CHAPTER I
INTRODUCTION

The widely used signal processing component 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. The design of the filter determines the loss and degree of phase shift that may occur during the insertion and the rejection. The filters are broadly classified into Analog filters and Digital filters.

1.1 Analog Filters

Filters are components that process signals in a frequency-dependent manner. The basic concept of a filter can be explained by examining the frequency dependent nature of the impedance of capacitors and inductors (Evangelos F, 2006). Consider a voltage divider where the shunt leg is reactive impedance. As the frequency is changed, the value of the reactive impedance changes, and also the voltage divider ratio changes. This mechanism yields the frequency dependent change in the input/output transfer function that is defined as the frequency response.

Filters have many practical applications. A simple, single-pole, low-pass filter (the integrator) is often used to stabilize amplifiers by rolling off the gain at higher frequencies where excessive phase shift may cause oscillations. A simple, single-pole, high-pass filter can be used to block DC offset in high gain amplifiers or single supply circuits. Filters can be used to separate signals, passing those of interest, and attenuating the unwanted frequencies.

An example of this is a radio receiver, where the signal you wish to process is passed through, typically with the gain, while attenuating the rest of the signal. In data conversion, filters are also used to eliminate the effects of aliases in A/D systems. They
are used in reconstructions of the signal at the output of a D/A as well, eliminating the higher frequency components, such as the sampling frequency and its harmonics, thus smoothing the waveform. There are a large number of texts dedicated to filter theory. No attempt will be made to go heavily into much of the underlying math: Laplace transforms, complex conjugate poles and the like, although they will be mentioned.

While they are appropriate for describing the effects of filters and examining stability, in most cases examination of the function in the frequency domain was more illuminating. An ideal filter will have an amplitude response that is unity (or at a fixed gain) for the frequencies of interest (called the pass band) and Zero everywhere else (called the stop band). The frequency at which the response changes from pass band to stop band is referred to as the cut off frequency.

Figure 1.1 (A) shows an idealized low-pass filter. In this filter the low frequencies are in the pass band and the higher frequencies are in the stop band. The functional complement to the low-pass filter is the high-pass filter. Here, the low frequencies are in the stop-band, and the high frequencies are in the pass band. Figure 1.1 (B) shows the idealized high-pass filter.

If a high-pass filter and a low-pass filter are cascaded, a band pass filter is created. The band pass filter passes a band of frequencies between a lower cutoff frequency, \( f_l \), and an upper cutoff frequency, \( f_h \). Frequencies below \( f_l \) and above \( f_h \) are in the stop band. An idealized band pass filter is shown in Figure 1.1 (C). A complement to the band pass filter is the band-reject, or notch filter. Here, the pass bands include frequencies below \( f_l \) and above \( f_h \). The band from \( f_l \) to \( f_h \) is in the stop band. Figure 1.1 (D) shows a notch response.
The idealized filters defined above, unfortunately, cannot be easily built. The transition from pass band to stop band will not be instantaneous, but instead there will be a transition region. Stop band attenuations will not be infinite. The five parameters of a practical filter are defined in Figure 1.2, opposite.

The cutoff frequency (fc) is the frequency at which the filter response leaves the error band (or the −3 dB point for a Butterworth response filter). The stop band frequency (Fs) is the frequency at which the minimum attenuation in the stop band is reached. The pass band ripple (Amax) is the variation (error band) in the pass band response. The minimum pass band attenuation (Amin) defines the minimum signal attenuation within the stop band. The steepness of the filter is defined as the order (M).
of the filter. M is also the number of poles in the transfer function. A pole is a root of the denominator of the transfer function. Conversely, a zero is a root of the numerator of the transfer function. Each pole gives a $-6 \text{ dB/octave or } -20 \text{ dB/decade}$ response. Each zero gives a $+6 \text{ dB/octave, or } +20 \text{ dB/decade}$ response.

Note that not all filters will have all these features. For instance, all-pole configurations (i.e. no zeros in the transfer function) will not have the ripple in the stop band. Butterworth and Bessel filters are examples of all-pole filters with no ripple in the pass band.

Typically, one or more of the above parameters will be variable. For instance, if you were to design an anti-aliasing filter for an ADC, you will know the cutoff frequency (the maximum frequency that you want to pass), the stop band frequency,
(which will generally be the Nyquist frequency (\(= \frac{1}{2} \text{ the sample rate}\)) and the minimum attenuation required (which will be set by the resolution or dynamic range of the system). You can then go to a chart or computer programs to determine the other parameters, such as filter order, \(F_0\), and \(Q\), which determines the peaking of the section, for the various sections and/or component values.

It should also be pointed out that the filter will affect the phase of a signal, as well as the amplitude. For example, a single-pole section will have a 90° phase shift at the crossover frequency. A pole pair will have a 180° phase shift at the crossover frequency. The \(Q\) of the filter will determine the rate of change of the phase.

An analog filter can be defined as a filter which operates on continuous-time signals. Particularly, an analog filter is characterized by their impulse response with respect to time. Normally the analog filters are described by a differential equation. In order to compute the transfer function Laplace transform is used instead of \(z\)-transform. Since the sampling-rate is allowed to go to infinity, analog filters can be considered as 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.

1.1.1 Applications of analog filters

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

2. The combining and later separation of multiple telephone conversations onto a single channel

3. The selection of a chosen radio station in a radio receiver and rejection of others and much more
Normally, any passive linear analog filters are described with the linear differential equations. The analog filters are fabricated with capacitors, inductors and occasionally the resistors. These analog filters are designed such that it will operate on continuously varying the signal that is analog in nature.

The role of analog filters is very crucial, particularly in the telecommunication fields. In the initial stage, the analog filters were connected along with the transmission lines and hence the transmission line theory end up with the filter theory. In order to filter the mechanical vibrations or acoustic waves, one can design a linear analog filters with the help of mechanical components. If transducers are added in such mechanical filters, then 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 of most active in avoiding wound components required by any passive topology. Analog filters find its applications, especially in the lower order simple filtering tasks and are often at higher frequencies where digital technology is still impractical, or at least, less cost effective. However the recent years, it is often preferred to carry out filtering in the digital domain where complex algorithms are much easier to implement.

1.2 Digital Filters

A digital filter can be defined as a filter which operates on digital signals, such as sound represented inside a computer. In order to convert an output digital signal into analog form, it is necessary to perform additional signal processing to obtain the perfect result and is demonstrated in Figure 1.3.
The process of converting an analog signal into digital form is performed by sampling with a finite sampling frequency (Bull, D.R 1991). If an input signal contains frequency components higher than half the sampling frequency, it will cause distortion to the original spectrum.

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. 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. Analog Filters, Software-Based and Hard-wired Digital Filters Owing to the way that analog and digital filters are physically implemented, an analog filter is inherently more size and power-efficient, although more component-sensitive, than its digital counterpart.
It can be implemented in a straightforward manner. In general, as signal frequency increases, the disparity in efficiency increases. Characteristics of applications where digital filters with more size and efficient than analog filters are: linear phase, very high stop band attenuation, 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.

For example, Intersil Corporation offers a family of standard digital filter products with several others in development. Some hardware oriented digital filters can now sample at rates approaching 75 Mega Samples per Second (MSPS). 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. In other words, 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 filters uses phase shift as a function of frequency,
which is a time domain operation.

A 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. A filter is designed with a frequency domain impulse response
which is as close to the desired ideal response as 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 to the coefficients
of the filter.

1.2.1 Implementation of Digital filters

The realization of digital filter results from the transfer function. The goal of
digital filter design is to find a suitable structure which passes requirements and
minimizes numerical and quantization errors. There are few types of realizations and
their combinations which can be used for implementation:

1. Universal processor.

2. Digital signal processor (DSP processor) with instruction set optimized for
signal processing.

3. Special hardware such as Field programmable gate array (FPGA) or
Application-specific integrated circuit (ASIC).
1.2.2 Realization with Universal processor

Instruction sets of universal processors are optimized for general algorithms so they are not advantageous to signal processing applications. Algorithms of digital signal processing commonly use multiplication and addition. These operations are usually processed in a few clock cycles in universal processors so the computation performance is low. Some universal processors use Von Neumann architecture with memory space that is shared for data and program, so the data transfers between memory and CPU are slower. There are usually not integrated any multipliers or MAC units on chip to improve the performance. Universal processor is possible to use only for signal processing application with low demands, in cases when is not cost effective to use other integrated circuit for signal processing.

1.2.3 Realization with DSP processor

The architecture of modern DSP processor is usually based on modified Harvard architecture with multi-buses system (Hutchings, B. L 2001). Combination of advanced Harvard architecture, high speed memory integrated on chip and optimized instruction set is the reason why DSP processors are suitable for digital signal processing.

The multi-buses system usually consists of one program memory bus and several data memory buses. The memory of DSP processor is divided into program and data memory. There are separate address generators for program and data memory. This architecture allows executing numeric instructions in one clock cycle and rapidly increases computation performance. The main control center of DSP processor is central processing units (CPU), which executes the instructions stored in program memory. The typical 16-bit DSP processor has following features:
1) CPU features

1. Multi-buses architecture with several data and address buses.
2. 40-bit arithmetical logic unit (ALU) with 40-bit shift register and independent 40-bit accumulators.
3. Parallel multiplier (typically 17x17 bits) connected with 40-bit adder, which allows multiply and accumulate operation in one clock cycle (MAC units).
4. Separated address generators for program and data memory

2) Instruction set

b. Instructions which supports 32-bit operands.
c. Support of parallel executions of instructions (write, read).
d. Instructions for parallel reading of operands.

3) Memory

Memory are divided into data, program and I/O space (modified Harvard architecture).

4) Other features

a) The clock frequency up to a few of gigahertz.
b) Multi-core architecture in high performance DSP processor.
c) Peripherals integrated in some DSP processors (UART, USB etc.).

1.3 Realization with FPGA

The most common FPGA architecture consists of an array of configurable logic blocks (CLBs), Input / Output blocks and reconfigurable matrix of interconnects CLBs are used for realization of main logic in FPGA. It typically consists of 4-input Lookup Tables (LUT), multiplexors and flip-flops. LUT is used for realization of
combinational logic. It is a 16-bit configurable memory that is capable of realizing all 4-input combination logical functions. The output of the LUT can be registered to realize sequential functions.

In modern FPGAs, CLBs with more than 4 inputs are integrated into the single slices. Multiplexers in CLB can be also used for realization of logic function with more than 4 inputs, they allow combined output of LUTs CLBs for realization of main logic in FPGA. It typically consists of 4-input Lookup Tables (LUT), multiplexers and flip-flops.

LUT is used for realization of combinational logic. It is a 16-bit configurable memory which is capable to realize all 4-input combination logical functions. The output of the LUT can be registered to realize sequential functions. In modern FPGAs are also CLBs with more than 4 inputs integrated. Multiplexors in CLB can be also used for realization of logic function with more than 4 inputs, they allow the combination of the outputs of LUTs in FPGA is higher.

On the other hand, SRAM memory does not remember the configuration without the power, thus the configuration must be loaded to the SRAM before the start of the FPGA function. It increases boot-time and power consumption. Nonvolatile FPGAs uses anti fuses or some type of nonvolatile memory (FLASH, EEPROM). The advantages of this solution are lower power consumption and higher resistance to radiation.

Beside basic blocks described above, there are also some others block integrated in most of the FPGAs. Delay-Locked Loops are used for clock signal generation. They synchronize clock signal in whole FPGA a can be also used as clock divider or multiplier. In some application is necessary to use a lot of memory. In these cases, there are Block RAMs integrated in FPGA. Block RAM is usually a dual-port RAM with independent control signals for each port. Because FPGAs is widely used for
DSP application, there are multipliers, adders with Carry chain architecture, MAC unit etc. integrated in some FPGAs to increase speed of computation.

### 1.3.1 FPGA vs. DSP realization

The main advantage of DSP processor in comparison with FPGA is that the implementation is simple. DSP processor can be programmed in assembler or C language. On the other hand, the computation performance is still limited by the architecture of DSP processor and program realization.

### 1.4 Types of Digital Filters

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

1. Finite Impulse Response digital filters (FIR) and
2. Infinite Impulse Response digital filters (IIR).

**Finite Impulse Response digital filters (FIR filters)**

An FIR filter is also called as recursive filter in which in addition to input values it also uses previous output values (Evangelos F, 2006). 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 \((x_\nu, x_{\nu-1}, x_{\nu-2}, \ldots)\) but also terms in \(y_{\nu-1}, y_{\nu-2}, \ldots\).

From this explanation, FIR 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 using a recursive filter generally requires a much lower order filter, and therefore fewer terms to be evaluated by the
processor, than the equivalent non-recursive filter. The term digital filter arises because these filters operate on discrete-time signals.

- The term finite impulse response arises because the filter output is computed as a weighted, finite term sum, of past, present, and perhaps future values of the filter input, i.e.,

\[
y[n] = \sum_{k=-M_1}^{M_2} b_k x[n - k]
\]  

(1.1)

Where both $M_1$ and $M_2$ are finite

- One of the simplest FIR filters that may be considered is a 3–term moving average filter of the form

\[
y[n] = \frac{1}{3} (x[n + 1] + x[n] + x[n - 1])
\]  

(1.2)

- An FIR filter is based on a feed-forward difference equation– Feed-forward means that there is no feedback of past or future outputs to form the present output, just input related terms.

The basic characteristics of Finite Impulse Response (FIR) filters are:

1. Linear phase characteristic

2. High filter order (more complex circuits) and

3. Stability

4. There is also a need to be able to store filter coefficients in memory
5. FIR designs by windowed sink or frequency sampling are simple

6. FIR equiripple designs allow better control of frequency breakpoints and of pass band and stop band ripple.

7. It is more complicated to implement, but that was solved 30 years ago...

8. More coefficients: closer to desired response

9. Wider transition band: less ripple in pass band, more stop band attenuation.

1.4.1 Infinite Impulse Response digital filters (IIR).

Here, current output \( y_n \) is calculated solely from the current and previous \textit{input} values \( x_n, x_{n-1}, x_{n-2} \ldots \). This type of filter is also said to be non-recursive filters.

The basic characteristics of Infinite Impulse Response (IIR) are:

a. Non-linear phase characteristic

b. Low filter order (less complex circuits) and

c. Resulting digital filter has the potential to become unstable.

Both types have some advantages and disadvantages that should be carefully considered when designing a filter. 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 filters has linear phase characteristics 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 ranges 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
industries. Therefore, it is necessary that a filter has linear phases characteristic to prevent loosing important information.

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 characteristic is of the essence, FIR filters should be used as they have linear phase characteristic. Such filters are of higher order and more complex. Otherwise, when it is only frequency responses 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.

IIR filters are digital filters with infinite impulse responses. Unlike FIR filters, they have the feedback and are known as recursive digital filters, hence due to this, IIR filters have far 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 that need 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. IIR filters can be designed using different methods. One of the most commonly used is via the reference analog prototype filter. This method is the best for designing all standard types of filters such as low-pass, high-pass, band-pass and band-stop filters.

FIR filters can have linear phase characteristics, 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 characteristics is not necessary,
such as speech signal processing, FIR filters are not good solutions. IIR filters should be used instead. The resulting filter order is considerably lower for the same frequency response.

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 can be computed. For instance, if the filter order is 10, it means that it is necessary to save 10 input samples plus 10 output samples preceding the current sample. All these 21 samples will affect the next output sample.

The IIR filter transfer function is a ratio of two polynomials of the complex variable \( z^{-1} \). The numerator defines the location of zeros, whereas the denominator defines location of the pole of the resulting IIR filter transfer function. The most commonly used IIR filter design method uses reference analog prototype filters. It is the best method to use when designing standard filters such as low-pass, high-pass, and band pass and band-stop filters. The initial step for filter design process starts with specification and requirements of the desirable IIR filter. A type of reference analog prototype filter to be used is specified according to the specifications and after that everything is ready for analog prototype filter design.

The next step in the design process is scaling of the frequency range of analog prototype filter into desirable frequency range. This is how an analog prototype filter is converted into an analog filter. After the analog filter is designed, it is time to go through the last step in the digital IIR filter design process. It is conversions from analog to digital filter. The most popular and most commonly used converting methods is bilinear transformation methods. The resulting filter, obtained in this way, is always stable. However, instability of the resulting filter, when bilinear transformations may be caused only by the finite word-length side-effect.
1.4.2 STRENGTH OF FIR FILTERS OVER IIR FILTER

Because 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 with poles, an 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 coefficients are symmetric, as is the case in most standard filtering applications.

A FIR’s implementation noise characteristics are easy to model, especially if no intermediate truncation is used. Generally most Intersil filter IC have more coefficient bits than data bits. An IIR filters 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. An FIR filter is always stable. FIR filters also allow development of computationally efficient architectures in decimating or interpolating applications.

1.5 Advantages of FIR Filters

Compared to IIR filters, FIR filters offer the following advantages:
1. They can easily be designed to be linear phase. Put simply, linear-phase filter, delay the input signal. But don’t distort its phase.
2. They are simple to implement.
3. Calculation of FIR can be done by looping a single instruction.
4. They are suited to multi-rate applications. By multi-rate, either "decimation" (reducing the sampling rate), "interpolation" (increasing the sampling rate), or both are meant. 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 it that output will be discarded (so the feedback will be incorporated into the filter).

5. 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.

6. They can be implemented using fractional arithmetic.

7. 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 when using fixed-point DSP's, because it makes the implementation much simpler.

1.6 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 such that high order filters are 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.

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. An IIR filter has both zeros and poles in its transfer function and can be unstable. Its impulse response of the theoretical 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.

1.7 Features of VLSI

Twenty years ago at the first conference in this series, there was consensus that the best way to apply VLSI technology to information processing problems was to build parallel computers from simple VLSI building blocks. Four of the seven papers in the architecture session addressed this topic [Sequin, 1979, Mago 1979, Brown, 1979, Davis, 1979] including two papers on trees machines, one on cellular automata, and one on data flows. Considerable research over the past two decades focused on the design of parallel machines and many valuable research contributions were made. The mainstream computer market, however, was largely unaffected by this research. Most computers today are uniprocessor and even large servers have only modest numbers (a few 10s) of processors.

The situation today is different. It is again to anticipate an exponential increase in the number of devices per chip. However, unlike 1979, there are few opportunities remaining to apply this increased density to improve the performance of uniprocessor. Adding devices to modern processors to improve their performance is already well beyond the point of diminishing returns. There are few credible alternatives to use the
increased device count, other than to build additional processors. Because the chip area of contemporary machines is dominated by memory, adding processors (without adding memory) boosts efficiency by giving a large return in performance for a modest increase in total chip area. To realize the potential of such fine-grain machines to convert VLSI density into application performance the challenges of locality, overhead and software should be addressed.

1.7.1 Twenty Years of VLSI Architecture

In 1979, anticipated scaling of VLSI technology favored the development of regular machines that exploited concurrency and locality and that were programmable. Twenty years was expected to bring more than a thousand fold increase in the number of grids, and hence the number of devices that could be economically fabricated on a chip. Clearly concurrency (parallelism) would need to be exploited to convert this increase in device count to performance.

Locality was required because the wire bandwidth at the periphery of a module was scaling only as the square root of the device count, much slower than the 2/3 power required by Rent’s rule [LanRus, 1971]. Also, even in 1979 it was apparent that wires, not gates, limited the area, performance, and power of many modules. The issue of design complexity motivated regularity and programmability. Designing an array of identical, simple processing nodes is an easier task than designing a complex multi-million transistor processor. A programmable design was called for so that the mounting design costs could be amortized over large numbers of applications.

In the twenty years, since the first conference, many of the hard problems of parallel machine design have been solved. The design of fast, efficient networks to connect arrays of processors together [Dally,1992 and DYN,1997] was realized. Mechanisms that allow processors to quickly communicate and synchronize over these networks have been developed.
1) A grid is an intersection of a horizontal and vertical wire. Hence the number of grids on a chip is the square of the number of wiring tracks that fit along one edge of a chip. As VLSI chips are limited by wiring, not devices, the number of grids is a better measure of complexity than the number of transistors.

2) The key here is to match the design of the network to the properties of the implementation technology rather than to optimize abstract mathematical properties of the network. Odds of programming parallel machines have been demonstrated. Research machines were constructed to demonstrate the technology, provide a platform for parallel software research, and solve the engineering problems associated with its realization [Seitz, 1985, NWD, 1993, SBSS1993]. The results of this research resulted in numerous commercial machines [Scott, 1996] that form the core of the high-end computer industry today.

3) MIMD machines are preferable to SIMD machines even for data-parallel applications. Similarly, general-purpose MIMD machines are preferable to systolic arrays, even for regular computations with local communication. Bit-serial processors loose more in efficiency than they gain in density.

4) A good general-purpose network (like a 3-D torus) usually outperforms a network with a topology matched to the problem of interest (like a tree for divide and conquer problems). It is better to provide a general-purpose set of mechanisms than to specialize a machine for a single model of computation.

While successful at the high end, parallel VLSI architectures have had little impact on the mainstream computer industry. Most desktop machines are uniprocessor and even departmental servers contain at most a few 10s of processors. Today’s mainstream microprocessor chips are dense enough to hold 1000 of the 8086s or
68000s of 1979, yet all of this area is used to implement a single processor. By many objective measures this would clearly be a more efficient architecture.

There are three main reasons for this course of events:

a) There was considerable opportunity to apply additional grids to improve the performance of sequential processors

b) Software compatibility favored sequential machines

c) High-overhead mechanisms used in early parallel machines motivated a coarse granularity of both hardware and software

In 1979 there was more than a factor of 100 difference in performance between the best microprocessors (0.5MIPS, 0.001MFLOPS) and a high-end CPU such as used in the Cray 1 (70MIPS, 250MFLOPS [Russel78]) or IBM 370. Only a small part of this difference, about a factor of 3, was due to the difference in gate delay between bipolar and MOS technology. Most of the difference was due to increased gate count that was used to aggressively pipeline execution and to exploit parallelism.

Between 1979 and 1999 microprocessors closed this gap by incorporating most of the advanced features pioneered in mainframes and supercomputers in the 1960s and 70s as well as a few new tricks. On-chip caches, on-chip memory management units, pipelined multipliers and floating-point units, multiple instruction issue, and even out-of-order instruction issue were added to processors during this period. The addition of these features, along with quadrupling the word width from 16-bits to 64-bits created a sufficient appetite for grids without resorting to explicit parallelism.

During the past 20 years, the performance of a high-end microprocessor increased from 0.5MIPS to 500MIPS, about a factor of 1000. It is observed that clock frequency increased by a factor of 80: a factor of 20 is due to gate delay and a factor of 4 is due to reducing the number of gates per clock. The remaining factor of 12.5
reflects a reduction in clocks per instruction (CPI) from about 10 for the un-pipelined microprocessors of 1979 to just under 1 for today’s 3- and 4-way multiple-issue superscalar processors.

d) They are an example of processors that are not yet to the point of diminishing returns.

Software evolved considerably during the last 20 years: from text-based applications running on proprietary operating systems (like VMS and MVS) to graphics-based applications running on third-party operating systems (like Windows and Unix). What remained constant, however, was the sequential nature of this software. Manufacturers wanting to sell machines that would run existing software needed to build fast sequential machines.

In summary, for most of the 80s and 90s software compatibility motivated building sequential machines; there was little economic advantage to coarse-grain parallel machines; and there were many obvious ways to use more grids to make a sequential CPU faster. Given this environment, it is no surprise that industry responded by making sequential CPUs faster and only building coarse-grain parallel machines.

1.7.2 The Next 20 Years

The next 20 years promise to be exciting ones in the area of VLSI architecture with a major revolution in the architecture of mainstream processors. Continued scaling of technology will give us yet another thousand fold increase in chip density. As in 1979 it is natural to think of developing architectures that are programmable and exploit concurrency and locality to exploit this increased density. Unlike 1979, however, there are three reasons why a revolution is likely now: First, sequential processors are out of steam. While clever architects will undoubtedly continue to develop new methods to squeak a few percentage points more performance from sequential processors, clearly the point of diminishing returns are well past.
1) Large amounts of chip area are spent on complex instruction issue logic and branch prediction hardware while yielding small improvements in performance.

2) To continue improving performance geometrically each year, there is no alternative except to exploit explicit parallelism.

3) To be precise, the point of diminishing returns is where the incremental return in performance per unit cost (area), drops below the incremental return for an alternative expenditure of area. Assuming linear speedup, for example, this point happens when. For actual systems, with sub linear speedup, it happens slightly later [Dally86].

4) Ironically, these heroic efforts to extend instruction-level parallelism (ILP) usually involve converting data parallelism, which is easy to exploit on a multicomputer, to ILP which is hard to exploit on a uniprocessor, for example by loop unrolling.

Second, technology scaling is rapidly making wires, not transistors, the performance limiting factor. Each time line widths halve, the devices get approximately twice as fast and a minimum width wire of constant length gets four times slower. Contemporary architectures, which depend on global control, global register files, and a linear memory hierarchy, don’t scale well with large wire delays. Instead, these slow wires motivate architectures that exploit locality by operating on data near where it is stored. Architectures that distribute a number of simple yet powerful processors throughout the memory, for example, are capable of exploiting this type of locality.

Finally, the cost of machines today is dominated by memory. Twenty years of increasing the memory capacity of machines to match the performance of the microprocessor has left us with computers that are mostly memory. A competent
500MHz, pipelined, in-order, dual-issue processor with floating point can be built today in an area of about 10M grids (about 1/20 of a modern 0.2mm chip). In contrast, a memory system with a 512MByte main memory and a 512KByte cache takes about 200M grid chips. Such a machine has a 400:1 ratio of memory to processor area. A parallel machine can turn over its expensive memory system more quickly than a sequential machine with the same amount of memory and hence can offer more economical operation.

That is the parallel machine ties up less time on the costly memory per problem and hence solves more problems. By adding inexpensive, yet capable, processors without adding more memory, the cost of the machine is increased modestly while its performance is multiplied considerably, even if speedup is far less than linear.

5) Here physical locality is refereed, rather than the more conventional notions of spatial and temporal locality that refer to address streams, not to physical locations.

6) Cost here is silicon area divided by yield. This reflects the true recurring cost of manufacturing and ignores the large difference in margin between commodity memories and certain (but not all) processors.

7) It is assumed that each processor is duplicated to keep the yield of these chips comparable to that of DRAM chips with the same capacity while preserving locality. More efficient redundancy schemes are possible.

8) It is also assumed that the processor is in the center of the tile, not on the edge.

9) One can trade some of this excess bandwidth for latency by making one or more of the metal layers thick with wider than minimum design rules. Assuming a constant clock rate, doubling the width, spacing, and thickness of the wires doubles their velocity, halving both latency and bandwidth of the chip boundary. A large fraction of the memory references from these processor must be captured on-chip. The architecture sketched here is more economical than a uniprocessor with the same
amount (8GBytes) of memory. Its cost (chip area) is only 20% greater than the uniprocessor

10) At the same time, it has 1024 processors to apply to the problem rather than one. At any speedup greater than 1.2, the fine-grain computer will be able to solve more problems than a uniprocessor

11) For typical speedups, in the range of 10-100, the cost-performance advantage is significant. The economy here comes from the use of inexpensive processors to more quickly re-use the expensive memory system.

1.7.3 Architecture Challenges

Considerable work is needed before the vision of a fine-grain machine becomes a reality. Two areas in particular demand attention: managing locality and reducing overhead. Methods must be developed to manage the data locality for irregular problems with significant global interaction. While the vast majority of real programs has an irregular structure that is data-dependent and often time-varying, most work to date on data layout has addressed only regular data structures.

A run-time mechanism to migrate data and tasks that simultaneously balances the load and minimizes communication is required. Such a mechanism will involve both hardware, to identify communication patterns and to remap addresses, and software, to implement migration and replication policies. As with much of computer architecture, the art is in defining the right interface between these two components. To simplify the task of extracting parallelism at fine granularity communication and synchronization overhead must be reduced to a minimum. Low overhead (a few clock cycles) makes it feasible to make every few loop iterations (50-100 clock cycles) a separate task. Reducing the task size greatly increases the amount of available parallelism and hence makes it easier to parallelize programs.
It is also important that synchronization be made as specific as possible. Many parallel programs have poor speedup because they are over synchronized, performing a barrier synchronization every loop iteration. By synchronizing on individual data elements, rather than on the flow of control, the end of one loop iteration can be overlapped with the beginning of the next iteration, eliminating a sequential bottleneck and greatly increasing performance.

Much progress has already been made on the design of low-overhead mechanisms. The J-Machine is able to perform remote task invocations in 50 clock cycles [NWD93]. The M-Machine offers single-cycle communication and synchronization between on-chip tasks [KDM+98]. More work is needed to further reduce overhead while at the same time offering a simple, abstract model to the programmer. Locality and overhead are fundamental issues that will determine how the next generation of computer systems will be organized.

Such fundamental questions cannot be addressed by the benchmark parameter studies that have become very popular in the computer architecture community. While such parameter studies are good at fine-tuning a machine, they are not suitable for the large scale exploration of the design space where programs must be significantly restructured to exploit a new design. Tuning is needed for fine-grain machines as well. Once the major issues have been resolved, parameter studies can be conducted to determine the appropriate ratio of processors to memory. If the 20% cost is considered too high, one can do nearly as well with 16 processor pairs on a chip, each with 64 m-bytes of memory, giving a cost increase of only 5%. As an aside, because it is better able to exploit locality, the fine-grain computer is able to outperform the uniprocessor even on completely sequential programs that are limited by memory latency [Burger97]. to evaluate alternative clustering strategies, and the division of memory resources across the memory hierarchy.
1.8 FPGA

The most common FPGA architecture consists of an array of configurable logic blocks (CLBs), Input / Output blocks and reconfigurable matrix of interconnects CLBs are used for realization of main logic in FPGA (Hutchings, B.L 2001). It typically consists of 4-input Lookup Tables (LUT), multiplexors and flip-flops. The LUT is used for realization of combinational logic. It is a 16-bit configurable memory which is capable to realize all 4-input combination logical functions. The output of the LUT can be registered to realize sequential functions.

In modern FPGAs are also CLBs with more than 4 inputs integrated. Multiplexors in CLB can be also used for realization of logic function with more than 4 inputs, they allow the combination of the outputs of LUTs CLBs are used for realization of main logic in FPGA. It typically consists of 4-input Lookup Tables (LUT), multiplexors and flip-flops. The LUT is used for realization of combinational logic. It is a 16-bit configurable memory which is capable to realize all 4-input combination logical functions. The output of the LUT can be registered to realize sequential functions. In modern FPGAs there are also CLBs with more than 4 inputs integrated. Multiplexors in CLB can be also used for realization of logic function with more than 4 inputs, they allow combine the output of LUTs FPGAs is higher.

On the other hand SRAM memory does not remember the configuration without the power, thus the configuration must be loaded to the SRAM before the start of the FPGA function. It increases boot-time and power consumption. Nonvolatile FPGAs use anti fuses or some type of nonvolatile memory (FLASH, EEPROM). This solution provides the advantages of lower power consumption and higher resistance to radiation. Beside basic blocks described above, there are also some others block integrated in most of the FPGAs. Delay-Locked Loops are used for clock signal generation. They synchronize clock signal in whole FPGA a can be also used as clock divider or multiplier. In some application is necessary to use a lot of memory. In these
cases, there are Block RAMs integrated in FPGA. Block RAM is usually a dual-port RAM with independent control signals for each port.

Because FPGAs are widely used for DSP applications, there are multipliers, adders with Carry chain architecture, MAC unit etc. integrated in some FPGAs to increase speed of computation.

The main advantage of DSP processor in comparison with FPGA is a simple implementation. DSP processor can be programmed in assembler or C language. On the other hand the computation performance is still limited by the architecture of DSP processor and program realization. FPGA stands for Field Programmable Gate Array. That’s the chip, which is programmed by user to perform the desired functionality.

There are many possibilities of programming (types of FPGA):

1. Anti-fuse technology - programmed only once
2. Flash based - programmable several times
3. SRAM based - programmable dynamically

The last possibility is dominating technology. It allows very fast, unlimited in system reprogramming. This type of FPGA chip is used for irradiation tests. Architectures of FPGA from different vendors may differ, but the main idea is almost the same. There will be shown on the example of Xilinx Spartan-IIE chip. FPGA architecture consists of: Flexible, programmable Configurable Logic Blocks (CLB), which provide the functional elements of constructing logic. Programmable Input/output Blocks (IOB), which provide the interface between the package pins and the internal logic. Delay-Locked Loops (DLL) for clock distribution. Dedicated internal memory (Block RAM). Varsatile multi-level interconnects structure. LBs form the central logic structure with access to all support and routings. IOB’s are located around the logic and memory elements for easy routing signals from and to chip.
Values stored in static memory cells control all the configurable logic elements and interconnect resources. These values must be loaded into the memory cells on power-up and can be reloaded to change the function of the device, almost any time. Logic Cell (LC). Two LCs form slice and two slices form CLB. Each LC includes a 4-input function generator, carry logic and a storage element. The function generator is implemented as 4-input Look Up Table (LUT).

Fig 1.4 Architecture of FPGA Spartan3 XC3S50 device.

The IOB features inputs and outputs that support a wide variety of I/O signaling standards. It consists of:

1. Three registers, which can work either as edge-triggered D-type flip-flop or as level-sensitive latches.
2. Programmable delay element in the input path, which eliminates pad-to-pad hold time.

3. Two multiplexers to control output from the FPGA.

**Delay-Locked Loop**

This block eliminates the skew between the clock input pad and internal clock-input pins throughout the device. Additionally delay is introduced such that clock edges reach internal flip-flops exactly one clock period after they arrive at the input.

**Block RAM**

That’s internal memory for used to store the data inside the chip. The word length can be configured by user.

**DDR memory**

The FPGA will be programmed with DDR interface. It will perform also test of this memory and report all the error to the PC computer. During experiment PC computer will perform FPGA memory configuration test and in case of SEU errors in configuration it will reprogram the FPGA. DDR memory will be refilled when too many errors will occur. Other components on FPGA board Other components like: voltage regulators, RS232C interface, oscillators seem to be less sensitive to the radiation. These elements will be tested passively during reset of experiments. In case of any problems, the board will be tested outside the tunnel to know what part is broken.

**Shields for electronic boards**

According to data collected in LEDs, optical fiber and TLS’s experiments there will be designed using proper shielding for neutron and gamma radiation. All proposed shields will be tested in accelerator tunnel, with all tests mentioned before.
The goal of this test is to design such shield for electronic board that gives the least number of errors.

**Configuration**

This is the process by which the bit stream of a design is loaded into the internal configuration memory of the FPGA. In this experiment, it’s done by JTAG port.

### 1.9 Advantages of FPGA

1. Any technique can prove its success if and only if it is has been implemented in real-time. In order to have a successful hardware implementation, the various constrains viz. Values of equipment, durability for completion and viability of commercial transactions should be overcome.

2. Field programmable Gate Arrays (FPGA) 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.

3. FPGAs are future-oriented building blocks, which allow perfect customization of the hardware at an attractive price even in low quantities.

4. FPGA components available today have usable sizes at an acceptable price. This makes them effective factors for cost savings and time-to-market when making individual configurations of standard products.

5. A time consuming and expensive Re design 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.

6. FPGA technology is indispensable wherever long-term availability or harsh industrial environments are involved. Another important aspect is long-term availability.
7. Many component manufacturers do not agree on any long-term availability. This makes it difficult or impossible for the board manufacturer to support his product for more than 10 years.

8. 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 describe 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.

9. In order to have a customized function, normally a device is programmed and is connected to the logic blocks through the transistors as interconnector. The major benefit of using FPGA has two fold, one is flexibility in design and the other one is fast time in completion of the task.
1.10 THESIS ORGANIZATION

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

**Chapter 1:** Provides information about the filters. It gives fundamental information based on filters. It deals with analog and digital filters, types and real time application of FIR filters, features and challenges in VLSI and advantage of the FPGA.

**Chapter 2:** Deals with the review of literature about the filters. It gives the detailed study of work before done and it reviews the filters to be applied or proposed.

**Chapter 3:** Provides information about the Design and Implementation of Low Power and Low Area FIR Filter Using CSM Architecture and also provides analyzed results.

**Chapter 4:** Design of a Low Power and Area Efficient New Reconfigurable Fir Filter for DSP Applications and analysis the area and power for different architecture.

**Chapter 5:** Discussion about the new Reconfigurable Architecture for Implementing High Speed FIR Filter. The architectural design and FPGA based implementation are presented.

**Chapter 6:** Concludes the overall work that has been done. It provides highlights on thesis work and suggestions for future work.