CHAPTER 6

IMPLEMENTATION OF DIGITAL FIR FILTER

6.1 INTRODUCTION

The digital FIR filters are common components in many digital signal processing (DSP) systems. There are various applications like high speed/low power error control [81], high performance processor [82]. Digital signal processing algorithms rely heavily on the efficient computation of inner products. Very efficient methods have been developed for implementation of digital filters in FPGAs and custom ICs. There are several basic structures for FIR filters such as canonical, pipelined and inverted form. The key functional units in a digital filter are delay elements, adders, and multipliers; out of which multiplier dominates the hardware complexity. It is well known that by representing filter coefficients as sum-of-powers-of-two (SOPOT), each multiplication in filtering can be replaced with simple shift-and-add operations [71-73]. The complexity of the FIR multiplier is dominated by the number of adders or subtractors employed in the coefficient multipliers. In this research work, implementation of FIR filters is presented using a distributed arithmetic scheme with CSD coefficient representation. It is shown that the filter coefficients coded using CSD are advantageous in terms of area, speed and power as compared to SOPOT representation [74]. In this research work an efficient coefficient coding scheme using CSD representation for implementing FIR filters is proposed. A new reversible logic based hardware reduction technique based on DA method is also proposed in this research work which is efficient in terms of device utilization and speed.
6.2 DIGITAL FIR FILTER

The processing of digital signals includes design and implementation of systems called filters. Filters are linear time invariant (LTI) systems; they need the three elements to describe digital filter structures i.e. adders, multipliers and delay elements. Figure 6.1 shows addition, multiplication and delay computation for LTI system. Figure 6.2 shows basic setup of FIR filter.

Figure 6.1 Addition, multiplication and delay computation for LTI system

Figure 6.2 Basic setup of FIR filter
A finite-duration impulse response filter has a system function of the form,

\[ H(z) = b_0 + b_1 z^{-1} + ... + b_{M-1} z^{1-M} = \sum_{n=0}^{M-1} b_n z^{-n} \]  

(6.1)

Hence the impulse response \( h(n) \) is,

\[ h(n) = \begin{cases} b_n & \text{if } 0 \leq n \leq M-1 \\ 0 & \text{else} \end{cases} \]

The difference equation representation is given in equation (6.1), which is a linear convolution of finite support.

\[ y(n) = b_0 x(n) + b_1 x(n-1) + ... + b_{M-1} x(n-M+1) \]  

(6.2)

The order of the filter is \( M-1 \), while the length of the filter is \( M \). Figure 6.3 shows structure of 4-tap FIR filter whose equation can be written as,

\[ H(z) = \sum_{n=0}^{3} h_n z^{-n} \]  

(6.3)

Figure 6.3 4-tap FIR filter structure
6.2.1 Direct form of FIR filter

The difference equation of FIR filter is implemented as a tapped delay line since there are no feedback paths. Filter \( H(z) = \frac{B(z)}{A(z)} \) with input \( x[n] \) and output \( y[n] \).

\[
y[n] = \sum_{k=0}^{M} b[k] x[n-k] - \sum_{k=1}^{N} a[k] y[n-k]
\]  \hspace{1cm} (6.4)

Direct forms use coefficients \( a[k] \) and \( b[k] \) directly. Direct form 1 involves direct implementation of difference equation. It can be viewed as \( B(z) \) followed by \( \frac{1}{A(z)} \). Figure 6.4 shows direct form 1 FIR filter structure.

Direct form 2 implements \( \frac{1}{A(z)} \) followed by \( B(z) \). Figure 6.5 shows direct form 2 FIR filter structure.

---

**Figure 6.4 Direct form 1 FIR filter structure**

**Figure 6.5 Direct form 2 FIR filter structure**
6.2.2 Cascade form of FIR filter

The cascade form can be written as,

\[ H(z) = b_0 + b_1 z^{-1} + \ldots + b_{M-1} z^{1-M} \]  \quad (6.5)

\[ = b_0 \prod_{k=1}^{K} (1 + B_{k,1} z^{-1} + B_{k,2} z^{-2}); K = \left\lfloor \frac{M}{2} \right\rfloor \]  \quad (6.6)

6.2.3 Linear phase form of FIR filter

For frequency-selective filters (e.g., low-pass filters) it is generally desirable to have a phase response that is a linear function of frequency. That is,

\[ \angle H(e^{j\omega}) = \beta - \alpha \omega, -\pi < \omega < \pi, \beta = 0 \text{ or } \frac{\pi}{2} \] \quad (6.8)

For a causal FIR filter with impulse over [0, M-1] interval, the linear-phase conditions are,

\[ h(n) = h(M-1-n); \beta = 0 \text{, } 0 \leq n \leq M-1 \] \quad (6.9)

\[ h(n) = -h(M-1-n); \beta = \frac{\pi}{2}, 0 \leq n \leq M-1 \] \quad (6.10)

Consider the difference equation with a symmetric impulse response \( y(n) \) given as,

\[ y(n) = b_0 x(n) + b_1 x(n-1) + \ldots + b_1 x(n-M+2) + b_0 x(n-M+1) \] \quad (6.11)

\[ = b_0 [x(n) + x(n-M+1)] + b_1 [x(n-1) + x(n-M+2)] + \ldots \] \quad (6.12)
6.2.4 Frequency sampling form of FIR filter

In this form the system function $H(z)$ of an FIR filter can be reconstructed from its samples on the unit circle,

$$H(z) = \left(\frac{1 - z^{-M}}{M}\right) \sum_{k=0}^{M-1} H(k) W_{M}^{-k} z^{-1}$$ (6.13)

$W_{M}^{-k}$ are the roots $(k = 0, ..., M - 1)$

$H(k)$ are the residues $(k = 0, ..., M - 1)$

It is also interesting to note that the FIR filter described by the above equation has a recursive form similar to an IIR filter because it contains both poles and zeros.

Using the symmetry properties of DFT and $(W_{M}^{-k})$ factor, $H_k(z)$ can be written as,

$$H_k(z) = \frac{H_k}{1 - p_i z^{-1}} + \frac{H_k^*}{1 - p_i^* z^{-1}}$$ (6.14)

$$= 2|H_k|^* \frac{\cos(\angle H_k) - z^{-1} \cos(\angle H_k - 2\pi k / M)}{1 - 2z^{-1} \cos(2\pi k / M) + z^{-2}}$$ (6.15)

Let,

$$p1 = W_{M}^{-k} = \exp(-j2\pi k / M)$$ (6.16)

$$= \cos(2\pi k / M) + j \sin(2\pi k / M)$$ (6.17)

$$H_k = |H_k|^* \exp(j\angle H_k)$$ (6.18)

$$= |H_k|^* (\cos(\angle H_k) + j \sin(\angle H_k))$$ (6.19)
Then,

\[
H(z) = \frac{1-z^{-M}}{M} \left\{ H_k(z) + \frac{H(0)}{1-z^{-1}} + \frac{H(M/2)}{1-z^{-1}} \right\} \quad (6.20)
\]

\[
L = \frac{M-1}{2} \quad \text{for M odd; } L = \frac{M}{2} - 1 \quad \text{for M even.}
\]

### 6.3 FILTER IMPLEMENTATION

The transfer function of digital FIR filter is,

\[
y(n) = \sum_{k=0}^{M-1} h(k)x(n-k) \quad (6.21)
\]

The zero-phase frequency response of a Type 1 linear phase FIR filter can be expressed as,

\[
H(\omega) = h(0) + 2 \sum_{n=1}^{N} h(n) \cos(n\omega) \quad (6.22)
\]

Figure 6.6 shows the proposed digital FIR filter structure and Figure 6.7 shows the proposed digital FIR filter structure implemented using multiplierless technique.
6.3.1 Comparison results

The implementation of FIR filter has been done using reversible gates and the delay computation is done. As shown in the Figure 6.8 the delay for the NTG logic is less as compared to the Rgate logic. The NTG logic based FIR filter designed using baugh wooley multiplier provides 10% reduced delay as compared to Rgate logic based design.
Design of 4-tap FIR filter using reversible logic

Figure 6.8 Delay comparison for 4-tap FIR filters

Design of FIR filters using reversible logic

Figure 6.9 Power consumption comparison for 4-tap FIR filters
The designed reversible multiplier structures are used to implement the FIR filter. It is shown in Figure 6.9 that the power consumption in the FIR filters using Rgate logic is more as compared to proposed logic. Reversible concept adds advantage to the filter design in terms of lower power computation. The NTG logic based FIR filter designed using wallace tree multiplier have 17% improvement in power as compared to the FIR filter using other logic.

Figures 6.10-6.15 are showing comparison results illustrating number of LUTs used, number of gates used, number of occupied slices, delay, power consumption and power delay product (PDP) respectively for 4-tap FIR filter designed using various add and shift multiplier techniques implemented using reversible logic. The optimization results are also compared by implementing the multipliers used in digital FIR filters using VLSI strength reduction techniques.
**Figure 6.11** Comparison results for number of gates used in design of 4-tap FIR filters using add and shift multipliers

**Figure 6.12** Comparison results for number of slices used in design of 4-tap FIR filters using add and shift multipliers
4-tap FIR filter design using add and shift multipliers

Figure 6.13  Delay comparison results for design of 4-tap FIR filters using add and shift multipliers

Figure 6.14  Power consumption results for design of 4-tap FIR filters using add and shift multipliers
Figure 6.15  Comparison of power delay product for design of 4-tap FIR filters using add and shift multipliers

Figure 6.16  Comparison results for number of LUTs used in design of 4-tap FIR filters implemented using SE as strength reduction
Figure 6.17  Comparison results for number of gates used in design of 4-tap FIR filters implemented using SE as strength reduction

Figure 6.18  Comparison results for number of occupied slices used in design of 4-tap FIR filters implemented using SE as strength reduction
4-tap FIR filter design using add and shift multipliers by implementing SE as strength reduction

Figure 6.19  Delay comparison for 4-tap FIR filters implemented using SE as strength reduction

Figure 6.20  Power consumption comparison for 4-tap FIR filters implemented using SE as strength reduction
Figure 6.21 Comparison of power delay product for 4-tap FIR filters implemented using SE as strength reduction

Figure 6.22 Comparison results for number of LUTs used in design of 4-tap FIR filters implemented using LT as strength reduction
Figure 6.23  Comparison results for number of gates used in design of 4-tap FIR filters implemented using LT as strength reduction

Figure 6.24  Comparison results for number of occupied slices used in design of 4-tap FIR filters implemented using LT as strength reduction
Figure 6.25  Delay comparison for 4-tap FIR filters implemented using LT as strength reduction

Figure 6.26  Power consumption comparison for 4-tap FIR filters implemented using LT as strength reduction
Thus, it is shown that strength reduction promises further complexity reduction and gives better results in terms of delay, power consumption and overall performance metric i.e. power delay product. Figures 6.16-6.21 are showing the comparison results illustrating number of LUTs used number of gates used, number of occupied slices, delay, power consumption and PDP respectively among 4-tap FIR filter implemented using subexpression elimination. Figures 6.22-6.27 are showing the comparison results illustrating device utilization, delay, power consumption and PDP among 4-tap FIR filter implemented using linear transformation approach of strength reduction respectively.
The computational complexity of finite impulse response (FIR) filters used in the signal processing blocks is dominated by the number of adders or subtractors employed in the multipliers. The use of software defined radio (SDR) technology is predicted to replace many of the traditional methods for implementing transmitters and receivers while offering a wide range of advantages including adaptability, reconfigurability, and multifunctionality encompassing modes of operation, radio frequency bands, air interfaces, and waveforms. Research in this field is mainly directed towards improving the architecture and the computational efficiency of SDR systems. The most computationally intensive part of an SDR receiver is the channelizer since it operates at the highest sampling rate.

### 6.4.1 Distributed arithmetic

In order to reduce the hardware requirement and to implement large filters with high throughput distributed arithmetic [85] is often preferred. Distributed arithmetic (DA) is a multiplication free method applicable to fixed-point data, and is based on table lookups of pre-calculated partial products [86]. Also, DA filters achieve these advantages while retaining full precision, unlike filters using reduced sums and differences of powers of two. Figure 6.28 illustrates basic concept of DA. DA provides multiplier free multiplication by using bit serial computation by storing all possible combination sums of filter weights in LUT. Distributed arithmetic is a reliable option for low power applications because it allows replacement of costly multiplies operations with shifts and table lookups [86].

Today’s major concern is battery lifetime of portable electronics as more functionality is incorporated into these devices. Therefore, low-power circuits for signal processing applications are in demand. The signal processing functions employed in these devices include finite-impulse response (FIR) filters, discrete cosine transforms (DCTs), and discrete Fourier transforms (DFTs). The common feature of these functions is that they are all based on the inner product. The digital signal processing applications are presented in [87]. Digital signal processing (DSP)
implementations typically make use of multiply-accumulate (MAC) units for calculation of these operations, and the computation time increases linearly as the length of the input vector grows.

Figure 6.28 Basic concept of distributed arithmetic

6.4.2 SOPOT representation of filter coefficients

The general representation of sum-of-powers-of-two (SOPOT) terms [72, 73] for the \(i^{th}\) filter coefficient is,

\[
h_i = \sum_{j=0}^{B-1} 2^{a_{ij}}\tag{6.24}
\]

where \(B\) is the number of digits in the power-of-two representation. The expression for \(h_i\) can be written as,

\[
h_i = 2^{a_{i0}} \sum_{j=0}^{B-1} 2^{a_{ij} - a_{i0}} = 2^{a_{i0}} \left[ \sum_{j=0}^{B-1} 2^{c_{ij}} \right]
\]

where \(c_{ij} = a_{ij} - a_{i0}\). The term \(a_{i0}\) is known as the upper limit value shift. The bracketed term is known as the normalized value (n value). The shift and the
normalized value are analogous to the exponent and mantissa in true floating point representations.

Consider two 2’s complement W-bit numbers $X$ and $Y$,

$$X = -x_{W-1} + \sum_{i=1}^{W-2} x_{i-1} 2^{-i} \quad (6.25)$$

$$Y = -y_{W-1} + \sum_{i=1}^{W-2} y_{i-1} 2^{-i} \quad (6.26)$$

The $(2W - 1)$-bit ideal product $P_I$ can be expressed as,

$$P_I = MP + LP \quad (6.27)$$

$$MP = -p_{2W-2} + \sum_{i=1}^{W-2} p_{2W-2-i} 2^{-i} \quad (6.28)$$

$$LP = \sum_{i=W}^{2W-2} p_{2W-2-i} 2^{-i} \quad (6.29)$$

$$P_Q = MP_Q + \sigma X 2^{-(W-1)} \quad (6.30)$$

### 6.4.3 Example system - Digital down converter

Software radio receivers [88] require mixing, filtering and down sampling of received signals to allow data to be processed at a suitable rate. Part of this process can be achieved in FPGAs using a digital down converter (DDC). Mixing the incoming real signal from the analog to digital converters (ADC’s) to extract the complex signal, a DDC must filter the complex signal to reject image components introduced by the mixing process and then down sampling is done. DDC’s are most commonly implemented in logic in field-programmable gate arrays or application-specific integrated circuits. While software implementations are also possible, operations in the DDC, multipliers and input stages of the low-pass filters all run at
the sampling rate of the input data. This data is commonly taken directly from ADC sampling at tens or hundreds of MHz, which is beyond the real time computational capabilities of software processors. For maximum software radio flexibility, the ADC, mixer and filters should sample as quickly as possible. Figure 6.29 shows the architecture of DDC. Multiplication of the intermediate frequency with the input signal creates images centered at the sum and difference frequency (which follows from the frequency shifting properties of the fourier transform). The low-pass filters pass the difference (i.e. baseband) frequency while rejecting the sum frequency image, resulting in a complex baseband representation of the original signal. In its new form, it can readily be down sampled and is more convenient to many DSP algorithms. The sample frequency is now much higher than required for the maximum frequency in frequency band and so the sample frequency can be reduced or decimated, without any loss of information. Hence, if the DDC is implemented on an FPGA, full-parallel techniques can be used to reach the required sampling rates. The calculation of low pass filter coefficients for DDC specifications used in this research work are calculated using MATLAB, where sampling frequency is 200MHz, cutoff frequency is 40Mhz and attenuation band of 6dB is designed using kaiser window. Tables 6.1 and 6.2 represent the filter coefficient values for 4-tap and 8-tap filter respectively. The phase and magnitude response of 4-tap and 8-tap filters are shown in Figures 6.30 and 6.31 respectively.

**Table 6.1 Calculated 4-tap filter coefficients**

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>H(0)</td>
<td>0.169001</td>
</tr>
<tr>
<td>H(1)</td>
<td>0.330998</td>
</tr>
<tr>
<td>H(2)</td>
<td>0.330998</td>
</tr>
<tr>
<td>H(3)</td>
<td>0.169001</td>
</tr>
</tbody>
</table>
Table 6.2 Calculated 8-tap filter coefficients

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>H(0)</td>
<td>-0.08265</td>
</tr>
<tr>
<td>H(1)</td>
<td>0</td>
</tr>
<tr>
<td>H(2)</td>
<td>0.202832</td>
</tr>
<tr>
<td>H(3)</td>
<td>0.379824</td>
</tr>
<tr>
<td>H(4)</td>
<td>0.379824</td>
</tr>
<tr>
<td>H(5)</td>
<td>0.202832</td>
</tr>
<tr>
<td>H(6)</td>
<td>0</td>
</tr>
<tr>
<td>H(7)</td>
<td>-0.08265</td>
</tr>
</tbody>
</table>

Figure 6.29 A digital down converter architecture with low pass filter
Figure 6.30  (a) 4-tap low pass FIR filter magnitude response (b) 4-tap low pass FIR filter phase response
Figure 6.31  (a) 8-tap low pass FIR filter magnitude response  (b) 8-tap low pass FIR filter phase response
### 6.4.4 Distributed arithmetic based filtering scheme

Distributed Arithmetic was first brought up by Croisier [89], and was extended to cover the signed data system. Then it was introduced into FPGA design to save MAC blocks with the development of FPGA technology. High performance FIR filters based on DA using LMS architecture are implemented in [90, 91].

If \( h[n] \) is the filter coefficient and \( x[n] \) is the input sequence to be processed, the N-length FIR filter can be described as,

\[
y = <h, x> = \sum_{n=0}^{N-1} h[n]x[n] \tag{6.31}
\]

Distributed Arithmetic is introduced into the design of FIR filters as follows. In the two's complement system, \( x[n] \) can be described as,

\[
x[n] = -2^b x_B[n] + \sum_{b=0}^{B-1} 2^b x_b[n] \tag{6.32}
\]

Substitute equation (6.31) into equation (6.33) yields,

\[
y = -2^b x_B[n]h[n] + \sum_{b=0}^{B-1} h[n] \sum_{n=0}^{N-1} 2^b x_b[n] \tag{6.33}
\]

The equation (6.33) can be changed into another form,

\[
\sum_{b=0}^{B-1} \sum_{n=0}^{N-1} 2^b x_b[n] h[n] = \sum_{b=0}^{B-1} 2^b \sum_{n=0}^{N-1} h[n] x_b[n] \tag{6.34}
\]

Substituting equation (6.34) into equation (6.33) yields to the final form of distributed arithmetic,

\[
y = -2^b x_B[n]h[n] + \sum_{b=0}^{B-1} 2^b \sum_{n=0}^{N-1} h[n] x_b[n] \tag{6.35}
\]
The values of \( \sum_{n=0}^{N-1} h[n]x[n] \) can be conserved into a LUT unit and the relevant value can be called out according to the input data to save MAC blocks. Then, the weighted sum of \( \sum_{n=0}^{N-1} h[n]x[n] \) is calculated through shift registers and the result is \( \sum_{b=0}^{B-1} 2^b \sum_{n=0}^{N-1} h[n]x[n] \). In signed system, the signed bit should be taken into consideration so \( -2^b x[n]h[n] \) is also added. As a result, the final form of distributed arithmetic is defined as equation (6.35) and the implementation can be achieved on FPGA through LUT units. Figure 6.32 shows the DA based filtering scheme using reversible logic.

![Figure 6.32 Concept of distributed arithmetic filtering](image)

6.4.5 Proposed DA based filtering scheme using multiplexer

The basic LUT-DA scheme on an FPGA would consist of three main components- the input registers, the 4-input LUT unit and the shifter/accumulator unit. Additionally, it would require a control unit to manipulate the filter operation, and an adder tree unit to perform addition of partial filter results. Applying this approach, the 4-input LUT unit will not be directly accessed instead 2-input LUT is
used based on multiplexer select. The use of multiplexer incorporates savings in terms of adder offsets and results in an overall improvement in area and performance metric.

The proposed architecture achieves high throughput and low complexity in two ways as explained in Figure 6.33. Firstly, by using multiplexer based distributed arithmetic which allows the accumulation to complete using only a single accumulator and secondly, parallelization of the coefficient computations for higher tap filter. The combination of the multiplexer with distributed arithmetic helps to compute higher filter taps without increasing the complexity of filter. Thus, the switching logic for both the input samples and the coefficients becomes faster. In addition, since the lookup tables contain all the possible combinations of the coefficients. The coefficients are stored in an already multiplexed fashion so extra logic is not required for computation.

![Figure 6.33 Concept of multiplexer based DA filtering scheme](image)

The multiplexer based DA filtering using reversible logic is shown in Figure 6.34. The particular 2-input LUT is selected which represents all the possible sum combinations of filter coefficients. It implies about 50% reduction in the number of LUT used with increased speed. To evaluate the performance of the proposed scheme, 4-tap and 8-tap low pass FIR filters are implemented using VHDL and synthesis is carried out in Xilinx-ISE8.1i. The results are compared with Rgate
logic [41,42] based DA implementation using SOPOT and CSD method for coefficient representation.

Figure 6.34 Multiplexer based DA filtering scheme using proposed logic

6.4.6 Comparison results

The evaluation of device utilization using proposed DA architecture can be explained easily with the help of the results in graphs shown below. Also, it is clear that CSD representation gives better results as compared to SOPOT representation. Figures A 3.1-A 3.14 of Appendix 2 are showing the simulation results.

Figure 6.35 reports the comparison of number of LUTs used among the various filter architectures designed using Rgate [41, 42] based DA and proposed multiplexer based DA method. It is shown that proposed multiplexer based DA filter using NTG logic comprehends the existing DA based low pass FIR filter using Rgate logic and the number of LUTs are reduced by 30% in case of 4-tap low pass FIR filter design and this reduction is 25% in case of 8-tap low pass FIR filter
design, which is due to the fact that the multiplexer based technique in DA has reduced size of LUTs so device utilization is improved. Also, it is clear that CSD coefficients representation gives better device utilization.

Figure 6.36 plots the comparison of number of occupied slices among various filter architectures designed using Rgate [41, 42] based DA and proposed method. It is shown that the proposed multiplexer based DA filter using NTG logic has 45% reduced number of occupied slices as compared to existing DA based filter using Rgate logic in case of 4-tap low pass FIR filter design and this reduction is 40% in case of 8-tap low pass FIR filter design on an average for both SOPOT and CSD coefficient representation methods.

![Comparison of number of LUTs used among filter architectures](image)

**Figure 6.35 Comparison of number of LUTs used among filter architectures**

Figure 6.37 reports the comparison of number of gates used among various filter architectures designed using Rgate [41, 42] based DA and proposed gate based DA method. It is shown that the proposed architecture has 40% lesser number of gates in case of 4-tap low pass filter design and number of gates are 32% lesser in case of 8-tap low pass filter design as compared to existing scheme to design DA based FIR filter.
Design of FIR filter using reversible logic

Figure 6.36 Comparison of number of occupied slices among filter architectures

Figure 6.37 Comparison of number of gates used among filter architectures
Figure 6.38 Delay comparison among filter architectures

Figure 6.38 represents the delay comparison for 4-tap and 8-tap filter designed using Rgate [41,42] based DA and proposed gate based DA method. The proposed method outperforms by 15% speed improvement.

Figure 6.39 represents power calculation results. Though the proposed method has bit higher power consumption as compared to Rgate [41, 42] based DA method because of the multiplexer based LUT which increases the number of look up access and thus switching. It is shown that using CSD coefficients the power consumption is reduced as compared to SOPOT coefficients for filter design because CSD computation requires lesser number of 1’s.

Figure 6.40 compares the delay and power consumption for existing DA based 4-tap FIR filter and proposed DA based 4-tap FIR filter using CSD coefficients. It is shown that 14% improvement in delay is achieved using proposed method. Similar results are obtained for 8-tap FIR filter. Though the proposed method has 10% higher power consumption as compared to existing DA based method because of increase in switching activity with increase in number of accesses per look up table. However, overall performance metric (i.e. power, delay
and number of gates product) for proposed design is improved by 45% as shown in Figure 6.41.

![Figure 6.39 Power consumption comparison among filter architectures](image1)

![Figure 6.40 Delay and power consumption comparison among filter architectures](image2)
Figure 6.41  Power, delay and number of gates product performance among filter architectures

6.5 SUMMARY

An efficient reversible logic based DA scheme is presented which is used to implement FIR filters using SOPOT and CSD representation for the filter coefficients. The device utilization of the proposed architecture is relatively less since it uses split LUT technique with multiplexer select logic. The 4-tap and 8-tap FIR filters designed in this research work can be extended even for more taps. A high speed and less area implementation is achieved. The simulation results indicate that the designed filter using proposed distributed arithmetic can work stable with high speed and can save almost 40 percent hardware resources.

Meanwhile, it is very easy to transplant the filter to other applications through modifying the order of filter and other parameters, and therefore they have great practical applications in digital signal processing. This research work discusses FPGA implementation of finite impulse response (FIR) filters using their application in digital down converters (DDCs) for software radio as a background based on reversible logic which has fault detection property. The implementation is based on distributed arithmetic (DA) which substitute multiply and accumulate operations
with a series of look-up-table (LUT) accesses. Canonical signed digit (CSD) representation is used for the coefficients and it is compared with sum-of-powers-of-two (SOPOT) technique of coefficients representation. The proposed DDC is implemented in VHDL and verified via simulation. The proposed method offers average reduction of 30% in the number of LUTs, 42% reduction in occupied slices and 38% reduction in the number of gates needed for low pass FIR filter implementation method. The proposed design shows 14% reduction in delay as compared to Rgate logic based DA architecture. Though there is power trade off but there is significant improvement in overall performance of FIR filter with 40% reduced hardware resources.