CHAPTER 6

PSM BASED MULTIPLIER ON FIR FILTER FOR SDR RECEIVER

6.1 INTRODUCTION

With the advent of SDR technology, FIR filter research has been focused on reconfigurable realizations. The fundamental idea of SDR is to replace most of the analog signal processing in the transceivers with digital signal processing in order to provide the advantage of flexibility through reconfiguration. The two important requirements of SDR receiver are reconfigurability and low complexity. To achieve these requirements, Constant Shift Method (CSM) and Programmable Shift Method (PSM) are proposed instead of normal multiplier in FIR block (Mitola 2000). The architectural design and FPGA based implementation are presented in this chapter.

The prime functions of the digital filters are nothing but intensive addition and multiplication. Over the years, the methodology, techniques and procedures deployed in realizing the associated hardware that undertake these tasks have matured. They are predominantly bound on utilizing the standard multiplier and adder structures in either fully parallel or time-multiplexed resource-sharing architectures.

However, there still remains a lot of redundancy in the arithmetic circuits and their associated computations as there is little sharing of the
low-level intermediate calculations. The multiplier block approach has addressed this gap and has resulted in significant reduction in power, area and delay of the multiple constant multiplications in the fully-parallel structures.

While describing about the multiplier, the shifting operations are considered to be very important. FIR filters, IIR filters, filter-banks, poly-phase filters and adaptive filters can all be implemented as time multiplexed structures. The application of the multiplier blocks to the constant shift method and programmable shift method are considered to be the best methods.

The proposed architectures consider coefficients as constants (as they are stored in LUTs) and input signal as variable. The coefficient multiplication in such a case is known as Multiple Constant Multiplications (MCM), i.e., multiplication of one variable (input signal) with multiple constants (filter coefficients). The MCM is then optimized for eliminating redundancy using a new common subexpression elimination algorithm to minimize the filter complexity.

In Section 6.2, proposed architectures for FIR filter are described. A new common Subexpression Elimination algorithm is presented in the Section 6.3. The architecture of CSM is explained in the Section 6.4 followed by the description of PSM architecture in the Section 6.5. The synthesis results are presented in the Section 6.6. Conclusion is given in Section 6.7.

6.2 PROPOSED ARCHITECTURES FOR FIR FILTER

In this section, the architecture of the proposed FIR filter is presented. The architecture is based on the transposed direct form FIR filter structure as shown in Figure 6.1.
Figure 6.1 Transposed direct form of an FIR filter

The Processing Element (PE-n) corresponding to the n<sub>th</sub> coefficient (Wang et al 2004). PE performs the coefficient multiplication operation with the help of a shift and add unit. The architecture of PE is different for proposed CSM and PSM. In the CSM, the filter coefficients are partitioned into fixed groups and hence the PE architecture involves constant shifters (Hartley 1996). But in the PSM, the PE consists of Programmable Shifters (PS).

The FIR filter architecture can be realized in a serial way in which the same PE is used for generation of all partial products by convolving the coefficients (Park et al 2000) with the input signal or in a parallel way, where parallel PE architectures are employed. The first option is used when power consumption and area are of prime concern.

The architecture of the reconfigurable MB is shown in the Figure 6.2.
The functions of different blocks of the PE are explained below:

i. **Shift and Add Unit**

It is well known that one of the efficient ways to reduce the complexity of multiplication operation is to realize it using shift and add operations (Park and Kang 2001). In contrast to conventional shift and add units used in previously proposed reconfigurable filter architectures, the BCSs-based shift and add units are used in the proposed CSM and PSM architectures.

In both the architectures (CSM and PSM) proposed in this chapter, the same shift and add unit is used. Thus, the use of 3-bit BCSs reduces the number of adders needed to implement the shift and add unit compared to conventional shift and add units.
ii. Multiplexer Unit

The multiplexer units are used to select the appropriate output from the shift and add unit. All the multiplexers share the outputs of the shift and add unit. The inputs to the multiplexers are the 8/4 inputs from the shift and add unit and hence 8:1/4:1 multiplexer units are employed in the architecture. The select signals of the multiplexers are the filter coefficients which are previously stored in a LUT. The CSM and PSM architectures basically differ in the way filter coefficients are stored in the LUT (Hartley 1996).

In the CSM, the coefficients are directly stored in LUTs without any modification whereas in PSM, the coefficients are stored in a coded format. The number of multiplexers is also different for PSM and CSM. In CSM, the number of multiplexers is dependent on the number of groups after the partitioning of the filter coefficients into fixed groups. The number of multiplexers in the PSM is dependent on the number of non-zero operands in the coefficients after the application of new common subexpression elimination algorithm (Potkonjak et al 1996, Mahesh et al 2008).

iii. Final Shifter Unit

The final shifter unit performs the shifting operation after all the intermediate additions (i.e. intra-coefficient additions) are done. This can be illustrated using the output expression:

\[
y = 2^{-4}x + 2^{-6}x + 2^{-15}x + 2^{-16}x
\]

By coefficient-partitioning,

\[
y = 2^{-4}(x + 2^{-2}x) + 2^{-15}(x + 2^{-1}x)
\]
After obtaining the intermediate sums \((x + 2^{-2}x)\) and \((x + 2^{-1}x)\) from the shift and add units with the help of multiplexer unit, the final shifter unit performs the shift operations \(2^{-4}\) and \(2^{-15}\) in Equation (6.2). The PSM and CSM architectures also differ in the nature of final shifters. In the CSM, the final shifts are constants and hence no Programmable Shifters are required. In the PSM, Programmable Shifters are used.

**iv. Final Adder Unit**

This unit will compute the sum of all the intermediate additions \(2^{-4}(x + 2^{-2}x)\) and \(2^{-15}(x + 2^{-1}x)\) as in Equation (6.2). As the filter specifications of different communication standards are different, the coefficients change with the standards. In conventional reconfigurable filters, the new coefficient set corresponding to the filter specification of the new communication standard is loaded in the LUT.

Subsequently, the shift and add unit performs a bitwise addition after appropriate shifts. On the contrary, the proposed CSM and PSM architectures perform a Binary Common Sub-expression (BCS)-wise addition (instead bitwise addition). Thus, the same hardware architecture can be used for different filter specifications to achieve the necessary reconfigurability (Demirsoy et al 2004). Moreover, the proposed BCS-based shift and add unit reduces addition operations and hence offers hardware complexity reduction.

### 6.3 A NEW COMMON SUBEXPRESSION ELIMINATION ALGORITHM

An improved CSE method based on the CSD representation of the filter coefficients is proposed that maximizes the elimination of common
subexpressions. The basic idea of this method is searching and selecting patterns with a look ahead to eliminate redundant Horizontal and Vertical Common Subexpressions (HCSs and VCSs) so that the subexpressions are maximized and least number of nonzero bits are left ungrouped. The grouping is done in such a way that the logic depth of the multiplier is kept minimal. As the proposed CSE uses CSD, the initial optimization space has fewer numbers of nonzero bits unlike BSE (Mahesh et al 2006).

The new CSE algorithm combines three techniques, binary Horizontal Subexpression Elimination (HCSE), binary Vertical Subexpression Elimination (VCSE) and hardwiring of the final stages which reduce the number of adders. This technique focuses on eliminating redundancy by searching and selecting patterns with a look ahead technique in coefficient multiplier.

By applying the method based on the BCS,

$x_3$ to $x_6$ are formed from the binary representation of coefficients as follows

\[
[0 \ 1 \ 1] = x_3 = 2^{-1} x_1 + 2^{-2} x_1 \quad (6.3)
\]
\[
[1 \ 0 \ 1] = x_4 = x_1 + 2^{-2} x_1 \quad (6.4)
\]
\[
[1 \ 1 \ 0] = x_5 = x_1 + 2^{-1} x_1 \quad (6.5)
\]
\[
[1 \ 1 \ 1] = x_6 = x_1 + 2^{-1} x_1 + 2^{-2} x_1 \quad (6.6)
\]
\[
x_3 = 2^{-1} x_1 + 2^{-2} x_1 = 2^{-1}(x_1+2^{-1}x_1) = 2^{-1}x_8 \quad (6.7)
\]
\[
x_6 = x_1 + 2^{-1} x_1 + 2^{-2} x_1 = x_8 + 2^{-2} x_1 \quad (6.8)
\]
The main disadvantage of the BSE is that BHCSs are formed without a look-ahead and therefore many bits are left ungrouped after obtaining the BHCSs. The proposed CSE method can be explained using the example of a 12-tap FIR filter coefficients shown in Table 6.1.

Table 6.1 CSD representation of filter coefficients

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>h₀</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>h₁</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>h₂</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>h₃</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>h₄</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>h₅</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.2 shows the conventional horizontal subexpression formation for an example filter \( h₀ \) and \( h₁ \), whereas Table 6.3 shows the grouping of bits using look-ahead method. In Table 6.2, the two bits are ungrouped whereas in Table 6.3, all the bits are grouped. This minimizes the number of adders. The HCSs \( x₃ = [1 \ 0 \ 1] \), \( x₄ = [1 \ 0 \ -1] \), \( x₅ = [1 \ 0 \ 0 \ 1] \), \( x₆ = [1 \ 0 \ 0 \ -1] \) and VCS \( x₂ = [1 \ 1] \).

Table 6.2 Grouping by the sequential method

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>h₀</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>h₁</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 6.3  Grouping by the look-ahead method

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>h₀</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>h₁</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.4 is formed from Table 6.1 by substituting HCSs, [1001]=5, [101]=3 and VCS [11]=2. Super-subexpression (SSs) is formed by identical shifts between them or HCS and nonzero bits. In Table 6.2, the SS 8 is formed from the HCS [101] and the bit ‘1’ with a shift difference of one between them and SS 9 is formed from the HCS [101] and the bit ‘-1’ with a shift difference of one between them.

Table 6.4  Final representation of FIR filter coefficients

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>h₀</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>h₁</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>h₂</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>h₃</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>h₄</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>h₅</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

From Table 6.4, the output of the example can be expressed as

\[ y_k = 2^{-3} x_2 + 2^{-5} x_6 + 2^{-10} x_3 + 2^{-5} x_2[-1] + 2^{-7} x_6[-1] + 2^{-10} x_3[-2] + 2^{-4} x_9 \\
[ -3] + 2^{-11} x_2[-3] + 2^{-4} x_9[-4] + 2^{-2} x_5[-5] + 2^{-7} x_5[-5] \]  \hspace{1cm} (6.3)
The number of Multiplier Block Adders (MBAs) required to implement the filter using the direct method (method using shifts and adds) in Table 6.1 is 18. The proposed CSE method needs only 11 MBAs (6 for subexpressions and 5 for actual realization), which is a reduction of 39% over the direct method. The reduced percentage is larger when higher order filters are considered. In SDR receivers, the channel filter coefficients need to be changed as the filter specification. So, reconfigurability is needed for SDR channel filters.

6.4 ARCHITECTURE OF CSM

In the CSM architecture, the coefficients are stored directly in the LUT. These coefficients are partitioned into groups of 3-bits and are used as the select signal for the multiplexers. The number of multiplexer units required is \([n/3]\), where \(n\) is the word length of the filter coefficients. The CSM can be explained with the help of an 8-bit coefficient \(h = "0.11111111."\) This coefficient \(h\) is the worst-case 8-bit coefficient since all the bits are nonzero and hence needs a maximum number of additions and shifts. In this case, \(n = 8\), and therefore the number of multiplexers required is 3. The output \(y\) is expressed as

\[
y = 2^{-1}x + 2^{-2}x + 2^{-3}x + 2^{-4}x + 2^{-5}x + 2^{-6}x + 2^{-7}x + 2^{-8}x
\]  \hspace{1cm} (6.4)

By partitioning into groups of three bits from Most Significant Bit (MSB), the following expression is obtained.

\[
y = 2^{-1}(x+2^{-1}x+2^{-2}x+2^{-3}x+2^{-4}x+2^{-5}x+2^{-6}x+2^{-7}x)
\]  \hspace{1cm} (6.5)

\[
y = 2^{-1}(x+2^{-1}x+2^{-2}x+2^{-3}(x+2^{-1}x+2^{-2}x)+2^{-6}(x+2^{-1}x))
\]  \hspace{1cm} (6.6)
The terms \( x + 2^{-1}x + 2^{-2}x \) and \( x + 2^{-1}x \) can be obtained from the shift and add unit. Then by using the three multiplexers (mux), two 8:1 mux for the first two 3-bit groups and one 4:1 mux for the last two bits of the filter coefficients, the intermediate sums shown inside the brackets of equation (6.6) can be obtained. The final shifter unit performs the shift operations \( 2^{-1}, 2^{-3} \) and \( 2^{-6} \). Since these shifts are always constant irrespective of the coefficients, programmable shifters are not required and these shifts can be hardwired.

![Architecture of CSM](image)

**Figure 6.3 Architecture of CSM**
The CSM architecture is shown in Figure 6.3 and the steps involved in CSM are as follows:

**Step 1:** Get the input $x$.

**Step 2:** Get the coefficients from the LUT and use as the select signal for the multiplexers.

**Step 3:** Perform the final shifting function on the output of the multiplexer.

**Step 4:** Perform the addition of intermediate sums using the final adder unit.

**Step 5:** Store the final result, $h^* x$, in the delay unit “D”.

**Step 6:** Go to step 2 if the coefficients in the LUT are not finished, else go to 1.

The three most significant bits of the coefficient are given as the select signal to the Mux1, the next 3-bits to Mux2 and so on till the least significant bits to the last multiplexer.

The coefficient wordlength is considered as 16 bits. The filter coefficients are stored in the LUT in sign-magnitude form with the MSB reserved for the sign bit. The first bit after the sign bit is used to represent the integer part of the coefficient and the remaining 16 bits are used to represent the fractional part of the coefficient. Thus, each 16-bit coefficient is stored as an 18-bit value in LUTs. Each row in LUT corresponds to one coefficient. Only half the number of coefficients need to be stored as FIR filter coefficients are symmetric.

The coefficient values corresponding to $2^0$ to $2^{-14}$ are partitioned into groups of three bits and are used as select signals to multiplexers Mux1 to Mux5. i.e., the set $(2^0, 2^{-1}, 2^{-2})$ forms the select signal to Mux1 and so on.
Since there are 3-bits, eight combinations are possible and hence Mux1 to Mux5 are 8:1 multiplexers. The value corresponding to $2^{-15}$ forms the select to a 2:1 multiplexer, Mux6. The output from the $i^{th}$ multiplexer is denoted as $r_i$. Even though the coefficient with values up to a precision of 16 bits are taken, the shifting of $2^{-1}$ is done finally as shown in Equations (6.5) and (6.6) and hence the maximum shift will be $2^{-15}$. Mux7 determines whether the output needs to be complemented based on the sign bit of the filter coefficient and hence it is a 2:1 multiplexer. In FIR filters, coefficient values are always less than one.

Hence, the integer bit has not been employed. However if an integer digit is required, the proposed architectures do not impose any restrictions to accommodate it. In the Figure 6.3, the shifts are obtained as follows. Let $r_1$ to $r_6$ denotes the outputs of Mux1 to Mux6, respectively. Then

$$y = 2^{-1}r_1 + 2^{-4}r_2 + 2^{-7}r_3 + 2^{-10}r_4 + 2^{-13}r_5 + 2^{-16}r_6.$$  \text{(6.7)}$$

The shifts are obtained by partitioning the 16-bit coefficient into groups of 3-bits.

By partitioning (6.7)

$$y = 2^{-1} [(r_1 + 2^{-3}r_2) + 2^{-6}((r_3 + 2^{-3}r_4) + 2^{-6}(r_5 + 2^{-3}r_6))].$$  \text{(6.8)}$$

Substituting $(r_1 + 2^{-3}r_2)$, $(r_3 + 2^{-3}r_4)$ and $(r_5 + 2^{-3}r_6)$ by $r_7$, $r_8$, and $r_9$, respectively in the Equation (6.8), then

$$y = 2^{-1}[r_7 + 2^{-6}(r_8 + 2^{-6}r_9)]$$ \text{(6.9)}$$

By substituting $(r_8 + 2^{-6}r_9)$ by $r_{10}$

$$y = 2^{-1}(r_7 + 2^{-6}r_{10})$$ \text{(6.10)}$$
By substituting \( (r_7 + 2^{-6}r_{10}) \) by \( r_{11} \)

\[
y = 2^{-1}(r_{11}) \tag{6.11}
\]

The expressions from the Equations (6-7) to (6.11) are represented in the Figure 6.3. The main advantage of the CSM architecture is that all the shifts are constants irrespective of the coefficients and hence can be hardwired resulting in high speed operation of the filter.

### 6.5 ARCHITECTURE OF PSM

The PSM approach is based on the new common subexpression elimination algorithm. Unlike the CSM method where constant shifts are used, the PSM employs programmable shifters. The advantage of PSM over CSM is that the former architecture always ensures the minimum number of additions and thus minimum power consumption. This is because PSM has a pre analysis part. The filter coefficients are analyzed using the new CSE algorithm. Thus the redundant computations (additions) are eliminated and the resulting coefficients in a coded format are stored in the LUT.

The coefficient word length is fixed as 16 bit. The number of multiplexers has been fixed as 5 (same as the number of non-zero operands). The LUT consists of two rows of 18 bits for each coefficient of the form SDDDDXXDDDDXXMMML and DDDXXDDDDXXDDDDXX, where “S” represents the sign bit, “DDDD” represents the shift values from \( 2^0 \) to \( 2^{-15} \) and “XX” represents the input “x” or the BCSs obtained from the shift and add unit. In the coded format, \( XX = “01” \) represents “x,” \( “10” \) represents \( x+2^{-1}x \), \( “11” \) represents \( x + 2^{-2}x \), and \( “00” \) represents \( x + 2^{-1}x + 2^{-2}x \), respectively.
Thus, the two rows can store up to five operands which is the worst case number of operands for a 16-bit coefficient. In most of the practical coefficients, the number of operands is less than the worst case number of operands, 5. In that case “MMMML” can be used to avoid unnecessary additions. The values “MMMM” will be given as select signal to the Mux6 and “L” to Mux8. “MMMML” indicates the presence of five operands. A “1” in each position indicates the presence of each operand.

Thus, for all operands to be present will be indicated by “MMMML” = “11111.” This means the Mux6 will select the output from the output of adder, A_4 and Mux8 will select the output of adder, A_2. If only first operand is present, “MMMML” = “10000.” This means the Mux8 will select the output of PS, shr_4 and Mux6 will select the output of PS, shr_1. As a result of this none of the adders shr_1 to shr_4 will be loaded saving significant amount of dynamic power.

The coding can be explained as given below. Consider the positive coefficient h,

\[
h = [1010011001010011]
\]

(6.12)

By using the CSE, substituting 2 = [1 1], 3 = [1 0 1], Equation (6.7) becomes

\[
h = [3000020003000020]
\]

(6.13)

It is stored in LUT and has only four operands, the fifth operands values ‘DDDDXX’ are substituted as ‘000000’ and ‘MMMML’ as ‘11110’. The XX values are given as select signals for Mux1 to Mux5. The values of DDDD are fed to the corresponding PS. Mux6 and Mux8 select the
appropriate output. The use of Mux6 and Mux8 reduces the number of adders utilized by selecting the output from the appropriate adder.

The shift and add unit is identical for both PSM and CSM. The number of multiplexer units required can be obtained from the filter coefficients after the application of new CSE. The number of multiplexers will be corresponding to the coefficient that has the maximum number of operands. The architecture for the PSM method with Programmable Shifts (PS) is shown in the Figure 6.4.

The steps involved in PSM are as follows:

**Step 1:** Obtain the BCSs from filter coefficients using the CSE algorithm.

**Step 2:** Store the resultant coefficients in the prescribed format as in Equation (6.13) in the LUT.

**Step 3:** Get the input x.

**Step 4:** Get the coefficients from the LUT and use as the select signal for the multiplexers and the programmable shifters.

**Step 5:** Perform the final shifting function on the output of the multiplexer using PS.

**Step 6:** Perform the addition of intermediate sums using the final adder unit.

**Step 7:** Store the final result, h^x, in the delay unit ‘D’.

**Step 8:** Go to step 4 if the coefficients in the LUT are not finished, else go to 3.
In this section, the synthesis results of the CSM and PSM architectures are presented and parameters like area, power and delay are compared. Table 6.5 shows the synthesis results of CSM and PSM 16-tap FIR filter that has a coefficient word length of 16 bits. The number of LUTs, slices used and the power consumptions are presented in Table 6.5.
Table 6.5 Gate count, delay and power comparison for CSM and PSM

<table>
<thead>
<tr>
<th>Parameters</th>
<th>PSM</th>
<th></th>
<th>CSM</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Existing</td>
<td>Proposed</td>
<td>Existing</td>
<td>Proposed</td>
</tr>
<tr>
<td>Gate Count</td>
<td>1457</td>
<td>1423</td>
<td>1489</td>
<td>1448</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>129</td>
<td>98</td>
<td>254</td>
<td>189</td>
</tr>
</tbody>
</table>

The simulation output of the proposed CSM based FIR filter is shown in Figure 6.5.

Figure 6.5 Simulation result of CSM Based FIR filter
The RTL schematic diagram of the proposed CSM based FIR filter is as shown in Figure 6.6.

![RTL Schematic for CSM Based FIR Filter](image)

**Figure 6.6** RTL Schematic for CSM Based FIR Filter

The simulation output of the proposed PSM based FIR filter is shown in Figure 6.7.

![Simulation result of PSM Based FIR filter](image)

**Figure 6.7** Simulation result of PSM Based FIR filter
The RTL schematic diagram of the proposed PSM based FIR filter is as shown in Figure 6.8.

**Figure 6.8 RTL schematic diagram of PSM based FIR filter**

Figures 6.9 to 6.11 shows the comparison of the gate count, delay and power for CSM and PSM.

**Figure 6.9 Comparison of gate count**
Figure 6.10 Comparison of delay (ns)

Figure 6.11 Comparison of power (mW)
6.7 CONCLUSION

The proposed CSM and PSM methods make use of architectures with fixed number of multiplexers and the reduction in complexity is achieved by applying the new CSE algorithm. The CSM method is to split the filter coefficients into groups of 3 bits and use these groups as selectors to multiplexer unit and obtain $h^*x$. But it doesn’t always guarantee the minimum number of additions to be performed. In PSM, since the new CSE algorithm is employed, the number of additions to be performed will always be minimum. This reduction is significant for higher order filters. Since the SDR needs large number of taps to meet the adjacent channel attenuation specifications, the PSM is best suited for the channel filters in SDRs.

In the case of PSM, the final shifting is done based on the values from LUT using programmable shifters where as in the case of CSM, the shifts are constants and the filter coefficients are always partitioned into groups of 3 bits. The synthesized results show that the PSM requires less number of additions compared to CSM. So, the power dissipation for PSM is minimum. Due to the presence of programmable shifts, the delay associated with PSM may be slightly higher than CSM.