# ANALYSIS AND DESIGN
# OF
# POWER EFFICIENT MULTIPLIERLESS DIGITAL FIR FILTER USING RADIX-$2^r$ ALGORITHM

*A Thesis Submitted in Partial Fulfillment of the Requirement for the Award of the Degree of*

MASTER OF TECHNOLOGY

In

VLSI Design

Submitted By

**Vivek Negi**

REG No. 601662020

Under Supervision of

**Dr. Sanjay Kumar**

**Associate Professor**


ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY

(A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB

JULY, 2018

# DECLARATION

I, Vivek Negi hereby declare that the work presented in this thesis entitled **"Analysis and Design of Power Efficient Multiplierless Digital FIR Filter Using Radix-2$^r$ Algorithm"** in partial fulfilment of the requirement for the award of degree of Master of Technology in VLSI Design submitted at Electronics and Communication Engineering department, Thapar Institute of Engineering & Technology (Deemed to be University), Patiala is an authentic record of work carried out under supervision of **Dr. Sanjay Kumar** (Associate Professor, ECED, Thapar Institute of Engineering & Technology) from 2017 to 2018. The matter presented in this has not been submitted either in part or full to any other university or institute for the award of any other degree.

Date: 21st August, 2018

**Vivek Negi**
REG No. 601662020

It is certified that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: 21st August, 2018

**Dr. Sanjay Kumar**
Associate Professor, ECED
TIET, Patiala

21/08/2018

# ACKNOWLEDGEMENT

I would like to convey my deep sense of gratitude to my thesis guide, **Dr. Sanjay Kumar, Associate Professor, ECED** who is a constant source of motivation and firm support in carrying out this thesis. The support and supervision that he gave has helped me to progress in the project. His co-operation is highly appreciated and I highly oblige to him for his valuable comments and moral support during this research period. I value his concern and support at all times, good and bad. He has always emphasis on self-motivation during rough or bad periods and appreciated in good days. The words are not enough to thank him.

I am also thankful to Thapar Institute of Engineering & Technology for the facilities and healthy environment for study. I also express my sincere thanks to my Head of the Department, **Dr. Alpana Agarwal** for providing me adequate environment for carrying the work.

A big thanks to my friends for their support in accomplishment of my course work. They always taught me the patience and never give up attitude in the research work. I would like to thanks my parents for raising me, believing in me, allowing me to do things in my way and agreeing with me even if they don't want to. The presence of my brother and sisters is always supporting and loving in every aspect of my life.

Finally, I would like to extend my gratitude to all those persons who directly or indirectly helped me in the process and contributed towards this work.

Above all, I thank the Almighty God who is being with me and showers his blessings and his grace towards me in all walks of my life.

**Vivek Negi**
**(601662020)**

# ABSTRACT

The advent of the information era has made the use of digital device such as digital cameras, hand phones, hearing aids etc. more and more pervasive. In all these digital devices, the Digital Finite Impulse Response (FIR) filters act as an essential component. In the past few decades, the design and implementation of low hardware cost and low power FIR filters has been a most dedicated topic of research and many efforts are being made in order to save the cost of production of these products along with an increased battery life. There are many techniques available for the designing of such type of filters but the design and implementation of multiplierless FIR filters served as the most successful one because in this technique, the general coefficient multipliers can be replaced by adders and shifts. The fact that the shifts are cost free because they can be hard-wired and only the numbers of adders used, constitute the hardware cost of the multiplierless FIR Filters reduces the overall cost of the filters. It is also believed that the power consumption is also indirectly reduced by the use of adders. Hence, it can be said that the existing algorithm mainly aims at minimizing the order cost but this is achieved at the cost of increased computational complexity for the designing of long filters and the choice of minimum number of adders to be used for minimum power consumption is too coarse and inaccurate.

Therefore, the research work in this thesis deals with the issues of designing multiplierless FIR filters to minimize the power consumption. Reducing the adder depth and switching activity of the filters is the key foundation. Radix-$2^r$ multiple constant algorithm is used for reducing the adder depth and a reduction in switching power is achieved using a low power technique. Hence, overall power reduction is achieved.

# Table of Contents

# List of Figures

# List of Tables

# List of Acronyms

| | |
|---|---|
| **ASIC** | Application Specific Integrated Circuit |
| **CSD** | Canonic Signed Digit |
| **CSE** | Common Subexpression Elimination |
| **FA** | Full Adder |
| **FIR** | Finite Impulse Response |
| **GA** | Genetic Algorithm |
| **GB** | Graph Based |
| **HA** | Half Adder |
| **IIR** | Infinite Impulse Response |
| **LTI** | Linear Time Invariant |
| **MBA** | Multiplier Block Adders |
| **MCM** | Multiple Constant Multiplication |
| **MILP** | Mixed Integer Linear Programming |
| **RAG-N** | Reduced Adder Graph-N |
| **SA** | Structural Adders |
| **SAN** | Shift And Add Network |
| $\mathbf{A_{TH}}$ | Adder Depth |
| $\mathbf{A_{VG}}$ | Average Number Of Additions |
| **BHM** | Bull Horrocks Modified |
| **BIGE** | Bounded Inverse Graph Enumeration |
| **BMA** | Booth Multiplication Algorithm |
| **CAD** | Computer-Aided Design |
| **CMOS** | Complementary Metal Oxide Semiconductor |
| **DAC** | Digital to Analog Converter |
| **ADC** | Analog to Digital Converter |
| **DAG** | Directed Acyclic Graphs |
| **DBNS** | Double Base Number System |
| **DSP** | Digital-Signal-Processor/Processing |
| **FF** | Flip-Flop |
| **FPR** | Fixed-Point Representation |
| **FWL** | Finite Word Length |
| **H(*K*)** | Heuristic With *K* Extra Nonzero Digits |
| **HCUB** | Cumulative Benefit Heuristic |
| **HDL** | Hardware Description Language |
| **MAC** | Multiply-And-Accumulate |
| **MEMS** | Micro-Electro-Mechanical Structure |

| | |
|---|---|
| **MSD** | Minimal Signed Digit |
| **NP-HARD** | Non-Deterministic Polynomial-Time Hard |
| **SD** | Signed Digit |
| **SCM** | Single Constant Multiplication |
| **LUT** | Lookup Table |
| **VLSI** | Very-Large-Scale Integration |

# CHAPTER 1

# INTRODUCTION

Earlier the aim was to develop devices with smaller in size and faster in operation. The power consumption was hardly an issue but with the demand of long battery life for real time applications it's becoming an important aspect of any design. So, industries are facing challenge to design low power, portable and communication devices. This drives the researchers towards finding algorithms that are power efficient. Being the central task in digital signal processing [1], filters having low power can significantly reduce overall power of any electronic system.

For many decades, digital filters are extensively used in digital signal processing (DSP) applications. Compared to their analog counterparts, digital filters offer outstanding performance and flexibility [2]. Frequency selective digital filters are required in most of DSP applications. Digital filters are used in wide range of products and systems such as digital communication systems, digital audio systems, instrumentation systems, image processing systems, radar and sonar, biomedical diagnostics and consumer electronics etc.

Filter design is like another engineering problem that needs a trade-off between performance and low complexity. In integrated circuits, demand for low power consumption restrict the hardware requirement. Hence, digital filter techniques that fulfil low power consumption and low hardware utilization is the current research trend. Finite impulse response (FIR) filter is the most widely used component in many DSP applications. Figure 1.1 shows the application of digital filter in analog conditioning path.

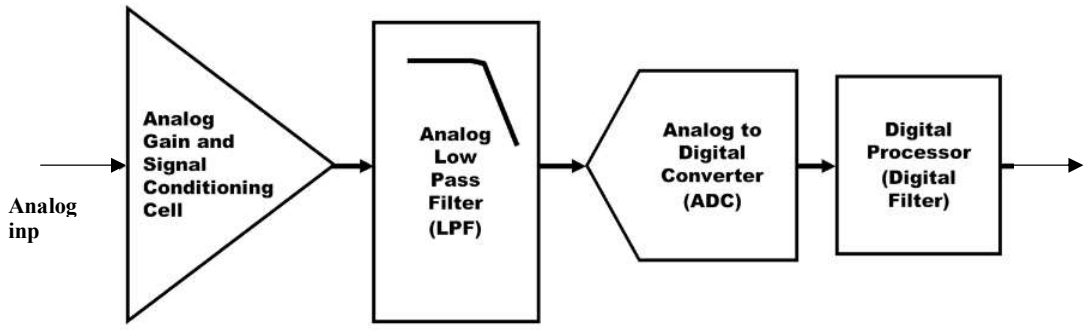



**Figure 1.1 Block Diagram Representing Application of Digital Filter**

## 1.1 Analog Vs Digital filters

The key difference between analog and digital filters is the need of converter. Digital filters need analog to digital (ADC) to convert an analog signal into set of binary numbers for further processing. Whereas in analog filter the signal remains in its original analog form for processing. The important differences between analog and digital filters are given in Table 1.1 on the basis of nature of working signal and implementation characteristics [3].

**Table 1.1 Differences between Analog and Digital Filters**

| Characteristics | Digital filter | Analog filter |
|---|---|---|
| Operating Signals | Operate on digital samples of signal. | Operate on actual analog signals |
| Filter Representation | Represented by linear difference equations. | Represented by linear differential equation. |
| Components Required for implementation | Implementation require adders/substractor, delays, multipliers etc. | Implementation require resistors, capacitors and inductors. |
| Frequency Response | Coefficient searching is done for meeting required frequency response. | Approximation problem is formulated for achieving expected response. |
| Transfer function | Transfer function is rational function of z-transform. | Transfer function is rational function of Laplace variable. |
| Location of Poles | Lie inside the unit circle of z-plane. | Lie on left-half of s-plane. |
| Environmental Tolerances | Less sensitive towards environment and disturbances. | More sensitive towards environment because of component tolerances. |
| Adaptability | More adaptive because of programmability. | Less adaptive. |
| Flexibility | More flexible because control programs can be modified. | Less flexible. |
| Additive Noise | Quantization noise due to quantization of signals. | Thermal noise due to components used. |
| Cost | Less costly | More cost because of analog component. |
| Coefficients | Programmable coefficients | Not programmable |
| Design | Easy to design and simulated on software tool. | Difficulty in design and simulate because of many components. |
| Required Processing Tools | High performance DAC,DSP and ADC tools needed | Not need of converters. |

## 1.2 Mathematical Representation of Digital Filters

In 1960s, the digital technology emerged and unlocked a new era of applications. Digital filters has various advantages over their counterpart analog filters.

- Digital filters are immune to component tolerances because precise level of digital signal is not much important.
- Response of digital filters is invariant to temperature and time.
- Digital filters can be easily programmed on digital hardware.
- Design complexity of digital filters are very low.
- Cost of any digital filter is always less than its analog counterpart.
- Digital filters can produce versatile desired responses.

Any linear time invariant (LTI) discrete system can be formulated in constant coefficient difference equation as

$$y(n) = \sum_{k=0}^{N-1} a\,(k)x(n-k) - \sum_{K=1}^{M} b\,(k)y(n-k) \qquad (2.1)$$

Digital filter is also a LTI discrete system. In (2.1), $a(k)$ denotes forward tap coefficient and $b(k)$ denotes feedback tap coefficients [4]. Transfer function of above equation can be obtained by taking z Transform and rearranging. i.e.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N-1} a\,(k)z^{-k}}{1 + \sum_{k=1}^{M} b\,(k)z^{-k}} \qquad (2.2)$$

In (2.2) two subsets of digital filters can be formulated:

- Finite Impulse Response (FIR) filters (Non-Recursive filters)
- Infinite Impulse Response (IIR) filters (Recursive filters)

The distinction can be made between these two sub classes on the basis that poles are not present in transfer function. Only numerator remains and denominator terms vanishes. Hence the transfer function can be written as [5]

$$H(z) = \sum_{k=0}^{N-1} a\,(k)z^{-k} \qquad (2.3)$$

It is evident from the above equation that impulse response exhibit a finite length and therefore the finite impulse response filter transfer function.

Another basis for distinction can be the physical nature i.e. existence of feedback. IIR filters or recursive have infinite impulse response because of feedback present from the output. FIR or non-recursive digital filters have no feedback hence the finite duration impulse response.

Impulse response of FIR filter of length $N$ i.e. order $N\text{-}1$ can be written as $h = [h_0, h_1, h_2 \cdots h_{N-1}]^T$ and the transfer function of this filter is formalised using Z transform as given in (2.4).

$$H(z) = \sum_{n=0}^{N-1} h_n \, z^{-n} \tag{2.4}$$

The response in frequency domain is determined by substituting $z = e^{j\omega}$ in transfer function.

$$H\left(e^{j\omega}\right) = \sum_{n=0}^{N-1} h_n \, e^{-j\omega n} \tag{2.5}$$

In (2.5) frequency response of FIR digital filter is formulated.
Frequency response can also be written as a product of magnitude response and phase response as given below in (2.6)

$$H\left(e^{j\omega}\right) = H_a(\omega)\theta(\omega) \tag{2.6}$$

where $H_a(\omega) = |H(e^{j\omega})|$ and $\theta(\omega) = \angle H(e^{j\omega})$ are magnitude response and phase response respectively [5]. The group delay of filter is defined as

$$\tau_g(\omega) = -\frac{\partial\theta(\omega)}{\partial\omega} \tag{2.7}$$

Guaranteed linear phase or constant group delay in an FIR filter is achieved by means of symmetry in the impulse response of filter. Impulse response can be symmetrical or anti-symmetrical. Based on the symmetry type (symmetrical or anti-symmetrical) and filter order being even or odd, four types of FIR filters having linear phase can be classified as given in Table 1.2.
From Table 1.2, investigation of frequency response $H\left(e^{j\omega}\right)$ expression conclude following points [5]:

- Phase response $\frac{N}{2}\omega$ is proportional to the frequency and hence the phase is linear.
- Proportionality factor is filter order N.
- Pushing aside the term $e^{j\frac{N}{2}\omega}$, the remained expression is called zero-phase frequency response.
- Addition of duplicated frequency response term with final cosine term due to Eular's formulation results in cosine terms in Symmetric filters.
- Subtraction of duplicate frequency leads to sine terms in Anti-symmetric filters.

**Table 1.2 Types of Liner Phase FIR Filter [6]**

| Type | Order (N) | Coefficients Symmetricity | Frequency Response $H(e^{j\omega})$ |
|---|---|---|---|
| Type 1 | Even | Symmetric with respect to midpoint. | $e^{j\frac{N}{2}\omega}[h(\frac{N}{2}) + 2\sum_{n=1}^{N/2} h(\frac{N}{2} - n)\cos(n\omega)]$ |
| Type 2 | Odd | Symmetric with respect to midpoint (not actual midpoint). | $e^{j\frac{N}{2}\omega}[2\sum_{n=1}^{(N+1)/2} h(\frac{N+1}{2} - n)\cos((n - \frac{1}{2})\omega)]$ |
| Type 3 | Even | Anti-symmetric with respect to midpoint. | $e^{j\frac{N}{2}\omega}[h(\frac{N}{2}) + 2\sum_{n=1}^{N/2} h(\frac{N}{2} - n)\sin(n\omega)]$ |
| Type 4 | Odd | Anti-symmetric with respect to midpoint (not actual midpoint). | $e^{j\frac{N}{2}\omega}[2\sum_{n=1}^{(N+1)/2} h(\frac{N+1}{2} - n)\sin((n - \frac{1}{2})\omega)]$ |

## 1.3 Filter Design Methodology and Specifications

FIR filters are preferred over Infinite impulse response (IIR) filter because of their sure stability and linear phase [7]. Some of important difference between FIR and IIR are listed in Table 1.3. Designing a filter needs its specification to find the impulse response $h(n)$ and these specification contain four parameter i.e.

$$f_{spec} = (\omega_p, \omega_s, \delta_p, \delta_s)$$

where $\omega_p$ is the passband cutoff frequency,

$\omega_s$ is the stopband cutoff frequency,

$\delta_p$ is the maximum allowable passband ripple error and

$\delta_s$ is the maximum allowable stopband ripple error

$\delta_p$ and $\delta_s$ are always provided in absolute error or in decibels. $\omega_p$ and $\omega_s$ given in normalised form or in radians per second [5]. Normalization of radians per second is done by diving by $\pi$ or $2\pi$. When hertz is used for frequency then the sampling frequency should likewise be given such that the $2\pi$ radians/s becomes the sampling frequency and $\pi$ radians/s to half the sampling frequency. Frequencies having value half of that of sampling frequency are completely recuperated in a sampled signal because of sampling theorem.

**Table 1.3 FIR vs. IIR**

| Filter | Phase or group delay | Stability | Order Required | Implementation |
|---|---|---|---|---|
| FIR Filter | Linear phase always possible | Always stable | Long | Can be multirate and have polyphase implementations |
| IIR Filter | hard to control | Can exhibit unstable behaviour and limit cycles | Small | No multirate or polyphase |

Hence, sampling frequency must be expanded to guarantee every one of the frequencies must lie on less than half of sampling frequency, if specification of an application needs frequencies higher than the half of sampling frequency.

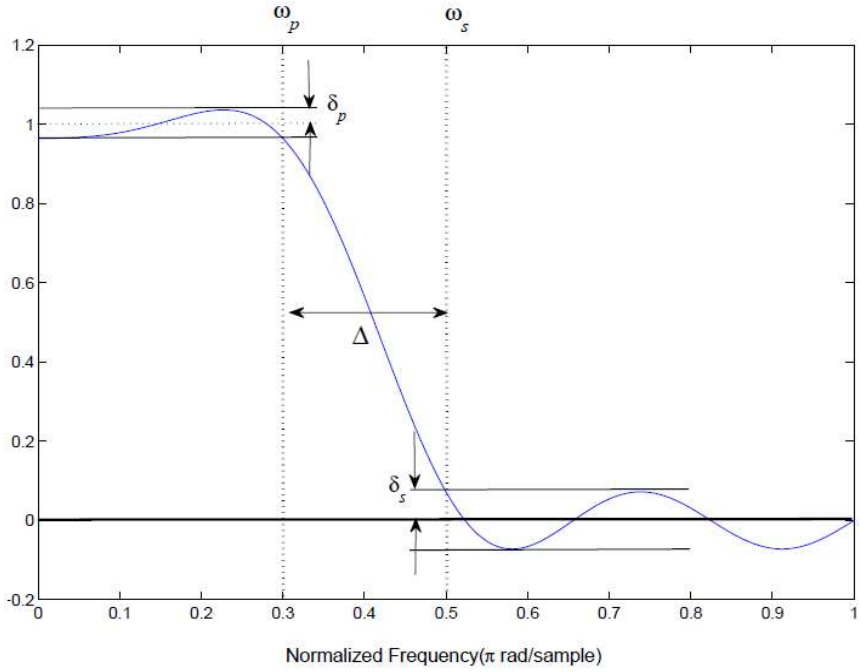



**Figure 1.2 Frequency specification of a lowpass FIR filter [8]**

In Figure 1.2 the four parameter of frequency specification can be clearly seen. Also $\triangle$ is the transition bandwidth of the filter or the distance between passband cutoff edge and stopband cutoff edge. A filter is said to meet its specifications when zero-phase response has passband and stopband ripple error less than that of specified $\delta_p$ and $\delta_s$ .

Order of any filter can be estimated by [8]

$$N \approx D_\infty(\delta_{p;}\delta_s)/\triangle + f(\delta_{p;}\delta_s) + 1 \tag{2.8}$$

where

$$D_\infty(\delta_p, \delta_s) = 10\delta_s[a_1(\log_{10}\delta_p)^2 + a_2\log_{10}\delta_p + a_3]$$
$$-[a_4(\log_{10}\delta_p)^2 + a_5(\log_{10}\delta_p) + a_6]$$

and

$$f(;\ \delta_s) = 11.01217 + 0.51244[\log_{10}\delta_p - \log_{10}\delta_s].$$

The value of unknown coefficients $a_1$ to $a_6$ are

$$a_1 = 0.005309, a_2 = 0.07114, a_3 = -0{:}4761,$$

$$a_4 = 0.00266, a_5 = 0.5914, a_6 = -0{:}4278,$$

In (2.8) a fact can be drawn when filter approaches to ideal, ripple tolerances are very small and transition width becomes very narrow then the order of the filter increases.

## 1.4 FIR Filter Structures

FIR filters can be implemented using many structures because mathematics of all filter in different structure remains same. But when various filters structures are implemented in hardware there is always a differences in hardware cost and the power consumed by the filters. Three mostly used FIR filter structures by researchers are:

- Direct form structure.
- Transposed form structure
- Cascade form structure

Direct form FIR filter structure is shown in Figure 1.3 and it can be noted that the delay line is used to fed input signal samples, the samples generated after delay process are then multiplied with the filter coefficients, and finally these products are summed up for generating output sample. It can be clearly seen in Figure 1.3 the longest zero delay path or critical path of direct form is N adders plus one multiplier, where N is the order of filter. Hence, this type of realization tend to have very long critical path when the coefficients are many i.e. filter order is high.

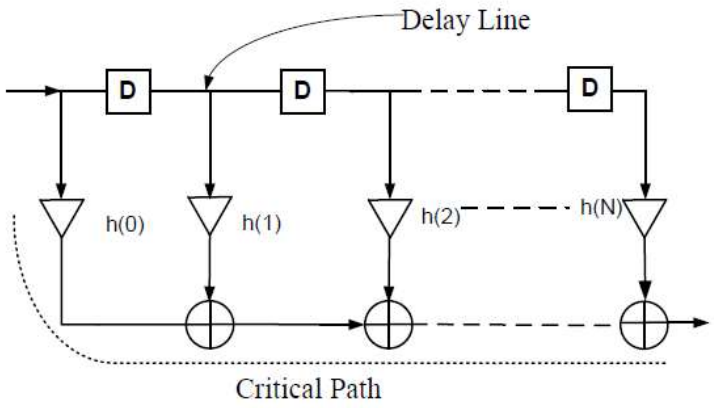



**Figure 1.3 Direct form realization of a FIR filter**

Transposed direct form realization of filter is shown in Figure 1.4 and it is very clear that critical path is very less that of direct form. Critical path consist of only one adder and multiplier i.e. critical path doesn't depends on the order of filter in this type of realization. Hence, throughput of FIR filter realized using transposed form is high not only for low order filter but also for higher order filters. As the delay line width is increasing the hardware requirement of transposed form is higher than that of direct form

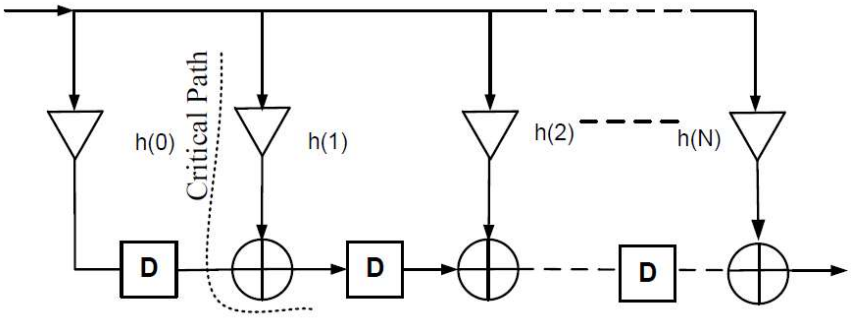



**Figure 1.4 Transposed Direct form realization of a FIR filter**

i.e. more number of filp-flops are required for storing the large bitwidth. Transposed direct form FIR filters have large input capacitance that that of direct form filters.

Cascade form realization of FIR filters is also a popular structure apart from direct and transposed direct form. In this type of realization filter is parted into many low-order sections. These subsections are realized using direct form or transposed form as required by the particular application. In (2.9) z-transform transfer function of N order filter realized using cascade realization is described. The z-transform is decomposed into $N_1$ and $N_2$ subsections or subfilters.

$$H(z) = \sum_{n_1=0}^{N_1} h_1(n_1) z^{-n_1} \sum_{n_2=0}^{N_2} h_2(n_2) z^{-n_2} \tag{2.9}$$

Order of filter N is summation of $N_1$ and $N_2$ i.e. $N= N_1+N_2$. Realization of transfer function is shown in Figure 1.5 where each of subfilter is realized using direct form. The advantage of cascade form realization of FIR filters are their low sensitivity of coefficient along with low computation complexity, compared with the other two forms of structure [9].

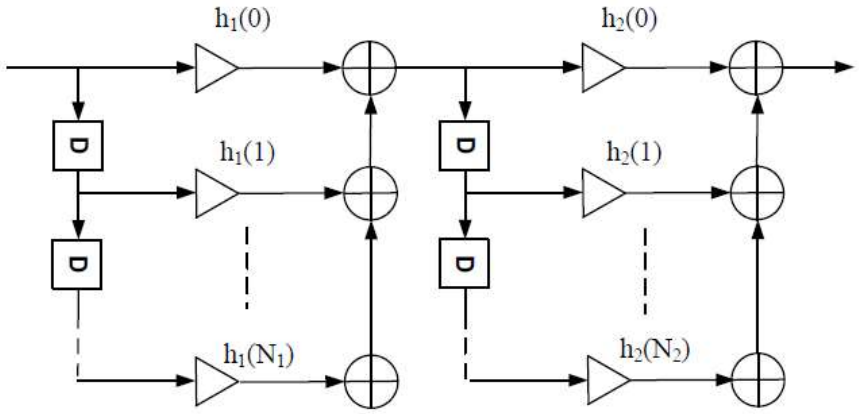



**Figure 1.5 Cascaded form realization of a FIR filter**

If particular application require high speed, the preferred realization structure has to transposed direct form because of least critical path. Hence, Selection of realization structure depends on the application requirement.

## 1.5 Mutliplierless FIR filters

In very-large-scale integration (VLSI) implementation of DSP algorithms, multiplication is the source of complexity in addition with expensive multiplier. Also multiplication consumes high power and is the main source of degradation of throughput, in addition to its large amount of area consumption. FIR filters have a disadvantage in comparison with IIR filters, which are the high power consumption and computational complexity. Researchers made many efforts for design of low power and low complex FIR filters. The most prominent and successful technique is the multiplierless design of FIR filters.

In multiplierless filtering multiplication of coefficients with delayed input is replaced by a network of adders and shifts. Implementation complexity and power consumption significantly reduced as the multiplier being the most expensive component and power-hungry component in VLSI realization of FIR filters. These algorithms are called multiple constant multiplications (MCM) because of the fact that each of pre-designed discrete constant coefficient is multiplied by input signal in transposed direct form filter.

An area of $O(L^2)$ is consumed by any array multiplier in comparison with an adder consumption of $O(L)$ area having length L [10]. Comparison can be seen in Figure 1.6 and Figure 1.7 that reducing number of non-zero bits in constant coefficient sets can reduce signification amount of power.

```
x3 x2 x1 x0    x    1 0 0 1    =          x3 x2 x1 x0
                                  +   x3 x2 x1 x0          .
                                      y6 y5 y4 y3 y2 y1 y0
```

**Figure 1.6 Multiplication of constant 1001 using adders**

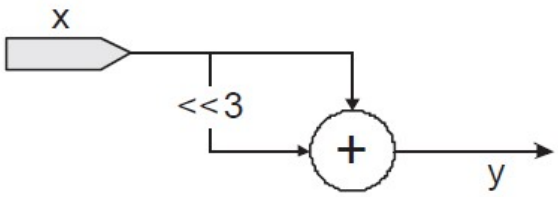



**Figure 1.7 Multiplication of constant 1001 using shift and add**

Since the coefficients sets of any FIR filter are not unique to satisfy filter specification, there exit some other coefficients sets that meets filter specification and can be realized with low power and low hardware consumption.

The research on multiplierless implementation of FIR filter can be divided into three divisions:

- SPT Techniques: In these techniques encoding of set of constant coefficients is done using signed power-of-two (SPT) representations [11]. SPT representation contains less number of non-zero digits as compared to conventional binary encoding that leads to efficient add/subtract and shift network implementation. As number of SPT terms directly affects the hardware cost, hence for a given filter specification SPT techniques based algorithms aim to reduce number of SPT terms. In other words the techniques aim to minimize the ripple in frequency response with fixed number of SPT terms.
- MCM Techniques**:** Multiple constant multiplication (MCM) techniques uses the fact that common SPT terms are shared among filter coefficients. These SPT terms are called common subexpressions. Using these common subexpressions all constant coefficients are implement together and share common SPT terms to form the add-and-shift network compared to synthesizing of coefficients separately [12]. As input sample is multiplied with multiple constant and then delayed in transposed form of realization FIR filters hence these techniques are called multiple constant multiplication techniques.
- MCM and Coefficient Joint Optimization Techniques: For a particular filter specification there exit different coefficients set to meet FIR filter specification i.e. filter coefficients sets are not unique. Search of coefficients set such that the synthesis of these coefficients consume low power and have less hardware requirement in addition to the satisfying filter specifications. These techniques use of MCM algorithms with coefficient optimization further reduces the power consumption and hardware cost. Synthesis of constant coefficient according to MCM techniques and search for

constants are accomplished alternatively. Tree search algorithms are used for investigating solution of discrete coefficients bounded by search space when there is no limit on computation resources and search duration. But searching of coefficient set for higher order filters require more complex computation and to solve this problem many heuristic algorithms are proposed. These heuristic techniques uses reduced search space also called sub-optimal methods [13].

## 1.6 Outline of the Thesis

The entire thesis is organized into six chapters. The chapter wise summarization is given below:

**Chapter 2: Literature Review**

This chapter presents the rigorous literature review for the research work. It presents the evolution design techniques for designing discrete coefficient filters such as signed power-of-two, common sub-expression elimination and multiple constant multiplication techniques. Review of low power techniques is also presented.

**Chapter 3: Arithmetic for Coefficient Representation**

This chapter covers the fundamentals of the arithmetic for coefficient representation. Mathematical analysis of different number systems along with their comparison is demonstrated. Radix-$2^r$ arithmetic which is used in thesis for coefficient representation is also discussed in detail.

**Chapter 4: MCM Algorithm and Power Reduction Technique**

This chapter discusses multiple constant multiplication (MCM) algorithm used in the thesis to implement the multiplierless FIR digital filter. Further, power consumption of multiplierless FIR filter has been discussed. Switching activity reduction technique which is used in this thesis is also explained.

**Chapter 5: Simulation and Implementation Results**

This chapter describes the simulation results of five benchmark filters. Utilization and delay of filters in reconfigurable designs is tabulated. Power analysis of each filter is done and compared with most prominent algorithms for power reduction.

**Chapter 6: Conclusions and Future scope of the work**

Finally, the conclusions of the proposed work along with the future scope is presented and discussed in this chapter.

# CHAPTER 2
# LITERATURE SURVEY

Filtering process plays a key role in every digital signal processing applications. Planning about the optimal linear phase FIR filters has always been the main consideration as a result of their stamped prevalence over IIR filters. FIR filters provide stability in finite wordlength designs hence their implementation is an active area of research. Reducing area and power has been a major research concern for multiplierless designing of linear phase FIR filters. Recent VLSI trends need low-power devices thus the study of power reduction techniques and algorithms that tend to have low logical depth are the foundations of this literature.

The following sections discuss the design techniques for designing discrete coefficient filters such as signed power-of-two, common sub-expression elimination and multiple constant multiplication techniques. Finally, literature that are fundamental backbone of thesis are presented.

## 2.1 Linear Phase FIR Filter Design in Sum of Power-of -two Space

Kodek [14] firstly presented the discrete FIR filter design. Integer programming has been used for upscaling the coefficient and representing those in integer format. The technique of branch and bound was successful for designing of discrete filter design. The time required for designing crosses the gain in error because of unimportant design as compared to infinite precision designs that uses less time.

For the justification of more time elapsed Lim *et al.* [15] presented the techniques where discrete coefficients are represented in power of two subspace. Hardware resources used by power-of-two were nil because simple hardware shifts generation of coefficients representation hence it is proven to be very attractive in achieving reduction of hardware complexity. Author represented coefficients as:

$$h(n) = \sum_{i=1}^{N} S_i \times 2^{p_i(n)} \tag{2.1}$$

where, $N = 2, S_i(n) \in -1,0,+1$ and$-9 \leq p_i(n) \leq 0$.

The filter was designed with a maximum length of 40 because of limited computation resources available at that time but the work opened the doors for other researchers to work in the field of optimal designing of discrete FIR filters.

In [16] Lim changed the objective to Normalized Peak Ripple Magnitude (NPRM) filter design. Unity gain was inconsequential as purpose of filter was to allow incoming signal over some set of frequencies and attenuate the other set of frequencies. Constraint on passband was must because in one hand higher gain meant good signal-to-noise ratio but on other hand overflow caused was a big issue. While representing coefficients in power-of-two, the lower bound passband gain was inherit to be half of upper bound gain. For getting the same NPRM,

gain could be multiplied or divided by a particular amount of power-of-two in the respective range. The gain sectioning technique has been introduced for finding coefficient set having optimal NRPM which divide the range of gain into other gain sections and using upper bound with the constraints of lower bound over the section of gain for the designing of discrete FIR filter. The simple technique that can be applied was to section the range [0.7, 1.4] and select the best suitable solution among all the sections. There was a trade-off between timing of design and quality of design because of increased design cycle time as partitioning in more sections leads to better design. Also, another technique which was based on elimination was studied.

Samueli [17], represented the discrete FIR filter coefficients in another type of representation technique called Canonic Signed Digit (CSD). CSD coding tend to be special coding scheme in which number of non-zero digits were minimum and no two adjacent digits were non-zero. Representation of coefficients was done with 1, -1, and 0. Hardware complexity of a subtraction and addition are same hence the scheme exhibit 33% fewer non-zero terms efficiently over $2^s$-compliment representation of coefficients. Representation of a number in CSD coding is unique. Density of CSD code are very high in small value of coefficients and codes using non uniform L power-of-two terms (in opposition to the uniform 2's complement) led to an extra non-zero bit to be used for representing filter coefficient of magnitude greater than or equal to 0.5. Justification of extra bit was that it improved the frequency domain response of filter while having a negligible effect on hardware usage.

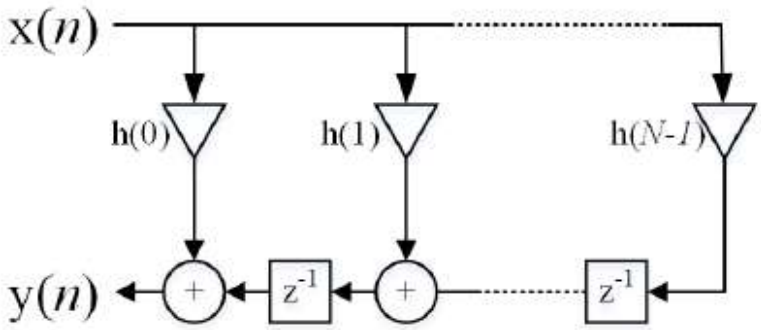



**Figure 2.1 Transposed Direct Form**

Impulse response of low pass filters follow $sinx/x$ function hence, the percentage of large integer coefficients was small. In this proposed algorithm, first calculate the scaling factor and then search in local bivariate neighbor around the calculated scaled coefficients. After searching the coefficients are rounded according to the algorithm.

Lim *et al.* [18] developed a SPT term for representing filter coefficients based on sensitivity to frequency response. Coefficients are allotted different number of SPT terms and then an integer-programming algorithm was used for optimization of coefficients. In this technique, more focus was given on larger coefficients.

## 2.2 Multiple Constant Multiplications (MCM) Algorithms

Bull *et al.* [19] presented a work on filtering the input signal as multiple constant multiplication problem given in (2.2) by transposed direct form structure of FIR filter design (Figure 2.1) where output of respective multiplier is given by $y(i)$. Authors contended that chip area and throughput are affected greatly by the multiplication operation. They noted that the complexity of array multiplier in terms of area was $\mathbb{O}(B^2)$ where B denotes the wordlength of output whereas complexity of an adder was $\mathbb{O}(B)$. Hence, primary goal was to reduce implementation cost of multiplier for reducing the overall implementation cost of filter.

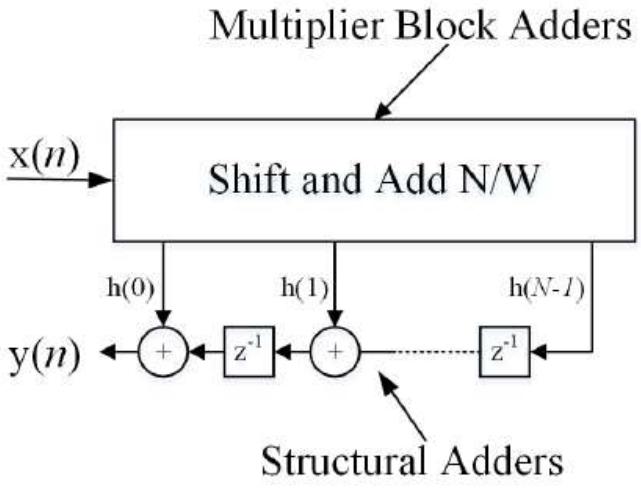
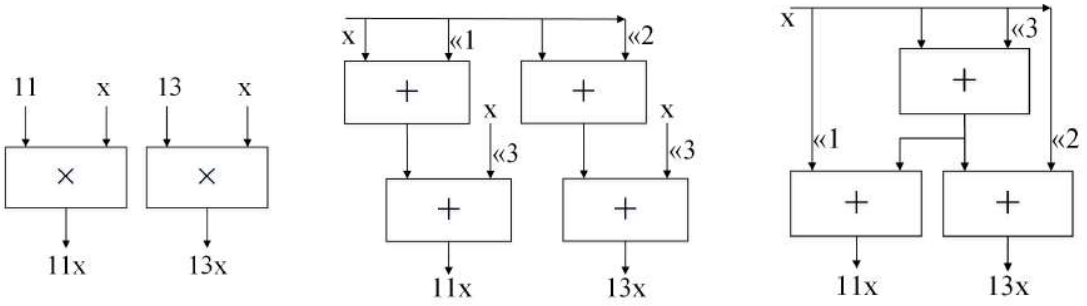



**Figure 2.2 Shift and Add Network**

In fixed coefficient filters, the common expressions can be eliminated and the results generated partly could be used for generation of multiple coefficients. Implementing filtering and DSP transform gets more attention with the research in MCM algorithms.

$$y(i) = h(i)x \; i = 0,1 \cdots N \tag{2.2}$$

Shift and Add Network (SAN) used in place of multipliers for problem of Multiple Constant Multiplication in fixed constant operation filtering. Also in these, a transposed direct form structure is used for filter design. In Figure 2.2 the Multiplier Block Adders (MBAs) represents the adders used for implementation of coefficients inside the SAN and the Structural Adders represent the summing elements of delayed and weighted input signal. The optimization problem of MCM algorithm has been researched a lot and is proved to be a NP-complete problem. This justifies the use of heuristic algorithms in order to solve the problem [20]. Comparison of three techniques to implement the FIR filter coefficients shown is Figure 2.3.




(a) Multiplier Approach (b) SOPT Approach (c) MCM Approach

**Figure 2.3 Filter Design Approach**

MCM problem can be solved by two techniques as studied in literature. The techniques are Common Subexpression Elimination (CSE) by Yurdakul *et al*. [21] and Potkonjak *et al*. [22] and Graph Based (GB) by Dempster *et al*. [23].

In CSE algorithms, the first step is to define the filter coefficients to be multiplied in any representing scheme e.g. Binary, CSD, or MSD. Second step is to search for common expression among the constants and the most shared subexpression is used for further implementation. In opposition to CSD technique, GB techniques do not utilize any number representation scheme for adder network instead the network ends at parent branches from child branches as a graph.

The RAG-*n* algorithm [23] is most prominent graph base algorithm for multiplier implementation inside a filter or a DSP transform. There are two main parts of this algorithm, one being exact and other one heuristic. The exact one outperforms all the Common Subexpression algorithms but the application of algorithm is restricted by some particular set of numbers. The limitation of this algorithm was that it can only represent constants which are calculated as the sum of other two coefficients by using an adder and at least, there exists a constant which is a cost one constant [24]. Being specific to a representing scheme and search is done only over common subexpressions, CSE algorithm lacks behind the RAG-n algorithm at that time.

### 2.2.1 Common Sub-Expression Elimination for Design of Hardware Efficient FIR Filters

Hartley [25] worked on the FIR filter canonical structure which reduces the number of adders to half. The table shown in Figure 2.4 is Hartley's table which arranges the representations in an NxL format, where N denotes the number of coefficients and L is wordlength of the filter. The common patterns (subexpressions) were searched in this table. The representations effected the number and type of subexpressions. Thus,

Hartley's method is classified into four methods (Figure 2.5), the horizontal common-subexpressions (HCSE) method, the vertical common-subexpressions (VCSE) method, the oblique common-subexpressions (OCSE) method and the mixed common-subexpression elimination (MCSE) method. Another classification includes a multiplier block (MB) method which works only on the transposed structure of the filter. Thus, the MB methods can be sub-classified into an HCSE, which works on the MB and was denoted by HCSE-M, and independent (RI) methods.

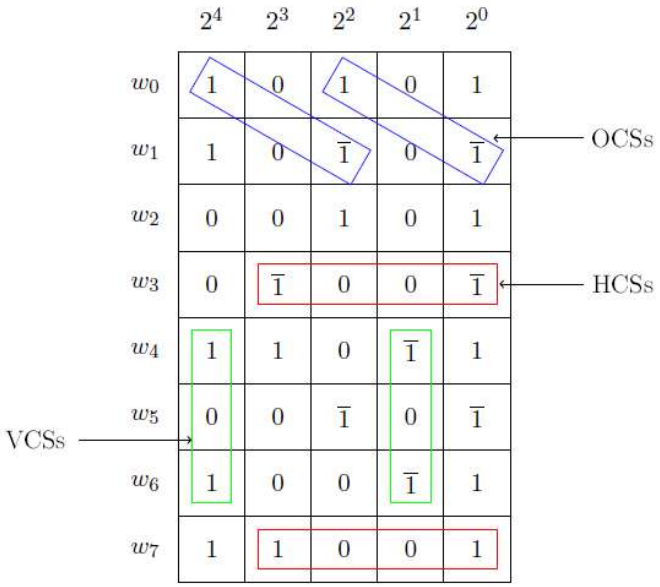



**Figure 2.4 Patterns in Hartley's table**

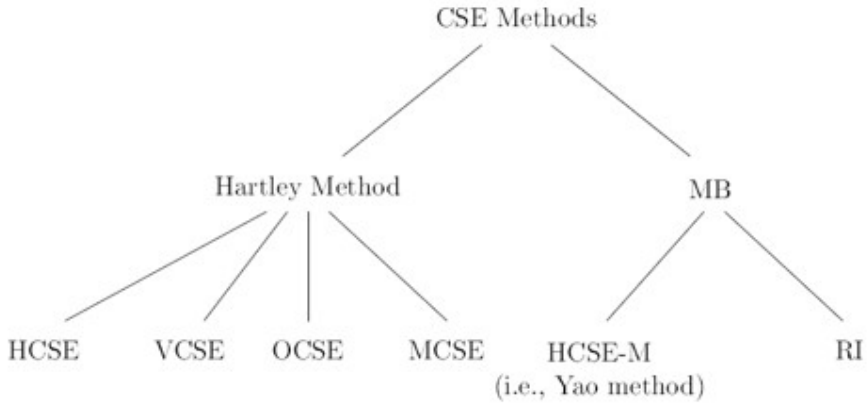



**Figure 2.5 Classification of CSE Methods**

Potkonjak *et al* [22] presented a CSE algorithm known as the iterative matching algorithm (ITM). In this algorithm, the best matched patterns across each coefficient pair were searched along with the Sign digit (SD) representations. The authors used a greedy algorithm to select the best SD representations.

Farahani *et al.* [26] presented a modified iterative matching (MITM) algorithm. In this algorithm, the number of non-zero bits for all pairs of coefficients were calculated to detect and eliminate more common subexpressions.

Pasko *et al.* [27] proposed another variation of HCSE in which CSE are calculated as a linear transform where LxN matrix was Hartley table. Common patters among CSD representation of coefficients are searched on the basis of hamming weight of each coefficient. A steepest plummet approach was utilized to choose the highest frequency pattern. Matrix is divided in two sub-matrixs, the common-subexpression matrix and the remainder matrix because of selecting pattern with maximum frequency. Marcos *et al*. [28] modified the technique of Pasko *et al.* by using common subexpressions recursively due to decrement of subexpression sharing but the at the cost of increasing the logical order.

Vinod *et al.* [29] presented a better technique for eliminating common subexpressions called VCSE method. It searches and eliminates the VCs and this gives better elimination than the HCSE for some coefficients. Authors also mentioned that for realizing symmetric coefficients the HCSE method may not be useful in some cases because of additional requirement of adder that increases the number of logical order. It was also shown that OCSE is like VCSE because symmetry of the coefficients in symmetric filters maybe destroyed and there exists more subexpression elimination procedures that are based on Hartley Table. Mixing of HCSs and VCSs, HCSs and OCSs are such possible pattern finding algorithms.

Samadi *et al.* [30] proposed an identification graph which have all the horizontal and vertical subexpression information. The problem definition was to find CS using the Hamiltonian Walk. The reason for using a genetic algorithm was to find and eliminate maximum number of subexpressions.

Vinod *et al.* [31] proposed a concept of super-subexpressions (SSs) formation made in Hartley's table. CSD representation was used and SSs are formed using three or four non-zero digits. Mahesh and Vinod [32] presented the same technique but used binary digits for representation of coefficients sets because of the formation of two bits from most of the non-zero bits. As a result, most of bits are paired and smaller number of bits are unpaired in comparison with the CSD coding.

Hatai *et al.* [33] presented a vertical-horizontal binary common subexpression elimination for real time FIR filter such that the algorithm can be applicable to dynamically changing coefficients of reconfigurable filter. Firstly, a 2-bit BCSE is applied vertically across nearby vertical coefficients then a horizontal application of BCSE was used within each of coefficient. The proposed technique reduced the switching activity of MB adders by 19.6% as compared to previously known VHBCSE

algorithms. FPGA and ASIC implementations are done and results in an efficient fixed point reconfigurable FIR filter design.

Chen *et al*. [34] proposed a new algorithm that uses the frequency response specification to synthesize FIR filter. Previously proposed literature has an over fit and under fit response of filter but in this paper these effects are overcome by examination of sensitivity of sub-expression on error response during elimination of subexpression and making filter tap along with filter length as a variable determined by examination of frequency response with additional tap and expanded bit is applied respectively. Over 20.9% of area and 72.7% of power saving is achieved on benchmark FIR filters.

### 2.2.2 Directed Acyclic Graph (DAG) Based Algorithms

Bernstein *et al.* [35] proposed a Directed Acyclic Graphs (DAG) Algorithm. These are base up techniques that iteratively develop the graph speaking to the multiplier square. The graph was developed by the heuristic that decides that the following graph vertex has to be added to the graph. The nodes of the graph describe the coefficients. The edges are directed and speak to the values that are equivalent to the coefficients present in the parent node which were shifted by some sum. The coefficient value that is put away at each node is acquired by including up the qualities the approaching edges to the node. GB algorithms are not limited to a particular representation thus, offering more degrees of freedom and commonly create arrangements with the least number of operations.

Voronenko *et al.* [36] developed the Heuristic of Cumulative Benefit (Hcub) algorithm. There were two parts of heuristic i.e. the optimal and the heuristic part. Number of definitions are required by this non-trivial heuristic. All things considered, the primary thought behind Hcub is to utilize a superior heuristic for incorporating middle of the synthesizing fundamentals. Hcub has more computational cost than its DAG partners since it investigates a substantial space of conceivable middle vertices. Also, it doesn't require a pre-generated optimal SCM lookup table as different calculations do. Hence, Hcub is storage productive and in its immaterialness is just restricted by the computation time.

### 2.2.3 Hybrid Algorithms (CSE & DAG)

Hybrid algorithms combine different algorithms from different classes. CSE are already considered as mixed algorithms since they rely on a digit recoding followed by a pattern matching. But what is generally meant by "hybrid" in the literature, are those algorithms combining CSE and DAG. A Typical example is the recent Thong's BIGE algorithm.

Thong *et al.* [37] presented a minimized form of Bounded Inverse Graph Enumeration (BIGE) algorithm. It is an exhaustive SCM calculation, as it combines two CSE heuristics (H(k)-ODP) and an optimal DAG calculation. Digit clashing problem was cured by Overlapping digit pattern (ODP) [16]. CSE calculations are helped by ODP for searching and replacing the nonstandard patterns. Upper bound obtained by the heuristic (H(k)-ODP).For lower bound exhaustive search using DAG was done. Lower bounding exhaustive search continuously done until they meet or until a solution is found. Only n-1 adders are considered by exhaustive search when any heuristic uses n adders for getting the result. The exhaustive search is done by means of lookup table till 4 adders. For exhaustive search of 5 or more adders, inverse graph enumeration (the "IGE" in BIGE) is utilized.

### 2.2.4 Genetic Algorithm Based Filter Design

A genetic algorithm was presented by Ye *et al.* [38] for designing multiplierless filter. A reduced search space was created around a base solution. For a corresponding gain, the base solution was obtained by discretizing a continuous solution. There was a possibility of encoding the difference between the possible values the GA for a base solution because the value of chromosomes of the GA leads to a shorter encoding scheme. Thus, the search space was reduced along with the acceleration of the search process of the GA. The mutation and crossover operations were also modified with adaptive rates.

The objective of the algorithm was to reduce the number of adders in the implementation. The RAG-n [23] algorithm was used to determine the number of adders. As the search process is independent for each passband thus, each problem was casted to a different machine and ran in parallel. Since, the RAG-n algorithm consumes a substantial time so a fitness function, that inhibits the algorithm for running the RAG-n algorithm, was also formulated for solutions that were unable meet the error constraints.

### 2.2.5 New Recoding Algorithm (Radix-$2^r$)

After performing a survey on binary arithmetic's, it is evident that the radix-$2^r$ is the best suited algorithm for the aforementioned requirements. It has been fully exploited for designing of efficient multiplier cores, which serve as the heart of the linear systems. Radix-$2^r$ algorithm is the general form of SD arithmetic ($r$=1) and CSD algorithm (the mathematical proof is missing for the time being) thus, it could lead to higher speed and less logic resources than CSD since $r$ bits are processed simultaneously.

Sam [39] introduced radix-$2^r$ representation of n-bit complement number in which the number can be split into more two's complement slices whose length are determined by the formulation given by author.

Oudjida A. K. and Nicolas C. [40] first time introduced radix-$2^r$ arithmetic as coefficient representation scheme in MCM problem. Authors focused on minimization of adder cost. In comparison with CSD algorithm presented, it required 23.12% and with DBNS it required 2.07% less average additions for constant of 64-bit.

Oudjida A. K. *et al***.** [41] presented an improved version of heuristic for MCM. The recording was based on sublinear-runtime with the assistance of complexity on upper bound. Author's present more improved analytical expressions of maximum adder depth and average amount of additions required for the process. Redundant recording is done then common-digit-elimination technique is used for improvement of heuristic. Later Oudjida *et al*. [42] modified the previous work by applying the heuristic on high-complexity MCM problems. The proposed technique was only one which could be feasible to run on high-complex runtime problem. The one more advantage of this technique was the achievement of shortest adder depth in comparison to best published MCM algorithms.

Liacha A *et al.* [43] presented radix-$2^r$ arithmetic based implementation of FIR filter design. The radix-$2^r$ was used for generation of multiplier block and a HDL code was written. Then the synthesis was done for comparison with existing heuristics on the basis of optimization. It has shown that radix-$2^r$ exhibits shortest logical adder-depth, hence it was promising to give best results in respect of speed and power. Also they were not competitive to Radix-$2^r$ in high order filters.

## 2.3 Analysis of Switching Activity for Low Power Designs

Switching activity increases the power consumption of any design hence, for low power designs it should be minimized. Researchers proposed many techniques for reducing the switching activity. Literature on techniques that are base of thesis are presented here.

Johansson *et al.* [44] proposed a model for analysis of switching activity of constant multipliers. Presented model could be used for full adder architectures and algorithm that were for low power designs. Simulations was done to verify the model, by focusing on carry-save arithmetic. The model can be successfully used for MCM problems as these algorithms need low power techniques.

Ye *et al.* [45] worked on a new model for calculating switching activity of multiple constant multiplication block of any multiplier by using the spatial dependence among the considered input of adders. Authors presented a new power measurement model for estimation of MCM block power consumption of FIR filters. Several benchmark FIR filters are taken for

demonstrating the accuracy of proposed model on the basis of dynamic analysis of power and proposed model outperforms the existing power model.

Lou *et al.* [46] performed analysis of critical path of MCM algorithm block for low complexity implementation. Authors proposed a signal propagation path-delay model for critical path of MCM block more precisely in comparison to other models that used adder depth and cascaded full adders. They also presented a Dual objective configuration optimization (DOCO) to optimize A-operation for high-speed and low-complexity MCM blocks. Algorithm is implemented for fundamental set present along with additional set of fundamentals. These additional fundamental are searched with genetic algorithm proposed in paper. DOCO was applied to these searched fundamentals for optimization of configuration of A-operation. Experimental results show that power, critical path delay, power delay and area delay product are reduced in comparison to existing methods for optimization.

Lou et *al.* [47] studied that the implementation, optimization of FIR filter design and stated that product accumulation block (PAB) optimization of a filter in transposed form was ignored by many researchers. Authors calculated the power consumption of multiplierless filters and verified them with theoretical results and concluded that PAB plays a significant role in overall power consumption of filter. Hence, two techniques of registering the outputs of MCM block are presented. These methods reduces the power consumption of PAB and simulation of the same results in the reduction of total power at the cost of area usage is also performed.

Ye *et al.* [48] presented an optimization of multiplierless FIR filter by focusing on power consumed by the structural adders (SAs). In the proposed algorithm optimization the objective was set on power index and average depth of SAs in discrete coefficient searching process. In comparison with best algorithms which aim to reduce the number of total adders used, the proposed technique consumes less power. Gate level simulation is done for comparing results of benchmark filters. The maximum power saving on existing designs of filter was 19.6%.

# CHAPTER 3
# ARITHMETIC FOR COEFFICIENT REPRESENTATION

This chapter covers the fundamentals of the arithmetic for coefficient representation. Most commonly used number schemes of any computer arithmetic are in fixed-point and floating-point formats. Precision and dynamic range are two main parameters that are used to compare these formats. Number representation schemes that are widely used, are based on fixed-point such as canonical-signed-digit representation, the double-base number system, the residue number system, and the radix-$2^r$ number system. In hardware implementation of linear systems, addition and multiplication functions are most important arithmetic operations. Hence, these operations are investigated in this chapter.

## 3.1 Introduction to the Binary Arithmetic

In binary arithmetic, logic 0 and logic 1 are two symbols that used to represent any number. Thus, strings of 0's and 1's are used to represent a binary number. Basic electronic component i.e. a Transistor is used to build complex components using on/off position to represent 1/0 symbol. Presence of an electrical current in transistor is denoted by 1 and absence is denoted by 0. Binary numbers are being used currently because there is no technology to create "switches" which can store or have more than two possible sates. Such switches are theoretically possible at a quantum level, but quantum computers are not on sale for the time being.

These symbols ‘0’ and ‘1’, in mathematical numeric systems are known as digits, quantity of these symbols is called base or radix. Radix-2 is used to express binary numbers. The commonly used notation for representation of any number is $(x)_y$, where number expressed is x in the base y. Numeral systems such as octal (radix-8), hexadecimal (radix-16) are used to help in manipulating long strings of 0’s and 1’s in any hardware design. The most efficient and effective way for implementation of any hardware has been realized by binary (radix-2).

Mathematical field that belong to binary arithmetic focuses on:

- Investigation of number representations keeping in mind the end goal of reducing the logic redundancy in recording and finding the smaller numeric bases that can represent the binary numbers with least number of digits.
- Proposing algorithms for effective arithmetic operations $(-, +, \times, /, a^n, etc)$ based on investigated number representation scheme.
- Investigation of techniques for implementing on any computational device such as DSP, FPGA, or an ASIC.

Many binary arithmetic exits, each having characteristics that depends on format used for representation of numbers. Arithmetic that have been widely used by researchers are fixed-point arithmetic and floating-point arithmetic. The implementations and optimizations undertaken within each arithmetic are drastically different [37]. Precision and dynamic range desired by particular application plays important role in choosing arithmetic. Complexity of any arithmetic is directly proportional to the required number of operations. For example, the linear system always have properties that are simple to understand and easier to manipulate than the nonlinear system.

### 3.2 Formats for Number Representation

Infinite, continuous set of real numbers can be represented with a finite set of machine numbers but this type of representation being complex and commutative, have inherit property of compromise with requirements of application. The requirements of any application can be many but the important ones are listed below [49].

- *Speed:* In any computation-intensive system such as DSP, it is a critical factor that describe the overall performance of application.
- *Precision:* A fast application producing incorrect result is worse than a sluggish application producing correct result.
- *Range:* Need of representing large as well as small numbers.
- *Portability:* Application developed must operate on different machine will no modification.
- *Ease of use and implementation:* Arithmetic must be simple so that everyone can use it.

In literature various kinds of number formats exit [50]. The most ordinarily utilized number formats are given below:

- *Floating-point number format:* Most commonly used approach, because of large dynamic range with high precision so that it can handle huge number as well as small number.
- *Fixed-point number format:* Offers a limited range and/or precision, but easy to implement. Very convenient for high-speed and low-power applications. It handles integer numbers $x \in I = \{-N, \dots, N\}$ as well as rational numbers of form $x = a/2^f$ ("binary" rational), $a \in I$ and $f$ is a positive integer.
- *Logarithmic number format:* Representation of numbers done by means of logarithms and signs. Appealing for wide dynamic range and low precision applications.
- *Rational number format:* Ratio of two integers used for approximating a real number. Arithmetic operations are very complex.

This thesis is limited to fixed-point representations format. Hence, discussion of fixed-point representation is the main focus of this chapter.

### 3.2.1 Fixed-Point Format

An Ordered *n*-tuple is used for representing a number in fixed-point format. Each of the elements of the *n*-tuple is called a digit, and the *n*-tuple is called a digit-vector [37]. Both non-negative and signed integers are discussed below.

#### *3.2.1.1 Representation of Non-negative Integers*

Any integer $x$ can be represented in digit-vector as:

$$x = (x_{n-1}, x_{n-2},, x_1, x_0) \tag{3.1}$$

Representation of $x$ in Equation (3.1) consist of following elements:

- 'n' number of digits
- A set of numerical values for the digits. Calling $D_i$ the set of values of $x_i$. The cardinality of the set $D_i$ is denoted by$|D_i|$. For example, $\{0,1,2,\dots,9\}$ is $|D_i|$ with cardinality 10.
- Rule that governs the mapping between $D_i$ and $x$ called interpretation rule. Weighted number systems are most frequently used and their mapping as given below:

$$x = \sum_{i=0}^{n-1} x_i \times w_i \tag{3.2}$$

where, $w = (w_{n-1}, w_{n-2}, \dots, w_1, w_0)$ called the weight-vector.

In weighted number system such as radix number system, $w$ is related to radix-vector $r = (r_{n-1}, r_{n-2},, r_1, r_0)$ as follows:

$$w_0 = 1; \quad w_i = w_{i-1} \times r_{i-1} \ (1 \le i \le n-1) \tag{3.3}$$

$$w_0 = 1;\ w_i = \prod_{j=0}^{i-1} r_j \tag{3.4}$$

Classification of radix number system with respect to radix-vector are fixed-radix and mixed radix systems. Fixed radix consist of same $r$ value. Hence, the weight-vector is represented in Equation (3.5) and the digit sets are represented in Equations (3.6) and (3.7).

$$w = (r^{n-1}, r^{n-2}, r^2, r1) \tag{3.5}$$

$$D_i = D \ (1 \le i \le n-1) \tag{3.6}$$

$$x = \sum_{i=0}^{n-1} x_i \times r^i \tag{3.7}$$

Power-of-two are mostly used radices, for example 2 (binary), 4 (quaternary), 8 (octal), 16 (hexadecimal) etc. Range of number $x$:

$$0 \leq x \leq r^n - 1 \tag{3.8}$$

Set of digit values can be redundant and non-redundant systems. In redundant number system, there is a unique representation of an integer for example binary, quaternary, octal. The hexadecimal number system have non-redundant digit sets {0,1 }, {0,1,2,3 }, {0,1,2, … ,7}, {0,1,2, … ,15}, respectively.

A redundant system have more than one representation of a value for example, in binary system $\{-1,0,1\}$ the digit vectors $(0,0,1,1,1,1,0)$ and $(0,1,0,0,0,-1,0)$ both represent integer "thirty" as given in Table 3.1.

Well known system called canonical signed digit (CSD) is non-redundant even if $|D_i| > r_i$ used to produce minimum number of digits and hence used in design of number of LTI systems [51]. A system with fixed positive radix *r* and non-redundant set of digit values is called a radix-*r* conventional number system.

Digits vectors are represented by bit-vectors for implementation of arithmetic algorithms in digital systems. A code is defined and then the digits of digit-vectors are mapped one by one according to this code. Higher radices are used to represent integers because of identical bit-vectors. In the binary case, each bit corresponds to a digit, while in the radix-r case, groups of $\log_2(r)$ bits form a digit. For example, the bit-vector

$$x = (110001011101)_2 \tag{3.9}$$

$$= [10]\,[001]\,[011]\,[101])_8 \tag{3.10}$$

$$= ([1100]\,[0101]\,[1101])_H \tag{3.11}$$

Represents octal digit-vector $(6135)_8$ and hexadecimal digit-vector $(C5D)_H$.

**Table 3.1 Representation of the integer "thirty" in different number systems**

| Number system Digit vector | Number system Digit vector |
|---|---|
| Conventional radix-2 system (binary) | $(0011110)_2$ |
| Conventional radix-3 system | $(0001010)_3$ |
| Conventional radix-4 system | $(0000132)_4$ |
| Conventional radix-10 system | $(0000030)_{10}$ |
| Redundant Radix-2 system with digit set$\{-1 = 1,0,1\}$ | (0011110)2<br>$(01000\text{-}10)_2$ |

### *3.2.1.2 Signed Integer Representation*

Signed integer are either positive or negative and can be represented by two most commonly used representation as described below:

- *Sign-Magnitude (SM):* In this type of representation, any integer $x$ is represented as a pair of $x_s$ and $x_m$ i.e.$(x_s, x_m)$, where $x_s$ is the sign of integer and $x_m$ is magnitude of the integer which can be represented in any system. Traditionally sign + represented as binary "0" and – represented as binary "1". For radix-r system (conventional) the signed integer range in magnitude representation is $0 \leqq x_m \leq r^n - 1$ for n digits. In SM zero has more than one representation i.e. $x_s = 0, x_m = 0$ (positive zero) and $x_s = 1, x_m = 0$ (negative zero).
- *True-complement (TC):* In this type of representation, the sign and magnitude have no separation but integer is represented by a positive number. There are two forms true and the complement. True for positive integers and complement for negative integers. TC is general for any radix-r and the two's complement representation is a special case of radix-2 when r = 2. Radix-2 has been used by researchers because of its simplicity, hence it's described in detail. Any signed integer or bit-vector $x$ with $n$ digits can be formulated as follows:

$$x = -x_{n-1} \times 2^{n-1} + \sum_{i=0}^{n-2} x_i \times 2^i \qquad (3.12)$$

Most significant bit $(x_{n-1})$ of $x$ denotes the negative weight of the value when converted from a bit-vector and the bits remained of bit-vectors have positive weights. In example given below the MSB is 1 hence the negative number and other bits are for the magnitude.

$$x = (11010) \rightarrow -16 + 8 + 2 = -6$$

Considering (3.12), it can be noted that:

- Only one representation of zero is there i.e. when all bits are set to zero.
- As $x = -2^{n-1}$ can be represented but $x = 2^{n-1}$ cannot hence, the symmetricity of range.

### *3.2.1.3 Fixed-Point Arithmetic of Two's Complement Numbers*

Formalization of number representation must be done to describe the arithmetic operations in fixed-point format [52]. If number of bits of a signed number $x$ is $\beta$ and $\alpha, \gamma$ are the number of digit of integer and fraction part respectively such that $(\alpha = \beta - \gamma)$ then $FPR_x$ is the tuple $(\beta, \alpha, \gamma)$ that denotes the fixed-point representation of $x$. $x = x_{Int} + x_{fr}$ can be written as $x = (x_{\alpha-1}. x_1 x_0 \cdot x_{-1}. x_{-\gamma+1} x_{-\gamma})$ , such that

$$x = -x_{\alpha-1} \times 2^{\alpha-1} + \sum_{i=-\gamma}^{\alpha-2} x_i \times 2^i \tag{3.13}$$

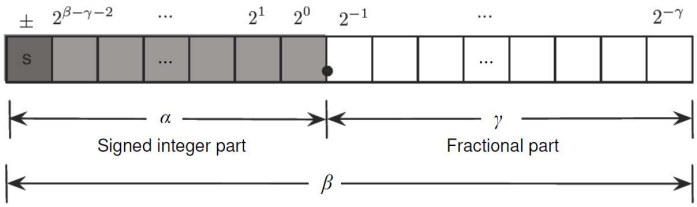



**Figure 3.1 Fixed-point representation of a signed real number in two's complement**

Following steps are taken for conversion of $x$ into FPR.

- $\beta$ is predefine as it denotes the bit-width of data.
- Number of bits that denotes integer part is $\alpha_x = \lfloor \log_2 |x| \rfloor + 2$.
- Calculation fraction part as $\gamma_x = \beta_x - \alpha_x$.
- FPR is given by: $FPRx = (\beta_x, \alpha_x, \gamma_x)$.
- $x$ is represented by the integer $N_x$ as $N_x = [x \times 2^{\gamma_x}]$.
- Thus, $x$ is approximated as $x = N_x \times 2^{-\gamma_x}$.

In any linear system repetitive multiplication followed by addition is a most common practice in complex equations. In fixed-point representation they can be handled as discussed below.

### *3.2.1.4 Multiplication*

Multiplication operation $x$ and $y$ denoted as $z = x \times y$, where $x$ and $y$ have tuple $(\beta_x, \alpha_x, \gamma_x)$ and $(\beta_y, \alpha_y, \gamma_y)$ repectively. The $FPR_z$ can be written as:

$$FPR_z = \left(\beta_x + \beta_y, \alpha_x + \alpha_y, \gamma_x + \gamma_y\right) \tag{3.13}$$

$N_z \leftarrow N_x \times N_y$ denotes the multiplication operation. In (3.13) since $\beta_z = \beta_x + \beta_y$ Hence, there exist double-precision multiplication and the bit-width $(W_{dp})$ of result should be $(\beta_x + \beta_y)$ but generally it's smaller.

Considering $\beta_{op} = W_{dp}$ as bit-width, the expression for $FPR_z$:

$$FPR_z = \beta_{op}, \alpha_x + \alpha_y, \beta_{op} - [\alpha_x + \alpha_y]) \tag{3.14}$$

Hence, the multiplication operation becomes

$$N_z \leftarrow (N_x >> s_x) \times \left(N_y >> s_y\right) \tag{3.15}$$

where, right shift operation on $N_x$ and $N_y$ are $s_x$ and $s_y$ respectively. If $\beta_{op} = (\beta_x + \beta_y)$, then $s_x = s_y = 0$ else $s_x + s_y = (\beta_x + \beta_y) - \beta_{op}$. Figure 3.2 describes the two cases as discussed above, here $S_{cst}$ denotes to the number of truncation bits for $Y$ such that $\beta_{op} = W_{dp}$.

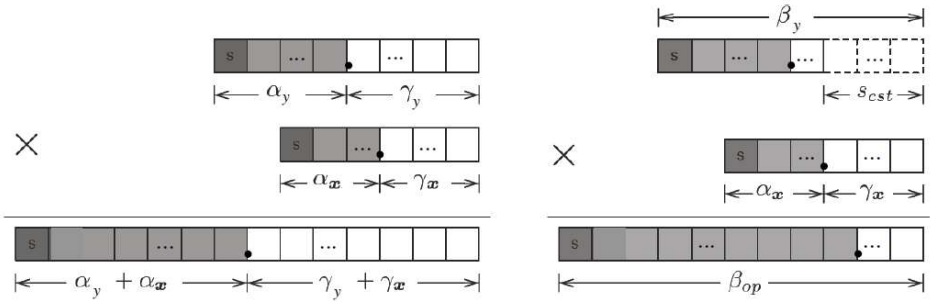



**Figure 3.2 Double (a) and Simple (b) Precision multiplication**

### *3.2.1.5 Addition Operation*

Fundamental requirement of fixed-point addition is to have same position of point. Considering operation $z = x \times y$, with $(\beta_x, \alpha_x, \gamma_x)$ and $(\beta_y, \alpha_y, \gamma_y)$. $FPR_z$ is formulated as:

In Double precision:

$$\begin{cases} \alpha_z = \max(\alpha_x + \alpha_y) + 1 \\ \gamma_z = \max(\gamma_x + \gamma_y) \\ \beta_z = \alpha_z + \gamma_z \end{cases} \tag{3.16}$$

In limited precision where $\beta_{op} = W_{dp}$ is given by:

$$\begin{cases} \alpha_z = \max(\alpha_x + \alpha_y) + 1 \\ \gamma_z = \beta_{op} - \max(\alpha_x + \alpha_y) - 1 \\ \beta_z = \beta_{op} \end{cases} \tag{3.17}$$

Hence, the operation can be realized as

$$N_z \leftarrow (N_x >> s_x) \times (N_y >> s_y) \tag{3.18}$$

with $s_x = \gamma_x - \gamma_z$ and $s_y = \gamma_y - \gamma_z$.

### *3.2.1.6 Overflow Detection*

Overflow happens in two's complement representation when the operands to be added are of same sign and as a result sign is opposite to that of operands. The most significant bit $(x_{n-1})$ is the key for detection of overflow and the expression for detection is given by:

$$AVF = (\overline{x} \times \overline{y} \times z_{n-1}) + (x_{n-1} \times y_{n-1} \times Z_{n-1}) \tag{3.19}$$

## 3.3 Number Representation Systems

For arithmetic operations, number representation systems rely on number representation formats for sophisticated algorithms. In order to reduce the complexity of any arithmetic operation, researchers always try to develop new algorithms. Complexity of computation in any algorithm depends on total number of zeros of input stream. Arithmetic operation are different in each number system because of unique numerical properties of respective system. Some of number systems that are most frequently used by researches in past decade has discussed below.

### 3.3.1 Canonical Sign Digit (CSD)

In 1961, Avizienis [53] developed a new representation technique called CSD i.e. the canonical form of signed digit (SD). It is the redundant form of fixed-radix for which $r = 2$ and can be represented in (3.20).

$$x = \sum_{i=0}^{n-1} x_i \times 2^i \tag{3.20}$$

where, $x_i \in D = \{\overline{1}, 0,1\}$.

The digit $x_i$ called trit and has ternary digit behavior. The addition/subtraction operations can be done without any carry propagation and it is a major advantage of SD. It makes its operations fast. Effect of this advantage is more prominent in large operands addition/subtraction. For example, the integer "7" can be represented more than one form as shown below:

$$(0111) = 4 + 2 + 1 = 7;$$

$$(10\overline{1}1)_2 = 8 - 2 + 1 = 7;$$

$$(1\overline{1}11)_2 = 8 - 4 + 2 + 1 = 7;$$

$$(100\overline{1})_2 = 8 - 1 = 7.$$

Rational numbers can also be represented by more than one form in SD. For example:

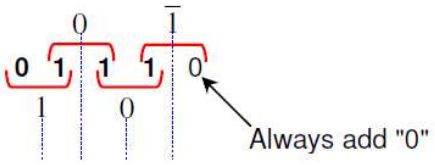
$$\frac{5}{8} = 0.625 = (0.101)_2 = (1.\overline{1}01)_2 = (1.\overline{1}1\overline{1})_2 = (1.0\overline{11})_2.$$

Many conversion methods are given by researchers till now but most common and simplest method is to use Booth Encoding in depicted in Table 3.2.

**Table 3.2 Booth Encoding**

| $Y_j$ | $Y_{j-1}$ | D |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | $\overline{1}$ |
| 1 | 1 | 0 |

For converting any value to SD, steps should be followed as given in Figure 3.3 for 2's complement integer $7 = (0111)_2$.




$$(1\,0\,0\,\bar{1})_2 = 8 - 1 = 7$$

**Figure 3.3 Signed Digit Representation Conversion Steps For Positive Integer "7" Using 2's Compliment Notation**

Another signed digit notation that is famous for its minimum number of non-zero digits is known as minimum signed digit representation. There are many MSD representations in literature that has been used by researchers but a particular MSD representation has been in focus in which two adjacent digits should not be non-zero, i.e. $x_{i+1} \times x_i = 0$ called the canonical signed digit (CSD) representation. Due to uniqueness of each integer number in CSD it is a non-redundant representation. CSD representation of number "7" can be given as $(100\bar{1})_2$ and for "5/8" as (0.101).

Converting signed digit (SD) to canonical signed digit (CSD) can be simply done by the following steps:

- Convert long string of $1's$: $011 \ldots 11 \Rightarrow 100 \ldots 0\bar{1}$;
- Convert long string of $\bar{1}'s$: $0\bar{1}\bar{1} \ldots \bar{1}\bar{1} \Rightarrow \bar{1}00 \ldots 01$;
- Merge adjacent digits of opposite signs: $1\bar{1} \Rightarrow 01$ ; $\bar{1}1 \Rightarrow 0\bar{1}$.

Following are the properties of any $n$ -bit value after converting to CSD.

- Bound in total number of non-zero digits: $(n+1)/2$.
- Asymptotic average value: $(n/3) + (1/9)$.
- Compared to binary representation average number of digits required for representation: $nl2$.

Advantage that attracts researchers towards CSD representation over traditional binary representation, is that the saving in non-zero digit is as much as 33%. This opens the doors of 33% less addition/subtraction operations in a constant multiplication and leads to more compact multiplications implementation of LTI systems [51].

Despite the fact that CSD minimizes the number of nonzero digits for the constant representation, it is far from being optimal. It is possible to decompose the *n*-bit value to further reduce the number of operations. This can be achieved using more complex number systems.

### 3.3.2 Double Base Number System (DBNS)

Drimitov [54] developed Double base number system (DBNS). An integer $x$ can be represented in DBNS using base 2 and 3 as given below:

$$x = \sum_{i,j} d_{i,j} \times 2^i \times 3^j \tag{3.21}$$

where, $d_{i,j} \in D = \{0,1\}$

It can be depicted simply by an example as follows:

Let $x = (10599)_{10}$ then DBNS it can be represented as,

$x = (3^2 \times 2^8) + (3 \times 2^5) + (3^0 \times 2^{13}) + (3^0 \times 2^3) - (3^0 \times 2^0) = (10599)_{10}$.

Some key points should be noted about DBNS. These are as follows:

- DBNS is not a radix system but it is a weighted system.
- Putting $j = 0$ in (3.16) it becomes conventional binary system.
- Canonical form of DBNR that can be used to get minimum number of $(2^i \times 3j)$ is a NP-complex problem.

The discussed points show the fact that DBNS cannot guarantee minimal result of any arithmetic operation. Hence, the author proposed a greedy algorithm called near canonical DBNR (NCDBNR). This was a minimization heuristic in which input integer $x$ and an output of two-integers is $a_i$,such that $\sum a_i = x$. Largest of two-integers are founded by algorithm i.e. $w$, smaller than or equal to $x$, and recursively applies the same for $(x - w)$ until reaching zero. It is shown that the algorithm terminates after $O(\log(x)/\log(\log(x)))$ steps. Procedure of finding near canonical DBNR is very important with respect to basic arithmetic operations for reducing the neighbour non-zero digits. Additionally, ready DBNR is also in literature which uses the reduction of nearby non-zero digits based on some rules but these techniques are beyond scope of this thesis.

### 3.3.3 Radix-$2^r$ Number System

Sam [39] in 1990 developed a $radix - 2^r$ representation. Any n-bit 2's complement number, $x$ can be written in $radix - 2^r$ as given below:

$$x = \sum_{j=0}^{\left(\frac{n}{r}\right)} (x_{rj-1} + 2^0 x_{rj} 1 + 2^1 x_{rj+1} + 2^2 x_{rj+2} + \cdots + 2^{r-2} x_{rj+r-} - 2^{r-1} x_{rj+r-}) \times 2^{rj} \tag{3.22}$$

$$x = \sum_{j=0}^{\left(\frac{n}{r}\right)-1} Q_j \times 2^{rj} \tag{3.23}$$

where, $Q_j \in D = \{-2^{r-1}, -2^{r-1}+1, \cdots, -1,0,1, \cdots, 2^{r-1}-1, -2^{r-1}\}$, $x_{-1} = 0$ and $r \in N^*$.

It was assumed that r is a divider of N for the purpose of simplicity and with general meaning.

- $x$ is an integer whose 2's complement representation is split into $n/r$ number of slices that are also in 2's complement, represented as $(Q_j)$
- Two's complement slices $(Q_j)$ have length $r + 1$ bit.
- Each pair of $Q$ slice have one bit common in continuity.
- For $r = 1$ the radix-$2^r$ becomes SD representation.

Some important features of $Q_j$ are as follows:

- The sign of the term $Q_j$ is given by the $x_{rj+r-1}$ bit.
- The magnitude of $Q_j$ is $|Q_j| = 2^{k_j} \times m_j$, with $k_j \in \{0,1,2, \dots, r-1\}$ and
- $m_j \in OM(2^r) \cup \{0\}$.Set of odd positive digits in radix-2r recording is $OM$ and it can be defined as $OM(2^r) = \{13,5, \dots, 2^{r-1} - 1\}$ with $|OM(2^r)| = 2^{r-2}$
- $|Q_j| = 0$ corresponds to $m_j = 0$.

Careful examination of the above points and the (3.23) shows that $x$ can be written as given below:

$$x = \sum_{j=0}^{\left(\frac{n}{r}\right)-1} (-1)^{x_{rj+r-1}} - 1 \times m_j \times 2^{rj+k_j} \qquad (3.24)$$

Procedure for converting an integer into radix-$2^r$ using (3.19) can be demonstrated through following example.

Given integer number $x$ is 10599, steps to convert it into Radix-$2^r$ are given below:

- Represent $x$ in 2's complement form i.e. 0101001011001111.
- Size of constant $x$ in 2's complement form is $n = 15$.
- $r$ value is chosen, in this case let's take $r = 4$.
- If $n$ is not a multiple of $r$ then extend the sign-bit and make $n = 16$.
- Apply above two equation for $x$ i.e. $x = \sum_{j=0}^{3} Q_j \times 24j$ , and $x = \sum_{j=0}^{3}(-1)^{c_{4j+3}} \times m_j \times 2^{4j+k_j}$.
- These equations converges to four $Q_j$ terms.
- The unknown values $c_{4j+3}$, $m_j$, and $k_j$ are to be determined using respective radix-$2^r$ table. In this is case Table 3.3 i.e. $radix - 2^4$ look-up table for determination of $Q_j$ terms.
- Table 3.4 depicts the values of four $Q_j$ with their corresponding triplets$(c_{4j+3}, m_j, k_j)$.

**Table 3.3 Radix-$2^4$ Look-Up Table [42]**

| $Q_j$ | | | | | $(m_j, k_j)$ | |
|---|---|---|---|---|---|---|
| $x_{4j+3}$ | $x_{4j+2}$ | $x_{4j+1}$ | $x_{4j}$ | $x_{4j-1}$ | $m_j$ | $k_j$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 3 | 0 |
| 0 | 0 | 1 | 1 | 0 | 3 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 2 |
| 0 | 1 | 0 | 0 | 0 | 1 | 2 |
| 0 | 1 | 0 | 0 | 1 | 5 | 0 |
| 0 | 1 | 0 | 1 | 0 | 5 | 0 |
| 0 | 1 | 0 | 1 | 1 | 3 | 1 |
| 0 | 1 | 1 | 0 | 0 | 3 | 1 |
| 0 | 1 | 1 | 0 | 1 | 7 | 0 |
| 0 | 1 | 1 | 1 | 0 | 7 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 3 |
| 1 | 0 | 0 | 0 | 0 | 1 | 3 |
| 1 | 0 | 0 | 0 | 1 | 7 | 0 |
| 1 | 0 | 0 | 1 | 0 | 7 | 0 |
| 1 | 0 | 0 | 1 | 1 | 3 | 1 |
| 1 | 0 | 1 | 0 | 0 | 3 | 1 |
| 1 | 0 | 1 | 0 | 1 | 5 | 0 |
| 1 | 0 | 1 | 1 | 0 | 5 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 2 |
| 1 | 1 | 0 | 0 | 0 | 1 | 2 |
| 1 | 1 | 0 | 0 | 1 | 3 | 0 |
| 1 | 1 | 0 | 0 | 1 | 3 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 |

**Table 3.4 Four $Q_j$ with their Corresponding Triplets**

| $Q_j$ | $c_{4j+3}$ | $m_j$ | $k_j$ |
|---|---|---|---|
| $Q_0$ | 0 | 7 | 0 |
| $Q_1$ | 0 | 3 | 1 |
| $Q_2$ | 1 | 7 | 0 |
| $Q_3$ | 0 | 3 | 0 |

Using above two tables x can be written as:

$$x = Q_3 \times 2^{12} + Q_2 \times 2^8 + Q_1 \times 2^4 + Q_0 \tag{3.25}$$

i.e.

$$x = (3 \times 2^{12}) - (7 \times 2^8) + (3 \times 2^5) + (7 \times 2^0) = (10599)_{10} \tag{3.26}$$

Hence the given integer is split into as shown in Figure 3.4.

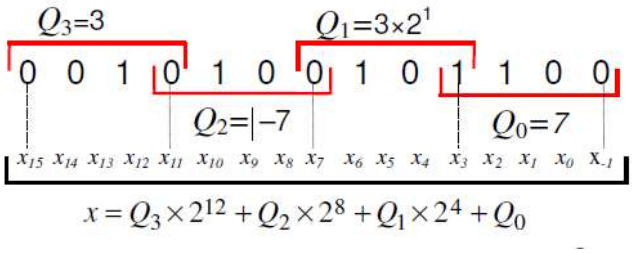



**Figure 3.4 Partitioning of constant $(10599)_{10}$ in Radix-$2^4$ [41]**

## 3.4 Comparison of the Number Systems

The main features of different number systems has been studied. To compare a given integer in different number system can be done by expressing $x = (10599)_{10}$ in studied number representation. Comment on hardware after implementing each of number system is not possible in this stage but an approximate idea can be achieved by carefully observing Table 3.5. Typically, more speed is achieved when number of digits are low and that in turns takes less hardware resources.

**Table 3.5 Comparison of the Number Systems**

| Number System | Arithmetic expression | Number of Digits |
|---|---|---|
| Radix-$2^r$ | $x = (3\times2^{12})-(7\times2^8)+(3\times2^5)+(7\times2^0)$ | 4+2=6 |
| CSD | $x = 2^{13}+2^{11}+2^9-2^7-2^5+2^3-2^0$ | 7 |
| Binary | $x = 2^{13}+2^{11}+2^8+2^6+2^5+2^2+2^1+2^0$ | 8 |
| SD | $x =2^{13}+2^{12}-2^{11}+2^9-2^8+2^7-2^5+2^3-2^0$ | 9 |

# CHAPTER 4

# MCM ALGORITHM AND POWER REDUCTION TECHNIQUE

This chapter discusses multiple constant multiplication algorithm used in thesis to implement the multiplierless FIR digital filter and the problems surrounds it. The discussion accompanied with complexities like upper-bound, adder depth and average has been discussed in this chapter. Further, power consumption of multiplierless FIR filter has been discussed. Switching activity reduction technique which is used in this thesis is also explained for overall power reduction of filter.

## 4.1 Formulation of LTI Systems

Any linear time invariant system have two properties called additivity and homogeneity. These properties together can be written as:

$$f(c_1 \times x_1 + c_2 \times x_2 + \cdots + c_n \times x_n) = c_1 \times f(x_1) + c_2 \times f(x_2) + \cdots + c_n \times f(x_n) \quad (4.1)$$

where, $x_n$ are input vectors and $c_n$ are scalars.

A Linear linear-time-invariant (LTI) systems can be written using (4.1) as follows:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_m \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ C_{m1} & C_{m2} & \dots & C_{mn} \end{bmatrix} \times \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} \quad (4.2)$$

where, C is an $m \times n$ transformation matrix representing $(ij)$ constant as $C_{ij}$. X and Y are input vector and output vector respectively. The output $Y_i$ is the multiplication of $ith$ row of the matrix $C$ with $n$ input samples of $X$.

$$Y_j = \sum_{j=1}^{n} C_{ij} \times X_j \quad (4.3)$$

The formulation in (4.3) can be used in FIR filter where $C_{ij}$ represent the coefficients and $X_j$ is an input sampled signal.

## 4.2 Single-Constant Multiplication (SCM)

In 1970s, a constant multiplication problem occurred while implementing multiplication in software. Solving the problem of constant multiplication would leads to the reduction of execution time because the multiply instructions in software take more clock cycles. Thus, efficient hardware implementation needs to propose algorithms that runs in software and the solution produce by these algorithms should implement constant multiplication efficiently in hardware. In SCM, the focus is multiplication of one variable with one constant

In (4.2) multiplication of each variable X with each of $C_{ij}$ constant i.e $C_{ij} \times X_j$ must be multiplierless for efficient hardware implementation which can be done by

addition/subtraction operations and shifts. Only the adder/subtractor has importance because the shift operation needs only rewiring. Hence, the smart operation of any SCM into additions and shifts leads to benefits of area, power, throughput, and delay of operation. Because the SCM has a very large solution space, the designer has to minimize the number of additions using some kind of heuristics. It has a computational complexity that is not known and figured as a NP-Hard [37] problem.

Illustrating a special case of SCM i.e. $45 \times V_j$, where $V_j$ is a variable to be multiplied with constant. Many solutions can be possible for this multiplication but only four are shown here for understanding.

- The basic method of SCM is shown in (4.4). Binary representation of constant "45" is written and then shifted according to position. Finally, summation is done of resulted shifted values. The whole operation require three shifts and three adders and the number of additions are one less than the number of non-zero digits.

$$45 \times V_j = (101101)_2 \times V_j = V_j \times 2^5 + V_j \times 2^3 + V_j \times 2^2 + V_j \tag{4.4}$$

- In second method, the constant is decomposed into subtraction of 1's complement of constant from constant of same length having all 1's. The term $(V_j \times 2^6 - V_j)$ represents 63 and other two terms $V_j \times 2^4$ and $V_j \times 2$ represents 16 and 2 i.e. 16+2=18.

$$\begin{aligned} 45 \times V_j &= (63 - 18) \times V_j \\ &= [(111111)_2 - (010010)_2] \times V_j \\ &= \left(V_j \times 2^6 - V_j\right) - V_j \times 2^4 - V_j \times 2 \end{aligned} \tag{4.5}$$

- In this method constant is encoded with least number of non-zero digits using CSD representation. But in this case CSD is not providing any reductions in number of additions.

$$45 \times V_j = (10\bar{1}0\bar{1}01)_2 \times V_j = V_j \times 2^6 - V_j \times 2^4 - V_j \times 2^2 + V_j \tag{4.6}$$

- The fourth method provides benefit over others resulting in minimum number of adders.

$$45 \times V_j = G \times 2^2 + G \text{ with } G = V_j \times 2^3 + V_j \tag{4.7}$$

The sharing of $U = 9 \times X_j$ leads to minimum number of additions as shown in Figure 4.1.

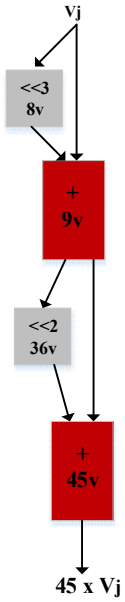



**Figure 4.1 Minimum Number of Additions for $45 \times V_j$**

In Figure 4.1 the grey and red nodes indicates shifting and addition operation respectively. The sign " $\ll x$" is for shift of $x$ positions.

Hence, the objective SCM Heuristic is to provide the predictability of area and speed. Area prediction can be judge by number of additions and speed prediction by critical path. It also results in optimal solutions having less calculation time.

## 4.3 Multiple-Constant Multiplication (MCM)

MCM problem is superset of SCM where single variable is multiplied with more than one constant. Considering (4.2) the variable $X_j$ is multiplied with entire column $j$ i.e. set of coefficients $C_{1j}, C_{2j}, C_{3j}, \cdots, C_{mj}$. MCM problem is also defined a NP-hard problem. The simple method for solving MCM problem is optimization of each of single constant independently. MCM can lead to lesser number of operations than the SCM by the method of sharing the common expressions required by each constant term.

Illustrating an example of MCM problem in which two constants “81” and “23” are to be multiplied with a single variable $V_j$. In Figure 4.2, the two optimized multiplications of $23 \times V_j$ and $81 \times V_j$ are performed individually. The grey and red nodes indicates shifting and addition operation respectively. As a result, for solving this MCM problem there is a requirement of a total to four adders.

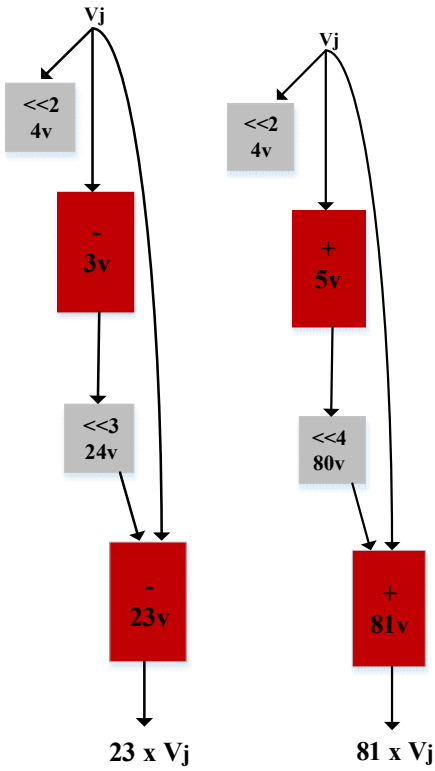



**Figure 4.2 Optimized Multiplication of each variable independently**

Obtimization of these two varibles can be done simultenoulsy by sharing a common intermediate computation $9 \times V_j$ to reduce the number of adders to three i.e. reduction of one addition is achieved as shown in Figure 4.3.

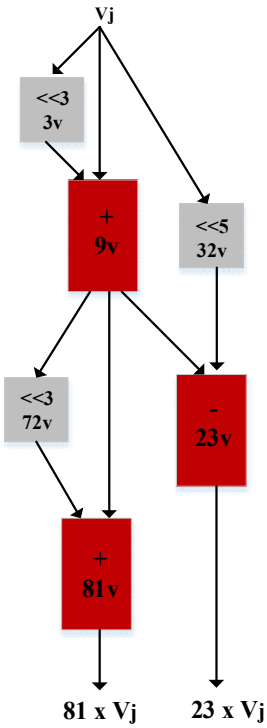



**Figure 4.3 Optimization of Variables Simultaneously**

From Figure 4.3 we can conclude that optimization of (4.2) can be achieved better in case of MCM than the SCM. Here, with increase in critical path computational time increases but this cannot be true when there are many constants.

### 4.4 Metrics of SCM/MCM Algorithms

Efficient hardware implementation of any FIR digital filter means low power, high speed and low area utilization. These all factors can be achieved with SCM/MCM algorithms. The amount of silicon required can be judged by means of number of additions/subtractions used to implement the logic circuit. Number of additions/subtractions is most frequently used metric in the area of reduction of amount of silicon [37]. Experimental illustrations in [55] shows that more accurate metric can increase the run time. Also, amount of area covered in silicon is dependent on implemented logic circuit fabric and efficiency of computer-aided design (CAD) tools. Hence, need of good metric that drives less runtime as well as independent on technology must be a solution. In any hardware, adders and subtractors consume same amount logic resource. Hence, in defining the metrics both are considers as "adders".

Definitions of metrics are given below [43] for N-bit constant $C_i$**:**

*Definition 1:* Upper-bound (Upb): For each N - bit constant $C_i$, corresponds $A_i$ additions for the implementation of $C_i \times X$. $Upb = \max(A_i)$ .

*Definition 2:* Adder-Depth (Ath): Let $D_i$ be the number of adders that we pass through along any path i from the input to any of the outputs in the constant multiplication logic circuit. $Ath = \max(D_i)$ .

*Definition 3:* Average(AVg): For each N - bit constant $C_i$, corresponds $A_i$ additions for the implementation of $C_i \times X$. $Avg = \sum_{i=1}^{m} A_i / m$, where m is the total number of $C_i$ constants.

### 4.5 Radix-$2^r$ SCM Algorithm

In section 3.3.3 Radix-$2^r$ arithmetic has been explained and that can be extended to provide explanation of Radix-$2^r$ algorithm for single constant multiplication. In radix-$2^r$ any N-bit constant C can be formulated as follows:

$$C = \sum_{j=0}^{\left(\frac{n}{r}\right)-1} \left( c_{rj-1} + 2^{o} c_{rj} \quad + 2^{1} c_{rj+1} + 2^{2} c_{rj+2} + \quad \cdots \quad + 2^{r-2} c_{rj+r-2} - 2^{r-1} c_{rj+r-1} \right) \times 2^{rj} \quad (4.8)$$

$$C = \sum_{j=0}^{\left(\frac{n}{r}\right)-1} Q_j \times 2^{rj} \quad (4.9)$$

where $r \in N^*$ and $c_{-1} = 0$. Generally, N is taken as multiple of r for purpose for simplicity. In (4.8) the C is represented in 2's complement form and parted into a total of $N/r$ slices ($Q_j$). As each slice goes from $2^0$ to $2^{r-1}$ hence the bit length of each slice is taken as r. From the (4.9) it can be seen that there is a requirement of additional bit ($c_{rj-1}$) in $Q_j$ which is the MSB of previous digit $Q_{j-1}$.

$$-2^{r-1}c_{rj+r-1} \times 2^{rj} + c_{rj+r-} \times 2^{r(j+1)} = c_{rj+r-1} \times 2^{rj+r-1} \tag{4.10}$$

Formulation in (4.9) transform the conventional radix-$2^r$ into signed digit radix-$2^r$. A digit-set can be defined for (4.8) as:

$$Q_j \in DS(2^r) = \{-2^{r-1}, -2^{r-1}+1, \dots, -1, 0, 1\ 2^{r-1}-1, 2^{r-1}\} \tag{4.11}$$

Now, the multiplication of constant C with input X can be written as:

$$C \times X = \sum_{j=0}^{\frac{N}{r}-1} X \times Q_j \times 2^{rj} \tag{4.12}$$

The magnitude of each $Q_j$ term can be is given by $|Q_j| = 2^{k_j} \times m_j$ and sign is given by $c_{rj+r-1}$ bit, with $k_j \in \{0, 1 2\ r-1\}$ and $m_j \in OM(2^r) \cup \{0\}$. Where, the $OM(2^r)$ is the set of positive odd digits in recording i.e. $OM(2^r) = \{1, 3, 5, \dots, 2^{r-1}-1\}$ such that $|OM(2^r)| = 2^{r-2}$.
Finally, the multiplication can be rewritten as follows:

$$C \times X = \sum_{j=0}^{\left(\frac{N}{r}\right)-1} (-1)^{C_{rj+r-1}} - 1 \times (m_j \times X) \times 2^{rj+k_j} \tag{4.13}$$

The number of different values of $m_j$ corresponds to total number of partial-products (PP) induced by the coding process of $Q_j$ terms. First, the generation of PP is done then they are hardwired shifted by $rj + k_j$ position and at last depending on the sign bit of $c_{rj+r-}$ of $Q_j$ term the result is negated $i.e. (-1)^{c_{rj+r-1}}$.
Figure 4.4 defines the metric equation for radix-$2^r$ SCM having non-negative *n*-bit constant and Figure 4.4 shows the partitioning of N- bit constant into slices where *W* is lambert function and ⌈ ⌉ is ceiling function .

| Metrics | Equations |
|---|---|
| Adder cost | $Upb(r)=\left\lceil \frac{N+1}{r} \right\rceil + 2^{r-2} - 2$ with $r=r_1$ or $r=r_2$ |
| Adder depth | $Ath(r)=\left\lceil \frac{N+1}{r} \right\rceil + r - 3$ with $r=r_1$ or $r=r_2$ |
| Average | $-1+Avg_{pp}+Avg_{om} \le Avg(r) \le -2+Avg_{pp}+2^{r-2}$ with $Avg_{pp}=\left(1-2^{-r}\right)\times\left\lceil \frac{N+1}{r} \right\rceil$, $Avg_{om}=\sum_{j=0}^{\left\lceil \frac{N+1}{r} \right\rceil -1}\left\{\sum_{k=1}^{2^{r-2}-1}P(m_{jk})\times\left[1-P(m_{jk})\right]^{j}\right\}$, $P(m_{jk})=\frac{\log_2\left\lceil \frac{2^{r-1}}{2\times k+1} \right\rceil}{2^{r-1}}$, and $r=r_1$ or $r=r_2$ |
| $r_1=2\cdot W\left[\sqrt{(N+1)\cdot log(2)}\right]/log(2)$ | $r_2=W\left[4.(N+1).\ log(2)\right]/log(2)$ |

**Figure 4.4 Metrics for Radix-2$^r$ SCM Algorithm [42]**

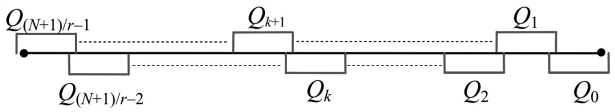



**Figure 4.5 Slice partitioning of N-bit constants in Radix-2$^r$ SCM [42]**

## 4.6 Radix-2$^r$ MCM Algorithm

Radix-2$^r$ MCM Algorithm is the extension of radix-2$^r$ SCM algorithm discussed in Section 4.5. In this algorithm, a set of constants of bit-size N are multiplied with single input variable X. The non-trivial PP set are shared among the M constants. i.e. $C_1$,$C_2$,$C_3$,….,$C_M$. The Figure 4.6 defines the metric equation for radix-2$^r$ SCM having nonnegative *n*-bit constant.

.

| Metrics | Equations |
|---|---|
| Adder cost | $Upb(r')=\sum_{i=0}^{M-1}\left\lceil \frac{N_i+1}{r'} \right\rceil + 2^{r'-2} - 1 - M$ with $r'=r_1$ or $r'=r_2$ |
| Adder depth | $Ath(r')=\left\lceil \frac{\max(N_i)+1}{r'} \right\rceil + r' - 3$ with $i=0..M-1$, $r'=r_1$ or $r'=r_2$ |
| Average | $-M+Avg_{pp}+Avg_{om} \le Avg(r') \le -M-1+Avg_{pp}+2^{r'-2}$ with $Avg_{pp}=\left(1-2^{-r'}\right)\times\sum_{i=0}^{M-1}\left\lceil \frac{N_i+1}{r'} \right\rceil$, $Avg_{om}=\sum_{j=0}^{\sum_{i=0}^{M-1}\left\lceil \frac{N_i+1}{r'} \right\rceil -1}\left\{\sum_{k=1}^{2^{r'-2}-1}P(m_{jk})\times\left[1-P(m_{jk})\right]^{j}\right\}$, $P(m_{jk})=\frac{\log_2\left\lceil \frac{2^{r'-1}}{2\times k+1} \right\rceil}{2^{r'-1}}$, and $r'=r_1$ or $r'=r_2$ |
| $r_1=2\cdot W\left[\sqrt{\left(\sum_{i=0}^{M-1}(N_i+1)\right)\cdot log(2)}\right]/log(2)$ | $r_2=W\left[4.\left(\sum_{i=0}^{M-1}(N_i+1)\right).\ log(2)\right]/log(2)$ |

**Figure 4.6 Metrics for Radix-2$^r$ MCM Algorithm [42]**

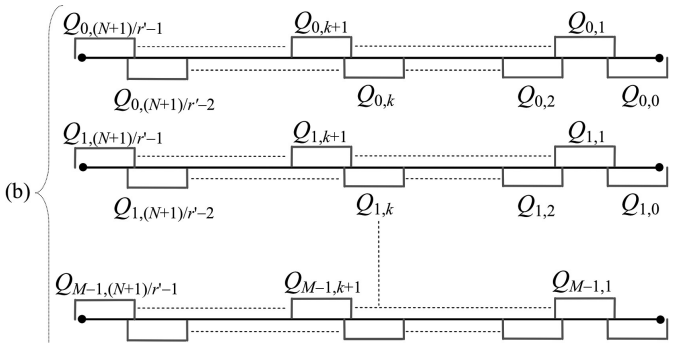



**Figure 4.7 Slice partitioning of N-bit constants in Radix-2$^r$ MCM [42]**

Figure 4.7 shows the partitioning of N- bit constant into slices where $W$ is lambert function and ⌈ ⌉ is ceiling function. The adder depth, upper-bound and average are lowest bounds known so far for multiplication by a constant [42]. A FIR digital filter can be generated with shortest critical-path by selecting "$r$" value for metric adder depth. In this thesis, minimum adder depth is concerned for low power FIR digital filter design. Finally, radix-2$^r$ arithmetic and its MCM algorithm is a powerful mathematical tool that can be studied further for exploring addition-cost complexities.

## 4.7 Power Consumption for Mutliplierless FIR Filters

The hardware cost of any FIR filter depends on both multiplier block adders (MBA) and structural adders i.e. total number of adders as shown in Figure 4.8. But number of adders cannot alone investigate the power consumption of multiplierless FIR filter. The more serious power consumption which is the dynamic power consumption. The main reason behind the dynamic power consumption is the switching activity of the circuit or the circuit transitions. The dynamic power consumed any circuit is given by [56]:

$$P_{dynamic} = \frac{1}{2}\alpha C_L V_{dd}^2 f \tag{4.14}$$

where $\alpha$ is switching activity, $C_L$ is the load capacitance, $V_{dd}$ is supply voltage and $f$ is the clock frequency. In (4.14) it is clear that dynamic power consumption is very much related to the total number of transitions in circuit i.e. switching activity $\alpha$. The total number of transitions in FIR digital filter is directly proportional to the adder depth [57].

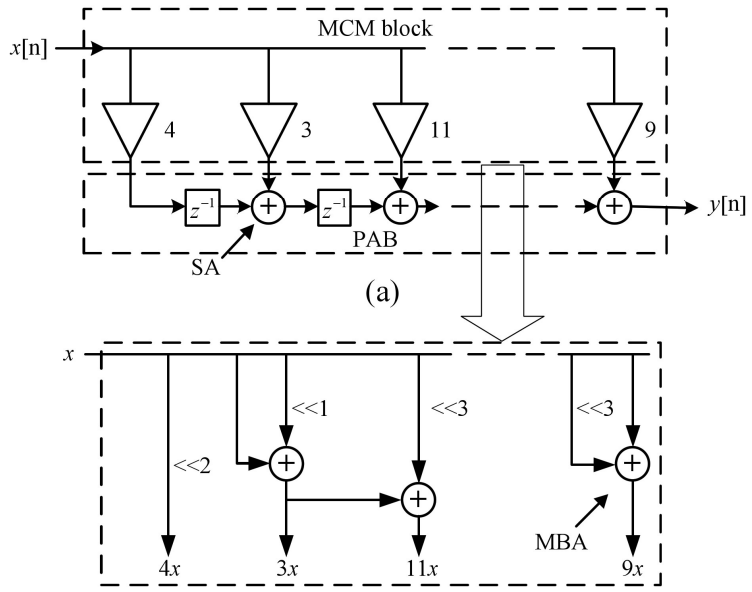
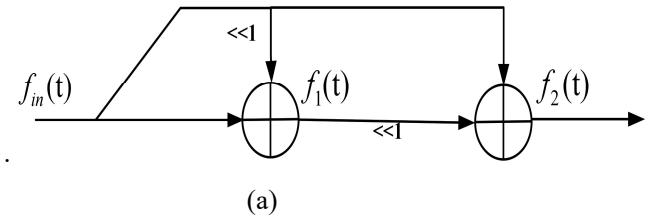


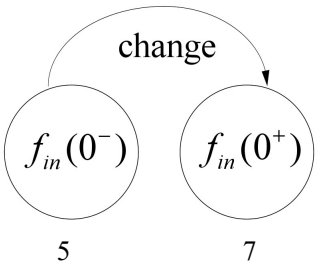

**Figure 4.8 Structural adders (SA) and Multiplier block adders (MBA) in a multiplierless FIR filter [47]**

### 4.7.1 Understanding Adder Depth

The number of series adders through which the primary input signal reaches to the output of that adder (without any delay) is known as the adder depth. The illustration of adder depth is shown in Figure 4.8 (a). For the generation of $f_1(t)$, the primary input $f_{in}(t)$ passes through one adder and therefore the calculated adder depth for output signal is one. Similarly, for $f_2(t)$ the adder depth for output signal is two due to presence of one more adder. It is clear in Figure 2.8(a) that $f_1(n) = 3f_{in}(n)$ and $f_2(n) = 2f_1(n) + f_{in}(n)$.




(a)



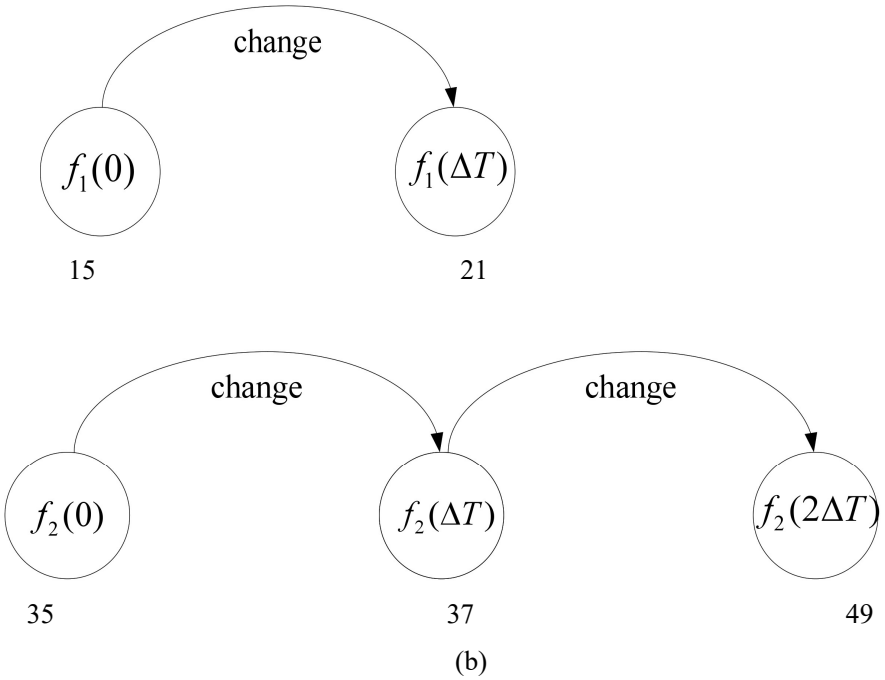



**Figure 4.8 (a) Illustration of Adder Depth (b) Change in output as primary input changes [45]**

More changes in adder leads to more transitions in the circuit and hence results in high switching activity i.e. high power consumption. Figure 4.8 depicts the higher adder depth results in higher power consumption. Assuming that time taken by circuit to process an addition is $\Delta T$ thus the output $f_2$ is stable at $2 \triangle T$. For timing, assumption is taken that at time $0^-$, output $f_1(0)$ and $f_2(0)$ are stable with values 15 and 35, respectively. Initially input is taken as 5.

Figure 4.7 (b) depicts that at time $\triangle T$, $f_1$ changes from 15 to 21 and at time 0, $f_1(0^+)$ changes to 7. However, a incorrect output 37 is generated at time $\triangle T$. The correct output ($f_2$) value 49 is generated at $2 \triangle T$. Finally, it can be seen from Figure 4.8 (b) that the adder in the depth 1 only changes once when the primary input changes, while the adder in the adder depth 2 to need to change two times in order to generate the correct output.

Hence, for reducing power consumption of FIR filter the adder depth must be reduced as much as possible.

## 4.8 Power Reduction Technique

Logic depth of structural adders (SAs) are much larger than that of multiplier block adders (MBAs), which results in higher power consumption. Logical depth of SAs Becomes much more important in higher order filters. Hence, the reduction of logical depth at product accumulation block (PAB) can lead to reduction in overall power consumption of FIR filter.

Dynamic power of any digital circuit is mainly due to switching power. The switching power of a node is given by:

$$P_{\text{switching}} = \alpha C_L V_{DD}^2 f \tag{4.15}$$

and the total switching power of digital circuit is given by:

$$P_{\text{switching\_total}} = \sum_{i=1}^{M} P_{\text{switchin i}} \tag{4.16}$$

where, $\alpha$ is switching activity and $M$ is total number of nodes that are determined by filter implementation. Other parameters $C_L$ (load capacitance), $V_{DD}$ (supply voltage) and $f$ (clock frequency) are fixed by circuit specification and foundry process.

For any circuit higher adder depth have a tendency of higher switching activity. This is because of the fact that glitches caused by multiple delay paths passes on through the adder stages. These glitches can increase switching activity up to 70% in a digital circuit but do not generate any circuit error [58]. According to the analysis done in [47], most of the logic complexity is produced by the SAs and also have higher switching activity when filters are implemented using transposed direct form. Hence, total power is dominated by the power consumption of PABs.

### 4.8.1 Implementation of Product Accumulation Block for Low Powers

By registering the output signals coming from MBA i.e. input to SAs, the increase in switching activity caused by glitch propagation can be reduced. The propagation of glitches can be terminated because the overhead registers fill in the signals only at definite clock edges.

There are two possibilities to insert a register at each output of multiple constant multiplication block. These are mentioned below.

- Registering each of output signal from MCM block:
  In multipliers FIR filter design only fundamentals i.e. positive and odd coefficients, are implemented in the MCM block because even coefficients can be generated by shifting and addition of already available fundamentals. Hence the number of registers for storage of fundamentals are significantly less than the total number of coefficients of FIR filter. As shown in Figure 4.9 only fundamentals are registered at the input of SAs.

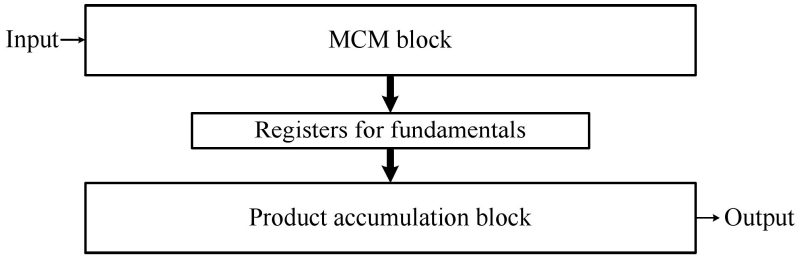



**Figure 4.9 Registering all the MCM outputs [47]**

- Registering heavily reused signals from MCM blocks:
  Outputs that are mostly used for the generation of coefficient-input product are registered and pipeline stages inside the MCM block or at the input of the filter are balanced for the reduction of overhead registers. In Figure 4.10 depicts the power reduction technique that results into negligible area overhead.

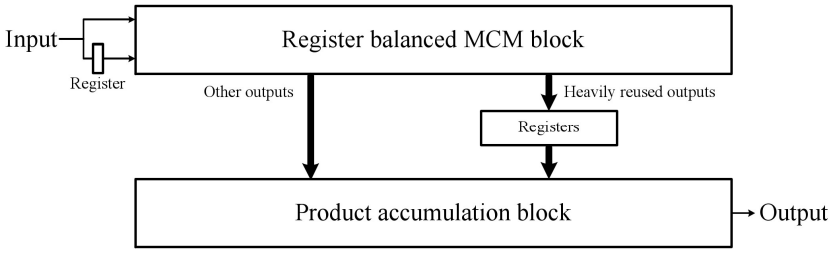



**Figure 4.10 Registering significantly used MCM outputs [47]**

In these two techniques of power reduction only number of reuses has been considered for registration of MCM output. But other parameters such as logical depth, hardware complexity and filter length of the MCM block also affects the power consumption of FIR filter. Hence, logic approach that includes all these parameters can be developed for registering outputs of MCM blocks.

# CHAPTER 5

# SIMULATION AND IMPLEMENTATION RESULTS

This chapter describes the simulation results of five benchmark filters. The set of constant coefficients of all five FIR digital filters has been taken from [59]. Filters are realized using direct transposed form structure for high speed. Radix-$2^r$ arithmetic is used for fixed point representation of constant coefficients of FIR filters. Radix-$2^r$ multiple constant multiplication algorithm is used for implementation of multiplierless FIR digital filters. In Section 5.1, coefficients of all five benchmark filters are recorded using Radix-$2^r$ algorithm. The Vivado 2016.4 has been used to write the Verilog code for the reconfigurable design of filters. The Artix-7 FPGA xc7a35ticpg236-1L is used. Section 5.2 describes the simulation and implementation results of all the filters. Power analysis of filters is done at a frequency of 200 MHz by using the Synopsys gate level tool Design Vision along with the comparison between different filters using low power technique is mentioned in Section 5.3.

The filter specifications are listed in Table 5.1. The specifications include the pass-band edge ($\boldsymbol{\omega_p}$), stop-band edge ($\boldsymbol{\omega_s}$), pass-band attenuation ($\boldsymbol{\delta_p}$) and stop-band attenuations ($\boldsymbol{\delta_s}$).

**Table 5.1 Specifications of five Benchmark Filters**

| Filters | Filter order | $\omega_p$ | $\omega_s$ | $\delta_p$ | $\delta_s$ |
|---|---|---|---|---|---|
| G1 | 15 | $0.2\pi$ | $0.5\pi$ | 0.01 | 0.01 |
| Y1 | 30 | $0.3\pi$ | $0.5\pi$ | 0.00316 | 0.00316 |
| Y2 | 33 | $0.3\pi$ | $0.5\pi$ | 0.001 | 0.001 |
| A1 | 58 | $0.125\pi$ | $0.225\pi$ | 0.01 | 0.001 |
| L2 | 62 | $0.2\pi$ | $0.28\pi$ | 0.028 | 0.001 |

## 5.1 Recording of filter coefficients

The Recording is done using Radix-$2^r$ algorithm for minimum adder depth. Minimum adder depth leads to less power consumption. Table 5.2 - 5.6 shows the Radix-$2^r$ coefficient recording for filters G1, Y1, Y2, A1 and L2 respectively along with the optimum value of r for lowest adder depth. The set of odd and positive fundamentals needed for recording of coefficients is also mentioned. The symbol "<<" denotes the shifting operation.

*Parameters for Table 5.2:*

Optimal value of r (Radix-$2^r$) for adder-depth is 4.

Upper-bound in Radix-$2^r$ (adders connected in series) is 3.

Odd positive fundamentals: 5 = +1<<2 +1<<0; 7 = +1<<3 -1<<0; 3 = +1<<1 +1<<0.

**Table 5.2 Radix-2$^r$ recoding of G1 filter coefficients**

| Coefficients | Radix-2$^r$ recoding |
|---|---|
| 235 | -5<<0 -1<<4 +1<<8 |
| 732 | +7<<2 -5<<6 +1<<10 |
| 646 | +3<<1 +5<<7 |
| 28 | +7<<2 |
| -191 | +1<<0 -3<<6 |
| 31 | -1<<0 +1<<5 |
| 33 | +1<<0 +1<<5 |
| -10 | -5<<1 |

*Parameters for Table 5.3:*

Optimal value of r (Radix-2 $^r$) for adder-depth is 6

Upper-bound in Radix-2 $^r$ (adders connected in tree structure) is 3

Odd positive fundamentals:

23 = +15<<0 +1<<3; 13 = +15<<0 -1<<1; 3 = +1<<1 +1<<0; 15 = +1<<4 -1<<0

**Table 5.3 Radix-2$^r$ recoding of Y1 filter coefficients**

| Coefficients | Radix-2$^r$ recoding |
|---|---|
| -2 | -1<<1 |
| -8 | -1<<3 |
| 0 | 0 |
| 17 | +1<<4+1<<0 |
| 16 | +1<<4 |
| -21 | -17<<0 -1<<2 |
| -46 | -23<<1 |
| 0 | 0 |
| 84 | +17<<2 +1<<4 |
| 68 | +17<<2 |
| -92 | -23<<2 |
| -205 | -13<<0 -3<<6 |
| 0 | 0 |
| 527 | +15<<0 +1<<9 |
| 994 | -15<<1 +1<<10 |

*Parameters for Table 5.4:*

Optimal value of r (Radix-$2^r$) for adder-depth is 5

Upper-bound in Radix-$2^r$ (adders connected in series) is 3

Odd positive fundamentals:

3 = +1<<1 +1<<0; 11 = +9<<0 +1<<1; 5 = +1<<2 +1<<0; 15 = +1<<4 -1<<0

**Table 5.4 Radix-$2^r$ recoding of Y2 filter coefficients**

| Coefficients | Radix-$2^r$ recoding |
|---|---|
| 6 | +3<<1 |
| 6 | +3<<1 |
| 9 | -1<<3 -1<<0 |
| -22 | -11<<1 |
| 0 | 0 |
| 43 | +11<<0 +1<<5 |
| 36 | +1<<5 +1<<2 |
| -48 | -3<<4 |
| -101 | -5<<0 -3<<5 |
| 0 | 0 |
| 171 | +11<<0 +5<<5 |
| 137 | +9<<0 +1<<7 |
| -182 | +5<<1 -3<<6 |
| -404 | -5<<2 -3<<7 |
| 137 | +9<<0 +1<<7 |
| 1020 | -1<<2 +1<<10 |
| 1920 | +15<<7 |

*Parameters for Table 5.5:*

Upper-bound in Radix-$2^r$ (adders connected in series) is 3.

Optimal value of r (Radix-$2^r$) for adder-depth is 4.

Odd positive fundamentals:

7 = +1<<3 -1<<0; 5 = +1<<2 +1<<0

**Table 5.5 Radix-$2^r$ recoding of A1 filter coefficients**

| Coefficients | Radix-$2^r$ recoding |
|---|---|
| 4 | +1<<2 |
| 6 | +1<<2 +1<<1 |
| 8 | +1<<3 |
| 8 | +1<<3 |
| 4 | +1<<2 |
| -3 | -1<<1 -1<<0 |
| -14 | -7<<1 |
| -24 | -1<<4 -1<<3 |
| -32 | -1<<5 |
| -32 | -1<<5 |
| -21 | -5<<0 -1<<4 |
| 0 | 0 |
| 28 | +7<<2 |
| 56 | +7<<3 |
| 75 | -5<<0 +5<<4 |
| 75 | -5<<0 +5<<4 |
| 50 | -7<<1 +1<<6 |
| 0 | 0 |
| -67 | -3<<0 -1<<6 |
| -134 | -3<<1 -1<<7 |
| -180 | +3<<2 -3<<6 |
| -184 | -7<<3 -1<<7 |
| -128 | -1<<7 |
| -4 | -1<<2 |
| 171 | -5<<0 -5<<4 +1<<8 |
| 402 | -7<<1 -3<<5 +1<<9 |
| 632 | -1<<3 +5<<7 |
| 833 | +1<<0 +1<<6 +3<<8 |
| 969 | -7<<0 -3<<4 +1<<10 |
| 1018 | -3<<1 +1<<10 |

*Parameters for Table 5.6:*

Upper-bound in Radix-$2^r$ (adders connected in series) is 4

Optimal value of r (Radix-$2^r$) for adder-cost is 5

Odd positive fundamentals:

3 = +1<<1 +1<<0; 5 = +1<<2 +1<<0; 11 = +9<<0 +1<<1

**Table 5.6 Radix-$2^r$ recoding of L2 filter coefficients**

| Coefficients | Radix-$2^r$ recoding |
|---|---|
| 4 | +1<<2 |
| 8 | +1<<3 |
| 12 | +3<<2 |
| 13 | +15<<0 -1<<1 |
| 9 | +1<<3 +1<<0 |
| 0 | 0 |
| -10 | -5<<1 |
| -16 | -1<<4 |
| -13 | -15<<0 +1<<1 |
| 0 | 0 |
| 19 | -13<<0 +1<<5 |
| 35 | +3<<0 +1<<5 |
| 36 | +9<<2 |
| 18 | +9<<1 |
| -15 | -1<<4 +1<<0 |
| -49 | +15<<0 -1<<6 |
| -64 | -1<<6 |
| -48 | -3<<4 |
| 0 | 0 |
| 60 | +1<<6 -1<<2 |
| 102 | -13<<1 +1<<7 |
| 96 | +3<<5 |
| 32 | +1<<5 |
| -72 | -9<<3 |
| -170 | +11<<1 -3<<6 |
| -203 | -11<<0 -3<<6 |
| -124 | +1<<2 -1<<7 |
| 79 | +15<<0 +1<<6 |
| 371 | -13<<0 +3<<7 |
| 678 | -13<<1 +11<<6 |
| 911 | +15<<0 -1<<7 +1<<10 |

## 5.2 Simulation and implementation results of Filters

The simulation results of all filters is shown in the Figure 5.1-5.5 at a clock frequency of 200MHz. Asynchronous reset is used as an input to the filters for resetting multiply and accumulate (MAC) registers.

Figure 5.1 shows the output waveform for the filter G1. The 21- bit output (Y) is observed at the positive edge of input clock (clk) with time period of 5ns, 50% duty cycle and offset of 1nS. The output becomes stable after 78.50ns.

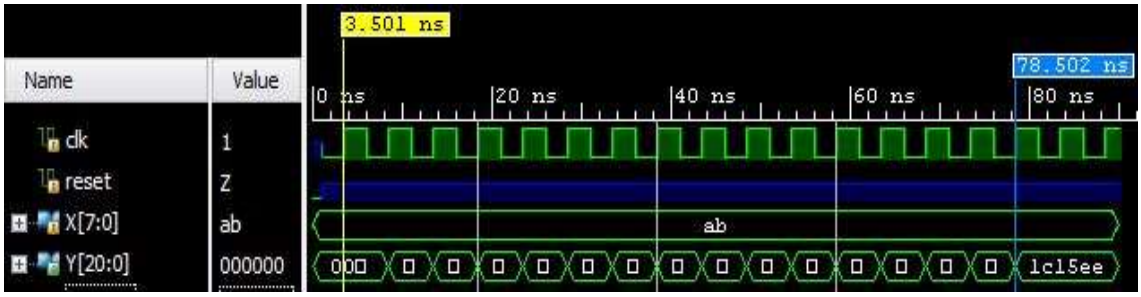



**Figure 5.1 Output waveform of filter G1**

The output waveform for the filter Y1 is shown in the Figure 5.2. The 22- bit output (Y) is observed at the positive edge of input clock (clk) with time period of 5ns, 50% duty cycle and offset of 1ns. The output of Y1 filter becomes stable after 153.50ns.

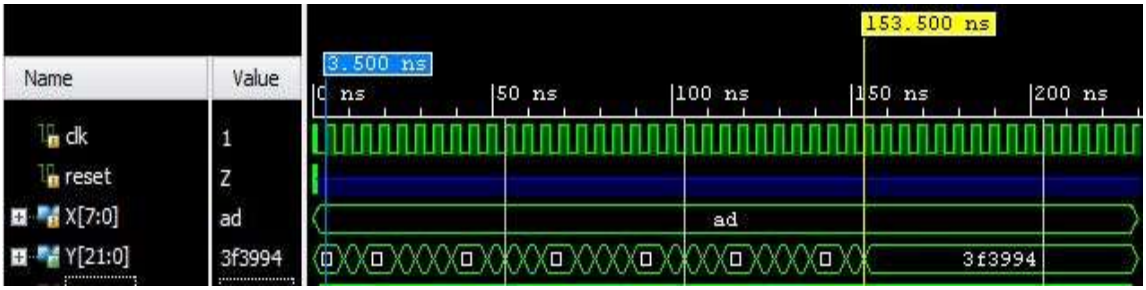



**Figure 5.2 Output waveform of filter Y1**

Figure 5.3 shows the output waveform for the filter Y2. The 23- bit output (Y) is observed at the positive edge of input clock (clk) with time period of 5ns, 50% duty cycle and offset of 1ns. The output becomes stable after 173.50ns.

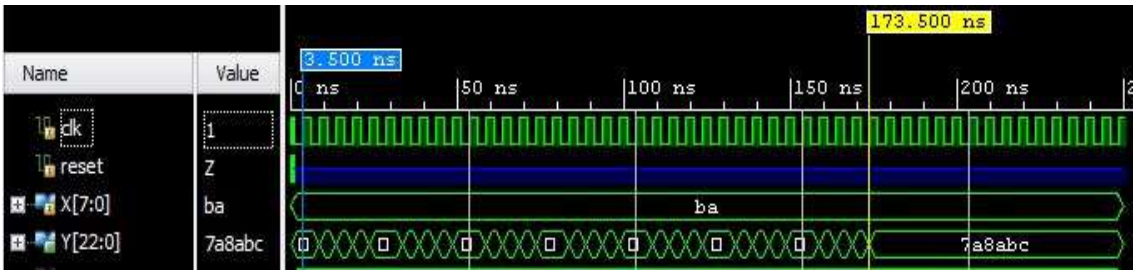



**Figure 5.3 Output waveform of filter Y2**

The output waveform for the filter A1 is shown in Figure 5.4. The 23- bit output (Y) is observed at the positive edge of input clock (clk) with time period of 5ns, 50% duty cycle and offset of 1ns. The output becomes stable after 298.50ns.

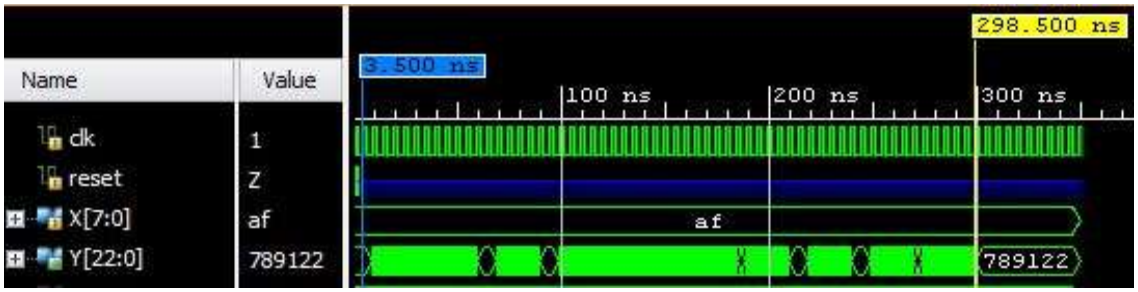



**Figure 5.4 Output waveform of filter A1**

Figure 5.5 shows the output waveform for the filter L2. The 22- bit output (Y) is observed at the positive edge of input clock (clk) with time period of 5ns, 50% duty cycle and offset of 1ns. The output becomes stable after 318.50ns.

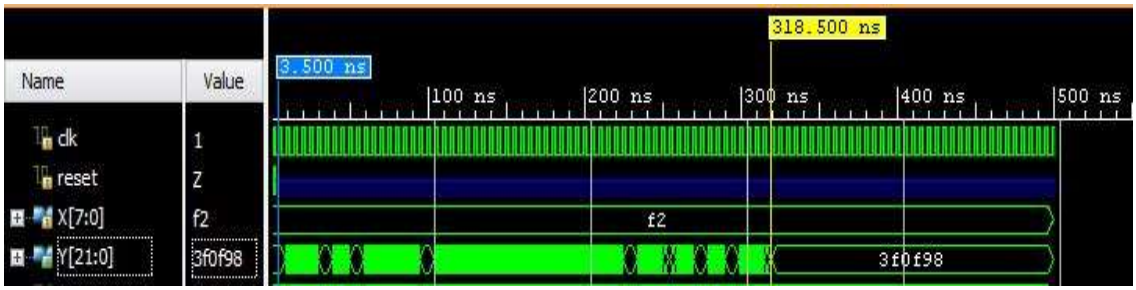



**Figure 5.5 Output waveform of filter L2**

Table 5.7 shows the total number of look-up table (LUT) used for benchmark filters and the total number of slices is mentioned in Table 5.8. These parameters define the utilization of filters when implemented in Artix-7 FPGA xc7a35ticpg236-1L. The mentioned FPGA has 33,280 logic cells in 5200 slices and each slice contains six input LUTs and eight Flip-flops.

**Table 5.7 Total number of LUTs used in filters**

| Filter | LUT Used | Utilization (%) |
|---|---|---|
| **G1** | 387 | 1.86 |
| **Y1** | 554 | 2.66 |
| **Y2** | 663 | 3.19 |
| **A1** | 1290 | 6.20 |
| **L2** | 1336 | 6.42 |

**Table 5.8 Total number of LUTs used in filters**

| Filter | Slices Used | Utilization (%) |
|---|---|---|
| **G1** | 120 | 1.47 |
| **Y1** | 173 | 2.12 |
| **Y2** | 216 | 2.65 |
| **A1** | 402 | 4.93 |
| **L2** | 420 | 5.15 |

Table 5.9 shows the total number of clock cycles consumed by filters for obtaining a stable output value along with the delay. With increasing filter order, the number of clock cycles and delay is also increasing with filter L2 having the maximum number of 63 clock cycle and 315ns of delay in output.

**Table 5.9 Total number of clock cycles used in filters**

| Filter | Clock Cycles | Delay (ns) |
|---|---|---|
| **G1** | 15 | 75 |
| **Y1** | 30 | 150 |
| **Y2** | 34 | 170 |
| **A1** | 59 | 295 |
| **L2** | 63 | 315 |

Figure 5.6 shows the total number of adders utilized by benchmark filters. As observed from the figure, the number of adders keeps on increasing with increasing filter order. Adders utilized in filters G1, Y1 and Y2 are very much greater than those proposed in [60] but with higher order filters i.e. A1 and L2, the number of adders are comparable with [48].

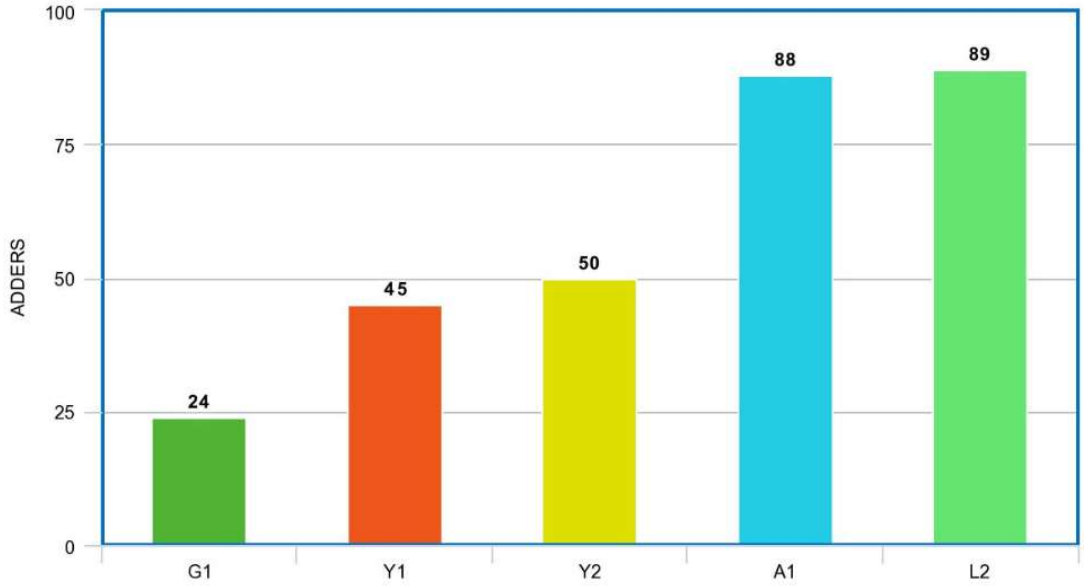



**Figure 5.6 Number of adders utilized by Benchmark filters**

## 5.3 Power Analysis of Filters

The overall power of the filter can be reduced by reducing the switching power. The reason being, the reduction in the switching activity of the filter. Figure 5.8-5.12 shows the reduction in the switching power of the benchmark filters when they are designed using the low power technique with Radix-$2^r$ algorithm in comparison to the Radix-$2^r$ algorithm only.

Figure 5.8 shows the comparison between the switching powers of the Filter G1. The switching power is reduced by a value of 0.05mW thus, saving 7.6% of switching power.

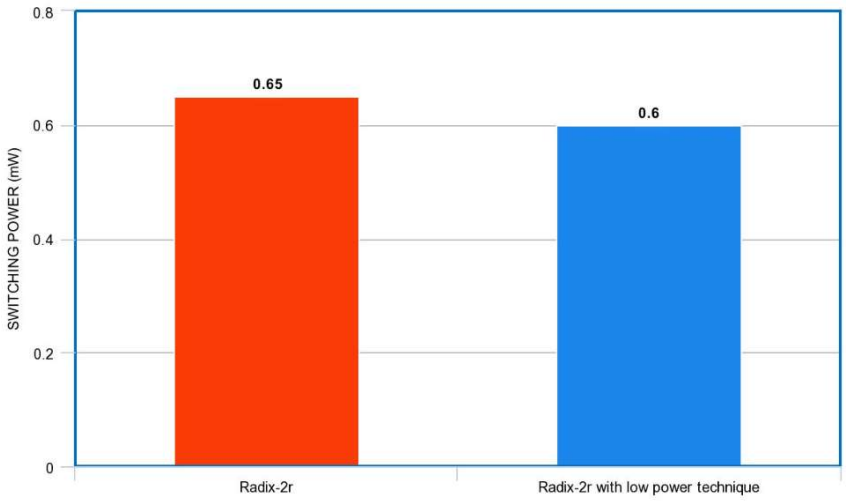



**Figure 5.7 Switching power of filter G1**

Figure 5.9 shows the comparison between the switching powers of the Filter Y1. The switching power is reduced by a value of 0.12mW thus, saving 8.8% of switching power.

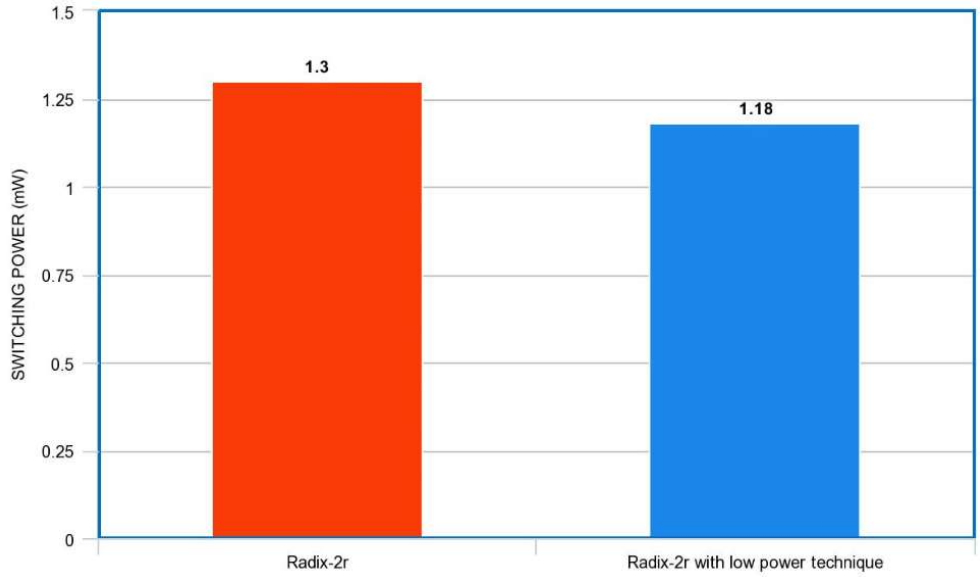



**Figure 5.8 Switching power of filter Y1**

Figure 5.10 shows the comparison between the switching powers of the Filter Y2. The switching power is reduced by a value of 0.19mW thus, saving 9.4% of switching power.

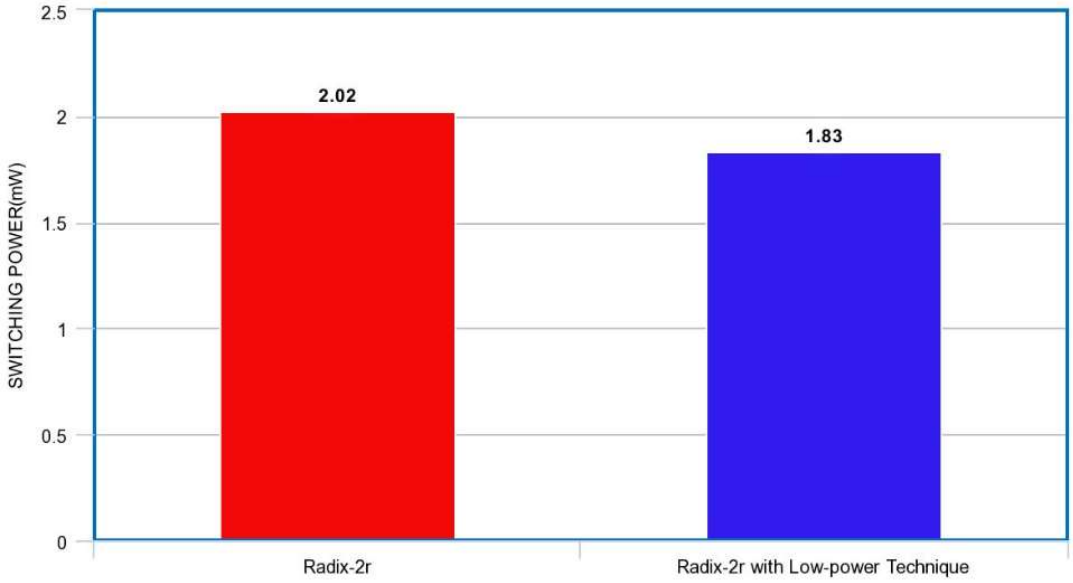



**Figure 5.9 Switching power of filter Y2**

Figure 5.11 shows the comparison between the switching powers of the Filter A1. The switching power is reduced by a value of 0.27mW thus, saving 11.94% of switching power.

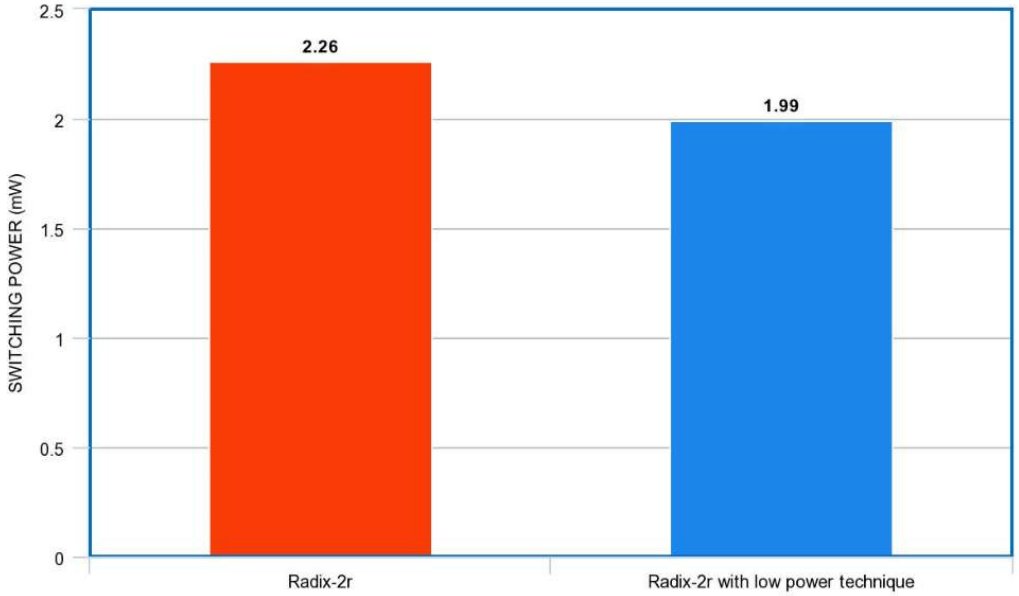



**Figure 5.10 Switching power of filter A1**

Figure 5.12 shows the comparison between the switching powers of the Filter L2. The switching power is reduced by a value of 0.33mW thus, saving 14.1% of switching power.

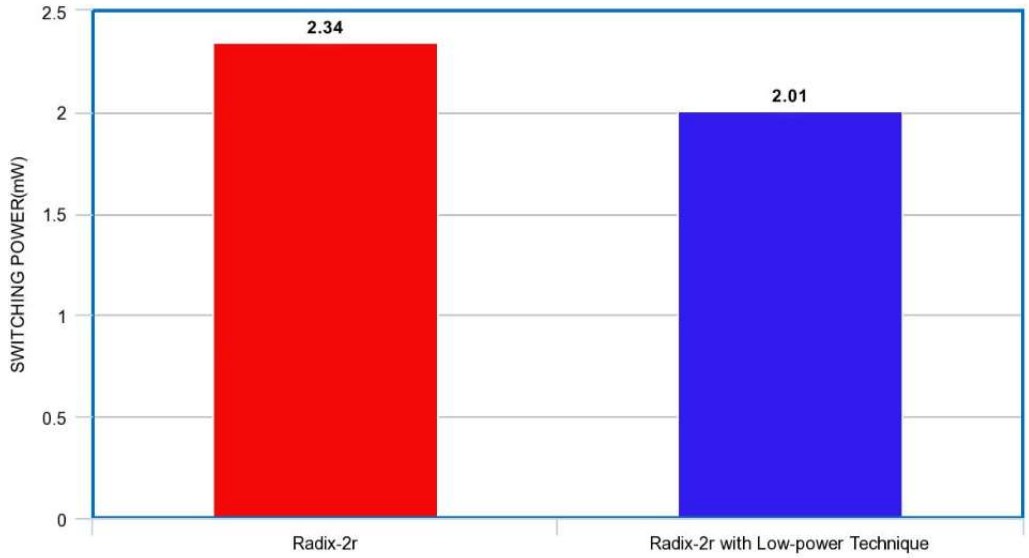



**Figure 5.11 Switching power of filter L2**

Table 5.10 shows the comparison of total power of the filter designed by using Radix-2r with low power technique to those mentioned in the literature survey. As observed from the Table, the power saving is more in high order filters due low logical depth and reduced switching activity. The low logical depth is achieved because of Radix-$2^r$ technique and the low power technique leads to reduced switching activity.

**Table 5.10 The Power comparisons of proposed algorithm with the algorithm in [60] and [48]**

| Filter | Method | Power (mW) | Power Saving (%) |
|---|---|---|---|
| G1 | SHI [60] | 1.75 | - |
| | WEN [48] | 1.41 | 19.4 |
| | Proposed | 1.65 | 5.7 |
| Y1 | SHI | 4.49 | - |
| | WEN | 3.61 | 19.6 |
| | Proposed | 4.41 | 1.7 |
| Y2 | SHI | 6.60 | - |
| | WEN | 5.61 | 15.0 |
| | Proposed | 5.21 | 21.0 |
| A1 | SHI | 9.44 | - |
| | WEN | 8.21 | 13.0 |
| | Proposed | 7.42 | 21.3 |
| L2 | SHI | 9.26 | - |
| | WEN | 8.08 | 12.7 |
| | Proposed | 7.18 | 22.4 |

# CHAPTER 6
# CONCLUSION AND FUTURE SCOPE

The limitation of existing algorithms lies in two aspects. First, the computational complexity to achieve the minimum number of adders is too high to be used in the design of long filters, and second, the number of adder criterion is too coarse and inaccurate for the minimization of power consumption. The research in this thesis deals with the issues of designing multiplierless FIR filters to minimize the power consumption. Reducing adder depth and switching activity is the key foundation. Radix-$2^r$ multiple constant algorithm is used for reducing the adder depth and a reduction in switching power is achieved using a low power technique. Hence, overall power reduction is achieved. The power saving is more in high order filters due low logical depth and reduced switching activity.

**FUTURE SCOPE OF WORK**

- A model can be developed in the future to control the registering of necessary outputs of MCM blocks smartly.
- Application of radix-$2^r$ to high-order filters, where it is expected to provide the best results both in speed and power due to its lowest adder-depth.
- Validated using FPGA as a preliminary step, an ASIC implementation based on a standard-cell library is necessary for an ultimate validation of the whole optimization work.

# REFRENCES


[1] Nekoei F, Kavian YS and Strobel O (2010). Some schemes of realization digital FIR filters on FPGA for communication applications. *Microwave and Telecommunication Technology*, [20th: International Crimean Conference IEEE 2010], pp. 616-619.

[2] Proakis JG, and Dimitris M. *Digital signal processing*. India: Pearson Education, 1996.

[3] Ciletti, Michael D. *Advanced digital design with the Verilog HDL*. vol. 1. Upper Saddle River: Prentice Hall, 2003.

[4] Hsiao SF, Jian JHZ, and Chen MC (2013). Low-cost FIR filter designs based on faithfully rounded truncated multiple constant multiplication/accumulation. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 60(5), 287-291.

[5] Oppenheim AV, and Ronald WS. *Discrete-time signal processing*. USA: Pearson Education, 2014.

[6] Frerking M. *Digital signal processing in communications systems*. Germany: Springer Science & Business Media, 2013.

[7] Rabiner L and Schafer R. (1971). Recursive and non-recursive realizations of digital filters designed by frequency sampling techniques. *IEEE Transactions on Audio and Electroacoustics*, 19(3), 200-207.

[8] Herrmann O, Rabiner LR, and Chan DSK (1973). Practical design rules for optimum finite impulse response low-pass digital filters. *Bell System Technical Journal*, 52(6), 769-799.

[9] Adams J and Willson A (1983). A new approach to FIR digital filters with fewer multipliers and reduced sensitivity. *IEEE Transactions on Circuits and Systems*, 30(5), 277-283.

[10] Bull DR and Horrocks DH (1987). Reduced-complexity digital filtering structures using primitive operations. *Electronics Letters*, 23(15), 769-771.

[11] Bull DR and Horrocks DH (1991). Primitive operator digital filters. *IEE Proceedings G (Circuits, Devices and Systems)*, 138(3), 401-412.

[12] Aktan M, Yurdakul A and Dundar G (2008). An algorithm for the design of low-power hardware-efficient FIR filters. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 55(6), 1536-1545.

[13] Shahein A *et al.* (2012). A novel hybrid monotonic local search algorithm for FIR filter coefficients optimization. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 59(3), 616-627.

[14] Kodek D (1980). Design of optimal finite wordlength FIR digital filters using integer programming techniques. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *28*(3), 304-308.

[15] Lim YC and Parker S (1983). FIR filter design over a discrete powers-of-two coefficient space. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 31(3), 583-591.

[16] Lim YC (1990). Design of discrete-coefficient-value linear phase FIR filters with optimum normalized peak ripple magnitude. *IEEE Transactions on Circuits and Systems*, 37(12), 1480-1486.

[17] Samueli H (1989). An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients. *IEEE Transactions on Circuits and Systems*, 36(7), 1044-1047.

[18] Lim YC *et al.* (1999). Signed power-of-two term allocation scheme for the design of digital filters. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 46(5), 577-584.

[19] Hounsell BI, Arslan T and Thomson R (2004). Evolutionary design and adaptation of high performance digital filters within an embedded reconfigurable fault tolerant hardware platform. *Soft Computing*, 8(5), 307-317.

[20] Cappello P and Steiglitz K (1984). Some complexity issues in digital signal processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(5), 1037-1041.

[21] Yurdakul A and Dündar G (2002). Fast and efficient algorithm for the multiplierless realisation of linear DSP transforms. *IEE Proceedings-Circuits, Devices and Systems*, 149(4), 205-211.

[22] Potkonjak M, Srivastava MB, and Chandrakasan AP (1996). Multiple constant multiplications: efficient and versatile framework and algorithms for exploring common subexpression elimination. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(2), 151-165.

[23] Dempster AG and Macleod MD (1995). Use of minimum-adder multiplier blocks in FIR digital filters. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 42(9), 569-577.

[24] Dempster AG and Macleod MD (1994). Constant integer multiplication using minimum adders. IEE Proceedings-Circuits, Devices and Systems, 141(5), 407-413.

[25] Hartley RI (1996). Subexpression sharing in filters using canonic signed digit multipliers. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 43(10), 677-688.

[26] Farahani MA, Guerra EC and Colpitts BG. Efficient implementation of FIR filters based on a novel common subexpression elimination algorithm. *Electrical and Computer Engineering (CCECE)* [23rd: Calgary, AB, Canada: 2010], pp. 1-4.

[27] Pasko R *et al.* (1999). A new algorithm for elimination of common subexpressions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(1), 58-68.

[28] Martínez-Peiró M, Boemo EI, and Wanhammar L (2002). Design of high-speed multiplierless filters using a nonrecursive signed common subexpression algorithm. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 49(3), 196-203.

[29] Vinod AP *et al.* (2010). An improved common subexpression elimination method for reducing logic operators in FIR filter implementations without increasing logic depth. *Integration, the VLSI journal*, 43(1), 124-135.

[30] Samadi P and Ahmadi M. Common Subexpression Elimination for Digital Filters Using Genetic Algorithm. *Electronics, Circuits and Systems (ICECS)* [14th: Marrakech, Morocco: 2007], pp. 246-249.

[31] Vinod AP *et al.* (2003). FIR filter implementation by efficient sharing of horizontal and vertical common subexpressions. *Electronics Letters*, 39(2), pp.251-253.

[32] Mahesh R and Vinod AP (2008). A new common subexpression elimination algorithm for realizing low-complexity higher order digital filters. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 27(2), pp. 217.

[33] Hatai I, Chakrabarti I and Banerjee S. (2018). A Computationally Efficient Reconfigurable Constant Multiplication Architecture Based on CSD Decoded Vertical–Horizontal Common Sub-Expression Elimination Algorithm. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(1), pp.130-140.

[34] Chen J *et al.* (2016). A new cost-aware sensitivity-driven algorithm for the design of FIR filters. *IEEE Trans Circuits Syst. I PP*, *99*, 1-11.

[35] Bernstein R (1986). Multiplication by integer constants. *Software: practice and experience*, 16(7), 641-652.

[36] Voronenko Y and Püschel M (2007). Multiplierless multiple constant multiplication. *ACM Transactions on Algorithms (TALG)*, 3(2), 1-11

[37] Thong J and Nicolici N (2011). An optimal and practical approach to single constant multiplication. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(9), 1373-1386.

[38] Ye WB and Yu YJ (2013). Single-stage and cascade design of high order multiplierless linear phase FIR filters using genetic algorithm. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(11), 2987-2997.

[39] Homayoon S and Gupta A (1990). A Generalized Multibit Recoding of Two's Complement Binary Numbers and its Proof with Application in Multiplier Implementation. *IEEE Trans. on Computers (TC)*, 39(8).

[40] Oudjida AK and Chaillet N (2014). Radix-2r Arithmetic for Multiplication by a Constant. *IEEE Transactions on Circuits and Systems-II: Express Briefs*, 61, 349-353.

[41] Oudjida AK, Chaillet N and Berrandjia ML (2015). Radix-2 r arithmetic for multiplication by a constant: further results and improvements. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 62(4), 372-376.

[42] Oudjida *et al.* (2016). Multiple Constant Multiplication Algorithm for High-Speed and Low-Power Design. *IEEE Trans. on Circuits and Systems*, 63(2), 176-180.

[43] Liacha A, Oudjida AK and Ferguene F. Radix-$2^r$ arithmetic for FIR filter design optimization. *Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)* [Istanbul, Turkey: 2015], pp. 1-4.

[44] Johansson K, Gustafsson O and DeBrunner LS. Estimation of the switching activity in shift-and-add based computations. *Circuits and Systems (ISCAS*) [17th: Taipei, Taiwan: 2009], pp. 3054-3057.

[45] Ye WB and Yu YJ. Switching activity analysis and power estimation for multiple constant multiplier block of FIR filters. *Circuits and Systems (ISCAS)* [Rio de Janeiro, Brazil: 2011], pp. 145-148

[46] Lou *et al.* (2015). Fine-Grained Critical Path Analysis and Optimization for Area-Time Efficient Realization of Multiple Constant Multiplications. *IEEE Trans. on Circuits and Systems*, 62(3), 863-872.

[47] Lou X, Ye W and Yu YJ (2017). Investigation on power consumption of product accumulation block for multiplierless FIR filters. In *Digital Signal Processing (DSP), 2017 22nd International Conference on* (pp. 1-5). IEEE.

[48] Ye WB, Lou X, and Yu YJ (2017). Design of Low-Power Multiplierless Linear-Phase FIR Filters. *IEEE Access*, *5*, 23466-23472.

[49] Muller JM *et al. Handbook of floating-point arithmetic*. Boston: Birkhäuser, 2010.

[50] Behrooz P. *Computer arithmetic: Algorithms and hardware designs*. USA: Oxford University Press, 2000.

[51] Kastner R, Anup H and Farzan . *Arithmetic optimization techniques for hardware and software design*. England: Cambridge University Press, 2010.

[52] Lopez B, Hilaire T, and Didier LS. Sum-of-products evaluation schemes with fixed-point arithmetic, and their application to IIR filter implementation. *Design and Architectures for Signal and Image Processing (DASIP)* [4th: Karlsruhe, Germany: 2012], pp. 1-8

[53] Avizienis A (1961). Signed-digit numbe representations for fast parallel arithmetic. *IRE Transactions on electronic computers*, (3), 389-400.

[54] Dimitrov VS, Jullien GA and Miller WC (1999). Theory and applications of the double-base number system. *IEEE Transactions on Computers*, 48(10), 1098-1106.

[55] Thong J, New Algorithm for Constant Coefficient Multiplication in Custom Hardware. Master Thesis, McMaster University, Hamilton, Ontario, Canada, 2009.

[56] Seidel PM, McFearin LD and Matula DW (2005). Secondary radix recordings for higher radix multipliers. *IEEE Transactions on Computers*, 54(2), 111-123.

[57] Dimitrov VS, Jarvinen KU and Adikari J (2011). Area-efficient multipliers based on multiple-radix representations. *IEEE Transactions on Computers*, 60(2), 189-201.

[58] Lim YC (1979). Linear phase FIR digital filter without multipliers. In *1979 IEEE International Symposium on Circuits and Systems*.

[59] Suite of constant coefficient fir filters. Available at http://www.firsuite.net/ (Accessed on 10th October 2017 )

[60] Shi D and Yu YJ (2011). Design of linear phase FIR filters with high probability of achieving minimum number of adders. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 58(1), 126-136.