}$$

Mohanty in [9] has suggested the systolic structures based on the (1c). It is further explored in [9] that conventional systolic design for block processing contains many common delayed-input samples at each systolic stage, as highlighted in the expression of $S_i(m, n)$. Therefore, in place of internal dedicated systolic architecture for each of the block inputs, a common systolic architecture is designed to generate all the required delayed-input samples, as depicted in IRU block in Fig. 2. Using the IRU blocks in Fig. 2, a 2-D block FIR architecture is proposed in [9], which is reproduced in Fig. 3.

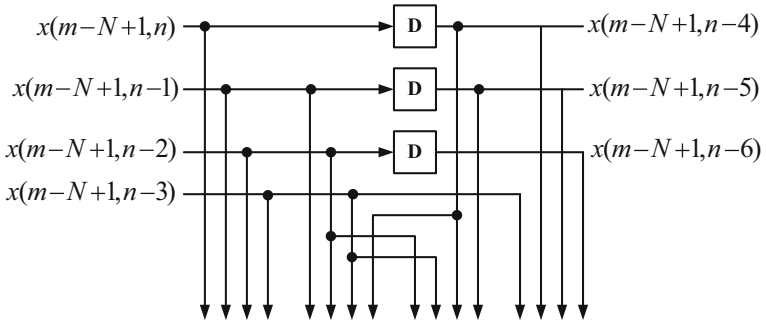

Fig.2 Internal register unit (IRU) (For N=4 and L=4)

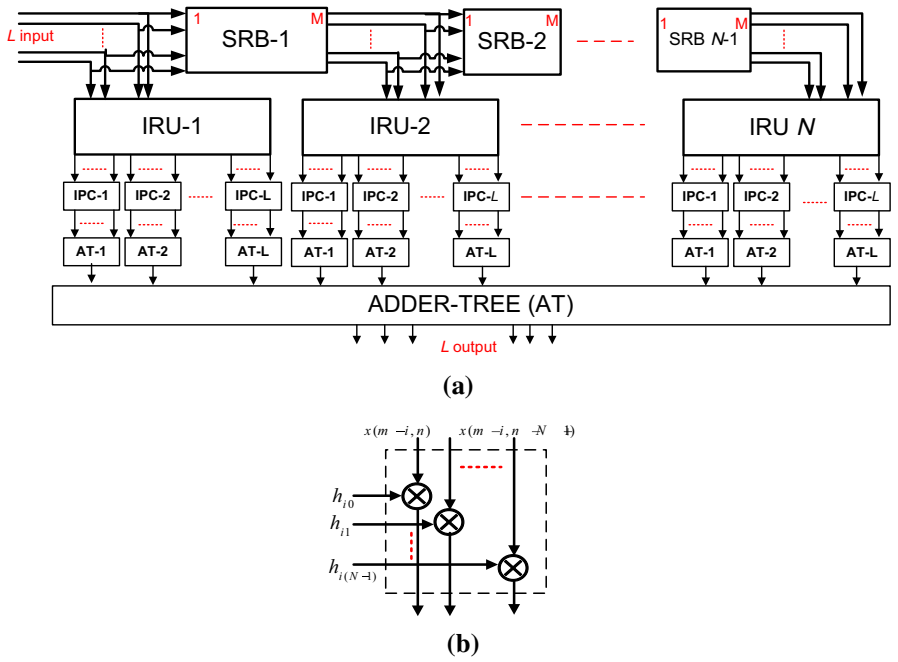

Fig.3 **a** 2-D block FIR architecture proposed in [9], **b** Inner-product computation (IPC)

A careful inspection of Fig. 3 reveals that internal systolic structures with respect to z_1 (i.e., the IRU blocks) are not needed since all the desired delayed-input samples are already present in the SRB block (see Fig. 1) at each systolic stage. Hence, these required delayed-input samples can be directly accessed from the SRB block. For the last systolic stage (Nth systolic stage), the required delayed-input samples cannot be accessed directly from SRB so an IRU block is required only for the last systolic stage to generate the desired input samples. Furthermore, the architecture in [9] (see Fig. 3) offers a longer critical path since outputs of all internal blocks are added directly using the $2\log_2 N$-stage adders tree, implying that stages in adder tree (AT) increase with the size of the filter, thereby increasing the critical path.

Now, by doing the rearrangement of delay elements in (1c), the longer critical path required in architecture in Fig. 3 can be reduced. The rearrangement of delay elements in (1c) can be expressed as:

$$
\begin{aligned}
Y^k(z_2, z_1) = &\left(\sum_{j=0}^{N-1} X(z_2, z_1) z_1^{-(k+j)} h_{0j} + z^{-(M_2-s)} \left(\left(\sum_{j=0}^{N-1} X(z_2, z_1) z_1^{-(k+j)} h_{1j}\right) z^{-s}\right.\right. \\
&+ z^{-(M_2-s)}((..)z^{-2s} \cdots + z^{-(M_2-s)} \\
&\left.\left.\left(\left(\sum_{j=0}^{N-1} X(z_2, z_1) z_1^{-(k+j)} h_{(N-2)j}\right) z^{-(N-2)s}\right.\right.\right.
\end{aligned}
$$

$$+ z^{-(M_2-s)}\left(\left(\sum_{j=0}^{N-1} X(z_2, z_1) z_1^{-(k+j)} h_{(N-1)j}) z^{-(N-1)s}\right)\cdots\right)\Bigg). \quad (1d)$$

In (1d), M_2 length delay element at each systolic stage is divided into $M_2 - s$ length delay element and s length delay element. The $M_2 - s$ length delay elements are used at the input side and s length delay elements are used at the output side of each systolic stage. At this point, it is worth mentioning that systolic reorganization of delay in (1d) can help to restrict the adder delay in critical path, e.g., $2\log_2 N$-stage adder delay in Fig. 3, which can be reduced to $\log_2 N$-stage adder delay.

Here, the authors mention that if we use DA technique along with systolic reorganization in (1d), then we can expect to achieve a much more efficient filter architecture than the one proposed in [9]. With this motivation, we consider decomposition of internal MAC using the DA algorithm and systolic delay arrangements as expressed in (1d).

3 DA Formulation

We now consider the 2-D block FIR filter given in (1) for DA formulation. Equation (1a) can be represented as:

$$y^k(m, n) = \sum_{i=0}^{N-1} y_i^k, \quad (2)$$

where

$$y_i^k = \sum_{j=0}^{N-1} x(m - i, n - k - j) h_{ij}. \quad (3)$$

To obtain DA decomposition, h_{ij} may be expressed in terms of weighted 2's complement representation. If we let the B as the bit length of filter coefficients h_{ij} and $|h_{ij}| < 1$, then

$$h_{ij} = -h_{i[0]j} + \sum_{b=1}^{B-1} h_{i[b]j} 2^{-b}. \quad (4)$$

Substituting (4) in (3), we get

$$y_i^k = \sum_{j=0}^{N-1} x(m-i, n-k-j)\left[-h_{i[0]j} + \sum_{b=1}^{B-1} h_{i[b]j} 2^{-b}\right]. \tag{5}$$

Equation (5) may be further expressed as:

$$y_i^k = \sum_{b=0}^{B-1} y_{i,b}^k 2^{-b}, \tag{6a}$$

where

$$y_{i,b}^k = \sum_{j=0}^{N-1} x(m-i, n-k-j) h'_{i[b]j}, \tag{6b}$$

and

$$h'_{i[b]j} = \begin{cases} -h_{i[0]j} \; if \; b = 0 \\ h_{i[b]j} \; if \; b \neq 0 \end{cases}.$$

In (6b), $y_{i,b}^k$ is the summation of the product of weight vector $x(m-i, n-k-j)$ and bit vector $h'_{i[b]j}$ for $0 \leq j \leq N-1$. Note that bit vector $h'_{i[b]j}$ can be either one or zero. Therefore, for $0 \leq j \leq N-1$, $y_{i,b}^k$ could be expressed as partial sum of weight vector of input samples $x(m-i, n-k)$, $x(m-i, n-k-1) \ldots\ldots x(m-i, n-k-N+1)$, and there exists 2^N different combination of bit vectors for $h'_{i[b]}$. Hence, corresponding to 2^N bit-vectors, 2^N partial sums are possible for $y_{i,b}^k$. All these possible partial sums may be precomputed and saved in an LUT, such that when the bth bit sequence of coefficients $h'_{i[b]}$ (e.g., $h'_{i[b]0}, h'_{i[b]1}, \ldots\ldots, h'_{i[b]N-1}$) is fed to the LUT as an address, the partial sum corresponding to that bit sequence can be fetched. Since $0 \leq b \leq B-1$, LUT should be accessed B times, for the computation of y_i^k. Further, the accessed partial sums can be shifted and accumulated as per (6a).

4 Proposed Architecture Design

In the proposed algorithm, a block of L input samples $(x(m-i, n), x(m-i, n-1), x(m-i, n-2), \ldots\ldots x(m-i, n-L+1))$ is processed in each iteration, to generate a block of L outputs $y^k = (y^0, y^1, y^2 \ldots\ldots y^{L-1})$. For computation of all these block outputs, we need to calculate the intermediate expressions y_i^k for $0 \leq k \leq L-1$ at every $(i+1)$th systolic stage, where $0 \leq i \leq N-1$. So, in accordance with (6b), L DA-LUT is required to compute the expressions $y^k = (y_i^0, y_i^1, y_i^2 \ldots . y_i^{L-1})$.

Further, it may be observed that input x varies from sample to sample so the partial sums cannot be stored in advance. To deal with this problem, we have organized the LUT as a combinational circuit, as shown in Fig. 4, which provides the appropriate partial sums. To generate the partial sums of DA-LUT, an adder tree-based hardware

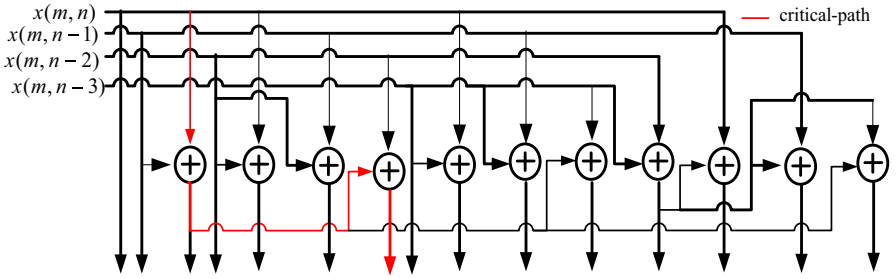

Fig. 4 Hardware-based DA-LUT (DAHLUT) (for $N = 4$)

structure (DAHLUT) is shown in Fig. 4, which comprises $(2^N - N - 1)$ adders of $\log_2 N$-stage tree. In order to explore the partial sum sharing possibilities among the DA-LUTs at every systolic stage, partial sums of DA-LUTs of $(i+1)$th systolic stages are illustrated in Fig. 5. It can be further observed from Fig. 5 that, except for the first DA-LUT, 50% of the partial sums are common among the rest of the DA-LUTs. The same is also highlighted in Fig. 5. Therefore, in place of designing individual DAHLUT for each input of block inputs, a single DAHLUT for un-highlighted partial sums in Fig. 5 is designed. Now the partial sums corresponding to $h'_{i[b]}$ are accessed by the 2^p:1 MUX, and the accessed partial sums are shifted and accumulated using shift accumulation block (SAB) as given in (6a). The detailed structure of SAB is shown in Fig. 6.

The bit vectors $h'_{i[b]}$ are fed one after another, starting from LSB to MSB, into the MUX. Since bit vectors are processed from LSB to MSB, the accumulated sum should be shifted right with each clock to get the desired shifting. The SAB architecture facilitates this shifting by feeding the sum bit of a full adder to the input of adjacent full adder, and LSB of the sum is shifted into a D flip-flop. For b = 0, $h'_{i[0]j} = -h_{i[0]j}$, so XOR gates in the architecture provide 2's complement sign convention.

As mentioned earlier, a DAHLUT block requires $(2^N - N - 1)$ adders so a total of $NL\left(2^N - N - 1\right)$ adders are required by all the DAHLUTs in the proposed architecture; however, it also adds $(\log_2 N)T_{\mathrm{PA}}$ delay in combinational path, where T_{PA} is the delay of parallel adder. As the size of filter and block input samples increases, the number of adders required for DAHLUT increases exponentially and hence the delay in the combinational path. Furthermore, increasing the size of the filter also increases the order of the MUX, which results in the increased access time.

Now, to explore the possibility of DA-LUTs sharing as suggested in [10] for 1-D adaptive filtering and to reduce the combinational delays in higher-order filters, comparable to lower-order filters, higher-order accumulation in (6b) can be divided into multiples of low-order summation. So, if we take p as the low-order summation, then $y^k_{i,b}$ in (6b) can be decomposed into $c = N/p$ low-order summations as summation as:

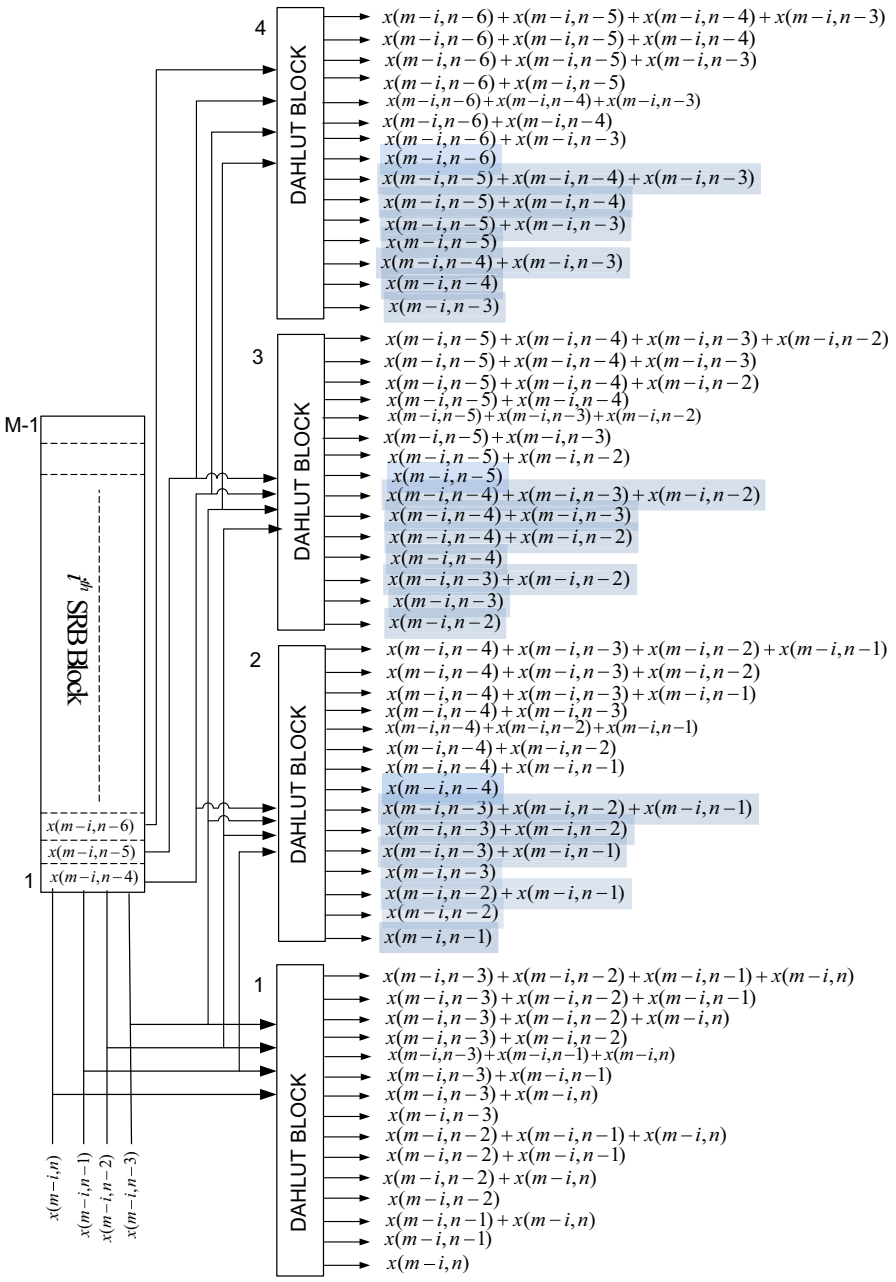

Fig. 5 Partial sums sharing for block input processing

$$y_{i,b}^{k} = \sum_{j=0}^{p-1} x(m-i, n-k-j) h'_{i[b]j} + \sum_{j=0}^{p-1} x(m-i, n-k-p-j) h'_{i[b](j+p)}$$

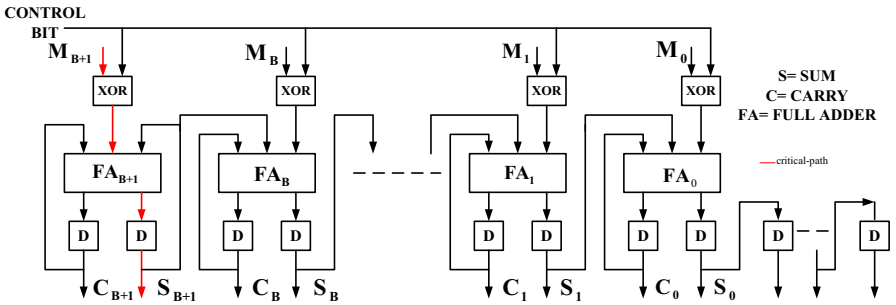

Fig. 6 Architecture of shift accumulation block (SAB)

$$+\cdots\cdots\cdots+\sum_{j=0}^{p-1} x(m-i,n-k-(c-1)p-j)h'_{i[b](j+(c-1)p)}. \tag{7}$$

It may be noted that each summation in (7) is similar to (6b), the lower-order filter of order p, and combining these summations together produces the required higher-order filter. Further, if a summation contains input weight vector which is similar to the input weight vector of another summation, then both the summations share same DA-LUT. Therefore, DAHLUT architectures will be the same for same input weight vector expressions and DAHLUT architecture will be different only for those input weight vectors which are not same. The sharing possibilities to design DAHLUT for the filter in (7) are explored below for the filter size $N=8$, $p=4$ and block size $L=8$ giving $0 \le k \le 7$.

$$y^0_{i,b} = \sum_{j=0}^{3} x(m-i,n-j)h'_{i[b]j} + \sum_{j=0}^{3} x(m-i,n-4-j)h'_{i[b](j+4)}, \tag{8a}$$

$$y^1_{i,b} = \sum_{j=0}^{3} x(m-i,n-1-j)h'_{i[b]j} + \sum_{j=0}^{3} x(m-i,n-5-j)h'_{i[b](j+4)}, \tag{8b}$$

$$y^2_{i,b} = \sum_{j=0}^{3} x(m-i,n-2-j)h'_{i[b]j} + \sum_{j=0}^{3} x(m-i,n-6-j)h'_{i[b](j+4)}, \tag{8c}$$

$$y^3_{i,b} = \sum_{j=0}^{3} x(m-i,n-3-j)h'_{i[b]j} + \sum_{j=0}^{3} x(m-i,n-7-j)h'_{i[b](j+4)}, \tag{8d}$$

$$y_{i,b}^{4} = \sum_{j=0}^{3} x(m-i, n-4-j)h'_{i[b]j} + \sum_{j=0}^{3} x(m-i, n-8-j)h'_{i[b](j+4)}, \quad (8e)$$

$$y_{i,b}^{5} = \sum_{j=0}^{3} x(m-i, n-5-j)h'_{i[b]j} + \sum_{j=0}^{3} x(m-i, n-9-j)h'_{i[b](j+4)}, \quad (8f)$$

$$y_{i,b}^{6} = \sum_{j=0}^{3} x(m-i, n-6-j)h'_{i[b]j} + \sum_{j=0}^{3} x(m-i, n-10-j)h'_{i[b](j+4)}, \quad (8g)$$

$$y_{i,b}^{7} = \sum_{j=0}^{3} x(m-i, n-7-j)h'_{i[b]j} + \sum_{j=0}^{3} x(m-i, n-11-j)h'_{i[b](j+4)}. \quad (8h)$$

It may be observed that input weight vector of second summation term in (8a), (8b), (8c) and (8d) is similar to the input weight vector of first summation term in (8e), (8f), (8g) and (8h), respectively. Therefore, instead of 16 DAHLUTs to evaluate (8), we need four DAHLUTs which are common and eight DAHLUTs which are not common.

We now consider the case of the filter size $N = 16$ with $L = 16$ and $p = 4$. Similar to (8), the filter will have 16 equations where each equation can be written as sum of $4(N/p = 16/4)$ summation expressions. It can be again observed that input weight vectors in three summations in $y_{i,b}^{0}$ and $y_{i,b}^{4}$ are common. Similar observations can be made for $y_{i,b}^{1}\&y_{i,b}^{5}$, $y_{i,b}^{2}\&y_{i,b}^{6}$, etc. After decomposition at each ith systolic level, conventionally it requires 64 DAHLUTs but due to such decomposition approach only 28 DAHLUTs are needed. It saves 36×11 adders at each systolic stage, and in overall structure, $16 \times 36 \times 11$ adders can be saved. Similarly, for $N = 32$, $L = 32$ and $p = 4$, we can observe that input weight vectors in seven summations are same between various $y_{i,b}^{k}$ (e.g., $y_{i,b}^{0}\&y_{i,b}^{4}$, $y_{i,b}^{1}\&y_{i,b}^{5}$, etc.). Alternately saying that if we design DAHLUTs for evaluating $y_{i,b}^{k}$ for $0 \leq k \leq 3$, then we need to design only one DAHLUT for the evaluation of each $y_{i,b}^{k}$ for $4 \leq k \leq 31$. In this case, almost 76.5% of DAHLUTs can be saved and approximately 37% of the adder requirement can be reduced by using partial sums sharing approach as shown in Fig. 5.

The final architecture (Fig. 7) is designed using the systolic rearrangement of delay elements as expressed in (1d). 1d shows that the inputs are delayed by M-s unit at each systolic stage and outputs are also delayed in a systolic manner by using s unit delay at each systolic stage. In the proposed design, we have considered $s = L$ and hence, at the output side, only one delay input is required at each systolic stage. The complete computational block of $(i+1)$th systolic stage of proposed design is shown in Fig. 7a. At each systolic stage, the desired DAHLUT is designed and required partial sums are fed to the corresponding processing unit (PU). After PU, one-stage pipelining is used to limit the combinational path delay. As given in (7), outputs of all the PUs are added

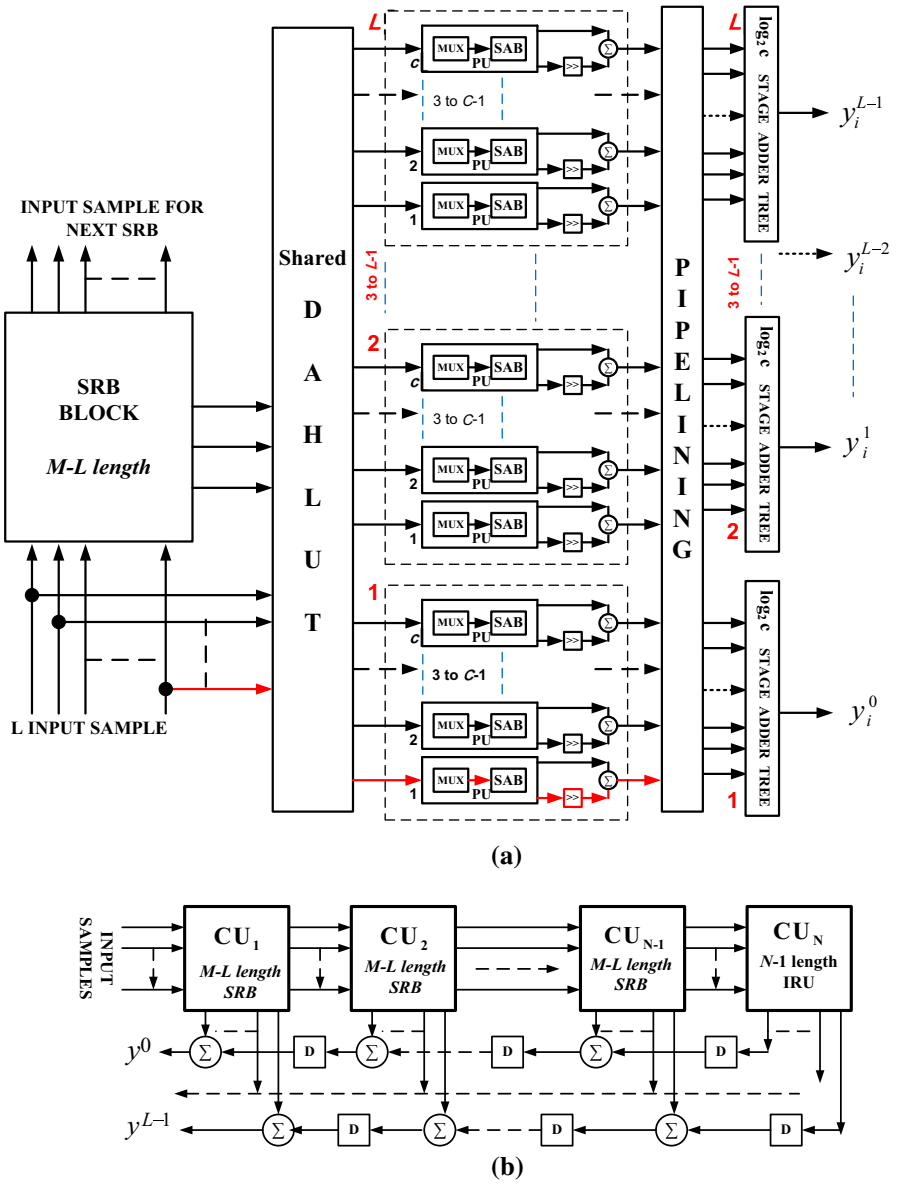

Fig. 7 **a** Computational unit (CU). **b** Block diagram of the proposed structure

together using $\log_2 c$-stage adder tree to generate the output y_i^k for $0 \leq k \leq L-1$. The final block outputs are computed by systolic shifting and adding the y_i^k as shown in Fig. 7b.

Table 1 Comparison of hardware and time complexities of the proposed structure and the structures of [4, 9]

Design	Throughput	Multipliers	Adders	Registers
Khoo [4]	$1/(T_M + 2T_{PA})$	N^2	$N^2 - 1$	$(M+N)(N-1)$
Mohanty [9]	$L/(T_M + T_1)$	LN^2	$L(N^2 - 1)$	$(M+N)(N-1)$
Proposed structure[a]	$L/(3T_{PA} + BT_{PU})$	0	$N(7(L+N) - 28) + L((3N^2/4) - 1)$	$M(N-1)+NLc$

B: bit width of filter coefficients, M: image width, T_M: delay of the multiplier, T_{PA}: delay of the parallel adder. $T_1 = T_{PA} + 2T_{FA}(\log_2 N - 1)$, $T_1 = T_{PA} + 2T_{FA}(\log_2 N - 1)$, $T_{DAHLUT} = 2T_{PA}$, $T_{PU} = T_{MUX} + T_{SAB}$, $T_{SAB} = T_{FA} + T_{XOR} + T_D$, where T_{MUX}, T_{FA}, T_{XOR} and T_D are the delay of MUX, full adder, XOR gate and D flip-flop, respectively

[a]Proposed structure requires NLc MUX, where $c = N/p$in which p is the order of small sub-block, L: block inputs and N: filter size

Table 2 Comparison of synthesis results of the proposed structure and the structures of [4, 9]

Design	N	L	MSP (ns)	Area (μm^2)	Power (mw)	ADP (μm^2 ns)	EPO (pj)	Throughput (MHz)
Khoo [4]	8	1	13.01	356,293.0216	9.3807	4,636,084	469.03	76.80
	16	1	14.65	1,154,123.0880	26.9941	16,907,903	1319.07	68.25
Mohanty [9]	8	8	11.79	1,720,962.1471	50.0106	2,537,558	312.56	678.19
	16	16	13.33	12,661,783.52	181.165	10,550,431	566.14	1200
Proposed Structure	8	8	6.53	651,615.4709	20.1069	531,880	175.66	1225.11
	16	16	6.91	4,936,081.6759	102.3432	2,131,770	319.82	2315.48

Power consumption of structures is estimated at 20 MHz frequency, and width of the image is taken $M = 64$. In [4, 9], for multiplication, Wallace tree multiplier is used. Bit width of filter coefficients is $B = 8$

MSP minimum sampling period, *ADP* area delay product (area × MSP/L), *EPO* energy per output (power × clock period/L), throughput L/MSP

5 Complexity Consideration

The hardware and time complexities of the proposed architecture and earlier reported architectures [4] and [9] for the 2-D FIR filter are listed in Table 1. It is evident that the large numbers of multipliers, which occupy most of the chip area, are required in the case of [4] and [9], whereas the proposed architecture is a multiplier-less design. Moreover, normalized registers required per input in the proposed architecture are less than those in [4], but slightly higher than in the architecture of [9]. Due to mid-stage pipelining, the complete architecture is divided into two parts, namely input stage and output stage. The input stage delay is $T_1 = T_{DAHLUT} + BT_{PU} + T_{PA}$, where T_{PU}, T_{DAHLUT} ($T_{DAHLUT} = 2T_{PA}$) and T_{PA} are the delays in PU, DAHLUT and parallel adder, respectively. Output stage delay is $T_2 = ((\log_2 c) + 1)T_{PA}$. The stage having maximum delay will decide the critical path of the structure. T_1 is independent of the size of filter, whereas T_2 increases as order of the filter increases. For the practical 2-D filters, T_1 will be greater. The architectures reported in [4, 9] and the proposed architectures in this paper are coded in Verilog HDL and synthesized by Synopsys

Design compiler using a 90-nm TSMC standard CMOS library. The generated reports are summarized in Table 2. It is evident from Table 2 that the proposed architecture has high-throughput, requires less area and consumes less power than that the architectures presented in [4, 9]. Furthermore, throughput of the proposed architectures is 80.64 and 92.95% higher than that of the architecture of [9], for filter length 8 and 16, respectively. Moreover, a reduction of 79.70% in ADP and 77.01% reduction in EPO are also observed in the proposed design as compared to [9], for filter length 16.

6 Conclusions

In this study, we have discussed the shortcomings of existing architecture [9] for the 2-D block FIR filter and explored the possibilities of improvements. Next, we have presented an efficient architecture for 2-D FIR filter for block processing using DA formulation. Memory requirement in realization of the architecture has been reduced by using memory overlapping. Further, an efficient DA-LUT sharing method is used to reduce the number of DA-LUTs which resulted in the reduction in the number of DA-LUTs by 76.50% for $N = 32$, $L = 32$ and $p = 4$. Furthermore, the requirement of adders for DAHLUT is reduced by approximately 37% by using the partial sums sharing among the DA-LUTs at each systolic stage. Moreover, for higher-order filters, DA-LUT has been divided into small-order parallel DA-LUTs, which reduces the delay in critical path, and the use of one-stage pipelining makes the critical path independent of the order of the filter. Synthesis results show that the proposed design requires 77.01% less EPO and 79.70% less ADP than the structure proposed in [9] for $N = 16$, $L = 16$ and $p = 4$.

Acknowledgements The authors would like to acknowledge the editor and anonymous reviewers for their significant contributions in the form of valuable suggestions and critics that enriched the content of this manuscript.

References

1. D.J. Allred, H. Yoo, V. Krishnan, W. Huang, D.V. Anderson, LMS adaptive filters using distributed arithmetic for high throughput. IEEE Trans. Circuits Syst. I Reg. Pap. **52**(7), 1327–1337 (2005)
2. 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 *Proceedings of the International Conference on Information, Communications and Signal Processing (ICICS)* (Tainan, Taiwan, 2013), pp. 1–4
3. R. Guo, L.S. DeBrunner, Two high-performance adaptive filter implementation schemes using distributed arithmetic. IEEE Trans. Circuits Syst. II Exp. Briefs **58**(9), 600–604 (2011)
4. 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 *Proceedings of the IEEE MWSCAS* (Seattle, WA, USA, 2010), pp. 426–529
5. P. Kumar, P.C. Srivastava, M. Tiwari, A. Dhawan, ASIC implementation of area efficient high-throughput 2-D IIR filter using distributed arithmetic. Circuits Syst. Signal Process. **37**(7), 2934–2957 (2018). https://doi.org/10.1007/s00034-017-0698-z
6. X. Lou, Y.J. Yu, P.K. Meher, Lower bound analysis and perturbation of critical path for area-time efficient multiple constant multiplications. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **36**(2), 313–324 (2017)

7. P.K. Meher, S.Y. Park, High-throughput pipelined realization of adaptive FIR based on distributed arithmetic, in *VLSI Symposium on Technical Digest* (2011), pp. 428–433
8. B.K. Mohanty, P.K. Meher, A high-performance energy-efficient architecture for FIR adaptive filter based on new distributed arithmetic formulation of block LMS algorithm. IEEE Trans. Signal Process. **61**(4), 921–932 (2013)
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 Trans. Circuits Syst. I Reg. Pap. **61**(1), 120–133 (2014)
10. B.K. Mohanty, P.K. Meher, S.K. Patel, LUT optimization for distributed arithmetic based block least mean square adaptive filter. IEEE Trans. Very Large Scale Integr. Syst. Regul. Pap. **24**(5), 1926–1935 (2016)
11. K.K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation, ch. 7* (Wiley, New York, 1999)
12. S.Y. Park, P.K. Meher, Efficient FPGA and ASIC realizations of a DA-based reconfigurable FIR digital filter. IEEE Trans. Circuits Syst. II Exp. Briefs **61**(7), 511–515 (2014)
13. M.A. Sid-Ahmed, *Image Processing: Theory, Algorithms, and Architectures* (McGraw-Hill, New York, 1995)