Design and Development of a Power Efficient Code for FIR Filter Coefficients Using Genetic Algorithms

A Thesis

Submitted in the partial fulfillment of requirements
for the award of the degree of

Master of Engineering

in

Electronic Instrumentation and Control Engineering

Submitted
by

SWARAJYA AGNIHOTRI
Regn.No-80651022

Under the esteemed guidance of:
Dr. YADUVIR SINGH
Associate Professor

Department of Electrical and Instrumentation Engineering
THAPAR UNIVERSITY
PATIALA (PUNJAB)-147004
July – 2008

CERTIFICATE

This is to certify that the thesis titled, "**Design and development of a power efficient code for FIR filter coefficients using genetic algorithms"**, being submitted by Swarajya Agnihotri, Regn. No.: 80651022, in partial fulfillment for the requirements of the award of the degree of Master of Engineering in Thapar University, Patiala, is a record of student's own work carried out under my supervision and guidance and this thesis has not been submitted to any other university or institute for award of any degree.

Date: **Swarajya Agnihotri**

Regn No. 80651022

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

Mentor & Supervisor

Dr. Yaduvir Singh
Associate Professor EIED,
Thapar University
Patiala

ConutersignedBy:

Dr. Smarajit Ghosh
Professor and Head
EIED, Thapar University
Patiala

Dr. R.K.Sharma
Dean of Academic affairs
Thapar University
Patiala

DEDICATED TO MY PARENTS

ACKNOWLEDGEMENT

Words are often too less to reveals one's deep regards. An understanding of the work like this is never the outcome of the efforts of a single person. I take this opportunity to express my profound sense of gratitude and respect to all those who helped me through the duration of this thesis.

First of all I would like to thank the **Supreme Power,** one who has always guided me to work on the right path of the life. Without His grace this would never come to be today's reality.

This work would not have been possible without the encouragement and able guidance of my supervisor, **Dr. Yaduvir Singh.** Their enthusiasm and optimism made this experience both rewarding and enjoyable. Most of the novel ideas and solutions found in this thesis are the result of our numerous stimulating discussions. Their feedback and editorial comments were also invaluable for the writing of this thesis.

No words of thanks are enough for my **dear parents** whose support and care makes me stay on earth. Thanks to be with me.

At the end, I would like to thank all the faculty members of the department and my friends who directly or indirectly helped me in completion of my thesis.

Swarajya agnihotri

(Regn No. 80651022)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ANNEXURES

NOMENCLATURE

CMOS	Complementary Metal Oxide Semiconductor
DSP	Digital Signal Processing
Etot	Sum of errors throughout discrete frequency domain
HD	Hamming Distance
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
f_x	Sampling frequency
f_c	Cut-off frequency
GA	Genetic Algorithm
$\mathrm{H}\left(e^{jw}\right)$	Magnitude response of filter
LMS	Lest mean square
H_d	Desire magnitude response of filter
MAC	Multiply and accumulate
MOP	Multi objective optimization problem
N	Number of filter coefficients
P_{db}	Pass band ripples
S_{db}	Stop band attenuation
$x*$	Optimal point

a	Switching activity
V_{dd}	Supply Voltage
ω	Angular frequency
w_i	Weighting factor for multi objective optimization

ABSTRACT

With the explosive growth of wireless communication system and portable devices, the power reduction has become a major problem. In applications, such as personal communication systems, and portable storage devices, low power dissipation, hence longer battery life time is a must. With the rapid growth of internet and information on demand, handled wireless terminals are becoming increasingly popular. With limited energy in a reasonable size battery, minimum power dissipation in digital communication devices is necessary. Many of the communication system today utilize digital signal processors (DSP) to resolve the transmitted information. Finite impulse response (FIR) filters have been and continued to be important building blocks in many digital processing systems (DSP).

Signal switching activity is a major component of power dissipation in CMOS circuits. Hamming distance is a measure of switching activity corresponding to the number of energy consuming transition in multiplier and accumulate (MAC) of filter while implementing on digital signal processors (DSP). The transition densities of multiplier input depend on the hamming distance between the successive filter coefficient values. For a multiplier the power is directly dependent on the transition densities and the probabilities of multiplier inputs. The hamming distance between consecutive coefficient values and the number of signal toggling in opposite directions thus forms the measure of bus power dissipation.

Genetic algorithms can implemented as a computer simulation in which a population of abstract representations (called chromosomes or the genotype or the genome) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

It can be seen that evolutionary algorithms differ substantially from more traditional search and optimization methods. The most significant differences are:

- Evolutionary algorithms search a population of points in parallel, not a single point.
- Evolutionary algorithms do not require derivative information or other auxiliary knowledge; only the objective function and corresponding fitness levels influence the directions of search.
- Evolutionary algorithms use probabilistic transition rules, not deterministic ones.
- Evolutionary algorithms are generally more straightforward to apply
- Evolutionary algorithms can provide a number of potential solutions to a given problem. The final choice is left to the user. (Thus, in cases where the particular problem does not have one individual solution, for example a family of pareto-optimal solutions, as in the case of multi objective optimization and scheduling problems, then the evolutionary algorithm is potentially useful for identifying these alternative solutions simultaneously.)

In this thesis the hamming distance of fir filter is minimized by minimizing the switching activity using "Genetic Algorithms" optimization technique to reduce the power dissipation and to increase the battery life of portable multimedia devices.

LITERATURE SURVEY

The advancement in VLSI technology results the systems with highly complex functionalities during the last decade. Major research is carried out in the area of digital signal processing (DSP) where high throughput is most desirable. Now in these days DSP systems are capable of handling and performing complex and computationally intensive signal processing tasks at speeds suitable for real-time operation. Example applications include speech recognition a video signal processing systems.

One of the fastest growing areas in the computing industry is the provision of high throughput DSP systems in a portable form. A major limitation of these systems is the operating time provided by the battery, which is dependent on the characteristics of the battery and the power requirement of the system. Battery technology has peaked in recent years, whereas the number of applications requiring portability is increasing, placing an increasing demand on the power budget. A major concern in both portable and non portable systems is heat dissipation. Overheating can reduce throughput and operation life. Cooling fans and heat sinks significantly add to the overall cost of a system.

Due to the reasons mentioned above, low power design techniques and methodologies are required to be adapted by VLSI system designers, especially those for portable DSP applications.

Digital filters are now widely used throughout the electronics industry, with one of the most common types being the Finite Impulse Response (FIR) filter. FIR filters can have exactly linear phase response and suffer less from the effects of finite word length than IIR filters. However, the drawback is the hardware complexity. A typical FIR filter may requires many stages in order to provide a suitable response and therefore, rather than having a respective multiplier for each filter stage, a more economical means of realizing such a device is to use a CMOS based digital signal processing device wherein all multiplication operations are multiplexed through a single multiplier is of increasing concern to engineers however, is the power consumption of such CMOS based devices, due in particular t their widespread use in a variety of portable, battery powered electronic devices such as mobile phones. It has been shown that the main source of power dissipation in a typical CMOS logic gate is due to switching power, $P_{dynamic}$. Finite

Impulse Response (FIR) digital filters are an essential component of digital signal processors, optimizing the algorithm employed by these filters, has a significant effect on the power consumption of the system.

Anantha P. Chandrakasan, Samuel Sheng and Robvert W. Brodersen have given variety of considerations that must be taken into account in low-power design which include the style of logic, the technology used, and the logic implemented. Factors that were shown to contribute to power dissipation included spurious transitions due to hazards and critical race conditions, leakage and direct path current, pre charge transitions due to hazards and critical race conditions, leakage and direct path current, pre charge transitions, and power consuming transitions in unused circuitry. A pass-gate logic family with modified threshold voltages was found to be the best performer for low power designs, due to the minimal number of transistors required to implement the important should be used if the parasitic capacitances are less than the active gate capacitances in a cascade of logic gates [8].

Henry Samueli has been presented an improved FIR filter coefficient optimization algorithm which allocates additional nonzero digits in the CSD code to compensate for the nonuniform distribution of the CSD coefficient set. The two-stage algorithm consists of search for an optimum scale factor followed by a bivariate local search in the neighborhood of the scaled and rounded CSD coefficients. The increase in filter complexity resulting from the additional CSD digits is minimal; however a significant improvement in the frequency response is contained. This technique can also be used in conjunction with other techniques for reducing FIR filter complexity such as the pre filter structures of the interpolated FIR filters. These techniques essentially decompose the filter into a cascade of lower complexity structures. Thus the sealing and local search approach presented in this paper can be applied to the individual sections of the cascade to make the entire Structure multipliers [6].

Dusan Kodek an Kenneth Steiglitz have presented a comparison between an optimal (branch an bound) algorithm and a suboptimal (local search) algorithm for the design of finite word length finite impulse response (FIR) digital filters and showed experimental results for filters of coefficient length from 15 to 35. Finally conclude that when computer resources are not

available for the optimal method, it is still worth applying the local search method to the filter with rounded coefficients [1].

Farid N. Najam has observe that a common thread that runs through most causes of runtime failure is the extent of circuit activity, the rate at which its nodes are switching and propose a new measure of activity, called the transition density, which may be defined as the "average switching rate" at a circuit node [9].

Based on a stochastic model of logic signals, also presented algorithm to propagate density values from the primary inputs to internal and output nodes. To illustrate the practical significance of this work, demonstrated how the density values at internal nodes can be used to study circuit reliability by estimating

- The average power and ground currents.
- The average power dissipation.
- The susceptibility to electro migration failures.

The density propagation algorithm has been implemented in a prototype density simulator. Suing this, presented experimental results to assess the validity and feasible of the approach. However, the algorithm for computing the density in this paper is very basic and does not take into account the effect of internal delays of logic gates

Darren N. Pearson, Keshab K. Parhi have presented a novel approach for low power implementations of finite impulse response (FIR) filters with less hardware overhead than traditional FIR implementations. Parallel of block processing, with duplication of hardware can reduce power consumption parallel processing by block size L requires the critical path to the charged in L times longer time as compared with the sequential implementation and the identical critical path can be charge in longer time with lower supply voltage which leads to lower power consumption. The hardware cost of this approach increases linearly with the block size L and proposed a general techniques for block implementation of FIR filters which requires fewer multipliers than the straightforward block implementation. The use of this approach can lead to a further reduction in power consumption and hardware cost [12].

Anantha P. Chandrakasan, Miodrag Potkonjak, Renu Mehra, Jan Rabaey, and Robert W. Brodersen have presented an automated high level synthesis system, HYPER-LP, for optimizing power consumption in algorithm specific data path intensive circuits using a variety of architectural and computational transformations. The synthesis approach consisted of applying transformation primitives in a well defined manner in conjunction with efficient high level estimation of power consumption. It was found that the optimal supply voltage for minimizing maintaining the system throughput. It was found that the optimal supply voltage for minimizing power was much lower than existing standards and was around 1.5V for most of the examples investigated. This work has addressed some key problems in the automated design of low-power systems, and provides a good starting point for addressing other research problems like detailed power estimation, module selection, partitioning, and scheduling for power optimization [11].

Chetana Nagendra, Robert Michael, Owens Mary and Jane Irwin have shown that the gate pipelined MAC3 circuits, on account of the simplicity of each pipeline stage, can be clocked at very high speeds. However, this advantage is overshadowed by the high power dissipation in the clock drivers, especially for large circuits such as FIR filters and multipliers. In comparison, the half-bit pipelined MAC2 designs have comparable performance (only slightly slower), but have lesser area, power dissipation and present about half the clock load of the MAC3 design [13].

N. Sankarayya, Kuushik Roy, Purdue U, W Lafayette and Debashis Battacharya have presented a set of new algorithms for low power realization of FIR filters which use various orders of differences between coefficients for computing the convolution with the input data. Also the results of computations are stored and reused, thus requiring more storage and storage accesses. These techniques result in a reduction in the computations necessary per convolution as compared to directly using the coefficients. It is shown analytically that this computational reduction at the price of more storage results in a reduction in net energy dissipated. They also analyzed realization of some FIR filters using these algorithms [15].

A.T. Erdogan and T. Arslan have proposed a new multiplication scheme, for application to single multiplier CMOS based DSP processors, for the implementation of low power digital FIR filter through the reduction of switching activity within the multiplier section of the filter. The scheme operates in conjunction with a transpose direct form FIR filter structure and a modified DSP

processor architecture, though a significant reduction in power can be obtained by using algorithms to order the filter coefficients. This reduction has demonstrated using two basic examples, with different word lengths and filter orders, achieving up to 63% reduction in switching activity [18].

R. Hezar and V. K. Medisetti have proposed in this paper an efficient design procedure for digital FIR filters whose coefficients are restricted to the ternary set {-1, 0, +1}, cascaded by a multiplication-free architecture. A dynamic programming algorithm, minimizing the instantaneous error, also proposed to assist in the search for the optimal ternary filter coefficient set. Good power reductions in VLSI implementation appeared feasible when compared to standard implementations. Compared to the other dynamic programming approaches and obtained around 10 dB more stop-band attenuation in our low pass filter designs. Current efforts seek to implement these designs in VLSI [19].

Mahesh Mehendale, S.D. Sheriekar and G. Venkatesh have presented low power realization of FIR filters using multi rate architectures in this paper. They have analyzed the computational complexity of these architectures and shown it to be lower than the both the direct form and block FIR implementations. The architectures enable reducing the frequency and the supply voltage, thus significantly reducing the power dissipation. The results show that with multi rate architectures, the power can be reduced by as much as 73% without resulting in any data path area overhead. Have also showed how the multi rate architecture can be implemented using the 'C2x/' C5x DSPs and presented results to show up to 43% power reduction. Multi rate architectures are thus the most power efficient architectures for both the dedicated ASIC and the programmable DSP based implementations [17].

D.H. Horrocks, T. Arslan and A.T. Erdogan have provided an overview of the techniques and methodologies that have emerged in the past few years for DSP system design. These include techniques for minimizing power at architectural and algorithm levels including DSP programming issues. In addition, this paper also listed the power estimation techniques; the paper indicates some potential design directions [16].

Mahesh Mehendale, S.D. Sheriekar and G. Venkatesh have presented algorithmic and architectural transforms for low power realization of finite Impulse (FIR) filters which

implemented in both, as software on programmable DSPs and as hardwired macros. For the programmable DSP based implementation, these transform address power reduction in the program memory address and data busses and also the multiplier. Also propose architectural extensions to support some of these transformations. The transforms for hard-wired FIR filters aim at reducing the supply voltage while maintaining the throughput and present transforms that reduce the computation for the FIR filters, thus achieve power reduction [14].

P.K.Merakos, K. Masselos, O. Koufopavlou, S. Nikaolaidis and C.E. Goutis have presented a novel high level transformation for low power implementation of FIR filters in the paper. The new idea is the recording of the filter coefficients aiming at the minimization of the switching activity. As a measure of the switching activity the Hamming Distance (HD) between successive coefficients, stored in a memory, is used. The transformation can be incorporated both in application specific architectures and in general purpose programmable architectures. The recording of the N coefficients, for the HD optimization of their sequence, can be modeled by a Travelling Salesman Problem (TSP), which is a well known NP complete problem. A novel heuristic algorithm for a fast and accurate solution to this problem is proposed. Experimental results show that the proposed technique leads to significant power savings in terms of switching activity reduction [20].

Chris J. Nicol and Patrik Larsson have described Booth encoded multipliers and their used in FIR filters and other DSP applications where one input is random and the other is highly correlated. Selecting the correct multiplier configuration for a given application can reduce power by more than 50% depending on the filter response. It has shown that applying the coefficients of an FIR filter to the Booth encoded input gives less switching activity in the multiplier than when applied to the multiplicand input and also show that power saving are possible when using multiplexed multipliers to compute several filter taps. The techniques are supported with measurements from a full custom adaptive equalizer chip for broadband communications [21].

M.S. Brigth and T. Arslan have showed that the reduction in overall power is due to a decrease in switching activity at coefficient inputs of the multiplier and both data and coefficient memory buses by a factor determined by the data input block size. Results are presented which

demonstrate up to 34% savings in power. The paper presented the scheme, outlines its implementation using an example and demonstrated results with various filter orders and word lengths [24].

William L. Freking and Keshab Parhi have demonstrates that residue arithmetic can result in implementation of low power FIR digital Filters. It has showed that, for word lengths up to 32 bits, the power consumption of residue arithmetic based FIR filters is dramatically less than two's complement based FIR filters. The power reduction is possible since the use of residue arithmetic transforms the filtering problem into multiple smaller word length filters for various module which are operated in parallel. These compact filters can be operated with lower supply voltage for a specified sample speed, thus obtaining decreased power consumption compared to binary. Power reduction factors for residue arithmetic implementation become increasingly favorable as the system word length is increased [27].

S.D. Sherlekar, G.Venkatesh and Mahesh Mehendale have addressed the problem of reducing power dissipation of finite impulse response (FIR) filters implemented on programmable digital signal processors (DSP's). It has described a generic DSP architecture and identifies the main sources of power dissipation during FIR filtering and present seven transformations complement each other and together operate at algorithmic, architectural, logic and layout levels of design abstraction. Each of the transforms is discussed in detail and the results are presented to highlight its effectiveness. They have showed that the power dissipation can be reduced by more than 40% using these transforms. The transformation has been encapsulated in frame work that provides a comprehensive solution to low-power realization of FIR filters on programmable DSP's [23].

A. T. Erdogen and T. Arslan have given a new multiplication algorithm for the low power implementation of digital filters on CMOS based digital signal processing systems. The algorithm decomposes individual coefficients into two less complex. Sub components. The decomposition, performed using heuristic approach, divides a given coefficient such that a part is produced which can be implemented using a single shift operation, leaving another part with a reduced word length to be applied to the coefficient input of the hardware multiplier. This results in a significant reduction in the amount of switched capacitance and consequently power consumption and described the algorithm and present associated results, including the effects of

overheads due to shift operations, showing up to 63% saving in power [25]. They also introduced a framework for the implementation of high throughput FIR filters for low power applications]. The design methodology incorporated in this framework results in highly flexible FIR filter cores, which are open to exploitation by coefficient and data manipulation techniques resulting in significant power savings. Power savings are achieved within the multiplier units and on system buses making the resulting Fir filters ideal for use as IPS on SoC based platforms. The paper described the design methodology, evaluation environment and presented results with a number of FIR filter examples benchmarks under different design constraints [32].

Tian Sheuan Chang, Yuan Hua Chu, and Chein Wei Jen have presented a low power finite impulse response filter design by using differential coefficients and inputs instead of using them directly such that fewer bits are required, thereby reducing the size of arithmetic units and power dissipation. Present an improved algorithm to effectively generate differential coefficients so that the differential coefficients methods can be applied to full bandwidth (BW) of filters instead of only narrow-band filters in previous approaches. Simulations with fixed coefficient filters indicate reduction in transition activity ranging from 1-53% over the full range of filter BW's. The presented results can be applied to adaptive filters to generate suitable differential coefficients [31].

Vijay Sundararajan and Keshub k. Parhi have presented novel low power synthesis technique for the design of folded or time multiplexed programmable coefficient FIR filters where storage areas is trade off for lower power consumption. A systematic technique is developed for low power mapping of FIR filters to architectures with arbitrary number of multipliers and adders. Power consumed in multipliers is reduced by, reducing switching activity as both the data input as well as the coefficient input. Common input operands can be exposed by unfolding, which however leads to a memo increase. Simulation results are obtained for folding 65 and 129 tap band pass FIR filters. The average power consumed in a multiplier for a fixed number of hardware multipliers with varying unfolding factors compared. It has observed that most of the gains due to unfolding are obtained for relatively small unfolding factors and therefore for relatively small memory area overhead. Depending on the unfolding factor employed the average power consumed in a multiplier seen to reduce anywhere from 54.75% to 81.73% when

transpose FIR filters are synthesized as opposed to synthesizing direct-form FIR filters with no unfolding [30].

Zhan Yu, Meng Lin Yu, Kamran Azadet and Alan N. Willson have exploited the sign-extension property of a 2's complement number in this work. A reduced representation for 2's complement numbers has been proposed to avoid sign extension and the switching of the sign extension bits. The maximum magnitude of a 2's complement number is detected and its reduced representation is dynamically generated to represent the signal. A constant error introduced by the reduced representation and this error is also dynamically compensated. The proposed signal representation is particularly useful in digital filters where the coefficients are slowly varying and having small magnitudes. The proposed technique has been implemented in an adaptive filter and shown to reduced power dissipation by over 38% [34].

Chang Young Han, Hyoung Joon Park, and Lee Sup Kim have focused on power reduction during multiplication and exploited the spatial redundancy of images. Multipliers were separated into two parts, the higher and lower parts of multiplication. Also optimized the multiplication and removed unnecessary multiplications by accessing it cache which stores the higher 4-bits multiplication result. a replacement strategy, employing an incrementor, has proposed for adequate cache architecture for image data processing. The total power was reduced by 14% in the multiplier and 10% in the 1-D 7-tap FIR filter. The speed increase was about 17% due to the use of the smaller multipliers. Therefore, dropping the supply voltage with the same clock frequency and constant throughput can save more power [35].

Harry J. M. Veendrick has derived a simple formula for quick calculations of the maximum short-circuit dissipation of static CMOS circuits. A detailed discussion of this short-circuit dissipation is given based upon the behavior of the inverter when loaded with different capacitances. It was found that if each inverter of a string is designed in such a way that the input and output rise and fall times are equal, the short-circuit dissipation will be much less than the dynamic dissipation <20 percent. This result has been applied to a practical design of a CMOS driving circuit (buffer), which is commonly built up of a string of inverters [3].

Keshab K. Parhi has reviewed several approaches for low power implementation of building blocks for digital subscriber line (DSL) systems. Low power implementations of fast Fourier

transforms (FFTs), FIR filters, and equalizers, and reduction of power consumption by use of dual supply voltage have addressed. It has shown that use of separate Galois Field functional units for multiply accumulates and degree reduction can reduce the energy consumption of RS coders dramatically. A hybrid feed forward and feedback commutator scheme based FFT has shown to require less area and full hardware utilization efficiency. Reduction of switching activity at one or both inputs of the multipliers is a key to reduction of power consumption in FIR filters and equalizers. The switching activity reduced by use of transpose structure and by time-multiplexing of an unfolded filter. A well established retiming approach can be generalized to find those non critical gates which can be operated with lower supply voltages to reduce the overall system power consumption [36].

Youngbeom Jang and Sejung Yang have presented a high speed, low power canonic signed digit (CSD) linear phase finite impulse response (FIR) filter structure using vertical common sub expression. In the conventional linear phase CSD filter, the horizontal common sub expression method has been widely utilized due to the inherent symmetrical filter coefficients. In this, use has been made of the Fact that the most significant bits of adjacent filter coefficients in the linear phase filter are also equal since they have mostly similar values. It has shown that the proposed structure is more efficient in the case where bit precision of implementation is lower [40].

Alberto Garcia, Lukusa D. kabulepa and Manfred Galasner have proposed an efficient technique for estimating the node transition activity in MAC architectures implementing FIR filters. The emphasis has been on the estimation of the activity at the output of the MAC multiplier. It is based on divide and conquers approach, an accurate yet efficient method yet efficient procedure is developed. The technique has been evaluated for different synthetic and real data sets. the results depict only very slight discrepancies with respect to precise bit level simulation [37].

Kyung Saeng Kim and Kwyro Lee have designed a 32-tap FIR filter with two 16-tap macros according to the derived condition on the basis of the one chip criterion such as the product of the occupied area and power consumption, which is especially applicable to an FIR filter with multiple taps. The chip is implemented in 0.6μm CMOS technology with three levels of metal and occupies 2.3×2.5 mm^2 of silicon area having operating frequency of 20 MHz and consumes 75mW at a V_{dd} = 3.3v [41].

A.T. Erdogan, M. Hasan and T. Arslan have presented a number of novel architectures for the implementation of low power FIR filtering cores. These architectures have directly from flexible algorithms which exploit data and coefficient correlation in order to minimize that effective switched capacitance on the multiplier, and data/coefficient buses. Another characteristic of these algorithms is that they can be combined to form more power efficient algorithms which in turn could be mapped to more effective architectures. It has described the FIR filtering architectures, the arithmetic processing cores which characterize individual architectures, and provides results which demonstrate up to 39% reduction in power [38].

Jongsun Park, Woopyo Jeong, Hamid Mahmoodi Meimand, Yongtao Wang, Hunsoo Choo and Kaushik Roy have presents programmable digital finite impulse responses (FIR) filter for high performance and low power applications. The architecture is based on a computation sharing multiplier (CSHM) which specifically targets computation reuse in vector scalar products and can be effectively used in the low complexity programmable FIR filter design. Efficient circuit level techniques, namely a new carry select adder and conditional capture flip-flop (CCFF), are also used to further improve power and performance [39].

Alok A. Katkar and James E. Stine have proposed truncated multiplication technique which provides and efficient method for reducing the power dissipation and area of rounded parallel multipliers in digital signal processing systems. This technique provides significant savings in terms of power dissipation for unsigned multiplication. Although previous implementations involved unsigned and signed array and tree multipliers, this technique can be equally applied to multiplication using Booth encoding and compressors for signed multiplication. Initial estimates indicate that truncated parallel multipliers dissipate less power than standard parallel multipliers for operand sizes of 16 bits [44].

D. Suckely has shown that, for finite impulse response (FIR) filter, a genetic algorithm can be used to provide a design method that is automatic, efficient, and produces filter realization that are close to the minimum possible computational complexity. Although it has demonstrated that filter realization of guaranteed minimum computational complexity can be produced automatically, optimization of the method is still needed to produce acceptable run times. The

need for more efficient design procedures has prompted the work on the application of the genetic algorithm [7].

G. Wade, A. Roberts and G. Williams have described a new synthesis technique for multiplier less FIR digital filters consisting of a cascade of primitive linear phase sections. For medium order filters the search space for an optimal cascade is typically of the order of 10^{30} and this can be examined in a computation efficient way using the parallel search capability of a genetic algorithm (GA). A particular form of GA based upon multilevel or structured chromosomes has been developed for the primitive cascade problem. Initial results suggest that, for the cost of increased filter delay, a typical 2:1 advantage can be achieved in both VLSI chip area and clock rate compared to filters designed using the usual equiripple method [10].

Mehmet Oner has used a genetic algorithm to design and optimize digital FIR filter coefficients. Given the desired amplitude response of the filter to be designed, algorithm generates the filter coefficients with the specified number of taps and bits per coefficients. The linearity of the phase response is satisfied by making filter coefficient symmetric. Algorithm generates a population of genomes that represents the filter coefficients and compares the amplitude response of each genome to that of the desired amplitude response. New genomes are generated by crossover, mutation as well as deterministic pruning method. The algorithm directly generates digital coefficients, so while implementing a filter in hard ware there is no need to truncate coefficients [29].

A. Lee, M. Ahmadi, G.A. Jullien, W.C. Millier, and R.S. Lashkari have presented an iterative method to design digital FIR filter using genetic algorithm. Additional processes are used to ensure that the filter coefficients are canonical signed-digits. It has shown that GA is robust in both encoding and parent selection schemes. The combinations of encoding with either modified ranking or roulette wheel method produce similar results. Mini-max strategy generally produces better filters than LMS. The modified GA structure presented in this paper can be extended to solve different kind of constraints problems [28].

Mare S. Bright and Tughrul Arslan have presented a new tool for the synthesis of low power VLSI designs, specifically, those designs targeting digital signal processing applications. The synthesis tool genetic algorithm for low power synthesis (GALOPS) uses a genetic algorithm to

apply power reducing transformations to high level signal processing designs that satisfy power requirements as well as timing and area constraints. GALOPS uses problem specific genetic operators that are specifically tailored to incorporate VLSI based digital signal processing design knowledge. A number of signal processing benchmarks are used to facilitate the analysis of low power design tools, and to aid in the comparison of results. Results demonstrate that GALOPS achieves significant power reductions in the presented benchmark design [33]. They also presented a Genetic Algorithm for the synthesis of Very Large Scale Integration low power Digital Signal Processing systems in this paper. The genetic algorithm operates on a high level signal flow graph of the system, which contains functional blocks such as adders, multipliers, etc. Evaluation of each design involves consideration of issues at different levels throughout the design hierarchy, such as functionally and silicon level implementation. A multi objective genetic algorithm is used to concurrently track aspects of speed, area and power to produce optimum low power designs. A distinct feature of the genetic algorithm is the use of a library of high level transformations which is referenced by the genetic operators. The paper describes the Genetic Algorithm in detail and presents results showing its effectiveness with a number of signal processing systems. Proposed a block processing scheme for low power implementation of FIR filters on single multiplier CMOS based Digital Signal Processors (DSPs) [22].

Yong Lian and Ling Cen have presented a Genetic Algorithm for the design of low power high speed FIR digital filters. The proposed method is based on the factorization of a long FIR filter into a several short filters to increase the speed and lower power dissipation. With the help of GA encoding scheme, the coefficients of all the sub filters are quantized into signed power-of-two values simultaneously. It has also shown by means of example, that the filter optimized by GA yields significant savings in terms of hardware cost [42].

Dragan Cvetkovic and Ian C. Parmee have described a new preference method and its use in multi objective optimization. These preferences are developed with a goal to reduce the cognitive overload associated with the relative importance of a certain criterion within a multi objective design environment involving large numbers of objectives. Their successful integration with several Genetic Algorithm based design search and optimization techniques such as weighted sums, weighted Pareto, weighted co-evolutionary methods, and weighted scenarios are described

and theoretical results relating to complexity and sensitivity of the algorithms are presented and discussed [43].

Christakis Charalambous have discussed the problems of decision making where there are many conflicting objective is known as the multi criterion optimization (MCO) problem, and this forms the basis for most engineering designs. The purpose of this paper is to present a new method for generating efficient points for the MCO problem. A new function, the shifted mini-max function has defined that has the property that any efficient point of the MCO problem can be generated by minimizing the new function with appropriately chosen shifts. Using the new idea for generating efficient points the optimality conditions for the MCO problem can be easily derived. The new method has also applied to the simultaneous amplitude and group delay approximation problem of 1-D digital filters [5].

Michel. R. Lightner and Stephen. W. Director have examined multi criterion optimization (MCO) one aspect of multi criterion decision making (MCDM) problem. The ideas and method of MCO problem are reviewed. Then they developed a weighted ∞-norm method for generating noninferior solutions to the MCO problems. User oriented weight selection heuristics have developed and applied this technique to the MCO design of an MOSFET NAND gate [2].

CHAPTER 1

INTRODUCTION

An important trend that has been a major driver of the electronics industry in recent years is the growing demand for portable computing and communication devices. This demand has been fueled by the quality of life and business productivity improvements that have been provided by these devices. There has been a tremendous interest in laptop computers, personal digital assistants, mobile phones, and pagers. This is only the beginning; however, as there are more sophisticated devices on the horizon that will provide increasingly sophisticated capabilities and features, In future mobile, wireless terminals provide users with ubiquitous and untethered access to multimedia content and computing services available from a high-bandwidth backbone network of computers.

A portable multimedia terminal must provide two fundamental capabilities: the ability to process multimedia information and the ability to communicate that information through various wire and/or wireless communication channels. Speech, audio, video, and graphics are examples of the types of data that are processed by a typical multimedia terminal. The technology that provides the underlying algorithms to process these data types is Digital Signal Processing (DSP). Digital signal processing is also the technology that is applied to process the signals that are used to communicate information over a wired or wireless communication channel. Improvements in computational performance provided by advances in integrated circuit fabrication technology allow the use of more and more sophisticated signal processing techniques that allow greater functionality and performance and richer modes of communication in portable multimedia terminals.

A major problem associated with increases in the processing power and the sophistication of signal processing algorithms is the increasing levels of power dissipation. A mobile terminal is typically powered by batteries, a limited source of energy. For a portable device to be useful it must have a reasonable amount of run time before, the batteries run out and need to be recharged. Another problem with high levels of power dissipation is the cost of packing and cooling. Low-power integrated circuits can be placed in inexpensive and compact packages.

High-power devices, on the other hand, require expensive and bulky packages and cooling mechanisms. High levels of power dissipation also mean high operating temperatures that adversely affect the reliability of an integrated circuit.

Power dissipation is not a new problem. In the middle of 1980s, designers faced the same problem. The solution then was to switch from NMOS technology, which suffered from static power dissipation, to CMOS technology. CMOS was far more energy-efficient than NMOS and was the most effective and economically viable way to build more and more powerful microprocessors. In fact, CMOS was so energy efficient, that power became an afterthought once again. But in recent years, increasing levels of computing performance have made power an important and challenging problem once again, and this time, there is no magic technological solution to make it disappear.

To provide the required computing power and to increase the energy efficiency of signal processing circuits, designers have developed numerous design techniques that can be applied in custom, application-specific integrated circuits. While this approach has been successful in increasing energy efficiency, it suffers from the drawback that the resulting devices can only provide the limited functionality that they were designed to provide, i.e., they are not very flexible. In reality, however, a variety of multimedia data types and services, modes of communication and associated standards are in use, and it is highly desirable to have devices that can deal with this variety. Therefore, it is highly desirable that flexible, programmable components be used to implement the processing functions required in a modern computing/ communication device.

1.1 LOW POWER DESIGN METHODOLOGIES

The design variables that effect the dynamic power consumption of a CMOS circuit have been identified. Now the power minimization problem from various design aspects will be investigated that effect power dissipation: technology, circuit techniques, architectures and algorithms.

1.1.1 TECHNOLOGY

An optimization that could be done at this level is driven by voltage scaling. It is necessary to scale supply voltage for a quadratic improvement in energy per transition. Unfortunately, we pay

a speed penalty for a V_{dd} reduction with delays increasing, as V_{dd} approaches the threshold voltage of the devices. The simple first order relationship between V_{dd} and gate delay, t_d for a CMOS gate is given in (1.1),

$$t_d \alpha \frac{1}{(V_{dd} - V_t)^2} \tag{1.1}$$

The objective is to reduce power consumption while keeping the throughput of the overall system fixed. Therefore compensation for these delays at low voltages is required.

1.1.2 CIRCUIT TECHNIQUES

There are a number of options available in choosing the basic circuit approach and topology for implementing various logic and arithmetic functions. Choices between static vs. dynamic implementations, pass-transistor vs. conventional CMOS logic styles, and synchronous vs. asynchronous timing are just some of the options open to the system designer.

1.1.3 ARCHITECTURE OPTIMIZATION

As seen in equation (1.1), gate delays increase drastically, when supply voltage approaches the threshold voltage of the MOS transistor. There are two architectural techniques that can improve the speed of the circuit under reduced supply voltage.

- **Pipelining:** It is a powerful transformation of the data path to reduce the critical path of the system and improve the speed. It involves the insertion of delay elements, flip flops at specific points of a data flow graph of an algorithm/architecture. The speed gained by this transformation can be traded for low power by voltage scaling.
- **Parallelism:** It is similar to pipelining in that it exploits parallelism in a system, however here this is achieved by duplicating hardware in order to perform a number of similar tasks concurrently.

1.1.4 ALGORITHM

Choosing the algorithm to implement the application at one hand, represents the most important decision in meeting the power constraints. From the section (1.1.3), it can be deduced that in

order to reap the greatest architectural gains, the ability to parallelize an algorithm will be critical, and the basic computation must be optimized, as the basic theme in low power design is voltage reduction.

Therefore, at the algorithm level, transformations that can be used to increase speed and allow lower voltages are useful. Often these approaches translate into larger silicon area hence the approach has been termed trading area for power.

At the algorithm level, the size and complexity of a given algorithm i.e. operation counts, word lengths and so on determine the activity. If there are several algorithms for a given task, the one with the least number of operations (arithmetic operation, memory access etc.) is generally preferable.

1.2 DIGITAL SIGNAL PROCESSING

Digital Signal Processing (DSP) is the processing of signals by digital means. Historically the origins of signal processing are in Electrical Engineering, and a signal means an electrical signal carried by a wire or telephone line, or by a radio wave,. More generally, a signal is a steam of information representing anything from stock price to data from a remote sensing-satellite. In many cases, the signal is initially analog electrical voltage or current produced by a microphone or some other type of transducer. In some situation data is already in digital form, such as output from the readout system of a compact disk. Analog signal must be converted into digital form before DSP techniques are applied. An analog electrical voltage signal, for example, is digitized using an integrated electronic circuit (IC) device called analog-to-digital converter (ADC). This generates digital form in the form of binary numbers whose value represents the electrical voltage input to the device.

Signals commonly need to be processed in a variety of ways. For examples, the output signal from a transducer may well be contaminated with unwanted electrical “noise”. The electrodes attached to a patient’s chest when an ECG is taken measure tiny electrical voltage changes due to the activity of the heart and other muscles. The signal is often strongly affected by “mains pickup” due to electrical interference from the mains supply. Processing the signal using a filter circuit can remove or at least reduce the unwanted part of the signal. Increasingly nowadays the

filtering of signals to improve signal quality or to extract important information is done by DSP techniques rather than by analog electronics are used filtering of signals of improving signal quality or to extract important information.

The development of digital signal processing dates from the 1960s with the use of mainframes digital computers for number-crunching application such as the Fast Fourier Transform (FFT), which allows the frequency spectrum of a signal to be computed rapidly. These techniques were not widely used at that time, because suitable computing equipment was available only in universities and other scientific research institutions. The introduction of the microprocessor in the late 1970s and early 1980s made it possible for DSP techniques to be used in a much wider range of applications. However, general-purpose microprocessors such as the Intel 8086 family are not ideally suited to the numerically-intensive requirements of DSP, and during the 1980s the increasing importance of DSP led several major electronics manufactures to develop Digital Signal Processor chips-specialized microprocessors with architectures designed specifically for the types of operations required in digital signal processing. Like a general-purpose microprocessor, a DSP is a programmable device, with its own negative instruction code. DSP chips are capable of carrying out millions of floating point operation per second, and like their better-known general-purpose cousins, faster and more powerful versions are continually being introduced.

DSP technology is nowadays common place in such devices as mobile phones multimedia computers, video recorders, CD players, hard disc drive controllers and modems, and will soon replace analog circuitry in TV sets and telephones. An important application of DSP is in signal compression and decompression. In CD systems, for example, the music recorded on the CD is in a compressed form (to increase storage capacity) and must be decompressed for the recorded signal to be reproduced. Signal compression is used in digital cellular phones to allow a greater number of calls to be handled simultaneously within each local “cell”. DSP signal compression technology allows people not only to talk to one another by telephone but also to see one another on the screens of their PCs, using small video cameras mounted on the computer monitors, with only a conventional line linking them together.

Although the mathematical theory underlying DSP techniques such as Fast Fourier and Hilbert Transforms, digital filter design and signal compression can be fairly complex, the numerical operations required to implement these techniques are in fact very simple, consisting mainly of operations that could be done on a cheap foru-function calculator. The architecture of a DSP chip is designed to carry out each operation incredibly fast, processing up to tens of millions of samples per second, to provide real-time performance that is, the ability to process a signal "live" as it is sampled and then output the processed signal, for example to a loudspeaker or video display. All of the practical examples of DSP applications mentioned earlier, such as hard disc drives and mobile phones, demand real-time operation. The major electronics manufactures have invested heavily in DSP technology. Because they now find application in mass-market products, DSP chips account for a substantial proportion of the world market for electronic devices. DSP chips account for a substantial proportion of the world market for electronic devices. Sales amount to billions of dollars annually, and seem likely to continue to increase rapidly.

1.3 ANALOG AND DIGITAL FILTERS

Filters are widely employed in signal processing and communication circuits systems in applications such as channel equalization, noise reduction, radar, audio processing, video processing, biomedical signal processing. In a radio receiver, band pass filter or tuner is used to extracts the signals in a radio channel. In an audio graphic equalizer the input signal is filtered into a number of sub-bands signals and the gain for each sub-band can be varied manually with a set of controls to change the perceived audio sensation. The Dolby system is another example of the application of filters. In this case pre-filtering and post-filtering are used to minimize the effect of noise. In hi-fi audio a compensating filter may be included in a preamplifier to compensate for the non ideal frequency response characteristics of the speakers. Filters are also used to create perceptual audio/visual effect for entertainment and in broadcast audios. The primary functions of the filters are one of the following.

- To confine a signal into prescribed frequency band as in low-pass and band-pass filters.
- To decompose a signal into two or more sub-bands as in filter banks, graphic equalizers, sub-band coders, frequency multiplexers.

- To modify the frequency spectrum of a signal as in telephone channel equalization and audio graphic equalizers.
- To model the input –output relation of a system such as telecommunication channels, human vocal tract, and music synthesizers.

Depending on the form of the filter equation and structure of implementation, filters may broadly be classified into the following classes:

- Linear filters versus non linear filters.
- Adaptive time varying filters versus non adaptive time invariant filters
- Recursive versus non recursive filters
- Direct form, cascade form, parallel and lattice structures.

In signal processing, the function of a filter is to remove unwanted parts of the signal, such as random noise, or to extract useful parts of the signal, such as the components lying within a certain frequency range. The block diagram shown in fig. 2.1 illustrates the basic idea.

Figure 1.1: Basic Filtering Operations

There are two main kinds of filter, analog and digital. They are quite different in their physical makeup and in how they work.

An analog filter uses analog electronic circuits made up from components such as resistors, capacitors and op-amps to produce the required filtering effect. Such filter circuits are widely used in such applications as noise reduction, video signal enhancement, graphic equalizers in hi-fi system, and many other areas. There are well established standard techniques for designing an analog filter circuit for a given requirement. At all stages, the signal being filtered is an electrical voltage or current which is the direct analogue of the physical quantity (e.g. a sound or video signal or transducer output) involved. A digital filter uses a digital processor to perform

numerical calculations on sampled values of the signal. The processor may be a general purpose computer such as a PC, or a specialized DSP (Digital Signal Processor) chip.

The analog input signal must first be sampled and digitized using an ADC (analog to digital converter). The resulting binary numbers, representing successive sampled values of the input signal, are transferred to the processor, which carries out numerical calculations on them. These calculations typically involve multiplying the input values by constant and adding the products together. If necessary, the results of these calculations, which now represent sampled values of the filtered signal, are output through a DAC (digital to analog converter) to convert the signal back to analog form.

In a digital filter, the signal is represented by a sequence of numbers, rather than a voltage or current. The main advantages of digital filters over the analog filters are

- A digital filter is programmable, i.e. its operation is determined by a program stored in the processor's memory. This means the digital filter can easily be changed without affecting the circuitry (hardware). An analog filter can only be changed by redesigning the filter circuit.
- Digital filters are easily designed, tested and implemented on a general-purpose computer or workstation.
- Unlike their analog counterparts, digital filters can handle low frequency signals accurately. As the speed of DSP technology continues to increase, digital filters are being applied to high frequency signals in the RF (radio frequency) domain, which in the past was the exclusive preserve of analog technology.
- The characteristics of analog filter circuits (particularly those containing active components) are subject to drift and are dependent on temperature. Digital filters do not suffer from these problems, and so are extremely stable with respect both to time and temperature.
- Digital filters are very much more versatile in their ability to process signals in a variety of ways; this includes the ability of some types of digital filter to adapt to changes in the characteristics of the signal

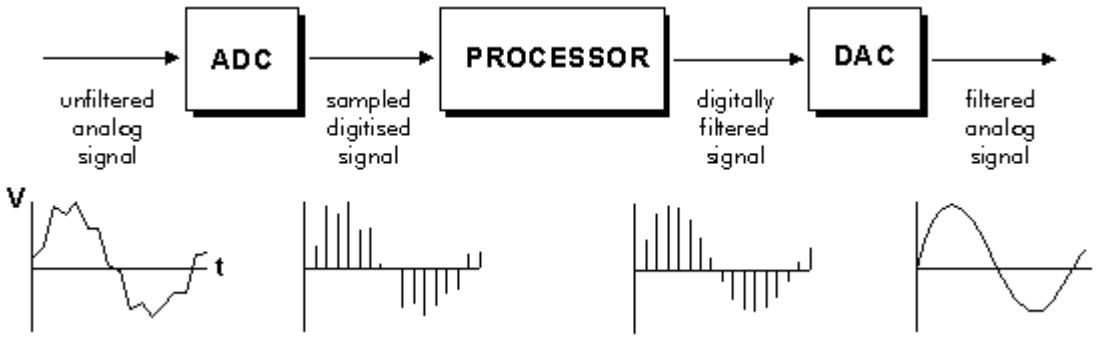


Figure 1.2: Implementation Of Filter On Digital Signal Processor.

Fast DSP processors can be handle complex combinations of filters in parallel or cascade (series), making the hardware requirements relatively simple and compact in comparison with the equivalent analog circuitry. The block diagram of DSP processor is shown in fig 1.2.

The analog input signal must first be sampled and digitized using an ADC (analog to digital converter). The resulting binary numbers, representing successive sampled values of the input signal, are transferred to the processor, which carries out numerical calculations on them. These calculations typically involve multiplying the input values y constants and adding the products together. If necessary, the results of these calculations, which now represent sampled values of the filtered signal, are output through a DAC (digital to analog converter) to convert the signal back to analog form.

1.4 OPTIMIZATION

Optimization is the act of obtaining the best result under given circumstances. Optimization can be defined as the process of finding the condition that gives the maximum or minimum value of function. If $x*$ corresponds to the minimum value of function $f(x)$, the same point also corresponds to the maximum value of the function $-f(x)$. Thus optimization can be taken to mean minimization since the maximum of the function can be found by seeking of the negative of the same number. The optimum seeking methods are also known as mathematical programming techniques. These techniques are useful in finding a minimum of a function of a function of several variables under a prescribed set of constraints. Stochastic process techniques can be used to analyze problems described by a set of random variables having known

probability distributions and the statistical methods enable one to analyze the experimental data and build empirical models to obtain the most accurate representation of the physical situation.

1.4.1 STATEMENT OF AN OPTIMIZATION PROBLEM

An optimization or a mathematical programming problem can be stated as follows

Find $X = [x_1 x_2 x_3 .....x_n]^T$

Subjected to the constraints

$g_j(X) \leq 0. \qquad j = 1, 2.....m$

$h_j(X) \leq 0. \qquad j = 1, 2.....p$

Where X is an n-dimensional design vector, $f(X)$ is termed the objective function and $g_j(X)$ and $h_j(X)$ are known as inequality and equality constraints, respectively. The number of design variables n and number of constraints m and/or p need not be related any way. The stated problem is called a constrained optimization problem. Some problems do not involve any constraints and can be stated as

Find $X = [x_1 x_2 x_3 .....x_n]^T$ which minimizes $f(X)$ are called unconstrained optimization problems.

1.4.2 DESIGN VECTOR

Every engineering system or component is defined by a set of quantities some of which are viewed as variables during the design process. Certain quantities are fixed at the outset and these are called pre assigned parameters. All the other quantities are treated as variables in the design process and are called design or decision various $x_i, i = 1, 2.....n.$ the design variables are collectively represented as a design vector

$X = [x_1 x_2 x_3 .....x_n]^T$.

1.4.3 CONSTRAINTS

In practical design problems, the design variables cannot chosen arbitrary rather, they have to satisfy certain specified functional and other requirements. The restrictions that must be satisfied to produce an acceptable design are collectively called design constraints. Constraints that represent limitations or performance of the system are termed behavior or functional constraints. Constraints that represent physical limitation on design variables are known as geometric constraints or side constraints.

1.4.4 OBJECTIVE FUNCTION

The purpose of the optimization is to choose the best one of many acceptable designs available. Thus a criterion has to be chosen for comparing the different alternative acceptable design and for selecting the one. The criterion, with respect to which the design is optimized, when expressed as a function of the design variables, is known as objective function. While expressing as a function of the design variables if only one criterion is satisfied the problem is called single objective programming problem. In some situations, there may be more than one criterion to be satisfied simultaneously as optimization problem involving multi objective programming problem. With multiple objectives, there arises a possibility of conflict, and it is handled by constructing an overall objective function as a linear combination of the conflicting multiple objectives functions. If $f_1(X)$ and $f_2(X)$ denote two objective functions, a new objective function for optimization is constructed as

$$f(X) = a_1 f_1(X) + a_2 f_2(X)$$

$f(X)$ is a new objective function a_1 and a_2 are constants whose values indicate the relative importance of one objective function relative to the other.

1.5 OPTIMIZATION ALGORITHMS

The optimization problems reveal the fact that the formulation of engineering design problems could differ from problem to problem. Certain problems involve linear terms for constraints and objective functions but certain other involve non linear for them. In some problems, the terms are not explicit functions of the design variables. Unfortunately there does not exist a single

optimization algorithm which will work in all optimization problems equally efficiently. Some algorithms perform better on one problem but may perform poorly on other problems. That is why the optimization literature contains a large number of algorithms, suitable to solve a particular type of problem. The choice of a suitable algorithm for an optimization problem is, to a large extent, dependent on the user's experience in solving similar problems. The optimization algorithms can be classified in to a number of groups, which are briefly discussed as following

1.5.1 SINGLE VARIABLE OPTIMIZATION ALGORITHMS

These algorithms optimize only single objective. These algorithms provide a good understanding of the properties of maximum and minimum points in a function and how optimization algorithms works iteratively to find the optimum point in a problem. These are algorithms are classified into two categories, direct methods and gradient based methods. Direct methods do not use any derivative information of the objective function; only objective function values are used to guide the search process. However gradient based methods use derivative usually contain more than one design variables, single variable algorithms are mainly used as unidirectional search methods in a multivariable optimization algorithms.

1.5.2 MULTI VARIABLE OPTIMIZATION ALGORITHMS

These algorithms optimize more than one objective. These algorithms demonstrate how the search for the optimum point progress in multiple dimensions. Depending on whether the gradient information is used or not used, these algorithms are also classified into direct and gradient based techniques.

1.5.3 CONSTRAINED OPTIMIZATION ALGORITHMS

These algorithms use the single variable and multi variable optimization algorithms repeatedly and simultaneously maintain the search effort inside the feasible search region. These algorithms are mostly used in engineering optimization problems.

1.5.4 SPECIALIZED OPTIMIZATION ALGORITHMS

There exist a number of structured algorithms, which are ideal for only a certain class of optimization problems. Two of these algorithms integer programming and geometric programming are often used in engineering design. Integer programming methods can solve optimization problems with integer design variables. Geometric programming methods solve optimization problems with objective functions and constraints written in a special form.

1.5.5 NON TRADITIONAL OPTIMIZATION ALGORITHMS

There exists a number of other search and optimization algorithms which are comparatively new are becoming popular in engineering design optimization problems in the recent past. Two such algorithms are genetic algorithms and simulated annealing. Genetic Algorithms (Gas) mimic the principle of natural genetics and natural selection to constitute search and optimization procedures. Simulated Annealing mimics the cooling phenomenon of molten metals to constitute a search procedure. Since both these algorithms are abstractions from natural phenomenon so they are very different search methods from the other.

1.6 OPTIMALITY CRITERIA

The purpose of optimization algorithms is find the optimal value of a variable or optimal point in the entire search space, for a point to be optimal there are three different types of criteria.

1.6.1 LOCAL OPTIMAL POINT

A point or solution x^* is said to be a local optimal point if there exist no point in the neighborhood of x^*. In the parlance of minimization problems a point x^* is a locally minimal point if no point in the neighborhood as a functional value smaller than $f(x^*)$.

1.6.2 GLOBAL OPTIMAL POINT

A point or solution x^* is said to be Global Optimal point, if there exists no point in the entire search space which is better than the point x^*, similarly a point x^* is a global minimal point if no point in the entire search space has a functional value smaller than $f(x^*)$.

1.6.3 INFLECTION POINT

A point or solution x^* is said to be inflection point, if the function value increases locally as x^* increase and decrease locally as x^* reduces or if the functional value decrease locally as x^* increase and increase locally x^* decreases.

1.7 OBJECTIVE OF THE THESIS

Finite Impulse Response (FIR) filter is implemented as a series of multiply and accumulate operations on a programmable Digital Signal Processor (DSP). The multiply and accumulate (MAC) unit of a digital signal processor experiences high switching activity due to signal transitions which results in higher power dissipation. Hamming Distance forms a measure of the switching activity during implementation of the filter. The Objective of the thesis is to minimize the Hamming distance and reduce the signal toggle by using optimization technique, Genetic Algorithm (GA), so that its power dissipation is reduced while its implementation on a Digital Signal Processor.

CHAPTER 2

DIGITAL SIGNAL PROCESSING

Digital Signal Processing is one of the most powerful technologies that will shape science and engineering in the twenty-first century. Revolutionary changes have already been made in a broad range of fields: communications, medical imaging, radar & sonar, high fidelity music reproduction, and oil prospecting, to name just a few. Each of these areas has developed a deep DSP technology, with its own algorithms, mathematics, and specialized techniques.

2.1 THE BREADTH AND DEPTH OF DSP

The combination of breath and depth makes it impossible for any one individual to master all of the DSP technology that has been developed. DSP education involves two tasks: learning general concepts that apply to the field as a whole, and learning specialized techniques for your particular area of interest. This chapter starts our journey into the world of Digital Signal Processing by describing the dramatic effect that DSP has made in several diverse fields. The revolution has begun.

2.2 THE ROOTS OF DSP

Digital Signal Processing is distinguished from other areas in computer science by the unique type of data it uses: signals. In most cases, these signals originate as sensory data from the real world: seismic vibrations, visual images, sound waves, etc. DSP is the mathematics, the algorithms, and the techniques used to manipulate these signals after they have been converted into a digital form. This includes a wide variety of goals, such as: enhancement of visual images, recognition and generation of speech, compression of data for storage and transmission, etc. Suppose we attach an analog-to-digital converter to a computer and use it to acquire a chunk of real world data. DSP answers the question: What next? The roots of DSP are in the 1960s and 1970s when digital computers first became available. Computers were expensive during this era, and DSP was limited to only a few critical applications. Pioneering efforts were made in four key areas: radar *&* sonar, where national security was at risk; oil exploration, where large amounts of money could be made; space exploration, where the data are irreplaceable; and medical imaging, where lives could be saved. The personal computer revolution of the 1980s and 1990s caused DSP to explode with new applications. Rather than being motivated by military and government needs, DSP was suddenly driven by the commercial marketplace. Anyone who thought they

could make money in the rapidly expanding field was suddenly a DSP vendor. DSP reached the public in such products as: mobile telephones, compact disc players, and electronic voice mail. Figure 2.1 illustrates a few of these varied applications. This technological revolution occurred from the top-down. In the early 1980s, DSP was taught as a graduate level course in electrical engineering. A decade later, DSP had become a standard part of the undergraduate curriculum. Today, DSP is a basic skill needed by scientists and engineers.

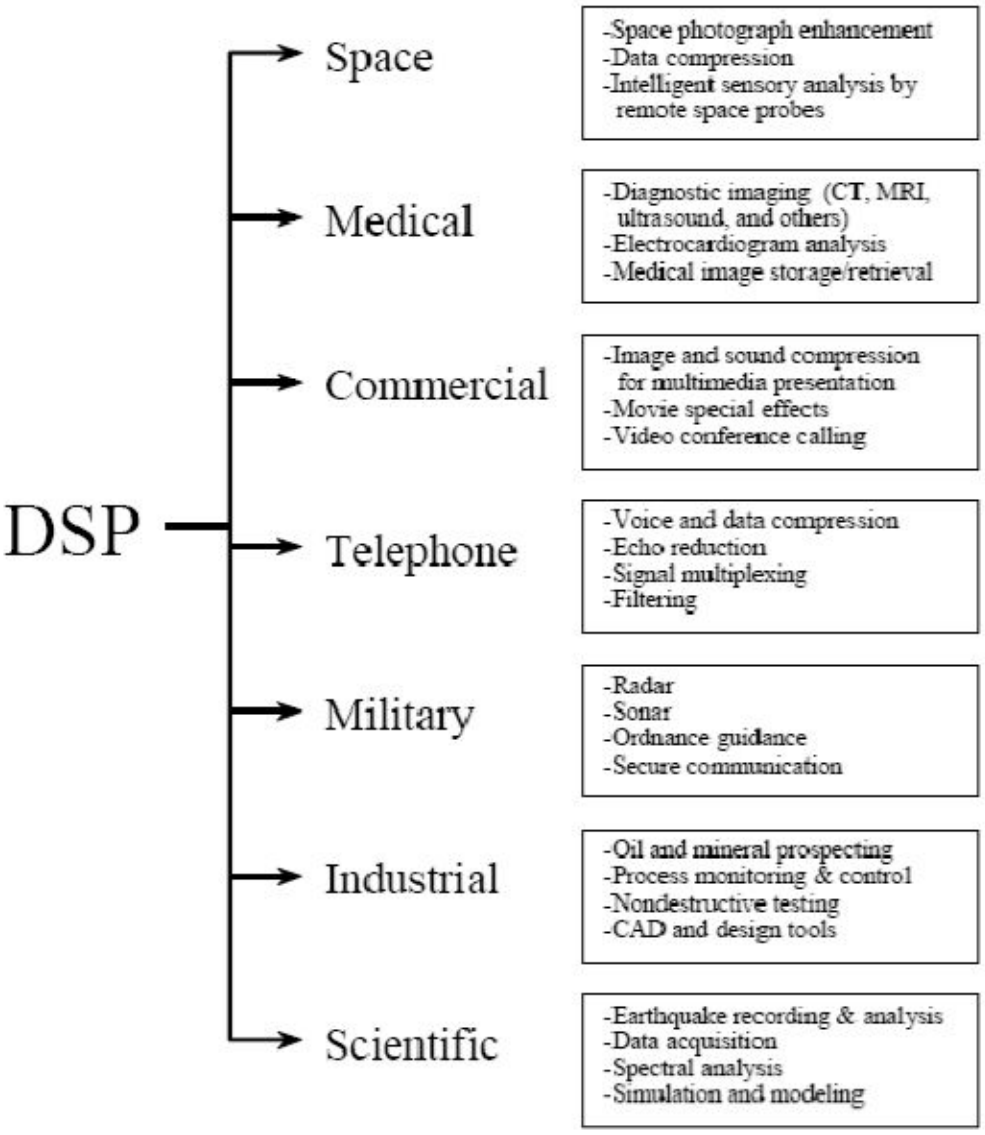


Figure 2.1: Revolutionized Diverse Applications Of DSP.

In many fields. As an analogy, DSP can be compared to a previous technological revolution: electronics. While still the realm of electrical engineering, nearly every scientist and engineer as some background in basic circuit design. Without it, they would be lost in the technological world. DSP has the same future. This recent history is more than a curiosity; it has a tremendous impact on your ability to learn and use DSP. Suppose you encounter a DSP problem, and turn to textbooks or other publications to find a solution. What you will typically find is page after page of equations, obscure mathematical symbols, and unfamiliar terminology. It's a nightmare! Much of the DSP literature is baffling even to those experienced in the field. It's not that there is anything wrong with this material, it is just intended for a very specialized audience. State-of-the-art researchers need this kind of detailed mathematics to understand the theoretical implications of the work. A basic premise of this book is that most practical DSP techniques can be learned and used without the traditional barriers of detailed mathematics and theory. Interdisciplinary, relying on the technical work in many adjacent fields. Fig. 3.2 suggests, the borders between DSP and other technical disciplines are not sharp and well defined, but rather fuzzy and overlapping. If you want to specialize in DSP, these are the allied areas you will also need to study.

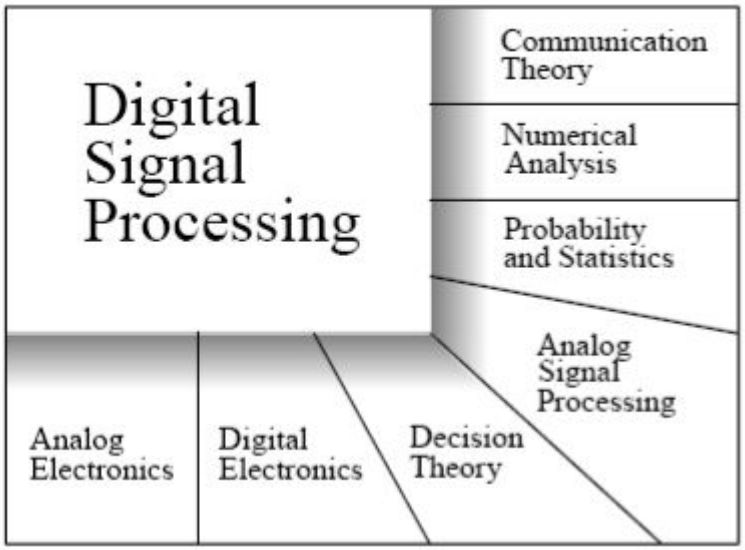


Figure 2.2: Digital Signal Processing Has Fuzzy and Overlapping Borders with Many Other Areas of Science, Engineering and Mathematics.

2.3 PROGRAMMABLE DSPs

Programmable DSPs are specialized microprocessors for real-time number crunching. Because of their specialized applications, programmable DSPs have evolved architectures that are significantly different from conventional microprocessors. With special arithmetic capabilities and data addressing modes, DSPs have consistently outperformed microprocessor in signal processing applications. One could say that a programmable DSP is a domain specific processor that targets signal processing.

Moreover, the current trend in the electronics market indicates that wireless technologies for mobile applications are becoming a reality for the new millennium. The vision of future telecommunication is "information at any time, any place, any in any form". In the core of these sophisticated applications lie intensive signal processing algorithms thus an increasing need for DSP processor in general. Realizing that DSP processors have already become a driving force in both multimedia and communication, conventional microprocessors have added increasingly more DSP extensions to their products over the past three years.

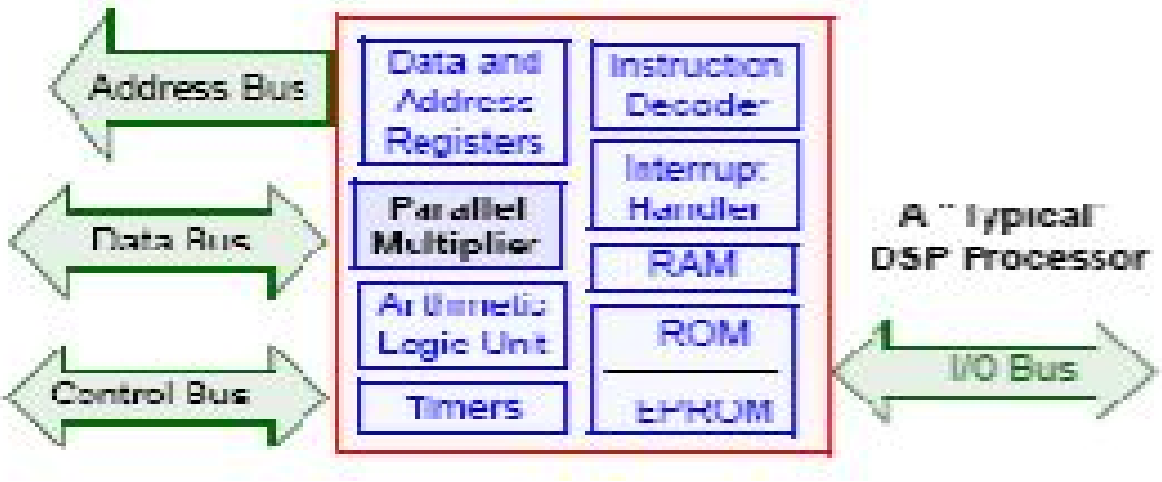


Figure 2.3: DSP Processor Architecture

As these batteries powered constantly evolving / changing mobile applications push for flexible and low power system-on-chip solutions, DSP vendors are putting more effort into architecture and process enhancements in order to obtain energy efficient DSP processors. One such

approach taken by DSP vendors is to optimize DSP architectures with an application domain in mind i.e., to design by DSP vendors is to optimize DSP architectures with an application domain in mind i.e., to design domain specific DSPs. For instance, Texas Instruments' C54x family is optimized for wireless applications .This processor has a domain specific compare, select, and store unit (CSSU) to accelerate the butterfly operations that are part of many communications algorithms. Texas Instruments extended the basic architectures of c54x family further by adding one more MAC unit, thereby increasing instruction level parallelism. The end low power DSP product family is called the c55x family. Other DSPs on the market that target wireless applications are the Lucent 16000 series and the AD121xx series from Analog Devices.

2.4 FILTER ALGORITHMS

The term filter is often used to describe a device in the form of physical hardware or software that is applied to a noisy set of data in order to extract information about a prescribed quantity of interest. In general linear time-invariant discrete-time filters are characterized by the general linear constant coefficient difference equation given in (3.1)

$$y(n) = -\sum_{k=1}^{N} a_k . (n-k) + \sum_{k=0}^{M} b_k . x(n-k) \qquad (2.1)$$
$$x \varepsilon X, d^+, d^-$$

Filters are divided into two categories according to their impulse response, those that have a finite duration impulse response (FIR) and those that have an infinite duration impulse response (IIR).

2.5 FINITE IMPULSE RESPONSE (FIR) FILTERS

In general a FIR filter is described by the difference equation given in (2.2)

$$y(n) = \sum_{k=0}^{M} b_k . x(n-k) \qquad (2.2.)$$

Such a difference equation can be implemented using a transversal filter as shown in figure 2.4. The transversal filter, which is also referred to as a tapped delay line filter, consists of three basic elements:

- Unit-delay element
- Multiplier, and
- Adder

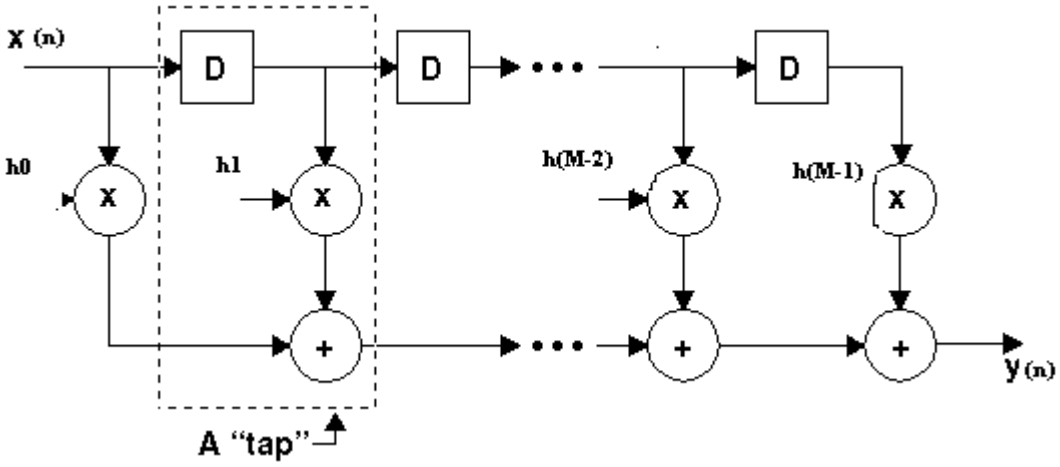


Figure 2.4: Transversal FIR Filter

The number of delay elements used in the filter determines the finite duration of its impulse response. The number of delay elements, shown as M in figure 3.1 is commonly referred as the filter order. The role of each multiplier in the filter is to multiply the tap input by a filter coefficient called tap weight.

Because of its linear phase response, FIR filters are extensively used in existing DSP applications. For an FIR filter to have linear phase, its impulse response should satisfy the following symmetry (+), asymmetry (-), condition.

$$h(n) = \pm\left(M-1-n\right) \quad n \quad 0,1=.....M-1 \tag{2.3}$$

Here M is the order of the FIR filter.

FIR filters are particularly useful for applications where exact linear phase response is required. The FIR filter is generally implemented in a non-recursive way which guarantees a stable filter. FIR filter design essentially consists of two parts

- Approximation problem
- Realization problem

The approximation stage takes the specification and gives a transfer function through four steps. They are follows:

- A desired or ideal response is chosen, usually in the frequency domain
- An allowed class of filters is chosen (e.g. the length N for a FIR filters).
- A measure of the quality of approximation is chosen.
- A method of the quality of approximation is chosen

The realization part deals with choosing the structure to implement the transfer function which may be in the form of circuit diagram or in the form of a program.

2.6 FIR ADVANTAGES

FIR filters have the following advantages

- FIR filters can easily be designed to have constant phase delay and/or constant group delay.
- FIR filters implemented with non recursive techniques will always be stable and free from the limit-cycle oscillations that can plague IIR designs.
- Round-off noise (which is due to finite precision arithmetic performed in the digital processor) can be made relatively small for non recursive implementations.
- FIR filters can also be implemented using recursive techniques if this is desired.

2.7 FIR DISADVANTAGES

Despite their advantages, FIR filters still exhibit some significant disadvantages

- An FIR filter's impulse response duration, although finite, may have to be very long to obtain sharp cutoff characteristics.

The design of FIR filters to meet specific performance objectives is generally more difficult than the design of IIR filters for similar applications

2.8 FIR DESIGNING METHODS

There are essentially three well known methods for FIR filter design namely:

- The window method
- The frequency sampling technique
- Optimal filter design method

2.8.1 THE WINDOW METHOD

In this method, from the desired frequency response specification $H_d(w)$, corresponding unit sample response $h_d(n)$ is determined using the following relation

$$h_d(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(w)^{ejwn}\, dw \tag{2.4}$$

$$H_d(w) = \sum_{n=-\infty}^{\infty} h_d.(n) e^{-jwn} \tag{2.5}$$

In general, unit sample $h_d(n)$ obtained from the above relation is infinite in duration, so it must be truncated at some point say $n = M-1$ to yield an FIR filter of length M(i.e. 0 to M-1). This truncation of $h_d(n)$ to M-1 is same as multiplying $h_d(n)$ by the rectangle window defined as

$$\begin{aligned} &w(n) = 1 && 0 \le n \le M-1 \\ &0 && \text{otherwise} \end{aligned} \tag{2.6}$$

Thus the unit sample response of the FIR filter becomes

$$h(n) = h_d(n)\, w(n)$$

$$\begin{aligned} &= h_d(n) \quad 1 && 0 \le n \le M-1 \\ &= 0 && \text{otherwise} \end{aligned} \tag{2.7}$$

Now, the multiplication of the window function $w(n)$ with $h_d(n)$ is equivalent to convolution of $h_d(w)$ with $W(w)$, Where $W(w)$ is the frequency domain represent of the window function

$$W(w) = \sum_{n=0}^{M-1} w(n) e^{-jwn} \tag{2.8}$$

$$h_d(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(v) W(w-v)\, dw \tag{2.9}$$

The frequency response can also be obtained using the following relation

$$H(w) = \sum_{n=\infty}^{M-1} h(n) e^{-jwn} \tag{2.10}$$

But direct truncation of $h_d(n)$ to M terms to obtain h(n) leads to the Gibbs phenomenon effect which manifests itself as a fixed percentage overshoot and ripple before and after an approximated discontinuity in the frequency response due to the non-uniform convergence of the Fourier series at a discontinuity. Thus the frequency response obtained by using (2.10) contains ripples in the frequency domain. In order to reduce the ripples, instead of multiplying $h_d(n)$ with a rectangular window w(n), h_d9n0 is multiplied with a window function that contains a taper and decays towards zero gradually, instead of abruptly as it occurs in a rectangular window. As multiplication of sequences $h_d(n)$ and w(n) in time domain is equivalent to convolution of $H_d(w)$ and W(w) in the frequency domain, it has the effect of smoothing $H_d(w)$.

The several effects of windowing the Fourier coefficients of the filter on the result of the frequency response of the filter is as follows

- A major effect is that discontinuities in H (w) become transition bands between values on either side of the discontinuity.
- The width of the transition bands depends on the width of the main lobe of the frequency response of the window function, w(n) i.e. W(w).
- Since the filter frequency response is obtained via a convolution relation, it is clear that the resulting filters are never optimal in any sense.
- As M (the length of the window function) increase, the main lobe width of W(w) is reduced which reduces the width of the transition band, but this also introduces more ripple in the frequency response.

- The window function eliminates the ringing effects at the band edges and does result in lower side lobes at the expense of an increase in the width of the transition band of the filter.

Some of the windows commonly used are as follows

2.8.1.1 BARTLETT TRIANGULAR WINDOW

$$W(n) = \frac{2(n+1)}{N+1} \quad = n \quad 0, 1, 2, \ldots\ldots\ldots\ldots, (N-1)/2$$

$$= 2 - \frac{2(n+1)}{N+1} \quad n = (N-1)/2, \ldots\ldots\ldots\ldots, N-1$$

$$= 0 \quad \text{otherwise} \qquad (2.11)$$

2.8.1.2 GENERALIZED COSINE WINDOWS

- Rectangular
- Henning
- Hamming
- Blackman

The general expression for the above four windows may be given as follows

$$W(n) = a - b\cos\left(2p(n+1)/(N+1) + \cos\left(4p(n+1)/(N+1)\right)\right) n \quad 0, 1, \ldots\ldots\ldots\ldots, N-1$$

$$0 \quad \text{otherwise} \qquad (2.12)$$

2.8.1.3 KAISER WINDOW WITH PARAMETER β

$$W(n) = \frac{I_o\left(\beta\sqrt{1-\left(2(n+1)/(N+1)^2\right)}\right)}{I_o(\beta)} \quad n \quad 0, 1, \ldots\ldots\ldots N-1 \qquad (2.13)$$

$$= 0 \quad \text{otherwise}$$

The general cosine window has four special forms that are commonly used. These are determined by the parameters a,b,c

TABLE 2.1 VALUES OF COEFFICIENTS a, b AND c FOR VARIOUS WINDOWS

Window	a	b	c
Rectangular	1	0	0
Henning	0.5	0.5	0
Hamming	0.54	0.46	0
Blackman	0.42	0.5	0.48

The Bartlett window reduces the overshoot in the designed filter but spreads the transition region considerably. The Henning, Hamming and Blackman windows use progressively more complicated cosine functions to provide a smooth truncation of the ideal impulse response and a frequency response that looks better. The best window results probably come from using the Kaiser window, which has a parameter β that allows adjustment of the compromise between the overshoot reduction and transition region width spreading.

The major advantage of using window method is their relative simplicity as compared to other methods and ease of use. The fact that well defined equations are often available for calculating the window coefficients has made this method successful. There are following problems in filter design using window method

- This method is applicable only if $H_d(w)$ is absolutely integrable i.e. only if (2.12) can be evaluated. When $H_d(w)$ is complicated or cannot easily be put into a closed form mathematical expression, evaluation of $h_d(n)$ becomes difficult.

- The use of windows offers very little design flexibility e.g. in low pass filter design, the pass band edge frequency generally cannot be specified exactly since the window smears the discontinuity in frequency. Thus the ideal LPF with cut off frequency f_c, is smeared by the window to given a frequency response with pass band response with pass band cutoff frequency f_1 and stop band cut off frequency f_2.

- Window method is basically useful for design of prototype filters like low pass, high pass, band pass etc. This makes its use in speech and image processing applications very limited.

2.8.2 THE FREQUENCY SAMPLING TECHNIQUE

In this method, the desired frequency response is provided as in the previous method. Now the given frequency response is sampled at a set of equally spaced frequencies to obtain N samples. Thus, sampling the continuous frequency response $H_d(w)$ at N points essentially gives us the N point DFT of $H_d(2pnk/N)$. Thus by using the IDFT formula, the filter coefficients can be calculated using the following formula.

$$H(n) = \frac{1}{N}\sum_{n=0}^{N-1} H(k) e^{j(2\pi n/N)k} \tag{2.14}$$

Now using the above N-point filter response, the continuous frequency response us calculated as an interpolation of the sampled frequency response. The approximation error would then can be exactly zero at the sampling frequencies and would be finite in frequencies between them. The smoother the frequency response being approximated, the smaller will be the error of interpolation between the sample points.

One way to reduce the error is to increase the number of frequency samples. The other way to improve the quality of approximation is to make a number of frequency samples specified as unconstrained variables. The values of these unconstrained variables are generally optimized by computer to minimize some simple function of the approximation error e.g. one might choose as unconstrained variables the frequency samples that lie in a transition band between two frequency bands in which the frequency response is specified e.g. in the band between the pass band and the stop band of a low pass filter.

There are two different set of frequencies that can be used for taking the samples. One set of frequency samples are at $f_k = k/N$ where $k = 0,1,.....N-1$. The other set of uniformly spaced frequency samples can be taken at $f_k = \left(k + 1/2\right)/N \text{ for } k = 0,1,.....N-1$. The steps involved in this method are as follows

- The desired magnitude response is provided along with the number of samples, N. Given N, the designer determines how fine an interpolation will be used. It was found by that for designs they investigated, where N varied from 15 to 256, 16N samples of H(w) lead to reliable computations, so 16 to 1 interpolation was used.

- Given N values of H_k, the unit sample response of filter to be designed, h(n) is calculated the inverse FFT algorithm.

In order to obtain values of the interpolated frequency response two procedures were suggested are

- 1h(n) is rotated by N/2 samples (N even) or (N-1)/2 samples for N odd to remove the sharp edges of impulse response, and then 15N zero valued samples are symmetrically placed around the impulse response.
- h(n) is split around the N/2nd sample, and 15N zero-valued samples are placed between the two pieces of the impulse response.
- The zero augmented sequences are transformed using the FFT algorithm to give the interpolated frequency responses.

2.8.2.1 MERITS AND DEMERITS OF FREQUENCY SAMPLING TECHNIQUE

- Unlike the window method, this technique can be used for any given magnitude response.
- This method is useful for the design of non-prototype filters where the desired magnitude response can take any irregular shape.
- There are some disadvantages with this method i.e. the frequency response obtained by interpolation is equal to the desired frequency response only at the sampled points. At the other points, there will be a finite error present.

2.9 OPTIMAL FILTER DESIGN METHODS

Many methods are present under this category. The basic idea in each method is to design the filter coefficients again and again until a particular error is minimized. The various methods are as follows

- Least squared error frequency domain design.
- Weighted Chebyshev approximation.
- Nonlinear equation solution for maximal ripple FIR filters.
- Polynomial interpolation solution for maximal ripple FIR filters.

The first two methods are discussed in the next section.

2.9.1 LEAST SQUARED ERROR FREQUENCY DOMAIN DESIGN

As seen in the previous method of frequency sampling technique there is no constraint on the response between the sample points, and poor results may be obtained. The frequency sampling technique is more of an interpolation method rather than an approximation method. This method controls the response between the sample points by considering a number of sample points larger than the order of the filter. The purpose of most filters is to separate desired signal from undesired signals or noise. As the energy of the signal is related to the square of the signal, a squared error approximation criterion is appropriate to optimize the design of the FIR filters. The frequency response of the FIR filter is given by (2.8) fir a N-point FIR filter. An error function is defined as follows

$$Error(E) = \sum \left| H\left(w_k\right) - H_d\left(w_k\right) \right|^2 \tag{2.15}$$

Where $w_k = (2\pi k)/L \; and \; H_d(wk) \; are \; L$ samples of the desired response, which is the error measure as a sum of the squared differences between the actual and desired frequency response over a set of L frequency samples. The method consists of the following steps.

- First ‘L’ samples from the continuous frequency response are taken, where L >N (length of the impulse response of filter to be designed)

Then by using the following formula; L point filter impulse response is calculated.

$$h(n) = \frac{1}{N}\sum_{n=0}^{L-1} H(k)\, e^{j(2\pi n/N)k} \tag{2.16}$$

- Then the obtained filter impulse response is symmetrically truncated to desired length N.
- The frequency response is calculated using the following relation.

$$H(w) = \frac{1}{N}\sum_{n=0}^{N-1} h(n)\, e^{-jwn} \tag{2.17}$$

- The magnitude of the frequency response at these frequency point for $w_k = (2\pi k)/L$ will not be equal to the desired ones, but the overall least square error will be reduced effectively this will reduce the ripple in the filter response.

To further reduce the ripple and overshoot near the band edges, a transition region will be defined with a linear transfer function. Then the L frequency samples are taken $w_k = (2\pi k)/L$ using which the first N samples of the filter are calculated using the above method. Using this method reduces the ripple in the interpolated frequency response.

2.9.2 WEIGHTED CHEBYSHEV APPROXIMATION

In this method, following terms are defined

$H_d(w)$ = the desired (real) frequency response of the filter

H(w) = the frequency response of the designed filter

W(w) = the frequency response of the weighting function

The weighting function enables the designer to choose the relative size of the error in different frequency bands. The frequency response of linear phase filters for four different types can be written as follows

$$H(w) = e^{-jw(N-1)/2} e^{j(p/2)L} H*(w)$$

TABLE 2.2 DIFFERENT EXPRESSIONS FOR $H*\left(e^{jw}\right)$ FOR DIFFERENT TYPES OF FILTER

CASES	L	$H*\left(e^{jw}\right)$
Case 1-N odd Symmetrical impulse response	0	$\sum_{n=0}^{(N-1)/2} a(n)\cos(wn)$
Case 1-N odd Symmetrical impulse response	0	$\sum_{n=0}^{N/2} b(n)\cos(w(n-1)/2)$
Case 3-N odd Anti-symmetrical impulse response	1	$\sum_{n=0}^{(N-1)/2} c(n)\sin(wn)$
Case 4-N odd Anti-symmetrical impulse response	1	$\sum_{n=0}^{N/2} d(n)\sin(w(n-1)/2)$

Now each of the expression for $H*\left(e^{jw}\right)$can be written as a product of a fixed function of w, Q(w) and a term that is a sum of cosines, P(w). The expression for P(w) and Q(w) are as follows

TABLE 2.3 EXPRESSION FOR P(w) AND Q(w) FOR DIFFERENT TYPES OF FILTER

CASES	Q(w)	P(w)
Case 1	1	$\sum_{n=0}^{(N-1)/2} \tilde{a}(n)\cos(wn)$
Case 2	Cos(w/2)	$\sum_{n=0}^{(N/2)-1} \tilde{b}(n)\cos(wn)$
Case 3	Sin(w)	$\sum_{n=0}^{(N-1)/2} \tilde{c}(n)\cos(wn)$
Case 4	Sin(w/2)	$\sum_{n=0}^{N/2} \tilde{d}(n)\cos(wn)$

The weighted error of approximation E(w) is by definition

$$H(w) = W(w)\left[H_d(w) - H^*(w)\right] \tag{2.18}$$

$$E(w) = W(w)\left[H_d(w) - P(w)Q(w)\right] \tag{2.19}$$

As Q(w) is a fixed function of frequency, we can factor out Q(w) is a fixed function of frequency, we can factor out. As Q(w) is a fixed function of frequency, we can factor out Q(w)

$$E(w) = W(w)\left[H_d(w)/Q(w) - P(w)\right] \tag{2.20}$$

The two more terms are defined

$$\tilde{W}(w) = W(w)\,Q(w) \tag{2.21}$$

$$H_d^*(w) = H_d(w)/Q(w) \tag{2.22}$$

The error function can now be written as

$$E(w) = \hat{W}(w)\left[H_d^* - P(w)Q(w)\right] \tag{2.23}$$

Thus the Chebyshev approximation consists of finding the set of coefficients $\tilde{a}$ to $\tilde{d}$ so as to minimize the maximum absolute value of E(w) over the frequency bands in which the approximation is being performed. The Chebyshev approximation problem may be stated mathematically as

$$|E(w)| = \min\left[\max|E(w)|\right] \tag{2.24}$$

The solution to this problem is given by Parks and McClellan, who applied a theorem in theory of Chebyshev approximation called the alternation theorem.

2.10 IMPLEMENTATION OF AN FIR FILTER ON DSPS

A low power programmable processor core that will handle the FIR filters described above should have a multiply-accumulate (MAC) unit as one big combinational unit coupled with accumulators register .Having an MAC unit provides several advantages in terms of reducing power consumption. The entire operation sequence $add \rightarrow multiply \rightarrow add$ is done in a single step and controlled by a single instruction. This reduces a significant amount of the effort required in instruction fetching, decoding, and controlling the data path, if the processor core were to implement this operation sequence with separate instructions, such as add, multiply and add. The MAC unit also avoids temporary values to be write/read to/from the register file, thus preventing excessive data movement. These temporary variables are mapped to the wires connecting the adder, multiplier and the accumulator of the MAC unit.

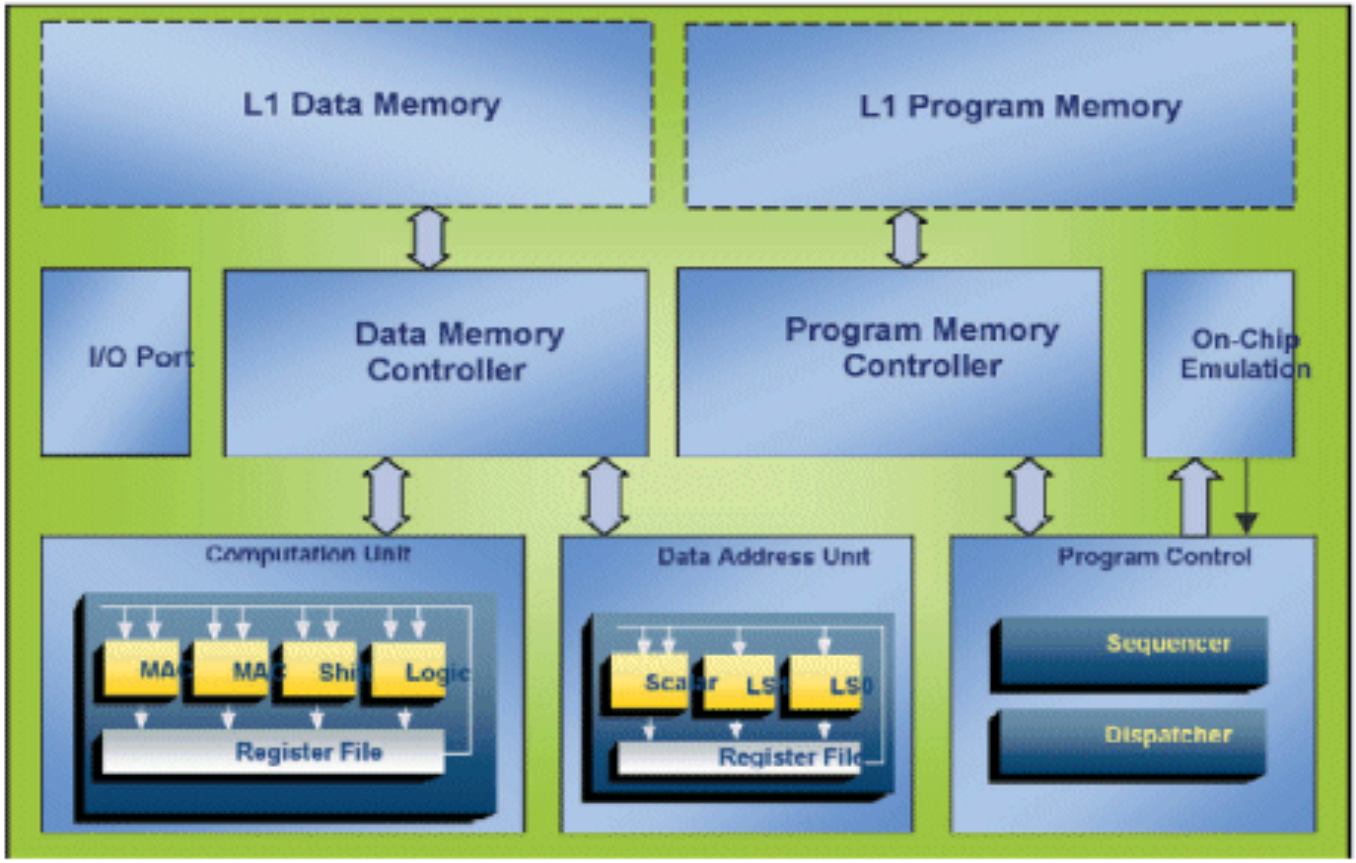


Figure 2.5: DSPs Architecture

In order to perform a multiply and accumulate operation in MAC unit, we need to feed this unit with data values and a coefficient simultaneously. This implies a dual port data memory, which holds delay line elements of the FIR filter, and a coefficient memory.

It should have a data address generation unit that should support circular buffering technique. This technique is used to implement address pointer wraparound and therefore allows shifting the delay line of an FIR filter in a power efficient way. For instance when a new input is shifted into the delay line, it is replace with the "oldest" delay element and the address pointer is incremented by one, now pointing to the new "oldest" delay element. So instead of shifting all the delay elements, it modifies the pointer to get the same effect. When the address pointer reaches the end of the delay line buffer, it is automatically wrapped around to the beginning of the same buffer. However, each modification to the address pointer should be checked if the address pointer is still within the bounds that specify the start and end of the delay line buffer.

An alternative way to implement FIR delay line would be to cascade registers and form a shift register block with M registers, where M is the filter order. But this implementation will suffer from excessive switching activity while shifting all the delay elements. This method may only make sense if the order of the filter or equivalently the number of registers to be cascade is relatively small.

The FIR algorithm can then be mapped into this architecture as a series of MAC instructions. It could be noted that during the execution of the FIR algorithm, the coefficient values directly impact the signal switching activity especially in the coefficient memory data bus and the multiplier. The coefficients can be optimized so as to reduce this signal switching activity and thus reduce power dissipation.

2.11 POWER DISSIPATION-FACTORS AND MEASURES

The motivation for low power electronics has stemmed from three reasonably distinct classes of requirement

- The earliest and most demanding of these is for portable battery operated equipment that is sufficiently small in size and weight and long in operating life. The goal is to satisfy the user of communication and portable multimedia devices.
- The most recent need is for ever increasing packing density in order to further enhance the speed of high performance systems, which imposes severe restrictions on power dissipation density.

Viewed together, these three classes of need appear to encompass a substantial majority of current applications of electronic equipment. Low power electronics has become the mainstream of the effort to achieve gigascale integration (GSI).

2.11.1 SOURCES OF POWER CONSUMPTION

In CMOS circuits the total power dissipation can be obtained from the sum of the three components as summarizes in (2.25)

$$P_{avg} = P_{dynamic} + P_{static} + P_{Leakage} \tag{2.25}$$

There are two major sources of power dissipation

2.11.2 DYNAMIC DISSIPATION

Dynamic dissipation is due to switching transient (short circuit current) and charging and discharging of load capacitance. The dynamic power may further divided into two components

$$P_{dynamic} = P_{switching} + P_{short-circuit} \tag{2.26}$$

The switching component, $P_{switching}$, arises when the capacitive load, C_L, of a CMOS circuit is charged through PMOS transistors to make a voltage transition from 0 to the high voltage level, which is usually the supply, V_{dd}.

For an inverter circuit as shown in figure 2.6, the power dissipated because of a 0 to 1 transition can be determined from the product V_{dd}*I_C where I_C is the transient current drawn from the supply. The time duration for this current flow is t. It can be written in (2.27)

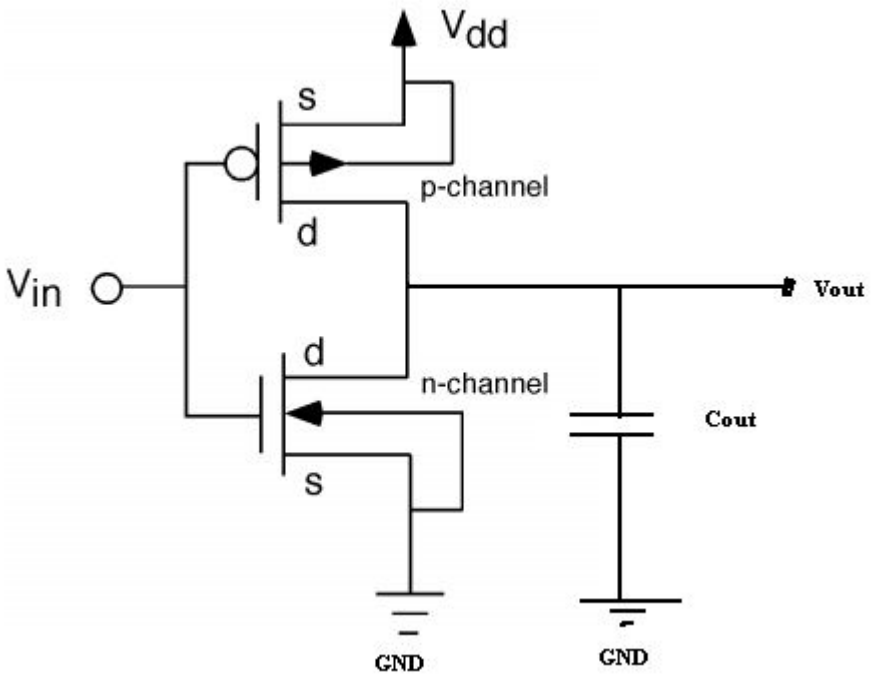


Figure 2.6: CMOS Inverter

$$I_C = C_L \frac{dV_{out}}{dt} \tag{2.27}$$

The energy from power supply is given as

$$E_{0\to1} = \int_0^T V_{dd}.I_C(t)dt = V_{dd}\int_0^T C_L.dV_{out} = C_L.dV_{dd} \tag{2.28}$$

Half of the energy given in (2.28) is stored in the output capacitor and half of it is dissipated in the PMOS transistor .On the 1 to 0 transition at the output, no charge is drawn from the supply, however the energy stored in the output capacitor is consumed. If these transitions occur at a clock rate, $f_{clk,}$ the power drawn from the supply is $C_L * dV_{dd} * f_{clk}$. However, in general the switching will not occur at the clock rate but rather at some reduced rate, which is best described probabilistically. $\alpha_{0\to1}$ is defined as the average number of times in each clock cycle that a node with a capacitance C_L will make a power consuming transition (0 to 1), resulting in an average switching component of power for a CMOS gate to be

$$P_{dynamic} = \alpha_{0\to1} \cdot C_L \cdot V^2{}_{dd} \cdot f_{clk} \tag{2.29}$$

Another dynamic component of power dissipation is $P_{short-circuit}$ .At some point during the switching transient, both the NMOS and PMOS devices in figure3.3 will be turned on. This occurs for gate voltage between V_{in} and V_{dd-1} $V_{tp}1$ where V_{th} and $1V_{tp}1$ are threshold voltages of the NMOS and PMOS transistors, respectively. During this time, a short-circuit exists between V_{dd} and ground. Therefore currents are allowed to flow. If $V_{dd} - G_{nd} < V_{in} + 1V_{tp}1$ is satisfied then a short circuit path between the power supply and ground will never exist, meaning that this component of (2.25) can be eliminated. But even through $P_{short-circuit}$ cannot always be ignored, it certainly is not the dominant component of power consumption.

2.11.3 STATIC DISSIPATION

Static dissipation is due to the leakage current or other current drawn continuously from the power supply. Ideally, CMOS circuits dissipate no static (DC) power since in the steady state there is not direct path from V_{dd} to ground. Of course, this scenario can never be realized in

practice since in reality the MOS transistor is not a perfect sw3itch. Static power dissipation, $P_{leakage}$, stems from the leakage current, $I_{leakage}$, which can arise from substrate injection and sub threshold effects and primarily determined by fabrication technology considerations. This current is typically in the nA region and contributes little to the overall power consumption. However, in future deep submicron technologies, leakage power will become a problem.

2.11.4 PHYSICAL CAPACITANCE

Dynamic power consumption depends linearly on the physical capacitance being switched. The physical capacitance in CMOS circuits stems from two primary sources: devices and interconnect. Capacitances can be kept at a minimum by using less logic, smaller devices, and fewer and shorter wires. Some techniques reducing the active area include resource sharing, logic minimization and gate sizing. Techniques for reducing the interconnect includes, register sharing, common sub-function extraction, placement and routing. However we are not free to optimize capacitance independently. For example reducing device sizes reduces physical capacitance, but it also reduces the current drive ability of the transistors making the circuit operate more slowly. This loss in performance might prevent us from lowering V_{dd} as much as it might otherwise be able to do. If the designer is free to scale voltage it does not make sense to minimize physical capacitance without considering the side effects. Likewise, if voltage and/or activity can be significantly reduced by allowing some increase in interconnect capacitance, this may result in a net decrease in power.

2.11.5 SWITCHING ACTIVITY

A chip can contain a huge amount of physical capacitance, but if it does not switch then no dynamic power will be consumed. The switching activity determines how often this switching occurs. As given in (2.29) there are two components to switching activity. The first is the data rate, f_{clk} which reflects how often on average; new data arrives at each node. This data might or might not be different from the previous data value. In this sense, the data rate f_{clk} describes how often on average, switching could occur. For example, in synchronous system f_{clk} might correspond to the clock frequency.

The second component of activity is the data activity, $\alpha_{0\to1}$ corresponding to the expected number of energy consuming transitions that will be triggered by the arrival of each new piece data. So while f_{clk} determines the average periodicity of data arrivals, $\alpha_{0\to1}$ determines how many transitions each arrival will spark. For circuits that do not experience glitching $\alpha_{0\to1}$ can be interpreted as the probability that an energy consuming (zero to one) transition will occur during a single clock period.

The data activity $\alpha_{0\to1}$ can be combined with the physical capacitance C_L to obtain an objective capacitance, $C_{eff} = \alpha_{0\to1} * C_L$ which describes the average capacitance charged during each $1/f_{elk}$ period. This reflects the fact that neither the physical capacitance nor the activity alone determines dynamic power consumption. Evaluating the effective capacitance of a design is nontrivial, as it requires knowledge of both the physical aspects of the design (such as technology parameters, circuit structure, and delay model) as well as the signal statistics (data activity and correlations).

CHAPTER 3

GENETIC ALGORITHM

Genetic Algorithm is an emerging optimization algorithm for signal processing and considered a powerful optimizer in away areas. Gas has extensively used in signals processing application due to its some distinguishable features like parallelism, robustness, multi objectives, multimodality, number representation and constraints. Parallelism increases the computational speed of the GA based signal processing system. The parallelism may be global migration or diffusion. These categories reflect different ways in which parallelism is exploited in GA and the nature of the population structure and recombination mechanism is used. Robustness made the characteristics of a system variable and adaptive to dynamic signal behavior and able to sustain the environmental disturbances in signal processing. It is common to have more than one objective in signal processing applications. The design of IIR filters is a typical example. The GA has been demonstrated a powerful method for these multi objective problems, enabling to obtain the pareto optimal set instead of single solution. It is common to have a real value searching space in signal processing problems. Genes used for the GA optimization process can be handled directly through binary or even array encodings.

Being a powerful optimization tool, the GA has explored a large number of applications in signal processing. There are large classes of problems that appear to be more amenable to solution by Gas than by any other available optimization techniques. These tasks often involve multiple objective problems or optimization with constraints like the stability assurance of IIR and FIR design. GA can jump out of local optima, it is more desirable for multimodal problems like direct time delay estimation. The most encouraging areas of application are the implementing Artificial Intelligence (AI) system. The use of GAs with artificial neural networks (ANN) and fuzzy logic is expected to receive more intention in the future. For example GAs may be used optimize the membership functions of the fuzzy system or to assist ANN operation through the determination of suitable ANN structures. It is foreseen that more hybrid systems will be launched for signal processing applications and GAs are an important component of these developments.

3.1 GENETIC ALGORITHMS

Genetic algorithms are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. As you can guess, genetic algorithms are inspired by Darwin's theory of evolution. Simply said, problems are solved by an evolutionary process resulting in a best (fittest) solution (survivor) - in other words, the solution is evolved.

Rosenberg introduced evolutionary computing in the 1960s in his work "Evolution strategies" (Evolutions strategies in original). Other researchers then developed his idea. Genetic Algorithms (GAs) were invented by John Holland and developed by him and his students and colleagues. This lead to Holland's book" Adaption in Natural and Artificial Systems" published in 1975. In 1992 John Koza has used genetic algorithms to evolve programs to perform certain tasks. He called his method "genetic programming" (GP). LISP programs were used, because programs in this language can be expressed in the form of a "parse tree", which is the object the GA works on.

3.2 BIOLOGICAL BACKGROUND:

Key features that distinguish a GA from other search methods include a population of individuals where each individual represents a potential solution to the problem to be solved. Individuals are typically binary strings, but in the context of routing we will take them to be sequences of nodes. The terms individual, solution, and genome will be used interchangeably to refer to one potential solution in a GA population. A fitness function which evaluates the utility of each individual as a Solution, a selection function which selects individuals for reproduction based on their fitness, a selection function which exploits useful information currently existing in a population. The GA population will consist of individuals who represent paths between the source node and potential destination nodes. We have use a variable length representation, which is expected to allow the GA more flexibility to evolve in response to, changes in the network. The fitness of a path is determined from the measurement data returned by an ACK that is received in response for sending a dumb packet along the path. GA’s were introduced as a computational analogy of adaptive systems. They are modeled loosely on the principles of the evolution via natural selection, employing a population of individuals that undergo selection in the presence of

variation-inducing operators such as mutation and recombination (crossover). The Algorithms include randomly generating an initial population M (0), computing and saving the fitness u (m) for each individual m in the current population M (t), defining selection probabilities p (m) for each individual m in M (t) so that p (m) is proportional to u (m), generating M (t< 1) by probabilistically selecting individuals from M (t) to produce offspring via genetic operators, repeating the computation for of fitness function for each individual until satisfying solution is obtained. The appeal of GAs comes from their simplicity and elegance as robust search algorithms as well as from their power to discover good solutions rapidly for difficult high-dimensional problems. GAs are useful arid efficient when - the search space is large, complex or poorly understood, domain knowledge is scarce or expert knowledge is difficult to encode to narrow the search space, no mathematical analysis is available, traditional search methods fail.

3.2.1 CHROMOSOME

All living organisms consist of cells. In each cell there is the same set of chromosomes. Chromosomes are strings of DNA and serve as a model for the whole organism. A chromosome consists of genes, blocks of DNA. Each gene encodes a particular protein. Basically, it can be said that each gene encodes a trait, for example color of eyes. Possible settings for a trait (e.g. blue, brown) are called alleles. Each gene has its own position in the chromosome. This position is called locus. Complete set of genetic material (all chromosomes) is called genome. Particular set of genes in genome is called genotype. The genotype is with later development after birth base for the organism's phenotype, its physical and mental characteristics, such as eye color, intelligence etc.

3.2.2 REPRODUCTION

During reproduction, recombination (or crossover) first occurs. Genes from parents combine to form a whole new chromosome. The newly created offspring can then be mutated. Mutation means that the elements of DNA are a bit changed. This changes mainly caused by errors in copying genes from parents. The fitness of an organism is measured by success of the organism in its life (survival).

3.2.3 SEARCH SPACE

If we are solving a problem, we are usually looking for some solution which will be the best among others. The space of all feasible solutions (the set of solutions among which the desired solution resides) is called search space (also state space). Each point in the search space represents one possible solution. Each possible solution can be "marked" by its value (or fitness) for the problem. With GA we look for the best solution among a number of possible solutions - represented by one point in the search space.

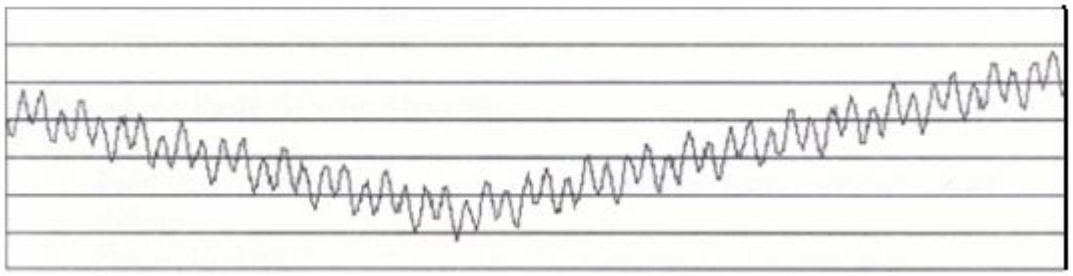


Figure 3.1: G.A. Search Space

Looking for a solution is then equal to looking for some extreme value (minimum or maximum) in the search space. At times the search space may be well defined, but usually we know only a

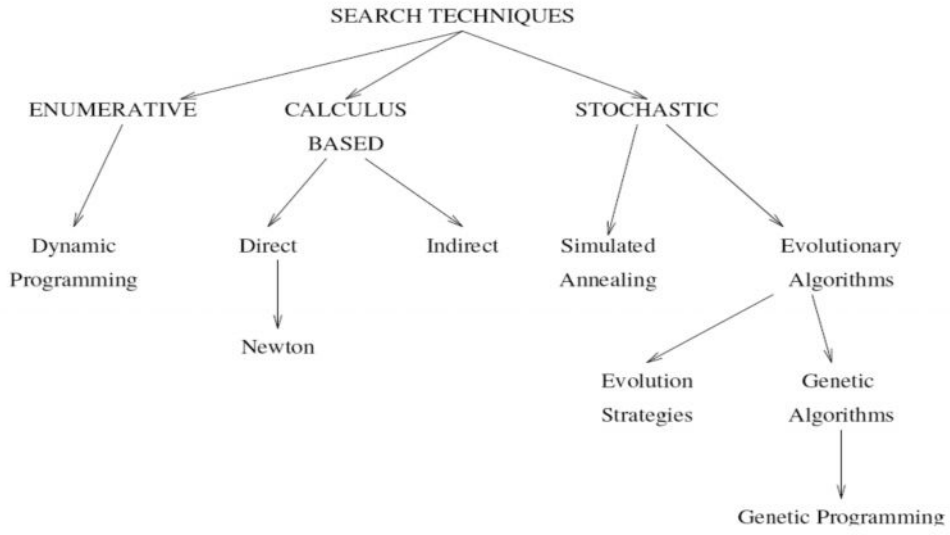


Figure 3.2: Different Search Techniques

few points in the search space. In the process of using GA, the process of finding solutions generates other points (possible solutions) as evolution proceeds. The problem is that the search can be very complicated. One may not know where to look for a solution or where to start. There are many methods one can use for finding a suitable solution, but these methods do not necessarily provide the best solution. Some of these methods are hill climbing, tabu search, simulated annealing and the genetic algorithm. The solutions found by these methods are often considered as good solutions, because it is not often possible to prove what the optimum is. These algorithms are inspired by Darwin's theory of evolution. Solution to a problem solved by genetic algorithms uses an evolutionary process (it is evolved). Algorithms begin with a set of solutions (represented by chromosomes) called population. Solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. Solutions which are then selected to form new solutions (offspring) are selected according to their fitness - the more suitable they are the more chances they have to reproduce. This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied. The search is conducted directly in the solution space and each solution is encoded in a certain way and is called an individual. The search is parallel in the sense that a population of individuals is maintained and the quality of the individuals is calculated by a fitness function. The population is improved by crossover, recombination of genetic material from different individuals. This is based on a hypothesis that a good solution can be built up from shorter partial solutions. Genetic diversity is maintained by a mutation operation, making random changes in the individuals. To summarize, genetic algorithm consists of five components:

- A chromosomal representation of solutions.
- A way to create an initial population of solutions.
- A fitness function
- Genetic operators (selection, crossover, mutation).
- Parameter values for the genetic algorithm (population size, probabilities for applying genetic operators etc).

3.3 WORKING PRINCIPLE

To illustrate the working principle of GA consider a unconstrained optimization problem Maximize $f(X)$

$$X_i^L \le X_i \le X_i^U \qquad \text{for } i = 1, 2, ...N \tag{3.1}$$

If $f(X)$, for $f(X) > 0$ is to be minimized, then the objective function is written as maximize

$$\frac{1}{1 + f(X)} \tag{3.2}$$

If $f(X) < 0$ instead of minimizing $f(X)$, maximize $[-f(X)]$. Hence both maximization and minimization problems can be handled by GA.

3.3.1 ENCODING

Since genetic algorithms search directly in the solution space, it needs a way to encode solutions in a way that can be manipulated by the genetic algorithm. This representation of a solution is called a genetic or chromosomal representation of the solution. To be able to apply the fitness function, the genes have to be transformed to the problem domain. This can be compared to animals in our real world; the genes themselves are not evaluated for fitness. Instead, they carry the blueprints for building an individual whose fitness is consequently evaluated by the surrounding environment. To separate these two aspects of an individual, the genetic representation of the individual is called the genotype and the physical appearance that is caused by the genes is termed phenotype.

Exactly what a gene is has been the item of some discussion. In genetics and evolutionary biology a gene can be defined as a functional block of DNA that encodes some functionality of the individual and lasts for enough generations to serve as a unit of natural selection. In genetic algorithms, each singular locus in the genotype is often referred to as a gene. Genes that code for conflicting functionalities are called alleles in genetics. In the genetic algorithm literature the possible values at each locus are called alleles. In the binary bit string example, the alleles are

simply the values [0; 1] at each locus. In Holland's original genetic algorithm, solutions were always represented by bit strings. Other encoding techniques have been proposed as well, e.g. real valued encodings and sequential coding for ordering problems such as the traveling salesman problem. These kinds of encodings have not been analyzed as much as the binary encoding of the simple genetic algorithm. The encoding has profound implications operators. Several strategies has been suggested for selecting an encoding but until there is more rigorous theory about genetic algorithms and different encodings, the best philosophy seems to be to choose an encoding that is natural for the problem and design a genetic algorithm that can handle this encoding. For most problems there is more than one natural encoding however, so the choice must often be based on trial and error. As Encoding of chromosomes is the first question to ask when starting to solve a problem with GA. Encoding depends on the problem heavily. Common classes under this category are discussed below.

3.3.1.1 BINARY ENCODING

Binary encoding is the most common one, mainly because the first research of GA used this type of encoding and because of its relative simplicity. In binary encoding, every chromosome is a string of bits - 0 or 1.

Chromosome A	10110010110010101011100101
Chromosome B	11111110000011000001111 1

Figure 3.3: Examples of Chromosomes with Binary Encoding

Binary encoding gives many possible chromosomes even with a small number of alleles. On the other hand, this encoding is often not natural for many problems and sometimes corrections must be made after crossover and/or mutation

3.3.1.2 PERMUTATION ENCODING

Permutation encoding can be used in ordering problems, such as traveling salesman problem or task ordering problem.

Chromosomes A	1 5 3 2 6 4 7 9 8
Chromosomes B	8 5 6 7 2 3 1 4 9

Figure 3.4: Chromosomes with Permutation Encoding

In permutation encoding, every chromosome is a string of numbers that represent a position in a sequence. Permutation encoding is useful for ordering problems. For some types of crossover and mutation corrections must be made to leave the chromosome consistent (i.e. have real sequence in it) for some problems.

3.3.1.3 VALUE ENCODING

Direct value encoding can be used in problems where some more complicated values such as real numbers are used. Use of binary encoding for this type of problems would be difficult. In the value encoding, every chromosome is a sequence of some values. Values can be anything connected to the problem, such as (real) numbers, chars or any objects.

Chromosome A	1.2324 5.3243 0.4556 2.3293 2.4545
Chromosome B	ABDJEIFJDHDIERJFDLDFLFFEGT
Chromosome C	(back), (back), (right), (forward), (left)

Figure 3.5: Examples of Chromosomes with Value Encoding

Value encoding is a good choice for some special problems. However, for this encoding it is often necessary to develop some new crossover and mutation specific for the problem.

3.3.1.4 TREE ENCODING

Tree encoding is used mainly for evolving programs or expressions, i.e. for genetic programming. In the tree encoding every chromosome is a tree of some objects, such as functions or commands in programming language.

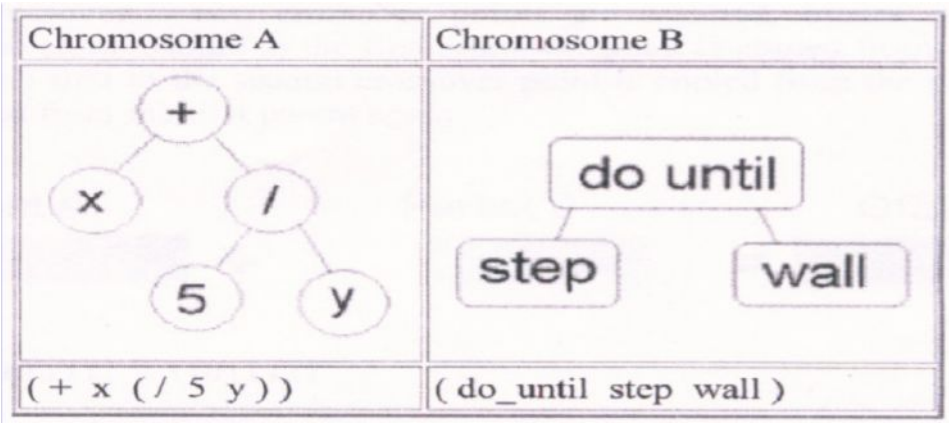


Figure 3.6: Chromosomes with Tree Encoding

3.3.2 POPULATION

The Population is usually set up by randomizing an initial set of solutions. The population size can be variable but is usually fixed to a certain size. It is by far most common that the population is purely generational. This means that the entire population is super ceded by their offspring which makes up the next generation, except individuals preserved if an elitism operator is used.

Some other initialization schemes are also possible. The initialization does not need to be purely random; distribution of initial solutions can for example be weighted with known prior knowledge of the problem domain. Previously known good solutions, e.g. human solutions or solutions generated by some heuristic and perturbations of such solution can be included in the initial population. In simple genetic algorithm the population is randomized.

3.4 THE OBJECTIVE AND FITNESS FUNCTION

The objective function is used to provide a measure of how individuals have performed in the problem domain. In the case of a minimization problem, the most fit individuals will have the

lowest numerical value of the associated objective function. This raw measure of fitness is usually only used as an intermediate stage in determining the relative performance of individuals in a GA. Another function, the fitness function, is normally used to transition the objective function value into a measure of relative fitness, thus:

$$F(x) = g(f(x)) \tag{3.3}$$

Where f is the objective function, g transforms the value of the objective function to a non negative number and f is the resulting relative fitness. This mapping is always necessary when the objective function is to be minimized as the lower objective function values correspond to fitter individuals. In many cases, the fitness function value corresponds to the number of offspring that an individual can expect to produce in the next generation. A commonly used transformation is that of proportional fitness assignment. The individual fitness $f(x_i)$, of each individual is computed as the individual's raw performance, $f(x_i)$, relative to the whole population, i.e.

$$F(x_i) = \frac{f(x_i)}{\sum_{i=1}^{N_{ind}} f(x_i)} \tag{3.4}$$

Where N_{ind} is the population size and x_i is the phenotypic value of individual i. whilst this fitness assignment ensures that each individual has a probability of reproducing according to its relative fitness. Certain Genetic operators require that fitness function be non negative, although certain operators do not have this requirement. Consider the following transformations

$F(X) = f(X)$ For maximization problem

$F(X) = 1/f(X)$ For minimization problem, if $1/f(X) \neq 0$

$F(X) = 1/(1+f(X))$, If $f(X) = 0$ (3.5)

3.5 GENETIC OPERATORS

The basic functionality of the genetic algorithm is provided by the genetic operators. These are the functions that up the algorithm itself, the population and fitness function can be viewed as external entities that can be plugged in and changed. Even the operators and encoding can be allowed to adapt.

3.5.1 SELECTION

Selection is the process of deciding which individuals should be allowed to contribute to the next generation with their genetic material. This is the equivalent of the survival of the fittest; the individuals that get selected are the ones that survive long enough to reproduce. Selection can be done in a number of ways, but most often some form of relative and probabilistic selection is done. To do this the fitness function has to be normalized with the total fitness of the population.

3.5.1.1 ROULETTE WHEEL SELECTION

Parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. Imagine a roulette wheel where all the chromosomes in the population are placed. The size of the selection in the roulette wheel is proportional to the value of the fitness function of every chromosome- the bigger the value is, the larger the section is. The following diagram depicts the example-

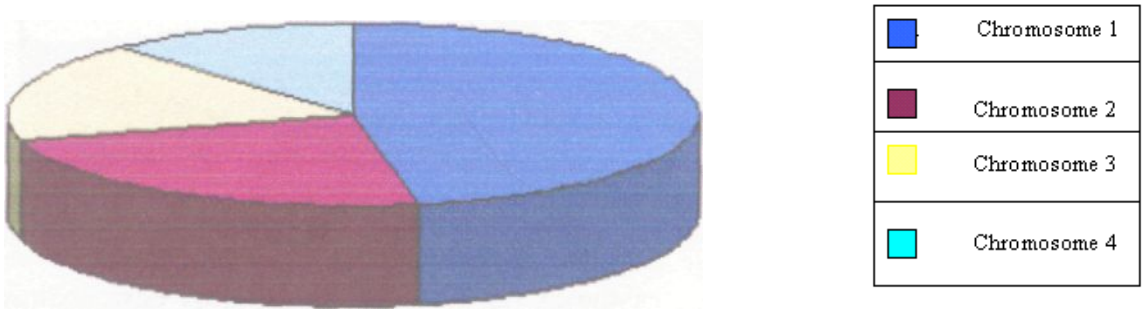


Figure 3.7: Roulette Wheel Selections

A marble is thrown in the roulette wheel and the chromosome where it stops, is selected. Clearly, the chromosome with bigger fitness value will be selected more times. The following algorithm can describe this process.

- [Sum] Calculate the sum of all the chromosomes' fatnesses in population- sum S
- [Select] Generate random number from the interval(0,S) – r
- [Loop] Go through the population and sum the fatnesses from 0- sum S.When the sum S is greater than r, stop and return the chromosome where you are.

Of course, the step 1 is performed only once for each population.

3.5.1.2 RANK SELECTION

Rank selection ranks the population first and then every chromosome receives fitness value determined by this ranking. The worst case will have the fitness 1, the second worst 2 etc. and the best will have fitness N (number of chromosomes in population).The following figure shows how the situation changes after changing fitness to the numbers determined by the ranking.

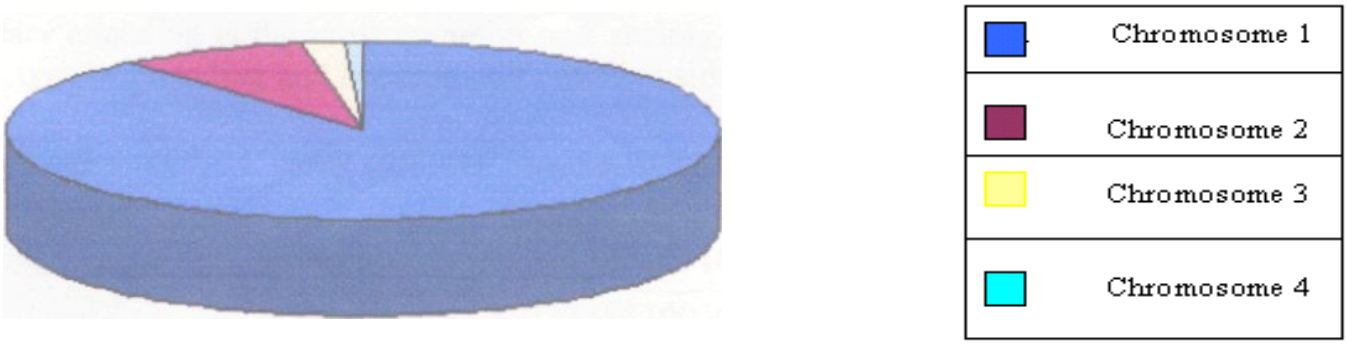


Figure 3.8: Situation Before Ranking

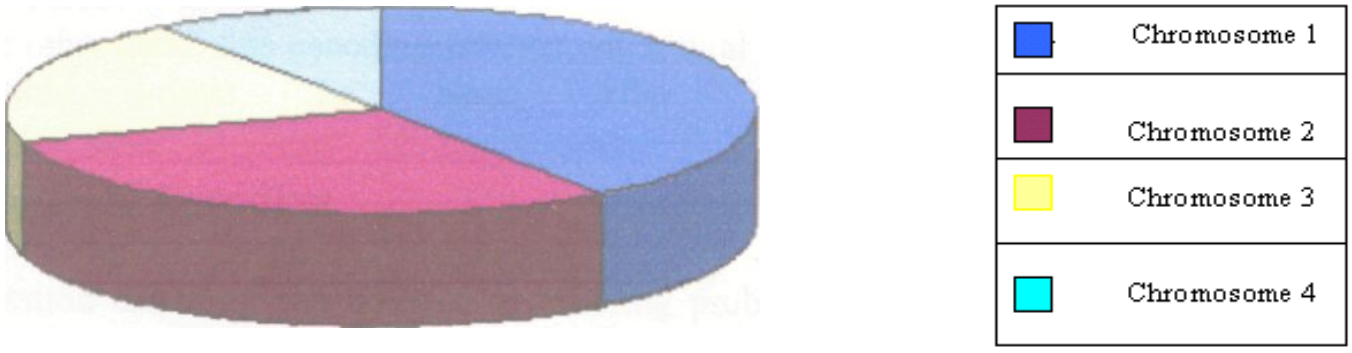


Figure 3.9: Situations After Ranking (Graph Of Fitness)

3.5.1.3 CROSSOVER

Crossover recombines genetic material from selected individuals and is inspired by reproduction in biological systems. There are many possible implementation of the crossover operator and the choice of operator is heavily dependent on the encoding. In the simple genetic algorithm, the standard crossover point is determined randomly. A new individual is created by combining the genes before the crossover point from the first individual and the trailing part from the second individual. The role of crossover is to recombine substrings that give high fitness under the assumption that combination of such partial solutions will yield individuals with higher fitness.

3.5.1.3.1 CROSSOVER PROBABILITY

This decides how often crossover will be performed. If there is no crossover, offspring are exact copies of parents. If there is crossover, offspring are made from parts of both parent's chromosome. If crossover probability is 100%, then all offspring are made by crossover. If it is 0%, whole new generation is made from exact copies of chromosomes from old population (but this does not mean that the new generation is the same!). Crossover is made in hope that new chromosomes will contain good parts of old chromosomes and therefore the new chromosomes will be better. However, it is good to leave some part of old population survives to next generation.

3.5.1.4 MUTATION

In natural evolution, mutation is a random process where one allele of a gene is replaced by another to produce a new genetic structure. In GAs, mutation is randomly applied with low probability, typically in the range 0.001 and 0.01, and modifies elements in the chromosomes. Usually considered as a background operator, the role of mutation is often seen as providing a guarantee that the probability of searching any given string will never be zero and acting as a safety net to recover good genetic material that may be lost through the action of selection and crossover. The mutation operator does some random changes to genes in the individuals. In the simple genetic algorithm, this is done by flipping bits in the genotypes based on a mutation probability. Mutation can be viewed as a way to preserve diversity of genes. Due to the selection pressure, all individuals in the population may come to have a one in the first bit position. Even though this has yielded the best solutions so far there is no guarantee that the optimal solution

will start with a one. With only crossover it would be impossible to obtain such strings in the scenario described. Mutation can be seen as a safeguard against such loss of diversity. Generation will be directly proportional to the chance that an instance of this schema will be selected. If we use a population of size n and a fitness function f this is

$$m(S,t+1) = m(S,t)n\frac{f(S)}{f_{avg}} \tag{3.6}$$

Where f_{avg}, is average fitness of the population. If we make a further simplification that the fitness of S will remain above the average by a constant c we get

$$f(S) = f_{avg}(1+c) \tag{3.7}$$

If we combine this with equation (3.7) we can see that

$$m(S,t) = m(S,0)(1+c)^t \tag{3.8}$$

This means that if the fitness of S is above the average fitness (positive c), the number of instances of S will grow exponentially and if the fitness is below average (negative c) the number of instances will decrease exponentially. The various mutation techniques are given below

- Bit inversion – selected bits are inverted.
- Order changing – two numbers are selected and exchanged

3.5.1.4.1 MUTATION PROBABILITY

This refers how often parts of chromosome will be mutated. If there is no mutation, offspring are generated immediately after crossover (or directly copied) without any change. If mutation is performed, one or more parts of a chromosome are changed. If mutation probability is 100%, whole chromosome is changed, if it is 0%, nothing is changed. Mutation generally prevents the GA from falling into local extremes. Mutation should not occur very often, because then GA will in fact change to random search.

3.6 MULTIOBJECTIVE OPTIMIZATION

Real world engineering design problems are usually characterized by the presence of many conflicting objectives, so the engineering design problems are treated as multi objective optimization problems. In a multi objective optimization problem (MOP) there may not exists one solution that is best with respect to all objectives. Usually the aim is to determine the trade off surface, which is a set of non dominated solution points, known as Pareto optimal (PO) or non inferior solutions. In view of the fact that one of the solutions in the non dominated set is absolutely better than any other, any one of them is an acceptable solution. The choice of one solution over the other requires problem knowledge and a number solution. The choice of one solution over the other requires problem knowledge and a number of problem related factors.

As most optimization problems are multi objective in nature there are many methods available to tackle this kind of problem. Generally the multi objective optimization problem (MOOP) is handled in four different ways depending upon when the decision maker articulates its preference concerning the different objectives: never, before, during or after the actual optimization procedure. In the first two approaches, the different two objectives are aggregated to one overall objective function. Optimization is then conducted with one optimal design as a result. The result is then strongly dependent on how the objectives were aggregated. Different methods have been developed in the literature to support the decision maker in aggregating the objectives. The third approach is an iterative process where the decision maker progressively articulates its preferences on the different objectives. The underlying assumptions are that once the search for an optimal solution has started and the decision makes has been presented with some alternatives, it will be better equipped to value the objectives. In the fourth and final approach, optimization is conducted without the decision maker articulating any preferences among the objectives. The outcome of this optimization is a set of Pareto optimal solutions, which elucidate the tradeoff between the objectives. The decision marker then has to trade the objectives against each other in order to select the final design. Thus optimization is conducted before the decision maker articulates his preferences.

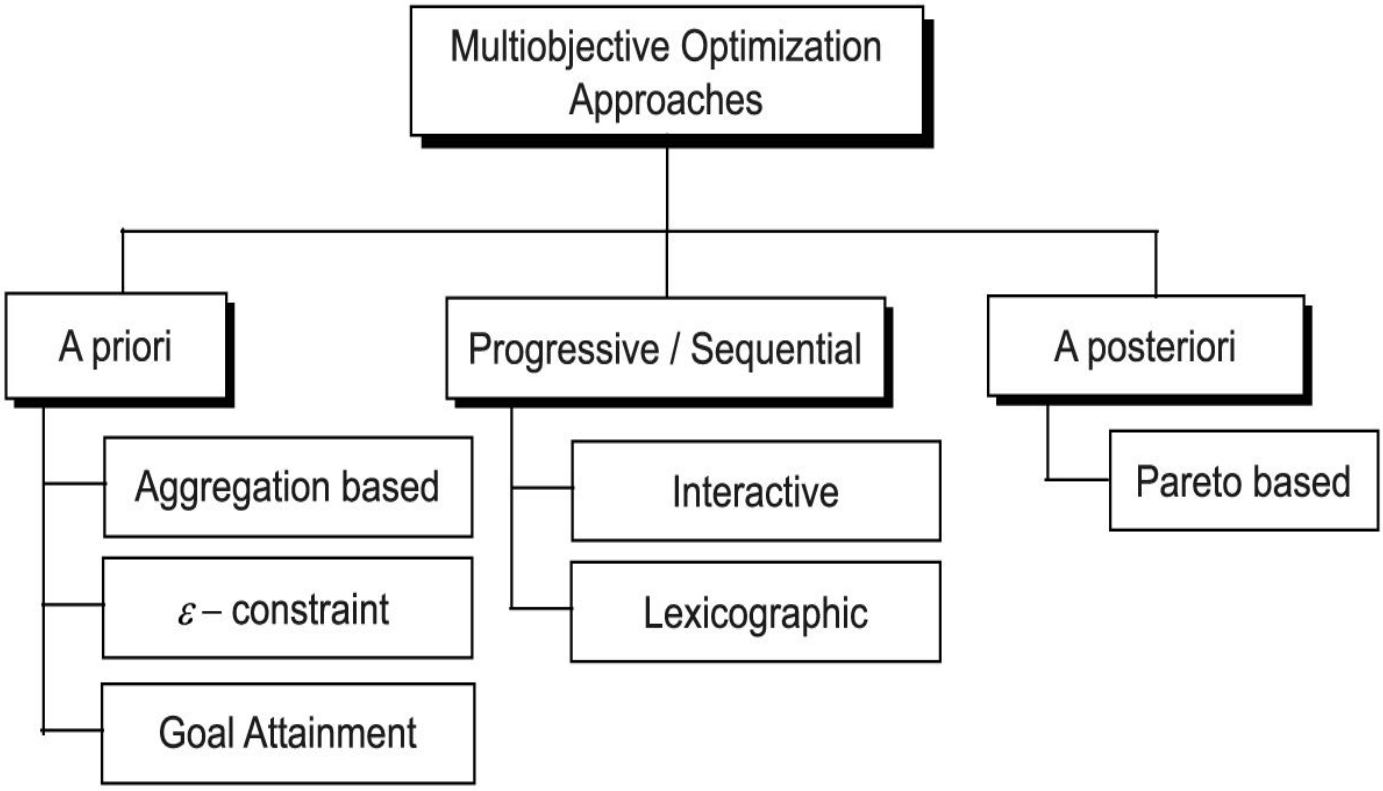


Figure 3.10: Classifications of Methods of Multi Objective Optimization

3.7 SOLUTION METHODOLOGIES

Combining the multiple objectives into one scalar objective whose solution is Pareto optimal point for the original MOP almost always solves the multi objective problem. Most algorithms have been developed in the linear frame work (linear frame work and linear constraints), some important techniques described below are used to solve multi objective optimization.

3.7.1 WEIGHTED SUM STRATEGY

The weighted sum strategy converts the multi objective problem of minimizing the vector F (X) into a scalar problem by constructing a weighted sum of all the objectives

$$\min \quad f(x) = \sum_{i=1}^{m} w_i F_i(x) \qquad (3.9)$$
$$x \in \Omega$$

Subjected to $\sum_{i=1}^{m} w_i = 1 \text{and } w_i \geq 0$ (3.10)

The problem is then optimized using a standard unconstrained optimization algorithm. The problem here is in attaching weighting coefficients to each of the objective. The weighting coefficients do not necessarily correspond directly to the relative importance of the objectives or allow tradeoffs between the objectives to be expressed. Further, the non inferior solution boundary can be non concurrent, so that certain solutions are not accessible. The Pareto optimal minimization of the combined function depends upon the user to choose the appropriate weights. Until recent, consideration of computational expense forced users to restrict themselves to perform only one such minimization. Newer, more ambitious approaches aim to minimize convex sums of the objectives for various settings of the weight, therefore generating various points in the Pareto set. Though computationally more expensive, this approach gives an idea of the shape of the Pareto surface and provides the user with more information about the trade off among the various objectives.

3.7.2 Є-CONSTRAINT METHOD

The method optimizes one of the objective functions while the required to have specified upper bounds. In other words, it minimizes one objective function and simultaneously maintains the maximum acceptable levels for the objective functions. The formulation adopted in this work is as follows:

Minimize $F_i(X), \qquad i = 1, 2.....r$ (3.11)

Subjected to $g_j(X) \leq 0, \qquad j = 1, 2.....m$ (3.12)

$h_j(X) = 0, \qquad j = 1, 2.....p$ (3.13)

$F_i(X) \leq c_j, \qquad j \quad 1, 2..=.....and\ i \neq j$ (3.14)

The selection of $F_i(X)$ and of this method are not c_j straightforward and depend on the particular problem under consideration. As shown in the above formulation, the optimization

problem can be solved for all $F_i(X)'s\ (i = 1, 2.....r)$ and the optimum solution that is best suited to the problem can be chosen among the r solutions. But this involves much computational effort. The choice c_j is arbitrary and any choice may also be used depending upon the problem. But the basic philosophy of this method does not alter with different selections of c_j. In general, the higher value of c_j's mean a wider feasible region for the single objective optimization problem and this may in turn give a more improved solution for $F_i(X)$ at the expense of the other objective functions.

3.7.3 GOAL PROGRAMMING METHOD

In this method, the designer sets goals to be attained for each objective and a measure of the deviations of the objective functions from their respective goals is minimized. The generalized goal programming goals b_j are specified for each objective function $F_j(X)$. Then the total deviation from goals $\sum_{j=1}^{k}\left|d_j\right|$ is minimized, where d_j is the deviation from the goal b_j for jth objective. To model the absolute values, d_j is split into positive and negative parts, such that $d_j = d_j^+ - d_j^-$, *with* $d_j^+ \geq 0$; d_j^- *and* $d_j^+\ d_j^- = 0$ *consequently*, $\left|d_j\right| = d_j^+ + d_j^-$. d_j^+ *and* d_j^- Represent underachievement and overachievement, respectively, where achievement implies that a goal has been reached. The optimization problem is formulated as follows:

$$\text{Minimize } = \sum_{i=1}^{k}\left(d_i^+ + d_i^-\right) \qquad (3.15)$$
$$x \varepsilon X, d^+, d^-$$

$$\text{Subjected to } F_j(x) + d_j^+ - d_j^- = b_j, \qquad j = 1, 2.....k \qquad (3.16)$$

$$d_j^+ - d_j^- \geq 0, \qquad j = 1, 2.....k \qquad (3.17)$$

$$d_j^+ d_j^- = 0, \qquad j = 1, 2.....k$$

3.7.4 NORMAL BOUNDARY INTERSECTION (NBI) METHOD

This method provides a means for obtaining an even distribution of Pareto optimal points for a consistent variation in the user-supplied parameter vector w, even with a non convex Pareto optimal set. The approach is formulated as follows.

$$\begin{aligned} &\text{Minimize } \lambda \\ &x \varepsilon X, \lambda \\ &\text{Subjected to } \phi w + \lambda n = F(x) - F^0 \end{aligned} \tag{3.18}$$

Φ is a $k \times k$ pay off matrix in which the ithcolumn is composed of the vector $F(x_i *) - F^0$, where $F(x_i *)$ is the vector of objective functions evaluated at the minimum of the ith objective function. The diagonal elements of Φ are zeros. W is a vector of scalars such that $\sum_{i=1}^{k} w_i = 1$ and $w \geq 0. n = - \Phi$ e, where e ϵ R^k is column vector of ones in the criterion space. n is called a quasi-normal vector. Since each component of Φ is positive, the negative sign ensures that n points towards the origin of the criterion space. n is gives the NBI method the property that for any 'w', a solution point is independent of how the objective functions are scaled. As 'w' is systematically modified, the solution to (2.10) yields an even distribution of Pareto optimal points representing the complete Pareto set. Technically, the NBI method finds the portion of the boundary of Z that contains the Pareto optimal points. However, the method may also yield non Pareto optimal points; it does not provide a sufficient condition for Pareto optimally. This is not necessarily a disadvantage, since such points help construct a smoother approximate of the Pareto boundary.

3.8 APPLICATIONS OF GA

Genetic algorithms have been used for difficult problems (such as NP-hard problems), for machine learning and also for evolving simple programs. They have been also used for some art, for evolving pictures and music. The advantage of GAs is in their parallelism. GA is traveling in a search space using more individuals (and with genotype rather than phenotype) so that they are less likely to get stuck in a local extreme like the other methods. They are also easy to implement. Once you have the basic GA algorithm implemented, you have just to write a new

chromosome (just one object) to solve another problem. With the same encoding you just change the fitness function, and you are done. However, for some problems, choosing and implementation of encoding and fitness function can be difficult. The disadvantage of GAs is in the computational time. GAs can be slower than other methods. But since we can terminate the computation in any time, the longer run is acceptable (especially with faster and faster computers). To get an idea about some problems solved by GAs, here is a short list of some applications:

- Nonlinear dynamical systems - predicting, data analysis
- Designing neural networks, both architecture and weights
- Robot trajectory
- Evolving LISP programs (genetic programming)
- Strategy planning
- Finding shape of protein molecules
- TSP and sequence scheduling
- Functions for creating images

CHAPTER 4

PROBLEM FORMULATION

Signal processing theory plays an increasingly central role in the development of modern telecommunication and information processing systems, and has a wide range of applications in multimedia technology, audio-visual signal processing, cellular mobile communication, adaptive network management, radar systems, pattern analysis, medical signal processing, financial data forecasting, decision making systems, etc. The theory and application of signal processing is concerned with the identification, modeling and utilization of patterns and structures in a signal process.

4.1 POWER DISSIPATION IN FIR FILTERS

Each step in FIR filter algorithm involves getting the appropriate coefficient and data value and performing multiply-accumulate computation. Thus address and data buses of both the memories and multiplier see experiences the most signal switching activity during FIR filtering. Hence these form the main sources of power dissipation.

4.1.1 MEASURES OF POWER DISSIPATION IN BUSES

Signal switching activity is the major component of power dissipation in CMOS circuits. The power dissipation depends both on the frequency of switching and capacitive loading of the signal. For a typical embedded processor, address and data buses are networks with a large capacitive loading. Hence signal in these networks has a significant impact on power consumption. In addition to the capacitance of each signal, inter signal capacitance also contributes to bus power dissipation. The power dissipation due to inter signal capacitance varies depending upon the adjacent signal values. The current required for signals to switch between 5's (0101) and A's (1010) is about 25% more than the current required for the signals to switch between 0's (0000) and FFFF.

4.1.2 MEASURES OF POWER DISSIPATION IN MULTIPLIER

Due to high speed requirements, parallel array architecture are used for implementing dedicated multiplier in programmable DSPs .The power dissipation of the multiplier is directly proportional to the number of switching at all the internal nodes of the multiplier. The number of internal node switching depends on the multiplier input values. This dependence can be analyzed using the measure of circuit activity the transition density. Transition density of a signal is the average number of transition depends on the transition densities and the probabilities of the multiplier inputs. The transition density of the multiplier inputs depends upon the Hamming distance between successive input values. The input signal probability depends upon the number of 1s in the input values of the multiplier. These two thus forms the measures of multipliers power dissipation. It can also be shown that the transitions in least significant (LSBs) of the multiplier input contribute more to the power dissipation than the most significant bits (LSBs) of the multiplier input contribute more to the power dissipation than the most significant bits (MSBs). Thus while minimizing transition densities of all the input is important, larger gains are achieved by focusing on lower order bits of the input signal.

4.1.3 POWER REDUCTION IN DATA BUSES AND MULTIPLIERS

During FIR filtering, the coefficient and data memory data buses provide successive coefficients and data values for the weighted sum computation. The power dissipation in the coefficient memory data bus hence depends on the successive coefficient values and the power dissipation in the data memory data bus depends upon the successive data values. Since the data memory value forms the input to the FIR filter, the data memory values forms the inputs to the FIR filter, the data bus power dissipation cannot be controlled during FIR filter synthesis. The coefficient memory data bus power however can very much be minimized by optimizing the filter coefficient so to reduce the Hamming distance between the successive coefficients values and also by reducing the total number of signal toggling in opposite directions between successive coefficients. The coefficients and the input data samples form the inputs to the multiplier during FIR filtering. The multiplier power dissipation thus depends upon the number of toggles and also on the number of 1s in these inputs. The coefficient optimization for reducing the coefficient

memory data bus power thus also reduces the multiplier power. Higher power reduction can be achieved by focusing on the lower significant bits of the coefficients during minimization.

4.2. HAMMING DISTANCE MINIMIZATION ALGORITHM

4.2.1 PROBLEM DEFINITION

The Hamming distance minimization problem using Steepest Decent approach stated as follows For a Given N-tap FIR filter with coefficient $A_i,\ i = 0,\ N-1$ that satisfy the filter response in terms of pass band ripples, stop band attenuation and linear phase, find a new set of coefficient $A_i,\ i = 0,\ N-1$ such that the total Hamming distance between successive coefficients is minimized while still satisfied the desired filter characteristics in terms of pass band ripple and stop band attenuation. Also retain the linear phase characteristics if such this constraint is specified.

4.2.2 PROBLEM FORMULATION

The Hamming Distance minimization problem is formulated as a local search problem, where the optimum coefficient values are searched in their neighborhood. This is done by using an iterative improvement process. During the each iteration one or more coefficients are suitably modified so as to reduce the total Hamming distance while still satisfying the desired filter characteristics. The optimization process continues till no further reduction is possible.

The coefficient optimization is done in two phases. In the first phase, all the coefficients are scaled uniformly. The advantage of such an approach is that it does not affect the filter characteristics in terms of pass band ripples and stop band attenuation and phase response. The sealing results in the same gain /attenuation ratio. In the second phase of optimization one coefficient is perturbed in the each iteration. In case of requirement to retain the linear phase characteristics, the coefficients are perturbed in pairs (A_i and A_{n-1-i}) so as to preserve coefficients symmetry. The selection of coefficient for perturbation and the amount of perturbation has the direct impact on overall optimization quality. Various strategies can be adopted for coefficient perturbation. The strategies adopted here include ‘Genetic Algorithms’. The Genetic Algorithms

are the evolutionary algorithm which generates the random numbers and selects the best fit value according to the fitness function and search the whole space to find the global value.

4.3 HAMMING DISTANCE MINIMIZATION ALGORITHM-COMPONENTS

4.3.1 COEFFICIENT SEALING

The first phase of the algorithm involves uniformly scaling the coefficient so as to reduce the total Hamming distance between successive coefficients. For N-tap filter with N coefficients $(A_i,\ i = 0,\ N-1)$, the output Y (n) is given by equation.

$$Y(n) = \sum_{i=0}^{N-1} (A_i * X_{n-1}) \tag{4.1}$$

Scaling the output preserve the filter characteristics in terms of pass band ripples and stop band attenuation, but results in overall magnitude gain equal to the scale factor. For a scale factor K, from equation (5.1)

$$K * Y(n) = K * (\sum_{i=0}^{N-1} (A_i * X_{n-1})) = \sum_{i=0}^{N-1} ((K * A_i) * X_{n-1}) \tag{4.2}$$

Thus the coefficient of a scaled filter is given by $(K * A_i)$. The scaled coefficients have different bit patterns in their binary representation than the unscaled coefficients. Given the allowable range of scaling $(\pm 3\text{db})$, an optimal scaling factor can be found such that the total Hamming distance between successive coefficients $(\mathrm{K} * \mathrm{A_i})$ is least. The scaled coefficients from the initial solution for the phase second of the algorithm.

4.3.2 COMPUTING FILTER RESPONSE

In phase second of the algorithm, the filter characteristics are computed for every perturbation. The overall computational efficiency of the algorithm thus depends on how fast the filter characteristics (pass band ripple and stop band attenuation) are derived. The frequency response of the filter is can be computed by taking Fourier Transform of its unit impulse response .The frequency response of an FIR filter can be computed by taking Fourier Transform of a sequence

consisting of a filter coefficients padded with 0s. The radix-2 FFT (Fast Fourier Transform) technique is used to compute the filter frequency response. The numbers of zeros to the make the total number of points a power of 2. Multiply the number of coefficients with 8 and pick the nearest highest power of 2 to decide the total number of points for FFT computation. The pass band and stop band frequency ranges, the pass band and stop band attenuation are be computed by analyzing the frequency response.

4.3.3 COEFFICIENT PERTURBATION

Perturbing a coefficient, the selected coefficient A_i is modified in such a way that the Hamming distance between the new coefficient value A_i and its adjacent coefficient values $(A_i - 1)$ is less than the Hamming distance between the current coefficient values A_i and the adjacent coefficient values. The change in coefficient value needs to be as small as possible, so as to minimally impact the filter characteristics. The neighborhood of the selected coefficient (A_i) is searched so as to find the nearest highs and near lower coefficient so as to reduce the Hamming distance. The maximum allowable difference between the perturbed coefficients and the original coefficients can be controlled so as to focus on the lower significant bits during the optimization.

4.3.4 COEFFICIENT SELECTION

Using Steepest Decent strategy, for every coefficient its nearest higher and nearest lower coefficient values are identified. A new set of coefficient is formed by replacing one of the coefficient with its nearest higher or nearest lower value. This approach is used to generate 2N set of coefficients for an N tap filter, during the each iteration of the optimization process. From the 2N sets of coefficients for the next iteration, the gain function γ is computed as follows.

$$\gamma = \left(P_{db\_req} - P_{db}\right) / P_{db\_req} + \left(S_{db\_req} - S_{db}\right) / S_{db\_req} * HD_{\_red} \tag{4.3}$$

Where $HD_{\_red}$ is the reduction in the total Hamming distance for the new set of coefficients compared to the total Hamming Distance for the current set of coefficients, P_{db_req} is the desired pass band ripple, S_{db_req} is the desired stop band attenuation, P_{db} is the pass band ripple of new set of coefficients and S_{db} is the stop band attenuation for the new set of coefficients.

4.3.5 ALGORITHMS FOR HAMMING DISTANCE MINIMIZATION

Step1:- For a given FIR filter coefficients A[i] (i = 1, N-1) and given pass band ripples $\left(P_{db\_req}\right)$ and stop band attenuation $\left(S_{db\_req}\right)$. Calculate the Hamming Distance between.

A[i], A[i-1] and A[i], A[i+1]

Step 2:- Now perturb each coefficient (increase the value of each coefficient one by one by 1) and calculate new hamming distance between the coefficients.

A[i+], A[i-1] and A[i+], A[i+1]

Such that

HD(A[i], A[i-1]) +HD (A[i], A[i+1])> HD(A[i+], A[i-1])+HD(A[i+], A[i+1])

And

Euclidian distance (A[i+], A[i+1]) is minimum

Step 3:- Replace A[i+],and A[i+1] to get a new set of coefficients.

Step 4:- Compute pass band ripples $\left(P_{dbi+}\right)$ and stop band attenuation $\left(S_{dbi+}\right)$ from a new set of coefficient A[i+]

Step 5:- If pass band ripples $P_{dbi+} < P_{db\_req}$ and stop band attenuation $S_{dbi+} > S_{db\_req}$ calculate tolerance

$$T_{oli} = \left(P_{db\_req} - P_{dbi+}\right)/ P_{db\_req} + \left(S_{db\_req} - S_{dbi+}\right)/ S_{db\_req}$$

Else

$Tol_{i+} = 0$

Step 6:- Now again perturb each coefficient (decrease the value of each coefficient one by one 1) and calculate new hamming distance between the coefficients.

A[i-], A[i-1] and A[i-], A[i+1]

Such that

HD(A[i], A[i-1]) +HD (A[i], A[i+1])> HD(A[i-], A[i-1])+HD(A[i-], A[i+1])

And

Euclidian distance (A[i-], A[i+1]) is minimum

Step7:- Replace A[i] with A[i-] to get a new set of coefficients.

Step8:- Compute pass band ripple (P_{dbi-}) and stop band attenuation (S_{dbi-}) from a new set of coefficient A[i].

Step9:- If pass band ripples $P_{dbi-} < P_{db\_req}$ and stop band attenuation $S_{dbi-} > S_{db\_req}$ calculate tolerance

$$T_{oli-} = (P_{dbreq} - P_{dbi-}) / P_{dbreq} + (S_{dbi-} - S_{dbreq}) / S_{dbreq}$$

Else

$Tol_{i-} = 0$

Step10:- Calculate gain function γ for new coefficient values *A[i+] and [i-]*

γ = (Tolerance * HD reduction) is maximum

for γ >0

Replace original coefficients with new value

4.4 GENETIC APPROACH FOR HAMMING DISTANCE MINIMIZATION OF FIR FILTER

Genetic Algorithms are successfully used for the design of FIR filters. The problem is formulated as error minimization between the Ideal frequency response and the desired frequency response as per the design specification in terms of pass band ripple, stop band

attenuation and linear phase. Here one more objective added is the Hamming distance between the successive values of the designed filter should be minimum than the ideal filter coefficients. As the Hamming Distance is the measure of the signal switching activity it should also be minimized to reduce the power dissipation in the multipliers while implementing the FIR filtering operation on digital signal processors. So the problem is multi objective optimization problem and it is solved by using weighted sum approach, converting the problem into single objective by assigning the appropriate weights. The problem is then solved using Genetic Algorithms.

4.4.1 PROBLEM FORMULATION

Minimax and least mean square errors are generally used to evaluate the fitness function of Chromosome (filter coefficients). But minimax error criteria produced better results with higher speed than mean square error strategy. Digital FIR filter transfer function is defined as

$$H(z,\phi)=\sum_{n=0}^{N} a_n z^{-n} \tag{4.4}$$

Where ϕ is the vector of filter coefficients, [a_1 a_2 a_3 a_4 a_5……a_{N-1} a_N] and N is the order of filter. The corresponding freq. response is obtained by substituting $Z = e^{-j\omega nt}$

$$H(\omega,\phi)=\sum_{n=0}^{N} a_n e^{-j\omega nT} \tag{4.5}$$

Where T and ω are the sampling period and angular frequency respectively

4.4.2 MINIMAX ERROR

Minimax objective function is defined as follows. It searches for the maximum peak error throughout a discrete frequency domain by subtracting the magnitude response of the ideal filter to designed filter. The minimax criterion minimizes the maximum error between the filter frequency response $H_D(\omega,\phi)$ and ideal response $H_1(\omega)$. Beside the frequency response error, hamming distance is also minimized. So our problem becomes multi objective optimization

problem having two objectives to minimize the maximum frequency error and to minimize the Hamming distance

$$\text{Error} = \max\left|H_1(\omega) - H_D(\omega)\right|$$

Where $H_1(\omega)$ and $H_D(\omega,\phi)$ are defined as the frequency responses of the ideal and designed filters.

4.4.3 LEAST MEAN SQUARE (LMS) ERROR

Least mean square error objective function is defined as the sum of the errors throughout a discrete frequency domain by subtracting the magnitude response of the ideal filter to designed filter.

$$E_{tot} = Error = \left\{\sum \left[\left|H_1(\omega) - \left|H_D(\omega,\phi)\right|\right|\right]^2\right\}^{\frac{1}{2}} \tag{4.6}$$

Minimax strategy generates better results than least mean square error for binary encoding and roulette wheel selection.

4.4.4 FITNESS FUNCTION

The Genetic algorithm must find a design that satisfies user specifications and minimizes transition activity of digital gates. These two objectives, however, are hierarchical: specification fulfillment is mandatory, while reduction of Hamming distance is an additional quality

Therefore, a two-step fitness function has been devised. First of all, we consider filter specifications (given as a frequency mask): the frequency range is sampled at N equally spaced frequencies ω, and the filter response $H(\omega)$ is calculated for every ω in the pass-band and in the stop-band. The partial error ϵ_i is set to zero, if $H(\omega)$ lies within the specifications, otherwise it is proportional to the overshoot or undershoot with respect to the given tolerance. The total error E_{tot} is simply the sum of all partial errors. Fitness function is defined as

$$f_M = \frac{1}{1 + E_{tot}} \tag{4.7}$$

It measures the extent to which the filter complies with frequency specifications and $f_M = 1$ when specifications are completely met. The second step is the evaluation of the activity fitness, a measure of the Hamming distance for this the total Hamming distance (HD_1) of the ideal filter is calculated. The filter is designed such that the Hamming distance of the designed filter should be the less than that. The fitness function in defined as

$$f_M = HD_1 - HD_D \tag{4.8}$$

Where, N_T is the number of weighted digital transitions per input sample and a is a constant. Its contribution is higher for filters with low transition activity, which is responsible for power consumption. Finally, the overall fitness defined as.

$$f = f_M + f_M \tag{4.9}$$

4.4.5 SOLUTION METHODOLOGY

The intent of work is to optimize the coefficients of FIR filter, to minimize the Hamming distance and satisfying the desired filter characteristics in terms of pass band ripple and stop band attenuation. This section is organized to give a solution methodology, which minimizes the Hamming distance and least mean square error simultaneously in multi objective frame work.

The multi objective problem of minimizing the Hamming distance and mean square error is converted into a scalar problem by constructing a weighted sum of the objectives to generate Pareto optimal solution. The Pareto optimal solutions for different simulated weight combination are generated considering both the objectives simultaneously. To simulate weight combination, weights $w_{i,}\ i = 1, 2, \ldots\ldots\ldots L$ are varied from 0.1 to 1.0 in steps 0.1, so that their sum is 1.0. The weighting coefficients $w_1 \text{ and } w_2$ are used to select of error and Hamming distance. The weighted objective function is written as

$$f = w_1 f_M + w_2 f_H \tag{4.10}$$

Here $f_M \text{ and } f_H$ are the fitness functions for Mean square error and Hamming distance. The scalar optimization mentioned above is solved using Genetic Algorithm the random number

population is generated and the chromosomes are selected based upon the maximum fitness of the fitness function using the roulette wheel selection. The Genetic operator's crossover and mutation are applied; uniform crossover is applied at the defined in a mating pool to produce the new generation which are maximally fit. Then mutation is applied to further enhance the fitness at a defined rate, mutation increases the search space of the Genetic Algorithm. This process is repeated till some termination criterion is met and the value of fitness, error. Hamming distance and filter coefficients are saved.

4.4.6 ALGORITHM FOR HAMMING DISTANCE MINIMIZATION USING GA

Step1 Compute filter coefficients $h_1(n)$ and freq. response $H_1(\omega)$of ideal FIR filter for $0 \le n \le N-1$.

Step 2 Calculate the Hamming distance (HD_1) between the FIR filter coefficients $h_1(n)$.

Step 3 Set the Number of chromosomes *(k)*, mutation rate *(m)*, Cross over rate *(c),* Stopping criteria.

Step 4 Populate *k* sets of possible designed solutions, to produce symmetric coefficients $H_D(\mathrm{n})0 \le i \le k-1$ and $0 \le n \le N-1$

Step 5 Compute the frequency response of the coefficients chromosomes for $H_D(\omega)$ in population.

Step 6 Calculate the Hamming distance in each of the coefficient chromosome.

Step 7 Evaluate the fitness of the chromosomes

$$f(i) = f_M + f_H$$

Step 8 Apply Roulette wheel selection.

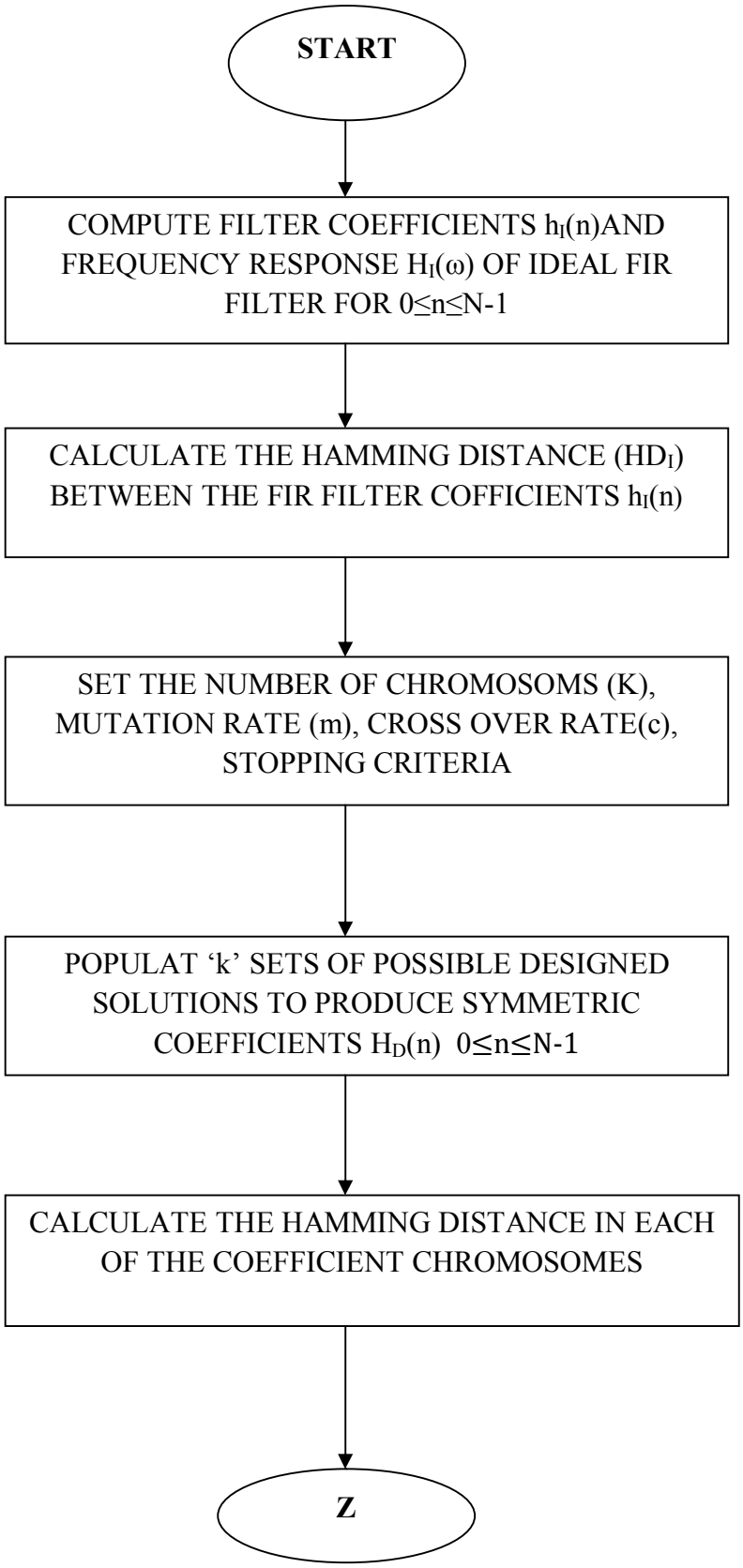
Step 9 Apply crossover operators at a desired rate.

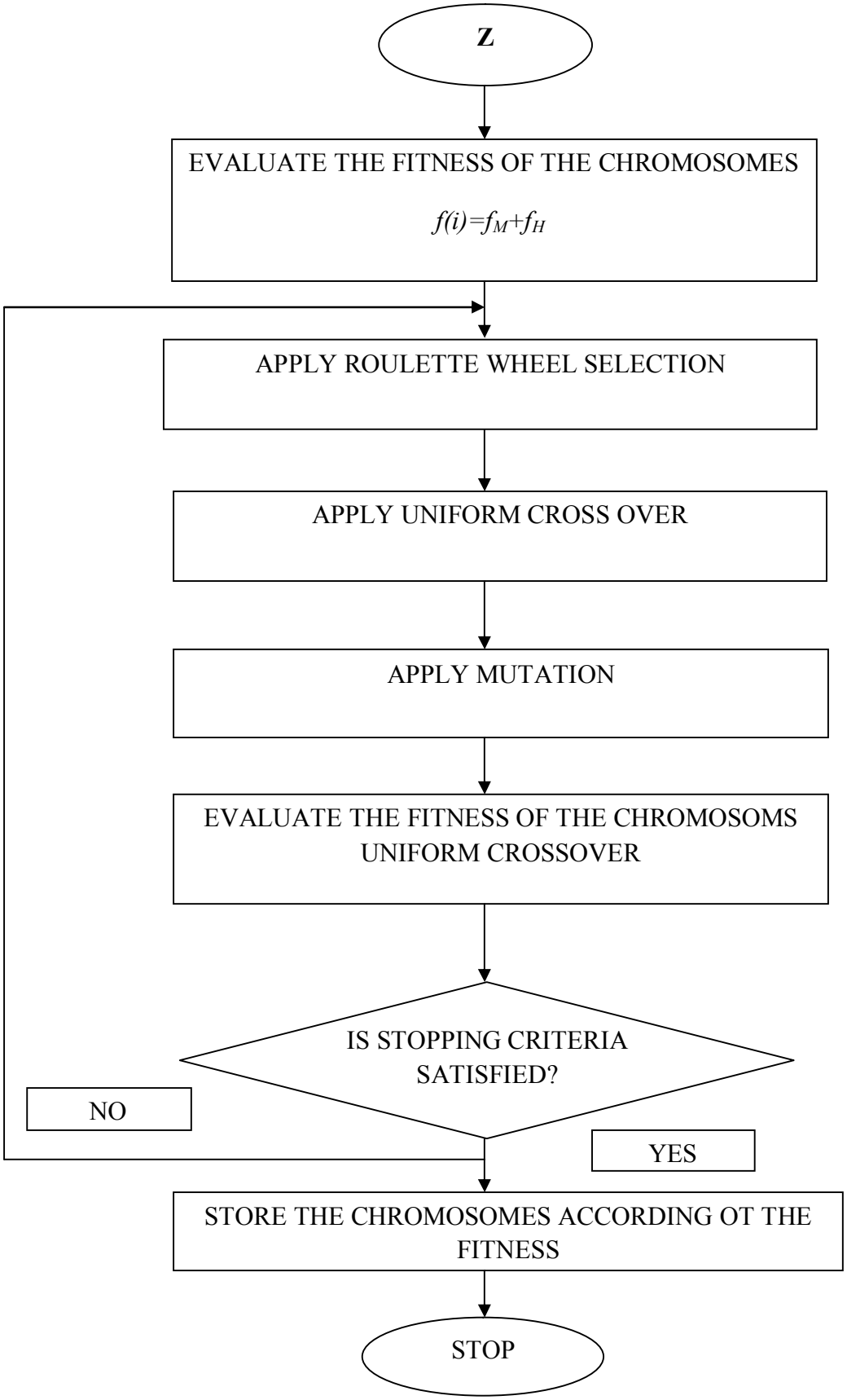
Step 10 Mutate at a desired rate

Step 11 Evaluate again the fitness of the chromosomes.

Step 12 If the Stopping criterion is met store the chromosomes according to the fitness, else go to stop 8.Flow chart for hamming distance minimization using G.A. is given in the next section

4.4.7 FLOW CHART FOR HAMMING DISTANCE MINIMIZATION USING GA




Z
EVALUATE THE FITNESS OF THE CHROMOSOMES
$f(i)=f_M+f_H$
APPLY ROULETTE WHEEL SELECTION
APPLY UNIFORM CROSS OVER
APPLY MUTATION
EVALUATE THE FITNESS OF THE CHROMOSOMS UNIFORM CROSSOVER
IS STOPPING CRITERIA SATISFIED?
NO
YES
STORE THE CHROMOSOMES ACCORDING OT THE FITNESS
STOP

CHAPTER 5

SIMULATION AND TESTING

5.1 OUTPUT OF THE ‘C’ PROGRAMME FOR CLCULATING TOGGALING

ENTER THE NUMBER OF COEFFICENT = 8

ENTER THE VALUE ONE BY ONE

ENTER THE 1ST VALUE = .1234

ENTER THE 2ST VALUE = .2345

ENTER THE 3ST VALUE = .3456

ENTER THE 4ST VALUE = .4567

ENTER THE 5ST VALUE = .5678

ENTER THE 6ST VALUE = .6789

ENTER THE 7ST VALUE = .7890

ENTER THE 8ST VALUE = .8901

THE NET VALUSES CONSIDERED ARE…

0.123400

0.234500

0.345600

0.456700

0.567800

0.678900

0.789000

0.890100

THE BINARY CONVERSION OF THE COEFFICIENTS ARE …..

0 0 0 1 1 1 1 1 1 0 0 1 0 1 1

0 0 1 1 1 1 0 0 0 0 0 0 1 0 0

0 1 0 1 1 0 0 0 0 1 1 1 1 0 0

0 1 1 1 0 1 0 0 1 1 1 0 1 0 1

1 0 0 1 0 0 0 1 0 1 0 1 1 0 1

1 0 1 0 1 1 0 1 1 1 0 0 1 1 0

1 1 0 0 1 0 0 1 1 1 1 1 1 0 1

1 1 1 0 0 0 1 1 1 1 0 1 1 1 0

THE TOTAL TOGGLING = 54

5.2 OUTPUT OF THE ‘C’ PROGRAMME FOR CLCULATING HAMMING DISTANCE

5.2.1 CASE.1: IF COEFFICIENTS ARE POSITIVE

ENTER THE NO. OF PARAMETERS: 8

ENTER THE VALUES OF PARAMETERS IN FRACTION.....

.1234

.2345

.3456

.4567

.5678

.6789

.7890

.8901

THE VALUES OF PARAMETERS ARE: 0.123400 0.234500 0.345600

0.456700 0.567800 0.678900 0.789000 0.890100

TRUNCATED VALUES ARE FOLLOWING:

0.123400

0.234500

0.345600

0.456700

0.567800

0.678900

0.789000

0.890100

0.345600

0.456700

0.567800

0.678900

0.789000

0.890100

BINARY EQUIVALENT OF ENTERED COEFFICIENTS

BINARY EQUIVALENT OF **0.123400** IS **111110**/10 TO POWER 9

BINARY EQUIVALENT OF **0.234500** IS **1111000**/10 TO POWER 9

BINARY EQUIVALENT OF **0.345600** IS **10110000**/10TO POWER 9

BINARY EQUIVALENT OF **0.456700** IS **11101000**/10TO POWER 9

BINARY EQUIVALENT OF **0.567800** IS **100100010**/10TO POWER 9

BINARY EQUIVALENT OF **0.678900** IS **101011010**/10TO POWER 9

BINARY EQUIVALENT OF **0.789000** IS **110010010**/10TO POWER 9

BINARY EQUIVALENT OF **0.890100** IS **111000110**/10TO POWER 9

EXOR OF **0.123400** AND **0.234500** IS **1000110**/10 TO POWER 9

EXOR =70\N

NUMBER OF ONES =3

EXOR OF **0.234500** AND **0.345600** IS **11001000/10** TO POWER 9

EXOR =200\N

NUMBER OF ONES =3

EXOR OF **0.345600** AND **0.456700** IS **1011000/10** TO POWER 9

EXOR =88\N

NUMBER OF ONES =3

EXOR OF **0.456700** AND **0.567800** IS **111001010**/10 TO POWER 9

EXOR =458\N

NUMBER OF ONES =5

EXOR OF **0.567800** AND **0.678900** IS **1111000**/10 TO POWER 9

EXOR =120\N

NUMBER OF ONES =4

EXOR OF **0.678900** AND **0.789000** IS **11001000**/10 TO POWER 9

EXOR =200\N

NUMBER OF ONES =3

EXOR OF **0.789000** AND **0.890100** IS **1010100**/10 TO POWER 9

EXOR =84\N

NUMBER OF ONES =3

TOTAL HAMMING DISTANCE AMONG THE FILTER COEFFICIENTS**=24**

5.2.2 CASE.2: IF COEFFICIENTS ARE NIGATIVE

ENTER THE NO. OF PARAMETERS: 5

ENTER THE VALUES OF PARAMETERS IN FRACTION.....

-.1234

-.2345

-.3456

-.4567

-.5678

THE VALUES OF PARAMETERS ARE: -0.123400 -0.234500 -0.34560

-0.456700 -0.567800

TRUNCATED VALUES ARE FOLLOWING:

-0.123390

-0.234490

-0.345600

-0.456690

-0.567790

BINARY EQUIVALENT

BINARY EQUIVALENT OF **-0.123390** IS **-111110**/10TO POWER 9

BINARY EQUIVALENT OF **-0.234490** IS **-1111000**/10TO POWER 9

BINARY EQUIVALENT OF **-0.345600** IS **-10110000**/10TO POWER 9

BINARY EQUIVALENT OF **-0.456690** IS **-11101000**/10TO POWER 9

BINARY EQUIVALENT OF **-0.567790** IS **-100100010**/10TO POWER 9

EXOR OF **-0.123400** AND **-0.234500** IS **1001010**/10 TO POWER 9

EXOR =74\N

NUMBER OF ONES =3

EXOR OF **-0.234500** AND **-0.345600** IS **11011000**/10 TO POWER 9

EXOR =216\N

NUMBER OF ONES =4

EXOR OF **-0.345600** AND **-0.456700** IS **1001000**/10 TO POWER 9

EXOR =72\N

NUMBER OF ONES =2

EXOR OF **-0.456700** AND **-0.567800** IS **111000110**/10 TO POWER 9

EXOR =454\N

NUMBER OF ONES =5

TOTAL HAMMING DISTANCE AMONG THE FILTER COEFFICIENTS =**14**

5.3 OUTPUT OF MATLAB DESIGNING CODE FOR WINDOWS

5.3.1 RECTANGULAR WINDOW

%THE PROGRAM FOR HANNING WINDOW GENERATION HAVING 51 SAMPLES IN MATLAB

N=51;

wvtool(boxcar(n));

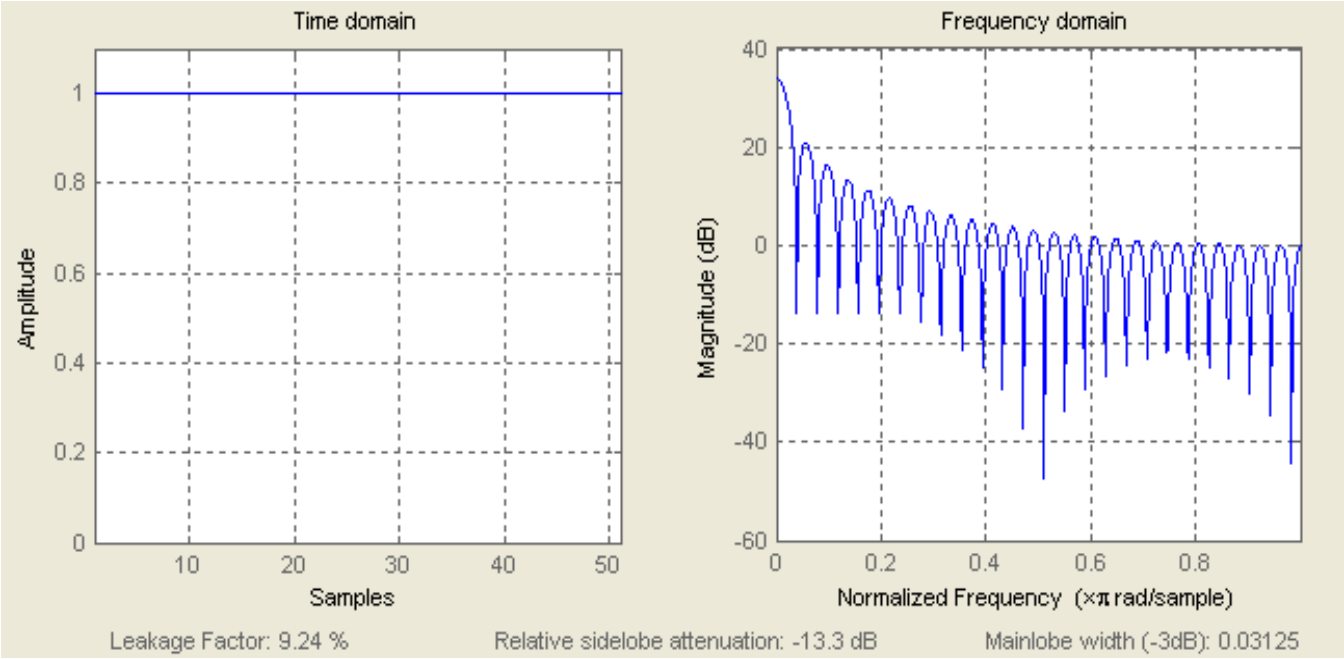


Figure 5.1: Representation of Rectangular Window in Matlab

5.3.2 HANNING WINDOW

%THE PROGRAM FOR HANNING WINDOW GENERATION HAVING 51 SAMPLES IN MATLAB

N=51;

wvtool(hann(n));

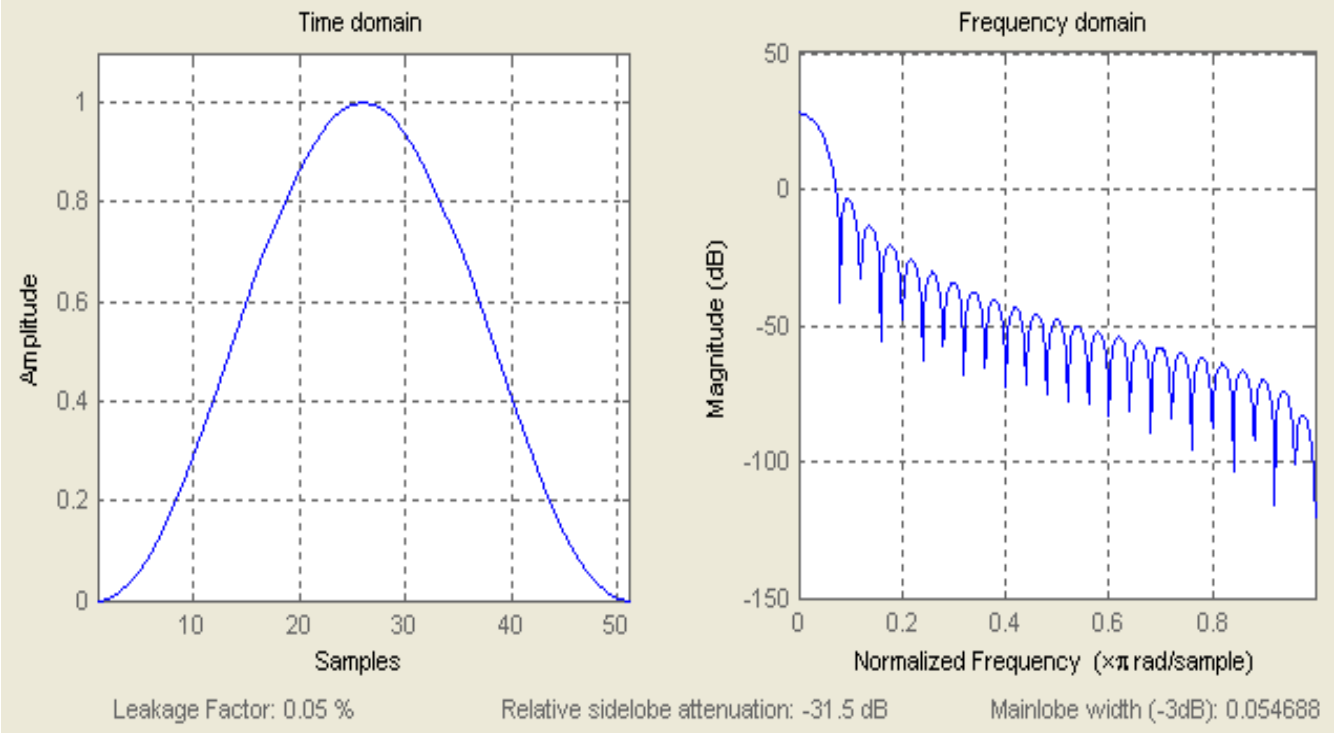


Figure 5.2: Representation of Hamming Window in Matlab

5.3.3 HAMMING WINDOW

%THE PROGRAM FOR HAMMING WINDOW GENERATION HAVING 51 SAMPLES IN MATLAB

N =51;

wvtool(hamming(n));

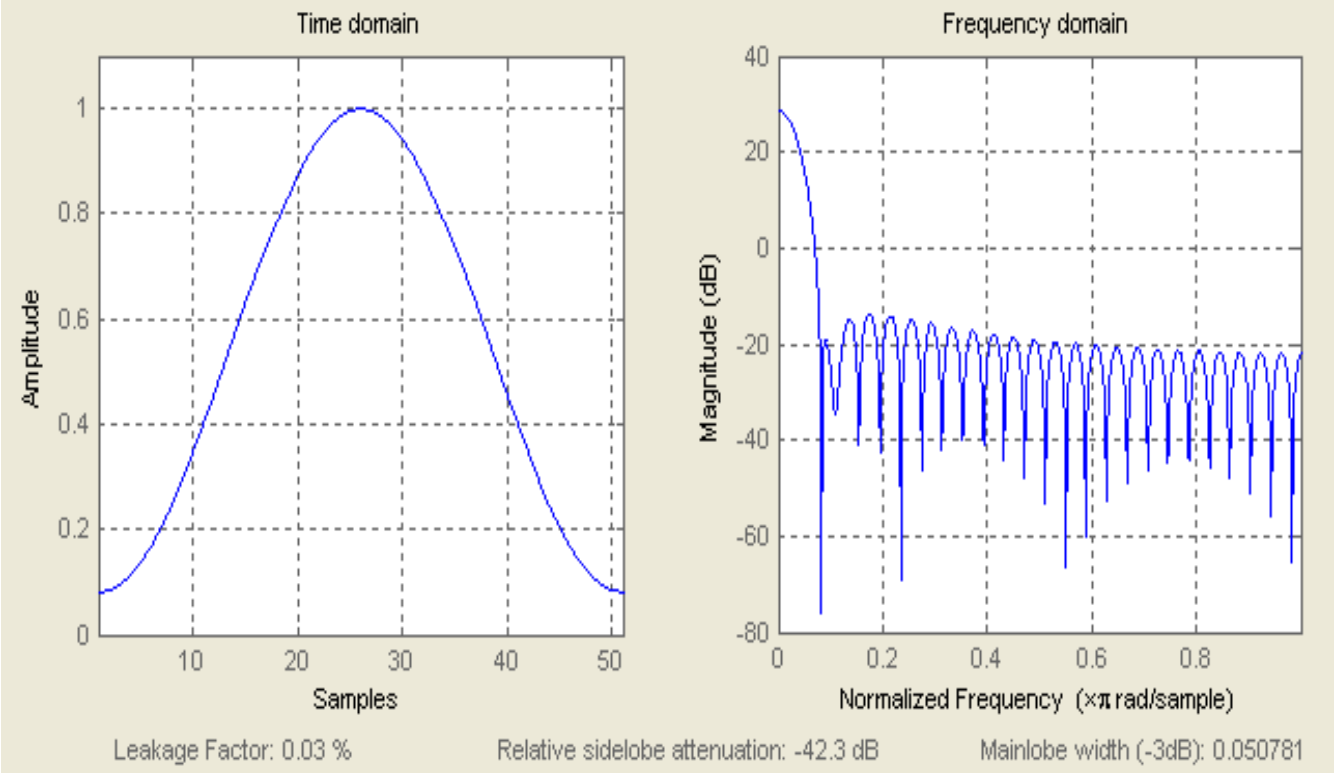


Figure 5.3: Representation of Henning Window in Matlab

5.3.4 BLACKMAN WINDOW

%THE PROGRAM FOR BLACKMAN WINDOW GENERATION HAVING 51 SAMPLES IN MATLAB

N=51;

wvtool(blackman (n));

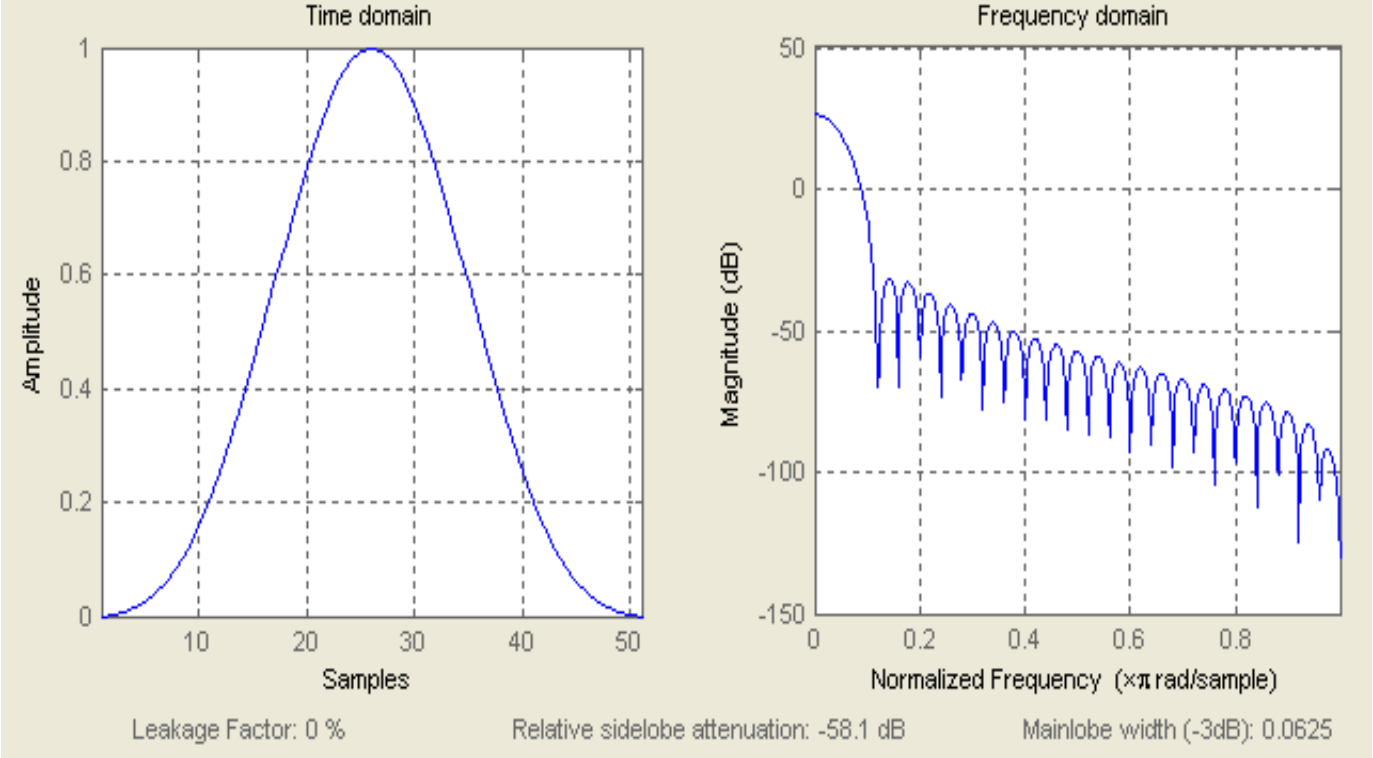


Figure 5.4: Representation of Henning Window In Matlab

5.3.5 HAMMING DISTANCE MINIMIZATION USING G.A.

This coming section presents the results of the hamming distance minimization by using the genetic optimization algorithm as discussed in chapter 4. The algorithms are implemented under the matlab environment on four different FIR filters. These filters varies in terms of desired filter characteristics and consequently in the number of coefficient. These filters have been designed using first using rectangular window function, second using Blackman window function, third hamming window function, and fourth is designed using Henning window function. The coefficient values quantized to 16bit 2's complement representation from the initial set of coefficient of optimization. The hamming distance minimization problem using G.A. is formulated as multi objective minimization problem which minimizes the mean square error and hamming distance simultaneously between the ideal and desired fir filters. To generate the non inferior solution, weighting method is used which convert the multi objective problem in to single objective problem by assigning the appropriate weight to both the objectives. The solution of the scalar objective problem is achieved by G.A. the non inferior solution is generated by making the tradeoff between the two objectives. The procedure is repeated for various simulated weight combinations to get number of solutions.

5.4 FILTER RESPONSE COMPARISION BY GRAPHS

5.4.1 CASE 1: NORMAL RESPONSE

FILTER INFORMATIONS ARE GIVEN BELOW

Window type	Rectangular
Sampling frequency	16 kHz
Cutoff frequency	4 kHz
Filter order	30
No of coefficients	29
Toggling among coefficients	61
Total hamming distance	241

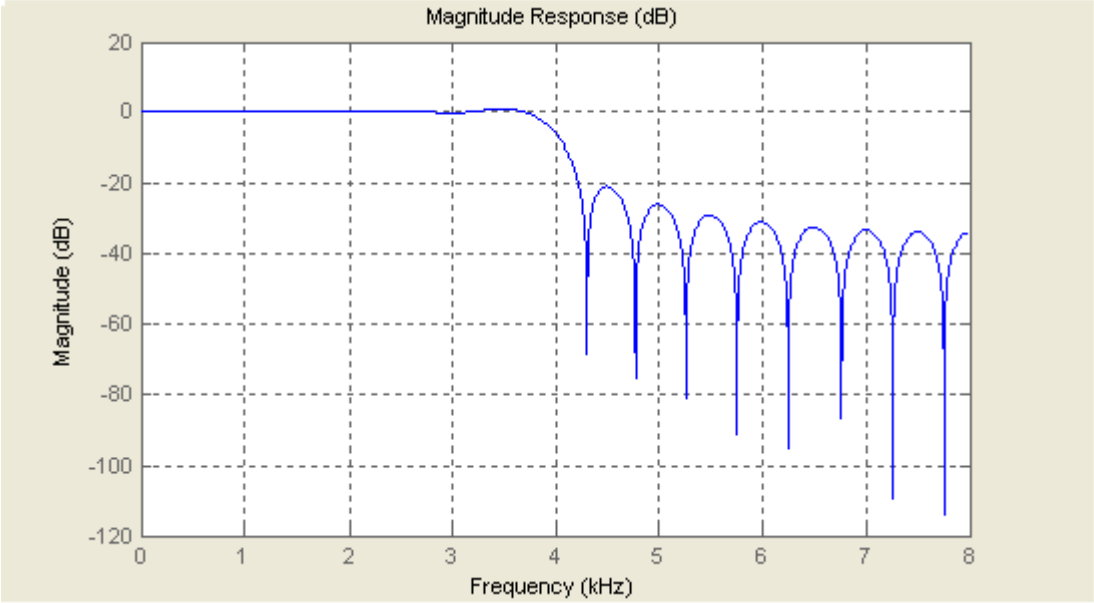


Figure 5.5: Low Pass Filter Normal Response

5.4.2 CASE 1: RESPONSE THROUGH G.A.

FILTER INFORMATIONS ARE GIVEN BELOW

Window type	Rectangular
Sampling frequency	16 kHz
Cutoff frequency	4 kHz
Filter order	30
Number of coefficients	29
Toggling among coefficients	31
Total hamming distance	160

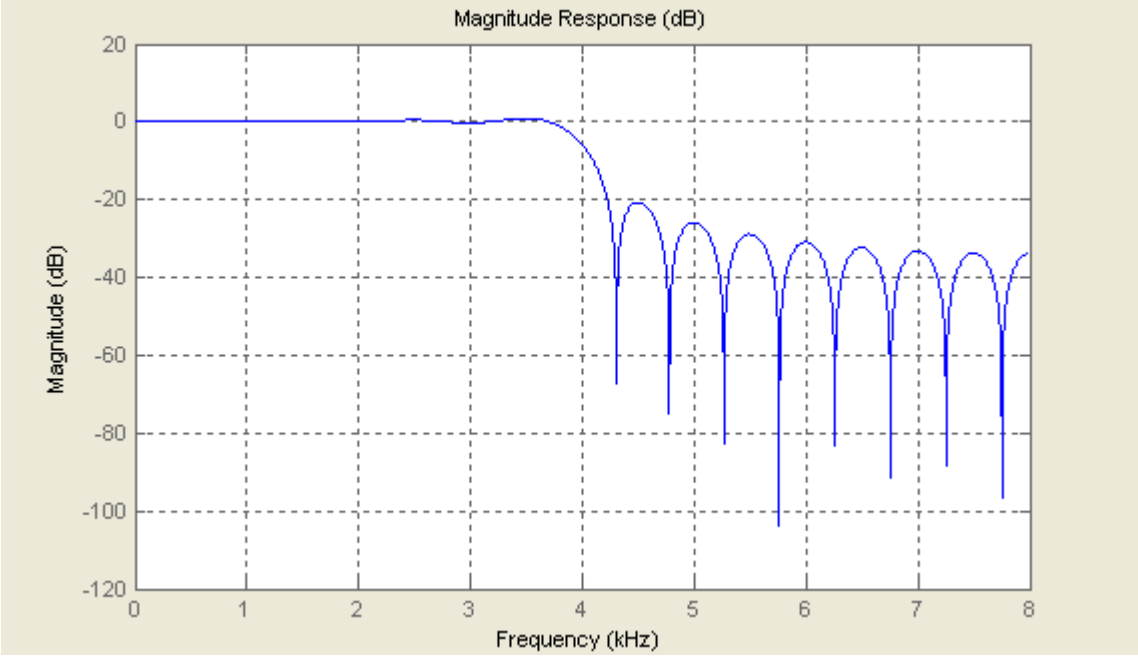


Figure 5.6: Response with G.A.

5.4.3 CASE 2: NORMAL RESPONSE

FILTER INFORMATIONS ARE GIVEN BELOW

Window type	Blackman
Sampling frequency	10 kHz
Cutoff frequency	2 kHz
Filter order	25
Number of coefficients	24
Toggling among coefficients	46
Total hamming distance	221

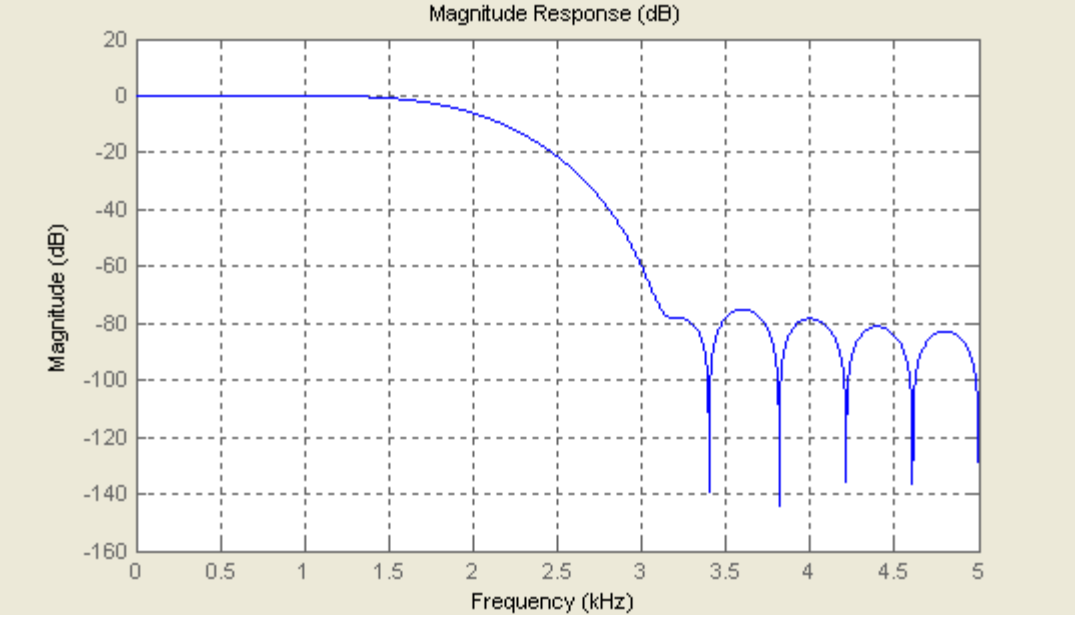


Figure 5.7: Low Pass Filter Normal Response

5.4.4 CASE 2: RESPONSE THROUGH G.A.

FILTER INFORMATIONS ARE GIVEN BELOW

Window type	Blackman
Sampling frequency	10 kHz
Cutoff frequency	2 kHz
Filter order	25
Number of coefficients	24
Toggling among coefficients	22
Total hamming distance	149

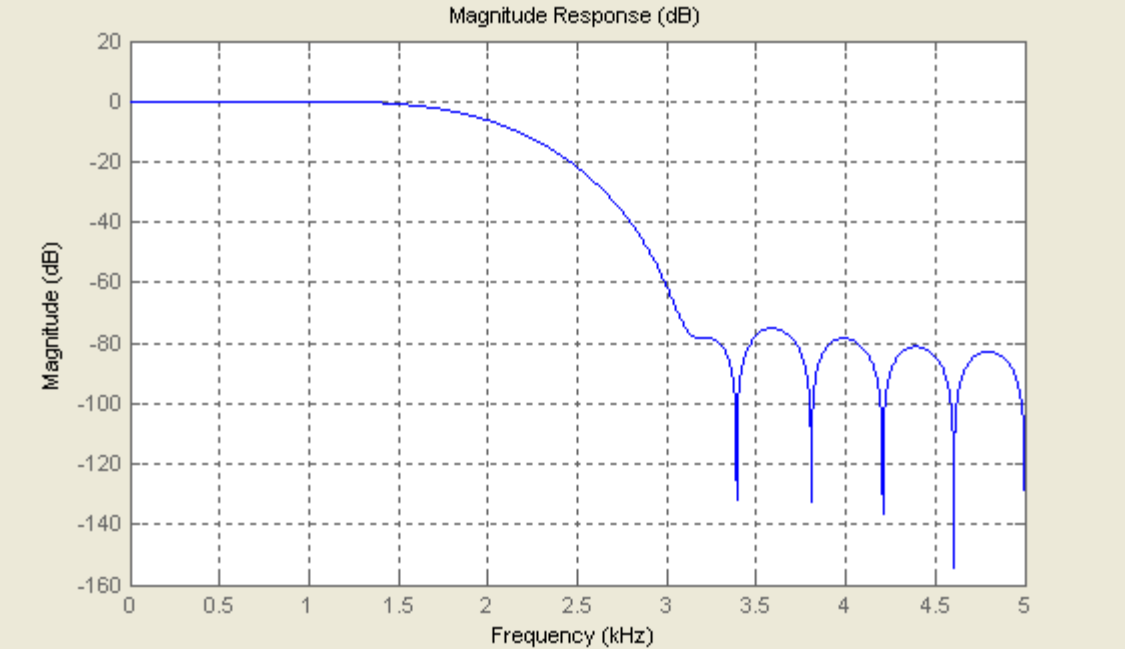


Figure 5.8: Response with G.A.

5.4.5 CASE 3: NORMAL RESPONSE

FILTER INFORMATIONS ARE GIVEN BELOW

Window type	Hamming
Sampling frequency	13 kHz
Cutoff frequency	1.5 kHz
Filter order	25
Number of coefficients	24
Toggling among coefficients	47
Total hamming distance	228

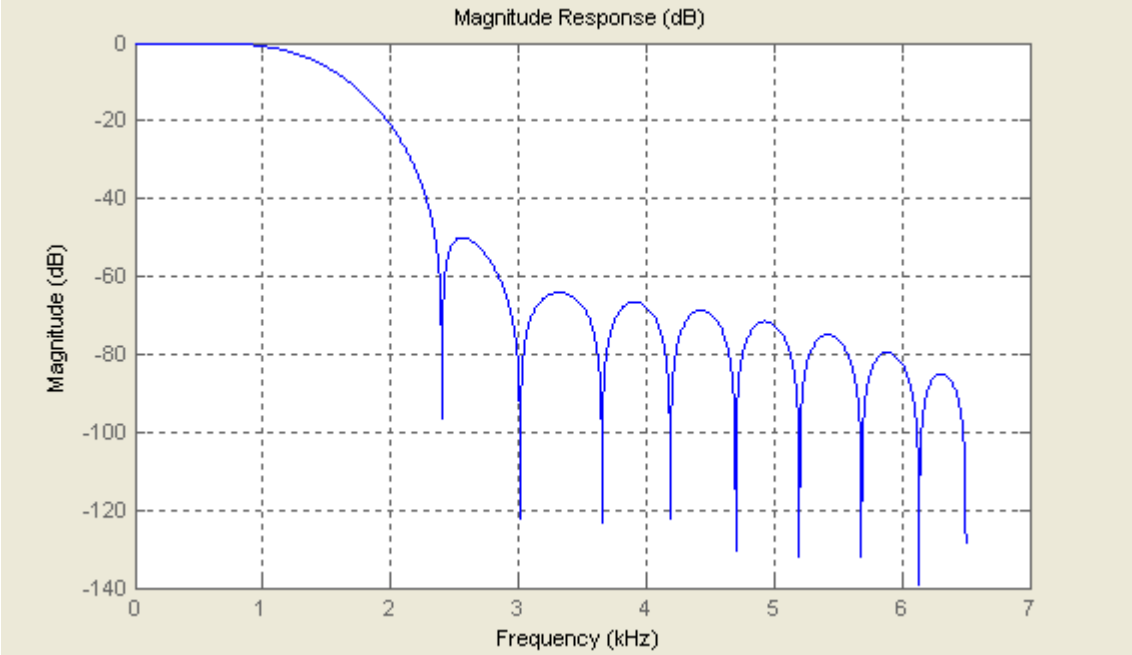


Figure 5.9: Low Pass Filter Normal Response

5.4.6 CASE 3: RESPONSE THROUGH G.A.

FILTER INFORMATIONS ARE GIVEN BELOW

Window type	Hamming
Sampling frequency	13 kHz
Cutoff frequency	1.5 kHz
Filter order	25
Number of coefficients	24
Toggling among coefficients	23
Total hamming distance	139

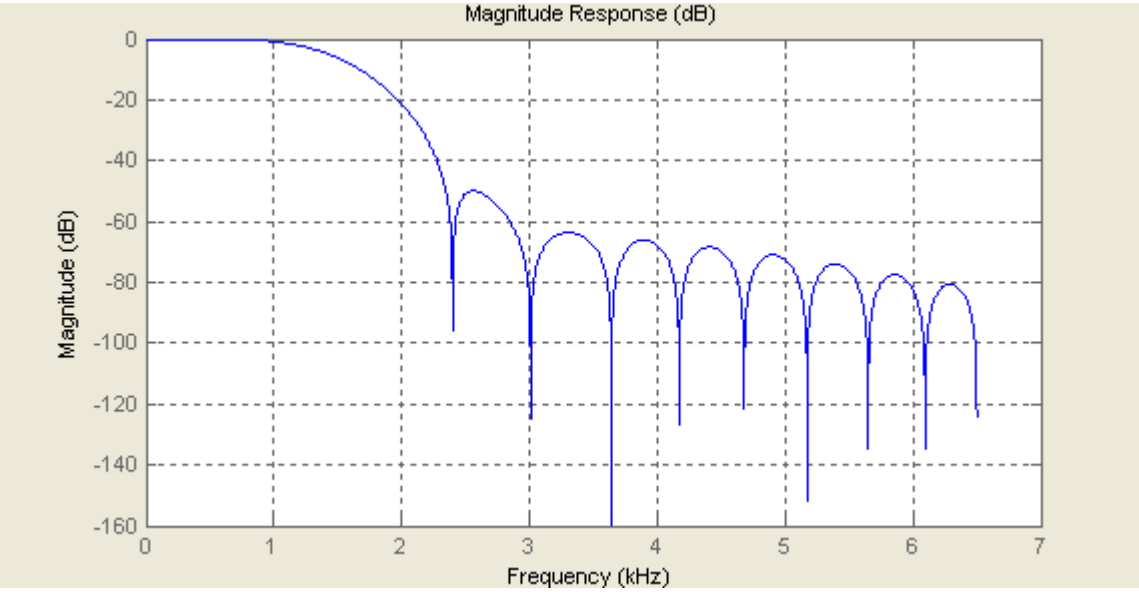


Figure 5.10: Response with G.A.

5.4.7 CASE4: NORMAL RESPONSE

FILTER INFORMATIONS ARE GIVEN BELOW

Window type	Henning
Sampling frequency	18 kHz
Cutoff frequency	4 kHz
Filter order	27
Number of coefficients	26
Toggling among coefficients	53
Total hamming distance	238

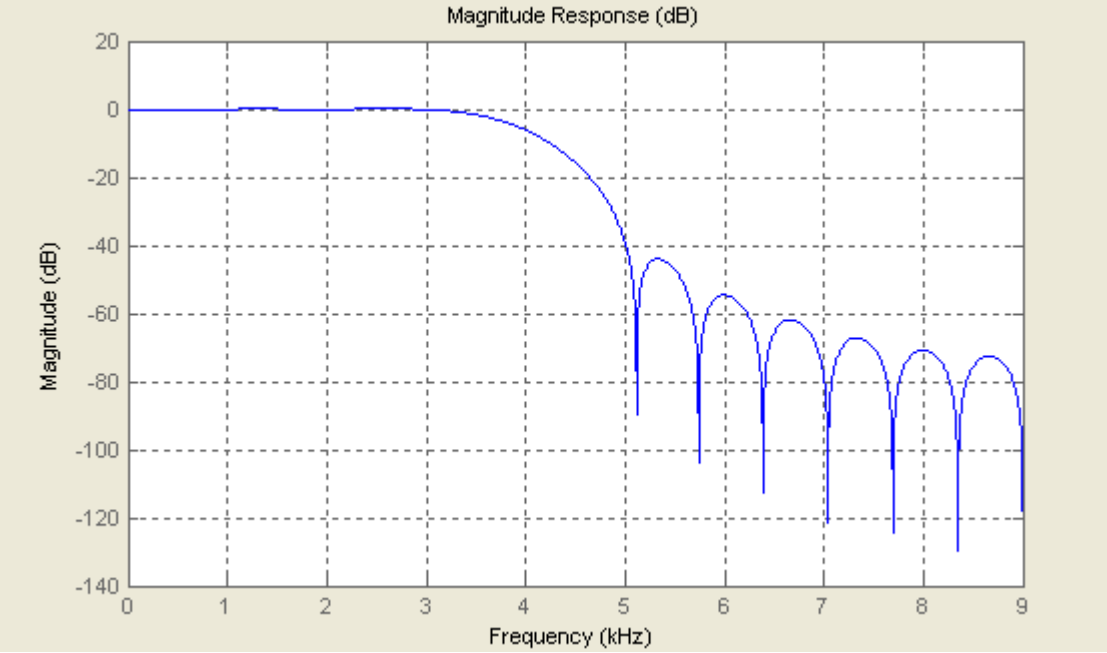


Figure 5.11: Low Pass Filter Normal Response

5.4.8 CASE 4: RESPONSE THROUGH G.A.

FILTER INFORMATIONS ARE GIVEN BELOW

Window type	Henning
Sampling frequency	18 kHz
Cutoff frequency	4 kHz
Filter order	27
Number of coefficients	26
Toggling among coefficients	26
Total hamming distance	151

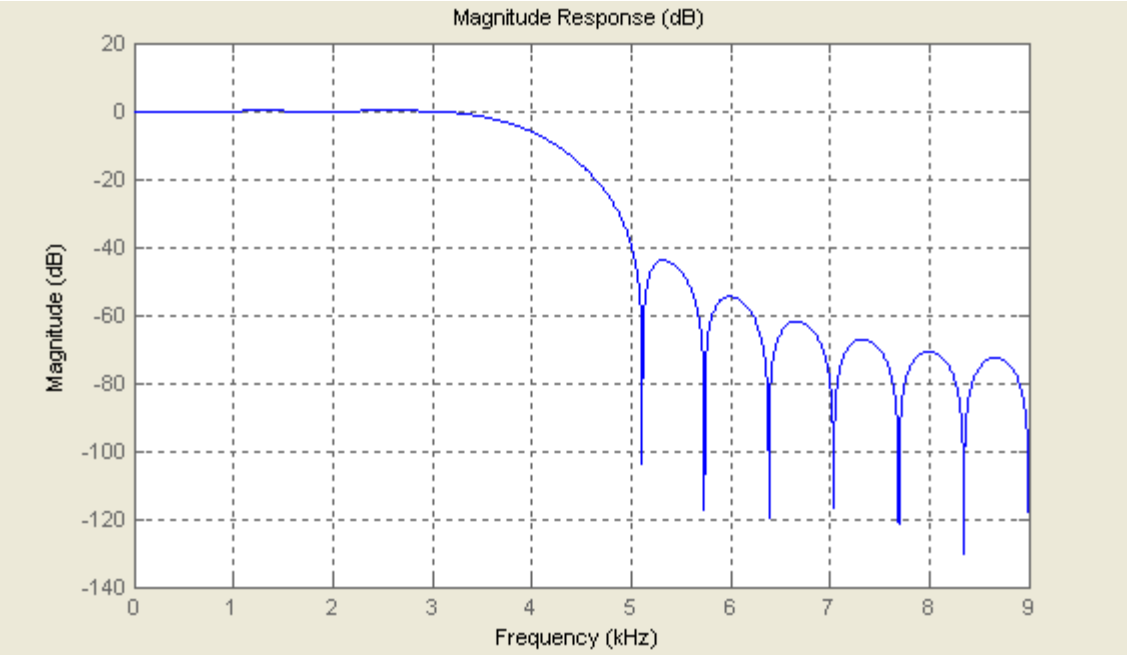


Figure 5.12: Low Pass Filter G.A. Response

CHAPTER 6

RESULTS AND DISCUSSION

This chapter presents the result of hamming distance minimization by using genetic optimization algorithm. As discussed in previous chapter. The algorithms are implemented under MATLAB environment on four different FIR filters. These filters vary in terms of the desired filter characteristic and consequently in the number of coefficients. These filters have been design using the window method. The filters are designed using different window for every filter, first filter is designed by using Rectangular window function, second using Blackman window function, third and fourth are designed using Hamming and Henning window function respectively. The coefficient values quantized to 16 bit 2's complement represent from the initial set of coefficients for optimization the results are presented using genetic algorithm.

The hamming distance minimization problem using genetic algorithm I formulated as multi objective minimization problem which minimize the mean square error and hamming distance simultaneously between the ideal and desired FIR filters. To generate the non inferior solution, weighting method is used which convert multi objective problem in to single objective problem by assigning the appropriate weight to both the objectives. The solution of the scalar objective problem is achieved by genetic algorithm. The non inferior solutions are generated by making the tradeoff between two objectives. The procedure is repeated for various simulated weight combination to get number of solutions. The best solution is selected on the basis of best fitness and hamming distance reduction or corresponds to the minimum square error and maximum hamming distance reduction. Genetic algorithm is applied to the four FIR filters with same specification.

The results of four FIR filters in terms of percentage hamming distance and number of signal toggling reduction genetic algorithm are summarized in table given below

6.1 COMPARATIVE ANALYSIS TABLES FOR THE RESULTS

TABLE-6.1: INITIAL TOGGLING VS FINAL TOGGLING FOR VARIOUS WINDOWS

WINDOWS	INITIAL TOGGLING	FINAL TOGGLING
RECTANGULAR	61	31
BLACKMAN	46	22
HAMMING	47	23
HENNING	53	26

TABLE-6.2: INITIAL VS FINAL HAMMING DISTANCE FOR VARIOUS WINDOWS

WINDOWS	INITIAL HAMMING DISTANCE	FINAL HAMMING DISTANCE
RECTANGULAR	241	160
BLACKMAN	221	149
HAMMING	228	139
HENNING	53	151

TABLE- 6.3: COMPARATIVE ADVANTAGE TABLE OF G.A. APPLICATION

WINDOWS	FIR FILTERS	NORMAL	GENETIC ALGORITHM	PERCENT REDUCTION
		H.D. TOGS	H.D. TOGS	H.D. TOGS
RECTANGULAR	LP_16K_4K_30_29	241 61	160 31	33.6% 49.2%
BLACKMAN	LP_10K_2K_25_24	221 46	149 22	32.5% 52.17%
HAMMING	LP_13K_1.5K_25_24	228 47	139 23	39.0% 51.06%
HANNING	LP_18K_4K_27_26	238 53	151 26	36.6% 50.9%

The results shows that with linear phase constant upto 39% reduction in total hamming distance and upto 51% reduction in adjacent signal toggling in opposite direction is achieved using genetic algorithm.

In terms of optimization strategies the genetic algorithm approach performs best in most of cases. Genetic algorithm searches the whole optimization in space to find a global value so the better hamming distance reduction is achieved with this technique.

CONCLUSION AND FUTURE SCOPE

In this present work hamming distance minimization algorithms are presented for the low power realization of FIR filters on programmable DSPs analysis is presented to show that the hamming distance and the total number of adjacent signals toggling between successive values forms the main measure of power dissipation.

Optimization Algorithms is presented in detail so as to minimize the hamming distance and signal toggling. One such technique 'Genetic Algorithm' is presented to minimize these measures of power dissipation. As astonishing and counterintuitive as it may seem to some, genetic algorithms have proven to be an enormously powerful and successful problem-solving strategy, dramatically demonstrating the power of evolutionary principles. Genetic algorithms have been used in a wide variety of fields to evolve solutions to problems as difficult as or more difficult than those faced by human designers. Moreover, the solutions they come up with are often more efficient, more elegant, or more complex than anything comparable a human engineer would produce. In some cases, genetic algorithms have come up with solutions that baffle the programmers who wrote the algorithms in the first place!

The hamming distance minimization results for four low pass FIR filter shows that the total Hamming distance can be reduced upto 39% and total no of signal toggles can be reduced upto 52% by Genetic Algorithms .This hamming distance reduction directly translates into power saving in multiplier while implementing FIR filter on digital signal processors.

Hamming distance is a common measure of power dissipation of data buses. The minimization of hamming distance and number of signal toggle using Fuzzy Logic and Artificial Neural Networks (ANN) with Genetic Algorithm is expected to receive more intention on the future. It is foreseen that more hybrid system will be launched for signal processing applications and GAs are an important component of these development.

REFERENCES

[1] Dusan Kodek and Kenneth Steiglitz, "*Comparison of Optimal and Local Search methods for Designing Finite word length FIR Digital Filters*", IEEE transactions circuits and systems vol. CAS-28, no.1, January 19812.

[2] Michel. R. Lightner and Stephen. W. Director, "*Multicriterion Optimization for the Design of Electronic Circuits*", IEEE Transactions on Circuits and Systems, Vol. CAS-28, No. 3, March 1981. pp. 169-179.

[3] Harry J. M. Veendrick, "*Short Circuit Dissipation of Static CMOS Circuitry and Its Impact on the Design of Buffer Circuits*", IEEE Journal of Solid State Circuits, Vol. SC-19, No. 4, August 1984. pp. 468-473.

[4] Edward A. Lee, "*Programmable DSP architecture: Part II*", IEEE ASSP Magazine, Vol. 6, Issue 1, January, 1989. pp. 4-14.

[5] Christakis Charalambous, "*A New Approach to Multicriterion Optimization Problem and Its application to the Design of 1-D Digital Filters*", IEEE Transactions on Circuits and Systems, Vol. 36, No. 6, June 1989. pp.773 -784.

[6] Henry Samueli, "*An Improved Search Algorithm for the Design of Multiplier less FIR Filters with power-of-Two Coefficients*", IEEE transactions circuits and system vol. 36, no.7 July 1989. pp. 1044-1047.

[7] D. Suckley, "*Genetic Algorithms in the design of IR filters*", IEE proceeding on Genetic Algorithms, vol. 138, 2, April 1991 .pp. 234-238.

[8] Anantha P. Chandrakasan, Samuel Sheng, and Robert W. Brodersen, "*Low Power CMOS Digital Design"*, IEEE Journal of Solid Stare Circuits, vol. 27, no. 4, January 1992. p. 473-484.

[9] Anantha P. Chardrakasan, Miodrag Potkonjak, Renu Mehra, Jan Rabaey, and Robert W. Brodersen, "*Optimizing Power Using Transformation*", IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, vol. 14, No. 1, January 1995. pp. 12-31.

[10] G. Wade, A. Roberts and G. Williams, "*Multiplier-less Filter Design using a Genetic Algorithm*", IEE proceedings on Vis. Image Signal processing, Vol. 44, No.3, June 1994. pp 175.

[11] Anantha P. Chardrakasan, Miodrag Potkonjak, Renu Mehra, Jan Rabaey, and Robert W. Brodersen, "*Optimizing Power Using Transformation*", IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, vol. 14, No. 1, January 1995. pp. 12-31.

[12] Darren N. Pearson, Keshab K. Parhi, "*Low Power digital Filter Architectures*", IEEE Symposium on Circuit and Systems, vol. 1, 28 April to 31 May 1995. pp. 231-234.

[13] Chetana Nagendra, Robert Michael, Owens Mary and Jane Irwin, "Low Power Consideration in the design of Pipelined FIR filters", IEEE symposium on Low power Electronics, October 9-11, 1995. pp. 32-33.

[14] Mahesh Mahendale , S.D. Sherlekar and G. Ventakesh, "Algorithms and Architectural Transformations for Low Power Realization of FIR Filters ", IEEE 11th Conference on VLSI Design, January 4-71996. pp 12-17.

[15] N. Sankarayya, Kuushik Roy, Purdue U, W Lafayette and Debashis Bhattacharya, "*Algorithms for Low Power Realization of FIR filters using Differential Coefficients*", IEEE 10th international conference on VLSI design 15-17 January 1996. pp 370-375.

[16] D.H. Horrocks, T. Arslan and A.T. Erdogan , "*Low Power Design for DSP*: *Methodologies and Techniques*", Elsevier Science Limited, Microelectronics Journal, January 27, 1996. pp. 731-744.

[17] Mahesh Mehendale, S.D. Sherlekar and G. Ventakesh, "*Low Power Realization of FIR Filters Using MultiRate Architectures*", IEEE 9th conference on VLSI Design July 1996. pp 370-375.

[18] A.T. Erdogan and T. Arslan, "*Low Power Multiplication Scheme for FIR Filter Implementation on Single Multiplier CMOS DSP Processors*", IEEE Electronics Letters vol. 32, No. 21 October. 10, 1996. pp 123-125.

[19] R.Hezar and V.K. Madisetti, ""*Low Power Digital Filter Implementation Using ternary Coefficients*", IEEE Workshop on VLSI Signal Processing, Oct. 30, Nov. 1 1996. pp. 179-188.

[20] P.K. Merakos, K. Masselos, O. Koufopavlou, S. Nikolaidis and C.E. Goutis, "*A Novel Transformation for Reduction of Switching Activity in FIR Filters Implementation*", IEEE 13th International Conference on Digital Signal Processing, July. 2-4 1997. pp.653-656.

[21] Chris J. Nicol and Patrik Larrson, "*Low Power Multiplication for FIR Filters*", IEEE international Symposium on Low Power Electronics and Design, Aug. 18-20, 1997. pp. 76-79.

[22] M.S. Bright and T. Arslan, "*A Genetic Algorithm for the High Level Synthesis of DSP Systems for low Power*", IEE/IEEE Conference on Genetic Algorithms in Engineering Systems: Innovation and Applications, September. 2-4 1997. pp. 174-179.

[23] S.D.Sherlekar, G. Venkatesh and Mahesh Mehendale, "Low Power Realization of FIR Filters on Programmable DSP's", IEEE transaction on VLSI Systems, Vol. 6, No. 4, January 1998. pp. 546-553.

[24] M.S. Bright and T. Arslan, "*Data Block processing for Low Power implementation of Direct Form FIR Filters on Single Multiplier CMOS DSPs*", IEEE International Symposium on Circuits and Systems, Vol. 5, May 31-June3, 1998.pp. 425-428.

[25] A. T. Erdogen and T. Arslan, "*Low Power Coefficient Segmentation Algorithm for FIR Filters Implementation*", IEEE 31st Electronics letters, Vol. 34, Issue 19, Sept. 17, 1998. pp. 1817-1819.

[26] Edward A. Lee, *"Programmable DSP architecture: Part I*", IEEE ASSP Magazine, Vol. 5, Issue 4, October, 1998. pp. 4-19.

[27] Wiliam L. Freking and Keshab Parhi, "*Low Power Residue Digital Filter using Residue Arithmetic*", IEEE 31st Asilomar Conference on Signal, Systems and Computers Vol.1 Nov.2-5, 1998.pp. 739-743

[28] A. Lee, M. Ahmadi. G.A. Jullien, W.C. Miller, and R.S. Lashkari, "*Digital Filter Design using Genetic Algorithm*", IEEE Proceedings of Symposium Advances in digital Filtering and Signal Processing, Victoria B.C, June 5, 1999. pp. 181-184.

[29] Mehmet Oner, "*A Genetic Algorithm for Optimization of Linear Phase FIR Filter Coefficients*", IEEE 32nd Conference on Signal, Systems and Computers, Alisomar, Vol. 2, November 1-4, 1999 .pp. 1392-4000.

[30] Vijay Sundararajan and Keshub K. Parhi, "*Synthesis of Low Power Folded Programmable Coefficient FIR Digital Filters*", IEEE Asia and South Pacific Design Automation Conference, January, 25-28, 2000 .pp. 153-156.

[31] Tian Sheuan Chang, Yuan-Hua Chu, and Chien Wei Jen, "*Low-Power FIR Filter Realization with Differential Coefficients and Inputs*". IEEE Transactions on circuits and systems-II; Analog and digital signal processing, Vol. 47, No.2, February 2000. pp. 137-145.

[32] A. T. Erdogen and T. Arslan, "High Throughput FIR Filter Design for Low Power SOC Application", IEEE 31st Annual International Conference, Sept. 13-16, 2000. pp. 374-378.

[33] M.S. Bright and T. Arslan, "*A Genetic Algorithm for the High Level Synthesis of DSP Systems for low Power*", IEE/IEEE Conference on Genetic Algorithms in Engineering Systems: Innovation and Applications, September. 2-4 1997. pp. 174-179.

[34] Zhan Yu, Meng Lin Yu, Kamran Azadet and Alan N. Willson, "*A Low Power FIR Filter Design Technique using Dynamic Reduced Signal Representation*", IEEE International Symposium on VLSI Technology, Systems and applications, April 18-20, 2001.pp. 113-116.

[35] Chang Young Han, Hyoung Joon Park, and Lee Sup Kim, "*A Low Power Array Multiplier using Separated Multiplication Technique*", IEEE Transactions on circuits and system-II: Analog and digital signal processing, Vol. 48, No. 9, September 2001. pp. 866-871

[36] Keshab K. Parhi, "*Approaches to Low-Power Implementations of DSP Systems*", IEEE Transactions on circuit and system-I, fundamental theory and application, Vol. 48, no.10, October 2001 .pp. 1214-1224.

[37] Alberto Garcia, Lukusa d. Kabulepa and Manfred Glasner, "*Efficient Estimation of Signal Transition Activity in Mac Architectures*", ACM's ISLPED 2002 Monterery, california, USA, August 12-14, 2002. pp. 319-322.

[38] A. T. Erdogan, M. Hasan and T. Arslan, "*Algorithmic Low Power FIR Cores*", IEEE Proceeding of Circuits, system and Devices, Vol. 150, No. 3, June 2003. pp. 23-27.

[39] Jongsun Park, Woopyo Jeong, Hunsoo Choo, Hamid Mahmoodi Meimand, Yongtao Wang, Kauhik Roy, "*Computation sharing Programmable FIR Filter for Low Power and High Performance Applications*", IEEE Journal of Solid State Circuits, Vol. 39, No. 2, February 2004. pp. 348-357.

[40] Youngbeom Jang and Sejung Yang, "*Low Power CSD Linear Phase FIR Filter Structure using Vertical Common Sub-expression*", IEEE electronic letters online, Vol. 38, No. 15, July 18, 2004. pp. 777-779.

[41] Kyung Saeng Kim and Kwyro Lee, *"Low Power and Area Efficient FIR Filter Implementation Suitable for Multiple Taps*", IEEE Transactions on VLSI Systems, Vol. 11, No. 1, February 2005. pp. 323-3325.

[42] Yong Lian and Ling Cen, "*A Genetic algorithm for the Design of Low Power High Speed FIR Filters*", IEEE 7th International Symposium on Signal Processing and Application, Vol. 1, July 1-4, 2005. pp. 181-184.

[43] Dragan Cvetkovic and Ian C. Parmee, "*Preferences and Their application in Multiobjective Optimization*", IEEE Transactions on Evolutionary Computation, Vol. 6, No. 1, Februrary 2006. pp. 42-57.

[44] Alok A. Katkar and James E. Stine, "*Modified Booth truncated Multiplier*", ACM, GSVLSI' 04, Boston, Massachusetts, USA, April 26-28, 2006. pp. 444-447.

URLs:

[1] http://www.mathworks.com/matlabcentral

[2] http://www.google.com

[3] http://www.ieee.org

[4] http://www.iiee.edu.pk

ANNEXURE 1

MATLAB CODES

1.1 MATLAB CODE FOR COMPLETE FILTER DESIGN

```
clc
clear
%-------------------------------------------------------------------------------------------------------------
fs=input('Enter the Sampling Frequency in Hertz fs = ');
tw=input(' Enter the Transition Width in Hertz (less than .5fs) tw = ');
if tw > max(.5*fs)
        disp(' Invalid Entry ')
        break
end
pbr=input(' Enter the Pass Band Ripple in dB pbr = ');
sba=input(' Enter the Stop Band Attenuation in dB sba = ');
%-------------------------------------------------------------------------------------------------------------
y=pbr/20;
dp=(10^y)-1;
y1=-sba/20;
ds=10^y1;
d=min(dp,ds);
df=tw/fs;
A=-20*log10(d);
```

```
%-------------------------------------------------------------------------------------------------------
if A < 22
        disp(' One can use any Window ')
        disp(' What Window do you want to use ')
        ch=input(' 1:Rectangular, 2:Hanning, 3:Hamming, 4:Blackman, 5:Kaiser ::');
elseif ((22 <= A) & (A < 45))
         disp(' One can use any Window except Rectangular Window ')
        disp(' What Window do you want to use ')
        ch=input(' 2:Hanning, 3:Hamming, 4:Blackman, 5:Kaiser ::');
elseif ((45 <= A) & (A < 54))
         disp(' One can use any Window except Rectangular Window and Hanning
         Window ')
         disp(' What Window do you want to use ')
        ch=input(' 3:Hamming, 4:Blackman, 5:Kaiser ::');
elseif ((54 <= A) & (A < 75))
         disp(' One can use any Window except Rectangular Window, Hanning
         Window and Hamming Window ')
         disp(' What Window do you want to use ')
         ch=input(' 4:Blackman, 5:Kaiser ::');
else
         disp(' Only Kaiser Window can be use ')
         disp(' What Window do you want to use ')
        ch=input(' 5:Kaiser ::');
```

```
end
%-------------------------------------------------------------------------------------------------------------
switch ch
case 1
M=round(.9/df);
if rem(M,2)==0
        N=M+1;
else
         N=M;
end
for n=1:(N+1)/2
        Wh(n)=1;
end
m=(N+1)/2;
for n=2:m
        W(m-n+1)=Wh(n);
end
W;
Z=[W Wh];
Z;
N;
%-------------------------------------------------------------------------------------------------------------
```

```
case 2
M=round(3.1/df);
 if rem(M,2)==0
          N=M+1;
else
         N=M;
end
for n=1:(N+1)/2
         Wh(n)=.5+.5*cos(2*pi*(n-1)/N);
end
m=(N+1)/2;
for n=2:m
         W(m-n+1)=Wh(n);
end
W;
Z=[W Wh];
Z;
N;
%-------------------------------------------------------------------------------------------------------
case 3
M=round(3.3/df);
if rem(M,2)==0
         N=M+1;
```

```
else
        N=M;
end
for n=1:(N+1)/2
        Wh(n)=.54+.46*cos(2*pi*(n-1)/N);
end
m=(N+1)/2;
for n=2:m
          W(m-n+1)=Wh(n);
end
W;
Z=[W Wh];
Z;
N;
%-------------------------------------------------------------------------------------------------------------
case 4
M=round(5.5/df);
if rem(M,2)==0
        N=M+1;
 else
        N=M;
 end
for n=1:(N+1)/2
```

```
        Wh(n)=.42+.5*cos(2*pi*(n-1)/(N+1))+.08*cos(4*pi*(n-1)/(N+1));
end
m=(N+1)/2;
for n=2:m
        W(m-n+1)=Wh(n);
end
W;
Z=[W Wh];
Z;
N;
%-------------------------------------------------------------------------------------------------------
case 5
M=round((A-7.95)/(14.36*df))
if rem(M,2)==0
        N=M+1;
else
        N=M;
end
if A <= 21
        b=0;
elseif 21 < A <50
         b=.5842*(A-21)^.4+.07886*(A-21);
else
```

```
        b=.1102*(A-8.7);
end
x=0;
for i=1:24
        x=x+((b/2)^i/factorial(i))^2;
end
Iob=1+x;
for n=1:(N+1)/2
        p=0;
for i=1:24
        p=p+(((b*sqrt(1-(2*(n-1)/(N-1))^2))/2)^i/factorial(i))^2;
end
 Io(n)=1+p;
end
N;
Wh=Io/Iob;
 m=(N+1)/2;
for n=2:m
        W(m-n+1)=Wh(n);
end
W;
Z=[W Wh];
Z;
```

```
%-------------------------------------------------------------------------------------------------------------

otherwise

        disp('Sorry you are unable to design a filter so leave the company immediately')

break

end


%-------------------------------------------------------------------------------------------------------------

end

end

end

end

%-------------------------------------------------------------------------------------------------------------

 disp(' What Filter type do you want to form ')

%-------------------------------------------------------------------------------------------------------------

k=input('please enter 1:Low pass filter,2:High pass filter,3:Band pass filter,4:Band stop
filter');

%-------------------------------------------------------------------------------------------------------------

switch k

%-------------------------------------------------------------------------------------------------------------

case 1

disp(' This is a Low pass filter ')

f=input(' Enter the Pass Band edge Frequency in Hertz f = ');

fp=(f/fs)+(df/2);
```

```
w=2*pi*fp;
for n=1:(N+1)/2
        if n==1
                hd(n)=2*fp;
        else
                 hd(n)=(sin((n-1)*w))/((n-1)*pi);
         end
   end
%-------------------------------------------------------------------------------------------------------------
case 2
disp(' This is a High pass filter ')
f=input(' Enter the Pass Band edge Frequency in Hertz f = ');
fp=(f/fs)+(df/2);
w=2*pi*fp;
for n=1:(N+1)/2
        if n==1
                hd(n)=1-2*fp;
        else
                hd(n)=-(sin((n-1)*w))/((n-1)*pi);
        end
end
%-------------------------------------------------------------------------------------------------------------
case 3
```

```
disp(' This is a Band pass filter ')
f1=input(' Enter the First Pass Band edge Frequency in Hertz f1 = ');
f2=input(' Enter the Second Pass Band edge Frequency in Hertz f2 = ');
fp1=(f1/fs)+(df/2);
fp2=(f2/fs)+(df/2);
w1=2*pi*fp1;
w2=2*pi*fp2;
for n=1:(N+1)/2
        if n==1
                hd(n)=2*(fp2-fp1);
        else
                hd(n)=(sin((n-1)*w2))/((n-1)*pi)-(sin((n-1)*w1))/((n-1)*pi);
        end
end
%-----------------------------------------------------------------------------------------------------------
case 4
disp(' This is a Band stop filter ')
f1=input(' Enter the First Stop Band edge Frequency in Hertz f1 = ');
f2=input(' Enter the Second Stop Band edge Frequency in Hertz f2 = ');
fp1=(f1/fs)+(df/2);
fp2=(f2/fs)+(df/2);
w1=2*pi*fp1;
w2=2*pi*fp2;
```

```
for n=1:(N+1)/2
        if n==1
                hd(n)=1-2*(fp2-fp1);
        else
                 hd(n)=(sin((n-1)*w1))/((n-1)*pi)-(sin((n-1)*w2))/((n-1)*pi);
        end
end
%-------------------------------------------------------------------------------------------------------------
otherwise
disp(' This is not the right choice for filter ')
break
end
%-------------------------------------------------------------------------------------------------------------
 hd;
%-------------------------------------------------------------------------------------------------------------
for n=1:(N+1)/2
        h(n)=hd(n)*Wh(n);
end
%-------------------------------------------------------------------------------------------------------------
m=(N+1)/2;
for n=1:m
        h1(n)=h(m-n+1);
end
```

```
%-------------------------------------------------------------------------------------------------------------------
for n=1:(m-1)
        h2(n)=h(n+1);
end
%-------------------------------------------------------------------------------------------------------------------
h3=[h1 h2];
c=1:fs;
H1=fft(h3,fs);
z=20*log10(abs((H1)/max(H1)));
subplot(4,1,1), plot(h3)
subplot(4,1,2), plot(Z)
subplot(4,1,3), plot(c,abs(H1))
subplot(4,1,4), plot(c,z)
%-------------------------------------------------------------------------------------------------------------------
```

1.2 MATLAB CODE FOR INDIVIDUAL WINDOW PLOTTING

1.2.1 RECTANGULAR WINDOW

```
clear all
wc=0.5*pi;
N=51;
alpha=(N-1)/2;
epa=0.001;
n=1:1:N;
hd=sin(wc*(n-alpha+eps))/(pi*(n-alpha+eps));
wr=boxcar(N);
hn=hd*wr;
w=0:0.1:pi;
h=freqz(hn,1,w);
plot(w/pi,abs(h));
xlabel('NORMALISED FREQUENCY');
ylabel('MGNITUDE');
```

1.2.2 HANNING WINDOW

```
clc
clear all
wc=0.5*pi;
N=25;
alpha=(N-1)/2;
eps=0.001;
n=1:1:N;
hd=sin(wc*(n-alpha+eps))/(pi*(n-alpha+eps));
wr=hann(N);
```

```
hn=hd.*wr';
w=0:0.1:pi;
h=freqz(hn,1,w);
plot(w/pi,abs(h));
xlabel('NORMALISED FREQUENCY');
ylabel('MGNITUDE');
```

1.2.3 HAMMING WINDOW

```
clc
clear all
wc=0.5*pi;
N=25;
alpha=(N-1)/2;
eps=0.001;
n=1:1:N;
hd=sin(wc*(n-alpha+eps))/(pi*(n-alpha+eps));
wr=hamming(N);
hn=hd.*wr';
w=0:0.1:pi;
h=freqz(hn,1,w);
plot(w/pi,abs(h));
xlabel('NORMALISED FREQUENCY');
ylabel('MGNITUDE');
```

ANNEXURE 2

‘C/C++’ CODES

2.1 ‘C’ CODE FOR TOGGELING CALCULATIONS

```
#include<stdio.h>
#include<conio.h>
main()
 {
          float h[36];
          int h1[36][15],m[36][15];
         float c;
         int i,w,m1,p1=0,r,j,k,t,n,p,l=0,q=0,f,g;
          clrscr();
           printf("enter the number of coefficent = ");
           scanf("%d",&w);
           printf("\n enter the value one by one");
           for(p=0;p<w;p++)
           {
                    printf("\n Enter The %dst value = ",p+1);
                     scanf("%f",&h[p]);

            }

           for(i=0;i<w;i++)
           {
                     c=h[i];
                     printf("\n %f",c);
                    for(j=0;j<=14;j++)
                    {
```

```
                    c=c*2.00;
                    k=(int)c;
                    c=c-k;
                    h1[i][j]=k;
                }
        }
        for(f=0;f<w;f++)
        {
                printf("\n\n");
                for(g=0;g<=14;g++)
                {
                        printf(" %d",h1[f][g]);
                }
        }

        for(j=0;j<w;j++)
        {
                m1=h1[j][0];
                for(t=0;t<=14;t++)
                {

                        if(m1!=h1[j][t])
                        {
                                p1=p1+1;
                                m1=h1[j][t];
                        }
                }
        }

printf("\n The Total Toggling = %d",p1);
getch();    }
```

2.2 'C' CODE FOR HAMMING DISTANCE CALCULATIONS

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
 {
        double precision(double d);
         long  dectobinfloat(double p);
        long exor(long l1,long l2);
        double p1,p2,h1,h2,h[100],p[100];
        long i,n,par,l1,l2,l[100],number_of_one[100],sum_of_one=0;;
        clrscr();
        printf("\n Enter the no. of parameters");
        scanf("%ld",&n);
        printf("\n Enter the value of parameters in fraction");
        for(i=0;i<n;i++)
        scanf("%lf",&h[i]);
        printf("\n--------------------------------------------------------------------------------");
        printf("\n The value of parameters are:");
        for(i=0;i<n;i++)
                printf("\t%lf",h[i]);
                printf("\nTruncated  values are following:");
                for(i=0;i<n;i++)
                  p[i]=precision(h[i]);
                  printf("\n--------------------------------------------------------------------------------");
                  printf("\n Binary Equivalent");
                  for(i=0;i<n;i++)
                       l[i]=dectobinfloat(p[i]);
                       printf("\n--------------------------------------------------------------------------------");
                      for(i=0;i<n-1;i++)
                      {
```

```
                printf("\nExor of %lf and %lf is  ",h[i],h[i+1]);
                number_of_one[i]=exor(l[i],l[i+1]);
                sum_of_one=sum_of_one + number_of_one[i] ;
            }
            printf("\nTotal number of ones=%ld",sum_of_one);
            /*printf("\n Enter 2 parameters");
            scanf("%lf%lf",&h1,&h2);
            p1=precision(h1) ;
            l1=dectobinfloat(p1);
            p2=precision(h2);
            l2=dectobinfloat(p2);
            exor(l1,l2);*/
            getch();
}
//Precision upto 5 digits after decimal point
double precision(double d)
 {
        double e,p;
         long c;
        c=d*100000;
        e=d*100000;
        if((e-c)>=0.44)
                c=c+1;
                p=(float)c/100000;
                // printf("\n%lf\t%ld\t%lf\t%lf",d,c,e,p);
                printf("\n%lf",p);
                return p;
 }
//Conversion from decimal point to binary
 long  dectobinfloat(double p)
 {
```

```
                long bintodec(long);
                double r,m,poriginal;
                long dec,ip,s,x,flag=0;
                poriginal=p;
                 s=0;
                x=8;
                while(x!=0)
                {
                        m=p*2;
                         ip=m;
                        s=s+ip*pow(10,x);
                        x--;
                        r=m-ip;
                         p=r;
                }
 printf("\nBinary equivalent of %lf is %ld/10to power 9",poriginal,s);
 if(s<0)
        {
                flag=1;
                s=(111111111-(-s));
        }
 printf("\n%ld", s);
 dec=bintodec(s);
 if (flag==1)
 dec=dec+1;
 flag=0;
 printf("\n%ld",dec);
 return dec;
}

//Binary to decimal
```

```
long bintodec(long bin)
{
          long rem,dec,quo,sum=0,inc=0;
         while(bin!=0)
         {
                  rem=bin%10;
                  bin= bin/10;
                  sum=sum+rem*pow(2,inc);
                   inc++;
         }
 printf("\n%ld",sum);
 return sum;
}
//implementation of ex-or function
 long exor(long firstnum,long secondnum)
 {
          long dectobin(long dec);
         long countofone(long bin);
          long bin,dec;
          long result;
          long cnt_one;
         result=firstnum^secondnum;
         bin=dectobin(result);
         printf("%ld/10 to power 9",bin);
         printf("\n Exor =%ld",result);
         cnt_one=countofone(bin);
         return cnt_one;
}
// decimal to binary conversion
 long dectobin(long dec)
 {
```

```
        long rem,quo,sum=0,inc=0;
        // printf("\nrem dec sum inc ");
         while(dec!=0)
         {
                  rem=dec%2;
                 dec= dec/2;
                 sum=sum+rem*pow(10,inc);
                 inc++;
                  // printf("\n%ld\t%ld\t%ld\t%ld",rem,dec,sum,inc);
         }
 return sum;
}
 // count of ones in exor result
 long countofone(long bin)
 {
              long count=0,remainder;
                 while(bin!=0)
                 {
                          remainder=bin%10;
                          if(remainder==1) count++;
                          bin=bin/10;
                  }
                 printf("\nNumber of ones =%ld",count);
                 return count;
  }
```

2.3 'C++' CODE FOR CROSSOVER CALCULATION

```
//Applying the genetic algo on binary representations
//To get the optimized route
cout<<"\n**The GA
begins**";
cout<<"\nThe number in binary form is :";
for(i=0;i<3;i++)
for(j=0;j<3;j++)
{
        cout<<"\n";
        for(k=0;k<8;k++)
        cout<<costbin[i][j][k];
}
int cr,mut,gen,t,tr,er;
time_t t1;
int cp;
int ones[3][3]={0,0,0,0,0,0,0,0,0};
//asking for the number of cross overs
cout<<"\nPlease enter the number of generations: ";
cin>>gen;
for(int loop=0;loop<gen;loop++)
{
        tempcost[jk][er]=costbin[jk][0][er];
        for(jk=0;jk<3;jk++)
                for(y=crp;y<8;y++)
                        costbin[jk][0][y]=costbin[jk][1][y];
                        for(jk=0;jk<3;jk++)
                        for(y=crp;y<8;y++)
                        costbin[jk][1][y]=tempcost[jk][y];
}
//checking for mutation
```

```
if(mut==1)
{
int mp;
cout<<"\nPlease enter the point of mutation: ";
cin>>mp;
int mutnode[3][3][3];
cout<<"\nPlease enter 1 if the node mutated zonewise and nodewise and
0 if not: ";
//applying mutation on the binary values
for(int qw=0;qw<3;qw++)
{
        for(int we=0;we<3;we++)
        for(er=0;er<3;er++)
        cin>>mutnode[qw][we][er];
        for(t=0;t<3;t++)
        for(tr=0;tr<3;tr++)
        for(er=0;er<3;er++)
        {
                if(mutnode[t][tr][er]==1)
                {
                        if(costbin[t][tr][mp]==1)
                        costbin[t][tr][mp]=0;
                        125
                        else
                        costbin[t][tr][mp]=1;
                }
        }
}
utputting the resultant binary values
cout<<"\nThe cost
utputtin after cross over are: \n";
```

```
for(t=0;t<3;t++)
{
        for(tr=0;tr<3;tr++)
        {
        cout<<"\n";
        for(er=0;er<8;er++)
                cout<<costbin[t][tr][er];
        }
}
//Rearranging the binry values in scendind
//Order of the number of ones each have
for(i=0;i<3;i++)
for(j=0;j<3;j++)
for(k=0;k<8;k++)
{
        if(costbin[i][j][k]==1)
        ones[i][j]++;
}
for(i=0;i<3;i++)
        for(j=0;j<3;j++)
        for(k=0;k<3;k++)
        {
                if(ones[i][j]<ones[i][k])
                {
                int tempo;
                126
                tempo=*costbin[i][k];
                *costbin[i][k]=*costbin[i][j];
                *costbin[i][j]=tempo;
                }
        }
```

```
utputting the optimized values in each node
cout<<"\nThe optimized binary values are:\n ";
for(i=0;i<8;i++)
{
        costbin[0][0][i];
        cout<<"\n";
        for(i=0;i<8;i++)
                cout<<costbin[1][0][i];
                cout<<"\n";
                for(i=0;i<8;i++)
                cout<<costbin[2][0][i];
                int costcon[3];
                for(i=0;i<3;i++)
                costcon[i]=0;
//back conversion to decimal form
for(i=0;i<3;i++)
        for(j=0;j<8;j++)
        costcon[i]=costcon[i]+costbin[i][0][j]*pow(2,j);
        cout<<"\nThe optimum coefficient in decimal form are:";
        for(i=0;i<3;i++)
                cout<<"\n"<<costcon[i];
                int sub[3][3];
                for(i=0;i<3;i++)
                for(j=0;j<3;j++)
                sub[i][j]=abs(costcon[i]-Ab[i][j]);
                int route[3];
                getch();
        }
```

ANNEXURE 3

PUBLICATIONS FROM THIS RESEARCH WORK

(1). Swarajya Agnihotri and Yaduvir Singh "*A Power Efficient Optimization Method for FIR Filter Coefficients using Genetic Algorithm*", in International Conference on Computational Intelligence for Modeling, Control and Automation (CIMCA08) slated for December 10-12-2008 at Vienna, Austria**.** (Communicated)

(2). Swarajya Agnihotri and Yaduvir Singh "*An Adaptive Algorithm for Speed Control of D.C. Motor: a Case Study*", in world congress on engineering (WCE) APRIL 2007, LONDON. The paper was considered for the best paper award in its corresponding workshop.

ANNEXURE 4

BRIEF BIODATA OF THE RESEARCHER

63-Extension Part-A, Sector2
Green Park Colony, Bareilly-243006, India
Email: swarajyaagnihotri@gmail.com
Phone: +91-9411285308,+91-9872390485

EDUCATION

M.E., Electronic Instrumentation and Control, T.I.E.T, Patiala (2006-2008) with C.G.P.A of 8.90 on the scale of 10.

- **Thesis**: Design And Development of A Power Efficient Optimization Code For FIR Filters Using Genetic Algorithms
- **Language**: C/C++, Matlab

B.Tech., Electronics and instrumentation, Institute of engineering and technology, M.J.P. Rohilkhand University, Bareilly, with C.G.P.A of 7.48 on the scale of 10.

- **Project Title:** Digital Temperature Indicator.
- **Advisor**: Dr S.K. Tomar (M.J.P.R.University, Bareilly)
- **Summary**: This system has been designed for converting and monitoring the temperature in degrees Fahrenheit (°F) signal within few second.
- G.I.C. Allahabad, **12th** with 68.6% aggregate.
- G.I.C. Allahabad, **10th** with 68.6% aggregate. .

PROJECT WORK (M. E)

D.N.A. computing: study of D.N.A. computation.

Artificial Intelligence: Detection of acoustic variability (speech recognition) of digits zero to nine using back propagation algorithm in artificial neural network (ANN).

Tonometery: A wide study on the eye pressure measurement, causes and solution methods.

PROJECT WORK (B.Tech.)

Process Controller: Designed a PC based level controller for controlling the water level in a process control loop.

Analog Circuit Design: Built an emergency light.

TRAINING & WORK EXPERIENCE

- Jun-Dec. 1997: software technology and software management curriculum with 'A' grade (91% marks) from NIIT, ALLAHABAD.
- Jun-July 2000: practical 2 months training on "Study of CNC Machines", at B.H.E.L, JHANSI.
- Jul2003-jun 2006: Working experience as service support engineer at RUPTRONICS INDIA LIMITED, AGRA.
- March 2008: one week Awareness program on sun Microsystems.
- April 2008: two days workshop under continuing engineering education programme (CEEP), on intelligent system design & soft computing techniques.

COMPUTER SKILLS

Software: - Introduction to Lab View 6.0 and Keil

Languages: - Introduction to C/C++, Mat Lab 6.0, LISP, and assembly

Platform: - Windows98 / XP/vista

PAPERS PUBLISHED

"World Congress on Engineering (WCE) APRIL 2007. The paper was considered for the best paper award in its corresponding workshop.

Topic: *Adaptive Algorithm for Speed Control of D.C. Motor: a Case Study*

AWARDS

- National scholarship holder.
- 1st rank holder for consecutive last four semester of B.Tech.
- 1st rank holder in Bhavishya Jyoti scholar ship test, conducted by NIIT, Allahabad.

- Best participant award in the annual cultural programme (SRIJAN), in B.Tech.

REFERENCES

- Dr. Yaduvir Singh, Dept of Electrical and Instrumentation, T.U., Patiala
- Mr Mandeep Singh, Dept of Electrical and Instrumentation, T.U., Patiala

Date: 15-JULY-2008 **SWARAJYA AGNIHOTRI**