Design of Low Power and Area Efficient ESPFFIR Filter Using Multiple Constant Multiplier

R. Solomon Roach¹, N. Nirmal Singh²

¹Department of ECE, Cape Institute of Technology, Tamilnadu-627114, India
E-mail: solroach@gmail.com

²Department of ECE, VV College of Engineering, Tamilnadu-627657, India
E-mail: n_nirmalsingh@hotmail.com

Abstract:
In Even Symmetric Parallel Fast Finite Impulse Response (ESPFFIR) filter adders and multipliers are the major processing elements. Multiple Constant Multipliers and low power adders reduce the area and power consumption in most of the Digital Signal Processing elements. This paper focuses on the design of ESPFFIR filter by replacing the existing Canonical Signed Digit (CSD) based Single Constant Multiplier (SCM) by Common Sub-expression Elimination (CSE) based Multiple Constant Multiplier (MCM). Carry Save Adder (CSA) is being replaced by Modified CSLA with a view to consume less power and to occupy less area. The Modified CSLA adders consume 20% less power and 15% less area than Carry Save Adder (CSA). Various CSE based algorithms like Heuristic of cumulative benefit (Hcub), n-dimensional Reduced Adder Graph (RAG-n), and Bull Horrocks Modified algorithm (BHM) have been used for designing Multiple Constant Multipliers, which are the major part in the design of ESPFFIR filter. While these algorithms are used for the design of Multiple Constant Multipliers, the CSE-Hcub-n use 40% less number of adders than CSD based SCM multipliers. It is understood that the CSE-Hcub-n algorithm is superior to other algorithms in designing the subfilters in the ESPFFIR filter. Here modified 12-tap 2-parallel ESPFFIR filter & 24-tap 2-parallel ESPFFIR filter have been compared together in power and area wise. The modified 12-tap 2-parallel ESPFFIR filter & 24-tap 2-parallel ESPFFIR filter consume 18% and 19.2% less area and power than the conventional design. Analysed using Cadence RC synthesize tools.

Keywords – Heuristic of cumulative benefit (Hcub), n-dimensional Reduced Adder Graph (RAG-n), Bull Horrocks Modified algorithm (BHM), Single Constant Multiplication (SCM).

1. INTRODUCTION

Due to the rapid evolution of battery operated electronic devices, the necessity of low power and area efficient Digital Signal Processing (DSP) systems increases rapidly day by day. Finite-impulse response (FIR) digital filters are the most extensively used datapath element in DSP systems, ranging from wireless communications to audio, video and medical signal processing. Applications like audio, video and image processing need the FIR filter to operate at high frequencies, whereas battery operated applications like mobile communication filter work with high throughput, low-power and less area. Parallel and pipelining processing are the two methods used in DSP applications to reduce the power consumption. Pipelining reduces the length of the longest path by interleaving pipelining stages along the datapath. Pipelining techniques increases the number of delay elements and the system latency, on the one hand parallel processing escalates the sampling rate by replicating resources on the other hand the multiple inputs can be processed in parallel, area and power increased due to resource replication. Due to this augment in power and area by the block size L, the parallel processing technique loses its advantage in implementation. The papers (Acha, 1989)(Chung,2002)(Chung,2004)(Chung,2005) (Chung,2007 )[Lin,1996] (Mou,1991)(Parhi,1999)(Parker,1997) proposed the ways to reduce the complexity of the parallel FIR filter. In (Chung,2002 ) (Mou,1991) (Parhi,1999)(Parker,1997) the authors used polyphase decomposition techniques, the small-sized parallel FIR filter structures were designed and then the larger block was constructed by cascading small-sized parallel FIR filtering blocks. In (Chung,2002 )(Mou,1991)(Parker,1997) Fast FIR algorithms (FFAs)are introduced to reduce the number of multipliers from (L X N) to (2N- N /L).The fast linear convolution is utilized in (Acha, 1989) (Chung,2004 )(Chung,2005)(Chung,2007 )[Lin,1996] . In (Yu-Chi,2012)symmetric convolutions and the symmetry of coefficients have been considered to reduce the number of multiplier into half, this reduces the area and power consumption of even and odd length parallel FIR filters. The transposed direct-form subfilter structures in (Yu-Chi,2012) are designed by using Canonical Signed Digit (CSD) based
Single Constant Multiplier (SCM) which consumes large area and power. In transposed direct-form FIR filter, the input sample is multiplied with all the filter coefficients. The Multiple Constant Multiplication (MCM) method is much suitable for the design of transposed direct-form FIR filter since input sample is multiplied with all the filter coefficients. The MCM technique reduces the number of adder and subtractor required for the design of multipliers by Common Subexpression Elimination (CSE) method. In (Andrew, 1995),(Dempster,1994),(Yevgen,2007) the authors highlighted the various methods used to reduce the adder logical length in MCM design.

In this paper, the Even Symmetric Parallel Fast Finite Impulse Response (ESPFFIR) filter (Yu-Chi,2012) is designed using Multiple Constant Multiplier (MCM) and Modified CSLA(Ramkumar,2012)instead of Carry Save Adder (CSA) to occupy less area and to consume less power. Various CSE based algorithms like Heuristic of cumulative benefit (Hcub), n-dimensional Reduced Adder Graph (RAG-n) and Bull Horrocks Modified algorithm (BHM) have been used for designing Multiple Constant Multipliers, which are the major part in the design structure of ESPFFIR filter. This paper is structured as follows. In Section 2, the ESPFFIR filter structures are presented. In Section 3 MCM based ESPFFIR subfilter design are discussed. Section 4 presents Modified CSLA based ESPFFIR filter are conferred. In Section 5, the Multiple Constant Multipliers (MCM) are analysed and compared. Section 6 gives the conclusion.

2. ESPFFIR FILTER STRUCTURE

In digital signal processing, parallel or block processing is a technique where the functional units are replicated so as to operate on different signals simultaneously. Parallel FIR filters also employ the same technique where multiple inputs are processed simultaneously to generate multiple outputs. The filters can be generally named L-parallel filters, where L is the number of inputs processed in parallel called the block length. Obviously, parallel processing increases the throughput of FIR filters. A L-parallel FIR filter working at the same clock frequency as the original filter produces L outputs per clock cycle compared to the single output produced per clock cycle by the original filter, thus, in effect working at L times the rate of original filter. Parallel processing can also be used in FIR filters for reduction of power consumption, which can be effected by reducing the supply voltage to the filter. The parallel filter structures discussed here are based on FFA. The traditional parallel filters increase the hardware as many times as the block length. Thus, in practical applications, FIR filter structures which consume lesser hardware are used. As an example of this, the architecture of parallel filters can be modified for reduced area and low power consumption when the coefficients are symmetric (Yu-Chi,2012). The regular 2-parallel structure is shown in Figure1.

![Figure1. 2-parallel FIR filter using FFA](image)

In a 2-parallel, 24-tap filter, let the coefficients be

\[ \{h(0), h(1), h(2), h(3)h(4), \ldots \, \ldots \ldots \, \ldots \, \ldots \, h(20), h(21), h(22), h(23)\} \]

If they are symmetric, that is \( h(0) = h(23), h(1) = h(22), \ldots \, h(11) = h(12) \)

Then the subfilter taps are generated as follows,
\( H_0 = \{ h(0), h(2), h(4), h(6), \ldots, \ldots, h(10) \} \)

\& \( H_1 = \{ h(1), h(3), h(5), h(7), \ldots, \ldots, h(11) \} \)

so,

\( H_0 + H_1 = \{ [h(0) + h(1)], [h(2) + h(3)], \ldots, \ldots, [h(10) + h(11)] \} \) \hspace{1cm} (1)

For a 2-parallel filter, using FFA, there are three subfilter blocks and there are 12 multipliers each for the subfilters \( H_0 \) and \( H_1 \). As the \( H_0 + H_1 \) subfilter block is symmetric, it requires only 6 multipliers which is a total of 30 for the whole filter. The 3-parallel filter structure is shown in Figure 2. Here the subfilters \( H_1 \) and \( H_0 + H_1 + H_2 \) are symmetric and they require 4 multipliers each and the remaining four subfilters have 8 multipliers each which is a total of 40 multipliers for the filter in total.

Using the symmetry of the coefficients, the structures are modified as in Figure 3 and Figure 4 where the number of symmetric subfitters in 2-parallel is increased to 2 from 1 and in 3-parallel from 2 to 4. Thus, the total multipliers in 2-parallel and 3-parallel are reduced to 24 and 32 respectively. Still the parallel filters have a large number of multipliers and adders which increase with the length of the filter. Using a single MCM instead of designing SCMs or CSD multiplier for each of the multiplications utilizes the redundancy of the whole structure thereby reducing the adder operations. The power and area can be brought down further if the adders used in the filter are area and power efficient especially when the filter length is large.
3. MCM BASED ESPFFIR SUBFILTER DESIGN

In MCM also, multiplication by each constant is realized using additions, shifts and subtractions. The aim is to find the lowest-cost combination of these for the implementation of multiplication. The operation of MCM is as shown in Figure 5, where the input X is being multiplied by the constants, $C_1$ to $C_n$.

The main advantage provided by MCM is the intermediate term sharing. This can be shown with an example. Consider an arbitrary number X which has to be multiplied by both 13 and 25. Performing these multiplications separately, as SCM problems, requires at least 2 adders each which is a total of 4 adders. But to use the intermediate terms efficiently we may decompose these as $13X = (9X) + (X << 2)$ and $25X = (9X) + (X << 4)$. As we can see, there is a common term, $9X$, which can be shared. Thus both 13X and 25X can be implemented with just 3 adders, of which one is for $9X = (X << 3) + X$. It helps to reduce more number of adder operations than SCMs.

3.1. Subfilters design using MCM

The non-symmetric and symmetric subfilter blocks using MCM can be implemented as shown in Figure 6 and Figure 7. As shown in figure, the non-symmetric subfilter block is implemented using a MCM with the same number of constants as in the subfilter block, whereas the MCM in symmetric is implemented for half the constants in subfilter block. The transposed direct form of FIR filter is used here.
3.1.1. Algorithms used in MCM - Bull Horrocks Modified algorithm (BHM)

The BHM algorithm is a modified form of BH algorithm (Dempster, 1994). BH algorithm is a graph based MCM heuristic which can be implemented using only additions, addition and subtraction, addition and shifts and a combination of the three. As only addition is performed in BH algorithm, the targets are created in the ascending order. That is, larger terms can be built from the smaller and not the other way around. The algorithm tries to minimize the distance between a target $t$ and its closest element in $R$.

Error is the smallest term, which when added, gives the target $t$. The error decreases with each iteration until it becomes 0. The error is given as

$$ e = \min_{r \in R} t - (r \ll n), \text{ for } n \geq 0, n \in \mathbb{Z}, t > (r \ll n) $$

(2)

where $R$ is the set of existing terms constructed so far that it will be containing the solution at the end of algorithm, $t$ the target set and $n$ represent shift.

If $n=0$, then no shift is allowed. As $t > (r \ll n)$, the error is always positive. If

$$ r' = (r \ll n), \text{ then } $$

$$ r' = t - e $$

(3)

This means that if $e \in R$, then $t$ can be constructed with one more adder operation. Though the fundamental error minimization approach is used in BHM, here the error is allowed to be negative for better utilization of subtraction and thus mitigating the disadvantage of BH algorithm. Thus, the error for BHM is

$$ e = \min_{r \in R} \vee t - (r \ll n) \vee, \text{ for } n \geq 0, n \in \mathbb{Z} $$

(4)

3.1.2. $n$-dimensional Reduced Adder Graph (RAG-$n$)
RAG-n algorithm was the first to introduce the idea of adder distance and successor set (Andrew, 1995) where adder distance is the number of adder-operations required to construct a target from the existing terms. The RAG-n algorithm has an optimal part and a heuristic part. Here if any of the unconstructed target values satisfies the condition $t \in S$, then that $t$ is constructed immediately, where $S$ is the set of targets that can be constructed with one adder operation. This is the optimal part of RAG-n. In RAG-n the targets are not constructed in predefined order as in BHM. If there is no target at distance 1, that is no $t \in S$, then the heuristic part is used. Here it is checked whether any target can be constructed with 2 adders. For this two tests are performed. The first is to check if $|t-r| \in C_1$ for each $t$ and $r \in R$. The second is to check if $|t-s| \in C_0$ for each $t$ and $s \in S$, where $C_0$ is the set of constants and the optimal SCM cost is $n$. If any of the two tests succeed, it shows that the target is at distance 2 and which intermediate term can be used to construct it. Otherwise, the target with the smallest single constant multiplication is constructed.

### 3.1.3 Heuristic of cumulative benefit (Hcub)

A common weakness of BHM and RAG-n is the inability to choose intermediate terms which are jointly beneficial to all the remaining targets (Yevgen, 2007). The Hcub algorithm tries to maximize the benefit i.e., the intermediate terms provide to all the remaining targets. This can be explained using an example where $t = \{23, 81\}$. 23 can be constructed as $23X = (3X\ll 3) - X$ where $3X = (X\ll 1) + X$. But then, the target 81 will be at a distance 2. But if 23 is constructed as $23X = (X\ll 5) - (9X$) where $9X = (X\ll 3) + X$ using which 81X can be constructed as $81X = ((9X\ll 3) + (9X))$ with only one more adder operation. Hcub efficiently reuses the optimal part of RAG-n in a computationally efficient manner. If ‘$r$’ is the newly constructed term on an iteration, then the new ready set is

$$R_{new} = R_{old} \cup \{r\}$$

(5)

The successor set is constructed with one adder operation on any two elements in $R$. Thus it has to consider all the possible pairs in $R$. When $r'$ is added to $R$, the new pairings, possible are $r'$ with $r$ and $r'$ with $R_{old}$ where $R_{old}$ is the ready set before addition of $r'$. Thus, the new element to be added to $S$ is

$$S_{update} = A(r', r') \cup A(r', R_{old})$$

(6)

For $t$ not $\in S$ for all $t$, Hcub finds a successor $S$ using

$$H_{cub}(R, S, T) = arg_{s \in S} \max \left\{ \sum_{t \in T} B'(R, S, T) \right\}$$

(7)

where,

$$B'(R, S, T) = 10^{-dist(RU[s], t)} (dist(R, t) - dist(RU[s], t))$$

$B'$ is the weighted benefit function and can be considered useful if the remaining adder distance decreases with its construction. That is $dist(RU[s], t) < dist(R, t)$ closer targets are given more benefit as they can be used for the construction of other terms. Figure 8 gives the MCM in H0 + H1 subfilter using Hcub.

### 4. MODIFIED CSLA BASED ESPFFIR FILTER

The ESPFFIR filters have preprocessing and postprocessing adders, the adder used in (Yu-Chi, 2012) are Carry Save Adders (CSA). In this paper the CSA adders are replaced by Carry Select Adder (CSLA), which consumes less area and power compared to CSA. The 2-parallel ESPFFIR filter requires 2 preprocessing and 4 postprocessing adder, when the number of parallel level increases the preprocessing and postprocessing adders also increase e.g., the 3-parallel ESPFFIR filter require 5 preprocessing and 12 postprocessing adders. To reduce the overall area and power consumption of ESPFFIR filter the CSA adders are replaced by modified CSLA adders. The CSLA adders are used to perform fast arithmetic functions in most of the data processing elements. The modified CSLA (Ramkumar, 2012) is designed based on binary to excess one converter.

Figure 9 shows the 4-bit Binary to Excess-one Converter (BEC) in (Ramkumar, 2012) for any binary input the BEC generates the excess one code. In this CSLA one row of full adders in the regular CSLA structure is replaced with BEC modules. The gate count in a BEC module is lesser than that of a full adder, hence the power
consumption and area occupied are reduced significantly. If \( \text{carry-in} = 1 \) then the BEC module is used to calculate the sum else RCA module is used to calculate the sum. The CSAs are usually used when more than two operands are to be added. In CSA, the generated carry-out is saved rather than propagated. Thus, it generates a sequence of partial sum bits and a sequence of carry bits. Hence, CSAs can be used to construct an adder tree for the summation of three or more operands, where the sequence of carry bits is added to the partial sum in the final stage of the tree. The final addition is usually done using ripple carry adder or look ahead carry adder. The look ahead carry adder increases additional hardware overhead for carry generation and propagation while the ripple carry adder increases the delay largely. On the other hand the CSLAs are very fast and also the modified BEC consumes lesser power and area than the regular CSLA.

\[
\begin{align*}
X << 3 &\quad 8x \\
-7x &\quad 64x \\
<<5 &\quad 224s \\
<<4 &\quad 112s \\
<<4 &\quad 112s \\
<<1 &\quad 14s \\
231x &\quad 105s \\
-105s &\quad 113s \\
<<2 &\quad 924s \\
<<2 &\quad 683s \\
<<2 &\quad 452s \\
<<1 &\quad 354s \\
-1 &\quad 177s \\
-14s &\quad Y1 \\
-14s &\quad Y6 \\
14s &\quad Y5 \\
14s &\quad Y3 \\
14s &\quad Y4 \\
14s &\quad Y2
\end{align*}
\]

Figure 8. MCM multiplier for \( H_0 + H_1 \) subfilter using Hcub algorithm
Figure 9. 4-bit Binary to Excess-one Converter (BEC)

Figure 10. 16-bit CSLA using BEC

In this paper a 16-bit BEC based CSLA adder/subtractor is designed as shown in Figure 10. The preprocessing and postprocessing block of the ESPFFIR filter is designed by using adders/subtractor datapath elements. The area, power and delay comparison of CSA and CSLA is provided in Table I. From the Table I it is clear that CSLAs are advantageous over CSAs in area, power and delay.

<table>
<thead>
<tr>
<th>Adder</th>
<th>Area($\mu m^2$)</th>
<th>Power($\mu W$)</th>
<th>Delay(ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSA</td>
<td>567</td>
<td>20.6</td>
<td>2.1</td>
</tr>
<tr>
<td>Modified CSLA</td>
<td>476</td>
<td>16.8</td>
<td>1.3</td>
</tr>
</tbody>
</table>

TABLE I
Comparison of CSA and CSLA
5. ANALYSIS AND COMPARISON OF MULTIPLE CONSTANT MULTIPLIERS

The number of operations performed to implement multiplication is provided in Table II for the algorithms BHM, Hcub, RAG-n and SCMs for L-parallel filters. It is evident that the intermediate term sharing reduces the operations required significantly. Around 40% reduction is provided by MCMs compared to SCM.

Table II

Comparison of required operations, required number of addition or subtraction (Add/Sub), required number of Shifts (Shift), Required number of negations (negation), maximum reduction of Add/Sub in MCM from SCM (M.R.A) are presented in table.

<table>
<thead>
<tr>
<th>Length</th>
<th>L</th>
<th>Operation</th>
<th>MCM</th>
<th>SCm</th>
<th>M.R.A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Add/Sub</td>
<td>Hcub</td>
<td>BHM</td>
<td>RAG-n</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td></td>
<td>22</td>
<td>26</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shift</td>
<td>27</td>
<td>31</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Negation</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>24</td>
<td>2</td>
<td>Add/Sub</td>
<td>20</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shift</td>
<td>38</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Negation</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>24</td>
<td>3</td>
<td>Add/Sub</td>
<td>36</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shift</td>
<td>51</td>
<td>53</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Negation</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>36</td>
<td>3</td>
<td>Add/Sub</td>
<td>52</td>
<td>60</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shift</td>
<td>76</td>
<td>84</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Negation</td>
<td>26</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>48</td>
<td>3</td>
<td>Add/Sub</td>
<td>66</td>
<td>70</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shift</td>
<td>97</td>
<td>99</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Negation</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>60</td>
<td>3</td>
<td>Add/Sub</td>
<td>85</td>
<td>116</td>
<td>116</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shift</td>
<td>153</td>
<td>184</td>
<td>174</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Negation</td>
<td>34</td>
<td>34</td>
<td>34</td>
</tr>
</tbody>
</table>

Table III

Comparison of Add/Sub for a 12-tap 2-parallel filter

<table>
<thead>
<tr>
<th>Subfilter</th>
<th>Coefficient</th>
<th>Fractional Rounding</th>
<th>MCM</th>
<th>SCm</th>
<th>CSD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Hcub</td>
<td>BHM</td>
<td>RAG-n</td>
<td></td>
</tr>
<tr>
<td>H1</td>
<td>0.12647434971185</td>
<td>0.12646484375</td>
<td>9</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>0.14347897460436</td>
<td>0.143310546875</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.17722976933873</td>
<td>0.177001953125</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.16449384638344</td>
<td>0.164306640625</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1242112565414</td>
<td>0.1240234375</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Table IV
Power Consumption

<table>
<thead>
<tr>
<th>L</th>
<th>Length</th>
<th>MCM-CSLA</th>
<th>SCM-CSA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Hcub</td>
<td>BHM</td>
</tr>
<tr>
<td>2</td>
<td>12-tap</td>
<td>13.5</td>
<td>14.9</td>
</tr>
<tr>
<td>2</td>
<td>24-tap</td>
<td>26.4</td>
<td>26.8</td>
</tr>
</tbody>
</table>

Table V
Area Consumption

<table>
<thead>
<tr>
<th>L</th>
<th>Length</th>
<th>MCM-CSLA</th>
<th>SCM-CSA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Hcub</td>
<td>BHM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SCM-CSA</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>12-tap</td>
<td>62254</td>
<td>67218</td>
</tr>
<tr>
<td>2</td>
<td>24-tap</td>
<td>115935</td>
<td>117472</td>
</tr>
</tbody>
</table>
### Table VI
Comparison of Area & Power

<table>
<thead>
<tr>
<th>Technology(µm)</th>
<th>Implementation</th>
<th>Area (µm²)</th>
<th>Power(mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.18</td>
<td>Hcub-CSLA</td>
<td>115935</td>
<td>26.4</td>
</tr>
<tr>
<td></td>
<td>BHM-CSLA</td>
<td>117472</td>
<td>26.8</td>
</tr>
<tr>
<td></td>
<td>RAG-n-CSLA</td>
<td>117472</td>
<td>26.7</td>
</tr>
<tr>
<td></td>
<td>SCM-CSLA</td>
<td>143594</td>
<td>32.3</td>
</tr>
</tbody>
</table>

![Figure 12](image-url) Representation of Table IV, V and VI (a) 12-tap filter (b) 24-tap filter
6. CONCLUSION

This paper presented the design of ESPFFIR filter using Multiple Constant Multiplier and modified CSLA. Multipliers are the foremost segment in power and area consumption for the Even Symmetric Parallel Fast Finite Impulse Response (ESPFFIR) filter implementation. Hence the multiplier in this ESPFFIR is replaced by Hcub-n based Multiple Constant Multiplier which consumes fewer adders compared with SCM,BHM and n-RAG multipliers. The CSA adder is replaced by BEC based CSLA adder/subtractor in the preprocessing and postprocessing datapath elements. The power and area of the modified CSLA adder/subtractors are 20.6(µW) and 567(µm²), respectively. The overall power and area consumption of 12-tap and 24-tap two parallel ESPFFIR filters are 26.4(mW) and 115935(µm²) respectively. The ESPFFIR filter can be designed using Mixed Integer Programming (MIP) technique based Multiple Constant Multiplier (MCM) in future.

REFERENCES


Yevgen Voronenko and Markus PuSchel (2007) “Multiplierless Multiple Constant Multiplication”, ACM Transactions on Algorithms, 3(2), Article No. 11.
