Abstract:

Realizing reconfigurable FIR filter with low complexity, low power and high speed is a challenging task. Systematic and random variation in process, temperature and supply voltage is posing major challenge to future high performance micro-electronic architectures. This thesis addresses the research problem of incorporating low power, high speed and reconfigurablity into the FIRfilter architecture for wireless receiver. Finite impulse response (FIR) filters are commonly employed in wireless receivers. Generally high order FIR filters are required to meet the stringent adjacent channel attenuation specification of wireless communication standards. Coefficients multiplication of such high order filters consume much power and chip area. The purpose of this project is to propose efficient method of multiplications in FIR filters. Apart from low complexity, reconfigurablity of these FIR filters. In this thesis, couple of methods are mentioned to realize reconfigurable low power FIR filter. In order to make FIR filter power efficient few things are taken into account, the number system in which the filter coefficients are represented and architecture of adders used to implement that FIR filter. In this thesis for both parts research has been done and few results have been shown. In this project following things have been done

- Literature review has been done to understand the concept of power efficient implementation of filter coefficient multiplication.
- Different taps of FIR filters have been implemented and their area and power are compared, using different ways of coefficient representation.
- In order to check effects of coefficient representation, I have extended the idea given by K.P Gopi Smitha for better representation of filter coefficients and algorithm to implement coefficients multiplication efficiently.
- Comparison of couple of methods proposed in literature, about power efficient implementation of filter coefficient's multiplication, to learn the way things progress in this field.
- Last but not the least full adder architectures have been checked and their comparison has been given keeping standard setting into mind. These full adders architecture have been compared in order to check process variation with respect to applied voltage. As the basic idea of the project is to implement power efficient filter realization.

Table of Contents

Table of figures:

Table of tables

Chapter 1

1 Introduction:

1.1Aims and Objectives:

Digital filtering is key function of the channel filter and it is achieved with the help of finite impulse response (FIR) filters due to their stability and linear phase property. The aim of this project is to study design considerations with respect to variations (process, temperature etc.) and low-power in re-configurable FIR filters. It is proposed to explore systematic design method and low power design methodologies considering variability into account for given specification of a FIR filter. A standard multi-standard filter specification will be considered and then filter will be design in Verilog or VHDL.

The objectives of the project are:

1. Research into design considerations with respect to variations and low-power in re-configurable FIR filter.
2. Explore re-configurable (including low power consideration) design methods, taking into account variations in the critical path of the filter.
3. Design and implement a multi-standard FIR filter. Then to analyse area, path delay and power overhead for re-configurability.
4. Wherever possible use common sub-expression elimination algorithm to reduce the complexity of multiple standard filter.

For wireless receiver, reconfigurable filters are required and they must be further optimized. The conventional method of achieving reconfigurablity is by switching the operation among distinct receivers based on the current mode of operation. This is not an efficient approach as complexity and power consumption will be high. Realizing reconfigurable filters with low complexity is a challenging task. The motivation behind the work in this thesis is to deal with these research issues related to the realization of low power reconfigurable digital filters mostly FIR.

1.2 Review:

The demand for new wireless services requiring high capacities and data rates have motivated the development of new generation cellular wireless communication systems. Emerging wireless communication systems are aimed at accommodating this demand through better resource management and improved transmission technologies. The coexistence of different cellular systems requires multi-mode, multi-band and multi standard mobile terminals. The complexity and portability of wireless systems places stringent requirements on the total power consumption of signal processing and communication systems. Providing low power while maintaining low cost is the primary challenge for communication systems as the carrier frequencies are approaching GHz. Systematic and random variation in process, temperature and supply voltage is posing major challenge to future high performance micro-electronic architectures. Technology scaling beyond 90nm posing is causing higher level of device process variation [11] [12]. The demand for low power cause supply voltage scaling hence making voltage variation as significant part of overall challenge. Finally the quest for growth in higher operating frequency has led to temperature variations as a part of challenge too.

Transceivers for multi-mode and multi-standard are often implemented by replicating the receiver circuits for each standard. On contrary, transceivers that can share circuit functions between different standards in adaptive manner have the advantages of low power consumption, small chip area, most importantly, the potential for low cost. Although high level of integration is possible on silicon, the additional radio frequency hardware required to implement this type of multi-standard radio increases the total consumption. The need for multi-standard transceivers which can operate simultaneously for each independent standard, by reconfiguring the hardware, resulted in the evolution of software defined radio. The term Software radio receiver signifies that, the hardware architecture can be reprogrammed to adapt with any wireless radio standard. Software radio receiver finds major application in mobile communication transceivers, generic cellular base stations, cognitive radio systems and military radio systems. Some of the benefits that will result with the realization of software radio receiver are [8],[9]

- Easier international roaming, enhanced flexibility for various services, improved personalization and choice for subscribers of mobile services
- The potential to rapidly develop and introduce new value added services and revenue streams with increased flexibility of spectrum management and usage for mobile network operators.

- The promise of increased production of flexibility contributing to rapid production advancement for handset and base station manufacturers.
- The prospect of increased spectrum efficiency resulting in better use of spectrum resources for regulators.

The cellular base stations employ distinct receiver chain comprising of analog components such as analog mixers, local oscillators and analog –to-digital convertors for each communication standard. Distinct chain of such components along with processing of baseband units linearly increases the complexity of communication receiver with number of received radio channels of different communication standards. But radio receiver based base stations employ a single analog and digital front end to receive the radio channels of all communication standards. The DFE, which does the entire signal processing tasks, is reconfigured to work for different cellular standards. Hence the cost of analog part is independent of received different standards. The rapid transition of communication technology from analog to digital domain has been fulfilled with the introduction of SDR. This has enabled different communication functionalities such as digital up/down conversion, modulation/demodulation, to be performed by software on an appropriate digital hardware. The use of such programmable digital hardware has improved the flexibility in communication receivers to a significant extent.

Digital front end replaces the analog signal processing with digital signal processing in software defined radio receiver. Digital front end comes immediately after analog to digital convertor; it has to operate at highest sampling rate. Due to large bandwidth and dynamic range of wideband input signal, the channel extraction needs to be implemented with low complexity and they have to be reconfigured for different wireless standards. Hence the complexity reduction, high speed operation along with reconfiguration is tricky task.

The focus of the work presented in this thesis is on low power implementation of reconfigurable FIR filters used in digital front end receivers. Receivers use same hardware that can be programmed for any number of multi standard channels by reconfiguring digital front end. In addition to these reconfigurablity requirements, the hardware employed for receivers must also meet the stringent power and speed specifications of wireless communications systems.[9] Whereas the main review of project can be summed up in the following diagram,

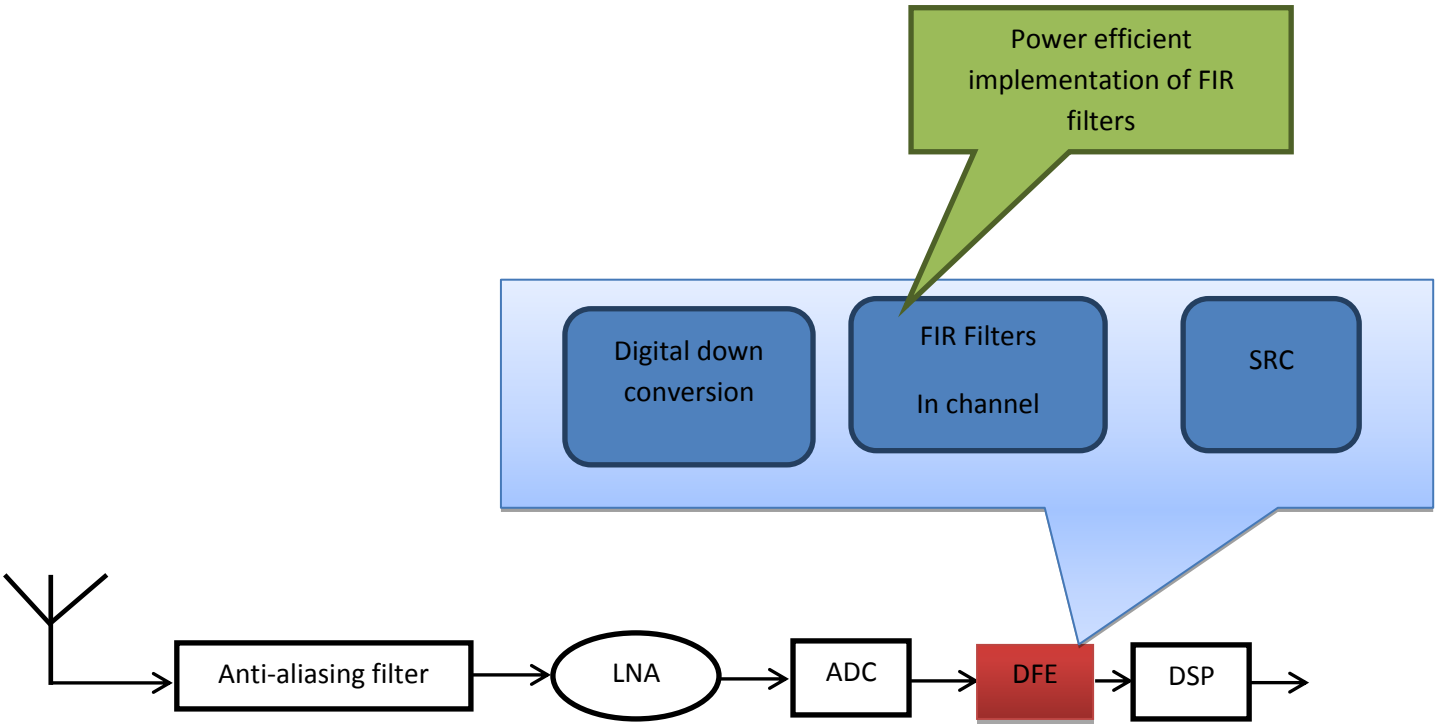

Figure 1: Main idea of project

1.3 Project Outline:

This thesis is organised as follows:

- Firstly present literature review of wireless receiver and its functionality, along with overview of FIR (channel) filter, a background of the work done so far.
- Secondly, the work done during this project mostly on coefficients of FIR filter which are the key part of whole discussion.
- the conclusions and future extensions.

Chapter 2

2 Background Review:

In this part, the basic concept of the receiver is presented. The emphasis of this part is to provide a good understanding of process variation, wireless receiver and also FIR filter [8],[9].

2.1 Wireless Receiver:

A wireless software receiver can be explained by following key points.

- Flexible transmitter (TX)/ receiver (Rx) controlled and programmable by software.
- Signal processing able to replace, as much as possible, radio functionalities.
- “Air-interface download ability”
- Software realization of terminals
- Transceiver, where following can be defined
    1. Frequency band and radio channel bandwidth
    2. Modulation and coding scheme
    3. Radio resource and mobility management protocol
    4. User application.

This parameter can be adapted and can be changed by network operator. The flexibility of software receiver is to operate in multi-service environment [9].

2.1.1 Computational Complexity:

The basic computation involved in digital filtering are multiply and accumulator (MAC). This is basically related to coefficient multiplication of FIR filters in this case mostly focus will be on addition. The accumulate operation is done by adders. The complexity of filter implementation is dominated by the number of multiplications as multiplication requires multiple adders and shifts for implementation. If dedicated multipliers are used for coefficient each multiplication in the channel filters, it will result in large chip area and huge power consumption. This is unacceptable from a hardware complexity point of view.[5]

In digital filter, the power consuming operation is multiplication, which is implemented by using adders and shifters. The number of adders needed to implement the co efficient multiplication operation dominates the complexity of

fixed coefficient digital filters as shift operation can be hardwired. The adder termed as logic operators (LO) are required for computing the sum of partial product in the coefficient multiplier. The numbers of LOs is significant in determining the area and power consumption requirement of filters.

2.1.2 Filter Reconfigurablity:

Several implementation approaches for reconfigurable FIR filters have been proposed in literature. These designs include either a fully programmable multiply-Accumulate (MAC) based filter process or dedicated architecture where coefficients are stored in registers. The method proposed in this thesis is multiband filter and masking method, as this channel filters are used for software defined radio receiver most specifically for digital front end of software radio receiver. Variations in power consumption can also be noted during this implementation. By the end of this project most probably efficient and less power consuming as well as reconfigurable FIR filter will be proposed with some basic simulations and their results will be present for analysis.

2.2Process Variation Analysis:

Systematic and random variation in process, temperature and supply voltage is posing major challenge to future high performance micro-electronic architectures. Technology scaling beyond 90nm (nano-technologies) posing is causing higher level of device process variation [11] [12]. The demand for low power cause supply voltage scaling hence making voltage variation as significant part of overall challenge. Finally the quest for growth in higher operating frequency has led to temperature variations as a part of challenge too. Process tolerance with low power consumption represents contradiction in design analysis.

Over the past few years, statistical design approach has widely been investigated as an effective method to ensure yield under process variations. In this approach, the design space is explored to optimize certain design parameter (e.g., power) to meet a timing yield and a target frequency. Several gate level sizing and/or Vth

assignment techniques [13] have been proposed recently addressing the minimization of total power while maintaining the timing yield. On the other end of the spectrum, design techniques have been proposed for post silicon process compensation and process adaptation (e.g., adaptive body biasing [14]) to deal with process-related timing failures. Due to quadratic dependence of dynamic power of a circuit on its operating voltage, supply voltage scaling has been extremely effective in reducing the power dissipation. Researchers have investigated logic design approaches that are robust with respect to process variations and, at the same time, suitable for aggressive voltage scaling. One such technique called Razor [3] uses dynamic detection and correction of circuit timing errors to tune processor supply voltage. This technique eliminates the need of voltage margins and supports dynamic voltage scaling for power reduction.

Keeping all this into view when designing a full adder transistors size and their voltage variations are also considered in order to obtain a power efficient circuit for required implementation and also for required frequency requirements. By the end of this project for full adder design these parameter should be taken into account and will be tested using HSPICE simulation starting from basic full adder circuitry. The basic emphasis of this project is to take into account of voltage variation by keeping in view the size, width and resistor parameters.

2.2.1 Test Circuit:

Usually, a simple replicable circuit or a benchmark circuit where performance and working can be easily monitored is chosen. A ripple carry adder consists of a chain of full adders where the carry output of the least significant bit (LSB) adder goes into the next adder. This way, the carry signal "ripples" through the chain of adders hence the term, ripple carry adder. For this thesis full adder is chosen to verify all the result

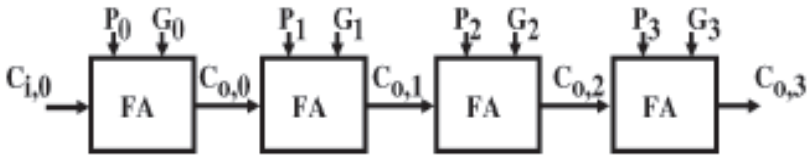

Figure 2: Ripple carry adder [15]

2.3Low Power and Reconfigurable Implementation Approaches for FIR Filters:

The main focus of this project is FIR filtering (channel), a process of bandpass filtering, where radio channels from a wideband input signal are extracted for further baseband processing. The channel of interest may be of equal or different bandwidths frequency band. Thus, the filter have to attenuate adjacent channel interferes and must meet the blocking characteristics specified. Due to these stringent adjacent channel specifications, high order channel filters are required. As order of filter increases complexity of filter increases, which in turn increase the area and power consumption? Therefore, implementing digital filter with small area and low power consumption is vital for reducing overall complexity of channelizer. The channel filters are directly placed after ADCs. Hence they have to operate at high sampling rates and consequently at high speed. This thesis represents low power, less complex and reconfigurable filter design approaches. [3], [7]

2.3.1 Filters for wideband receivers:

Digital filters can be infinite impulse response (IIR) filters or finite impulse response (FIR) filters. IIR filters can be realized using recursive algorithms as it is an infinite length response filters. The main advantage of IIR filters is that they can produce steeper slope for given number of coefficients when compared to FIR filters. But IIR filters are difficult to be reconfigured due to presence of pole. This is because IIR filters become unstable as new pole value comes after reconfiguration. But no such problem exists for FIR filters due to absence of pole. The main advantage of using FIR filter group delay is constant. This provides the capability to obtain both steep and linear phase response. The only disadvantage of FIR filter is they take more area and cost as compared to IIR filter. FIR filters are all zero filters hence are stable.[4],[6]

2.3.2 Advantages of FIR filters:

Compare to IIR filters FIR filters have some advantages are described below

They are linear phase filters. They delay the input signal but don not distort the signal. FIR filters are suited to multi rate applications. Whether the signal is decimated or interpolated, the use of FIR filters allows the omission of some calculations, thus providing advance computational efficiency. In contrast, if IIR filters are used, each output must be calculated individually even if that output is discarded. In practice, all digital filters must be implemented using finite-precision arithmetic, that is, a specific number of bits. The use of finite –precision arithmetic while using IIR filter cause stability issue. But FIR filters cannot have that issue as they do not have feedback. FIR filters are less sensitive towards coefficient quantization compared to IIR filters.

FIR filters are widely used as channel filter due to their linear phase property and stability. FIR filters can be realized by direct form (DF) or transposed direct form (TDF). The direct form realization which is the most intuitive implementation of the time domain filter transfer function, the FIR filter equation is

$$y[n] = \sum_{k=0}^{N} b[k]x[n-k]$$

And diagram for this equation is as given below.

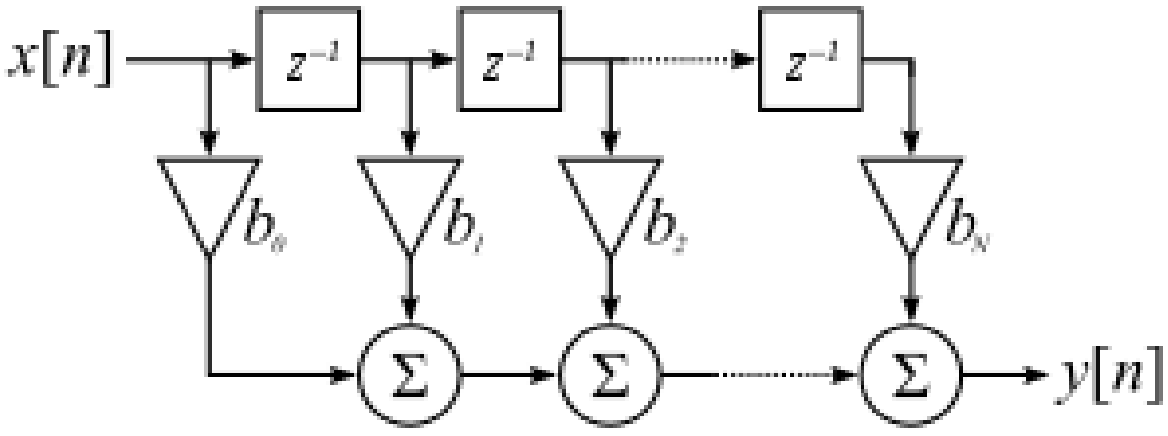

Figure 3: Filter discrete time FIR filter of order N

where

$x[n]$ is the input signal,

$y[n]$ is the output signal,

b_i are the coefficient of filter that make up impulse response,

N is the filter order.

A DF FIR filter output is formed by accumulating the result of delayed version of input signal, multiplied by filter coefficient (filter tap coefficients). The critical path delay of DF filter is given as

$$\mathrm{T}df \;=\; \mathrm{t}mult \;+\; (\mathrm{N}-1)add$$

Where $\mathrm{t}mult$ is the multiplier tap delay of adder? Thus the critical path length for DF FIR filter is directly proportional to the number of filter taps, N. For N-tap filter shown in figure 3 N multiplication and (N-1) additions are required as a result speed of DF operation is relatively low especially when high order filters are implemented. The critical path delay of DF structure can be reduced by applying transition theorem to obtain the TDF FIR filters as shown in figure

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k]$$

The critical path delay for the TDF FIR filter is given by:

$$Ttdf = tmult + tadd$$

Since the critical path is independent of number of taps, TDF has been the preferred architecture for high speed, high-order FIR channel filter implementations. For N-tap FIR filter, as filter coefficients are symmetric or anti-symmetric, the number of multiplications can be reduced to N/2 for N even and to(N+1)/" for N odd. Thus TDF structure offer less power consumption as compared to that of DF structure.it can be noted that multiplication in TDF is done parallel way[5]. This can be shown as given figure

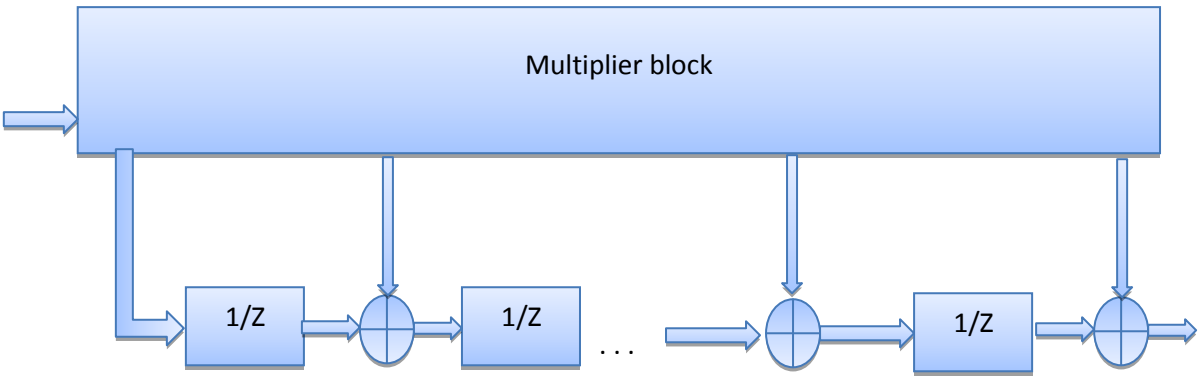

Figure 4: Modified TDF FIR filter structure with common multiplier.

Chapter 3

3Computational complexities Analysis of FIR Filters:

Major computation involved in digital filtering is multiplication and accumulation. The accumulation is done by adders; multiplication is done by multiple adders and shifts. While the complexity of filters is determined by multiplication, as this will also determine the chip area and power consumption for the filter. While digital filtering is implemented in battery powered mobiles so this should be taken into account that power consumption are less. The work presented here only Transpose direct Form filters are discussed, this is done because its critical path delay is independent of the filter length, and these are high speed filters as compared to direct form. As mentioned earlier that in digital filtering most power consuming part is co efficient multiplication, which is done by adders and shifts. Adders required for calculating sum of partial product in the coefficient multiplication. Where the numbers of adder are important in determining the area and power consumption of filter. While speed of filter depends on not only number of adders but also number of adder steps in the maximum path. The efficiency in reducing the cost metric (power, area and increasing speed) of the filter directly influences the performance of target hardware.

The number of adders required for coefficient multiplication is directly proportional to number of non-zero bits in quantized filter coefficient. Thus the number system with lesser number of non-zero digits is widely used for coefficient representation compared to conventional binary number system. Among such systems most widely used is canonical signed digit representation is used and also minimal signed digit is popular. Canonical singed digit number system is signed digit number system that minimize the number of non-zero digit, hence reduce the complexity, in this technique no two adjacent digit are non-zero i.e. A number c0c1c2c2c4…cn-1 is said to be canonical signed digit represented if ci*ci-1=0. On average canonical signed digit representation offer reduction of 33% of non-zero digits compared to binary representation. Hence 33% less number of adders required. While minimal signed digit representation gives lesser number of non-zero digit, but the only drawback is that it has many representation for same number. While the major problem of using these number representation system is filter needs to be reconfigured.

Once the number system is chosen for filter coefficient representation, the next step is to look for further hardware reduction (optimization) if possible by minimizing non-zero digits while satisfying the filter characteristics such as pass band ripple and stop band attenuation. While digital filtering can be treated as multiplication of one variable (input signal) with multiple constants (coefficients) named as multiple constant multiplication. This method was proposed for both direct form and transpose direct form filters to exploit the common computation between multiplications. It was given in the work done before that when filter coefficients are represented in canonical signed digit representation, a certain 3-bit common sub expression like [1 0 1] and [1 0 -1] and their negated versions are repeated many times. These repeated patterns results in redundant additions, hence they need to be implemented once. Thus by employing common sub expression elimination technique, it has been shown that 38% reduction in the number of adders is achieved for MCM based on FIR filters. Lots of work has been done on common sub expression within a coefficient called horizontal common sub expression and these methods referred to as horizontal common sub expression elimination methods. One example for this while number system used is canonical signed digit is given h[k] = [0.10-1010101010010-1]. In direct implementation using shift and add, the output of filter is

$$h[k] = 2^{-1}\,x1 - 2^{-3}x1 + 2^{-5}x1 + 2^{-7}x1 + 2^{-9}x1 + 2^{-11}x1 + 2^{-14}x1 - 2^{-16}x1$$

Where x1 is the input signal. It requires 7 adders to implement above equation. It can be noted that bit pattern [1 0 1] and [1 0 -1] are repeated twice in h[k], which can be expressed as x2=x1+2^{-2}x1 and x3=x1-2^{-2}x1. Using common sub expression filter output can be expressed as

$$h[k] = 2^{-1}\,x3 + 2^{-5}x2 + 2^{-9}x2 + 2^{-14}x3$$

here we need only 5 adders. While vertical common sub expression elimination techniques were also proposed in the literature.

Digital filtering can be treated as multiplication of one variable (input) with multiple constants (filter coefficients) this term is named as multiple constant multiplications. It is closely related to substitution of multiplication with shifts and additions. This is also named as transpose direct form of filtering which is much beneficial as compared to direct form. In direct form one variable (input) is multiplied with one constant and then shifted, this will increase delay and little computational overheads too.

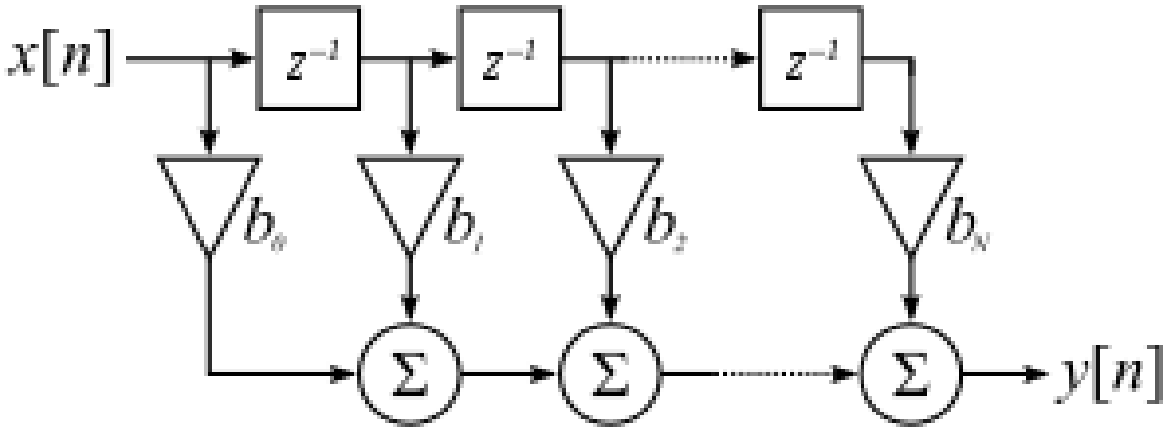

Figure 5: Direct form filtering.

The advantage of using transpose direct form is that it provided the chance for eliminating repetitive or redundant multiplications. This may enhance the scope of its power and domain of applications. Using various number representation system one can significally reduce number of adders in coefficient multiplication literature have many such methods those are discussed in detail how they reduce the complexity. While in this thesis our main concern is about binary representation of number (constants) and various proposed techniques for this such as common subexpression elimination techniques. In this context two techniques have been discussed one is binary subexpression elimination of 3 bits and other is binary common subexpression elimination of 4'bits. These two are for horizontal common subexpression elimination, while in literature one other method is discussed named as vertical common subexpression elimination technique, where the authors exploited the fact that many vertical common subexpression exists between adjacent filter coefficients. They have tried to exploit the similar bit pattern in between adjacent filter coefficients. This proposed structure is more efficient than the horizontals structure only for coefficients with word length is relatively small.

The vertical common subexpression elimination techniques can be explained using example in the table. The number in first row represents the number of bitwise right shifts. The coefficient multipliers are realized using vertical common subexpression elimination technique [1 1]=x4=x1+x[-1] and [1 -1]= x5=x1-x[-1], where x1[-k] represents input x1 delayed by k units. Using vertical common subexpression elimination technique, the filter tap output can be expressed as

$$y[k] = 2^{-1}\, x5[-k] - 2^{-3} x5[-k] + 2^{-5} x4[-k] - 2^{-8} x1[-k] \quad \text{(A)}$$

The output for the symmetric part is given by

$y[n-k-1] = -2^{-1}\, x5[n-k-1] + 2^{-3}x5[n-k-1] + 2^{-5}x4[n-k-1] - 2^{-8}x1[n-k-2]$(B)

Where n represents the filter length

Bit Shifts	1	2	3	4	5	6	7	8
H0	1	0	1	0	1	0	0	1
H1	1	0	1	0	1	0	0	0

Table 1: Showing Canonical signed digits

It has been reported that $y[n-k-1]$ can be obtained from y[k] by simple delay operation due to the sign differences. While we look at above (A), (B)there we can see one more difference that is off last term. In this case we need one extra adder. Thus to conclude, when vertical common sub expression is used, additional adders are needed not only when bits are of opposite sign but also when the sub expression are unpaired which exist in the symmetric second half coefficients have different delays compared to the symmetric first half coefficients.

A new binary based common subexpression elimination technique was proposed, which produce better reduction of adders compared to other number representation systems especially for higher order FIR filters. This gives better reduction in adder with less trade off speed. Few methods of binary system are proposed like 3-bit and 4-bit binary horizontal common subexpression elimination. As this uses binary representation of coefficients so no additional overhead will be needed as compared to other two discussed. However the overhead of additional adder is due to presence of opposite dign bit in the co efficient representation, but also due to the delay differences between nonzero terms of adjacent coefficients, when vertical common subexpression elimination is applied.

3.1 Low power FIR filters realization using binary common subexpression elimination technique:

For general higher order filter used in wireless receivers, must be realized to consume less power and operate at high-speed. Although programmable reconfigurable filters are available they are not best as they consume much power and their speed is also low. The number of adder used to implement the coefficient multiplication determines the complexity of FIR filters. A new 4-bit binary representation-based common subexpression elimination method and 3-bit binary representation-based common subexpression elimination method which employ 4-bit and 3-bit for implementing higher order filter with fewer number of adders as compared to canonical signed digit and other proposed.

3.1.1Binary common subexpression elimination technique:

A 3-bit binary is the best known common subexpression elimination technique, the of proposing 4-bit common subexpression elimination technique is to give better coverage, to reduce number of adders as more number of non-zero bits are grouped together. 4-bit binary subexpression elimination technique is different from 3-bit technique in two aspects

1. Binary common subexpression elimination 4-bit uses 4 bits while 3-bits uses 3 bits to be grouped together. As more number of non-zero bits are grouped together so large number of adder reduction is possible.
2. 4-bit binary common subexpression elimination excludes the use of vertical common subexpression so that also reduces adders used for overhead.

This can be explained by example, 8-tap, 12-bit word length FIR filter shown in table B

	1	2	3	4	5	6	7	8	9	10	11	12
H0	1	0	0	0	0	0	1	1	0	1	0	0
H1	1	0	0	0	1	0	1	0	1	1	0	1
H2	0	0	0	1	0	0	0	1	0	1	1	1
H3	0	0	0	1	0	0	0	1	1	1	0	0
H4=h3	0	0	0	1	0	0	0	1	1	1	0	0
H5=h2	0	0	0	1	0	0	0	1	0	1	1	1
H6=h1	1	0	0	0	0	0	1	0	1	1	0	1
H7=h0	1	0	0	0	0	0	1	1	0	1	0	0

Table 2: Binary representation of 8-taps FIR filter

The direct implementation is defined as the implementation of the multiplier using shifts and adds, without employing any common subexpression. In this example number of non-zero bits are 19, while it is 8 taps filter. So it requires

Number of adders = (no of non-zero bits -1) + [filter tap/2]

Thus it requires 22 adders. While reducing the number of adders, the horizontal common subexpression elimination is employed, where x2= [1 1], x3= [1 0 1] and x4= [1 1 1] are indicated in the table. If x1 is input signal, using above technique, the output y1 corresponding to coefficients h0 to h3 can be written as

$$y1 = 2^{-1}\,x1 + 2^{-7}x2 + 2^{-10}x1[-1] + 2^{-5}\,x3[-1] + 2^{-9}\,x2[-1] + 2^{-12}x1[-1] + 2^{-4}\,x1[-2] + 2^{-8}\,x3[-2] + 2^{-11}\,x2[\text{-}2] + 2^{-4}\,x1[-3] + 2^{-8}\,x3[-3]$$

Note that x4 has three ones still it requires single adder, while the output of x2 is shared while realizing x4.

If in above example vertical common subexpression elimination is employed, then [1 1] given x5=x1+x1[-1], is indicated inside dotted rectangles. Using this vertical common subexpression the expression for output y2 can be written as

$$y2 = 2^{-1}\,x5 + 2^{-7}x2 + 2^{-10}x1 + 2^{-5}\,x3[-1] + 2^{-9}\,x2[-1] + 2^{-12}x1[-1] + 2^{-4}\,x5[-2] + 2^{-8}\,x3[-2] + 2^{-11}\,x2[\text{-}2] + 2^{-8}\,x4[-3]$$

So by implementing the technique number of adders reduces as in this example 9 adders required for y2 and 4 adders to realize x2, x3, x4 and x5. Thus total 13 adders are needed to realize the symmetric first half of the filter. Though vertical common subexpression elimination reduces number of adders for first symmetric half of the filter coefficients but for second half it require additional adders to realize the symmetry. This is drawback of vertical subexpression technique.

While 4-bit common subexpression elimination technique uses only the bits that occur within each coefficient called as horizontal common subexpression and hence the symmetry property of FIR filters is completely exploited. Using 4-bit

technique, the output y4 corresponding to coefficients h0, h1, h2 and h3 can be expressed as

$y4 = 2^{-1}\, x1 + 2^{-7} x7 + 2^{-1} x1[-1] + 2^{-5}\, x3[-1] + 2^{-9}\, x7[-1] + 2^{-4} x1[-2] + 2^{-8}\, x3[-2] + 2^{-11}\, x2$[-2]$+2^{-4}\, x1[-3] + 2^{-8}\, x4[-3]$

Where x7= [1 1 0 1] and x2 to x4 are as defined earlier. While for this example adder steps are 2 for both techniques. Note that for largest 3-bit common subexpression, [1 1 1] = $\mathrm{x1} + 2^{-1}\,\mathrm{x1}+2^{-2}\,\mathrm{x1}$ is 2 adder steps, and for 4-bits, [1 1 1 1] = $\mathrm{x1} + 2^{-1}\,\mathrm{x1}+2^{-2}\,\mathrm{x1}+2^{-3}\,\mathrm{x1}$ is also 2 adder steps. The largest 3-bit and 4-bits represents the worst case among all the possible combination of 3-bit and 4-bits. It is clear that both 3-bits and 4-bits required same number of adder steps, while reason of not using 5-bits or more bits is number of adder steps. Using of higher number of bits was investigated although it gives us high bits coverage, number of adders to realize the filter increases. The frequency of higher order binary horizontal common sub expressions of 5-bits is analysed, which is realized as superset of 4-bit. The example filter with word length 8-bits to 24 bits, different pass band and stop band specifications for different filter length from 10 taps to higher is given and results checked with increased number of adders. The reduction in adders is less significant may be because of following reasons

The probability of repeated occurrence of such higher-order common sub expressions is less in practical filter.

Additional adders are required to realize the higher order filter which also increases the hardware overhead too.

Other reason of increase in adders is due to over head of extended number of adder steps, as greater than 4-bits are used to be grouped together.

Due to above disadvantages, the higher order common subexpression with more than 4-bits are not taken into account.

3.1.2Comparison of Binary Subexpression Elimination 3-bits and 4-bits:

Binary subexpression elimination 3-bits and 4-bits are widely used now a days they both have some advantages as well as some disadvantages, they both share some common advantages too, like they both have same adder steps which makes both of them faster and reliable. They both give non-zero bit coverage. In spite of all this binary common subexpression 4-bits have some extra advantages that make it more valuable as compared to 3-bit technique. These advantages are given as

1. Binary common subexpression 4-bits required fewer number of adders as it group large number of non-zero bits, hence gives better coverage of non-zero bits.
2. It uses only horizontal common subexpression unlike binary common subexpression of 3-bit that also uses vertical common subexpression too, thus this can reduce additional adder overheads.
3. It will have identical adder steps with respect to binary subexpression of 3-bits; hence maximum path length for both remains same.

While examining both of them the number of adder's reduction can be seen and this reduction is shown in the table given below

3.1.2.1 Experimental Analysis:

In this project BCSE 3'bits and BCSE 4'bits are discussed and number adder required for them with varying word length, the results are shown in histogram

Filter length of 256 taps,

Word length	BCSE 4'bits (number of adders)	BCSE 3'bits (number of adders)
12	78	80
14	113	118
16	142	148
18	164	194
20	207	251
24	320	356

Table 3: Adder's reduction keeping taps constant

The results shown in table (a) can also be represented in a graphical way, the graph below showing number of adder variations for BCS 3-bits and 4-bits when filter taps or filter length held constant (at 512 taps),

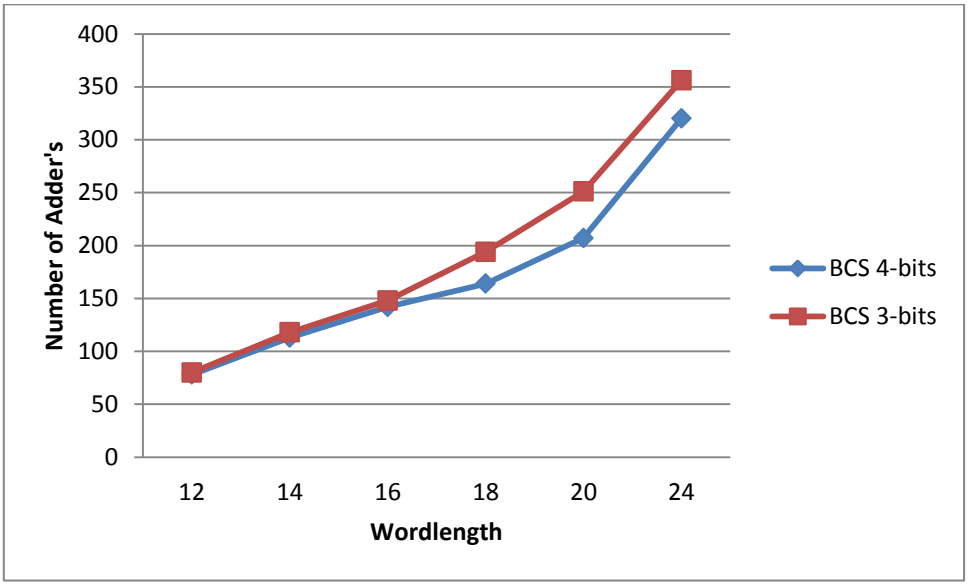

Figure 6: Graph showing number of adder's versus wordlength for BCS3-bit, 4-bit when filter length held constant (512 taps).

This reduction in adders is also found in lower order filters also. This reduction in number of adder's id significant for higher order filter as more common subexpression appears and can be grouped together to get better reduction while for lower order filter this cannot be seen.

3.1.2.2 Design Examples:

Several FIR filters design examples are presented here to show complexity comparison of various works in literature especially BCS 3bits and BCS 4-bits. The filter coefficients are generated using the Park-McClellan algorithm.

Here at first step keeping the filter taps constant and by only varying the wordlength the number adder reduction is analysed. Keeping filter length to 200 and at various wordlength the number of adders for BCS 3-bits and BCS 4-bits are analysed and results are given in table, for all analysis presented here the stopband frequencies and passband frequencies are kept to 0.5pi and 0.3pi respectively and all analysis done foe low pass filter.

Results when keeping filter tap to 200 and varying wordlength,

Word length	BCSE 4'bits (number of adders)	BCSE 3'bits (number of adders)
12	25	37
14	38	50
16	37	61
18	50	61
20	49	74
24	61	85

Table 4: Keeping filter length to 200 and varying wordlength

keeping filter length to 200 and varying wordlength,

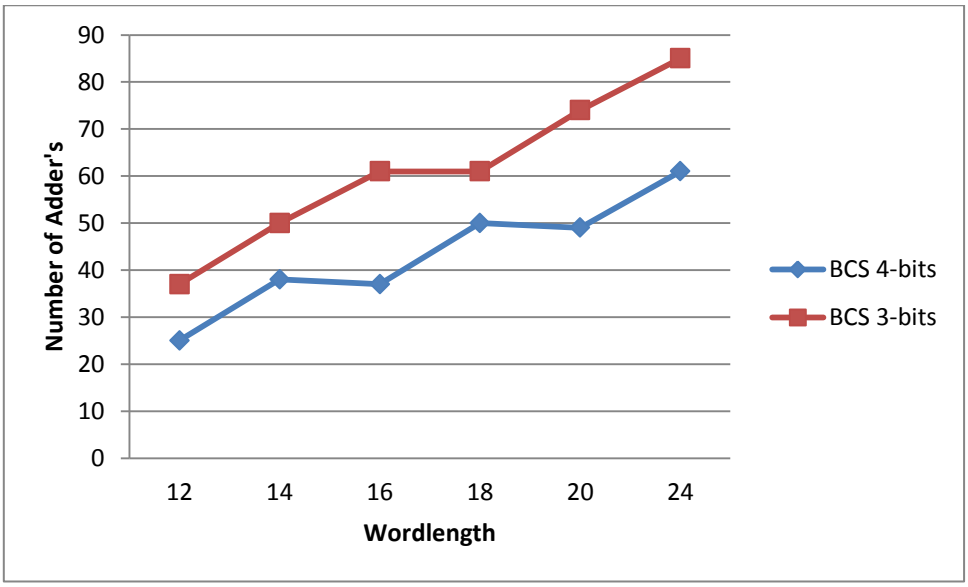

Figure 7: Graph showing number of adder's versus wordlength for BCS3-bit, 4-bit, when filter length kept constant (200 taps).

Similar kind of trend is found when keeping filter length to 512 and varying wordlength.

This reduction in number of adders can also be found when we keep wordlength constant while changing the number of filter taps, by careful analysis the results are mentioned in table

Filter Length (Number of taps)	BCS 3-bits (number of adders)	BCS 4-bits (number of adders)
20	17	15
60	21	19
90	34	23
120	31	25
200	34	21

Table 5: Adder reduction keeping wordlength constant of 12 bits

Adder reduction keeping wordlength constant of 12 bits

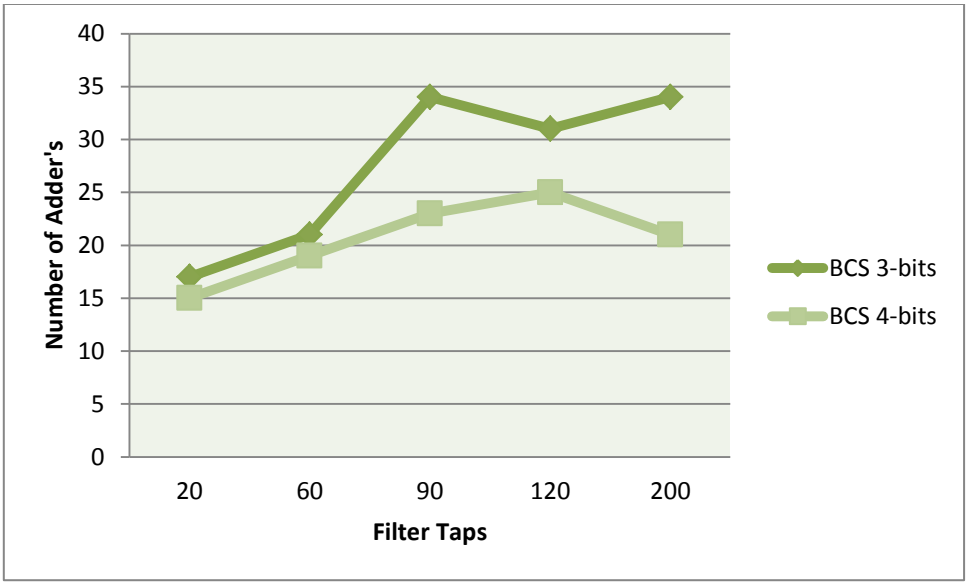

Figure 8: Graph showing number of adder variation for BCS 3-bits and 4-bits keeping wordlength constant (12 bits)

By increasing wordlength to 16-bits or 24 bits the results are changed a bit, these results are as given in the tables accordingly

Filter Length (number of taps)	BCS 3-bits (number of adders)	BCS 4-bits (number of adders)
20	18	16
60	33	23
90	44	34
120	48	37
200	61	48

Table 6: Adder reduction keeping wordlength constant of 16 bits

Adder reduction keeping wordlength constant of 16 bits

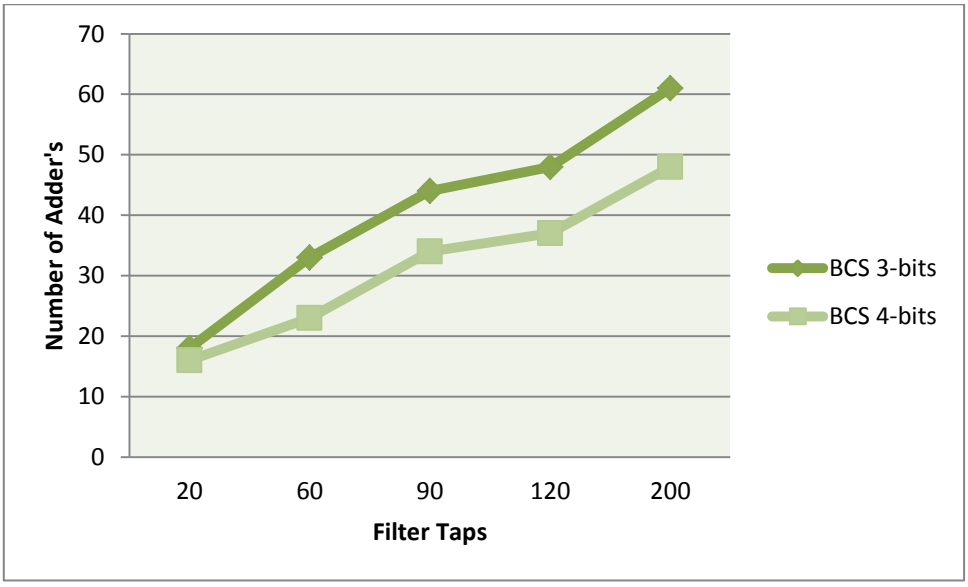

Figure 9: Number of adder's variations with filter taps keeping wordlength constant (16 bits).

Filter Length (number of taps)	BCS 3-bits (number of adders)	BCS 4-bits (number of adders)
20	36	33
60	84	78
90	99	87
120	118	109
200	84	61

Table 7: Adder reduction keeping wordlength constant of 24 bits

Adder reduction keeping wordlength constant of 24 bits

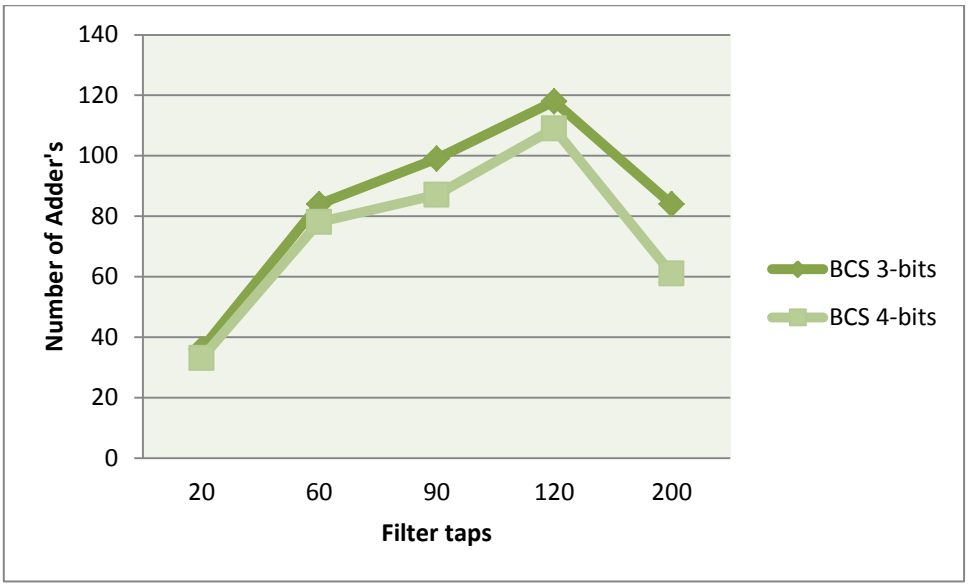

Figure 10: Number of adder's variations with filter taps keeping wordlength constant (24 bits).

You can see dramatic change in number of adders, it is due the reason when number of non-zero bits are available in abundance then any kind of groups can be made at this point the techniques uses large number of non-zero bit group always reduces the number of adders, hence BCS 4-bit is showing better results. When we analyse the filter with these specifications the important thing to note down is filter adder step increases in wordlength of 24 bits the filter adder steps is 4 adder steps for both BCS 3-bits and BCS 4-bits. So it is clear from the analysis that number of adder reduction is trade-off between the logical depth and reduced number of adders for both techniques.

The percentage reduction in number of adders for BCS 3-bits and BCS 4-bits is shown in the graph below

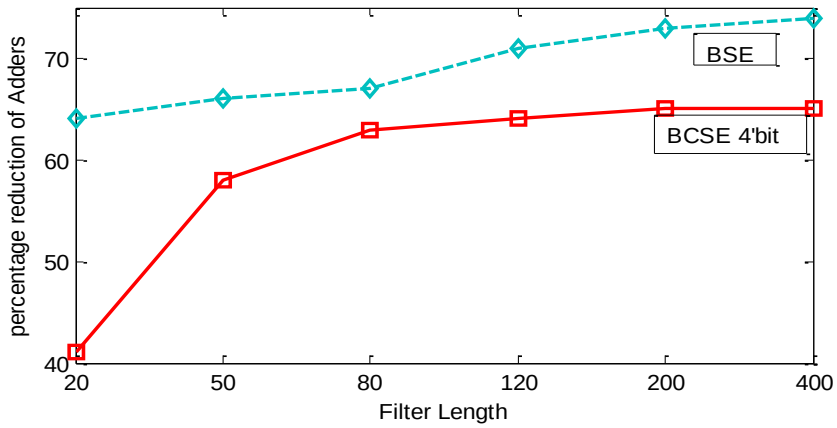

Figure 11: %age number of adder reduction for BCS 3-bits and BCS 4-bits

,

While different taps filter were synthesised and their area power and delays are measured the results are shown in the following tables

For area table (1)

Numbers of taps	BCS 3-bits (area in micro-meter sqr)	BCS 4-bits (area in micro-meter sqr)
30	100421	87729.89
60	121781.41	120850.70
90	175075.75	170005.50
120	248678.35	227657.75

Table 8: Area comparison of BCS 3-bit and 4-bit

Area comparison of BCS 3-bit and 4-bit,

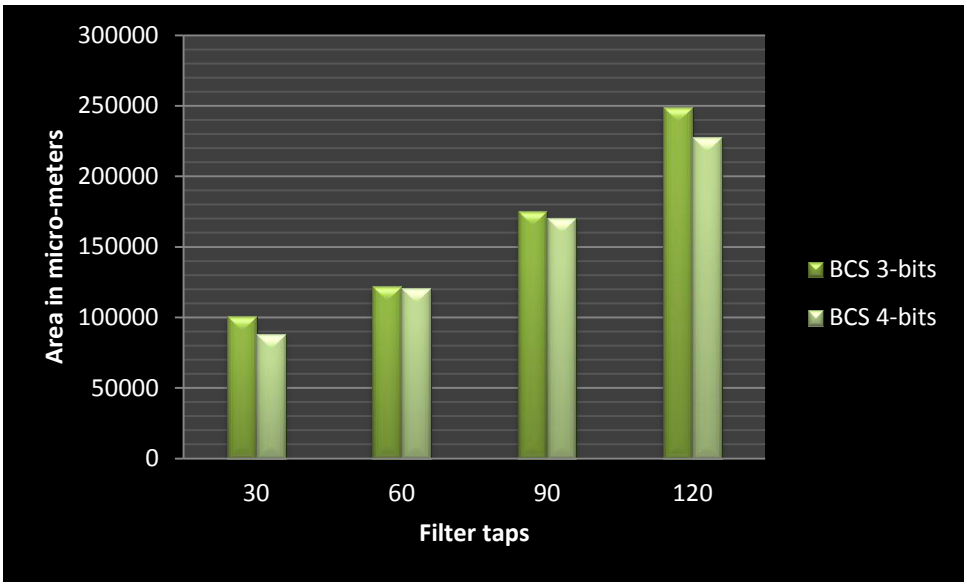

Figure 12: Graph showing Area comparison of BCS 3-bits and BCS 4-bits.

For power table (2)

Number of filter taps	BCS 3-bits (Power in mW)	BCS 4-bits (Power in mW)
30	4.4051	4.0912
60	6.9975	6.8546
90	9.6987	9.5012
120	11.9873	11.2345

Table 9: Power comparison analysis of BCS 3-bits and BCS 4-bits

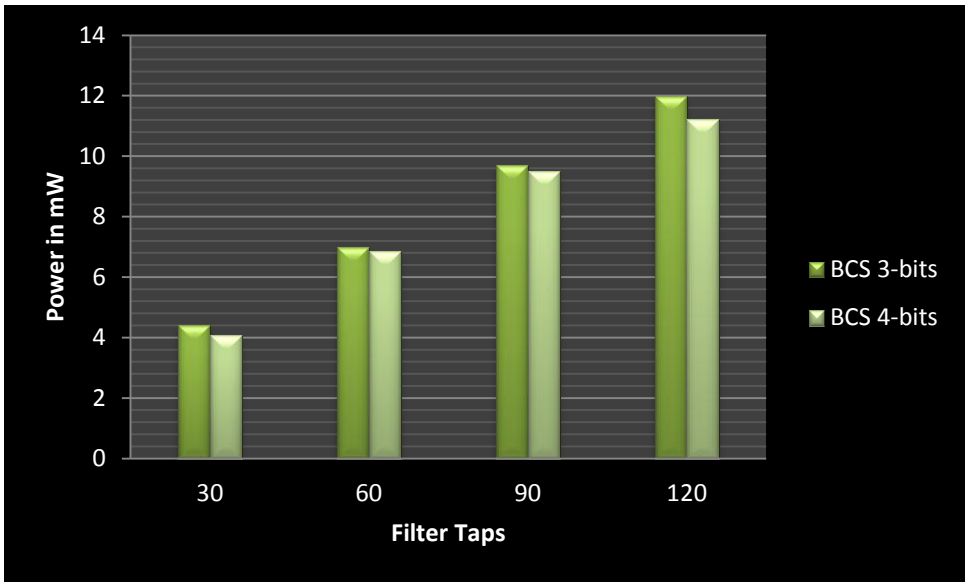

Figure 13: Graph analysing the power consumption.

For delay table (3)

Numbers of filter taps	BCS 3-bits (Delay in ns)	BCS 4-bits (Delay in ns)
30	6.57	6.55
60	6.25	6.21
90	6.20	6.18
120	6.815	6.805

Table 10: Delay comparison of BCS 3-bits and BCS 4-bits

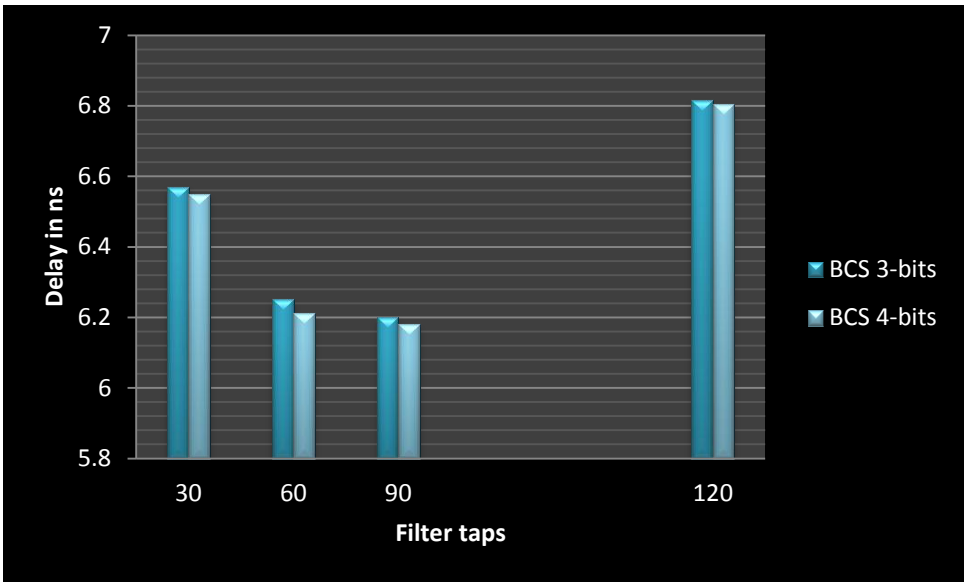

Figure 14: Delay analysis

By looking at all above analysis it is quite clear that BCS 4-bits is much better technique as compared to that of BCS 3-bits, not only it gives much better coverage but also it provide batter area power and delay.

3.1.3 Common Subexpression Elimination- A High –Level Synthesis Transformation:

The adder reduction achieved by using binary common subexpression elimination can be better justified from the perspective of high-level synthesis. In this high –level synthesis, the primary goal of transformation used to be optimized the dedicated architecture to get better trade-off between power, area and speed of operation. Common subexpression is used as powerful tool to reduce power consumption and area. The high-level synthesis uses two non-zero common bits as shown in the figure below, the operand a, b, c and d represents input signal of filter and its shifted versions. The sum e and f are the common sub expressions that are shared in minimize number of adders while s1, s2, s3 and s4 represent number of right shifts. Note that four adders are required to obtain the final expression g and h. the binary common subexpression 3-bit or 4-bit also follows the same pattern. While as shown in figure 3-bit uses the same technique while for 4-bit technique a super expression “i” is there this is basic cause of reduction of number of adders. Three adders are needed for 3-bit and 4-bit

binary common subexpression elimination technique. For higher order filters the number of super expression is more which allows the more reduction in adders hence adder reduction is found significant for higher order filters.

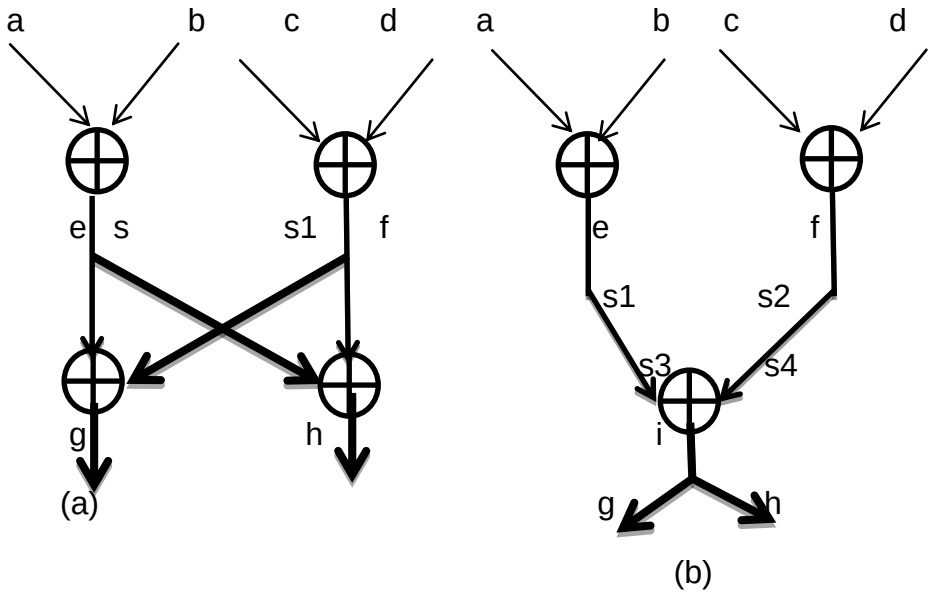

Figure 15: Subexpression sharing as a high-level synthesis transformation.

(a) Conventional 2-bit common subexpression elimination
(b) Binary 4-bit common subexpression elimination

Chapter 4

4 Reconfigurablity of FIR filters:

Several implementation approaches for reconfigurable FIR filters have been proposed in literature. These designs include either a fully programmable multiply-Accumulate (MAC) based filter process or dedicated architecture where coefficients are stored in registers. The architecture of a filter processor consists of the data path with a single MAC unit, data and programme memories, and a control unit. The datapath include 16-bit adder, a multiplier and 32-bit accumulator. The performance of the processor is mainly restricted by the delay if this datapath, more specifically that of multiplier. The main disadvantage of filter processor is that the area and the power requirements are significantly large.

The programmable digital FIR filter for high performance and low power application is proposed in literature. The architecture is based on a computation sharing multiplier which specifically targets computation re-use in vector-scalar products and can be effectively used in the low complexity programmable FIR filter design. Efficient circuit level techniques, namely a new carry select adder and conditional capture flip flop, are also used to future improve power and performance. [18]

A CSD based digital reconfigurable FIR filter architecture was proposed. This architecture is independent of the number of taps because the number of taps and nonzero digits in each tap are arbitrarily assigned. The intention of the author was to reduce the wavelength of the coefficients and thus the filter complexity without affecting the filter performance. But the architecture proposed requires huge hardware resources and this makes the method infeasible. [19]

A high speed programmable CSD-based FIR filter was proposed. The filter architecture consists of a programmable CSD-based booth encoding scheme and partial product Wallace adder tree. The final adder was carry look ahead adder. The method resulted in high speed but at a cost of high power consumption. [20]

Also a time multiplexed approach was proposed. These methods consider the coefficients as constants and use the graph depth algorithm for reducing

redundancy. But this resulted in low speed operation; also area of the architecture is linearly increased as filter length. [21]

All the architecture proposed in the literature are appropriate only for low-order filters and hence not suitable for channel filtering, which normally require higher order filter. From review study, it was found that, integration of low power, low complexity and re-configurability into hardware architecture for FIR filters and filter banks have been hardly addressed in the literature.

4.1 Process Variation Analysis In full-adder Designs:

Last step in this project was to check for process variation in full adders design and find the way to reduce power consumption, as full adder design also effects the power consumption of filter in many ways. In this part literature review is presented and few basic full adder design were taken and their comparative analysis is presented with brief introduction of these adders design comes first.

4.1.1 Full Adders – Hierarchy Design:

Binary addition is a fundamental operation in digital arithmetic circuits. Full Adder is one of the most important parts of each processor. It is used in all arithmetic operations such as division, multiplication, subtraction and addition, in floating-point unit (FPU) and in the Arithmetic Logic Unit (ALU). 1-bit full adder cell is the building block of an arithmetic unit of a system. Hence increasing the performance of a full adder cell is very effective in increasing the performance of the whole system.

Various hardware algorithms are used for two-operand adders and multi-operand adders. These hardware algorithms vary in performance, delay, power and leakage at different processes.

4.1.2 Ripple Carry Adder (RCA):

Ripple Carry Adder is one of the most straightforward implementation of a n-bit adder. It consists of cascading or 'rippling' of Full Adders in which the carry-out of the ith FA is connected to the carry-in of the (i+1)th FA. 28T transistor configuration is used to design the basic full adder cell. Figure 4.1 shows the inter-connection of 4-bit ripple carry adder that employs four full adders. The main drawback of a Ripple Carry Adder (RCA) is the delay, as it increases linearly with bit length. It is not very efficient when very large bit numbers are used. The carry- propagation chain determines the latency of the whole circuit of a RCA. Latency of a n n-bit adder can be derived using the following expression:

Delay (n-bit RCA) =Delay FA (a0, b0 -> C1) + (n-2)*Delay FA (Ci -> C (i+1))

+ Delay FA (Ci -> S (n-1)) – Equation 3

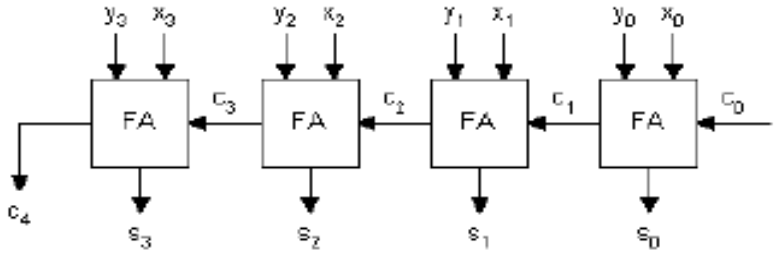

Figure 16: Ripple carry Adder block diagram

4.1.3 Carry Look Ahead Adder (CLA):

CLA is an adder designed to improve the delay of RCA. However, the trade-off is more complex hardware and increased power consumption. CLA calculates the carry signals in advance, based on the input signals. The idea is when carries are generated in parallel, waiting time can be avoided until the correct carry propagates from the stage (FA) of the adder where it has been generated.

Boolean Equations:

Pi = Ai (XOR) Bi -> Carry propagate

Gi = Ai (AND) Bi -> Carry generate

Si = Pi (XOR) Ci -> Sum

C(i+1)= Gi + Pi(AND)Ci -> Carry out

Signals P and G only depend on the input bits.

Applying these equations for a 4-bit CLA adder cell:

C1 = G0 + P0C0

C2 = G1 + P1C1 = G1 + P1G0 + P1P0C0

C3 = G2 + P2C2 = G2 + P2G1 + P2P1G0 + P2P1P0C0

C4 = G3 + P3C3 = G3 + P3G2 + P3P2G1 + P3P2P1G0 + P3P2P1P0C0

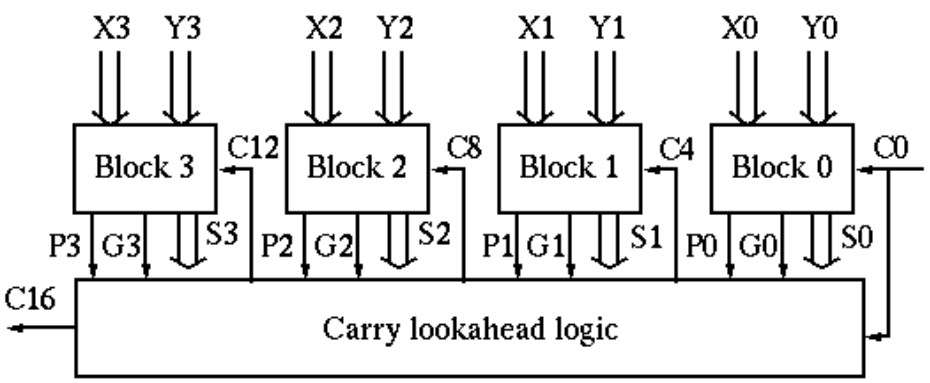

Figure 17: Carry Look Ahead Adders

Figure 4.2 shows the inter-connection of 4-bit Carry Look Ahead adder, which employs four CLA adder cells.

4.1.4 Carry Skip Adder (CSA):

A carry-skip adder reduces the carry-propagation time by skipping over groups of consecutive adder stages. It is usually comparable to CLA in terms of speed and it only requires less chip area and consumes less power. Each bit in a block is compared, and if all bits are unequal, then assuming no carry in that block, the

carry is skipped from previous block to the next one. All the carry-propagate signals in a block is given to an AND1 gate. The output of AND1 gate is fed to 2-input AND2 to which Cout of previous block is given. The output of AND2 is then fed to a 2-input OR gate of which the 2nd input is the Cout of current block. Fig 4.3 illustrates the operation of a Carry Skip adder.

Boolean Equations:

Pi = Ai (XOR) Bi -> Carry propagate

Si = Pi (XOR) Ci -> Sum

C(i+1)= Ai (AND) Bi + Pi(AND)Ci -> Carry out

P-signals depend only on input bits.

If Ai = Bi then Pi = 0, making the carry out, Ci+1, depend only on Ai and Bi .

Where, Ci+1= Ai Bi.

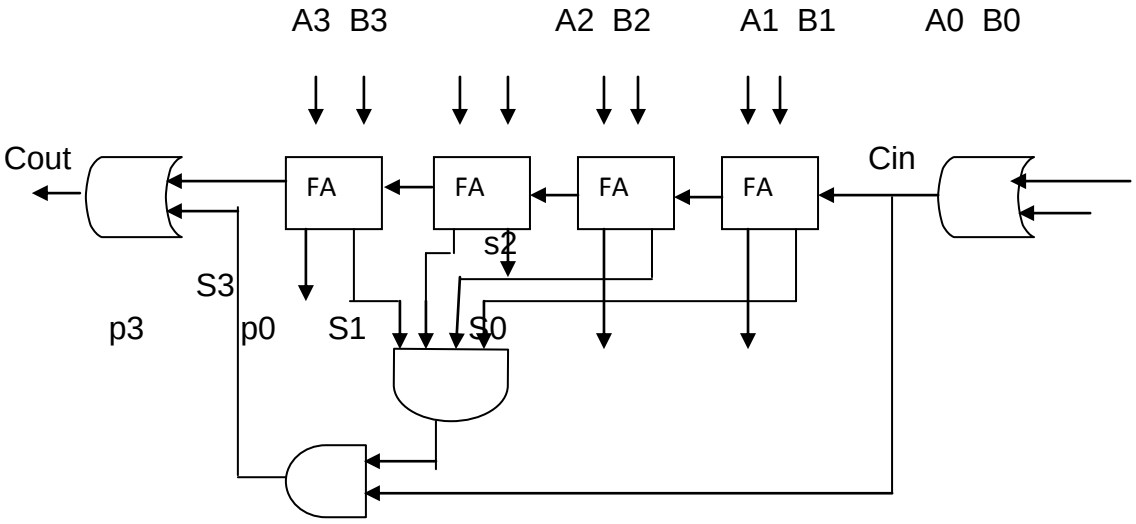

Figure 18: Illustrates Carry Skip Adder Operation

4.1.5 Kogge Stone Adder:

Kogge-Stone adder (KSA) is a parallel prefix form of carry look ahead adder (CLA). It is considered as one of the fastest adder designs. The structure has minimum logic depth, with minimum fan-out at each stage which increases performance, resulting in a fast adder. The main drawback is that it takes a large area. Too much wiring is also a concern for this structure as bit length increases.

Each stage produces a “propagate” and “generate” bit. In stage 0, propagate and generate bits are generated based on equations below:

Pi = Ai (XOR) Bi -> Carry propagate

Gi = Ai (AND) Bi -> Carry generate

Propagate and generate bits of further stages is generated using the equations below.

Pi(n) = Pi(n-1) (AND) Pprev (n-1)

Gi(n) =(Pi(n-1) (AND) Gprev (n-1)) (OR) Gi (n-1)

Where, n = stage number, i = bit numberprev = (i – 1) for stage 1 and prev = (i – 2) for stage 2, prev = (i – 4) for stage 3, and so on. Number of stages required can be determined from the ‘prev’ value. If the data is 4-bit ‘Pi’ and ‘Gi’ only need to be generated till stage 2 and for 16-bit only need to generated till stage 4, where prev = (i-8) and so forth.Once stage generation is over, sum and carry is calculated where,

Ci = Gi

Si = Pi (XOR) Ci

A=1100 B=1011 Sum=10111

A=1100 B=1011 Sum=10111

A3 B3 A2 B2 A1 B1 A0 B0

1 1 1 0 0 1 0 1

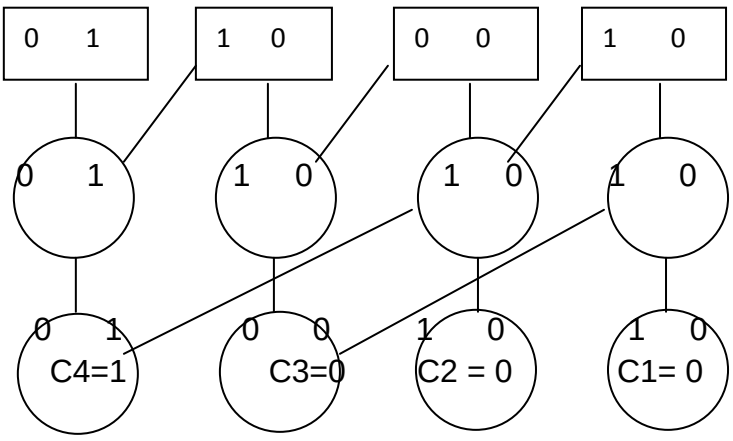

Figure 19: Illustrates the operation of a 4-bit Kogge Stone Adder

Number of carry bits generated by the tree is called sparsity. When each and every carry bit is generated it is referred to as Sparsity-1 and generating every other carry bit is Sparsity-2 and so on. These reduced no of carries are then given as ‘carry inputs’ to a reduced RCA which will generate the final sum bits. As sparsity increases, computation required is reduced and hence reduces wiring congestion, power and area.

4.1.6 Analysis:

When all these proposed full adders design were examined under keeping biasing voltage constant and by varying different other parameters like supply voltage etc the results were found those results are summarized here

By keeping biasing voltage constant and varying supply voltage for all four design the power required is varying and can be found that ripple carry adder is more efficient as it require less power as compared to others their comparison is given I the graph, as long as the complexity increase the power requirement increases in full adder designs, so keeping things simple is more power efficient way as compared to complex.

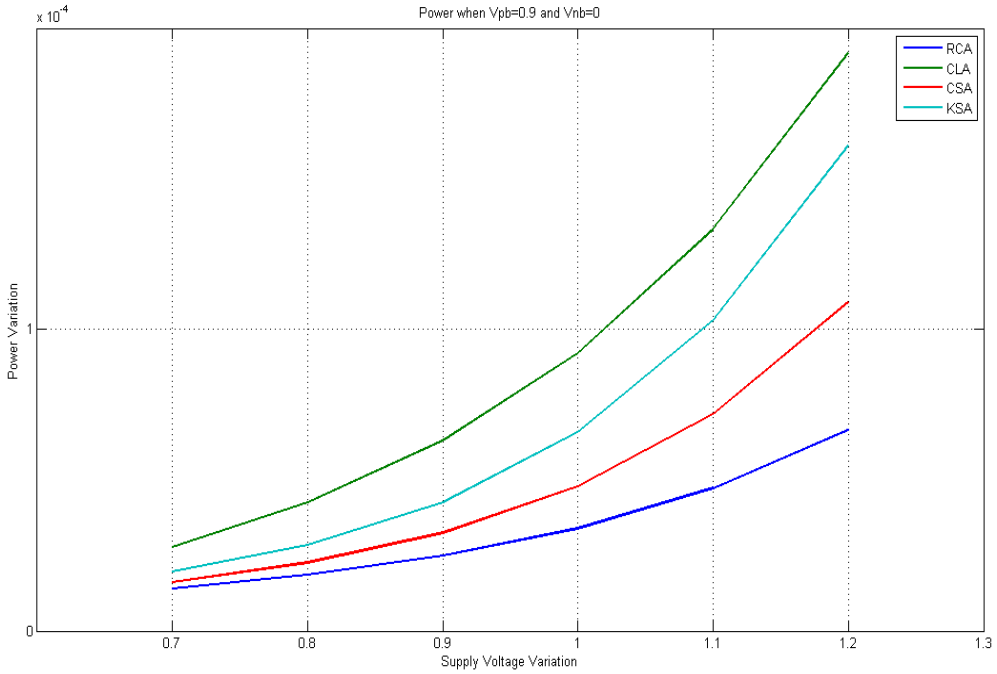

Figure 20: Graphical analysis to check power variations for full adder designs.

Similarly we can check the leakage voltage in the proposed full adder design as when this leakage voltage increases it also increases the power requirement for the adder hence increase power consumption of the design. As by analysis of leakage voltage variation it is found that leakage voltage is high in carry look ahead adder as compared to other adder design. Here the important point is the leakage voltage variations are much higher in ripple carry adder as compared to carry skip adder but due complexity of carry skip adder more power required for its computation.

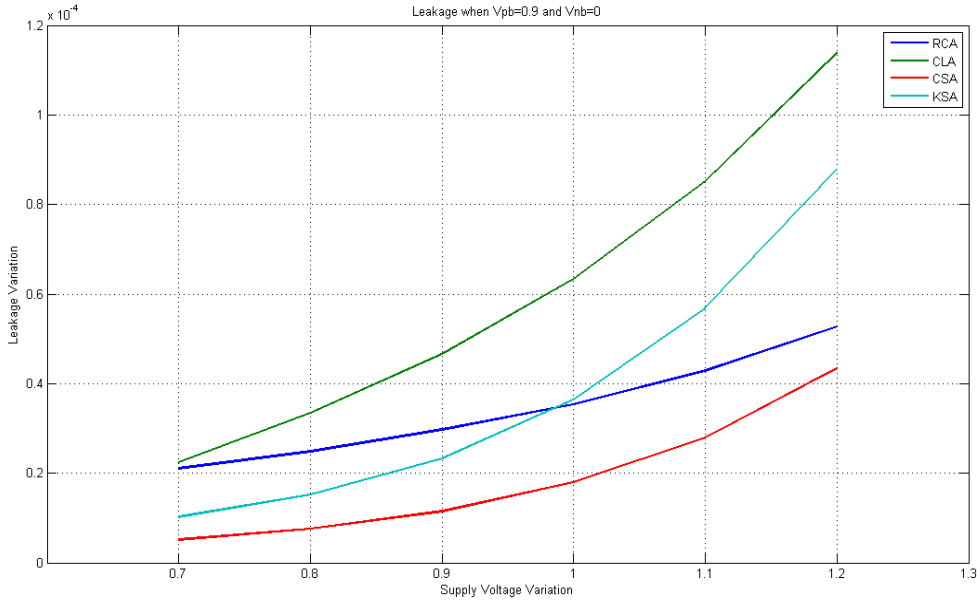

Figure 21: Graphical analysis to show leakage voltage variation in full adder designs.

Delay is another important factor while discussing the full adder designs proposed in literature, when analysing the design it is found that delay efficient design is Kogge stone adder, while delay is significant in other full adder designs, delay analysis for all these designs are given as

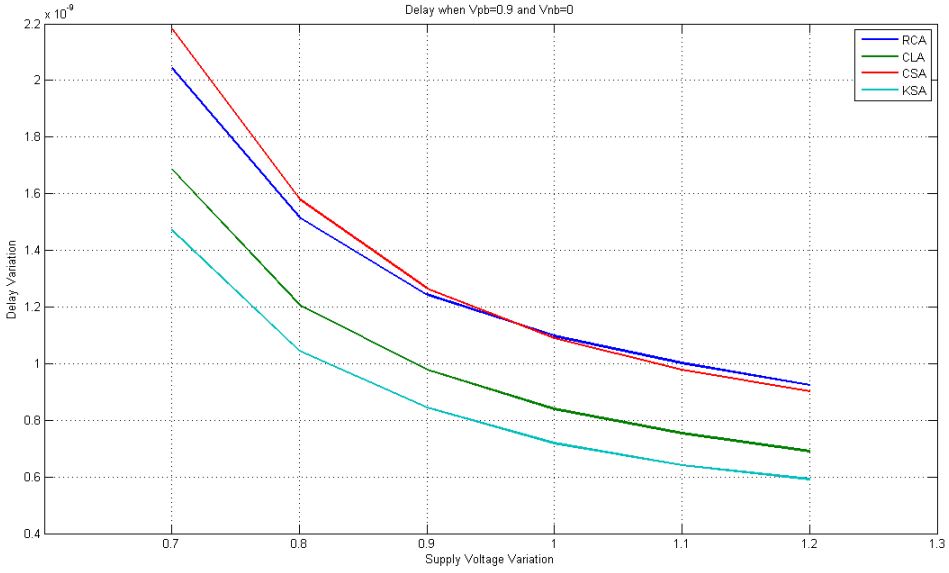

Figure 22: Delay variation analysis

Chapter 5

5 Project Analyses:

This thesis address the research problem of incorporating low power, high speed and reconfigurablity into the FIR filter architecture for wireless receiver. Finite impulse response (FIR) filters are commonly employed in wireless receivers. Generally high order FIR filters are required to meet the stringent adjacent channel attenuation specification of wireless communication standards. Coefficients multiplication of such high order filters consume much power and chip area. The purpose of this project is to propose efficient method of multiplications in FIR filters. Apart from low complexity, reconfigurablity of these FIR filters. In this thesis, couple of methods are mentioned to realize reconfigurable low power FIR filter. In order to make FIR filter power efficient few things are taken into account, the number system in which the filter coefficients are represented and architecture of adders used to implement that FIR filter. During selection of number system for representation of filter coefficients few thing can be taken into account, one how should one group the information means code that and how someone retrieve that. During this work I tried to go along this strategy but things were not as smooth as they look. In this thesis for both parts research has been done and few results have been shown. In order to check effects of coefficient representation, I have extended the idea given by K.P Gopi Smitha for better representation of filter coefficients and algorithm to implement coefficients multiplication efficiently. There are few other idea that can be explored and can be taken into account, but due to several reason I could not do that. Especially the idea of instead of going for only horizontal common subexpression elimination I should have gone for vertical analysis, which I tried but things gone weird and I could not continue with that. Basic problem with that was a delay issue which increases as we try to implement vertical common subexpression elimination. Comparison of couple of methods proposed in literature, about power efficient implementation of filter coefficient's multiplication, to learn the way things progress in this field. This part needs much of intention that I did but still could not find the best answer at all. After that much of surfing I could have some better solution for the problem that I do not have. Last but not the least full adder architectures have been checked and their comparison has been given keeping standard setting into mind. These full adders architecture have been compared in order to check process variation with respect to applied voltage. For this purpose best of process variation analysis can be done in much different way, while some new idea can be given while basic changes in the

architecture of full adders that would have helped the cause. I tried few but they did not worked for me as I expected them to do.

To put into nut shell, whole experience was quite interesting and I learnt allot from that. I know there is always ways to improve things or at least make things work in proper order still whichever is done is also little better than nothing. This thesis could have been done in much better way by giving some new idea in this field or at least in the field of full adder's design. This thesis is mostly emphasis on comparison of couple of implemented methods for low power and low complexity FIR filter that ultimately used for wireless receivers.

Chapter 6

Conclusions:

Digital filter is the basic building block of for any digital filter bank, used in wireless receivers. Hence optimization of digital filter is an essential requirement. FIR filters are widely employed in digital filtering as they have absolute stability and linear phase property which are essential for wireless receivers. The most computationally intensive operation in an FIR filter is coefficient multiplication. Since high order filter required for wireless receivers, with several hundreds of coefficients. To make them power efficient and fast enough is challenging task. Here in this thesis some of the techniques have been discussed like canonical signed digits that are helpful in reduction of number of adder's as it reduces number of non-zero bits. Then most of the thesis discuss binary common subexpression elimination techniques both 3-bits and 4-bits. These two also helpful in reduction of number of adder's not by reducing number of non-zero bits but by combining non-zero bits and hence reducing the number of adder's. Analysis of these two is presented in this thesis, by varying different variables. To use the FIR filters for wireless receivers require sharp transition bands because of adjacent channel attenuation. In this thesis reconfigurabilty of FIR filters is also discussed a bit. Different reconfigurable methods proposed in the literature are discussed in this thesis.

To design power efficient and reconfigurable FIR filter to be used in wireless receivers is a challenging task. Different methods have been discussed here the purpose of all of them is to implement low power FIR filter. For multiplication of coefficients full adders are used. In this thesis some basic building blocks of full adder design is also discussed and they compared at most basic level to check which one is better in provided conditions. Full adder design discussed here are from literature reviews.

The main focus of this project is to analyse process variation effects in low power and speedy FIR filter response. Efficient implementation of process variation tolerant adder which is fundamental block in FIRfilter is also explored in this thesis. This project is mostly research based and involves both circuit level simulation and higher level design aspects. Also some basic circuits have been checked for variation of voltage and also tried to implement full adder with simple

logic and variation of voltage has been checked, all this circuit analysis is done in HSPICE.

Future extensions:

In this dissertation, the challenges in designing low power reconfigurable high speed and low complexity FIR filter for wireless receivers have been investigated. Different methods have been discussed however many optimization possibilities to pursue further research in this area in future have been identified. Some of them are mentioned here,

1. One-shot design of reconfigurable Channel filters using FRM technique.
2. Channelizer based on Fast Filter bank
3. Optimization of RNS-FIR filter for multiple moduli

All these mentioned techniques are means of reducing complexity and improve power requirements for FIR filter used in them. Basic need of the time is to dig out some ways to improve the coefficient multiplication in a more efficient way, for this purpose number system in which filter coefficients are represented are also important so someone can improve efficiency by just improving number system for filter coefficients. Full adders are basic blocks for coefficient multiplication in FIR filters hence their design are also discussed and ways to improve them is also one way of improving in this field.

For full adders designs process variation as a major concern in the current and future generations of CMOS circuits manufacturing has already imposed a great deal of effort in design process. Process variations are generally divided into two main categories one of them is the random variations and the second is systematic variations. Therefore, physical designers of integrated circuits have to take into account the effects of random and systematic process variations in the design process. The effects of Front-End of Line systematic variations being critical dimension of transistors scale down have been proved to be a great contributor of performance variability of circuits. As a result, the performance of integrated circuit becomes highly sensitive to these variations. So for better result and for better reduction in power requirements process variations are checked.

In emerging technologies like FinFET and CNFET that operates in the moderate inversion region for energy efficiency, robustness and higher performance. The performance of the adder is improved by the optimum selection of important

process parameters like oxide and fin thickness in FinFET and number of carbon nanotubes, chirality vector and pitch in CNFET. The optimized CNFET-based full adder (OP-CNFET) has higher speed, lower power-delay product and lower power dissipation as compared to the MOSFET and FinFET full adder cells. Analysis of such full adder designs to implement power efficient coefficient multiplication in FIR filters can be done and more power efficient FIR filter can be designed to be used in the wireless receivers.

References:

1. Ricardo A. Losad “Practical FIR Filter Design in MATLABR” January 12, 2004
2. Tang Zhangwen, Zhang Zhanpeng, Zhang Jie, Min Hao, “ASIC& Syst. State Key Lab., Fudan Univ., Shanghai “A high-speed, programmable, CSD coefficient FIR filter”
3. Demirsoy, S.S. Kale, I. Dempster, A.G. Appl. DSP & VLSI Res. Group, Univ. of Westminster, London, UK “Efficient implementation of digital filters using novel reconfigurable multiplier blocks” November 2004.
4. T. Saramaki. “Finite impulse response filter design,”Chapter 4 in Handbook for Digital Signal Processing,S. K. Mitra and J. F. Kaiser eds. John Wiley and Sons, New York, 1993.
5. T. Saramaki, Y. Nuevo, and S. K. Mitra, “Design of computationally efficient interpolated FIR filters,” IEEE Trans. on Circuits and Systems, vol. 35, N0.1,pp. 70-88, January 1988.
6. I. W. Selesnick and C. S. Burrus, “Exchange algorithms that complement the Parks-McClellan algorithm for linear-phase FIR filter design,” IEEE Trans. on Circuits and Systems II, 44(2):137-142,February 1997.
7. Bull, D.R.; Horrocks, D.H. Sch. of Electr. Electron. & Syst. Eng., Wales Univ., Cardiff “Primitive operator digital filters” June 1991.
8. T. Hentschel and G. Fettweis, "Software radio receivers," in CDMA Techniques for Third Generation Mobile Systems. Dordrecht, the Netherlands: Kluwer Academic, 1999, pp. 257-283.
9. A. P. Vinod and E. Lai, "Low power and high-speed implementation of FIR filters for software defined radio receivers," IEEE Trans. Wireless Commun., vol. 5, no. 7, pp. 1669-1675, Jul. 2006.
10. T. Zhangwen, J. Zhang, and H. Min, "A high-speed, programmable, CSD coefficient FIR filter," IEEE Trans. Consumer Electron., vol. 48, no. 4, pp. 834-837, Nov. 2002.
11. Sery G., et al., Life is CMOS: why chase the life after? DAC 2002, 78-83.
12. Karnik, T., et al., “Sub-90nm Technologies—Challenges and Opportunities for CAD”, ICCAD 2002, pp. 203-206.
13. A. Srivastava, D. Sylvester, and D. Blaauw, “Statistical optimization of leakage power considering process variations using dual-Vth and sizing,” in Proc. Des. Autom. Conf., 2004, pp. 773–778
14. S. Borkar, T. Karnik, and V. De, “Design and reliability challenges in nanometre technologies,” in Proc. Des. Autom. Conf., 2004, p. 75.
15. J. M. Rabaey, Digital Integrated Circuits. Englewood Cliffs, NJ: Prentice-Hall, 1996.

16. http://bwrc.eecs.berkeley.edu/classes/icdesign/ee241_s00/LECTURES/lecture16-adders.pdf
17. http://writphotec.com/mano4/Supplements/Carrylookahead_supp4.pdf
18. J.park et al, “computation sharing programmable FIR filters for low power and high performance” IEEE Journel on solid state circuits, vol. 39, no. 2, pp. 348-357, Feb.2004.
19. K. H. Chen and T. D. Chiueh, “A low-power digit-based reconfigurable FIR filter,” IEEE Trans on Circuits and systmes-II, vol. 53, no. 8, pp. 617-621, Aug.2006.
20. T. Zhangwen and J. Zhang and H. Min, “A high-speed, programmable, CSD coefficient FIR filter,” IEEE Trans. On Consumer Electronics, vol.48, pp. 461-464, Nov.2004.
21. P. Tummeltshammer, J. C. Hoe and M. Puschel, “time-multiplexed multiple-constant multiplication,” IEEE Transaction on Computer-Aided Design of integerated Circuits and systems, vol.26, no.9, pp.869-874, Jun.1996.

Appendix:

```
//////////////////////////
Matlab code for number of adder's calculation BCS 4-bits
////////////////////////////////////
format long;
clear all;
fprintf('Program for Binary Common Subexpression Elimination (BCS 4-bits) for FIR Filter Co-
efficients in Binary form\n');
n=input('Input the number of filter taps \n');
f1=input('Input the pass band frequency (normalized in terms of sampling frequency) \n');
f2=input('Input the stop band frequency (normalized in terms of sampling frequency) \n');
b=input('Input the wordlength\n');
   f=[0 f1 f2 1];
   a=[1 1 0 0];
   h=firpm(n,f,a);%Getting Coefficients using remez exchange algorithm and storing in matrix h
h1=h;
%end
n1=ceil(n/2);%Considering only Symmetric Set Coefficients
for i=1:n1%Conversion to Binary set
   h1(i)=abs(h1(i));
   for j=1:b
      h1(i)=h1(i)*2;
      if (h1(i)>=1)
         c(i,j)=1;
         h1(i)=abs(1-h1(i));
      else
         c(i,j)=0;
      end
   end
end
e=c;%Binary values of coefficients in matrices c and e
%Elimiation of binary subexpressions[1 1 1 1],[1 1 0 1],[1 0 1 1], [1 1 1], [1 0 1], [1 1] and [1 0 0 1]
and 4=[1 1 1],3=[1 0 1],2=[1 1],5=[1 0 0 1]
f2=b-2;
f3=b-1;
f4=b-3;
%f5=b-4;
for i=1:n1
    k=1;
    for j=1:b %Elimination of [1 1 1 1]
```

```
    if (k<=f4)
       if (c(i,j)==1 && c(i,j+1)==1 && c(i,j+2)==1 && c(i,j+3)==1)
          c(i,j)=8;
          c(i,j+1)=0;
          c(i,j+2)=0;
          c(i,j+3)=0;
       end
       k=k+1;
    end
  end
   k=1;
  for j=1:b %Elimination of [1 0 1 1]
    if (k<=f4)
       if (c(i,j)==1 && c(i,j+1)==1 && c(i,j+3)==1)
          c(i,j)=6;
          c(i,j+2)=0;
          c(i,j+3)=1;
       end
       k=k+1;
    end
  end
    k=1;
  for j=1:b %Elimination of [1 1 0 1]
    if (k<=f4)
       if (c(i,j)==1 && c(i,j+2)==1 && c(i,j+3)==1)
          c(i,j)=7;
          c(i,j+1)=0;
          c(i,j+3)=1;
       end
       k=k+1;
    end
  end
 k=1;
  for j=1:b %Elimination of [1 1 1]
    if (k<=f2)
       if (c(i,j)==1 && c(i,j+1)==1 && c(i,j+2)==1)
          c(i,j)=4;
          c(i,j+1)=0;
          c(i,j+2)=0;
       end
       k=k+1;
    end
```

```
    end
    k=1;
    for j=1:b %Elimination of [1 1]
      if (k<=f3)
         if (c(i,j)==1 && c(i,j+1)==1)
            c(i,j)=2;
            c(i,j+1)=0;
         end
         k=k+1;
      end
    end
    k=1;
    for j=1:b %Elimination of [1 0 1]
      if (k<=f2)
         if (c(i,j)==1 && c(i,j+2)==1)
            c(i,j)=3;
            c(i,j+2)=0;
         end
         k=k+1;
      end
    end
end
for i=1:n1 %Elimination of [1 0 0 1]
   k=1;
    for j=1:b
       if (k<=f4)
          if (c(i,j)==1 && c(i,j+3)==1)
             c(i,j)=5;
             c(i,j+3)=0;
          end
          k=k+1;
       end
    end
end
%%Elimination of Common Subexpressions/bits shared Among Co-efficients. Also known as
CBCS
%corr contains the number of shared bits
%corr1 contains the actual subfilter
%corr2 contains the shared bits
for i=1:n1
   for j=1:n1
      corr(i,j)=0;
```

```
   end
end
for i=1:n1
   for j=1:n1
      corr1(i,j)=0;
   end
end
for i=1:n1
   for (j=(i+1):n1)
      for k=1:b
            if (c(i,k)==c(j,k)&& c(i,k)~=0)
               corr(i,j)=corr(i,j)+1;
            end
      end
   end
end
for i=1:n1
   noccur1=0;
   MAX=max(corr(i,1:n1));
   for j=(i+1):n1
      if (corr(i,j)==MAX)
         noccur1=noccur1+1;
      end
   end
   if (noccur1>1)
      for j=(i+1):n1
         noccur2=0;
         if (corr(i,j)==MAX && corr(i,j)~=0)
            for j1=i+1:n1
               if (corr(j1,j)>=MAX)
                  noccur2=noccur2+1;
               end
            end
            if (noccur2==0)
               corr1(i,j)=corr(i,j);
               corr(i,1:n1)=0;
               corr(1:n1,j)=0;
               corr(j,1:n1)=0;
            end
         end
      end
   end
```

```
   if (noccur1==1)
      for j=(i+1):n1
         if (corr(i,j)==MAX && corr(i,j)~=0)
            corr1(i,j)=corr(i,j);
            corr(i,1:n1)=0;
            corr(1:n1,j)=0;
            corr(j,1:n1)=0;
         end
      end
   end
end
ncorra=0;
ncorrna=0;
for i=1:n1
   for j=1:n1
      if (corr1(i,j)~=0 && corr1(i,j)~=1)
         ncorra=ncorra+corr1(i,j);
         ncorrna=ncorrna+1;
      end
   end
end
ncorra=ncorra-ncorrna;
for i=1:n1
   for j=1:b
      corr2(i,j)=0;
   end
end
for i=1:n1
   for j=1:n1
      if (corr1(i,j)~=0 && corr1(i,j)~=1)
         for k=1:b
            if (c(i,k)==c(j,k))
               corr2(i,k)=c(i,k);
               corr2(j,k)=c(i,k);
            end
         end
      end
   end
end
for i=1:n1
   for j=1:b
      c(i,j)=c(i,j)-corr2(i,j);
```

```
   end
end
ns=0;
for i=1:n1
   count=0;
   for j=1:b
      if (c(i,j)~=0 && c(i,j)~=1 && c(i,j)~=-1)
         count=count+1;
      end
   end
   if (count>1)
      ns=ns+count-1;
   end
end
num1=0;
for i=1:n1
   count=0;
   for j=1:b
      if (c(i,j)==1 || c(i,j)==-1)
         count=count+1;
      end
   end
   if (count>1)
      num1=num1+count-1;
   end
end
for j1=24:b %Elimination of subexpressions > shift difference between wordlength and input bit
length.In this case Input-bit length=8
   for i=1:n1
      for j=1:b
         if j1+j<=b
            if (c(i,j)==1 && c(i,j1+j)==1)
               c(i,j)=9;
               c(i,j1+j)=0;
            end
         end
      end
   end
end
n1d=0;
for i=1:n1 %No. of adders in Symmetric set including Structural adders
   for j=1:b
```

```
      if (c(i,j)~=0)
         n1d=n1d+1;
      end
   end
end
n2d=0; %No of adders for Mutlipier Block
for i=1:n1
   count=0;
   for j=1:b
      if (c(i,j)~=0)
         count=count+1;
      end
   end
   if (count>1)
      n2d=n2d+count-1;
   end
end
bse2=0;
bse3=0;
bse4=0;
bse5=0;
bse6=0;
bse7=0;
bse8=0;
for i=1:n1
   for j=1:b
      if (c(i,j)==2)
         bse2=1;
      end
      if (c(i,j)==3)
         bse3=1;
      end
      if (c(i,j)==4)
         bse4=1;
      end
      if (c(i,j)==5)
         bse5=1;
      end
      if (c(i,j)==6)
        bse6=1;
      end
      if (c(i,j)==7)
```

```
        bse7=1;
      end
      if (c(i,j)==8)
        bse8=1;
      end
    end
end
fprintf('No of adders after
BSE=%d\n',n2d+ncorra+bse2+bse3+bse4+bse5+bse6+bse7+bse8);%bse stands for the Basic Sub-
Expression & ncorra stands for Sharing among Co-efficients.
asmax=0;%For Finding Adder Step
for i=1:n1
   as=0;
   nb1=0;
   nb2=0;
   ncorrb=0;
   nb=0;
   nbmax=0;
   for j=1:b
      if (c(i,j)~=0)
         nb1=nb1+1;
      end
      if (c(i,j)==3 || c(i,j)==2 || c(i,j)==5 || c(i,j)==6 || c(i,j)==4 || c(i,j)==7)
         nb2=nb2+1;
      end
      if (c(i,j)==8)
         nb2=nb2+1;
      end
      if (corr2(i,j)~=0)
         ncorrb=ncorrb+1;
      end
   end
   nb=nb2+ncorrb+nb1;
   if (nb>nbmax)
      nbmax=nb;
   end
   if (nb~=0)
      as=ceil(log(nb)/log(2));
   end
   if (as > asmax)
      asmax=as;
   end
```

```
end
for i=1:n1
   M(i)=0;
end
for i=1:n1
   for j=1:b
      if (c(i,j)~=0)
         M(i)=M(i)+1;
      end
   end
end
MMAX=max(M(1:n1));
fprintf('Adder step=%d\n',asmax);
F=0; %Full Adder Calculation
n2=0;
n3=0;
n5=0;
n7=0;
for i=1:n1
   k=1;
   N=0;
   for l=1:24
   S(l)=0;% The span value
   end
   for l=1:24
   pre(l)=0; % Prefix value
   end
   for j=1:b
      if c(i,j)~=0
         N=N+1;
         if c(i,j)==1
            S(k)=b+j;   %b= input word length
            k=k+1;
         end
         if c(i,j)==2 || c(i,j)==6
            S(k)=b+2+j;
            k=k+1;
            n2=n2+1;
         end
         if c(i,j)==3 || c(i,j)==4
            S(k)=b+3+j;
            k=k+1;
```

```
            n3=n3+1;
        end
        if c(i,j)==5 || c(i,j)==7
            S(k)=b+5+j;
            k=k+1;
            n5=n5+1;
        end
         if c(i,j)==8
            S(k)=b+7+j;
            k=k+1;
            n7=n7+1;
        end
    end
  end
  if (N>1)
    if (mod(N,2)~=0)
        pre(N-2)=1;
        j=2;
        for i=1:(N-2)
            if j<N-2
                pre(j)=1;
                j=j+2;
            end
        end

F=F+(S(2)+1)+(pre(1)*(S(3)+1))+(pre(2)*(2*S(4)+3))+(pre(3)*(S(5)+1))+(pre(4)*(S(6)+1))+(pre(5)*
(2*S(7)+3))+(pre(6)*(3*S(8)+6))+(pre(7)*(S(9)+1))+(pre(8)*(S(10)+1))+(pre(9)*(2*S(11)+3))+(pre(
10)*(2*S(12)+3))+(pre(11)*(2*S(13)+3))+(pre(12)*(S(14)+1))+(pre(13)*(3*S(15)+6))+(pre(14)*(4
*S(16)+8))+(pre(15)*(S(17)+1))+(pre(16)*(S(18)+1))+(pre(17)*(2*S(19)+3))+(pre(18)*(2*S(20)+3)
)+(pre(19)*(2*S(21)+3))+(pre(20)*(S(22)+1))+(pre(21)*(3*S(23)+6));
    else
        if (N==2)
            F=F+S(2)+1;
        end
        if (N==4)
            F=F+(S(2)+1)+(2*S(4)+3);
        end
        if (N==6)
            F=F+(S(2)+1)+(2*S(4)+3)+(2*S(6)+3);
        end
        if (N==8)
            F=F+(S(2)+1)+(2*S(4)+3)+(S(6)+1)+(3*S(8)+6);
```

```
        end
        if (N==10)
           F=F+(S(2)+1)+(2*S(4)+3)+(S(6)+1)+(3*S(8)+6)+(2*S(10)+3);
        end
        if (N==12)
           F=F+(S(2)+1)+(2*S(4)+3)+(S(6)+1)+(3*S(8)+6)+(S(10)+1)+(3*S(12)+6);
        end
        if (N==14)
           F=F+(S(2)+1)+(2*S(4)+3)+(S(6)+1)+(3*S(8)+6)+(S(10)+1)+(2*S(12)+3)+(3*S(14)+6);
        end
        if (N==16)

F=F+(S(2)+1)+(2*S(4)+3)+(S(6)+1)+(3*S(8)+6)+(S(10)+1)+(2*S(12)+3)+(S(14)+1)+(4*S(16)+8);
        end
        if (N==18)

F=F+(S(2)+1)+(2*S(4)+3)+(S(6)+1)+(3*S(8)+6)+(S(10)+1)+(2*S(12)+3)+(S(14)+1)+(4*S(16)+8)+(2*S
(18)+3);
        end
        if (N==20)

F=F+(S(2)+1)+(2*S(4)+3)+(S(6)+1)+(3*S(8)+6)+(S(10)+1)+(2*S(12)+3)+(S(14)+1)+(4*S(16)+8)+(S(1
8)+1)+(3*S(20)+6);
        end
        if (N==22)

F=F+(S(2)+1)+(2*S(4)+3)+(S(6)+1)+(3*S(8)+6)+(S(10)+1)+(2*S(12)+3)+(S(14)+1)+(4*S(16)+8)+(S(1
8)+1)+(2*S(20)+3)+(3*S(22)+6);
        end
        if (N==24)

F=F+(S(2)+1)+(2*S(4)+3)+(S(6)+1)+(3*S(8)+6)+(S(10)+1)+(2*S(12)+3)+(S(14)+1)+(4*S(16)+8)+(S(1
8)+1)+(2*S(20)+3)+(S(22)+1)+(4*S(24)+8);
        end
      end
   end
end
if n2>0
   F=F+b+2;
end
if n3>0
   F=F+b+3;
```

```
end
if n5>0
   F=F+b+4;
end
if n7>0
   F=F+b+6;
end
fprintf('Number of Full Adders with out coefficient partioning =%d\n',F);
F=0;
n2=0;
n3=0;
n5=0;
n7=0;
for i=1:n1
   F1(i)=0;
end
for i=1:n1
   k=1;
   N=0;
   for l=1:24
   S(l)=0;% The span value
   end
   for l=1:24
   pre(l)=0; % Prefix value
   end
   y1=0;
   for j=1:b
      if c(i,j)~=0
         N=N+1;
         for y=1:b
            if (c(i,j)~=0 && y1==0)
               y1=j;
            end
         end
         if c(i,j)==1
            S(k)=8+j-y1;   %8= input word length
            pre(k)=j-y1;
            k=k+1;
         end
         if c(i,j)==2 || c(i,j)==-2 || c(i,j)==3 || c(i,j)==-3
            S(k)=8+2+j-y1;
            n2=n2+1;
```

```
            pre(k)=j-y1;
            k=k+1;
        end
    end
end
if (N>1)
    if (mod(N,2)~=0)
        x1=floor(N/2);
        x2=ceil(N/2);
        x3=x1;
        x4=x2;
        x=N;
        if (mod(x1,2)==0)
            for j=1:24
                if (x1~=2 && x1>1)
                    F=F+(2*(S(x1)+1));
                    F1(i)=F1(i)+(2*(S(x1)+1));
                    x1=x1-2;
                end
                if (x1==2)
                    F=F+S(2)+1;
                    F1(i)=F1(i)+S(2)+1;
                    x1=x1-1;
                end
            end
        else
            for j=1:24
                if (x1==floor(N/2) && x1~=1)
                    F=F+S(x1)+1;
                    F1(i)=F1(i)+S(x1)+1;
                    x1=x1-1;
                end
                if (x1>1 && x1~=2)
                    F=F+(2*(S(x1)+1));
                    F1(i)=F1(i)+(2*(S(x1)+1));
                    x1=x1-2;
                end
                if (x1==2)
                    F=F+(S(2)+1);
                    F1(i)=F1(i)+(S(2)+1);
                    x1=x1-1;
                end
```

```
        end
    end
    X=pre(x4);
    for j=x4:N
        S(j)=S(j)-X;
        pre(j)=pre(j)-X;
    end
    if (mod(x2,2)==0)
        for j=1:24
            if (x~=(x3+2) && x>x4)
                F=F+(2*(S(x)+1));
                F1(i)=F1(i)+(2*(S(x)+1));
                x=x-2;
            end
            if (x==(x3+2))
                F=F+S(x3+2)+1;
                F1(i)=F1(i)+S(x3+2)+1;
                x=x-1;
            end
        end
    else
        for j=1:24
            if (x==N)
                F=F+S(x)+1;
                F1(i)=F1(i)+S(x)+1;
                x=x-1;
            end
            if (x>(x3+1) && x~=(x3+2))
                F=F+(2*(S(x)+1));
                F1(i)=F1(i)+(2*(S(x)+1));
                x=x-2;
            end
            if (x==(x3+2))
                F=F+(S(x)+1);
                F1(i)=F1(i)+(S(x)+1);
                x=x-1;
            end
        end
    end
    F=F+X+2+S(N);
    F1(i)=F1(i)+X+2+S(N);
else
```

```
        x=N;
        for j=1:24
          if (x~=2 && x>1)
            F=F+(2*(S(x)-pre(x-1)+1))+pre(x-1);
            F1(i)=F1(i)+(2*(S(x)-pre(x-1)+1))+pre(x-1);
            x=x-2;
          end
          if (x==2)
            F=F+S(2)+1;
            F1(i)=F1(i)+S(2)+1;
            x=x-1;
          end
        end
      end
   end
end
F1max=max(F1(1:n1));
if n2>0
   F=F+b+2;
end
if n3>0
   F=F+b+3;
end
if n5>0
   F=F+b+4;
end
if n7>0
   F=F+b+6;
end
fprintf('Number of Full Adders with coefficient partioning =%d\n',F);
```