CHAPTER 1

INTRODUCTION

1.1 BASICS OF SIGNAL PROCESSING

Signal Processing is a method of extracting information from the signal which, in turn, depends upon the type of signal and the nature of information it carries. Signal Processing is concerned with representing signals in mathematical terms and extracting the information by carrying out algorithmic operations on the signal. In the Digital Signal Processing (DSP), all the signals are discrete. When the analog signals such as speech, music etc., are to be processed by the DSP system, such signals are converted to discrete form by A/D converters. After processing, the signals are converted into analog form by D/A converters.

Many applications demand the processing of signals in frequency domain. Frequency content, periodicity, energy and power spectrum etc., can be better analyzed in frequency domain. Hence the signals are transformed from time domain to frequency domain. Such transformations can be obtained with the help of Fourier Transform (FT) and Discrete Fourier Transform (DFT). For the discrete signals, DFT is used. DFT can also be used for linear filtering and correlation. Once the required analysis and processing are performed in frequency domain, the signals are transformed back in time domain by Inverse Discrete Fourier Transform (IDFT).

DSP has recently become an available technology in many areas. Many products that were historically based on analog or micro-controller systems are now being migrated to DSP microprocessor-based systems. The widely used signal processing technology in electronics is the filter and is considered to be the fundamental concept in any electronics applications. Primarily the function of any filter is to selectively allow the desired signal to pass through and suppress the undesired signal based on the frequency (Frerking 1994). The design of the filter determines the loss and degree of phase shift which occur during the insertion and the rejection. The filters are broadly be classified into Analog filters and Digital filters (Lindsey et al 1995).

1.2 ANALOG FILTERS

An analog filter can be defined as a filter which operates on continuous time signals (Williams 2006). Particularly, the analog filter is characterized by the impulse response with respect to time. Normally the analog filters are described by differential equations (Daniels 1974). In order to compute the transfer function, Laplace transform is used instead of z-transform (Van Valkenburg 1995). Since the sampling-rate is allowed to go to infinity, analog filters can be considered to be the limiting case of digital filters. Earlier to the use of digital computers, physical systems were simulated on analog computers. Here, the analog computer was similar to an analog synthesizer providing modular building-blocks that could be integrated together to build models of dynamic systems.

The applications of analog filters are:

- The separation of an audio signal before application to bass, mid-range and tweeter loudspeakers

- The combining and later separation of multiple telephone conversations onto a single channel
- The selection of a chosen radio station in a radio receiver and rejection of others and much more

The analog filters are fabricated with capacitors, inductors and occasionally the resistors. These analog filters are designed in such a way that they will operate on continuously varying signals that are analog in nature. The role of analog filters is very crucial particularly in the telecommunication fields (Zumbahlen 2009). In the initial stage, the analog filters are connected along with the transmission lines and hence the transmission line theory ends up with the filter theory.

In order to filter the mechanical vibrations or acoustic waves, one can design a linear analog filter with the help of mechanical components. If the transducers are added in such mechanical filters, they will become the electronic filters to convert to and from the electrical domain. The implementations of analog filters at low frequencies are found to be most active in avoiding wound components required by any passive topology.

Analog filters find their applications especially in lower order simple filtering tasks and are often at higher frequencies where digital technology is still impractical, or at least, less cost effective. However, in recent years it is often preferred to carry out filtering in the digital domain where complex algorithms are much easier to implement.

1.3 DIGITAL FILTERS

DSP has been increasing in popularity due to the declining cost of general purpose computers and application specific hardware. Since many telephony and data communication applications have been moving to digital, the need for digital filtering methods continue to grow (Hamming 1997). Hence, simulation techniques to model these complex systems are needed as well. Software simulators offer flexible schemes to code the algorithm from a choice of many languages but cannot always offer the speed that a hardware simulator can. Unfortunately, building hardware prototypes to model different systems can be costly and time consuming when constant changes have to be made. Therefore, a middle ground might be found using custom computing platforms or programmable logic. Such systems can offer similar flexibility as software and still retain some or all of the hardware acceleration at the cost of a shorter implementation cycle.

Digital filters or digital signal processors are small special purpose digital computers designed to implement an algorithm that converts an input sequence x(n) into a desired output sequence y(n). DSP system is demonstrated in Figure 1.1.

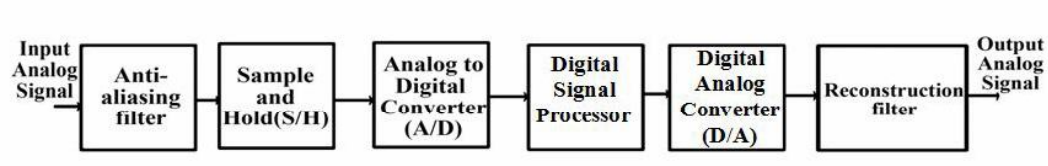

Figure 1.1 Block diagram representation of the DSP system

The process of converting an analog signal into digital form is performed by sampling with a finite sampling frequency. If an input signal contains frequency components higher than half the sampling frequency, it

will cause distortion to the original spectrum (Rabiner and Charles 1993). Basic Fourier transform theory states that the linear convolution of two sequences in the time domain is same as multiplication of two corresponding spectral sequences in the frequency domain. Filtering is in essence the multiplication of the signal spectrum by the frequency domain impulse response of the filter (Cappellini et al 1978). For an ideal low-pass filter, the pass band part of the signal spectrum is multiplied by one and the stop band part of the signal by zero.

Owing to the way that analog and digital filters are physically implemented, the analog filter is inherently more sized and power-efficient, although more component-sensitive, than its digital counterpart. In general, as signal frequency increases, the disparity in efficiency increases (Bogner et al 1980). Characteristics of applications where digital filters are more efficient than analog filters are:

- linear phase, very high stop band attenuation and very low pass band ripple
- the filter's response must be programmable or adaptive
- the filter must manipulate phase and very low shape factors (a digital filter's shape factor is the ratio of the filter's pass bandwidth plus the filter's transition bandwidth to the filter's pass bandwidth).

General purpose digital signal microprocessors, now commodity devices, are used in a broad range of applications and can implement moderately complex digital filters in the audio frequency range. Many standard signal processing algorithms, including digital filters, are available in software packages from digital signal processor and third party

vendors. As a result, software development costs are trivial when amortized over production quantities. The architectures of digital signal microprocessors are usually optimized to perform a sum-of-products calculation with data from RAM or ROM. They are not optimized for any specific DSP function. However, to get extended sampling rate performance from a digital filter requires hardware which is designed to perform the intended filter function at the desired sampling frequencies.

Higher performance is possible for high volume applications by limiting the range of parameters. Standard filter products strike a balance between optimized filter architectures and programmability by offering a line of configurable filters. That is, these products are function-specific, with optimized architectures and programmable parameters. Conceptual differences exist in Frequency-Domain Versus Time Domain. Thinking about analog filters, most engineers are comfortable in the time domain. For example, the operation of an RC low-pass filter can easily be envisioned as a capacitor charging and discharging through a resistor. Likewise, it is easy to envision how a negative-feedback active filter uses phase shift as a function of frequency, which is a time domain operation.

The digital filter is better conceptualized in the frequency domain. The filter implementation simply performs a convolution of the time domain impulse response and the sampled signal. The filter is designed with a frequency domain impulse response which is as close as to the desired ideal response can be generated while given with the constraints of the implementation. The frequency domain impulse response is then transformed into a time domain impulse response which is converted into the coefficients of the filter (Manolakis and Proakis 2007).

1.3.1 Types of Digital Filters

Filters can be classified in several different groups based on different criteria (Bailey 2009). The two major types of digital filters are:

(i) Finite Impulse Response filters (FIR)

(ii) Infinite Impulse Response filters (IIR).

1.3.2 Finite Impulse Response Filters (FIR)

It follows from the linearity and time-invariance properties (Oppenheim and Schafer 2000) that a linear Time Invariant System (LTI) is completely defined by its impulse response, denoted by h(n). An output sequence of the discrete-time LTI system, denoted by y(n) can be expressed in terms of h(n) and the input sequence, denoted by x(n) using the following sum

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)\,x(n-k) \tag{1.1}$$

The above difference equation is known to be a FIR discrete-time filter, if h(n) is of finite length, i.e.,

$$h(n) = 0 \text{ for } n < M_1 \text{ and } n > M_2 \text{ with } M_1 < M_2 \tag{1.2}$$

In this case, the FIR filter is defined by the convolution sum of the form (Kenneth 1993, Rabiner and Charles 1993, Mitra 2010)

$$y(n) = \sum_{k=M_1}^{M_2} h(k)\,x(n-k) \tag{1.3}$$

where $M_2 - M_1$ is called order of the filter usually denoted by M and the length of the filter is $M_2 - M_1 + 1$. An FIR filter is called causal if $M_1 \geq 0$. Alternatively, by selecting $M_1 = 0$ and $M_2 = M$ in Equation (1.3), the input-output relation is expressible in z domain as

$$Y(z) = H(z)\, X(z) \tag{1.4}$$

where X(z) and Y(z) are the z-transforms of the input sequence x(n) and the output sequence y(n), respectively, whereas H(z) is the z-transform of the impulse response, h(n). The function H(z) is called the transfer function of the FIR filter with non-zero impulse-response values for $0 \leq n \leq M$ and can be expressed as

$$H(z) = \sum_{n=0}^{M} h(n) z^{-n} \tag{1.5}$$

The transfer function plays a very crucial role in analyzing the filter performance. The frequency domain performance of the filter can be studied by substituting $z = e^{j\omega}$ in Equation (1.5), yielding

$$H(e^{j\omega}) = \sum_{n=0}^{M} h(n) e^{-j\omega n} \tag{1.6}$$

This function is the Discrete-Time Fourier Transform (DTFT) of the impulse response of an FIR filter and is called the frequency response of the filter. Based on the fact that the DTFTs of the input and output sequence, denoted by $X(e^{j\omega})$ and $Y(e^{j\omega})$ respectively, are related through

$$Y(e^{j\omega}) = H(e^{j\omega})\, X(e^{j\omega}) \tag{1.7}$$

the effect of the filtering on the input sequence can be studied in the frequency domain (Kenneth 1993, Rabiner and Charles 1993, Mitra 2010, Tapio Saramäki 1993).

1.3.3 Linear-phase FIR Filter Types

The linear-phase property in filters is necessary for many applications where phase distortion is not allowed. The typical example is an ECG signal, where the removal of the line frequency interference and possible breathing artifacts should be filtered so that the waveform of the original ECG signal is preserved.

The FIR filter can always be designed with an exact linear-phase response. The causal FIR filter has a linear phase if its impulse response is either symmetrical or antisymmetrical and if the order, defined in Section 1.3.2, of the filter is either even or odd. Therefore, there are the following four linear-phase FIR filter types (Mitra 2010)

Type 1: $h(M - n) = h(n)$, $n=0,1\ldots,M$ and M is even

Type 2: $h(M - n) = h(n)$, $n=0,1\ldots,M$ and M is odd

Type 3: $h(M - n) = -h(n)$, $n=0,1\ldots,M$ and M is even

Type 4: $h(M - n) = -h(n)$, $n=0,1\ldots,M$ and M is odd

An example of the impulse responses for the four linear-phase types is shown in Figure 1.2. Based on these impulse responses, the following observations can be made. First, for Type 1 and 2 filters (Type 3 and 4), the impulse responses are of even (odd) length. Secondly, for Type 1 and 2 (Type 3 and 4) filters, the impulse responses are symmetrical (antisymmetrical). Thirdly, at the center of the symmetry, N/2, there is (is not) an impulse response sample for Type 1 and 4 (Type 2 and 3) filters. Because of the

antisymmetrical impulse response, the central impulse-response value for Type 3 filter has to be zero.

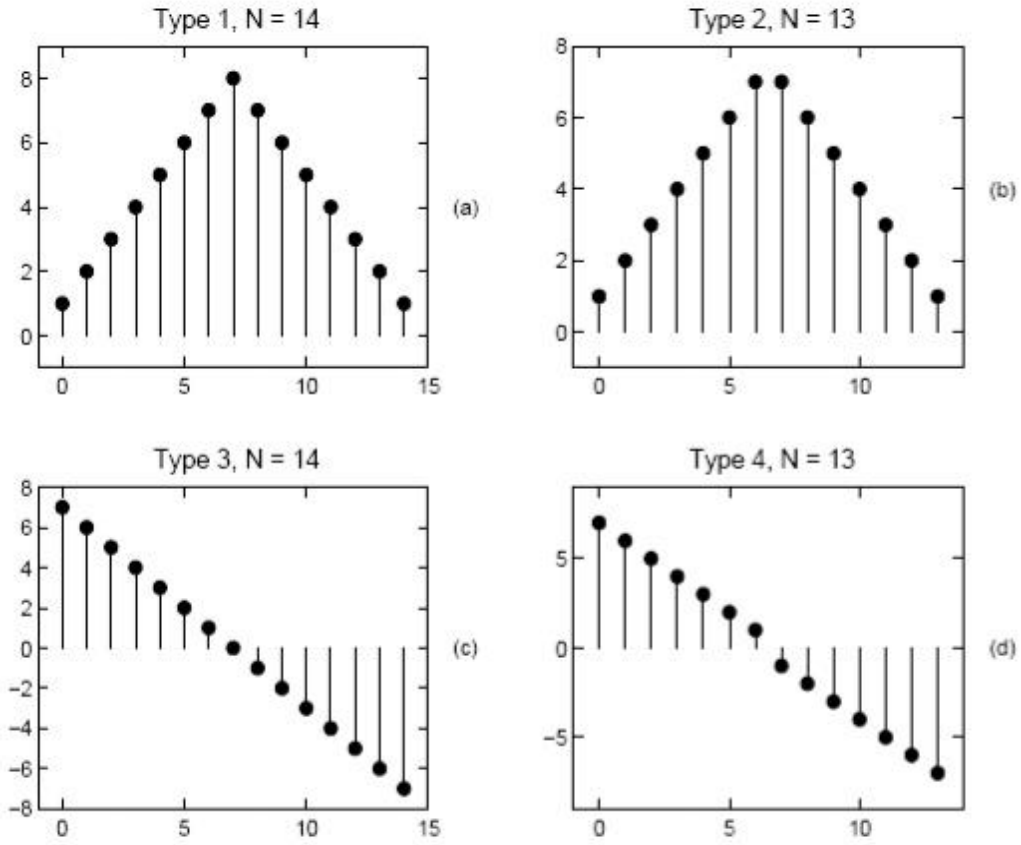

Figure 1.2 Impulse responses for the four linear-phase FIR filter types. (a) Type 1 FIR filter. (b) Type 2 FIR filter. (c) Type 3 FIR filter. (d) Type 4 FIR filter

It is worth emphazising that only Type 1 and 2 filters are used to design frequency selective filters since they provide a constant phase delay in the filter passbands, thereby quaranteeing that the waveform of the signal consisting of the components in these bands is preserved. This is not true for Type 3 and 4 filters. However, these filter types are suitable for FIR differentiators and Hilbert transformer design because of their antisymmetric properties and an additional 90 degree phase shift.

1.3.4 Structure of FIR

FIRs have the advantage of being much more realizable in hardware (Little et al 1993) because they avoid division and feedback paths. Despite needing twice the filter order of an IIR, FIRs are dependent on the data coming through the filter and on past values. FIRs can be graphically represented by a Direct Form realization as shown in Figure 1.3.

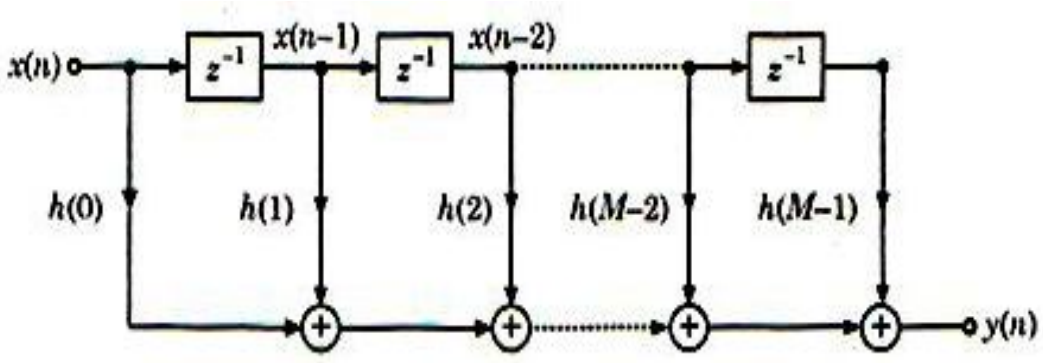

Figure 1.3 Direct Form realization signal flow graph of an FIR

Here, current output *y(n)* is calculated solely from the current and previous input values (x(n),*x(n-1), x(n-2)…*). This type of filter is also said to be non-recursive filters.

The basic characteristics of FIR filters are:

- Linear phase characteristic
- High filter order (more complex circuits) and
- Stability

The FIR realization can be highly replicable, which becomes important in the hardware design. One important aspect of FIRs is the linear phase characteristic, which makes it ideal for most digital signal processing

applications (Proakis et al 1988). Nonrecursive filters are always stable unlike the recursive or IIR filters which have to keep the pole placements in perspective. Again, FIRs have to have twice the order of an IIR because they cannot achieve the smaller side lobes in the stop band of the frequency response given the same number of parameters as an IIR (Proakis et al 1988).

To shape the frequency selective band of the FIR, "windowing functions" are convolved with the filter function. Examples of these functions include the Bartlett (rectangular), Blackman, and Hamming windows. The windowing technique tends to give the frequency response a sharper cutoff and "flatter" response in the passband (Oppenheim and Schafer 2009) Typically, a window with a taper and gradual roll off to zero produces less ringing in the sidelobes and lessens the oscillations in both the passband and stopband.

The oscillations commonly found in the frequency response are due to Gibbs phenomenon, which is due to abrupt truncations of the Fourier series representation of the frequency response. Unfortunately, when correcting excess ringing in the sidelobes, the window is widened, which means an increase in the width of the transition band of the filter, or a higher order filter. Despite the higher order of the FIR filter, the implementation is feasible in hardware and possesses the necessary linear phase property needed by channel models (Rappaport et al 1991).

1.3.5 Infinite Impulse Response Filters (IIR)

IIRs not only use the data values that pass through but also use other values of the output, which can be described by the following equation (Hamming 1997)

$$y(n) = \sum_{k=-\infty=}^{\infty} b_k \, x(n - k) + \sum_{k \; -\infty}^{\infty} a_k \, y(n - k) \tag{1.8}$$

where *a* and *b* represent the IIR coefficients and *x* the input data. The output y(n) is equal to the sum of past outputs that are scaled by the delay dependent feedback coefficient a_k, plus the sum of present and past inputs which are scaled by the delay dependent feed forward coefficient b_k.

The recursive system can also be described in a Direct Form II structure as shown in Figure 1.4.

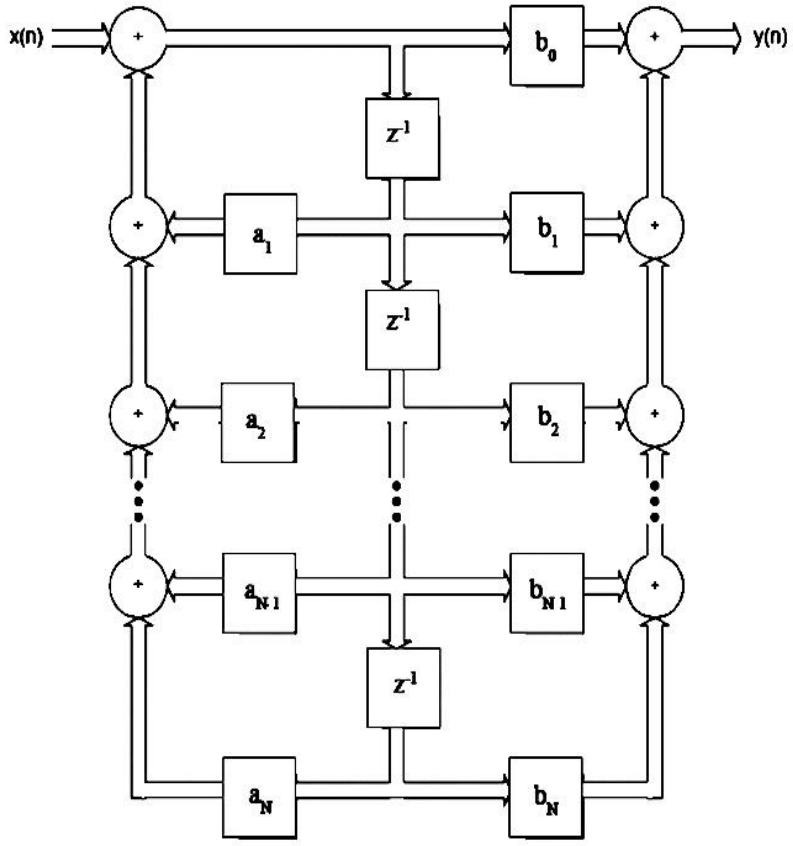

Figure 1.4 Direct form II realization signal flow graph of an IIR

Most importantly here, IIR implementation is not as easily realizable as that of the FIR even though IIRs typically require a lower order filter to accomplish the same function (Little et al 1993). But IIRs are preferred due to fewer parameters, less memory requirements, and lower computational complexity (Proakis 1988). The non-linear phase characteristic

generated by IIRs can be a severe drawback. Some transform techniques, such as the impulse invariance technique, do not have a direct filter frequency interpretation and may give unwanted effects such as aliasing and other phase problems (Cappellini et al 1978).

An IIR filter is also called as recursive filter in which in addition to input values, it also uses previous output values (Young et al 1995). These, like the previous input values, are stored in the processor's memory. The word *recursive* literally means “running back” and refers to the fact that previously-calculated output values go back into the calculation of the latest output. The expression for a recursive filter therefore contains not only terms involving the input values (*x(n),x(n-1), x(n-2).*…). but also terms in (*y(n-1), y(n-2).*…).

From this explanation, IIR filters require more calculations to be performed, since there are previous output terms in the filter expression as well as input terms. To achieve a given frequency response characteristic, a recursive filter generally requires a much lower order filter, and therefore fewer terms are to be evaluated by the processor, than the equivalent non-recursive filter.

The basic characteristics of IIR are:

- Non-linear phase characteristic
- Low filter order (less complex circuits) and
- Resulting digital filter has the potential to become unstable.

1.4 COMPARISON OF FIR AND IIR FILTERS

Both types have some advantages and disadvantages that should be carefully considered when designing a filter (Shaw and Ahmed 1999). Besides, it is necessary to take into account all fundamental characteristics of

a signal to be filtered as these are very important when deciding which filter to use. In most cases, it is only one characteristic that really matters and it is whether it is necessary that filter has linear phase characteristic or not.

Speech signal, for example, can be processed in the systems with non-linear phase characteristic. The phase characteristic of a speech signal is not of the essence and as such can be neglected, which results in the possibility to use much wider range of systems for its processing. There are also signals for which the phase characteristic is of the essence. Typical examples are signals obtained from various sensors in industry. Therefore, it is necessary that a filter has linear phase characteristic to prevent loosing important information (Shaw and Ahmed 1999).

When a signal to be filtered is analyzed in this way, it is easy to decide which type of digital filter is best to use. Accordingly, if the phase characteristics is of the essence, FIR filters should be used as they have linear phase characteristics. Therefore such filters are of higher order and more complex. Otherwise, when it is only frequency response that matters, it is preferable to use IIR digital filters which have far lower order, i.e. are less complex, and thus much easier to realize (Chao-Huang Wei et al 2005).

IIR filters are digital filters with infinite impulse response. Unlike FIR filters, they have the feedback and are known as recursive digital filters. Due to this, IIR filters have better frequency response than FIR filters of the same order. Unlike FIR filters, their phase characteristic is not linear which can cause a problem to the systems which needs phase linearity. Hence, it is not preferable to use IIR filters in digital signal processing when the phase is of the essence. Otherwise, when the linear phase characteristic is not important, the use of IIR filters is an excellent solution.

There is one problem known as a potential instability that is typical of IIR filters only. FIR filters do not have such a problem as they do not have the feedback. Hence, it is always necessary to check after the design process whether the resulting IIR filter is stable or not (Chao-Huang Wei et al 2005).

FIR filters can have linear phase characteristic, which is not typical of IIR filters. When it is necessary to have linear phase characteristic, FIR filters are the only available solution. In other cases when linear phase characteristic is not necessary. FIR filters are not good solution. IIR filters should be used instead. The resulting filter order is considerably lower for the same frequency response (Chao-Huang Wei et al 2005). The filter order determines the number of filter delay lines. That is the number of input and output samples that should be saved in order that the next output sample may be computed.

The most commonly used IIR filter design method uses reference analog prototype filter. It is the best method to use while designing standard filters such as low-pass, high-pass, band pass and band-stop filters. The initial step for the filter design process starts with specifications and requirements of the desirable IIR filter (Proakis et al 1988).

The next step in the design process is scaling of the frequency range of analog prototype filter into desirable frequency range. The most popular and most commonly used converting method is bilinear transformation method. The resulting filter, obtained in this way, is always stable. However, instability of the resulting filter, when bilinear transformation is used, may be caused only by the finite word-length side-effect.

1.4.1 Strength of FIR Filters over IIR Filters

As an IIR filter uses both a feed-forward polynomial (zeros as the roots) and a feedback polynomial (poles as the roots), it has a much sharper transition characteristic for a given filter order. Like analog filters, IIR filter usually has nonlinear phase characteristics. Also, the feedback loop makes IIR filters difficult to use in adaptive filter applications. Due to its all zero structure, the FIR filter has a linear phase response when the filter's coefficients are symmetric, as is the case in most standard filtering applications.

The FIR's implementation noise characteristics are easy to model, especially if no intermediate truncation is used. The IIR filter's poles may be close to or outside the unit circle in the Z plane. This means an IIR filter may have stability problems, especially after quantization is applied. The FIR filter is always stable and also allows development of computationally efficient architectures in decimating or interpolating applications.

In many filtering applications, FIR digital filters are preferred to their IIR counterparts due to their many favorable properties (Nakamura and Mitra 1982). These properties include, among others, the following strength over IIR filters:

- An FIR filter can be designed with an exact linear phase, which means that no phase distortion is caused in the input signal during the filtering operation (Saramäki 1987).
- Both the output noise due to multiplication round-off errors and the sensitivity to variations in filter coefficients are lower.

- Non-recursive realizations of FIR filters are inherently stable and free of limit cycle and parasitic oscillations when implemented on a finite-word-length system.
- They are suited to multi-rate applications. Multirate may be either “decimation” (reducing the sampling rate), “interpolation” (increasing the sampling rate). Whether decimating or interpolating, the use of FIR filters allows some of the calculations to be omitted, thus providing an important computational efficiency. In contrast, if IIR filters are used, each output must be individually calculated, even if that output is discarded (so the feedback will be incorporated into the filter).
- They have desirable numeric properties. In practice, all DSP filters must be implemented using finite-precision arithmetic, that is, a limited number of bits. The use of finite-precision arithmetic in IIR filters can cause significant problems due to the use of feedback, but FIR filters without feedback can usually be implemented using fewer bits and the designer has fewer practical problems to solve related to non-ideal arithmetic.
- Unlike IIR filters, it is always possible to implement a FIR filter using coefficients with magnitude of less than 1.0. (The overall gain of the FIR filter can be adjusted at its output, if desired.) This is an important consideration while using fixed-point DSP's, because it makes the implementation much simpler.

1.4.2 Real Time Applications of FIR Filters

Before the advent of computers and digital sampling, engineers dealt primarily with analog filters implemented in electrical circuits. These filters used resistors, capacitors and inductors to perform an "analog computation" known as a convolution. With the invention of digital computers and A/D converters, the convolution process can be performed by a series of multiplications and additions on binary data samples that represent a signal. Digital filters have some significant advantages over analog filters.

For example, the tolerance values of analog filter circuit components are large enough to make high order filters difficult or impossible to implement. With digital filters, such high order filters are easily realized. In addition, analog component values can change with age or temperature affecting the response of the filter. Digital filters do not have that problem. Another major advantage of digital filters is the ability to reprogram them by changing the coefficients. This greatly simplifies the implementation of adaptive filters (Lin Jieshan and Huang Shizhen 2009).

The FIR filter has only zeros and no poles in its transfer function. Thus it is always stable and cannot oscillate. Therefore, the impulse response of an FIR filter has finite length. Also, the FIR filter may be specified with an exactly linear phase response (Zhang Chi Guo Li Li 2006). An IIR filter has both zeros and poles in its transfer function and can be unstable. Its impulse response theoretically lasts to infinity. In other words, it is implemented with a feedback loop. An IIR filter cannot obtain a true linear phase response. However, it can approximate linear phase over regions of interest (Abdel-Hamid et al 2003).

1.5 DEVELOPMENTS OF VLSI ARCHITECTURE

The increasing costs of silicon technology have put considerable pressure on developing dedicated System-on-Chip (SoC) solutions and means that the technology will be used increasingly for high-volume or specialist markets (Woods et al 2008). An alternative is to use microprocessor style solutions such as microcontrollers, microprocessors and DSP micros, but in some cases, these offerings do not match well to the speed, area and power consumption requirements of many DSP applications.

More recently, the FPGA has been proposed as a hardware technology for DSP systems as they offer the capability to develop the most suitable circuit architecture for the computational, memory and power requirements of the application in a similar way to SoC systems.

This has removed the preconception that FPGAs are only used as ‘glue logic’ platform and more realistically shows that FPGAs are a collection of system components with which the user can create a DSP system. While the prefabricated aspect of FPGAs avoids many of the deep submicron problems met with while developing SoC implementations, the ability to create an efficient implementation from a DSP system description, remains a highly convoluted problem (Meyer-Baese 2001).

FPGAs emerged as simple ‘glue logic’ technology, providing programmable connectivity between major components where the programmability was based on either antifuse, EPROM or SRAM technologies (Maxfield 2004). This approach allows design errors which had only been recognized at this late stage of development to be corrected, possibly by simply reprogramming the FPGA thereby allowing the interconnectivity of the components to be changed as required. While this approach introduces additional delays due to the programmable interconnect,

it avoids a costly and time-consuming board redesign and considerably reduces the design risks.

Like many other electronics industries, the creation and growth in the market has been driven by Moore's law (Moore 1965), represented pictorially in Figure 1.5. Moore's law shows that the number of transistors has been doubling every 18 months.

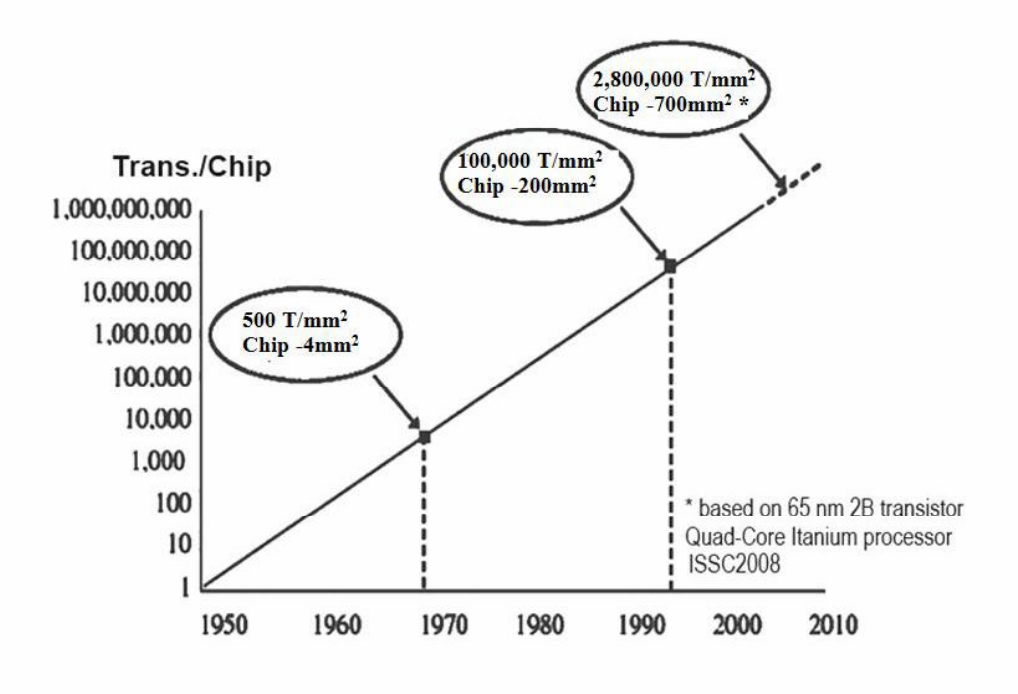

Figure 1.5 Moore's law (Moore 1965)

The incredible growth has led to the creation of a number of markets and is the driving force between the markets of many electronics products such as mobile telephony, digital musical products, digital TV to name but a few. This is because not only have the number of transistors doubled at this rate, but the costs have not increased, thereby reducing the cost per transistor at every technology advance. This has meant that the FPGA market has grown from nothing in just over 20 years to being a key player in the IC industry (Mead et al 1979).

On many occasions, the growth indicated by Moore's law has led people to argue that transistors are essentially free and therefore can be exploited as in the case of programmable hardware, to provide additional flexibility. This could be backed up by the observation that the cost of a transistor has dropped so much these years compared to the previous years. This observation could be argued to have been validated by the introduction of hardware programmability into electronics in the form of FPGAs (Villasenor and Mangione-Smith 1997).

In order to make a single transistor programmable in SRAM technology, the programmability is controlled by storing a '1' or a '0' on the gate of the transistor, thereby making it conduct or not. This value is then stored in SRAM cell which typically requires six transistors, involving a 600% increase for the introduction of programmability. The reality is that in an overall FPGA implementation, the penalty is nowhere as harsh as this, but it has to be taken into consideration in terms of ultimate system cost.

It is the ability to program the FPGA hardware after fabrication that is the main appeal of the technology as it provides a new level of reassurance in an increasingly competitive market where 'right first time' system construction is becoming more difficult to achieve (Pina et al 2001). It would appear that assessment was vindicated as in the late 1990s and early 2000s, when there was a major market downturn, the FPGA market remained fairly constant when other microelectronic technologies were suffering. Of course, the importance of programmability has already been demonstrated by the microprocessor, but this represented a new change in the way programmability was performed.

1.5.1 Programmability and DSP

A clear advantage of FPGA technology is in terms of the use of its programmability to reduce the risk of incorrectly creating PCBs or evolving the manufactured product to later changes in standards. While this might have been true in the early days of FPGA technology, evolution in silicon technology has moved the FPGA from being a programmable *interconnection* technology to make it into a system component (Trimberger 2007). If the microprocessor or microcontroller was viewed as *programmable* system component, the current FPGA devices must also be viewed in this vein, giving us a different perspective on system programmability.

In electronic system design, the main attraction of microprocessors/ microcontrollers is that it considerably lessens the risk of system development by reducing design complexity. As the hardware is fixed, all of the design efforts can be concentrated on developing the code which will make the hardware work to the required system specification. This situation has been complemented by the development of efficient software compilers which have largely removed the need for designer to create assembly language; to some extent, this can absolve the designer of having a detailed knowledge of the microprocessor architecture. A lot of this process has been down to the software developer's ability to exploit an underlying processor architecture, the Von Neumann architecture.

However, this advantage has also been the limiting factor in its application to DSP. In the Von Neumann architecture, operations are processed sequentially, which allows relative straightforward interpretation of the hardware for programming purposes. However, this severely limits the performance in DSP applications which exhibit typically, high levels of

parallelism and in which, the operations are highly data independent allowing for optimisations to be applied. This cries out for parallel realization and while DSP microprocessors (here called DSPμs) go some way to address this situation by providing concurrency in the form of parallel hardware and software 'pipelining', there is still the concept of one architecture suiting all sizes of the DSP problem.

This limitation is overcome in FPGAs as they allow what can be considered to be a second level of programmability, namely programming of the underlying processor architecture. By creating an architecture that best meets the algorithmic requirements, high levels of performance in terms of area, speed and power can be achieved. This concept is not new as the idea of deriving a system architecture to suit algorithmic requirements has been the cornerstone of application-specific integrated circuit or ASIC implementations (Zuchowski et al 2002).

In high volumes, ASIC implementations have resulted in the most cost effective, fastest and lowest energy solutions. However, increasing mask costs and impact of 'right first time' system realization have made the FPGA, a much more attractive alternative. In this sense, FPGAs capture the performance aspects offered by ASIC implementation, but with the advantage of programmability usually associated with programmable processors. Thus, FPGA solutions have emerged which currently offer several hundreds of giga operations per second (GOPS) on a single FPGA for some DSP applications which is atleast an order of magnitude better performance than microprocessors.

1.5.2 FPGAs

Custom computing platforms contain several FPGAs, to provide reconfigurability without penalties such as hardware modifications or timely programming. Programming of FPGAs takes on the order of milliseconds through software configuration (Jiang Xiaoyan et al 2010). Depending on the application size, different sized FPGAs give the designer the flexibility to increase or decrease the resources as needed. This thesis bases the design work around Xilinx FPGAs.

FPGAs are integrated circuit devices that can be programmed by a user with a Hardware Description (HDL) language, such as verilog or VHDL for any digital design. FPGAs are an alternative to ASIC devices. FPGAs have advantages over ASIC such as cost and flexibility in design. FPGAs are programmable devices and their architecture consists of columns and rows of configurable logic blocks (CLBs) surrounded by I/O cells.

In order to connect signals between CLBs, routing resources and programmable interconnects lay between the logic blocks (Naous et al 2008). The basic nature of FPGAs presents a general set of resources which allows the designer to configure the logic blocks, routing, and I/O cells for a tailored application that runs at the speed of hardware, yet can be easily modified as software.

Each FPGA vendor has its own FPGA architecture, but in general terms they are all variations of that shown in Figure 1.6. Apart from configurable logic blocks, configurable I/O blocks, and programmable interconnect, there will be clock circuitry for driving the clock signals to each logic block, and additional logic resources such as ALUs, memory, and decoders may be available.

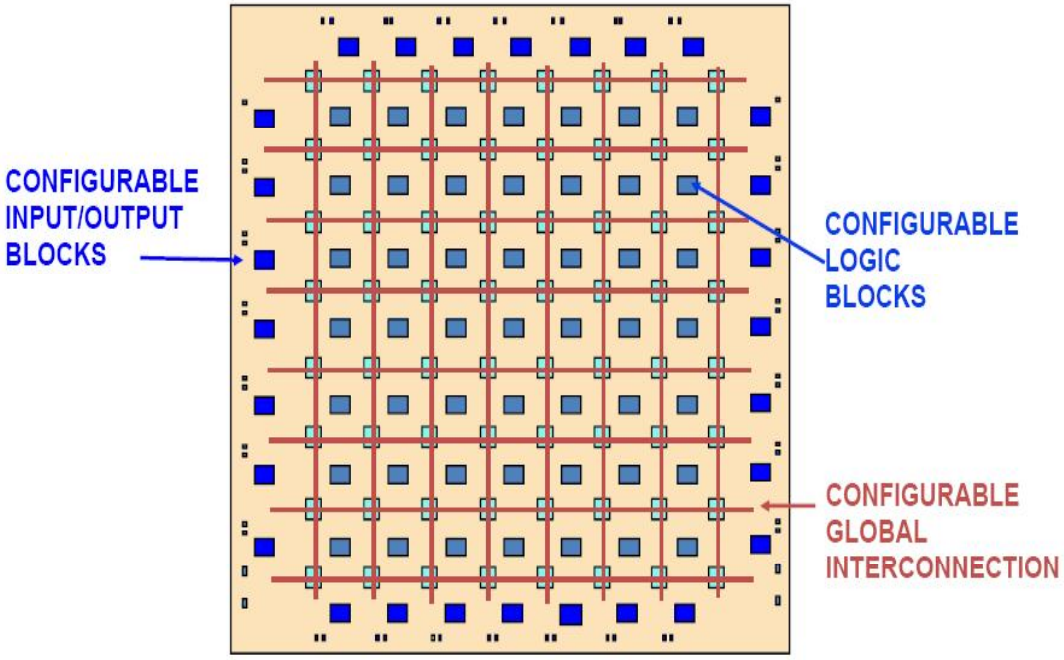

Figure 1.6 Field programmable gate array

The two basic types of programmable elements for an FPGA are Static RAM and anti-fuses. Configurable Logic Blocks contain the logic for the FPGA. In a large grain architecture, these CLBs will contain enough logic to create a small state machine. In a fine grain architecture, more like a true gate array ASIC, the CLB will contain only very basic logic.

The Configurable I/O Block is used to bring signals onto the chip and send them back off again. It consists of an input buffer and an output buffer with three state and open collector output controls. The interconnect of an FPGA is very different from a Complex Programmable Logic Device (CPLD), but is rather similar to that of a gate array ASIC. There are long lines which can be used to connect critical CLBs that are physically far from each other on the chip without inducing much delay. They can also be used as buses within the chip. There are also short lines which are used to connect individual CLBs which are located physically close to each other.

As the FPGA architecture evolves and its complexity increases, CAD software has become more mature as well. Today, most FPGA vendors provide a fairly complete set of design tools that allow automatic synthesis and compilation from design specifications in hardware specification languages, such as Verilog or VHDL, all the way down to a bit stream to program FPGA chips.

Design constraints typically include the expected operating frequencies of different clocks, the delay bounds of the signal path delays from input pads to output pads (I/O delay), from the input pads to registers (setup time), and from registers to output pads (clock-to-output delay). In some cases, delays between some specific pairs of registers may be constrained.

The second design input component is the choice of FPGA device. Each FPGA vendor typically provides a wide range of FPGA devices, with different performance, cost and power tradeoffs. The designer may start with a small (low capacity) device with a nominal speed-grade. But, if synthesis effort fails to map the design into the target device, the designer has to upgrade to a high-capacity device. Similarly, if the synthesis result fails to meet the operating frequency, it is necessary to upgrade a device with higher speed-grade. In both the cases, the cost of the FPGA device will increase by 50% or even by 100%. Thus, better synthesis tools are required since their quality directly impacts the performance and cost of FPGA designs (Deming Chen 2006).

1.6 FPGA DESIGN FLOW

The introduction of automated tools and progressively advanced configurable logic devices has facilitated the development environment for custom computing machines. Conventional methods of programmable logic

design consist of gate-level designs and schematic capture using complex CAD tools. With current design technology, hardware description language (HDL) compilers and synthesis tools allow designers to make alterations at a higher, abstract level.

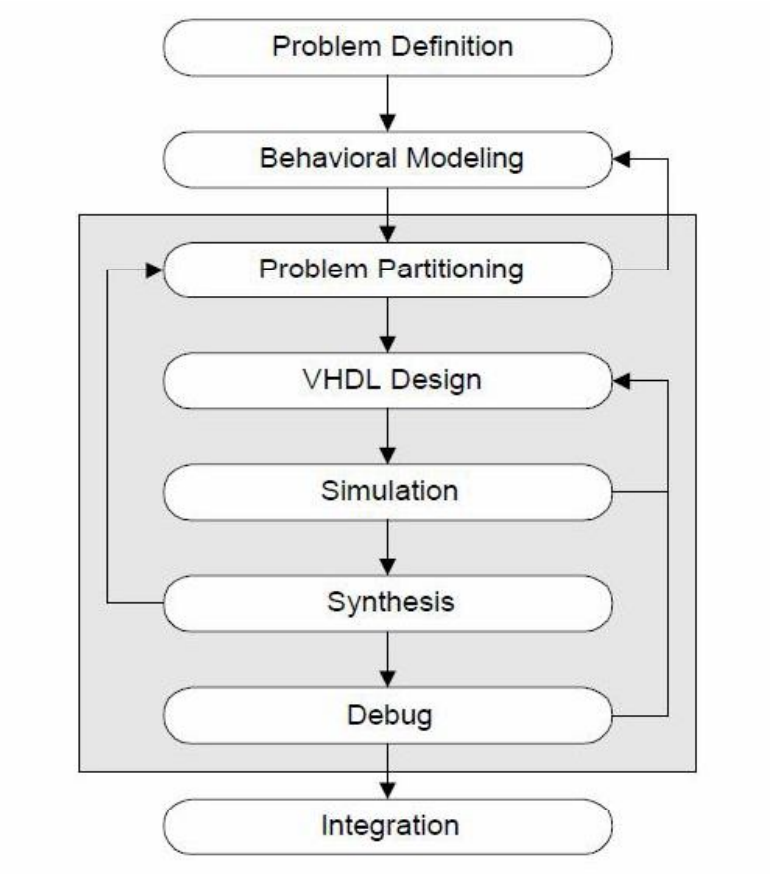

Figure 1.7 Application design process

Figure 1.7 illustrates the application design process. To facilitate the design process, the application designer should always generate a solid problem definition as seen in the first step. The designer may begin verification using high-level models with C, MATLAB, or behavioral VHDL to ensure problem definition compliance. In addition, the results obtained from the high-level verification may be used to confirm the implementation during later stages in the design process.

The problem partitioning step divides the algorithm among the processing elements in such a way that the partial computations contribute to the final result in some fashion. Partitioning generally requires consideration from three primary factors as described by Athanas et al (1995):time, area, and communication complexity. According to Gajaski (1988) time and area factors appear to be familiar problems.

The *time* factor relates to the amount of desired computation per clock cycle and *area* describes the amount of reconfigurable resources allocated to a given computation, to the total available reconfigurable resources within each processor board, and within each of the total number of processing elements on the board. Communications complexity involves careful consideration of how data paths between the partitioned algorithm should be routed.

The high-level, partitioned design can be implemented using a variety of FPGA CAD tools, although high-level language synthesis with VHDL provides an advanced development environment. Contemporary debugging tools exist for most environments that allow designers to step through the high-level code similar to other current high-level language integrated development environments.

The VHDL models undergo simulation to provide the designer with a level of correctness before the synthesis stage. Actual propagation delays in the Xilinx FPGAs are highly sensitive to the outcome of the place-and-route process and can have a disturbing effect on the application behavior (Athanas et al 1995). Debugging tools in the development environment allow designers to counter such problems introduced by the limited functional coverage of simulators.

As shown in Figure 1.7, several repetitions back to the coding stage of the design process may be necessary to achieve the desired results after synthesis. The design approach for partitioned algorithms should be done in an incremental fashion to alleviate design and integration time. When integrating smaller, working components debugging can be facilitated knowing that each part has been functionally tested on an individual basis (when possible).

Even though developments have been deemed as state-of-the-art, substantial amount of time must still be invested to produce optimal and high performance applications. Research efforts are focussed to improve automation of the stages shaded in gray of Figure 1.7.

1.6.1 Advantages of FPGA

Any technique can prove its success if and only if it is implemented in real-time. In order to have a successful hardware implementation, the various constrains viz. availability of equipment, durability for completion and viability of commercial transactions should be overcome. FPGAs are found to be the most cost-effective and least time consuming with simple solutions for designers to implement their findings in real-time environment. FPGAs are future-oriented building blocks which allow perfect customization of the hardware at an attractive price even in low quantities.

FPGA components available today have usable sizes at an affordable price. This makes them effective factors for cost savings and time-to-market when making individual configurations of standard products. A time consuming and expensive redesign of a board can often be avoided through application-specific integration of the desired circuit in the FPGA - an alternative for the future, especially for very specialized applications with only small or medium volumes. FPGA technology is

indispensable wherever long-term availability or harsh industrial environments are involved.

Another important aspect is long-term availability. Many component manufacturers do not agree on any long-term availability. This makes it difficult or impossible for the board manufacturer to support the product for more than 10 years. The remarkable advantage of FPGAs and their nearly unlimited availability lies in the fact that, even if the device migrates to the next generation, the code remains unchanged.

This is in accordance with norms like the (*European standards*) EN 50155 which prescribes that customized parts like FPGAs must be documented to allow reproduction and that the documentation and the source code must be handed out to the customer. In order to have a customized function, normally a device is programmed and is connected to the logic blocks through the transistors as interconnectors. The major benefit of using FPGA is two-fold, one is flexibility in design and the other is fast time in completion of the task.

1.7 AIM AND OBJECTIVES

The aim of the thesis is to design efficient low power multipliers with minimum hardware requirements suitable for FIR Filter implementation that can provide optimization of chip resources.

The main objectives of this thesis are :

- To carry out research with the classic optimization goal of minimizing power, area and logic depth of multiplier block
- To develop a novel shift and algorithm for low power and area efficient FIR filter

- To design a FIR Processor for filter processing
- To design a reconfigurable multiplier block on FIR filter using new MAG algorithm
- To use PSM architecture for designing efficient FIR filter
- To meet the demands for enhanced performance and reduced resource utilization for DSP applications
- To make suggestions on the future improvement of the system and development of the system in multimedia processing and real time applications.

1.8 OVERVIEW OF THE THESIS

A brief outline of the various chapters of the thesis is as follows :

Chapter 1 provides the information about the filters. It deals with digital filters, types, structures and real time applications of FIR filter. It highlights the strength of FIR filter over IIR filter in signal processing applications. It also gives insights of implementing digital signal processing systems using FPGA Technology. It also describes the developments of VLSI architecture, FPGA design flow and its advantages.

Chapter 2 provides a review of literature available in the field of design of FIR filters in FPGA. It gives a detailed study of the early work done in FIR architecture and multipliers. It also gives an insight into low power implementation and its applications in DSP.

Chapter 3 discusses the Novel shift and add algorithm for low power and area efficient FIR filter. The principles of multipliers and its architecture are discussed first. It describes the FIR filter based shift/add

multiplier. The method of multiplier block synthesis, multiplication hardware operation and its optimization goals are explained. The multiplier is designed using the proposed algorithm(N-RSG algorithm) and compares its synthesis results with RSG and FIR filter based shift and add multiplier. The results are discussed.

Chapter 4 presents a design of FIR processor on system-on-chip platforms for low power and high performance DSP applications. This chapter describes the System architecture for FIR processor. ALU, shifter unit, multiplier and adder unit and the instruction set are presented. The results are discussed.

Chapter 5 presents reconfigurable multiplier on FIR filter for SDR Receiver. The chapter describes efficient multiplier design. Binary Common Subexpression (BCS) algorithm, multiplier block synthesis and MAG extension and modifications are described. A new algorithm based on minimized adder graph is presented. The results are discussed.

Chapter 6 discusses the Reconfigurable multiplier for FIR filters. This chapter proposes a technique called Constant Shift Method (CSM) and Programmable Shift Method (PSM) instead of normal multiplier in FIR block. The architectural design and FPGA based implementation are presented.

Chapter 7 gives the conclusion of the thesis and explains the efficient multipliers which have been developed in this work. Moreover, it throws light on the importance of this thesis and gives valuable suggestions for future work.