Implementation of Efficient Constant Multiplier Architecture based On VHBCSE Algorithm for Reconfigurable FIR Filter

Siddique I¹, Lijesh L²
M.Tech Scholar in VLSI and Embedded Systems¹, Associate Professor²
Department of ECE
Musafir College of Engineering and Technology, Kerala, India

Abstract:

This paper proposes an efficient constant multiplier architecture based on vertical-horizontal binary common sub-expression elimination (VHBCSE) algorithm for designing a reconfigurable finite impulse response (FIR) filter whose coefficients can dynamically change in real time. To design an efficient reconfigurable FIR filter, according to the proposed VHBCSE algorithm, 2-bit binary common sub-expression elimination (BCSE) algorithm has been applied vertically across adjacent coefficients on the 2-D space of the coefficient matrix initially, followed by applying variable-bit BCSE algorithm horizontally within each coefficient. This technique is capable of reducing the average probability of use or the switching activity of the multiplier block adders, as compared to that of two existing 2-bit and 3-bit BCSE algorithms respectively. ASIC implementation results of FIR filters using this multiplier show that the proposed VHBCSE algorithm is also successful in reducing the average power consumption by 32% and 52% along with an improvement in the area power product (APP) by 25% and 66% compared to those of the 2-bit and 3-bit BCSE algorithms respectively. As regards the implementation of FIR filter, improvements of 13% and 28% in area delay product (ADP) and 76.1% and 77.8% in power delay product (PDP)

Key Words: BCSE algorithm, MCM, reconfigurable FIR filter, SDR system, VLSI design.

I. INTRODUCTION

FIR filter has wide application as the key component in any digital signal processing, image and video processing, wireless communication, and biomedical signal processing systems. Moreover, systems like Software Defined Radio (SDR) [1] and multi-standard video codec [2] need a reconfigurable FIR filter with dynamically programmable filter coefficients, interpolation factors and lengths which may vary according to the specification of different standards in a portable computing platform. Significant applicability of an inefficient reconfigurable FIR filter motivates the system designer to develop the chip with low cost, power, and area along with the capability to operate at very high speed. In any FIR filter, the multiplier is the major constraint which defines the performance of the desired filter. Therefore, over the past three decades, design of an efficient hardware architecture for fixed point FIR filter has been considered as the major research focus as reported in published literatures [3]–[14]. In FIR filter, the multiplication operation is performed between one particular variable (the input) and many constants (the coefficients) and known as the multiple constant multiplication (MCM). The algorithms proposed earlier to implement this MCM for an efficient FIR filter design can be categorized in two main groups: 1) graph based algorithms and 2) common sub-expression elimination (CSE) algorithms [15]–[21]. Most of these graph based or CSE algorithms presented earlier are used to obtain efficient FIR filter hardware architecture by running the algorithms on a particular (fixed) set of coefficients for some time (a couple of hours to days) on a highly efficient computing platform (like using 1–20 number of 3.2 GHz computers in parallel mode as mentioned in [7]). However, FIR filter implementation employing effective MCM design by running these algorithms on a fixed set of coefficients is not suitable for the application like SDR system because of the following two reasons: 1) coefficient of the filters in SDR system are dynamically programmable based on requirement of different standards and 2) highly computationally efficient platform needed for those algorithms is unaffordable in SDR system. Some techniques have been introduced for efficient reconfigurable constant multiplier design [22], [23] for any application where the filter's coefficients are changing in real time e.g. multi-standard digital up/down converter. Binary common sub-expression elimination (BCSE) algorithm is one of those techniques, which introduces the concept of eliminating the common sub-expression in binary form for designing an efficient constant multiplier, and is thus applicable for reconfigurable FIR filters with low complexity [13]. However, the choice of the length of the binary common sub-expressions (BCSs) in [13] makes the design inefficient by increasing the adder step and the hardware cost. The efficiency in terms of speed, power, and area of the constant multiplier has been increased in the work presented in [14] while designing one reconfigurable FIR filter for multi-standard DUC by choosing 2-bit long BCS judiciously. Choice of the BCS of fixed length (3-bit or 2-bit) in the earlier proposed BCSE algorithm based reconfigurable FIR filter designs [13], [14] leaves a scope to optimize the designed filter by considering the BCS across the adjacent coefficients as well as within a single coefficient. The convention considered for representing the input and the coefficient of the earlier designed FIR filter [13], [14] as signed magnitude format also gives a scope to modify the data representation to signed decimal number for wider applicability of the proposed FIR filter in any systems. On studying the above-mentioned literatures, it has been realized that the development of an efficient reconfigurable constant multiplier is very much needed for its applicability in any reconfigurable system.

International Journal of Engineering Science and Computing, September 2016 3004

http://ijesc.org/
II. FIXED BIT BCSE (FBCSE) ALGORITHMS

Considering the coefficients in binary pattern, the fixed bit BCSE (FBCSE) algorithms described in [13], [14] attempt to eliminate the redundant computation vertically by considering 3-bit or 2-bit BCS present across the adjacent coefficients. As defined and explained in [23] and [25], horizontal BCSE algorithm utilizes CSs occurring within each coefficient to get rid of redundant computations, while vertical BCSE uses CSs found across adjacent coefficients to eliminate redundant computations. According to BCSE algorithm, a total of $2^n-(n+1)$ BCSs can be formed out of an n-bit binary number and the number of adders required to generate the partial products for n-bit BCS is $2^{n-1}-1[23]$. In general, the adder step in BCSE algorithm which defines the critical path can be calculated as $\log_2 n$, where n is the number of non-zero elements present within the coefficients.

In a reconfigurable constant multiplier, the coefficient values can be dynamically programmable. Therefore, the idea behind the reconfigurable multiplier is to consider the worst case (which involves the largest number of addition steps) whereby all the relatively better cases will also be taken care of. Hence, considering a reconfigurable multiplier having 16-bit input (X) and the 16-bit coefficient (H), the worst case condition will occur for the coefficient values 16'HFFFF. Shift and add based multiplication operation between the inputs (X) with this coefficient (16'HFFFF) values can be written as

$$X \times H = X \times 2^4 + X \times 2^3 + X \times 2^2 + X \times 2^1 + X \times 2^0 + X \times 2^{-7} + X \times 2^{-8} + X \times 2^{-9} + X \times 2^{-10} + X \times 2^{-11} + X \times 2^{-12} + X \times 2^{-13} + X \times 2^{-14} + X \times 2^{-15} + X \times 2^{-16}$$

(1)

where the term $2^{-16}$ is due to the 3-bit BCS and the term $2^{-13}$ due to the word-length of the coefficients being 16 [according to equations (2) and (3)]

A. Complexity Analysis of 3-Bit BCSE Algorithm

The complexity of a single constant multiplier using 3-bit BCSE algorithm is as analyzed below.

Adder Cost (AC): The number of adders required to implement the FIR filter is known as the adder cost (AC). For 3-bit BCSE algorithm, number of adders required for the shift and add units is $2^{n-1}-1$. To sum up the partial products generated from each group of BCS, $16/3$-1=5 numbers of adders are required.

Adder Step (AS): Considering the 3-bit BCSE algorithm [13] for reconfigurable FIR filter design, the number of addition operations in the chain, i.e., AS will be

$$AS3BCS=[\log_2 3]+[\log_2 (16/3)] = 2+3=5 \rightarrow (6)$$

Fig. 1. Reconfigurable constant multiplier using 2-bit BCSE algorithm [14]

Considering the BCS of 3-bit lengths the partial product generated from each BCS will be as

$$X_1=X+X \times 2^4 + X \times 2^2 \rightarrow (2)$$

Substituting equation (2) in equation (1) one gets

$$X \times H = X \times 2^4 + X \times 2^3 + X \times 2^2 + X \times 2^1 + X \times 2^0 + X \times 2^{-7} + X \times 2^{-8} + X \times 2^{-9} + X \times 2^{-10} + X \times 2^{-11} + X \times 2^{-12} + X \times 2^{-13} + X \times 2^{-14} + X \times 2^{-15} + X \times 2^{-16}$$

(3)

Considering a 2-bit BCS, the partial-product generated from each BCS will be as

$$X_0=X+X \times 2^4 \rightarrow (4)$$

Substituting (4) in (1), one gets

$$X \times H = X \times 2^4 + X \times 2^3 + X \times 2^2 + X \times 2^1 + X \times 2^0 + X \times 2^{-7} + X \times 2^{-8} + X \times 2^{-9} + X \times 2^{-10} + X \times 2^{-11} + X \times 2^{-12} + X \times 2^{-13} + X \times 2^{-14} + X \times 2^{-15} \rightarrow (5)$$

Fig. 1 depicts a reconfigurable constant multiplier that can be realized in hardware using (5). The eight terms appearing on the right hand side of (5) correspond to the eight partial products (shown as M7-M0 in Fig. 1) generated by the 2-bit BCSE algorithm given in [14]. These are summed up by the multiplier adder tree (MAT) (shown as A1-A7 in Fig. 1), leading to the product according to (5). The name MAT stems from the tree-like configuration of adders used to realize multiplication as depicted in Fig. 1.

B. Complexity Analysis of 2-Bit BCSE Technique [14]

Adder Cost (AC): Implementation of a 16-bit constant multiplier using 2-bit BCSE algorithm requires $2^{16}-1$ adder(16-bit) to generate the partial products and $16/2$-1=7 adders (16-bit) to sum up the partial products generated by each of the BCS groups.

Adder Step (AS): In 2-bit BCS based BCSE technique, the adder step can be calculated as

$$AS2BCS=[\log_2 2]+[\log_2 (16/2)] = 1+3=4 \rightarrow (7)$$

where the term $2^{16}$ is due to the 2-bit BCS and the term $2^{13}$ is due to the word-length for the coefficients being 16 [according to equations (2) and (3)].

III. ILLUSTRATIVE EXAMPLE FOR 3-BIT AND 2-BIT BCSE ALGORITHMS

Let us consider an 8-tap symmetric FIR filter with first four coefficient values as [H0=0111111101110100, H1=0110110110111010, H2=0111011011011101, and...
H3=0111111111111111]. Implementation of the desired FIR filter requires four constant multiplier blocks for multiplying the coefficients with the input. Application procedure of 3-bit BCSE on 2-D coefficient matrix consisting of these four filter coefficients of 16-bit each has been shown in Fig. 2. Application of 3-bit BCS vertically to all these coefficients requires three 16-bit adders to generate the partial products. Each multiplier considering one of the coefficients H0, H1, H2, and H3 as one input needs five adders of 16-bit each to sum up the partial products. Implementation of the desired filter requires a total number of 368 (=3 16 (to generate partial products for 3-bit binary common sub-expressions (BCS)) + 4 5 16 (to sum up the partial products for four constant multipliers corresponding to the H0, H1, H2, and H3 filter coefficients)) full adder cells. Fig. 3 shows the procedure for applying the 2-bit BCSE on the 2-D matrix of these filter coefficients mentioned above. Each constant multiplier design considering H0, H1, H2 and H3 as one input requires 1, 2, 1, 2, and 1 number of adders consisting of 17-bit, 16-bit, 13-bit, 9-bit, and 5-bit respectively to sum up the partial products which requires a total number of 85=17+32+13+18+5 full adder cells. Now implementation of the desired FIR filter requires a total number of 357 (=1 17 (for partial product) + 4 85 (to sum up the partial products for four constant multipliers)) full adder cells.

IV. PROBLEMS IN DESIGNING A RECONFIGURABLE FIR FILTER

The complexity analyses mentioned above along with the results shown in [14] demonstrate that the 2-bit BCSE algorithm is more efficient in terms of area and speed than the 3-bit BCSE algorithm presented in [13]. But these two designs encounter some problems which are mentioned below:

1) These two architectures consider the signed magnitude number format for inputs as well as coefficients. But most of the systems follow the signed decimal format data representation. This entails some changes in these architectures in order to support signed decimal data representation considering its wider applications.

2) These two architectures apply the BCSE algorithm only in the first layer (vertically on the coefficient matrix) and then the adders (A0-A7 in Fig. 1) are used to sum up the partial product generated data. This increases the probability of use of these adders present at the lower level which results in a high power consumption because of the high probability of switching activities of these adders.

3) Optimization of the multiplier adder tree (MAT), which is used for summing up the partial products, is totally ignored in these two designs.

4) If we consider the filter coefficients which consists of small decimal values with negative sign, then consumption of the hardware and power increases. For example, consider a coefficient of -2=11111111111111110 (negative signed decimal). To sum up all the partial products generated by the 2-bit BCSE, all the multiplier block adders (A0-A7) are required for one reconfigurable constant multiplier. Designing a FIR filter consisting of coefficients with positive signed high values also has the same problem. For example, consider a filter coefficient value of -65535=11111111111111111 (positive signed decimal). Implementation of the reconfigurable constant multiplier using 2-bit BCSE requires all the multiplier block adders (A0-A7) to sum up the partial products.

5) In higher order and lower order filters, there are a large number of small valued negative coefficients and higher valued positive coefficients respectively, which create the high area and power consumption problem during their hardware implementation. This problem has occurred as 2-bit or 3-bit BCSEs have been applied vertically only in the first layer of MATs according to the earlier proposed fixed-bit BCSE algorithms.

V. PROPOSED SOLUTIONS

The algorithm proposed in this paper to solve the above addressed problems consists of the following steps:

1) At first, the filter coefficient has been multiplexed between its original and complemented values depending on the most significant bit (MSB) of the coefficient to support the signed decimal data representation. This technique helps in reducing the hardware complexity when the coefficients consist of small negative decimal numbers.

2) According to the proposed algorithm in layer-1 of MAT, the 2-bit BCSE has been applied vertically followed by conditional 4-bit and 8-bit BCSEs horizontally in layer-2 and layer-3 respectively to find out the common sub-expressions (CSs) present within the coefficients. This technique helps in solving the additional hardware consumption problem by eliminating more CSs.

3) Extending the BCSE in the lower level or applying the BCSE horizontally will reduce the probability of use of the MB adders present in the lower levels of the MAT. This will reduce the power consumption to a great extent by lowering the switching activities of these MB adders.

4) Application of different lengths of 2-bit, 4-bit, and 8-bit BCSE at different layers of MAT will produce the area optimized constant multiplier during execution of the high level synthesis procedure.

5) Our proposed technique can solve the problem of high power and area consumption problem for both the cases, viz. small valued negative coefficient and high valued positive coefficient. The proposed design also supports the data representation in signed decimal format. Apart from that, area and power efficient reconfigurable FIR filter can be obtained during the high level synthesis procedure using constant multiplier based on proposed VHCSE algorithm. All of these observations make the proposed constant multiplier an excellent candidate for designing any (higher or lower) order efficient reconfigurable FIR filter.

A. PROPOSED VERTICAL-HORIZONTAL BCSE ALGORITHM

Vertical and horizontal BCSEs are the two types of BCSE used for eliminating the BCSEs present across the adjacent coefficients and within the coefficients respectively in any BCSE method. Vertical BCSE produces more effective BCSE elimination than the horizontal BCSE as shown in [26]. However, this paper proposes one new BCSE algorithm which is a combination of vertical and horizontal BCSE for designing an efficient reconfigurable FIR filter. In our proposed algorithm, a 2-bit vertical BCSE has been applied first on the adjacent coefficient, followed by 4-bit and 8-bit horizontal BCSEs to detect and eliminate as many BCSEs as possible which are present within each of the coefficient.

The procedure of application of the proposed algorithm for designing the above mentioned (in Section II) 8-Tap symmetric FIR filter has been depicted graphically in Fig. 4. Here it can be noted that the coefficients of the FIR filter form a 2-D matrix where each row represents a single coefficient and the columns correspond to individual bits of the coefficients. Application of 2-bit VCSE to these filter coefficients to generate the partial products requires one adder of 17 full adder cells. Application of 4-bit and 8-bit
The proposed VHBCSE algorithm based constant multiplier uses 2-bit BCSE vertically instead of 3-bit BCSE because of its efficiency as shown in Section II. Our modified algorithm can work for signed decimal number of both the input and the coefficients along with a reduced probability of use of the adders (A0-A7 shown in Fig. 1) to sum up the partial product generator by extending the BCSE at the lower level.

The proposed algorithm is made of the following steps:

1) Get the input of 16-bits (x[15:0]).

2) Store coefficients of 17-bits (h[16:0]) in LUT.

3) Take 1’s complement of the 16-bits (except the MSB) coefficient (h[15:0]).

4) If the MSB, the 17th bit (h[16]) of the coefficient is 1, then choose the complemented version of coefficient (h'[15:0]); else consider the original coefficient (h[15:0]) to produce the multiplexed coefficient (hm[15:0]).

5) Partition the multiplexed coefficient (MC) (hm[15:0]) into fixed groups of 2 bits each and use these groups as the select lines to the corresponding multiplexer (M7-M0) at layer-1 as shown in Fig. 1.

6) Partition the MC into groups of 4 bits each (hm[15:12], hm[11:8], hm[7:4], hm[3:0]).

7) Compare hm[15:12] with hm[11:8] in the layer 2. If match is found then skip the output of the adder A2. Instead, use shifted (by 4-bit right) version of the first addition (A1) output as input to the adder A5. Otherwise, take the output of the adder A2.


9) Compare hm[15:12] with hm[3:0] in the layer-2. If match is found then skip the output of the adder A4. Instead use shifted (by 12-bit right) version of the output of the adder (A1) as input to the adder A6. Otherwise compare hm[11:8] with hm[3:0] in the layer-2. If match is found then skip the output of the adder (A4). Instead use shifted (by 8-bit right) version of the output of the adder (A2) as input to the adder A6. Otherwise, compare hm[7:4] with hm[3:0] in the layer-2. If match is found then skip the output of the adder (A4). Instead use shifted (by 4-bit right) version of the output of the adder (A3) as input to the adder A6. Else use the output of the adder (A4).

10) Partition the MC into fixed group of 8-bit (hm[15:8], hm[7:0]).

11) Compare hm[15:8] with hm[7:0] in the layer 3. If match is found then skip the output of the adder A6. Instead use shifted (by 8-bit right) version of the output of the adder (A5) as input to the adder A7. Else take the output of the adder A6.

12) Obtain the final addition result by performing 1-bit right shift on the output of the adder A7 (CFC).

13) Take 2’s complement of the output of A7 (CFCM[15:0]).

14) If the MSB the 17th bit (h[16]) of the coefficient is 1, then choose the complemented version (CFCM). Otherwise consider the original of the result of A7 (CFC).

15) Multiplication is completed. Store this result, h*x, in the register.

---

**B.COMPLEXITY ANALYSIS OF VHBCSE**

Adder Cost: Implementation of a 16-bit constant multiplier using 2-bit BCSE algorithm requires adder (16-bit) to generate the partial products and adders (16-bit) to sum up partial products generated by each of the BCS group. Adder Step: In the proposed VHBCSE algorithm based constant multiplier the adder step can be defined as

\[
\text{LD2BCS} = \log_2 2 + \log_2 (16/2) = 1 + 3 = 4 \rightarrow (8)
\]

where the term \(\log_2 2\) is due to the 2-bit BCS and the term \(\log_2 (16/2)\) is due to the fact that the word-length for the coefficients has been considered of 16 bits. Table I shows the summary of hardware complexities and the propagation delays required for the proposed VHBCSE, 2-bit BCSE [14] and 3-bit BCSE algorithms [13].

Table II shows the calculated probability of utilization (Total P) of adders for generating (A0) and summing up (A1-A7) the partial products in shift and add based constant multiplier designed by using 2-bit BCSE, 3-bit BCSE, and the proposed VHBCSE algorithms for different values of the constant coefficients. Validation of the calculated probability of utilization (TotalP) of the adders (A0-A7) has been done by performing the addition of each of the results (CFCM).
been performed using MATLAB by calculating the total number of adders/logic operators required for all the possible cases (\(2^{16}=65536\) for a 1-bit coefficient) of constant coefficient (Total AC). These results (TotalPp, and Average AC) clearly establish that the use of the VHBCSE reduces the switching activities or the probability of utilization of the adders (A0-A7) by 6.2% and 19.6% than those of the 2-bit and 3-bit BCSE method respectively. Reduction in the switching activity for the proposed case results in reduction in the power consumption of the designed FIR filter hardware.

### TABLE I
### COMPLEXITY ANALYSES OF THE DIFFERENT BCSE ALGORITHMS

<table>
<thead>
<tr>
<th>Type of Algorithm</th>
<th>Adder (number of adders)</th>
<th>Partial Adders (number of adders)</th>
<th>Functionality of Adders</th>
<th>Hardware (number of adders)</th>
<th>Power (number of adders)</th>
<th>Propagation Delay (number of adders)</th>
<th>Memory (number of adders)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-bit</td>
<td>3</td>
<td>10</td>
<td>High</td>
<td>7</td>
<td>2.5</td>
<td>3.2</td>
<td>1</td>
</tr>
<tr>
<td>3-bit</td>
<td>4</td>
<td>16</td>
<td>Low</td>
<td>8</td>
<td>3.5</td>
<td>4.3</td>
<td>2</td>
</tr>
<tr>
<td>Proposed</td>
<td>1</td>
<td>4</td>
<td>Low</td>
<td>8</td>
<td>3.5</td>
<td>4.3</td>
<td>2</td>
</tr>
<tr>
<td>Total AC</td>
<td>5</td>
<td>28</td>
<td>Low</td>
<td>24</td>
<td>15</td>
<td>17.3</td>
<td>6</td>
</tr>
</tbody>
</table>

### TABLE II
### PROBABILITY OF USE OF THE MB ADDER (A0-A7) IN DIFFERENT LAYERS FOR IMPLEMENTING A BCSE BASED MULTIPLIER

<table>
<thead>
<tr>
<th>Type of Algorithm</th>
<th>P of Adders in Unit</th>
<th>P of Adders in Unit</th>
<th>P of Adders in Unit</th>
<th>P of Adders in Unit</th>
<th>P of Adders in Unit</th>
<th>Total Pr</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-bit</td>
<td>0.525</td>
<td>0.525</td>
<td>0.525</td>
<td>0.525</td>
<td>0.525</td>
<td>0.525</td>
</tr>
<tr>
<td>3-bit</td>
<td>0.707</td>
<td>0.707</td>
<td>0.707</td>
<td>0.707</td>
<td>0.707</td>
<td>0.707</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.525</td>
<td>0.525</td>
<td>0.525</td>
<td>0.525</td>
<td>0.525</td>
<td>0.525</td>
</tr>
<tr>
<td>Total AC</td>
<td>0.525</td>
<td>0.525</td>
<td>0.525</td>
<td>0.525</td>
<td>0.525</td>
<td>0.525</td>
</tr>
</tbody>
</table>

### VI. ARCHITECTURE OF THE PROPOSED VHBCSE ALGORITHM BASED CONSTANT MULTIPLIER

The data flow diagram of the proposed vertical-horizontal BCSE algorithm based constant multiplier (CM) design is shown in Fig. 5. The designed multiplier considers the length of the input (Xin) and coefficient (H) as 16-bit and 17-bits respectively while the output is assumed to be 16-bit long. Herein, the sampled inputs are stored in the register first and then the coefficients are stored directly in the LUTs. Functionality along with hardware architecture of different blocks of the designed VHBCSE based multiplier are explained below in details.

1) Sign Conversion Block: Sign conversion block is needed to support the signed decimal format data representation for both the input and the coefficient. The architecture of the sign conversion block is shown in Fig. 6. There is one’s complementer circuit to generate the inverted version of the 16-bit (excluding MSB) coefficient. One 16-bit 2:1 multiplexer produces the multiplexed coefficients depending on the value of the most significant bit (MSB) of the coefficient. For negative value of the original coefficient, the multiplexed coefficient will be in the inverted form; otherwise it will be as it is.

2) Partial Product Generator (PPG): In BCSE method, shift and add based technique has been used to generate the partial product which will be summed up in the following steps/layers for producing the final multiplication result. Choice of the size of the BCS defines the number of partial products. In the proposed algorithm in the layer-1, 2-bit binary common sub-expressions (BCSs) ranging from “00” to “11” have been considered, which will produce 4 partial products. But, within four of these BCSs, a single adder (A0) will be required to generate the partial product only for the pattern “11”, the rest will be generated by hardwired shifting. For the coefficient of 16-bit length, 8 partial products of 17, 15, 13, 11, 9, 7, 5, and 3 bits (P8-P1) will be generated by right shifting the first partial product (P8) by 0, 2, 4, 6, 8, 10, 12, and 14 bits respectively. This technique helps in reducing the multiplexer’s size which is used next to select the proper partial product depending on the coefficient’s binary value. The architecture of this block is shown in Fig. 7.

3) Control Logic (CL) Generator: Control logic generator block takes the multiplexed coefficient (Hm[15:0]) as its input and groups it into one of 4-bit each (Hm[15:12], Hm[11:8], Hm[7:4], and Hm[3:0]) and another of 8-bit each (Hm[15:8], Hm[7:0]). According to the algorithm mentioned in Section IV, the CL generator block will produce 7 control signals depending on the equality check for 7 different cases. The architecture for the control signal generator block is shown in Fig. 8. The control signal for 8-bit equality check is seen to be produced through the control signals generated from the 4-bit equality check.

4) Multiplexers Unit: The multiplexer unit is used to select the appropriate data generated from the PPG unit depending on the coefficient’s binary value. At layer-1, eight 4:1 multiplexers are required to produce the partial products according to the 2-bit BCSE algorithm applied vertically on the MAT. The widths of these 8 multiplexers are 17, 15, 13, 11, 9, 7, 5, and 3-bit each instead of 16-bit for all, which would reduce the hardware and power consumption. The architecture of the multiplexers unit is shown in Fig. 11.

5) Controlled Addition at Layer-2: The partial products (PP) generated from eight groups of 2-bit Bih control signal (C7) which has been generated based on 8-bit BCSE from the CS generator block. From Fig. 10 it is concluded that the propagation delay will be maximum between the paths which has been used to generate AS2,AS3,AS4,i.e.,\(\max\{t_{\text{1bitadd}}+2t_{\text{1max}}(t_{\text{1bitadd}}+2t_{\text{2max}}),t_{\text{bitadd}}+3t_{\text{1max}}\}\) 6) Controlled Addition at Layer-3: The four multiplexed sums (AS1, AS2, AS3 and AS4) generated from layer-2 are now summed up in layer-3. In our algorithm, controlled additions are performed, instead of direct addition of these four sums as shown in Fig. 10. Hence, this addition (A6) is controlled by the control signal (C7) which has been generated based on 8-bit BCSE from the CS generator block. From Fig. 10 it is concluded that the propagation delay will be \(t_{\text{1bitadd}}+2t_{\text{1max}}\)

7) Final Addition on Layer-4: This block performs the addition operation between the two sums (AS5-AS6) produced by layer-3 to finally produce the multiplication result between the input and the coefficient. The block diagram of the over-all constant multiplication is shown in Fig. 11.
The VHBCSE algorithm based constant multiplier architecture, shown in Fig. 11, has been coded using Verilog hardware description language to synthesize in the targeted FPGA device using Xilinx ISE 9.2i synthesis tool. As there is trade-off between area and delay, for the fair comparison in Table III, the normalized slice-delay product [N(SDP)] using the equation (9) has been considered. The results depicted in Table III indicate that the N(SDP) for the present design are 21% and 84.6% better than those of earlier reported filter design using 2-bit BCSE [14] and 3-bit BCSE [13] respectively on XC2V3000-4FF1152 FPGA device. The proposed design possesses 31.7% and 63.3% improvement in the normalized SDP than DA-based FIR filter [12] and Xilinx FIR compiler [27] respectively while implemented on Virtex-5 FPGA device. Comparisons of the implementation results on Virtex-4 FPGA device with those of [8] reveal 57% improvement for the proposed VHBCSE algorithm.

\[
N(SDP) = \frac{SLUT \times MSP}{No. of Tap \times input\_wordlength} \times \frac{16}{coeff\_wordlength} (9)
\]

To show the efficiency of the proposed design in terms of power consumption on Xilinx XC2V2000E FPGA platform, three linear phases FIR filters comprising 16-Tap, 32-Tap, and 64-Tap have been designed by using VHBCSE algorithm based constant multiplier. Xilinx XPower EDA tool has been used to calculate the dynamic power consumption of each of these filters based on the switching activity file generated during the post-place and route simulation by running the design's clock, inputs and outputs at 50 MHz frequency by following the same specification used in [11]. The results of implementing three filter having different lengths, which are presented in Table IV, indicate that the proposed algorithm improves the power consumption by 20.8%, 10.4%, and 3% than that of the 2-bit BCSE algorithm based FIR filter [14] designed earlier whereas 22.5%, 42.7%, and 61.5% improvements have been found as compared to the DA-based FIR filter design mentioned in [11].

Fig. 5. Data flow diagram of the CM using VHBCSE algorithm

Fig. 6. Hardware architecture of the Sign Conversion Block

Fig. 7. Block diagram of the Partial Product Generator Unit.

To provide more clarification, we have designed one 20-tap symmetric reconfigurable FIR filter by using the proposed VHBCSE, 2-bit BCSE, and 3-bit BCSE algorithm based reconfigurable constant multipliers, considered as the generic case (where the inputs and the coefficients are not fixed to any values). The implementation results for the generic case listed in Table V show that the area power products (APP) are 16% and 60% less for the proposed design than that of the 2-bit BCSE and 3-bit BCSE algorithm based designs respectively. However, we have designed six 20-tap FIR filters (FIR1-FIR6) with different specifications for demonstrating the effectiveness of the proposed VHBCSE algorithm for area and delay profiling for the proposed algorithm along with the 2-bit and 3-bit BCSE algorithm. FIR1 has pass-band 0.15π and stop-band 0.25π respectively whereas FIR2 has pass-band of 0.02π and stop-band of 0.07π, both of them being low-pass filter. FIR3 is a high-pass filter with stop-band and pass-band of 0.37π and 0.5π respectively. FIR4 is also a high-pass filter with the pass-band and stop-band frequencies of 0.6173π and 0.6276π respectively. FIR5 is the example of a band-pass filter whose pass-band1, pass-band2, and stop-band2 are 0.15π,0.17π,0.25π and 0.27π respectively. It can be noted from Table V that our proposed algorithm improves the average APP by 32.6% and 26.6% for FIR1-FIR6 as compared to that of the 2-bit BCSE and 3-bit BCSE algorithm based design respectively.

Fig. 8. Block diagram of the control logic generator unit.

We have redesigned the earlier reported works on fixed-bit vertical BCSE algorithm, namely 2-bit and 3-bit BCSE and applied to seven different filter coefficient sets of 16, 24, 32, 40, 48, 56, and 64 tap low pass equiripple FIR filter based on the proposed VHBCSE algorithm. The coefficients are generated using MATLAB FDA tool and the performance measures viz. speed, power and area of each of these filters have been calculated by using Synopsys design compiler EDA tool along with Faraday 90 nm technology library. Table VI presents the results of performance comparison in terms of area and power consumption by keeping the clock period fixed at 5 ns. These results show that our proposed VHBCSE algorithm saves 32% and 52% average power.
when compared to those of 2-bit BCSE presented in [14] and 3-bit BCSE algorithm of [13] based FIR filter design respectively. Our proposed design consumes 2% more average area in comparison to 2-bit BCSE because of some modification in the algorithm for supporting signed decimal data representation whereas it achieves 29.4% more average area savings than that of 3-bit BCSE algorithm. Irrespective of all the modifications done in the proposed algorithm to support wide range of applicability in various systems, 25% and 66% improvement in average area power product (APP) than 2-bit BCSE and 3-bit BCSE algorithm respectively shows the efficiency of the proposed VHBCSE algorithm over these two algorithms. From Table VI it is concluded that the higher the filter order is, the higher will be the amount of saving in power consumption for the proposed algorithm over the other two algorithms.

The comparisons of our proposed algorithm's results and those of other reported works on FIR filter design implemented on ASIC platform using Faraday 90 nm technology library in terms of area per tap (A(tap)) and area delay product (ADP) are shown in Table VII. It is to be noted that different authors considered different filter specifications e.g. different length of the filters, word length of the input and the coefficients, along with different clock frequencies to calculate the power consumption. Accordingly, the area and power per tap (A(tap) and P(tap)) in Table VII have been normalized using (10) and (11)

\[
A(tap) = \frac{\text{Total.Area}}{\text{No.of.Taps}} \times \frac{16}{\text{Input.Wordlength}} \times \frac{16}{\text{Coeff.Wordlength}} \times \frac{100}{\#\text{No.of.Taps}} \times \frac{16}{\#\text{Input.Wordlength}} \times \frac{\#\text{Coeff.Wordlength}}{\text{Clk.Freq}}
\]

\[
P(tap) = \frac{\text{Total.Power}}{\#\text{No.of.Taps}} \times \frac{16}{\text{Input.Wordlength}} \times \frac{16}{\#\text{Coeff.Wordlength}}
\]

TABLE III

<table>
<thead>
<tr>
<th>Design</th>
<th>Target Device</th>
<th>FD</th>
<th>MSP (ns)</th>
<th>SREG</th>
<th>SLUT</th>
<th>N(SDP)</th>
<th>Imp. in SDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prop. 4</td>
<td>XC2V1000/0</td>
<td>20-Tap</td>
<td>16/17</td>
<td>16.2</td>
<td>62</td>
<td>320</td>
<td>588</td>
</tr>
<tr>
<td>Prop. 4</td>
<td>XC2V1000/0</td>
<td>49-Tap</td>
<td>16/17</td>
<td>16.0</td>
<td>6.5</td>
<td>50</td>
<td>N/M</td>
</tr>
<tr>
<td>Prop. 4</td>
<td>XC2V1000/0</td>
<td>49-Tap</td>
<td>16/17</td>
<td>16.0</td>
<td>8</td>
<td>24</td>
<td>N/M</td>
</tr>
<tr>
<td>[12] DA-Based</td>
<td>XC3VEX 9T</td>
<td>16-Tap</td>
<td>16/17</td>
<td>7.10</td>
<td>130</td>
<td>320</td>
<td>692</td>
</tr>
<tr>
<td>[12] DA-Based</td>
<td>XC3VEX 9T</td>
<td>16-Tap</td>
<td>16/17</td>
<td>7.10</td>
<td>130</td>
<td>320</td>
<td>692</td>
</tr>
<tr>
<td>[12] DA-Based</td>
<td>XC3VEX 9T</td>
<td>16-Tap</td>
<td>16/17</td>
<td>7.10</td>
<td>130</td>
<td>320</td>
<td>692</td>
</tr>
<tr>
<td>[12] DA-Based</td>
<td>XC3VEX 9T</td>
<td>16-Tap</td>
<td>16/17</td>
<td>7.10</td>
<td>130</td>
<td>320</td>
<td>692</td>
</tr>
<tr>
<td>[12] DA-Based</td>
<td>XC3VEX 9T</td>
<td>16-Tap</td>
<td>16/17</td>
<td>7.10</td>
<td>130</td>
<td>320</td>
<td>692</td>
</tr>
<tr>
<td>[12] DA-Based</td>
<td>XC3VEX 9T</td>
<td>16-Tap</td>
<td>16/17</td>
<td>7.10</td>
<td>130</td>
<td>320</td>
<td>692</td>
</tr>
<tr>
<td>[12] DA-Based</td>
<td>XC3VEX 9T</td>
<td>16-Tap</td>
<td>16/17</td>
<td>7.10</td>
<td>130</td>
<td>320</td>
<td>692</td>
</tr>
<tr>
<td>[12] DA-Based</td>
<td>XC3VEX 9T</td>
<td>16-Tap</td>
<td>16/17</td>
<td>7.10</td>
<td>130</td>
<td>320</td>
<td>692</td>
</tr>
<tr>
<td>[12] DA-Based</td>
<td>XC3VEX 9T</td>
<td>16-Tap</td>
<td>16/17</td>
<td>7.10</td>
<td>130</td>
<td>320</td>
<td>692</td>
</tr>
<tr>
<td>[12] DA-Based</td>
<td>XC3VEX 9T</td>
<td>16-Tap</td>
<td>16/17</td>
<td>7.10</td>
<td>130</td>
<td>320</td>
<td>692</td>
</tr>
</tbody>
</table>

FD: Filter Description, MSP: Maximum Sampling Period, MSF: Maximum Sampling Frequency, SREG: Slice Register, SLUT: Slice Look-Up-Table, N(SDP): Normalized Slice Delay Product, N/M: Not Mentioned

Fig. 11. Proposed Reconfigurable constant multiplier architecture.

Table VII reveals that the proposed architecture achieves 69%, 56%, and 59% less maximum sampling period (MSP) than those of the FIR filter designed earlier based on multiple constant multipliers/accumulators with faithfully rounded truncation (optimized), information theoretic based approach and LUT based multiplier respectively. In comparison to the DA-based reconfigurable FIR filter which was designed considering 8-bit input and coefficient word length, the proposed design (designed considering 16-bit input and 17-bit coefficient word length) has more MSP by 18.6%. The proposed design has 64.7% and 44.5% more area than [28] and [29] respectively. This is because of the fact that our design is an example of single constant multiplication whereas the two referred works [28] and [29] are examples of multiple constant multiplications. The proposed design consumes 59% and 69% less area per tap than those of the works reported in [12] and [11] respectively, which are examples of single constant multiplications. In terms of area delay product (ADP), the proposed architecture is found to outperform the work reported in [28], [29], [12] and [11] by 13%, 28%, 50%, and 88% respectively. Table VII clearly shows that P(tap) for the proposed design is better than each and every design presented herein. The efficiency in terms of power delay product (PDP) of the proposed design over the FIR filter design presented in [28], [29], [12] and [11] has been found out to be 76.1%, 77.8%, 70.2%, and 92.2% respectively.
TABLE IV
COMPARISON RESULTS FOR POWER CONSUMPTION ON XCV2000E

<table>
<thead>
<tr>
<th>Prop.</th>
<th>Length</th>
<th>CP (W)</th>
<th>LP (W)</th>
<th>TP (W)</th>
<th>Improvement in LP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simit</td>
<td>16 Taps</td>
<td>13.04</td>
<td>68.33</td>
<td>81.36</td>
<td>23.21</td>
</tr>
<tr>
<td>[1]</td>
<td>16 Taps</td>
<td>13.04</td>
<td>68.33</td>
<td>81.36</td>
<td>23.21</td>
</tr>
<tr>
<td>[1]</td>
<td>16 Taps</td>
<td>23.65</td>
<td>61.12</td>
<td>84.77</td>
<td>34.7</td>
</tr>
<tr>
<td>Prop.</td>
<td>32 Taps</td>
<td>19.26</td>
<td>65.62</td>
<td>84.88</td>
<td>21.63</td>
</tr>
<tr>
<td>[1]</td>
<td>32 Taps</td>
<td>18.66</td>
<td>74.98</td>
<td>93.64</td>
<td>37.6</td>
</tr>
<tr>
<td>[1]</td>
<td>32 Taps</td>
<td>42.61</td>
<td>106.89</td>
<td>149.50</td>
<td>38.7</td>
</tr>
<tr>
<td>Prop.</td>
<td>64 Taps</td>
<td>35.28</td>
<td>72.39</td>
<td>107.67</td>
<td>20.04</td>
</tr>
<tr>
<td>[1]</td>
<td>64 Taps</td>
<td>35.28</td>
<td>72.39</td>
<td>107.67</td>
<td>20.04</td>
</tr>
<tr>
<td>[1]</td>
<td>64 Taps</td>
<td>54.09</td>
<td>82.72</td>
<td>136.81</td>
<td>58.46</td>
</tr>
<tr>
<td>[1]</td>
<td>64 Taps</td>
<td>105.38</td>
<td>141.40</td>
<td>246.78</td>
<td>58.46</td>
</tr>
</tbody>
</table>


TABLE V
COMPARISON OF THE ASIC IMPLEMENTATION RESULTS FOR REALIZING DIFFERENT FIR FILTERS (FIR1-FIR6)

<table>
<thead>
<tr>
<th>Filter Length</th>
<th>Power (W)</th>
<th>Area (um²)</th>
<th>Improvement in ADP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Granule</td>
<td>3 taps</td>
<td>41.92</td>
<td>41.32</td>
</tr>
<tr>
<td>[1]</td>
<td>1.07</td>
<td>41.92</td>
<td>41.32</td>
</tr>
<tr>
<td>[1]</td>
<td>2.07</td>
<td>83.73</td>
<td>83.32</td>
</tr>
<tr>
<td>[1]</td>
<td>1.07</td>
<td>167.66</td>
<td>167.32</td>
</tr>
<tr>
<td>[1]</td>
<td>2.07</td>
<td>335.32</td>
<td>334.32</td>
</tr>
<tr>
<td>[1]</td>
<td>1.07</td>
<td>670.66</td>
<td>669.32</td>
</tr>
<tr>
<td>[1]</td>
<td>2.07</td>
<td>1341.32</td>
<td>1339.32</td>
</tr>
<tr>
<td>[1]</td>
<td>1.07</td>
<td>2682.66</td>
<td>2681.32</td>
</tr>
<tr>
<td>[1]</td>
<td>2.07</td>
<td>5365.32</td>
<td>5364.32</td>
</tr>
</tbody>
</table>

NC: non-combinatorial/sequential components, C: combinatorial

TABLE V
COMPARISON OF THE ASIC IMPLEMENTATION RESULTS FOR REALIZING DIFFERENT FIR FILTERS (FIR1-FIR6)

<table>
<thead>
<tr>
<th>Filter Length</th>
<th>Power (W)</th>
<th>Area (um²)</th>
<th>Improvement in ADP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Granule</td>
<td>3 taps</td>
<td>41.92</td>
<td>41.32</td>
</tr>
<tr>
<td>[1]</td>
<td>1.07</td>
<td>41.92</td>
<td>41.32</td>
</tr>
<tr>
<td>[1]</td>
<td>2.07</td>
<td>83.73</td>
<td>83.32</td>
</tr>
<tr>
<td>[1]</td>
<td>1.07</td>
<td>167.66</td>
<td>167.32</td>
</tr>
<tr>
<td>[1]</td>
<td>2.07</td>
<td>335.32</td>
<td>334.32</td>
</tr>
<tr>
<td>[1]</td>
<td>1.07</td>
<td>670.66</td>
<td>669.32</td>
</tr>
<tr>
<td>[1]</td>
<td>2.07</td>
<td>1341.32</td>
<td>1339.32</td>
</tr>
<tr>
<td>[1]</td>
<td>1.07</td>
<td>2682.66</td>
<td>2681.32</td>
</tr>
<tr>
<td>[1]</td>
<td>2.07</td>
<td>5365.32</td>
<td>5364.32</td>
</tr>
</tbody>
</table>


VIII. CONCLUSIONS
With a view to implementing an efficient fixed point reconfigurable FIR filter, this paper presents one new vertical-horizontal BCSE algorithm which removes the initial common sub-expressions (CSs) by applying 2-bit BCSE vertically. Further elimination of the CSs has been performed through finding the CSs present within the coefficients by applying BCSEs of different lengths horizontally to different layers of the shift and add based constant multiplier architecture. It has been shown that the proposed algorithm successfully reduces the average switching activities of the multiplier block adder by 6.2% and 19.6% while compared to those of 2-bit and 3-bit BCSEs (fixed-bit vertical BCSE) respectively. Reduction of switching activities during hardware implementations of different FIR filters results in lowering the average power consumption by 32% and 52% relative to these two algorithms respectively. Implementation results reveal that there are considerable amount of power savings for higher order filter as a large number of matches can be found for more number of coefficients. The proposed VHBCSE algorithm establishes improvements of efficiency of 13% and 28% in area delay product (ADP) and 81.6% and 82.9% in power delay product (PDP) when compared to those of earlier proposed MCM algorithm based FIR filter. Maximizing the efficiency and supporting the signed decimal data representation for both the input and coefficient make the proposed constant multiplier based on VHBCSE algorithm more suitable for next generation efficient systems like software defined radio.

IX. REFERENCES


http://ijesc.org/


