HIGH LEVEL POWER EFFICIENT SYNTHESIS OF FIR BASED DIGITAL SYSTEMS

by

Mustafa Aktan

B.S. Electrical & Electronics Engineering, Boğaziçi University, 1999

M.S. Electrical & Electronics Engineering, Boğaziçi University, 2001

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Electrical & Electronics Engineering

Boğaziçi University

2008

HIGH LEVEL POWER EFFICIENT SYNTHESIS OF FIR BASED DIGITAL SYSTEMS

APPROVED BY:

Prof. Günhan Dündar (Thesis Supervisor)	.....
Prof. Ayşın B. Ertüzün	.....
Assist. Prof. Mutlu Koca	.....
Prof. Lars Wanhammar	.....
Assoc. Prof. Arda Yurdakul	.....

DATE OF APPROVAL:

ACKNOWLEDGEMENTS

I would like to express my thanks to my thesis supervisor Prof. Günhan Dündar for his invaluable support and help during my thesis. I would also like to thank Assoc. Prof. Arda Yurdakul and Assist. Prof. Mutlu Koca for their encouragement and the constructive critiques they gave during the preparation of this work. Special thanks to the BETA members, Emre Arslan, Cem Çakır, Uğur Çini, Yasin Çitkaya, Selçuk Talay, and Umut Yazkurt for their friendship, help and accompaniment.

I would like to dedicate this study to my family for their patience and ever-lasting love.

This work was supported in part by The Scientific and Technological Research Council of Turkey (TUBITAK) under Project No: 104E139.

ABSTRACT

HIGH-LEVEL POWER EFFICIENT SYNTHESIS OF FIR BASED DIGITAL SYSTEMS

Digital filters are the most frequently used elements in signal processing applications. Among digital filters, FIR filters are preferred due to their stability, easily achievable linear-phase property, and low quantization wordlength sensitivity. All these desirable properties come with a drawback: increased computational workload. This, in turn, leads to excessive amount of power dissipation which is a bottleneck for today's low power demanding applications.

In this work, a low-power design methodology for the design of FIR filters is proposed. The methodology is implemented in a software tool where the user gives only the characteristics of the FIR filter. The tool generates the power optimized circuit/coefficient set depending on the type of realization of the filter: parallel/sequential. For the parallel realization using constant coefficients, power is related to the number of nonzero digits in the binary notation of the filter coefficients. On the other hand, the sequential realization of FIR filters is done on programmable processors where coefficients are successively applied to the inputs of a multiply accumulate unit. Hence, switching activity between successively applied coefficients is important for low power design. In this context, a novel algorithm for the design of low-power and hardware efficient linear-phase FIR filters is proposed which is the main contribution of this work. The algorithm finds filter coefficients with reduced complexity (number of ones in coefficients, switching activity between coefficients) given the filter frequency response characteristics. Although the worst case run time of the algorithm is exponential, its capability to find appreciably good solutions in a reasonable amount of time makes it a desirable CAD tool for designing low-power and hardware efficient FIR filters. The superiority of the algorithm on existing methods in terms of design time, hardware complexity, and power performance is shown with several design examples for both parallel and sequential realizations of FIR filters.

ÖZET

SONLU DÜRTÜ YANITLI SÜZGEÇ TEMELLİ SAYISAL SİSTEMLERİN YÜKSEK SEVİYEDE GÜÇ VERİMLİLİĞİNE GÖRE SENTEZİ

Sayısal süzgeçler işaret işleme uygulamalarında en sık kullanılan devrelerdendir. Sayısal süzgeçlerin bir çeşidi olan sonlu dürtü yanıtlı (SDY) süzgeçler kararlılık, doğrusal faz ve kuantalamaya karşı düşük hassasiyetli olmaları sebebiyle tercih edilmektedirler. Fakat, arzulanan bu özelliklerin yanında hesaplama külfeti artmaktadır. Dolayısıyla fazla güç harcamaktadırlar. Bu ise, az güç sarfiyatı gerektiren günümüz uygulamalarının ihtiyaçlariyla çelişmektedir.

Bu çalışmada, SDY süzgeçlerin az güç harcayacak şekilde tasarımını sağlayan bir metodoloji önerilmektedir. Önerilen metodoloji bir yazılım aracı şeklinde gerçeklenmiştir. Araca, kullanıcı tarafından sadece süzgeç karakteristikleri girilmektedir. Araç, süzgecin paralel veya ardışıl gerçeklenecek olmasına göre güç sarfiyatı optimize edilmiş devreyi veya süzgeç katsayılarını çıktı olarak vermektedir. Paralel gerçeklemede harcanan güç süzgeç katsayılarının ikili düzen gösteriminde içerdiği ‘bir’ sayısıyla ilintilidir. Öte yandan, ardışıl gerçekleme programlanabilir işlemciler üzerinde olmaktadır. Burada katsayılar ardışıl olarak işlemcinin çarpıcı toplayıcı ünitesine girilmektedir. Güç sarfiyatı birbiri ardısıra girilen katsayılar arasındaki anahtarlama sıklığına bağlı olarak artmaktadır. Bu çalışmanın ana katkısı, az güç harcayan ve az donanım gerektiren SDY süzgeç tasarımına uygun yeni bir süzgeç tasarım algoritmasıdır. Bu algoritmayla, daha az güç sarfederekten istenen süzgeç frekans karakteristiğini sağlayan süzgeç katsayıları bulunmaktadır. Önerilen algoritmanın, en kötü şart koşma süresi üstel olarak artmaktadır. Fakat, gayet kısa bir zaman içinde iyi sonuçlar bulabiliyor olması bir bilgisayar destekli tasarım aleti olarak kullanılmasına olanak saglamaktadır. Algoritmanın şimdiye kadar önerilmiş yöntemler olan üstünlüğü değişik açılardan (tasarım zamanı, güç sarfiyatı, donanım karmaşıklığı) çeşitli örnek süzgeçler tasarlanarak doğrulanmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

$A(\omega)$	Amplitude of the frequency response of a FIR filter
B	Coefficient quantization wordlength
C	Capacitance
$D(\omega)$	Desired frequency response of a FIR filter
f	System clock frequency
$\mathbf{H}$	Channel convolution matrix
$h[i]$	FIR filter coefficients
h_i	Channel filter coefficients
$H(\omega)$	Frequency response of a FIR filter
J	Mean square error of an equalizer
$J_{\max}$	Maximum allowed mean square error for an equalizer
M	Half the number of coefficients of a FIR filter
N	Number of coefficients of a FIR filter
V_{dd}	Supply voltage
V_i	Feasible value set of coefficient $h[i]$ and/or w_i
V_i^s	Refined feasible value set of coefficient $h[i]$ and/or w_i
$\mathbf{w}$	Equalizer coefficient vector
w_i	Equalizer coefficient
α	Switching activity factor
$\delta(\omega)$	Frequency response approximation error for a FIR filter
Ω	Frequency band for which a desired response is defined
Ω^t	Transition band
ASIC	Application Specific Integrated Circuit
BER	Bit Error Rate
CAD	Computer Aided Design
CMOS	Complementary Metal Oxide Semiconductor
CSD	Canonic Signed Digit

DSP	Digital Signal Processor
FIR	Finite Impulse Response
IC	Integrated Circuit
IIR	Infinite Impulse Response
ILP	Integer Linear Programming
LP	Linear Programming
MAC	Multiply Accumulate
MILP	Mixed Integer Linear Programming
MOS	Metal-Oxide Semiconductor
MSE	Mean Square Error
MMSE	Minimum Mean Square Error
NMOS	N-type Metal-Oxide Semiconductor
NPR	Normalized Peak Ripple
PMOS	P-type Metal-Oxide Semiconductor
SPT	Signed Power of Two
TSPC	True Single-Phase Clocked
VLSI	Very Large Scale Integration

1. INTRODUCTION

Low-power very large scale integrated (VLSI) circuit design has an important role in the electronics industry today. In the past, most research and development efforts focused on increasing the processing speed and reducing the complexity of the chip design. However, it is interesting to note that the main cause of switching between technologies such as from the vacuum tube to the bipolar, and there on to the metal-oxide semiconductor (MOS), and yet today to complementary MOS (CMOS) technology is power [1]. The power consumption of the chip, on the other hand, was given lower priority during the design phase until the '90s. The scenario has been changed since the advent of personal communications/computing devices. One reason is owing to the growing markets in portable computing and communication systems. The common feature of these devices is that they demand high-speed data/signal processing, but are constrained to work with a limited capacity power supply, i.e. a battery.

Battery-limited devices are not the only driving forces for low-power design. For the microprocessor manufacturer, an increased packaging cost due to excessive amount of heating of the processor caused by high power dissipation is the main bottleneck. For military applications reliability is the main concern, and high power consumption reduces reliability in an integrated circuit (IC) by causing electron migration failures and bouncing in supply rails due to excessive amount of driven current.

Having many driving forces, low-power devices are becoming more and more important. To meet industries needs, developing power optimization tools is an important issue. By using such tools, VLSI systems can be designed at various levels of abstraction (software, system, algorithm, architecture, circuit, logic, device, and technology) considering power.

1.1. Power in CMOS Circuits

CMOS circuits are preferred due to their inherently low power consumption which can be attributed to the elimination of static power consumption. That is, the p-type metal-

oxide semiconductor (PMOS) and N-type metal-oxide semiconductor (NMOS) devices in a CMOS circuit are never ON simultaneously when either at logic 0 or 1. Since there is no dc path between the supply and ground there will not be any static current flowing. The static power consumed in CMOS circuits is mainly due to leakage currents flowing through reverse biased junctions. As more and more transistors are built on ICs, leakage power will constitute an important amount of the total power consumption [1].

Dynamic power can still be attributed to be the main source of power consumption in CMOS circuits. It is the power consumed when the output of the circuit exhibits a transition from logic 0 (1) to 1 (0). There are mainly two sources of dynamic power dissipation: 1) short-circuit power consumption, 2) power consumed by charging the load capacitance. Both depend on the transitive behavior of the circuit, i.e. the switching activity. Short circuit power consumption is due to the occurrence of a DC path between supply and ground rails when both transistors become on during a transition at the output. However, with proper sizing of the transistors, short-circuit power can be reduced up to a certain level, which can be ignored when compared to the power consumed for charging the load capacitance.

The dominant source of dynamic power consumption is the charging of the load capacitance that can be formulated as follows

$$Power = \alpha C f V_{dd}^2 \tag{1.1}$$

where f is the system clock frequency, V_{dd} the logical voltage transition (which is taken to be equal to the supply voltage), α is the switching activity factor (i.e. occurrence probability of a capacitor charging event), and C is the capacitance on which switching activity is observed (a charging event occurs).

1.2. Low Power Digital Signal Processing System Design

Digital signal processing circuits constitute a big portion of the VLSI market. They require repetitive and hence excessive amounts of arithmetic calculations one of which is the multiply accumulate operation. Unfortunately, these operations are power hungry

operations and hence there is much work done to reduce the computational complexity of signal processing applications. These efforts actually provide a basis for a low power digital signal processing system design [2].

Digital filters are the most frequently used elements in signal processing applications. Among digital filters, finite impulse response (FIR) filters are preferred due to their stability, easily achievable linear-phase property, and low quantization word length sensitivity. All these desirable properties come with a drawback compared to their recursive counterparts infinite impulse response (IIR) filters: increased computational workload.

The filtering operation of an FIR filter having N taps can be expressed by the equation

$$y[n] = \sum_{k=0}^{N-1} h[n]x[n-k] \tag{1.2}$$

where x represents the input data stream, h the coefficients of the filter, and y the output data stream. This equation can be realized in parallel using as many multipliers as the number of taps or sequentially using a multiply accumulate unit.

Digital FIR filters are realized with application specific integrated circuits (ASIC) or can be implemented by programming of digital signal processors (DSP). They require sequential arithmetic calculations, consume large power, and require dedicated fast hardware resources. Therefore, power aware design of digital filters is essential. In this work, digital FIR filters are examined since they are the basic building block of most digital filtering structures.

The architectural level approaches for low power DSP design exploit parallelism or pipelining in the algorithm and increase the throughput by employing extra hardware [3]. Reducing voltage then reduces the processing speed in a second step. Other methods proposed for reducing power dissipation in DSP systems attempt to reduce power by identifying operations that are redundant in the sense that they repeat computational steps

that do not yield new information by executing a power consuming operation. Many such approaches have been applied to FIR digital filters.

1.3. Design of Low-Power FIR Digital Filters

Low-power implementations of FIR filters are of interest in wireless receivers, battery-powered mobile applications such as the cellular phone and they have been investigated at various levels of abstractions in literature [4], [5]. The low-power FIR filter design problem can be divided into two categories depending on the choice of implementation: Constant coefficient and variable coefficient FIR filter synthesis.

1.3.1. Constant Coefficient (Multiplierless) Implementation of an FIR Filter

Constant coefficient FIR digital filters can be realized in parallel using as many multipliers as the number of coefficients in the filter. However, since multipliers are power and area consuming circuits, it is a common practice to represent coefficients as sums of signed-power-of-two (SPT) terms. Then, the multipliers are replaced with shift and add circuits. Hence, constant coefficient implementation is also referred to as the multiplierless realization of an FIR filter.

From the power perspective, each adder contributes as capacitance and switching activity to the power budget of the filter. Hence, the fewer the number of adders the less the switched capacitance and hence, the less power the filter will consume. The number of adders depends on the number of nonzero bits (SPT terms) of the quantized coefficients. A practical way to provide reduced SPT terms is to use canonic signed digit (CSD) representation since it offers fewer SPT terms in the representation than two's complement representation.

Research in designing FIR filters has concentrated on the design of algorithms, which generate floating-point filter coefficients. The resulting filters may not be practical for a VLSI implementation because of the potentially large amounts of hardware required to implement sufficiently accurate filter coefficient multipliers. Instead, using finite precision coefficients implemented with CSD coding using subexpression sharing [6-10]

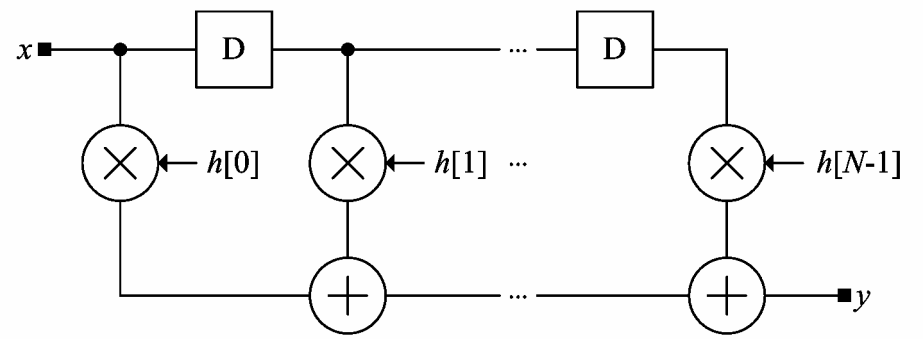

Figure 1.1. Parallel realization of an *N*-tap FIR filter

can be much more cost effective because only adders, subtractors, and shifters implement the coefficients. The cost (or coefficient complexity) of a parallel arithmetic VLSI implementation is largely dependent on the number of adders and subtractors required to implement the filter coefficients. Shifts are essentially free because they can be hardwired.

By simply quantizing infinite precision coefficients, the number of adders and subtractors cannot be controlled. Although coefficient recoding in CSD gives a reduction of 33 per cent, there is still no control over the total number of operations required. Hence, methods have been proposed in the literature for designing discrete coefficient FIR filters considering the reduction of hardware cost. These methods serve as a basis for the search for low power consumption. They rely on the idea that given the filter frequency characteristics (such as maximum pass-band ripple, minimum stop-band attenuation, etc.); the set of coefficients that satisfy the requirements are not unique. Therefore, one can search for a coefficient set that has reduced number of SPT terms and hence better area/power performance. Optimal [11,12] and suboptimal methods [13,14] have been proposed. There have also been attempts to combine the two approaches, where a set of coefficients with reduced number of ones is found and then sub-expression elimination is applied on the optimized coefficients [11,15], or the search for reduced number of SPT terms is changed to a search for reduced number of adders [16].

Coefficient scaling is extensively used in methods that search for reduced number of SPT term coefficients [13-15,17,18]. Scaling all coefficients by the same factor changes the distribution of SPT terms among coefficients without altering the shape of the frequency response. This can greatly improve the search for reduced SPT terms by providing a good starting point for a local search. Based on this, a method is proposed that

first searches for a scaling factor that minimizes the quantization error in the mean square sense [13]. In the second step of the algorithm, coefficients are optimized to minimize the normalized maximum ripple in the frequency response of the filter. An improved version of this method is proposed in [14] where the selection of the scaling factor is carried out considering the normalized peak ripple (NPR) of the frequency response of the filter. Then, a local search is performed starting from the quantized Remez coefficients scaled by the scaling factor that resulted in the minimum NPR. A common property of both algorithms is that they try to improve the frequency response of the filter by restricting the maximum number of SPT terms in each coefficient.

Another method that makes use of scaling is proposed in [18]. The algorithm differs from the previously mentioned methods in that the frequency response characteristics of the filter are taken as a constraint rather than as an objective. Furthermore, instead of finding the best frequency response (minimum NPR), the algorithm tries to find a coefficient set under the restriction of maximum number of SPT terms per coefficient with fewer SPT terms.

An optimization method extensively used in the design of discrete coefficient filters is mixed integer linear programming (MILP). The main reason for using MILP is that it can find optimum discrete coefficients for which NPR is much better than coefficients obtained by simply rounding infinite precision coefficients [19]. Given the filter length and the coefficient word-length, MILP can find the optimum frequency response [19-22]. However, the optimality criterion is NPR, not hardware or power cost. An MILP formulation where the optimization goal is to reduce the number of SPT terms, and hence hardware/power cost, is given in [12].

A drawback of MILP based methods is that solution time increases exponentially with the increase of number of taps of a filter. Therefore, it is not practical to use MILP for filters having large number of taps. In [15], a method that partially overcomes this problem by limiting the optimization variables to be the last D digits of the coefficients is introduced. D is taken to be no more than 3. The rest of the digits are initialized to the values obtained by quantizing/rounding the Remez solution of the filter.

The desired frequency response of an FIR filter imposes boundaries on the values coefficients can have, thereby limiting the coefficient search space. A linear programming (LP) formulation by which the minimum and maximum values are found for each coefficient is given in [11]. Within these boundary values, possible coefficient values exceeding the maximum number of SPT terms allowed per coefficient are eliminated. Then, the feasibility of each possible combination of coefficient values is checked. This is done using a branch and bound based search. By first checking the combination of coefficient values having the minimum number of SPT terms will eliminate the need for searching after a solution is found. However, the drastic increase of the search space, and hence search time, makes it inapplicable for filters having large number of taps. Moreover, the starting point of the search might be far away from the optimum solution causing even hard to find suboptimal initial solutions.

1.3.2. Variable Coefficient (Sequential) Implementation of an FIR Filter

Variable coefficient implementations of FIR filters are generally realized on DSPs where the filtering algorithm is translated into a series of multiply accumulate operations. The basic source of power consumption is the multiplication operation, which is performed on a dedicated multiplier unit. A filtering operation on a single multiply-accumulate (MAC) unit is shown in Figure 1.2. The power dissipated in a multiplier is related to the switching activity in the multiplier, which in turn is directly affected by the switching activity (Hamming Distance) at the inputs [23].

One approach targets programmable DSP architectures for identification of factors which contribute to dissipated energy and finding methods which reduce power hungry operations. Methods proposed for reducing power dissipated in the multipliers and busses of a generic Harvard architecture based digital signal processor use various techniques such as coefficient scaling, coefficient ordering, selective coefficient negation, removing common sub-expressions [4]. These techniques attempt to reduce power by identifying operations that are redundant in the sense that they repeat computational steps that do not yield new information by executing a power consuming operation. Bus power reduction is proposed by coefficient optimization [4], which attempts to reduce the Hamming distance

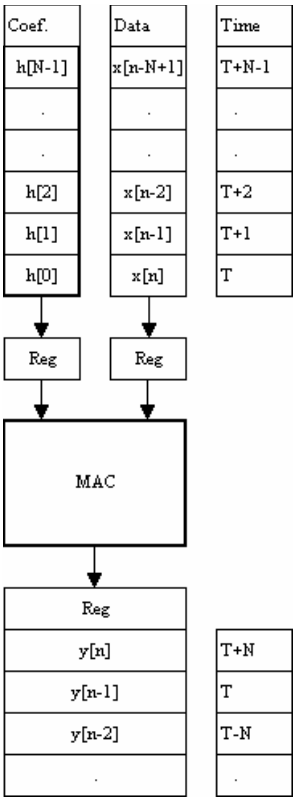

Figure 1.2. FIR filtering on a single Multiply-Accumulate unit

between successive coefficients in order to reduce activity on one of the multiplier inputs. Parallel processing architectures have also been proposed [5].

To reduce the power consumed in the MAC unit, the coefficients can be reordered so as to minimize the Hamming distance between successively applied coefficients [4,24,25]. However, reordering of coefficients requires reordering of data. It should be kept in mind that data is usually correlated and thus there are very few sudden jumps between consecutive data. This then may cancel out the reduced switching activity for the coefficients by increasing the Hamming distance in the data stream. This problem can be alleviated by both considering the Hamming distance of the data and coefficient stream simultaneously. In this case, the possible reordering of data, especially in real-time systems or even systems where data is stored in consecutive addresses in memory, may offset the expected gains in power. Thus, this approach should be restricted to problems where both data and the coefficients are readily available and reordering does not bring much power overhead.

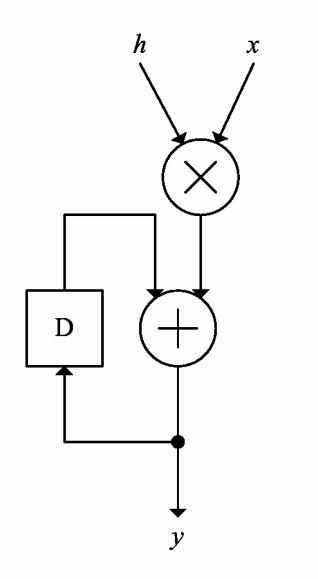

Figure 1.3. The structure of a Multiply-Accumulate unit

In [23], a method that only reduces the switching activity between filter coefficients is proposed. The method formulates the coefficient optimization problem as a local search problem to find low switching activity coefficients, thus resulting in suboptimal solutions. In [26], the same problem is formulated as an integer linear programming problem targeting low Hamming distance coefficients, thus, reducing the power consumed. However, it lacks the contribution of the number of ones in the coefficients thus resulting filters are optimum for Hamming distance but may not necessarily for power. The problem is formulated as an integer programming problem by doubling the number of variables, thus limiting the solution to filters having small number of taps (<70) [22].

1.4. Scope of Thesis

In this work, an algorithm for the design of low-power linear-phase FIR filters is proposed. The algorithm is a discrete coefficient FIR filter design algorithm which is discussed in Chapter 2. Depending on the choice of implementation, the algorithm optimizes filter coefficients for low power. Chapter 3 examines constant coefficient filters to be realized as an ASIC for which the number of nonzero digits in the coefficients is minimized. For variable coefficient filter implementations, which are studied in Chapter 4,

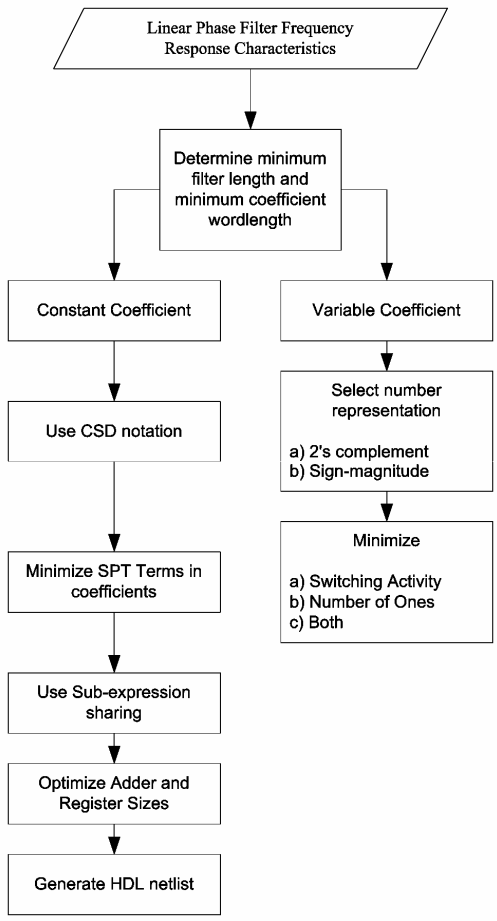

Figure 1.4. Low-power linear-phase FIR filter design flow

the switching activity between coefficients is minimized. Chapter 5 discusses the application of the algorithm for designing FIR based equalizers extensively used in communication systems. Conclusions and suggestions for future work are mentioned in Chapter 6.

2. DISCRETE COEFFICIENT LINEAR-PHASE FIR FILTER DESIGN

Optimal infinite precision filter coefficients obtained by a standard algorithm [27-29] are not very useful when it comes to realization. VLSI systems operate on finite wordlength. Therefore, quantization of the coefficients is a must for filters obtained by such algorithms. Quantization can be done by simply rounding the coefficients to the nearest integer or power-of-two values. However, the performance of the filter is significantly degraded from the optimal infinite precision coefficients. Hence, there has been much research on the design of discrete coefficient linear-phase FIR filters [19-22].

The driving force of the search for finite precision FIR filter design in early methods was the degradation in frequency response incurred by rounding and aimed to obtain the best frequency response filter given constrained by the coefficient wordlength B. Given the edge frequencies and weighings of the bands and the filter length, the best frequency response is searched in the minimax sense [19-22].

With the improvements in VLSI technology, FIR filters have started to be realized as standalone integrated circuits which steered the direction of the search for discrete FIR filter design methods to the design of low hardware implementation cost filters. The problem was re-defined as finding the filter with minimum hardware cost while still satisfying the filter frequency response characteristics [30,31].

Heuristic methods have been proposed where one has to be contended with the resulting frequency response of the low cost filter. That is, you could not exactly specify for example the stopband attenuation of a low pass filter. What you get is a filter that approximately satisfies the response [13,14,32].

Evolutionary strategies have been adapted to solve the discrete FIR filter design problem. The hardware cost can be added to the cost function with an appropriate weight. Again, the resulting filters trade off between hardware cost and frequency response

degradation [33-38]. The computational cost is high, and requires human intervention for parameter re-adjustment from one design to another design.

An optimization method by which you can find filters that exactly satisfy the frequency response characteristics is the mixed integer linear programming (MILP) method [12,15,21,22,39]. The main reason for using MILP is that it can find optimum discrete coefficients. Optimum was generally referred to in the minimax sense; however, the filter implementation cost is optimized as well. The basic shortcoming of MILP is its computational burden. The run-time of the algorithm grows exponentially with filter length, constraining it to filters with a length no more than 70 [22].

In this chapter, the GAM algorithm is presented for designing discrete coefficient linear-phase FIR digital filters. The algorithm uses linear programming to solve a set of equations and can find filters by keeping the quantization wordlength as small as possible. It is able to generate high-order filters in a reasonable amount of time. The generated filters exactly satisfy the required frequency response characteristics.

GAM can be used as a search algorithm for designing discrete coefficient linear-phase FIR filters having cost functions related to the number representation of the coefficients. If the target is a low-power filter to be realized on a generic DSP processor, the cost is the Hamming distance (switching activity) between successive coefficients represented either in two's complement or sign-magnitude notation. A nice property of GAM is that the number representation to be used in the final implementation does not affect the underlying problem, i.e. it does not increase the size of the underlying LP problem by introducing new variables (as it is the case with traditional MILP based approaches). This is because the actual cost function is not the cost function of the LP problem but is evaluated separately. This is desirable as far as the time spent to the LP problem is much longer than the time used for computing the cost function. Fortunately, this is generally the case with FIR filter design problems.

The outline of the chapter is as follows: First, a terminology for the description and design of linear-phase FIR filters is given. Then, the GAM algorithm is presented. An algorithm to find the minimum length and a formula for the minimum quantization

wordlength of a filter are given next. The effectiveness of the algorithm in designing discrete coefficient filters is shown in design examples.

2.1. Terminology

The frequency response $H(\omega)$ of a linear-phase FIR filter with impulse response $h[n]$ and length N is

$$H(\omega) = \sum_{n=0}^{N-1} h[n] e^{-j\omega n} \tag{2.1}$$

which can also be written in terms of the amplitude ($A(\omega)$) and phase terms as

$$H(\omega) = A(\omega) \mathrm{e}^{-j\omega(M-1)} \tag{2.2}$$

where M is approximately half the length of the filter N given by

$$M = \left\lceil \frac{N+1}{2} \right\rceil \tag{2.3}$$

The amplitude $A(\omega)$ is a real function of frequency given by

$$A(\omega) = \sum_{m=0}^{M-1} h[m] T_m(\omega) \tag{2.4}$$

where $T_m(\omega)$ is a trigonometric function determined by the length and type of symmetry of the filter. The values of $T_m(\omega)$ for the four possible types of linear-phase FIR filters are given in Table 2.1.

Filter design problems involve finding a filter with a frequency response that approximates a desired response to within a specified amount of error. The error is a function of frequency. Let $D(\omega)$ be the desired filter frequency amplitude response for which we are going to approximate a linear-phase FIR filter. Let $A(\omega)$ be the frequency

Table 2.1. The trigonometric function $T_m(\omega)$ for different types of linear-phase FIR filters

Type	N	Symmetry	$T_m(\omega)$
1	odd	symmetric	$\begin{cases} 1 & m = M-1 \\ 2\cos\left((M-m-1)\omega\right) & \text{otherwise} \end{cases}$
2	even	symmetric	$2\cos\left((M-m-0.5)\omega\right)$
3	odd	anti-symmetric	$2\sin\left((M-m-1)\omega\right)$
4	even	anti-symmetric	$2\sin\left((M-m-0.5)\omega\right)$

amplitude response of the resulting filter coefficients as given by (2.4). Let $\delta(\omega)$ be the approximation error, then the resulting linear-phase FIR filter should satisfy:

$$\left| A(\omega) - D(\omega) \right| \leq \delta(\omega) \tag{2.5}$$

where $\omega \in [0, \pi]$. In general, the frequency response of an FIR filter is defined for some disjoint frequency bands $\Omega_k \subset [0, \pi]$ with desired frequency response $D_k(\omega)$ and fixed error margin δ_k, $k = 1, 2, \cdots K$ such that (2.5) can be re-written for each k as,

$$\left| A(\omega) - D_k(\omega) \right| \leq \delta_k \tag{2.6}$$

Transition bands are the frequency bands for which no constraints are defined. They will be denoted with a superscript t as Ω^t. The width of a frequency band $\Omega = [\omega_1, \omega_2]$ is calculated by

$$\left| \Omega \right| = \omega_2 - \omega_1 \tag{2.7}$$

where ω_1 and ω_2 are called the edge frequencies of the band.

The aforementioned frequency response specifications for a low-pass linear-phase FIR filter are given in Table 2.2. The frequency response of the filter is given in Figure 2.1.

2.2. The GAM Algorithm

Given the wordlength B and filter length N, the GAM algorithm iteratively finds the coefficients of the resulting linear-phase FIR filter with zero-phase magnitude response $A(\omega)$ defined in (2.4), subject to the frequency response characteristics defined in (2.5). Since the filter is symmetric, the GAM algorithm should determine only M coefficients where M is calculated by (2.3).

There may be a lot of coefficients satisfying (2.5). However, once N is fixed, then the range of all possible values for each coefficient $h[i]$, $i = 0,1, \cdots, M-1$ can be determined by solving the following set of linear optimization problems independently:

Table 2.2. Frequency response characteristics of a low-pass FIR filter

Band (k)	Ω_k	$D_k(\omega)$	δ_k
1	$[0, \omega_1]$	1	δ_1
2	$[\omega_2, \pi]$	0	δ_2

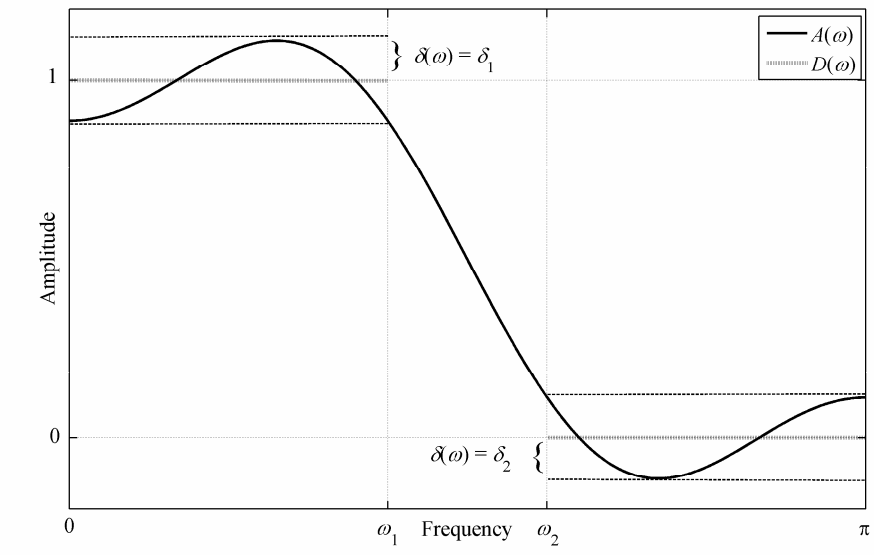

Figure 2.1. Frequency response characteristics of a low-pass FIR filter

$$h_{\min}[i] = \text{minimize } h[i]$$

such that

$$\left| \sum_{m=0}^{M-1} h[m] T_m(\omega) - D(\omega) \right| \leq \delta(\omega) \tag{2.8}$$

$$h_{\min}[i] = \text{maximize } h[i]$$

such that

$$\left| \sum_{m=0}^{M-1} h[m] T_m(\omega) - D(\omega) \right| \leq \delta(\omega) \tag{2.9}$$

where $\omega \in [0, \pi]$.

Note that $h_{\min}[i]$ and $h_{\max}[i]$ are real values and coefficient $h[i]$ is a number that appears in this range, i.e. $h[i] \in [h_{\min}[i], h_{\max}[i]]$. Without the finite wordlength constraint it can take over infinitely many numbers. However, since coefficients are restricted to be represented with a finite wordlength B all possible values of $h[i]$ form a finite set. Hence, let V_i be such a digital value set of $h[i]$. The selected coefficients from $\bigcup_{m=0}^{M-1} V_m$ must satisfy (2.5).

The algorithm that finds the feasible value sets is shown in Figure 2.2. At each iteration, a value set for the corresponding coefficient is found. If the value set happens to be empty, the algorithm returns an empty set. This means that the problem cannot be solved because either the number of taps N or wordlength B is less than required.

A simple branch-and-bound algorithm that runs on $\bigcup_{m=0}^{M-1} V_m$ will obviously find the optimal solution provided that enough memory and time is given. It is easy to observe that there exist $\prod_{m=0}^{M-1} |V_m|$ combinations to be searched. However, a design automation tool should find a considerably good solution in a reasonable time using a reasonable amount of memory. The GAM algorithm, whose pseudo-code is given in Figure 2.3, is a modified branch-and-bound algorithm; hence its worst case performance is exponential. Yet, it reaches a reasonably good result in a sufficiently short time since it refines the search

```
VALUE_SET(M, B, D(ω), δ(ω))
  i = 0;
  WHILE (i < M)
    Obtain h_min[i] and h_max[i] by solving equations (2.8) and (2.9);
    V_i = { ∀v ∈ [h_min[i],h_max[i]] | v is a B-digit binary number};
    IF (V_i = ∅)
      RETURN ∅;
    END-IF
    i = i + 1;
  END
  RETURN ⋃_{m=0}^{M-1} V_m ;
```

Figure 2.2. Algorithm VALUE_SET finds the feasible value sets of the coefficients

space during its execution. Obviously, a program that uses GAM as a subroutine can also be developed to find the optimal solution.

The operation of the GAM algorithm can be summarized as follows: Starting from the initial tap (i.e. $i = 0$), the algorithm iteratively selects a value v^* from the refined value set V_i^s of tap coefficient $h[i]$ such that v^* is closest to the average of minimum ($h^s_{\min}[i]$) and maximum ($h^s_{\max}[i]$) values for $h[i]$. After this value is assigned to $h[i]$, v^* is removed from the value set and moved to the solution set H^*, which is an ordered set. In other words, the first element in H^* corresponds to $h[0]$ and the last element in H^* corresponds to $h[M-1]$. At each iteration, only one tap is fixed. After fixing each tap, the following pair of optimization equations is solved for the next tap:

$$h^s_{\min}[i] = \text{minimize } h[i]$$

such that

$$\left| \sum_{m=0}^{M-1} h[m] T_m(\omega) - D(\omega) \right| \le \delta(\omega) \tag{2.10}$$

$$h^s_{\max}[i] = \text{maximize } h[i]$$

such that

$$\left| \sum_{m=0}^{M-1} h[m] T_m(\omega) - D(\omega) \right| \le \delta(\omega) \tag{2.11}$$

```
GAM(M , L, ⋃_{m=0}^{M} V_m )
  H* = ∅;
  Obtain h^s_min[0] and h^s_max[0] by solving equations (2.10) and (2.11);
  V_0^s = VALUE_SELECT(L, h^s_min[0], h^s_max[0], V_0);
  i = 0;
  WHILE (i ≥ 0)
    IF (V_i^s ≠ ∅)
      h[i] = v* such that v* is the first element of ordered set V_i^s;
      H* = H* ∪ {h[i]};
      V_i^s = V_i^s − {v*};
        IF (i < M−1)
          i = i + 1;
          Obtain h^s_min[i] and h^s_max[i] by solving equations (2.10) and (2.11);
          V_i^s = VALUE_SELECT(L, h^s_min[i], h^s_max[i], V_i);
        ELSE-IF (problem is feasible)
          H* is a solution to the problem;
        END-IF
    ELSE
      H* = H* − {h[i]};
      i = i − 1;
    END-IF
  END
```

Figure 2.3. The GAM Algorithm

where $\omega \in [0, \pi]$ and $h[m] = \begin{cases} \text{fixed} & 0 \le m < i \\ \text{free} & i \le m < M \end{cases}$ for both problems.

It should be noted that these equations are different from equations (2.8) and (2.9) so as to refine the search space for the i'th tap after fixing $i-1$ taps. The superscript s in these equations and in the pseudo-code stands for the refined variables. Fixing the values of $i-1$ taps moves the boundary values of the i'th tap towards each other and the number of possible values for this tap will be reduced. If the refined value set (V_i^s) of $h[i]$ is not empty, then the iteration goes on as usual, but the mid-value $h_{\text{mid}}[i]$ might change due to the possibility that the moving amount of each boundary might not be equal. However, if the refined value set of $h[i]$ is empty, then the selected value for $h[i-1]$ is removed from H^* and the next value in the value set of $h[i-1]$ is selected for the next iteration and this is repeated until a nonempty value set of $h[i]$ is obtained.

VALUE_SELECT(L, $h^s_{\min}[i]$, $h^s_{\max}[i]$, V_i)
  $V_i^s = \{ \forall v \in V_i \mid h^s_{\min}[i] \le v \le h^s_{\max}[i] \}$;
  $h^s_{\text{mid}}[i] = (h^s_{\min}[i]+h^s_{\max}[i])/2$;
  Arrange $V_i^s = (v_1, v_2, \ldots, v_{|V_i^s|})$ such that
    $|v_1 - h^s_{\text{mid}}[i]| < |v_2 - h^s_{\text{mid}}[i]| < \cdots < |v_{|V_i^s|} - h^s_{\text{mid}}[i]|$;
  Limit $V_i^s = (v_1, v_2, \ldots, v_{|V_i^s|})$ such that $|V_i^s| \le L$;
  RETURN V_i^s;

Figure 2.4. The algorithm used to select values from the value set into a refined value set

The algorithm finds a solution until all M taps are selected. Experimentally, it has been observed that the algorithm finds a solution in less than $2M$ iterations depending on the tightness of the search space. The algorithm finds a solution in at most L^M iterations where L is a parameter used to limit the maximum size of the refined value sets V_i^s. That is, when selecting the values from the feasible value set V_i^s of coefficient $h[i]$, L values in the vicinity of $h^s_{\text{mid}}[i]$ are selected. If L is set to infinity, all values in V_i^s are selected. The purpose of limiting the size of the refined feasible value set is to reduce search time by avoiding unnecessary search devoted to possibly infeasible combinations. How can one predict these infeasible combinations? Experimentally it is observed that feasible solutions are the combinations of coefficient values which are selected in the neighborhood of the middle values, $h^s_{\text{mid}}[i]$. The values close to the boundary values of $h[i]$ either do not or rarely produce feasible solutions. Practically, setting $L = 2$ is enough when the coefficient wordlength B is set to its lowest possible value.

2.3. Determining the Minimum Length of an FIR Filter

The length of the filter is important because it affects hardware sources and computational steps. Therefore, it is a practical issue to realize the filter with shortest possible length. There have been attempts to formulize the FIR filter length using the filter frequency response characteristics. One is the formula given in [40]

$$N \approx -\frac{2}{3}\log_{10}(10\delta_1\delta_2)\frac{2\pi}{\omega_2 - \omega_1} + 1 \tag{2.12}$$

where the parameters δ_1, δ_2, ω_1, and ω_2 correspond to the filter specifications as given in Figure 2.1. The filter length N determined by (2.12) is an estimation; however, it can be

used as the starting point for a search for the minimum length of the filter.

Recall from the previous section that by solving equations (2.8) and (2.9) feasible value sets for all the coefficients can be found independently. For the filter to be realizable with the given filter length, none of the value sets should be empty. For coefficient $h[i]$, without considering the wordlength, to have a non-empty feasible value set, the boundary values obtained by solving (2.8) and (2.9) should satisfy the relation $h_{\min}[i] \le h_{\max}[i]$. That is the range $[h_{\min}[i],h_{\max}[i]] \neq \varnothing$. Thus, one can check out the feasibility of the length the FIR filter by looking at the boundary values of any of the coefficients. Experimentally, it has been observed that V_0 is the smallest subset of $\bigcup_{m=0}^{M-1} V_m$. Therefore, $h[0]$ should be checked first.

An algorithm that finds the minimum length of a linear-phase FIR filter using the estimation of (2.12) as a starting point is shown in Figure 2.5. The operation of the algorithm is as follows: When equations (2.8) and (2.9) are solved for $h[0]$, there can be three cases. First, if the range $[h_{\min}[0],h_{\max}[0]] = \varnothing$, a filter with N coefficients cannot satisfy the filter specifications. Hence, we need to increase the length of the filter. This is

```
FILTER_LENGTH(D(ω), δ(ω))
  Obtain N by computing equation (2.12)
  M = ⌈(N+1)/2⌉;
  Obtain h_min[0] and h_max[0] by solving equations (2.8) and (2.9);

  IF ([h_min[0],h_max[0]] = ∅)  /* N is an under-estimate */
    DO
      N = N + 2;
      M = ⌈(N+1)/2⌉;
      Obtain h_min[0] and h_max[0] by solving equations (2.8) and (2.9);
    WHILE ([h_min[0],h_max[0]] = ∅)
  ELSE-IF (0∈ [h_min[0],h_max[0]]) /* N is an over-estimate */
    DO
      N = N − 2;
      M = ⌈(N+1)/2⌉;
      Obtain h_min[0] and h_max[0] by solving equations (2.8) and (2.9);
    WHILE (0∈ [h_min[0],h_max[0]])
  END-IF
  RETURN N;
```

Figure 2.5. The algorithm to find the minimum filter length

done until a non-empty range of values for $h[0]$ is obtained, i.e. $[h_{\min}[0],h_{\max}[0]] \neq \varnothing$. Second, if 0 is in the range $[h_{\min}[0],h_{\max}[0]]$, we can set $h[0] = 0$. Hence, the coefficient set can be tailored by removing $h[0]$ (and hence $h[N\text{-}1]$ due to symmetry) without affecting the frequency response. Thus, we can decrease the length of the filter. This makes the second coefficient, namely $h[1]$, the first coefficient. Then, re-phrasing the problem with the new filter length $N = N - 2$, the boundary values of the new first coefficient are computed. This procedure is followed until $0 \notin [h_{\min}[0],h_{\max}[0]]$. At this point the length of the filter is the minimum required, i.e. $N_{\min} = N$. Third, $h[0]$ has a feasible value range excluding 0, i.e. $[h_{\min}[0],h_{\max}[0]] \neq \varnothing$ and $0 \notin [h_{\min}[0],h_{\max}[0]]$. Hence, the filter length N is already equal to $N_{\min}$.

2.4. Wordlength Estimation in FIR filters

From the realization perspective of FIR filters, either on a generic DSP or dedicated hardware unit, short coefficient wordlength is a benefit. In the DSP implementation, fixed point DSPs are preferred for their speed and low cost where the coefficient wordlength is limited to the internal hardware wordlength. Therefore, it is a must to keep the wordlength of the filter coefficients at most equal to the DSP wordlength. On the other hand, in the hardware implementation, short wordlength reduces the hardware cost by reducing the size of the adders and delay elements.

Given the frequency response characteristics, is it possible to generate filter coefficients with any desired wordlength? The answer to this question is unfortunately, no. It is shown in [41] that the degradation in frequency response incurred by reducing the quantization wordlength B, can be compensated by increasing the filter length N. However, after a certain wordlength increasing the filter length will not suffice and the desired frequency response will no more be achievable. That is, for an FIR filter there is a lower bound for the coefficient wordlength that can satisfy the desired frequency response characteristics.

Methods proposed in the literature [41-46] consider degradation incurred by rounding optimal infinite precision coefficients. The filters discussed are optimal in the

minimax sense. The formula proposed in [43] gives a lower bound for the wordlength for a prescribed deviation in the degradation in in-band rejection (δ_k) for frequency band Ω_k. This bound is obtained by assuming that the infinite precision coefficients could achieve exactly the desired response D_k in the corresponding band, i.e. the error margin $\delta_k = 0$ (which corresponds to $-\infty$ in dB as it is given in [43]). The formula can be re-written using the terminology of this text as

$$B_{\min} = \left\lceil 1 + \log_2 \left(\frac{\delta_R}{\sqrt{\frac{2N-1}{3}}} \right) \right\rceil \tag{2.13}$$

where N is the length of the filter, δ_R is the error margin for rounded coefficients. The frequency band with the minimum ripple δ_R will determine the lower bound. This is an optimistic lower bound since there will always be an error margin introduced by the infinite precision coefficients due to finite filter length.

By using the filter length estimation formula of (2.12) and following the methodology presented in [41] a lower bound for the wordlength can be obtained as follows. Re-writing (2.12) for a frequency band weighing of one we obtain

$$N \approx -\frac{2}{3}\log_{10}\left(10\delta^2\right)\frac{2\pi}{\omega_2 - \omega_1} + 1 \tag{2.14}$$

where $\delta_1 = \delta_2 = \delta$ from (2.12). Note that δ is the error margin obtained from the infinite precision coefficients. Let the maximum error margin (i.e. ripple) after rounding be δ_F, then, the lower bound for the wordlength is given by

$$B_{\min} = \left\lceil 1 + \log_2 \left(\frac{\delta_F - \delta}{\sqrt{\frac{2N-1}{3}}} \right) \right\rceil \tag{2.15}$$

where N is the filter length obtained by (2.14). $\delta_F - \delta$ gives the amount of deviation in error.

The formula proposed in this work gives a lower bound for the quantization wordlength of a linear-phase FIR. It is obtained by designing several filters using the GAM algorithm. It is simple and can directly be calculated from the frequency response characteristics of the filter independent of the filter length.

For a linear-phase FIR filter let the frequency response be defined over K frequency bands. Then, a lower bound for the wordlength can be found by

$$B^*_{\min} = -\sum_{k=1}^{K} W_k \log_2(\delta_k) - \log_2\left(\frac{\left|\Omega^t\right|_{\min}}{\pi}\right) \tag{2.16}$$

where $\left|\Omega^t\right|_{\min}$ is the minimum transition band-width, W_k is the weight of a frequency band in terms of its band-width calculated as

$$W_k = \frac{\left|\Omega_k\right|}{\sum_{i=1}^{K}\left|\Omega_i\right|} \tag{2.17}$$

For the case where the ripple is equal in all bands, namely $\delta_k = \delta$, (2.16) reduces to

$$B^*_{\min} = -\log_2(\delta) - \log_2\left(\frac{\left|\Omega^t\right|_{\min}}{\pi}\right) \tag{2.18}$$

Note that the minimum wordlength obtained by (2.16) and (2.18) is a real value. It has been observed that the rounding method to be applied on $B^*_{\min}$ depends on the minimum transition band-width. The minimum wordlength for different range of transition band-widths can be found by

$$B_{\min} = \begin{cases} \text{round}\left(B^*_{\min}\right) - 1, & \left|\Omega^t\right|_{\min} < 0.02\pi \\ \text{round}\left(B^*_{\min}\right), & 0.02\pi \le \left|\Omega^t\right|_{\min} \le 0.2\pi \\ \text{round}\left(B^*_{\min}\right) + 1, & 0.2\pi < \left|\Omega^t\right|_{\min} \end{cases} \tag{2.19}$$

This lower bound is reached by increasing the minimum length of the filter by 4-5 per cent.

2.5. Design Examples

The performance of GAM algorithm is tested on several linear-phase FIR filters. The frequency response characteristics of the test filters are given in Table 2.3. The minimum filter length ($N_{\min}$) is obtained using the algorithm given in Figure 2.5. The lower bound for the coefficient wordlength ($B_{\min}$) is calculated using (2.19). L1 is a high pass filter. L2, S2, A, B, and C are low pass filters. D is a band pass filter and F is a multi-band filter. E is a low pass M-band filter where M is equal to 8.

The GAM algorithm is implemented in the C programming language. GAM uses the linear programming (LP) library of QSOPT [47]. The frequency grid on which the frequency response is evaluated consists of approximately 1000 frequency samples for all filters. This makes approximately 2000 constraints for the LP problem solved by GAM. All the run-times presented are the results obtained from running the GAM algorithm on a PC with a Pentium D 2.8GHz processor and 1 GB RAM.

The feasible value sets of coefficients are generated using the algorithm in Figure 2.2. Filters are designed for both minimum length and minimum wordlength cases. The minimum wordlength filter is obtained by increasing the filter length until the calculated minimum wordlength is attained. The filter length (N), coefficient wordlength (B), and run time of the GAM algorithm to find the first solution for all filters are given in Table 2.4.

Most of the time to find an initial solution is spent for the selection of all taps, starting from $h[0]$ to $h[M]$. This can be considered as a setup time for the search and the time spent during this selection process is unavoidable. At each selection, two LP problems ((2.10) and (2.11)) are solved to determine the boundary values of the coefficient. Most of

Table 2.3. Characteristics of the test filters

Filter	N_{min}	B_{min}	k	$\Omega_k(\times\pi)$	D_k (ω)	δ_k
L1	117	16	1	[0, 0.74]	0	0.0001
			2	[0.8, 1]	1	0.005756
L2	61	13	1	[0, 0.2]	1	0.028774
			2	[0.28, 1]	0	0.001
S2	59	13	1	[0, 0.042]	1	0.011512
			2	[0.14, 1]	0	0.001
A	56	13	1	[0, 0.125]	1	0.01
			2	[0.225, 1]	0	0.001
B	101	11	1	[0, 0.2]	1	0.01
			2	[0.24, 1]	0	0.01
C	311	13	1	[0, 0.125]	1	0.005
			2	[0.14, 1]	0	0.005
D	109	13	1	[0, 0.10]	0	0.001
			2	[0.15, 0.4]	1	0.01
			3	[0.45, 1]	0	0.001
E	471	13	1	[0, 0.12]	1	0.005
			2	[0.13, 1]	0	0.005
F	95	12	1	[0, 0.05]	0	0.01
			2	[0.1, 0.3]	1	0.01
			3	[0.35, 0.45]	0	0.01
			4	[0.5, 0.7]	0.5	0.005
			5	[0.75, 1]	0	0.001

the time is spent for solving these LP problems. The LP problem size, and hence its solution time, depends on the filter length and number of frequency response constraints. Since, for our case, the number of constraints is approximately same for all filters (~2000), the length of the filter determines the initial solution time. This is the reason for the long run-time of filters C and E which have a filter length above 300.

Table 2.4. Run-time of the GAM algorithm for finding an initial solution for the test filters

Filter	Minimum length			Minimum wordlength		
	N	*B*	Run-time	*N*	*B*	Run-time
L1	117	19	1m	127	16	2m
L2	61	14	<1m	63	13	<1m
S2	59	16	<1m	67	13	<1m
A	56	19	<1m	59	13	17m
B	101	14	<1m	105	11	1m
C	311	16	20m	323	13	17m
D	109	19	<1m	121	13	14m
E	471	14	1h 3m	477	13	59m
F	95	14	4m	103	12	<1m

2.6. Summary

An algorithm for designing finite wordlength linear-phase FIR filters is presented. Although the worst case run time of the algorithm is exponential, its capability to find a solution in a reasonable amount of time makes it a desirable CAD tool for designing discrete coefficient linear-phase FIR filters. The effectiveness of the algorithm is more apparent on high-order filters.

Here, we propose an empiric formula for the lower bound of the quantization wordlength of a linear-phase FIR. It is obtained by designing several filters using the GAM algorithm. It can directly be calculated from the frequency response characteristics independent of the filter length.

3. LOW-POWER DESIGN OF CONSTANT COEFFICIENT FIR FILTERS

Constant coefficient applications are also referred to as multiplierless implementation of FIR filters. In two's complement binary notation, digits can take over two values {0,1}. It is obvious that, in a multiplication operation, the zeros in the multiplying operand will cause the corresponding partial products to be zero. Furthermore, if this operand is constant then the multiplicand multiplied by the zero digits will always generate zero valued partial products. Using adders for these products is a waste of resources which is unavoidable when a multiplier is used. Therefore, it is a practical concern to replace the multiplier by a tree of adders which adds up the shifted partial products corresponding to the nonzero bits of the constant operand. Therefore, the less the number of nonzero bits in the constant operand, the less the number of adders required.

To reduce the number of nonzero bits in the coefficients, one can use signed digit (SD) representation other than the two's complement. In SD representation, the bits of the quantized coefficient can take over the values {−1,0,1}. The advantage of SD representation comes from the fact that it can represent numbers not only by sums of numbers but also differences of numbers. This can be viewed on the following example where a number (here 15) has four nonzero bits in two's complement representation 001111 (=8+4+2+1), and two nonzero bits in SD representation 01000$\underline{1}$ (=16−1), where $\underline{1}$ stands for a −1.

Canonic signed digit (CSD) representation is a SD representation in which no adjacent bits can take over nonzero values. Moreover in CSD representation numbers have unique representations. To see it on an example the number three can be represented in 4-bit SD as 0011 and 010$\underline{1}$. However, in CSD the only representation is 010$\underline{1}$.

In the hardware realization of constant coefficient FIR filters multipliers are replaced with adders and shifters. From the power perspective, the fewer the number of adders the less power the filter will consume. The number of adders depends on the number of nonzero bits (SPT terms) of the quantized coefficients.

In this chapter, a modified version of the GAM algorithm for designing low-power constant coefficient linear-phase FIR filters is proposed. The outline of the chapter is as follows: First, the formulation of the problem for minimizing SPT terms in the filter coefficients represented in CSD is presented. Then, the modifications required for the GAM algorithm to minimize SPT terms are given. Next, hardware implementation issues are discussed. These are identification of the parameters affecting the critical path of an FIR filter, calculation and further optimization of the adders and registers for the final implementation. At the end, the results and comparison of the filters designed by GAM and other algorithms appeared in the literature is given.

3.1. Problem Formulation

The coefficients ($h[n]$) of a linear-phase FIR filter can be written in B-bit CSD representation as:

$$h[i] = \sum_{j=0}^{B-1} x_{i,j} 2^{-j} \tag{3.1}$$

where $x_{i,j} \in \{-1,0,1\}$ corresponds to the j'th bit of coefficient $h[i]$ and will be referred to as an SPT term. Note that j=0 corresponds to the most significant bit. Equation (3.1) is actually the SD representation. Remember that in CSD, two adjacent bits cannot have nonzero values. So, to ensure CSD, the following constraint should be added

$$\left| x_{i,j} \right| + \left| x_{i+1,j} \right| \leq 1 \tag{3.2}$$

where $j = 0,1,\ldots,B-2$. Using (3.1), the cost of SPT terms in a coefficient can be written as

$$\sum_{j=0}^{B-1} \left| x_{i,j} \right| \tag{3.3}$$

If the filter length is N, the total number of SPT terms in the coefficients is given by

$$C_{\text{SPT}} = \sum_{i=0}^{N-1} \sum_{j=0}^{B-1} \left| x_{i,j} \right| \tag{3.4}$$

which can be rewritten for a symmetric (anti-symmetric) filter as

$$C_{\text{SPT}} = \begin{cases} 2\sum_{i=0}^{M-1} \sum_{j=0}^{B-1} \left| x_{i,j} \right| + \sum_{j=0}^{B-1} \left| x_{M,j} \right| & , \quad N \text{ odd} \\ 2\sum_{i=0}^{M} \sum_{j=0}^{B-1} \left| x_{i,j} \right| & , \quad N \text{ even} \end{cases} \tag{3.5}$$

where M is given by (2.3).

Now, suppose we want to minimize the total number of SPT terms in a linear-phase FIR filter having a magnitude response $A(\omega)$, satisfying the desired frequency response $D(\omega)$ with an error margin $\delta(\omega)$ as defined in (2.5). Then using (3.1) and (3.5), for an even length filter the problem can be written as

Minimize $$2\sum_{i=0}^{M-1} \sum_{j=0}^{B-1} \left| x_{i,j} \right|$$

Such that

$$\left| \sum_{i=0}^{M-1} \left(\sum_{j=0}^{B-1} x_{i,j} 2^{-j} \right) T_i(\omega) - D(\omega) \right| \le \delta(\omega) \tag{3.6}$$

and $\left| x_{i,j} \right| + \left| x_{i,j+1} \right| \le 1$ for $j = 0,1, \cdots, B-2$. If the number of SPT terms in each coefficient is to be constrained to a pre-determined value P the following constraint should be added to the problem for each coefficient $h[i]$.

$$\sum_{j=0}^{B-1} \left| x_{i,j} \right| \le P \quad , \quad i = 0, \cdots, M \tag{3.7}$$

The problem in (3.6) is a non-linear optimization problem due to the absolute value operator in the cost function. However, it can be easily converted to a linear optimization

problem by introducing two new variables for each variable $x_{i,j}$ [12]. That is, $x_{i,j}$ can be expressed as a difference of two variables, namely

$$x_{i,j} = x^{+}_{i,j} - x^{-}_{i,j} \tag{3.8}$$

where $x^{+}_{i,j}, x^{-}_{i,j} \in \{0,1\}$. Then, the new formulation of the problem is

Minimize $$2\sum_{i=0}^{M-1}\sum_{j=0}^{B-1}\left(x^{+}_{i,j} - x^{-}_{i,j}\right)$$

Such that

$$\left|\sum_{i=0}^{M-1}\left(\sum_{j=0}^{B-1}\left(x^{+}_{i,j} - x^{-}_{i,j}\right)2^{-j}\right)T_i(\omega) - D(\omega)\right| \le \delta(\omega) \quad , \quad \omega \in [0, \pi]$$

$$x^{+}_{i,j} + x^{-}_{i,j} + x^{+}_{i,j+1} + x^{-}_{i,j+1} \le 1 \quad \text{for} \quad j = 0,1, \cdots, B-2. \tag{3.9}$$

This is a combinatorial optimization problem which can be solved optimally using MILP. The number of variables to be determined is $2MB$. The absolute value in the constraint of the formulation does not harm the linearity of the problem since it can be written as

$$D(\omega) - \delta(\omega) \le \sum_{i=0}^{M-1}\left(\sum_{j=0}^{B-1}\left(x^{+}_{i,j} - x^{-}_{i,j}\right)2^{-j}\right)T_i(\omega) \le D(\omega) + \delta(\omega) \tag{3.10}$$

3.2. The GAM Algorithm for SPT Term Minimization

The GAM algorithm can be easily modified to a search algorithm for finding a coefficient set with reduced number of SPT terms. A pre-defined cost function can be evaluated when a solution is found. Here, the cost function is the total number of SPT terms in the coefficients which was already formulated as in (3.5). However, instead of calculating the cost when a solution is found it is better to evaluate it incrementally at each coefficient selection and add the number of SPT terms of the current selection given by (3.3) to the total SPT term cost. So, after the last coefficient is selected the total number of SPT terms is calculated automatically.

A nice property of GAM is that the number representation to be used in the final implementation does not affect the underlying problem, i.e. it does not grow the size of the underlying LP problem by introducing new variables like in (3.9). This is because the actual cost function is not the cost function of the LP problem but evaluated separately. Calculation of the cost for each selected value brings computational overhead. Although this cost is small compared to the computational burden incurred by the solution of (2.10) and (2.11) (which are solved by linear programming) to find the boundary values, it can be avoided by pre-calculating the number of SPT terms of the values in the feasible value sets of the coefficients and storing them separately. Limiting the coefficients to have at most P SPT terms does not require additional constraints to be added to the problem as in (3.9). The values that have SPT terms more than P are simply removed from their sets prior to the search.

The modified GAM algorithm whose pseudo-code is given in Figure 3.1 will be termed as SPTGAM. SPTGAM iteratively finds coefficients with reduced number of SPT terms given the wordlength B and filter length N. The resulting linear-phase FIR filter has zero-phase magnitude response $A(\omega)$ as defined in (2.4), and is subject to the frequency response characteristics as defined in (2.5).

An advantage of branch and bound based algorithms on enumeration based algorithms is that they employ a cut-off mechanism that compares the best solution and best obtainable solution at any point of the search tree and decide not to go deeper in the search tree if a better solution cannot exist. This, in turn, avoids searching all possible combinations. In the original GAM algorithm, since no cost function was defined, a cut-off mechanism was not employed. The algorithm can terminate whenever a solution is found. However, since the initial solution found by the SPTGAM algorithm may not necessarily be the best solution, the algorithm needs to probe further to find better solutions. Therefore, we need a cut-off mechanism that will avoid searching all possible combinations.

The cut-off mechanism in SPTGAM is as follows: The number of SPT terms of the best solution is denoted as O, which is set to infinity at the beginning since there is no solution. The number of SPT terms of the value v^* (used to fix coefficient $h[i]$) is denoted

```
SPTGAM (M , L, ⋃_{m=0}^{M−1} V_m )
  H* = ∅;
  O = ∞;
  Obtain U_i by computing (3.14) for i = 0,..,M−2;
  Obtain h^s_min[0] and h^s_max[0] by solving equations (2.10) and (2.11);
  V_0^s = VALUE_SELECT(L, h^s_min[0], h^s_max[0], V_0);
  F_0 = 0;
  i = 0;
  WHILE (i ≥ 0)
    IF (V_i^s ≠ ∅)
      h[i] = v* such that v* is the first element of ordered set V_i^s;
      p = number of SPT terms in v*;
      H* = H* ∪ {h[i]};
      V_i^s = V_i^s − {v*};
      IF (F_i + p + U_i < O) { /* If there can be a solution having SPT terms less than O */
        IF (i < M−1)
          i = i + 1;
          F_i = F_{i−1} + p;
          Obtain h^s_min[i] and h^s_max[i] by solving equations (2.10) and (2.11);
          V_i^s = VALUE_SELECT(L, h^s_min[i], h^s_max[i], V_i);
        ELSE-IF (problem is feasible)
          H* is a solution to the problem;
          O = number of SPT terms of H*;
        END-IF
      END-IF
    ELSE
      H* = H* − {h[i]};
      i = i − 1;
    END-IF
  END
```

Figure 3.1. The modified GAM algorithm for minimizing SPT terms

by p. The total number of SPT terms in the coefficients that are already fixed is denoted by F_i. The predicted minimum possible number of SPT terms of the unfixed coefficients is denoted by U_i. If $F_i + p + U_i \geq O$, there is no need to further branch to the next coefficient since there does not exist a better solution then the current solution. In this case, the algorithm proceeds by setting the current coefficient $h[i]$ to its next feasible value v^*. The main point here is to forecast the number of SPT terms of the unfixed coefficients ($h[i+1]$ to $h[M-1]$), namely U_i.

Although many other SPT term prediction strategies can be found, to give an idea of the effect incurred by selecting a strategy, three of them will be mentioned here.

i) U_MIN: Recall that V_i is defined to be the feasible value set of coefficient $h[i]$. In V_i, a value v which has the minimum number of SPT terms is found. Then, let $P_{i,\min}$ be the number of SPT terms of v. Then the predicted minimum number of SPT terms for the unfixed coefficients while fixing coefficient $h[i]$ is

$$U_i = \sum_{m=i+1}^{M-1} P_{m,\min} \tag{3.11}$$

ii) U_AVG: Unfortunately, (3.11) gives an underestimated value to the actual achievable number of SPT terms. Moreover, it causes unnecessary search effort devoted to the end of the tree, i.e. the search is stuck at the last coefficients. Instead, there is a need for another prediction value that allows the whole search space to be searched in a reasonable amount of time. One can use the average number of SPT terms a coefficient can have by averaging the SPT terms of the values in the feasible value set (V_i). Then let $P_{i,\text{avg}}$ be this value which is calculated as

$$P_{i,\text{avg}} = \sum_{k=1}^{|V_i|} p_k \Big/ |V_i| \tag{3.12}$$

where p_k is the number of SPT terms of value $v_k \in V_i$. Then

$$U_i = \sum_{m=i+1}^{M-1} P_{m,\text{avg}} \tag{3.13}$$

iii) U_SUB_AVG: It is observed that the predicted number of SPT terms U_i in (3.13) is an overestimate. This causes the search to conclude in a very short time with a highly suboptimal result. Choosing the lower integer bound of the average, instead of the average itself, is the best choice in terms of search time-optimality trade-off. Now the predicted number of SPT terms for $M - i$ unfixed coefficients is

$$U_i = \sum_{m=i+1}^{M-1} \lfloor P_{m,\text{avg}} \rfloor \tag{3.14}$$

MIN_SPT_FIRST (L, $h^s_{\min}[i]$, $h^s_{\max}[i]$, V_i) /* VALUE_SELECT*/
$V_i^s = \{ \forall v \in V_i \mid h^s_{\min}[i] \le v \le h^s_{\max}[i] \}$;
$h^s_{\text{mid}}[i] = (h^s_{\min}[i]+h^s_{\max}[i])/2$;
Arrange $V_i^s = (v_1, v_2, \ldots, v_{|V_i^s|})$ such that
$|v_1 - h^s_{\text{mid}}[i]| < |v_2 - h^s_{\text{mid}}[i]| < \cdots < |v_{|V_i^s|} - h^s_{\text{mid}}[i]|$;
Limit $V_i^s = (v_1, v_2, \ldots, v_{|V_i^s|})$ such that $|V_i^s| \le L$;
Arrange $V_i^s = (v_1, v_2, \ldots, v_{|V_i^s|})$ such that $p_1 < p_2 < \cdots < p_{|V_i^s|}$;
RETURN V_i^s;

Figure 3.2. The new value selection algorithm to select the values with the least SPT term count first

Another strategy that needs to be revised due to the addition of a cost function to the GAM algorithm is the selection order of values from the refined values sets. The strategy adopted in GAM was selecting the value closest to $h^s_{\text{mid}}[i]$ first, which was phrased in the algorithm in Figure 2.4. This strategy will be termed as MID_VAL_FIRST from now on. Another selection order, which will enable the algorithm to find the minimum solution faster, could be to select the values having the minimum cost first, where the cost is the number of SPT terms. This can be done by first constructing the refined value set V_i^s by selecting the first L values out of V_i according to their closeness to the middle value $h^s_{\text{mid}}[i]$. And then, the values in V_i^s are ordered in ascending according to their number of SPT terms. The modified algorithm is shown in Figure 3.2 where p_k denotes the number of SPT terms in value v_k.

The new strategy will be termed as MIN_SPT_FIRST. To our observations, MIN_SPT_FIRST should not be used unless the coefficient wordlength is greater than the minimum possible wordlength.

3.2.1. SPTGAM Example

A linear-phase low-pass FIR filter with 10 taps is to be designed, i.e. N=10, M=5. The coefficient wordlength is seven (B=7). The desired filter characteristics are given in Table 3.1. The boundary values of the coefficients are found by the algorithm in Figure 2.2. The value sets ($\bigcup_{m=0}^{M-1} V_m$) formed using the boundary values are listed in Table 3.2.

Table 3.1. Filter characteristics of a low-pass FIR filter

Band (k)	Ω_k ($\times\pi$)	$D_k(\omega)$	δ_k
1	[0, 0.1]	1	0.05
2	[0.25, 1]	0	0.05

Table 3.2. Value sets of the coefficients for the SPTGAM example

V_0	=	{−3, −2, −1}
V_1	=	{−4, −3, −2, −1}
V_2	=	{1, 2, 3, 4}
V_3	=	{12, 13}
V_4	=	{20, 21, 22}

The feasible coefficient values are given as integers for ease of demonstration. The actual values can be found by dividing the values with 2^6. The refined values set sizes are limited to 2, i.e. $L = 2$. The SPT term prediction strategy is U_MIN. The value selection strategy is MID_VAL_FIRST.

The algorithm starts by finding the boundary values of the first coefficient, $h[0]$. The values found are $h^s_{\min}[0] = -3.62$ and $h^s_{\max}[0] = -0.83$, from which the middle value is found as $h^s_{\text{mid}}[0] = -2.24$. The values in V_0 within the boundary values are {−3,−2,−1}. Since, the refined value set size is limited to 2, the two values that are closest to $h^s_{\text{mid}}[0]$ are selected, which makes $V_0^s = \{-2,-3\}$. The value closest to $h^s_{\text{mid}}[0] = -2.24$ in $V_0^s = \{-2,-3\}$ is −2, so $h[0] = -2$, $H^* = \{-2\}$ and −2 is removed from the refined value set making $V_0^s = \{-3\}$. The algorithm branches to the next coefficient, $h[1]$. The refined boundary values are found to be $h^s_{\min}[1] = -4.38$ and $h^s_{\max}[1] = -1.26$ making $h^s_{\text{mid}}[1] = -2.82$. The refined value set is $V_1^s = \{-3,-2\}$. $h[1]$ is set to −3 making $H^* = \{-2,-3\}$ and $V_1^s = \{-2\}$. The algorithm branches to the next coefficient, $h[2]$. The search goes on until all values in V_0^s are tried out.

The values of the variables for each iteration are given in Table 3.3. The status column in the table tells the current status of the problem. ‘S’ indicates that a solution is found. ‘C’ resembles a cutoff situation meaning that a better solution cannot be found going deeper in the search tree. ‘I’ denotes an infeasible situation. This is the case where an empty refined value set is encountered for the next coefficient after fixing the current coefficient. The search is also demonstrated as a search tree in Figure 3.3, where the leftmost branches are the first branched values. Coefficients are set to the values in the circles. The values in the circles correspond to the values of the coefficients written at the beginning of each row.

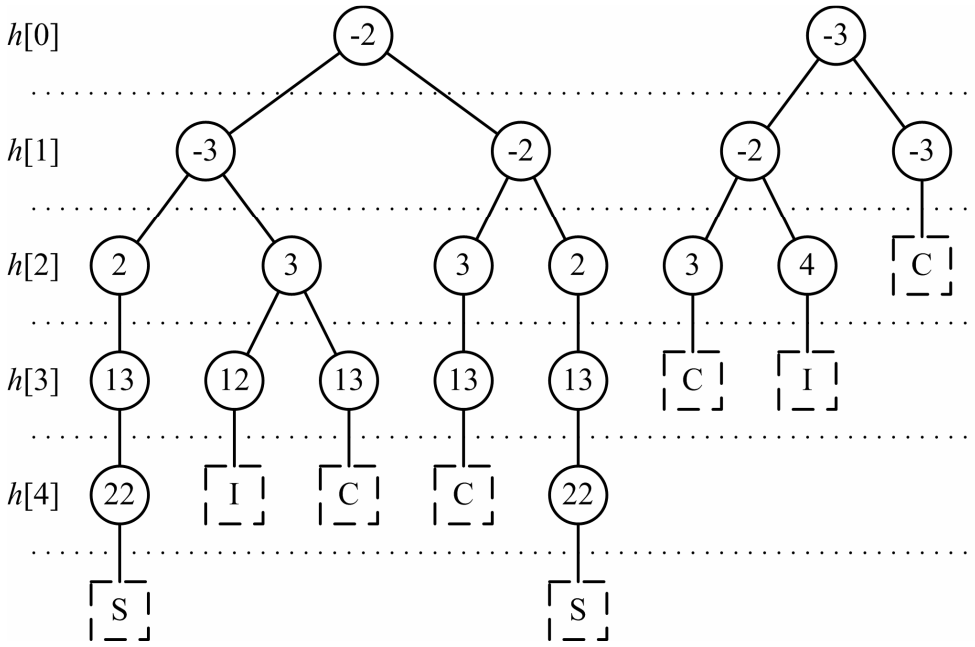

Figure 3.3. Search tree for SPTGAM example (S: Solution, I: Infeasible, C: Cut-off)

Table 3.3. The iterative solution of the SPTGAM example

I	$h^s_{\min}[i]$	$h^s_{\max}[i]$	$h^s_{\text{mid}}[i]$	V_i^s	$h[i]$	H^*	O	Status
0	-3.62	-0.83	-2.24	{-2,-3}				
				{-3}	-2	{-2}	∞	
1	-4.38	-1.26	-2.82	{-3,-2}				
				{-2}	-3	{-2,-3}		
2	1.01	3.57	2.29	{2,3}				
				{3}	2	{-2,-3,2}		
3	12.07	13.45	12.76	{13}				
				{}	13	{-2,-3,2,13}		
4	21.18	22.26	21.72	{22}				
				{}	22	{-2,-3,2,13,22}	10	S
3				{}		{-2,-3,2}		
2				{3}				
				{}	3	{-2,-3,3}		
3	11.97	13.15	12.56	{12,13}				
				{13}	12	{-2,-3,3,12}		
4	20.69	20.78	20.74	{}		{-2,-3,3,12}		I
3				{13}				
				{}	13	{-2,-3,3,13}		C
2				{}		{-2,-3}		
1				{-2}				
				{}	-2	{-2,-2}		
2	1.75	3.99	2.87	{3,2}				
				{2}	3	{-2,-2,3}		
3	12.03	13.66	12.85	{13}				
				{}	13	{-2,-2,3,13}		
4	20.69	21.37	21.03	{21}				
				{}	21	{-2,-2,3,13,21}		C
3				{}		{-2,-2,3}		
2				{2}				
				{}	2	{-2,-2,2}		
3	12.65	13.48	13.06	{13}				
				{}	13	{-2,-2,2,13}		
4	20.69	21.26	20.98	{21}				
				{}	21	{-2,-2,2,13,21}	9	S
3				{}		{-2,-2,2}		
2				{}		{-2,-2}		
1				{}		{-2}		
0				{-3}				
				{}	-3	{-3}		
1	-3.62	-1.28	-2.45	{-2,-3}				
				{-3}	-2	{-3,-2}		
2	2.65	4.01	3.33	{3,4}				
				{4}	3	{-3,-2,3}		C
				{}	4	{-3,-2,4}		
3	13.13	13.16	13.15	{}				I
2				{}				
1				{-3}				
				{}	-3	{-3,-3}		C
0				{}		{}		

3.3. Hardware Implementation Issues

In the hardware realization of constant coefficient FIR filters, adders/subtractors replacing the multipliers are called multiplier adders. The inter-tap adders are called structural adders [10]. One may choose among different adder topologies for the multiplier and structural adders. In terms of power and area the ripple carry adder seems to be a good choice. The delay elements can be realized with D-type flip-flops (D-FF).

The transposed form realization of an FIR filter, which is shown in Figure 3.4, is preferred because of its short critical path offering high speed operation. The critical path is 1 multiplier + 1 adder long, which is independent of the length of the filter, i.e. number of coefficients.

Two drawbacks of the transpose structure have been pointed out [30]. First, as the filter length increases the input signal bus becomes longer and has to be distributed to a larger number of tap inputs leading to large load capacitances. However, this problem can be alleviated by inserting data buffers on the input bus. Furthermore, by distributing the network in a tree-like structure, the capacitive loading can be reduced significantly.

The second drawback is that the registers used for the delays have to be larger in the transpose structure since they hold the accumulated sum instead of the input signal. This, in turn, increases the loading capacitance on the clock bus. To reduce the loading, again a tree structure similar to the data bus can be employed. The clock bus capacitance can be further reduced by carefully selecting the delay element structure to be used. For example,

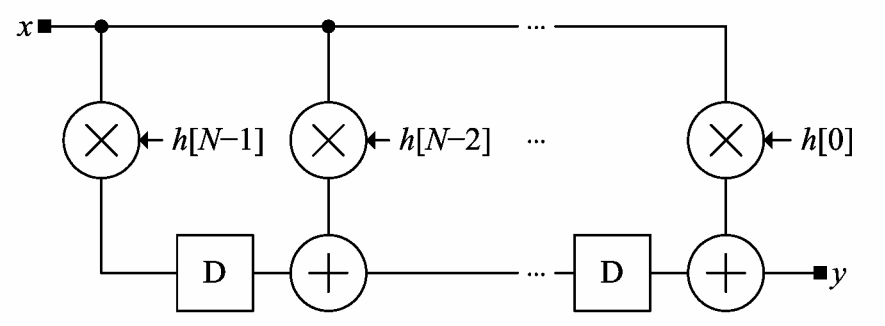

Figure 3.4. Transposed form realization of an FIR filter

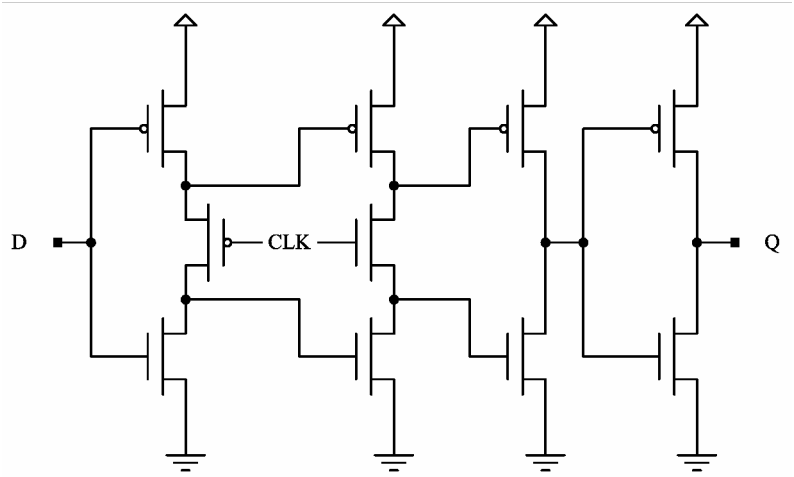

Figure 3.5. A TSPC D-FF with low capacitive loading to the clock bus

the true single-phase clocked (TSPC) D-FF proposed in [48] not only removes the need for an inverse clock (i.e. a single clock is used) but also reduces the capacitive load on the clock bus. A TSPC D-FF is shown in Figure 3.5.

When a coefficient multiplier is replaced with an adder tree, as it is shown in Figure 3.6, the number of adders and the depth of the adder tree depend on the number of SPT terms in the coefficient. Then, letting the number of SPT terms in coefficient $h[i]$ to be P_i, the depth of the adder tree used to replace the multiplier will be at most P_i-1 adders. If the maximum number of SPT terms per coefficient is $P_{\max}$, then the contribution of a replaced multiplier to the critical path will be at most $P_{\max}-1$ adders. Then for the transposed form realization of an FIR filter the critical path including the structural adder consists of $P_{\max}$ adders. Thus, by limiting the number of SPT terms per coefficient, one can limit the critical path length, and hence increase the performance of the filter. This formulation, however, neglects the effect of the structure of the adders. The length of the path is affected by the structure of the adders.

Here, an analysis for ripple carry adders is given. Ripple carry adders are composed of full adders. The critical path is the path for the carry signal generated by the least significant input bits to propagate to the most significant output sum. The contribution of an adder to the critical path can be approximated to be equal to the adder length of full adders. For example an 8-bit ripple carry adder will consist of eight full adders, and hence will have an eight full adder long critical path for the carry signal to travel to the most

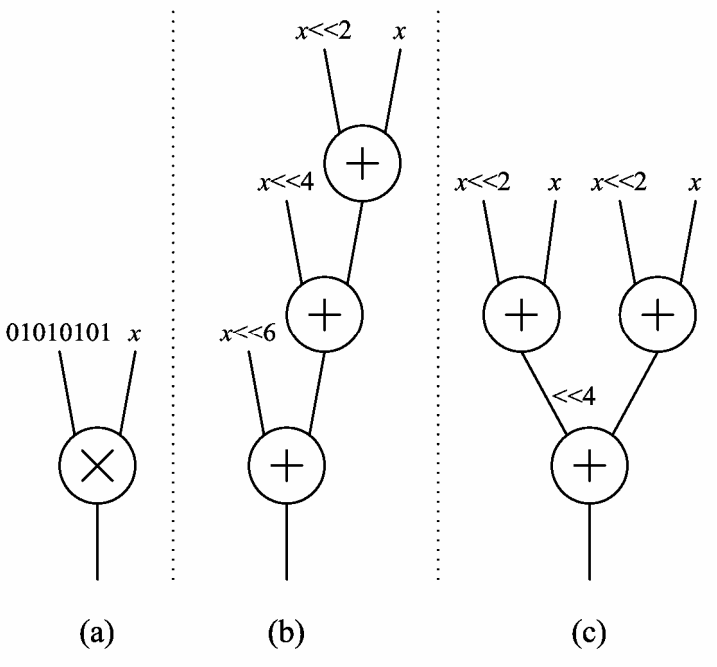

Figure 3.6. Constant coefficient multiplication (a) replaced by an adder tree (b) without considering depth and (c) with reduced depth

significant output sum. For an FIR filter, the maximum size of an adder, which is at the same time the output wordlength, can be calculated by the following formula:

$$W_{\max} = W_{in} + B - 1 + \left\lceil \log_2 \sum_{n=0}^{N-1} |h[n]| \right\rceil \tag{3.15}$$

where W_{in} is the input data wordlength (including sign bit), $h[n]$'s are filter coefficients $-1 \le h[n] < 1$, N is number of coefficients in the filter, and B is wordlength of the coefficients (including sign bit). $W_{\max}$ is the minimum adder size that ensures that no overflow will occur. This size is a limiting size for structural adders. The size of the multiplier adders will always be less than $W_{\max}$.

The critical path of an FIR filter having at most $P_{\max}$ SPT terms in a coefficient will be

$$\mathrm{CP} = W_{\max} + P_{\max} - 1 \tag{3.16}$$

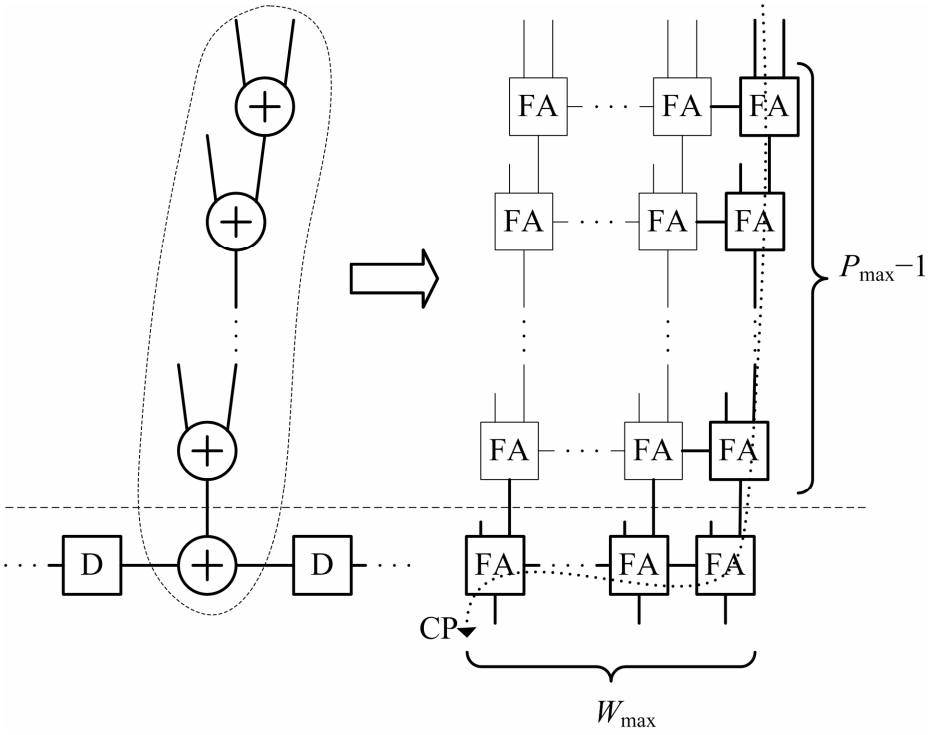

Figure 3.7. Critical path in an FIR filter

full adders long. This is also shown in Figure 3.7 From (3.16), it is clear that the critical path does not depend only on the maximum number of SPT terms ($P_{\max}$) a coefficient can have but also on the wordlength (B) of the coefficients. Therefore, keeping the wordlength small is an alternative method for critical path length minimization. Furthermore, when the multiplier adder tree depth can be kept minimum, i.e. the depth is $\lceil \log_2 P_{\max} \rceil$ adders, reducing the wordlength is even a more effective way (linear vs. logarithmic dependence).

3.3.1. Register and Adder Width Calculation

The filters designed by the SPTGAM algorithm are further optimized by the subexpression elimination tool of [8]. This tool generates a netlist of the filter in transposed form which is composed of adders and registers. Before the netlist is converted to structural level VHDL code to be mapped on a desired technology, the optimum width of the adders and registers is calculated. The calculation process will be explained on an example.

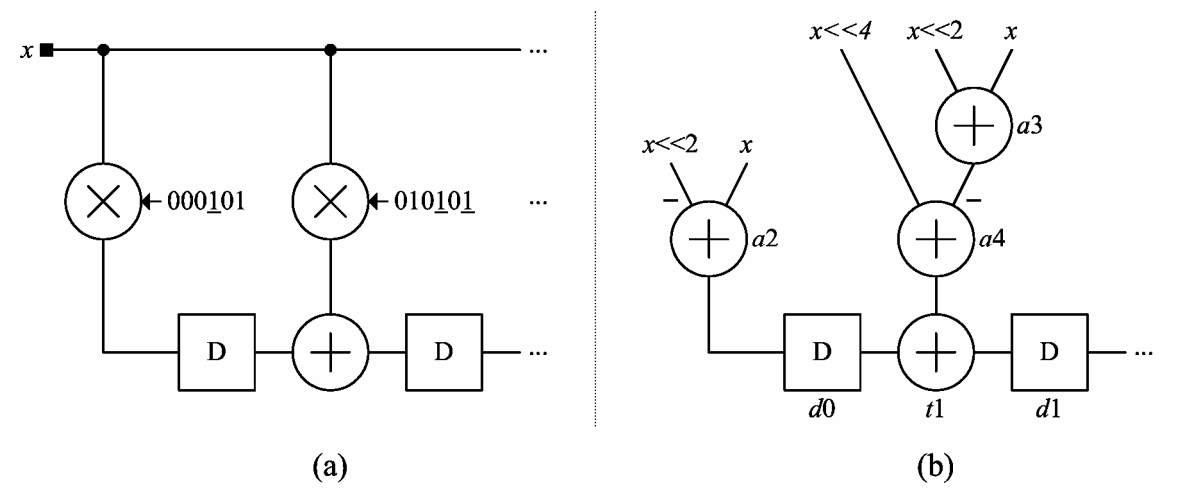

Figure 3.8. Hardware realization of an FIR filter (a) with multipliers and (b) multiplierless

In Figure 3.8 the hardware realization of the first two coefficients h[0] = 000101 and h[1] = 010101 of an FIR filter is shown. Here, x is the input, $a2$, $a3$, and $a4$ are coefficient adders, $t1$ is a structural adder, and $d0$ and $d1$ are delay elements. Note that $a2$ and $a4$ are in fact subtractors. To find the width of each element we apply the following procedure:

Input (x) : This parameter is given a priori and for our case it is 8 bits, i.e. $W_x = 8$.

Multiplier adder ($a2$, $a3$, $a4$) : Assume that the input x is set to 1. Then calculate the output value of the adders. Adders having an input from another adder will use the output value of that adder. Here $a4$ is such an adder. Then, the adder width is found by summing the input width and the smallest integer greater than the base two logarithm of the absolute value of the adder. For the above adders we can write down the formulae as follows:

$$W_{a2} = W_x + \lceil \log_2 | x - x<<2 | \rceil = 8 + \lceil \log_2 | 1-4 | \rceil = 8 + \lceil \log_2 3 \rceil = 10$$

$$W_{a3} = W_x + \lceil \log_2 | x + x<<2 | \rceil = 8 + \lceil \log_2 | 1+4 | \rceil = 8 + \lceil \log_2 5 \rceil = 11$$

$$W_{a4} = W_x + \lceil \log_2 | x<<4 - a3 | \rceil = 8 + \lceil \log_2 | 16-5 | \rceil = 8 + \lceil \log_2 11 \rceil = 12$$

Structural Adders ($t1$) : The method to find the width of a structural adder is the same as for a multiplier adder. The only difference is in finding the output value. For a structural adder (even it is a subtractor) the output value is found by adding the absolute value of the inputs. The value of a delay element is equal to the value of the input of the element. For

example, here, the value of $d0$ is equal to the value of $a2$ which is -3. The width formula for $t1$ is

$$W_{t1} = W_x + \lceil \log_2(|a2| + |a4|) \rceil = 8 + \lceil \log_2(|-3| + |11|) \rceil = 8 + \lceil \log_2 14 \rceil = 12$$

Registers ($d0$, $d1$) : The width of the registers is equal to the width of the element at its input. This can be a multiplier adder, a structural adder, another register or the input. For example, the width of register $d0$ is equal to the width the multiplier adder at its input, namely $a2$. So,

$$W_{d0} = W_{a2} = 10$$

Similarly, the width of register $d1$ is equal to the width the structural adder at its input

$$W_{d1} = W_{t1} = 12$$

Here the term width of an element is the output width of that element, i.e. the number of bit lines coming out of that element. For the delay elements this directly corresponds to the number of single bit D-type flip-flops. That is, if a delay element has a width of W then it will possess W D-type flip-flops. Although we can use the same convention for the adders and say that an adder having an output width W will use W full adder cells (in case it is realized with a ripple carry adder structure), this will introduce hardware overhead when we have shifted inputs. As an example, coefficient adder $a3$ will be analyzed. One of the inputs is left shifted by two. This means two zeros are added to the right of the input. Assuming x is 8 bits, $a3$ will do the following addition operation:

$$\begin{array}{cccccccccc} & x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 & 0 & 0 \\ + & x_7 & x_7 & x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 \\ \hline \end{array}$$

As can be seen from this operation, there is no need to use full adders for the least significant two bits due to the zero inputs. One can save up full adders as the amount of

shifting at the inputs. However, one should be careful when applying this procedure to subtractors. It should be applied to subtractors only if the negated input is the shifted input. Otherwise, the output needs to be negated introducing additional inverters that may cancel out the hardware gain obtained by removing the extra full-adders.

3.4. Experimental Results

In this section several example filters are designed using SPTGAM and compared to other methods from the literature. The methods used for comparison are the Remez algorithm, trellis algorithm [18], Li's algorithm [17], Lim's algorithm [20], Samueli's algorithm [14], Yao's algorithm [9,15], and MILP based optimum method of [12]. For the methods the abbreviations RMZ, TRE, LI, LIM, SAM, and PMILP are used respectively. For the Remez algorithm, MATLAB's remez function is used and the filter coefficients are quantized to the minimum wordlength satisfying the filter characteristics. The results for LIM, SAM, and PMILP are taken directly from the papers they appeared. The algorithms TRE, LI, and GAM are implemented in the C programming language. SPTGAM uses the linear programming (LP) library of QSOPT [47]. For the optimum MILP method, ILOG CPLEX [49] optimization package is used. The frequency grid on which the frequency response is evaluated consists of 1000 frequency samples for all filters. This makes 2000 constraints for the LP problem solved in SPTGAM. The implemented algorithms are run on a PC with a Pentium D 2.8GHz processor and 1 GB RAM. The hardware realizations of the filters are done using the transposed form structure. After optimization of the coefficients, for all filters the sub-expression sharing method in [8] is used. The input data width of the filters is taken as 8 bits. Adders are realized with ripple carry adders. Delay elements are realized with D-type flip flops. The filters are mapped to the AMS 0.35µ technology cell library for power simulations. The simulations are done with an event driven gate-level simulator. The operating voltage is 3.3 V. The input data applied to the filters is a random sequence of 16384 8-bit samples.

3.4.1. Example 1

In this example filters L1, L2, and S2 are designed. The filter frequency response characteristics were already given in Table 2.3. L1 and L2 are the example filters 1 and 2

given in [20]. S2 is the second example filter of [14]. The methods used for comparison are RMZ, LIM [20], SAM [14], and the PMILP algorithm [9,15].

The properties of the designed filters are given in Table 3.4. Here, N is the number of taps, B is the wordlength (including sign bit) of the coefficients, B_e is the effective wordlength excluding the sign bit and most significant zero bits [20], P is the maximum number of SPT terms found in each coefficient. The number of adders is given in terms of multiplier adders (MA) and structural adders (SA). The number of multiplier adders is the number after sub-expression elimination is applied. The total hardware complexity after realizing the adders with ripple carry adders and the delay elements with D-type flip-flops, is the sum of the number of full-adders (FA) and number of D-type flip-flops (D-FF). The power performance of the filters is given as μW/MHz. The SPT term and power gains are calculated taking the RMZ filters as reference.

Table 3.4. Designed filter properties of example 1

Filter	Method	N	B	B_e	P	SPT term		Hardware Complexity			Power	
						#	Gain (%)	FA	DFF	Total	μW/MHz	Gain (%)
L1	RMZ	121	19	16	8	483	-	3647	2907	6554	1090	-
	LIM	121	17	14	7	416	13.9	3146	2664	5810	938	13.9
	PMILP	121	-	15	6	442	8.5	3248	2748	5996	988	9.4
	GAM	121	17	14	6	362	25.1	2886	2654	5540	864	20.7
L2	RMZ	63	15	12	5	197	-	1510	1281	2791	422	-
	LIM	63	13	10	4	159	19.3	1311	1154	2465	349	17.3
	PMILP	63	-	11	4	163	17.3	1285	1182	2467	344	18.4
	GAM	63	13	10	4	140	28.9	1213	1154	2367	328	22.3
S2	RMZ	60	16	12	6	204	-	1494	1251	2745	421	-
	SAM	60	-	13	4	174	14.7	1455	1298	2753	402	4.5
	PMILP	60	-	12	4	174	14.7	1342	1255	2597	368	12.6
	GAM	60	15	11	5	160	21.6	1337	1180	2517	362	14.0

Table 3.5. Solution times for SPTGAM filters of example 1

Filter	First solution		Best solution		Total search time
	# of SPT	Time	# of SPT	Time	
L1	419	2m	362	32m	56m
L2	157	15s	140	26m	54m
S2	182	12s	160	23m	27m

It is clear from Table 3.4 that SPTGAM outperforms all methods in terms of the number of SPT terms. In terms of wordlength SPTGAM coefficients have either shorter or equal wordlength when compared to the other methods. The reduced SPT term count and wordlength has reduced the hardware cost and power consumption. Again looking at Table 3.4, SPTGAM filters have the least hardware cost in terms of full adders and D-FF. Note that the reduction in the number of SPT terms (taking the RMZ coefficients as reference) for the SPTGAM filters is in parallel with the reduction in power consumption.

The time spent to find the filters for SPTGAM is given in Table 3.5. The feasible value set size L for SPTGAM was set to 2. The number of SPT terms per coefficient was not limited. The SPT term prediction and value selection strategies used were U_SUB_AVG and MID_VAL_FIRST respectively. The first solution is the initial solution found by GAM. The best solution is best in terms of both the total number of SPT terms of the coefficients and maximum number of SPT terms in a coefficient. Since the compared algorithms were not implemented, a direct comparison to their search time was not possible.

As an example, the coefficients and frequency response of SPTGAM filter L1 are given in Table 3.6 and Figure 3.9 respectively.

Table 3.6. Coefficients of SPTGAM filter L1 with N=121, B=17

$h[0] = 2^{-14}$
$h[1] = -2^{-13} - 2^{-15}$
$h[2] = 2^{-12} + 2^{-16}$
$h[3] = -2^{-12} - 2^{-14} - 2^{-16}$
$h[4] = 2^{-12} + 2^{-15}$
$h[5] = -2^{-14} - 2^{-16}$
$h[6] = -2^{-12}$
$h[7] = 2^{-11} + 2^{-14} + 2^{-16}$
$h[8] = -2^{-10} + 2^{-12}$
$h[9] = 2^{-11} + 2^{-13}$
$h[10] = -2^{-13} - 2^{-15}$
$h[11] = -2^{-11} - 2^{-15}$
$h[12] = 2^{-10} + 2^{-13} + 2^{-15}$
$h[13] = -2^{-9} + 2^{-11} + 2^{-13} - 2^{-16}$
$h[14] = 2^{-10} + 2^{-15}$
$h[15] = -2^{-13} + 2^{-16}$
$h[16] = -2^{-10} - 2^{-15}$
$h[17] = 2^{-9} - 2^{-13} - 2^{-16}$
$h[18] = -2^{-9} + 2^{-13} - 2^{-15}$
$h[19] = 2^{-10} - 2^{-15}$
$h[20] = 2^{-11} + 2^{-13} + 2^{-15}$
$h[21] = -2^{-9} - 2^{-12}$
$h[22] = 2^{-8} - 2^{-10}$
$h[23] = -2^{-9} - 2^{-11} + 2^{-13} + 2^{-16}$
$h[24] = 2^{-11} - 2^{-13} - 2^{-16}$
$h[25] = 2^{-9} + 2^{-12} - 2^{-14}$
$h[26] = -2^{-8} - 2^{-13} + 2^{-15}$
$h[27] = 2^{-8} + 2^{-12}$
$h[28] = -2^{-9} - 2^{-12} - 2^{-15}$
$h[29] = -2^{-10} - 2^{-12} + 2^{-14}$
$h[30] = 2^{-8} + 2^{-11} + 2^{-13}$
$h[31] = -2^{-7} + 2^{-9} - 2^{-12}$
$h[32] = 2^{-8} + 2^{-10} - 2^{-14}$
$h[33] = -2^{-10} + 2^{-13}$
$h[34] = -2^{-8} - 2^{-12} - 2^{-14}$
$h[35] = 2^{-7} + 2^{-13}$
$h[36] = -2^{-7} - 2^{-11} + 2^{-13} - 2^{-16}$
$h[37] = 2^{-8} + 2^{-11} - 2^{-13} + 2^{-16}$
$h[38] = 2^{-9} + 2^{-11} + 2^{-16}$
$h[39] = -2^{-7} - 2^{-10} - 2^{-12}$
$h[40] = 2^{-6} - 2^{-8} + 2^{-12} + 2^{-14}$
$h[41] = -2^{-7} - 2^{-9} + 2^{-11}$
$h[42] = 2^{-10} + 2^{-12}$
$h[43] = 2^{-7} + 2^{-10} + 2^{-13}$
$h[44] = -2^{-6} - 2^{-11} - 2^{-13} - 2^{-15}$
$h[45] = 2^{-6} + 2^{-10} - 2^{-14}$
$h[46] = -2^{-7} - 2^{-11} + 2^{-14} - 2^{-16}$
$h[47] = -2^{-7} + 2^{-9} - 2^{-13}$
$h[48] = 2^{-6} + 2^{-8} + 2^{-11}$
$h[49] = -2^{-5} + 2^{-8} + 2^{-10} - 2^{-13} - 2^{-15}$
$h[50] = 2^{-6} + 2^{-8} + 2^{-10} - 2^{-13}$
$h[51] = -2^{-9} + 2^{-12}$
$h[52] = -2^{-5} + 2^{-7} + 2^{-12} + 2^{-14}$
$h[53] = 2^{-4} - 2^{-6} - 2^{-8}$
$h[54] = -2^{-4} + 2^{-6} + 2^{-10} + 2^{-12} - 2^{-16}$
$h[55] = 2^{-5} - 2^{-7} - 2^{-11} + 2^{-13}$
$h[56] = 2^{-5} - 2^{-7} + 2^{-9} - 2^{-12} + 2^{-14}$
$h[57] = -2^{-3} + 2^{-5} + 2^{-8} - 2^{-11} - 2^{-16}$
$h[58] = 2^{-3} + 2^{-5} + 2^{-12}$
$h[59] = -2^{-2} + 2^{-4} - 2^{-6} - 2^{-9} - 2^{-11} - 2^{-13}$
$h[60] = 2^{-2} - 2^{-5} + 2^{-8} + 2^{-10} + 2^{-12} - 2^{-16}$

$h[n] = h[120-n]$ for n=61, 62, 63, … ,120

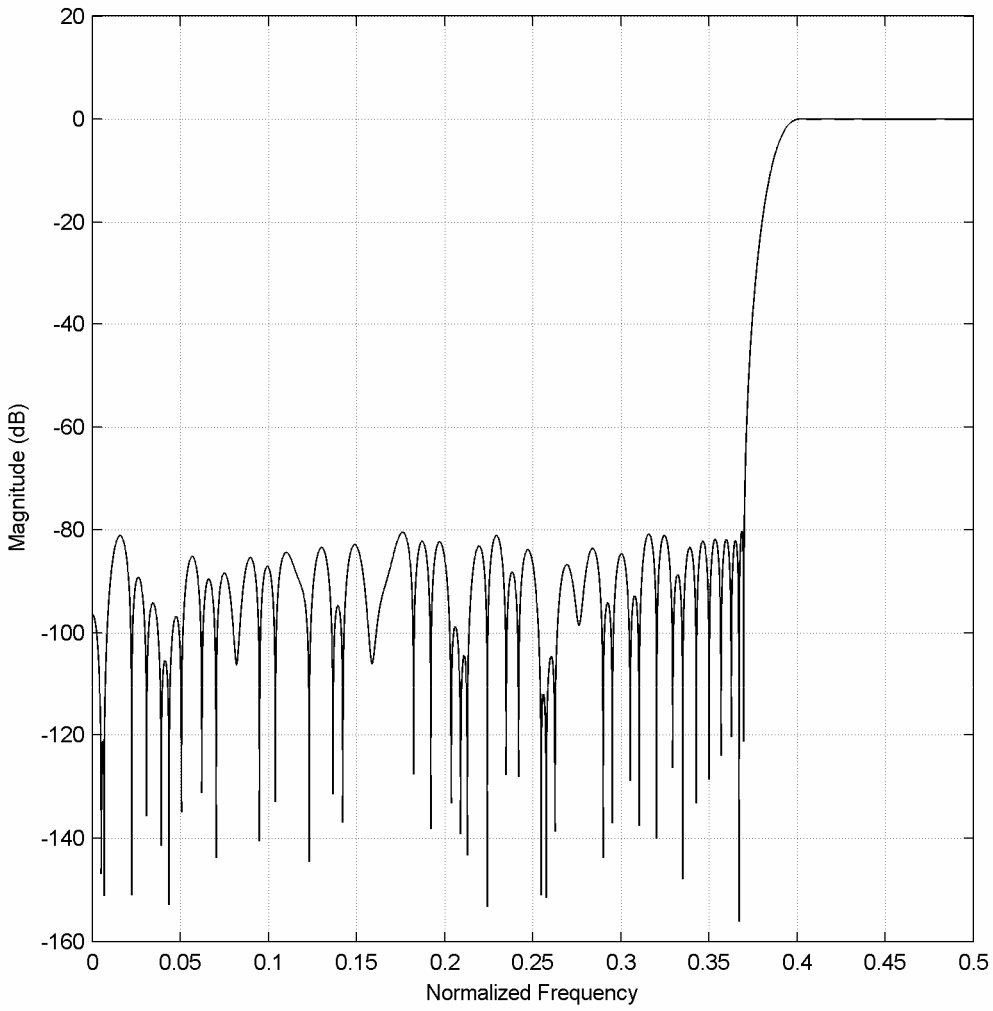

Figure 3.9. Frequency response of SPTGAM filter L1 with N=121, B=17

3.4.2. Example 2

In this example SPTGAM is compared to the algorithms TRE, LI, and MILP. The algorithms are implemented in the C programming language. The comparison is made with the filters A, B, and C of Table 2.3.

The properties of the designed filters are given in Table 3.7. The run times of the algorithms are listed in Table 3.8. For the MILP method passband scaling could be used, however, to make a direct comparison with the results of SPTGAM the passband gain is taken to be unity, i.e. $s = 1$ in [12]. LI and TRE directly make use of passband scaling in

Table 3.7. Designed filter properties of example 2

Filter	Method	N	B	B_e	P	SPT term		Hardware Complexity			Power	
						#	Gain (%)	FA	DFF	Total	µW/MHz	Gain (%)
A	RMZ	59	15	12	6	200	-	1449	1183	2632	412	-
	TRE	59	13	10	5	160	20.0	1263	1102	2365	340	17.4
	LI	59	13	10	4	151	24.5	1218	1097	2315	325	21.1
	MILP	59	13	10	5	145	27.5	1173	1063	2236	317	23.1
	GAM	59	13	10	5	145	27.5	1173	1063	2236	317	23.1
B	RMZ	105	13	10	5	232	-	2007	1958	3965	549	-
	TRE	105	12	9	3	199	14.2	1814	1836	3650	464	15.0
	LI	105	12	9	4	212	8.6	1853	1843	3696	487	11.3
	MILP	-	-	-	-	-	-	-	-	-	-	-
	GAM	105	11	8	4	169	27.2	1672	1739	3411	426	22.4
C	RMZ	325	15	12	5	820	-	7045	6818	13863	1884	-
	TRE	325	14	11	5	743	9.4	6650	6665	13315	1721	8.7
	LI	325	14	11	6	740	9.8	6608	6618	13226	1712	9.1
	MILP	-	-	-	-	-	-	-	-	-	-	-
	GAM	325	13	10	4	549	33.0	5687	6148	11835	1450	23.0

Table 3.8. Time spent by each algorithm to design the filters of example 2

Filter	Method	First solution		Best solution		Total search time
		# of SPT	Time	# of SPT	Time	
A	RMZ	200	-	200	-	-
	TRE	160	38m	160	38m	2h 57m
	LI	151	1s	151	1s	4s
	MILP		13 days	145	14 days	15 days
	GAM	153	17m	145	3h 2m	4h 14m
B	RMZ	232	-	232	-	-
	TRE	215	48m	199	3h 19m	4h 32m
	LI	221	1s	212	1s	3s
	MILP	-	-	-	-	24h
	GAM	179	1m	169	9m	24h
C	RMZ	820	-	820	-	-
	TRE	767	55m	743	20h 46m	24h
	LI	818	1s	740	18s	41s
	MILP	-	-	-	-	24h
	GAM	583	18m	549	13h 47m	24h

their optimization. LI searches for a range of scaling factors to find the optimum coefficients in terms of the number of SPT terms. For TRE, no method is given in [18] to determine a proper scaling factor. Since TRE is a polynomial time algorithm, multiple runs of the algorithm on different scaling factors are possible. Therefore, TRE is run for the same range of scaling factors used by LI. The run time of the algorithm TRE given in Table 3.8 is the result of multiple runs of the algorithm. The algorithms are run for at most 24 hours. The only exception is made for MILP when run on filter A. The reason is that filter A is the smallest filter and is the one that is most likely to produce a result in a reasonable amount of time.

In terms of search time LI seems to be the best algorithm. However, for large filters the gain in the number of SPT terms obtained by SPTGAM compensates the search time. Note that even the initial solutions obtained by SPTGAM are much better than the best solutions obtained by LI. It is interesting to note that the best solution found for filter A by GAM is the optimum solution. The complete search took only four hours for SPTGAM, whereas it took 15 days for the optimal MILP method. Hence, SPTGAM is almost 100 times faster than the optimal MILP method. For filters B and C, the MILP method could not find any solutions in one day.

3.4.3. Example 3

For the same frequency response characteristics of the filters of example 2, filters are re-designed with different number of taps using SPTGAM. The number of taps is increased starting from the minimum possible to the number beyond no reduction in the coefficient

Table 3.9. Designed filter properties of example 3

Filter	N	B	B_e	P	SPT term		Hardware Complexity			Power	
					#	Gain (%)	FA	DFF	Total	µW/MHz	Gain (%)
A	56	19	16	6	244	-	1708	1330	3038	412	-
	57	15	12	5	159	34.8	1264	1138	2402	340	32.0
	58	14	11	5	144	40.9	1190	1097	2287	315	39.2
	59	13	10	5	145	40.6	1173	1063	2236	317	38.8
B	101	14	11	4	235	-	2017	1985	4002	549	-
	102	13	10	4	206	12.3	1793	1893	3686	464	13.9
	103	12	9	4	189	19.6	1806	1813	3619	487	14.6
	105	11	8	4	169	28.1	1672	1739	3411	426	22.1
C	311	16	13	5	771	-	6771	6840	13611	1884	-
	312	15	12	5	674	12.6	6259	6547	12806	1721	8.7
	317	14	11	4	660	14.4	6219	6333	12552	1712	11.8
	325	13	10	4	549	28.8	5687	6148	11835	1450	20.1

wordlength could be achieved. The properties of the filters are given in Table 3.9. Power gain is calculated taking the filter having the minimum number of taps as reference.

An additional tap requires an additional structural adder and delay element. Therefore, increasing the number of taps, at first glance, might seem to increase the hardware cost. However, this increase is compensated by the reduction of the wordlength of the coefficients, which can be seen from Table 3.9.

3.5. Summary

An algorithm for designing low power/hardware cost linear-phase FIR filters was presented. The algorithm optimizes SPT terms in the coefficients given the filter frequency response characteristics. Although the worst case run time of the algorithm is exponential, its capability to find appreciably good solutions in a reasonable amount of time (at least 100 times faster than traditional optimum MILP based formulation) makes it a desirable CAD tool for designing low power/hardware cost linear phase FIR filters. The algorithm is compared to existing methods with several examples. The filters found by the proposed algorithm have fewer SPT terms and are shorter in wordlength than the filters found by the other methods. SPTGAM filters consume 20 per cent less power on average than unoptimized RMZ coefficients. The superiority over existing methods has been shown to be more apparent on high-order filters.

4. LOW-POWER DESIGN OF VARIABLE COEFFICIENT FIR FILTERS

In a generic DSP implementation of an FIR filter, the coefficients are held in a coefficient memory and are sequentially applied to the multiply accumulate (MAC) unit. A separate memory holds data, which is applied to the second input of the unit. The major source of power dissipation is the multiplier that computes $h[i]\cdot x[n-i]$ for $i = 0,1,\ldots,N-1$. In a typical multiplier unit, each 1-bit of a multiplicand corresponds to a shift and add operation. If the number of 1-bits can be reduced in the multiplier, we can reduce the number of additions required to compute the product $h[i]\cdot x[n-i]$, thereby reducing power.

A commonly used multiplier unit employs the Booth's algorithm for high-speed multiplication. The main idea is to recode the multiplier such that consecutive runs of 1-bits are represented by difference of two numbers each having only a single 1-bit. For example, the sequence 01111 can be represented as $10000 - 00001 = 1000\underline{1}$ which uses an add and a subtract operation in calculating a product with this number. In contrast, the original multiplier would have caused 4 add operations. Hence, with the recoded multiplier power can be saved.

Power dissipation in a multiplier is a function of the signal activity at the external and internal nodes. Booth encoding as mentioned above will reduce the number of computations to yield the product but cannot guarantee a reduced switching activity at the internal/external nodes, which is the main cause of power dissipation. The signal activity at the external nodes depends on the Hamming distance between successively applied inputs. The data part cannot be controlled but the coefficient part can. This can either be achieved by reordering the coefficients [24] so that Hamming distance between successive coefficients is minimized or by using coefficients already having low Hamming distance values between successive coefficients.

In our work, we converted the low-power FIR filter coefficient synthesis problem to a problem to find low switching activity (Hamming distance) and number of ones

coefficients which is then formulated as an integer quadratic programming problem. Moreover, since today's processors may possess multiple MAC units, the formulation proposed is extended to handle this situation. The resulting coefficients are optimum in terms of switching activity and number of ones for the desired number of multiplier units. A couple of example filters are designed and the power performances are tested on a pre-designed multiply-accumulate (MAC) unit. The effectiveness of our approach is also shown on a processor having multiple MAC units.

The integer programming based formulation is restricted to small filters (with length $N < 40$) due to its excessive run-time requirements. To alleviate this problem, a modified version of the GAM algorithm that minimizes switching activity, namely SWAGAM, is also proposed.

4.1. Problem Formulation

The coefficients $h[i]$ of a linear–phase FIR filter with impulse response can be written in B-bit two's complement representation as

$$h[i] = -x_{i,0} + \sum_{j=1}^{B-1} x_{i,j} 2^{-j} \tag{4.1}$$

where $x_{i,j} \in \{0,1\}$.

4.1.1. Formulation of the Cost of Switching Activity (Hamming Distance)

Formulation of the switching activity between successively applied coefficients is done as follows: A switching between coefficients $h[i]$ and $h[i+1]$, which are quantized according to (4.1), at bit j is said to occur when Boolean XOR of the two bits evaluates to a one. The arithmetic expression for the Boolean XOR operation is

$$\begin{aligned} x_{i,j} \oplus x_{i+1,j} &\equiv x^2_{i,j} + x^2_{i+1,j} - 2\, x_{i,j}\, x_{i+1,j} \\ &\equiv (x_{i,j} - x_{i+1,j})^2 \end{aligned} \tag{4.2}$$

Having defined the cost function for the switching activity between two bits, the cost of switching from coefficient $h[i]$ to $h[i+1]$ is given by

$$\sum_{j=0}^{B-1}\left(x_{i,j}-x_{i+1,j}\right)^2 \tag{4.3}$$

where B is the coefficient wordlength. The total cost of switching of an FIR filter having N taps is

$$C_{swa}=\sum_{i=0}^{N-2}\sum_{j=0}^{B-1}\left(x_{i,j}-x_{i+1,j}\right)^2 \tag{4.4}$$

Since, for a linear-phase filter the coefficients are symmetric, the above cost function reduces to

$$C_{swa}=2\sum_{i=0}^{M-2}\sum_{j=0}^{B-1}\left(x_{i,j}-x_{i+1,j}\right)^2 \tag{4.5}$$

where M is calculated using (2.3).

In the case when there are more than one MAC units the coefficients are assumed to be applied in the following sequence: Assuming N taps and P MAC units, coefficients applied to a MAC unit are $h[i]$, $h[i+P]$, $h[i+2P]$, ⋯. An example is shown in Figure 4.1 for four MAC units and an FIR filter having 10 coefficients. The new formulation of switching activity of successively applied coefficients for P MAC units is

$$C_{swa}=\sum_{p=0}^{P-1}\sum_{i=0}^{K-1}\sum_{j=0}^{B-1}\left(x_{p+iP,j}-x_{p+(i+1)P,j}\right)^2 \tag{4.6}$$

where $K=\lfloor (N-1-p)/P \rfloor$.

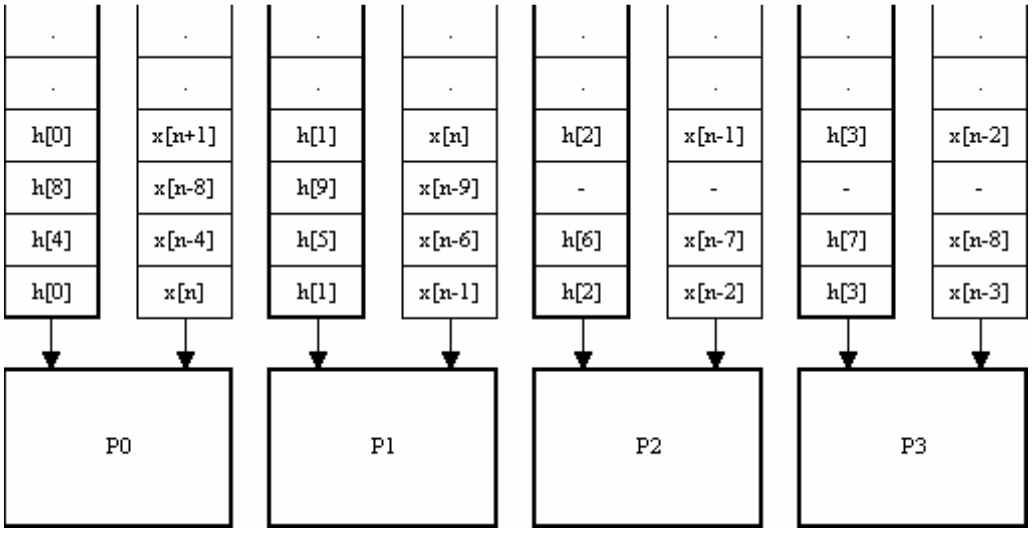

Figure 4.1. FIR filtering on multiple MAC units

4.1.2. Formulation of the Cost of Number of Ones

Formulation of the number of ones in the coefficients of an FIR filter is straightforward. For a linear-phase FIR filter having N taps and for which the coefficients are represented in two's complement notation using (4.1), the cost of number of ones in the coefficients can be expressed as

$$C_{one} = \sum_{i=0}^{N-1} \sum_{j=0}^{B-1} x_{i,j} \tag{4.7}$$

where $x_{i,j} \in \{0,1\}$. For a symmetric linear-phase FIR filter where the filter length N is odd (4.7) reduces to

$$C_{one} = 2 \sum_{i=0}^{M-2} \sum_{j=0}^{B-1} x_{i,j} + \sum_{j=0}^{B-1} x_{M-1,j} \tag{4.8}$$

and when N is even

$$C_{one} = 2 \sum_{i=0}^{M-1} \sum_{j=0}^{B-1} x_{i,j} \tag{4.9}$$

where M is calculated using (2.3).

4.1.3. Formulation of the Problem

Having defined the cost functions of switching activity and number of ones three sets of optimized coefficients can be obtained: minimum number of ones (MONE), minimum switching activity (MSWA) filters, and minimum switching activity and ones (MSWO) filters.

Using (2.4), (2.5), (4.1), and (4.9) the optimization problem for MONE filters can be written as

Minimize $$2\sum_{i=0}^{M-1}\sum_{j=0}^{B-1} x_{i,j}$$

Subject to

$$\left|\sum_{i=0}^{M-1}\left(-x_{i,0}+\sum_{j=1}^{B-1} x_{i,j}2^{-j}\right)T_i(\omega)-D(\omega)\right| \le \delta(\omega). \tag{4.10}$$

This problem can be solved optimally using MILP.

Similarly, using (2.4), (2.5), (4.1), and (4.6) the optimization problem for MSWA filters for a filter core having P MAC units can be written as

Minimize $$\sum_{p=0}^{P-1}\sum_{i=0}^{K-1}\sum_{j=0}^{B-1}\left(x_{p+iP,j}-x_{p+(i+1)P,j}\right)^2$$

Subject to

$$\left|\sum_{i=0}^{M-1}\left(-x_{i,0}+\sum_{j=1}^{B-1} x_{i,j}2^{-j}\right)T_i(\omega)-D(\omega)\right| \le \delta(\omega). \tag{4.11}$$

Due to the quadratic term in the objective function this problem can be formulated as an integer quadratic problem. By introducing new variables this problem can be converted to an integer linear programming problem as described in [26].

Again using (2.4), (2.5), (4.1), (4.6), and (4.9) the optimization problem for MSWO filters for a filter core having P MAC units can be written as

Minimize $$\sum_{p=0}^{P-1}\sum_{i=0}^{K-1}\sum_{j=0}^{B-1}\left(x_{p+iP,j}-x_{p+(i+1)P,j}\right)^2+\sum_{i=0}^{N-1}\sum_{j=0}^{B-1}x_{i,j}$$

Subject to

$$\left|\sum_{i=0}^{M-1}\left(-x_{i,0}+\sum_{j=1}^{B-1}x_{i,j}2^{-j}\right)T_i(\omega)-D(\omega)\right|\le\delta(\omega) \tag{4.12}$$

where the cost of switching activity is given the same weight as the cost for number of ones.

4.2. The GAM Algorithm for Switching Activity Minimization

The GAM algorithm can be easily modified to a search algorithm for finding a coefficient set with reduced switching activity. The modified algorithm which is named as SWAGAM is shown in Figure 4.2. SWAGAM iteratively finds coefficients with reduced switching activity given the wordlength B and filter length N. The resulting linear-phase FIR filter has zero-phase magnitude response as defined in (2.4), subject to the frequency response characteristics defined in (2.5).

Since the minimization of switching activity is defined as the goal of optimization, the cut-off mechanism of SWAGAM is re-defined as follows: The switching activity of the best solution is denoted as SC (which is set to infinity at the beginning since there is no solution). The switching activity cost of setting $h[i]$ to value v^* is denoted by p, which can be calculated by calculated by counting the different values digits of coefficient $h[i\text{-}1]$ and $h[i]$ using (4.3). The total switching activity between already fixed coefficients is denoted by F_i. The predicted minimum switching activity of unfixed coefficients is denoted by U_i. If $F_i + p + U_i \ge SC$ there is no need to further branch to the next coefficient since there does not exist a better solution then the current solution. In this case, the algorithm proceeds by setting the current coefficient $h[i]$ to its next feasible value v^*.

```
SWAGAM (M , L, ⋃_{m=0}^{M−1} V_m )
  H* = ∅;
  SC = ∞;
  Obtain U_i by computing (4.14) for i = 0,1,…,M−2;
  Obtain h^s_min[0] and h^s_max[0] by solving equations (2.10) and (2.11);
  V_0^s = VALUE_SELECT(L, h^s_min[0], h^s_max[0], V_0);
  F_0 = 0;
  i = 0;
  WHILE (i ≥ 0)
    IF (V_i^s ≠ ∅)
      h[i] = v* such that v* is the first element of ordered set V_i^s;
      p = switching activity between h[i−1] and h[i];
      H* = H* ∪ {h[i]};
      V_i^s = V_i^s − {v*};
      IF(F_i + p + U_i < SC)  /* If a solution with switching activity less than O exists*/
        IF (i < M − 1)
          i = i + 1;
          F_i = F_{i−1} + p;
          Obtain h^s_min[i] and h^s_max[i]  by solving equations (2.10) and (2.11);
          V_i^s = VALUE_SELECT(L, h^s_min[i], h^s_max[i], V_i);
        ELSE-IF (problem is feasible)
          H* is a solution to the problem;
          SC = switching activity of H*;
        END-IF
      END-IF
    ELSE
      H* = H* − {h[i]};
      i = i − 1;
    END-IF
  END
```

Figure 4.2. The SWAGAM algorithm for minimizing switching activity between successive coefficients

The predicted switching activity of the unfixed coefficients (U_i.) is found as follows: First, the average cost of switching activity between all values of coefficient $h[i]$ and $h[i+1]$ is found by

$$S_{i,\text{avg}} = \frac{\sum_{j=1}^{|V_i|}\sum_{k=1}^{|V_{i+1}|} s_{v_j \to v_k}}{|V_i||V_{i+1}|} \tag{4.13}$$

where $s_{v_j \to v_k}$ is the switching activity cost of a transition from value $v_j \in V_i$ to $v_k \in V_{i+1}$. Then

$$U_i = \sum_{m=i+1}^{M-2} \lfloor S_{m,\text{avg}} \rfloor \tag{4.14}$$

4.3. Design Examples

In this section the quadratic programming based formulation and SWAGAM are compared to the method of coefficient re-ordering on the test filters [23] given in Table 4.1.

Table 4.2 shows the results for the amount of switching activity and the number of ones in the coefficients generated using six methods, namely NOPT, RORD, MONE, MSWA, MSWO, and SWAGAM. NOPT coefficients are the coefficients generated using MATLAB's Remez function and quantized to 16 bits. They are the reference coefficients for which no optimization is done. RORD coefficients are NOPT coefficients re-ordered by the method given in [24] for minimum switching activity. MSWA coefficients are the coefficients optimized for minimum switching activity using the formulation in (4.11) with ILOG CPLEX integer programming tool. MONE coefficients are the coefficients

Table 4.1. Filter characteristics

Filter	N	K	$\Omega_k(\times\pi)$	$D_k(\omega)$	δ_k
M2	24	1	[0, 0.375]	1	0.0233
		2	[0.5625, 1]	0	0.0080
M4	28	1	[0, 0.3333]	1	0.0139
		2	[0. 5, 1]	0	0.0056
M5	34	1	[0, 0.3667]	1	0.0186
		2	[0.5167, 1]	0	0.0035
M6	29	1	[0, 0.4]	1	0.0580
		2	[0.6, 1]	0	0.0100

Table 4.2. Switching activity counts and number of ones in synthesized filters for one MAC unit

Filter	Method	Switching Activity	Number of ones	Optimization Time (Sec.)	Power (μW)	Power Reduction (%)
M2	NOPT	178	186	-	1317	-
	RORD	59	186	<1	1077	18.2
	MONE	94	88	20	890	32.4
	MSWA	82	150	800	982	25.4
	MSWO	90	90	340	850	35.4
	SWAGAM	90	92	120	852	35.3
M4	NOPT	188	244	-	1324	-
	RORD	68	244	<1	1083	18.2
	MONE	116	140	24	913	31.0
	MSWA	100	174	1200	957	27.7
	MSWO	102	144	80	874	34.0
	SWAGAM	116	174	10	1016	23.3
M5	NOPT	246	272	-	1321	-
	RORD	75	272	<1	1065	19.4
	MONE	148	140	300	978	26.0
	MSWA	122	290	13742	1137	13.9
	MSWO	140	146	5328	899	31.9
	SWAGAM	140	160	22	917	30.6
M6	NOPT	216	245	-	1249	-
	RORD	59	245	<1	993	20.5
	MONE	122	61	4	787	37.0
	MSWA	108	328	2240	1001	19.9
	MSWO	118	63	66	775	38.0
	SWAGAM	114	129	120	804	35.6

optimized for minimum number of ones using the formulation in (4.10) with ILOG CPLEX integer programming tool. MSWO coefficients are the coefficients optimized for both minimum number of ones and minimum switching activity using the formulation in (4.12) with ILOG CPLEX integer programming tool. The optimization problems were solved on a PC having INTEL P4 1.7GHz processor with 256 MB of RAM.

The power performance of the generated coefficients are tested on a single MAC unit having a 16 bit Booth encoded Wallace tree multiplier and 40-bit accumulator. The MAC

unit is synthesized with AMS 0.6μ technology cell library. FIR filtering is performed on 15625 samples of voice data quantized to 16 bits. Power simulations were done with an event driven gate-level simulator using a variable delay model which accounts for glitches. The operating frequency was taken to be 1MHz and supply voltage to be 5V. The resulting power dissipations are also given in Table 4.2.

The percentage power reduction is calculated by taking the NOPT coefficients' power as reference. The results indicate that by just reducing the switching activity between coefficients, the best power performance cannot be achieved. By reordering coefficients one can get 19 per cent reduction in power. MSWA coefficients could achieve a power reduction of 22 per cent on average. The best power performance is obtained from MSWO coefficients having a power reduction of 35 per cent on average. SWAGAM coefficients follow next with a 31 per cent power reduction on average. MONE coefficients have a comparable performance to MSWO coefficients with 30 per cent power reduction on average. When design time is important, which might be the case for filters having large number of coefficients, SWAGAM should be preferred.

Another set of coefficients were generated targeting a filter core having four MAC units. The coefficients are generated for filter B. The optimization method used is MSWO but now targeting 4 MAC units, i.e. P=4 in (4.12). The resulting coefficients' switching activity counts for each MAC unit are given in Table 4.3. The switching activity counts are compared to those coefficients generated using methods NOPT, and MSWO targeting one MAC unit.

The power performances of the coefficients are tested using the same MAC unit mentioned above. The operating frequency is 1MHz and supply voltage 5V. The resulting average power dissipation in each MAC unit is given in Table 4.4. The performance of the coefficients generated by the method MSWO targeting four units is the best, as expected. However there is a little performance increase (three per cent) over MSWO coefficients targeting one MAC unit.

Table 4.3. Switching activity counts and number of ones in synthesized filters for four MAC units

Filter	Method	Switching Activity				Number of Ones
		MAC 0	MAC 1	MAC 2	MAC 3	
M2	NOPT	47	38	37	46	186
	MSWO (*P*=1)	36	22	24	36	90
	MSWO (*P*=4)	22	16	16	22	92

Table 4.4. Power simulation results using four MAC units

Filter	Method	Power (μW)					Reduction (%)
		MAC 0	MAC 1	MAC 2	MAC 3	Total	
M2	NOPT	1523	1357	1341	1525	5746	-
	MSWO (*P*=1)	994	937	906	981	3818	33.5
	MSWO (*P*=4)	944	875	855	972	3646	36.5

4.4. Summary

In this chapter, the formulation of finding power optimum coefficients for the realization of digital FIR filters on programmable DSPs is demonstrated. The effectiveness of the integer quadratic formulation is tested on four low pass FIR filters taken from the literature. The results indicate that when minimization of switching activity is the goal, the most effective method is to re-order coefficients. However, when it comes to power performance, coefficients optimized with the proposed methods outperformed reordered coefficients in all cases. The effectiveness of the formulation on DSPs having multiple units is also shown on a design example. SWAGAM algorithm gives comparable results in a much shorter time than the optimum quadratic programming based formulation.

The formulation of the low switching activity filter design problem is given for two's complement number representation. However, the procedure can be easily applied to sign-magnitude notation as well.

5. POWER OPTIMIZATION IN FIR BASED SYSTEMS – EQUALIZER DESIGN EXAMPLE

Frequency response and/or linear-phase are not a design constraint for equalizing filters, namely equalizers, which are extensively used in communication systems. They are employed at the receiver end of the communication system to recover the original transmitted signal distorted when passing through the channel via effects such as additive noise and inter-symbol interference (ISI) [50,51].

In this work, we consider the design of low-power hardware efficient MMSE equalizer. The coefficients of the MMSE equalizer are obtained by minimizing the MSE between output and the actual transmitted data. The resulting coefficients have infinite precision and hence quantization is needed for a practical VLSI implementation. A simple way of quantization is to round the coefficients to a desired wordlength. However, from the implementation point of view, the hardware/power cost of the VLSI realization cannot be controlled by simple rounding. We propose the algorithm EQUGAM which is an extension to SPTGAM for finding equalizer coefficients with fewer non-zero digits in the coefficients. EQUGAM differs from SPTGAM in that MSE is the design constraint. Hence, the optimization problems to be solved are different. Moreover, filters designed by EQUGAM are not constrained to have linear phase. EQUGAM uses the MSE of the rounded MMSE coefficients as an upper bound to the MSE. Hence the resulting equalizers do not face performance degradation.

This chapter is organized as follows. First, the signal models of the channel and equalizer are given. The optimization problem is defined next. Then the proposed algorithm for solving the optimization problem is presented. Next, design examples are given comparing the proposed algorithm to simple rounding and this chapter is finished with concluding remarks.

5.1. Signal Model

A communication system with a linear discrete-time channel is shown in Figure 5.1.

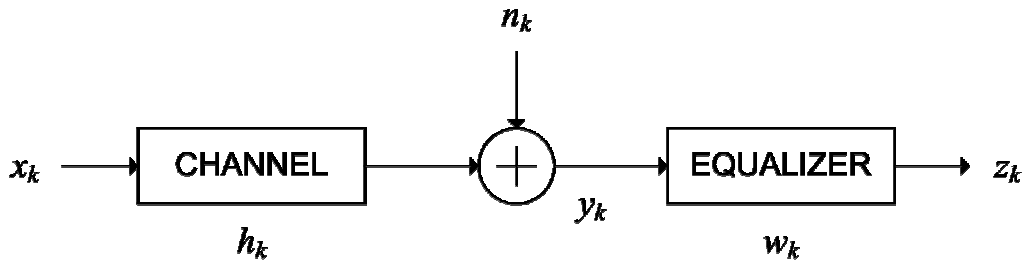

Figure 5.1. A communication system with an FIR filter equalizer

The output is

$$y_k = \sum_{i=0}^{N-1} h_i x_{k-i} + n_k \tag{5.1}$$

where x_k is the data to be transmitted over the channel, h_k is the channel impulse response with length M, and n_k is white Gaussian noise with power spectral density σ^2. The N-tap linear equalizer in Figure 5.1 is described by the vector $\mathbf{w} = [w_0\, w_1 \cdots w_{N-1}]^{\mathrm{T}}$. The equalizer output at time k is

$$z_k = \mathbf{w}^{\mathrm{T}}\mathbf{y}_k \tag{5.2}$$

where the channel output vector $\mathbf{y}_k = [y_k\, y_{k-1} \ldots y_{k-N+1}]^{\mathrm{T}}$ is given by

$$\mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \mathbf{n}_k \tag{5.3}$$

where $\mathbf{x}_k = [x_k\, x_{k-1} \cdots x_{k-M-N+1}]^{\mathrm{T}}$ is a vector of channel inputs, $\mathbf{n}_k = [n_k\, n_{k-1} \cdots n_{k-M+1}]^{\mathrm{T}}$ is a vector of noise samples, and $\mathbf{H}$ is the $N\times(M+N-1)$ channel convolution matrix

$$\mathbf{H} = \begin{bmatrix} h_0 & h_1 & \cdots & h_{M-1} & 0 & \cdots & & 0 \\ 0 & h_0 & h_1 & \cdots & h_{M-1} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \cdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{M-1} & 0 \\ 0 & & \cdots & 0 & h_0 & h_1 & \cdots & h_{M-1} \end{bmatrix} \tag{5.4}$$

The delay of the channel and equalizer system can be defined to be $D=\lfloor (M+N-1)/2 \rfloor$ so that the reciever output z_k is an approximation to the input sample x_{k-D}. Then, the equalizer coefficient vector $\mathbf{w}$ can be computed using the MSE criterion $J=E[(z_k-x_{k-D})^2]$. The MSE can be written in terms of the equalizer coefficients as

$$J = \mathbf{w}^T\mathbf{R}\mathbf{w} - \mathbf{p}^T\mathbf{w} - \mathbf{w}^T\mathbf{p} + 1 \tag{5.5}$$

where $\mathbf{R} = \mathbf{H}\mathbf{H}^T + \sigma^2\mathbf{I}$, and $\mathbf{p}$ is the $(D+1)$th column of $\mathbf{H}$. The optimum coefficient vector is

$$\mathbf{w}_{\text{MMSE}} = \mathbf{R}^{-1}\mathbf{p} \tag{5.6}$$

which provides the MMSE as

$$J_{\text{MMSE}} = 1 - \mathbf{p}^T\mathbf{R}^{-1}\mathbf{p}. \tag{5.7}$$

5.2. Problem Formulation

The coefficients $\mathbf{w} = [w_0\, w_1 \cdots w_{N-1}]^T$ of an N-tap equalizer can be written in B-bit canonic signed digit (CSD) representation as:

$$w_i = \mathbf{q}^T\mathbf{b}_i \tag{5.8}$$

where $\mathbf{q} = [2^0 2^{-1} \cdots 2^{-B+1}]^T$ is the B-bit quantization vector and $\mathbf{b}_i = [b_{i,0}\, b_{i,1} \cdots b_{i,B-1}]^T$ is the B-bit CSD expansion of coefficient w_i where $b_{i,j} \in \{-1,0,1\}$ and $b_{i,j} + b_{i,j+1} \leq 1$ for $j=0,\ldots,B-2$. Note that $b_{i,j}$ is the j'th bit and $j=0$ corresponds to the most significant bit of coefficient w_i. The non-zero digits $b_{i,j} \in \{-1,1\}$ are also referred to as signed power of two (SPT) terms in the literature.

Using (5.8), the total number of SPT terms (or equivalently the non-zero bits) in coefficient w_i is obtained by computing $\mathbf{b}_i^T\mathbf{b}_i$. If the equalizer length is N, the total number of SPT terms in the coefficients is obtained by computing $\mathbf{b}^T\mathbf{b}$ where $\mathbf{b}$ is the merged CSD expansion vectors of all the coefficients such that

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_{N-1} \end{bmatrix} \tag{5.9}$$

The coefficients that give the MMSE have infinite precision and are unique. To be able to make a search for coefficient vector with fewer SPT terms we need to have a feasible search space to exist. Then the coefficient vector **w** should satisfy

$$\mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w} - \mathbf{p}^{\mathrm{T}}\mathbf{w} - \mathbf{w}^{\mathrm{T}}\mathbf{p} + 1 \leq J_{\max}. \tag{5.10}$$

Obviously, for a feasible search space to exist $J_{\max} \geq J_{\mathrm{MMSE}}$. For the case $J_{\max} = J_{\mathrm{MMSE}}$ the coefficient vector **w** is unique and $\mathbf{w} = \mathbf{w}_{\mathrm{MMSE}}$.

Since quantization is inevitable and the MSE value of the quantized coefficients will always be greater than MMSE, a feasible choice is to set the upper bound $J_{\max}$ to the MSE value of that can be obtained by rounding the MMSE coefficients to the desired wordlength. Then, denoting the vector of MMSE coefficents quantized to B-bit by $\mathbf{w}_B$, the MSE of B-bit rounded coefficients can be found using (5.5) as

$$J_B = \mathbf{w}_B{}^{\mathrm{T}}\mathbf{R}\mathbf{w} - \mathbf{p}^{\mathrm{T}}\mathbf{w}_B - \mathbf{w}_B{}^{\mathrm{T}}\mathbf{p} + 1. \tag{5.11}$$

By setting $J_{\max} = J_{\mathrm{B}}$ it is guaranteed that the search results in a coefficient vector with an MSE either equal or better than the B-bit rounded MMSE coefficient vector.

Now, to minimize the total number of SPT terms in the coefficients of an equalizer for which the maximum allowed MSE is $J_{\max}$, the problem can be formulated as

Minimize $\mathbf{b}^{\mathrm{T}}\mathbf{b}$

Such that

$$\mathbf{b}^{\mathrm{T}}\mathbf{Q}\mathbf{R}\mathbf{Q}^{\mathrm{T}}\mathbf{b} - \mathbf{p}^{\mathrm{T}}\mathbf{Q}^{\mathrm{T}}\mathbf{b} - \mathbf{b}^{\mathrm{T}}\mathbf{Q}\mathbf{p} + 1 \leq J_{\max} \tag{5.12}$$

where **Q** is the quantization matrix defined as

$$\mathbf{Q} = \begin{bmatrix} \mathbf{q} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{q} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{q} \end{bmatrix}$$

from which the coefficient vector **w** can be obtained by $\mathbf{w} = \mathbf{Q}^{\mathrm{T}}\mathbf{b}$.

5.3. The EQUGAM Algorithm

There are $N*B$ number of variables to be solved in the formulation of (5.12). Moreover, it is a combinatorial optimization problem with a quadratic objective function and quadratic constraints. Instead of using this complex formulation, we translate the problem to an optimization problem which has a linear objective function with a single quadratic constraint. It requires only N variables to be determined. Furthermore, our formulation has a closed form solution that only requires matrix inversion with a maximum size of $N \times N$.

Given the equalizer length N, quantization wordlength B, and the desired mean square error bound $J_{\max}$, the search should end up with a coefficient vector $\mathbf{w}=[w_0\ w_1\ \cdots\ w_{N-1}]^{\mathrm{T}}$ with fewer SPT terms satisfying (5.10). Unlike the MMSE coeffcients, there may be more than one coefficient vector. Hence, the value of each coefficient is not unique, but falls in a range, i.e. they have lower and upper bounds. Let $w_{i,\min}$ and $w_{i,\max}$ denote the lower and upper bounds for coefficient w_i. The boundary values can be found by independently solving the following pair of optimization problems for each coefficient w_i, $i=0,1,\ldots,N-1$

$$w_{i,\min} = \text{minimum } w_i$$

such that

$$\mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w} - \mathbf{p}^{\mathrm{T}}\mathbf{w} - \mathbf{w}^{\mathrm{T}}\mathbf{p} + 1 \le J_{\max} \tag{5.13}$$

and

$$w_{i,\max} = \text{maximum } w_i$$

such that

$$\mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w} - \mathbf{p}^{\mathrm{T}}\mathbf{w} - \mathbf{w}^{\mathrm{T}}\mathbf{p} + 1 \le J_{\max} \tag{5.14}$$

for which the Lagrange multiplier based solutions are given in Appendix A. Note that $w_{i,\min}$ and $w_{i,\max}$ are real values and w_i belongs to the range $w_i \in [w_{i,\min}, w_{i,\max}]$. Due to the finite wordlength constraint w_i is a number that appears in this range and must be represented with B bits. As a result, all possible digital values for w_i form a finite set. Hence, let V_i be such a digital value set of w_i then.

$$V_i = \left\{ \forall v \in \left[w_{i,\min}, w_{i,\max} \right] \middle| v = \sum_{j=0}^{B-1} b_j 2^{-j}, b_j \in \{-1,0,1\} \right\}. \tag{5.15}$$

A filter can be composed by setting the coefficients to the values selected from their feasible value sets $\bigcup_{i=0}^{N-1} V_i$. The selected coefficients must satisfy (5.10).

Given the desired mean square error bound $J_{\max}$, the EQUGAM algorithm, shown in Figure 5.2, iteratively finds the coefficient vector $\mathbf{w}$ satisfying (5.10). The algorithm is based on the SPTGAM algorithm The basic difference is the optimization problems solved to compute the boundary values of the coefficients. EQUGAM finds the refined boundary values ($w^s_{i,\min}$, $w^s_{i,\max}$) by solving the following problems

$$w^s_{i,\min} = \text{minimum } w_i$$

Such that

$$\mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w} - \mathbf{p}^{\mathrm{T}}\mathbf{w} - \mathbf{w}^{\mathrm{T}}\mathbf{p} + 1 \le J_{\max} \tag{5.16}$$

$$w^s_{i,\max} = \text{maximum } w_i$$

Such that

$$\mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w} - \mathbf{p}^{\mathrm{T}}\mathbf{w} - \mathbf{w}^{\mathrm{T}}\mathbf{p} + 1 \le J_{\max} \tag{5.17}$$

where $\mathbf{w}=[w_0\, w_1 \cdots w_k \cdots w_{N-1}]^{\mathrm{T}}$ and

```
EQUGAM(N , L, J_max, ⋃_{i=0}^{N-1} V_i )
  W = ∅;
  O_max = ∞;
  Obtain w^s_{0,min} and w^s_{0,max} by solving problems (5.16) and (5.17);
  V_0^s = EQU_VALUE_SELECT(L, w^s_{0,min} , w^s_{0,max} , V_i);
  i = 0;
  WHILE ( i ≥ 0 )
    IF ( V_i^s ≠ ∅)
      w_i = v* such that v* is the first element of ordered set V_i^s;
      W = W ∪ {w_i};
      V_i^s = V_i^s − {v*};
      IF (O_W* ≤ O_max)
        IF (i < N − 1)
          i = i + 1;
          Obtain w^s_{i,min} and w^s_{i,max} by solving problems (5.16) and (5.17)
          V_i^s =EQU_VALUE_SELECT(L, w^s_{i,min} , w^s_{i,max} , V_i)
        ELSE-IF (J_W ≤ J_max) /* solution is feasible */
          W is a solution;
          O_max = O_W ;
        END-IF
    ELSE
      W = W − {w_i};
      i = i − 1;
    END-IF
  END
```

Figure 5.2. The EQUGAM algorithm

$$w_k = \begin{cases} \text{fixed} & 0 \le k < i \\ \text{free} & i \le k < N \end{cases}.$$

The Lagrange multiplier based solutions of (5.16) and (5.17) are given in Appendix A. It should be noted that these equations are different from (5.13) and (5.14) so as to refine the search space for the i'th coefficient after fixing the first $i-1$ coefficients. Fixing the values of $i-1$ coefficients moves the boundary values of the i'th coefficient ($w_{i,\min}$, $w_{i,\max}$) towards each other ($w^s_{i,\min} \ge w_{i,\min}$, $w^s_{i,\max} \le w_{i,\max}$) and the number of possible values for this coefficient will be reduced. Hence, the refined value set V_i^s is a subset of the value set V_i

$$V_i^s = \{ \forall\, v \in V_i \mid w^s_{i,\min} \le v \le w^s_{i,\max} \}.$$

EQU_VALUE_SELECT(L, $w^s_{i,\min}$, $w^s_{i,\max}$, V_i)
  $V_i^s = \{\forall\ v \in V_i \mid w^s_{i,\min} \le v \le w^s_{i,\max}\}$;
  $w^s_{i,\mathrm{mid}} = (w^s_{i,\min} + w^s_{i,\max})/2$;
  Order $V_i^s = (v_1, v_2, \ldots, v_L)$ such that
    $|v_1 - w^s_{i,\mathrm{mid}}| < |v_2 - w^s_{i,\mathrm{mid}}| < \cdots < |v_L - w^s_{i,\mathrm{mid}}|$;
  RETURN V_i^s;

Figure 5.3. The VALUE_SELECT algorithm of EQUGAM

The refined value set is an ordered set generated by the VALUE_SELECT algorithm in Figure 5.3. The elements in the refined values set V_i^s are ordered according to their closeness to the average of the refined boundary values ($w^s_{i,\mathrm{mid}}$). After a value v^* of the refined value set is assigned to w_i, it is removed from the value set and moved to the solution set W, which is an ordered set. The first element in W corresponds to w_0 and the last element in W corresponds to w_{N-1}.

The EQUGAM algorithm uses a cutoff mechanism to avoid unnecessary search. The cutoff mechanism decides whether to branch further or not depending on the guess of branching cost, i.e. the predicted number of SPT terms (O_{W^*}) at its current position in the search tree. That is, if there can not be a better solution than the current solution, i.e. $O_{W^*} \le O_{\max}$, it will not branch further, hence the tree is cut-off.

5.4. Experimental Results

In this section, we show the effectiveness of our algorithm in designing MSE equalizers. For all the examples the refined value set size $L = 2$. The SPT term prediction startegy is U_SUBAVG.

A total number of 32 randomly generated channels are used to compare the EQUGAM algorithm to simple rounding. The channels are characterized by an impulse response consisting of 11 coefficients. The BER vs E_b/N_0 performance of the length 31 MMSE equalizers for the randomly generated channels is shown in Figure 5.4. The plot is generated using data packets of size 1000 and a total packet count of 4000. The data is binary having values of {-1,1}. As can be seen from the figure the channels are not all good or all severe but distributed from severe to good.

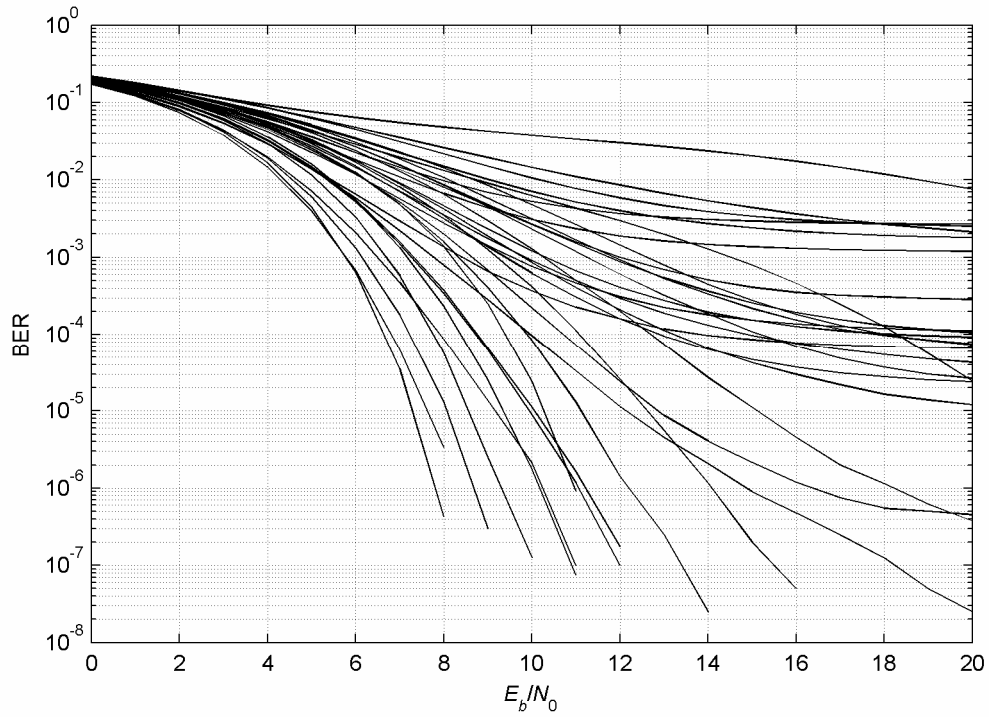

Figure 5.4. BER vs E_b/N_0 performance of the length 31 MMSE equalizers for 32 randomly generated channels

The performance of the EQUGAM algorithm is compared to rounded MMSE coefficients. Equalizers with lengths ranging from 11 to 51 with an interval of 5 are designed for E_b/N_0 levels of 10 dB, 15 dB, and 20 dB. The comparison is made for wordlengths of B=6, B=7, and B=8 bits. For the B-bit GAM filters, the upper bound of the MSE (J_{max}) is set to the MSE value of the B-bit rounded MMSE filters.

The average percentage gain in the number of SPT terms for the EQUGAM equalizers over the rounded MMSE equalizers is shown in Figure 5.5, Figure 5.6, and Figure 5.7. EQUGAM filters may have upto 18% fewer SPT terms than the rounded MMSE coefficients. Moreover, they have either smaller or equal MSE. The same results are plotted in Figure 5.8, Figure 5.9, and Figure 5.10 groupped according to the quantization wordlength B=6, B=7, and B=8 respectively.

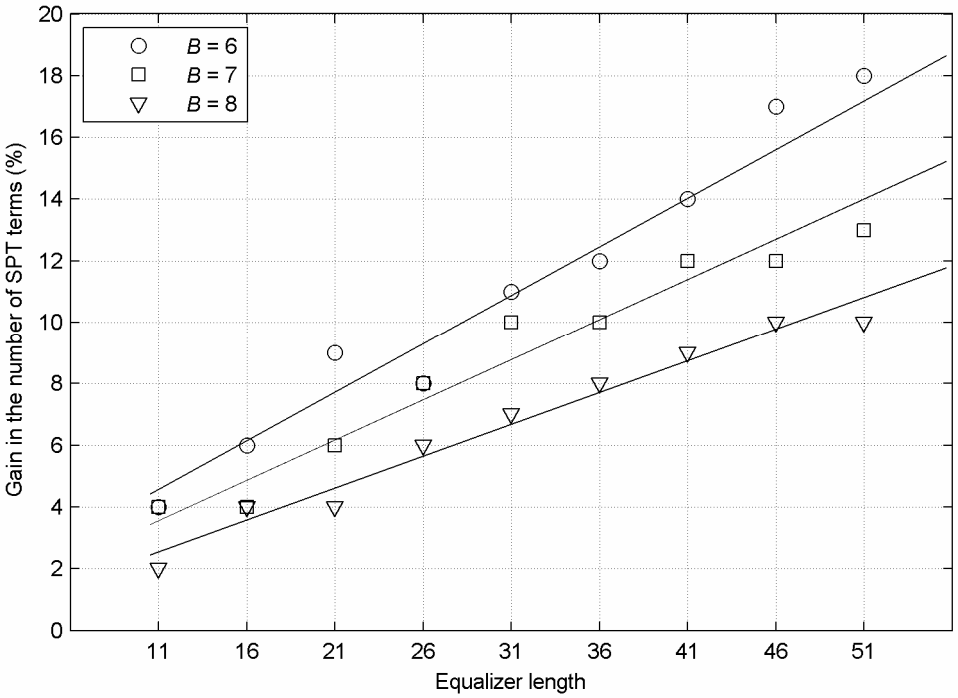

Figure 5.5. The average percentage gain in the number of SPT terms for the EQUGAM equalizers over the rounded MMSE equalizers: E_b/N_0 = 10 dB

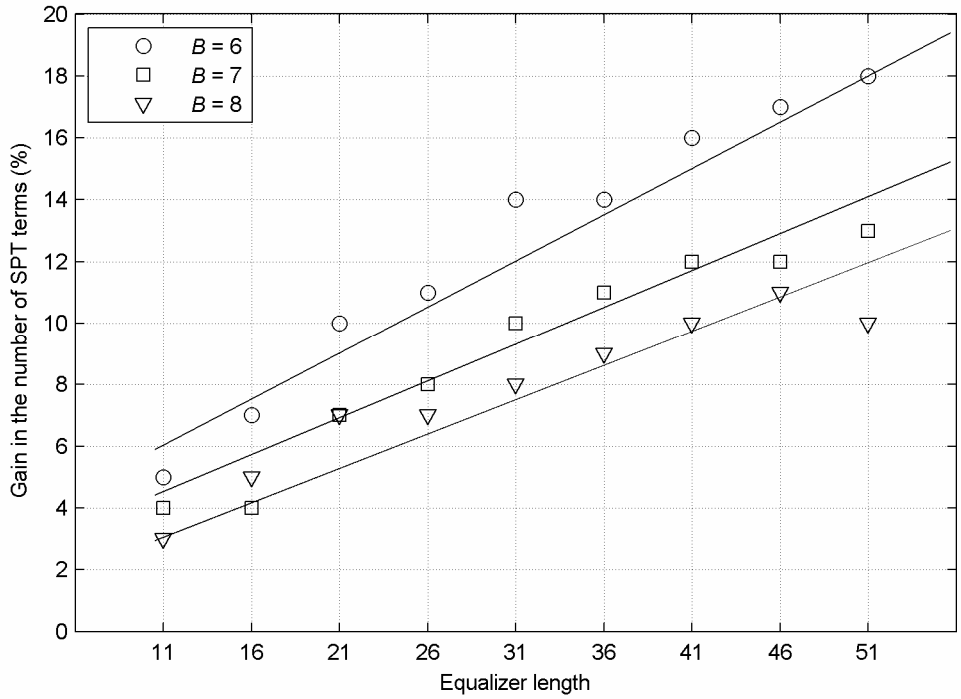

Figure 5.6. The average percentage gain in the number of SPT terms for the EQUGAM equalizers over the rounded MMSE equalizers: E_b/N_0 = 15 dB

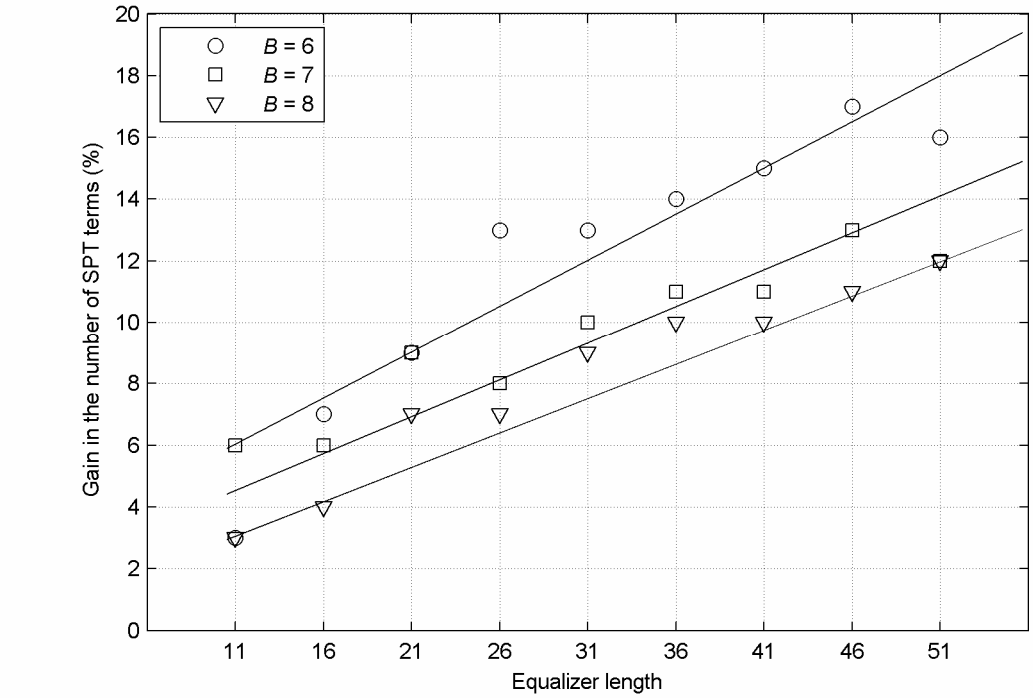

Figure 5.7. The average percentage gain in the number of SPT terms for the EQUGAM equalizers over the rounded MMSE equalizers: E_b/N_0 = 20 dB

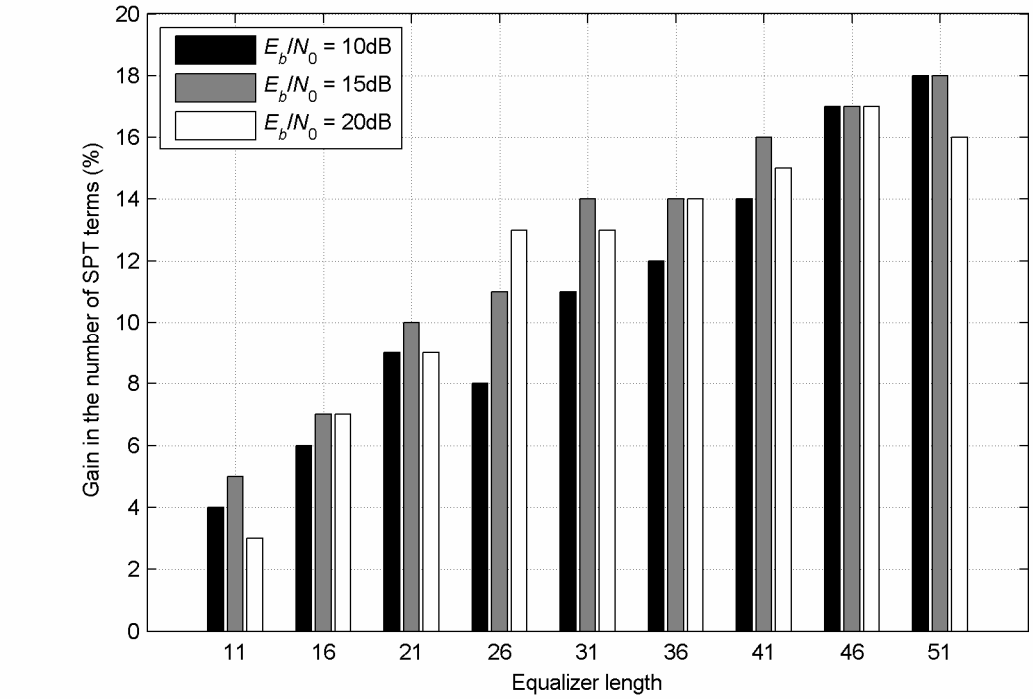

Figure 5.8. The average percentage gain in the number of SPT terms for the EQUGAM equalizers over the rounded MMSE equalizers: wordlength B = 6 bits

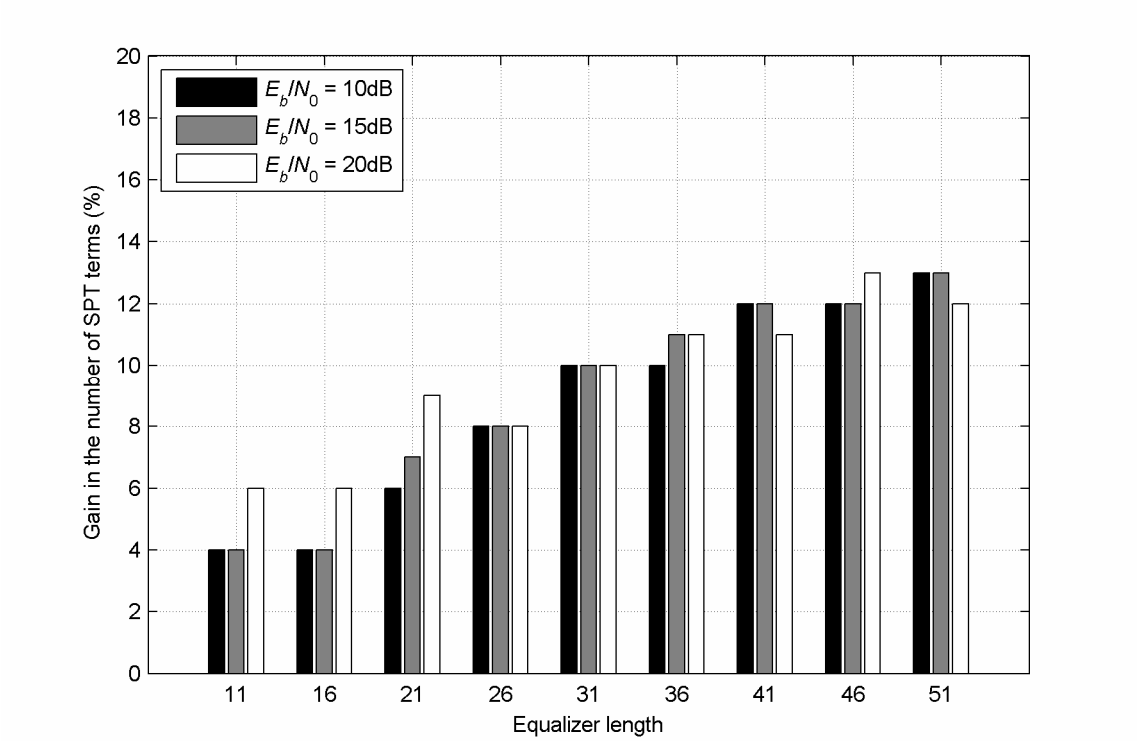

Figure 5.9. The average percentage gain in the number of SPT terms for the EQUGAM equalizers over the rounded MMSE equalizers: wordlength B = 7 bits

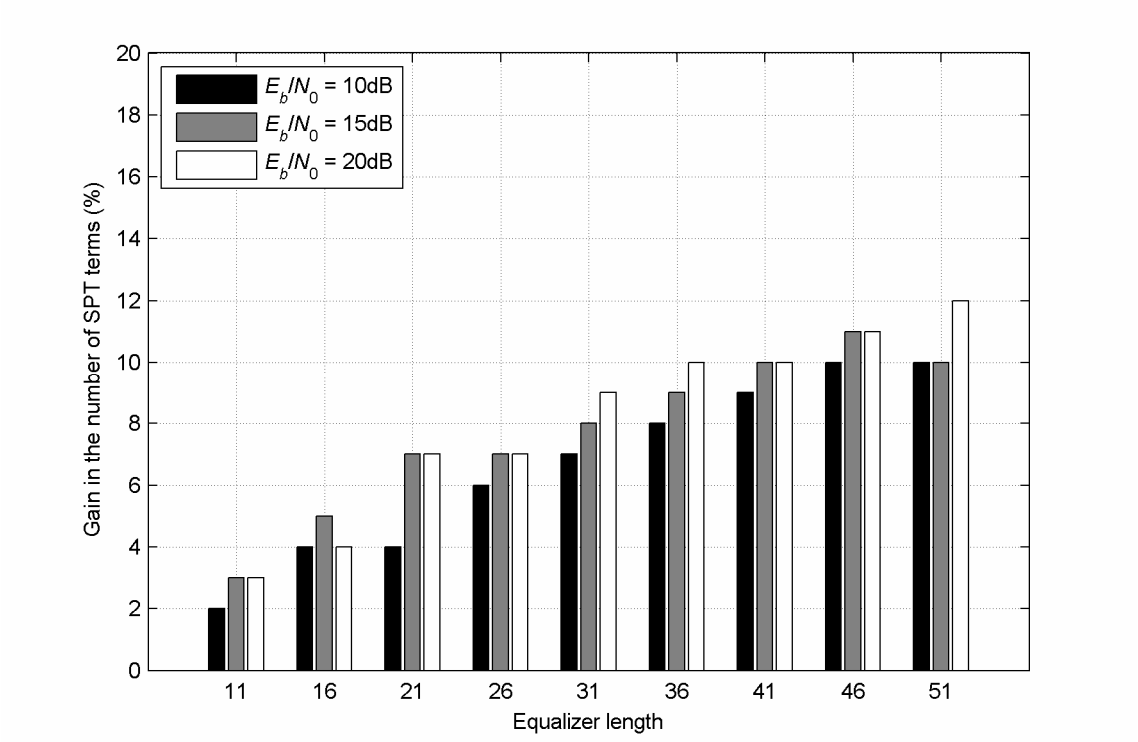

Figure 5.10. The average percentage gain in the number of SPT terms for the EQUGAM equalizers over the rounded MMSE equalizers: wordlength B = 8 bits

5.5. Summary

An algorithm for designing low power/hardware cost equalizing FIR filters was presented. The algorithm optimizes SPT terms in the coefficients given the maximum mean square error. Although the worst case run time of the algorithm is exponential, its capability to find appreciably good solutions in a reasonable amount of time makes it a desirable CAD tool for designing low power/hardware constant coefficient equalizers. The effectiveness of the algorithm is shown on different examples. The filters found by the proposed algorithm have upto 18% less SPT terms then obtained by rounded MMSE filters. With a slight modificiation the EQUGAM algorithm can also be used to optimize MSE given the length of the filter and wordlength of the coefficients.

6. CONCLUSION AND FUTURE WORK

In this thesis, the GAM algorithm for the design of low-power linear-phase FIR filters is developed. GAM is a discrete coefficient FIR filter design algorithm. Depending on the choice of implementation, GAM optimizes filter coefficients for low power. For parallel implementation of constant coefficient filters, the number of nonzero digits in the coefficients is minimized. For sequential realization of variable coefficient filters, the switching activity between coefficients is minimized. Although the worst case run time of the algorithm is exponential, its capability to find a solution in a reasonable amount of time makes it a desirable CAD tool for designing discrete coefficient linear-phase FIR filters. The effectiveness of the algorithm is more apparent on high-order filters when compared to other methods proposed in the literature.

The performance of the GAM algorithm strictly depends on the cost prediction mechanism. Choosing the lower integer bound of the average cost of the values in the feasible value set of a coefficient, namely U_SUB_AVG, seems to be best solution from the search time/optimality trade-off. However, this can be improved by developing other cost prediction strategies which is left as a future work.

An important issue in filter design is to keep the quantization wordlength of the coefficients as small as possible. In this context, an empiric formula for the lower bound of the quantization wordlength of a linear-phase FIR is developed. It is obtained by designing several filters using the proposed filter design algorithm. As opposed to other formulae proposed in the literature it can directly be calculated from the frequency response characteristics independent of the filter length. In its present form, the formula takes three different forms depending on the width of the transition band. Hence, there is a need for unification which is left as a future work.

As an example for power optimization in FIR based systems the EQUGAM algorithm is developed for designing equalizing FIR filters.The algorithm is a modified version of the GAM algorithm which optimizes nonzero digits in the coefficients given the maximum mean square error. As opposed to the linear-phase FIR filter design algorithm

GAM, linear programming could not be used due to the quadratic nature of the problem. Instead, a closed form solution is developed by using Lagrange multipliers. The effectiveness of the algorithm is shown on different examples. In its present form EQUGAM can be used for non-adaptive equalizers. However, where the variation in the characteristics of the channel is large, adaptive equalizers are to be used. Hence, methods for applying our algorithm for adaptive equalizers stays as an open area for future study.

APPENDIX A: PROBLEM SOLUTIONS

A.1. Solutions to the Optimization Problems of (5.16) and (5.17)

Minimize/maximize w_i

Such that

$$\mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w} - \mathbf{p}^{\mathrm{T}}\mathbf{w} - \mathbf{w}^{\mathrm{T}}\mathbf{p} + 1 \le J_{\max}$$

where $\mathbf{w} = [w_0\, w_1 \cdots w_k \cdots w_{N-1}]^{\mathrm{T}}$ and

$$w_k = \begin{cases} \text{fixed} & 0 \le k < i \\ \text{free} & i \le k < M \end{cases}.$$

The coefficient vector can be partitioned as:

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_{\mathrm{a}} \\ \cdots \\ w_i \\ \cdots \\ \mathbf{w}_{\mathrm{d}} \end{bmatrix} \tag{A.1}$$

where $\mathbf{w}_{\mathrm{a}} = [w_0\, w_1 \cdots w_{i-1}]^{\mathrm{T}}$ represents the coefficients whose value is fixed, w_i is the $(i+1)$'th coefficient in $\mathbf{w}$ for which the refined boundary values $(w^s_{i,\min}, w^s_{i,\max})$ are to be found, and $\mathbf{w}_{\mathrm{d}} = [w_i\, w_{i+1} \cdots w_{M-1}]^{\mathrm{T}}$ the coefficients that are currently free. Accordingly $\mathbf{R}$ and $\mathbf{p}$ can be partitioned as follows

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{\mathrm{a}} & \mathbf{a}_i & \mathbf{R}_{\mathrm{c}}^{\mathrm{T}} \\ \cdots & \cdots & \cdots \\ \mathbf{a}_i^{\mathrm{T}} & r_i & \mathbf{d}_i^{\mathrm{T}} \\ \cdots & \cdots & \cdots \\ \mathbf{R}_{\mathrm{c}} & \mathbf{d}_i & \mathbf{R}_{\mathrm{d}} \end{bmatrix} \tag{A.2}$$

$$\mathbf{p} = \begin{bmatrix} \mathbf{p}_{\mathrm{a}} \\ \cdots \\ p_i \\ \cdots \\ \mathbf{p}_{\mathrm{d}} \end{bmatrix} \tag{A.3}$$

The problem can be solved using Lagrange multiplier relaxation. First, the inequality constraint is converted to an equality constraint by adding a dummy variable z. Then the problem turns out to be

Minimize/maximize w_i

Such that

$$\mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w} - \mathbf{p}^{\mathrm{T}}\mathbf{w} - \mathbf{w}^{\mathrm{T}}\mathbf{p} + 1 - J_{\max} + z^2 = 0$$

for which we can write the lagrangian L as follows

$$L(\mathbf{w},\lambda,z) = w_i \pm \lambda\,(\mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w} - \mathbf{p}^{\mathrm{T}}\mathbf{w} - \mathbf{w}^{\mathrm{T}}\mathbf{p} + 1 - J_{\max} + z^2) \tag{A.4}$$

The sign ± is used intentionally to indicate wheter the problem is a minimization (+) or maximization (–) problem.

The variables in $\mathbf{w}$ are w_i and $\mathbf{w}_d$ and then from Kuhn-Tucker conditions [52]

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial z} = \frac{\partial L}{\partial \mathbf{w}_d} = \frac{\partial L}{\partial \lambda} = 0 \tag{A.5}$$

Differentiating L by w_i yields

$$\frac{\partial L}{\partial w_i} = 1 \pm \lambda(\mathbf{w}^{\mathrm{T}}\begin{bmatrix}\mathbf{a}_i \\ \cdots \\ r_i \\ \cdots \\ \mathbf{d}_i\end{bmatrix} - p_i^{\mathrm{T}}) = 0 \Rightarrow \lambda \neq 0 \tag{A.6}$$

Then, differentiating by z and using $\lambda \neq 0$ yields

$$\frac{\partial L}{\partial z} = 0 \pm 2\lambda z = 0 \Rightarrow z = 0 \tag{A.7}$$

Differentiating L by $\mathbf{w}_\mathrm{d}$

$$\frac{\partial L}{\partial \mathbf{w}_\mathrm{d}} = 0 \pm \lambda(\mathbf{w}^\mathrm{T}\begin{bmatrix}\mathbf{R}_\mathrm{c}^\mathrm{T}\\ \cdots\\ \mathbf{d}_i^\mathrm{T}\\ \cdots\\ \mathbf{R}_\mathrm{d}\end{bmatrix} - \mathbf{p}_\mathrm{d}^\mathrm{T}) = 0$$

and again using $\lambda \neq 0$

$$\begin{bmatrix}\mathbf{w}_\mathrm{a}^\mathrm{T} \vdots w_i^\mathrm{T} \vdots \mathbf{w}_\mathrm{d}^\mathrm{T}\end{bmatrix}\begin{bmatrix}\mathbf{R}_\mathrm{c}^\mathrm{T}\\ \cdots\\ \mathbf{d}_i^\mathrm{T}\\ \cdots\\ \mathbf{R}_\mathrm{d}\end{bmatrix} - \mathbf{p}_\mathrm{d}^\mathrm{T} = 0$$

which yields

$$\mathbf{w}_\mathrm{d}^\mathrm{T} = (\mathbf{p}_\mathrm{d}^\mathrm{T} - \mathbf{w}_\mathrm{a}^\mathrm{T}\mathbf{R}_\mathrm{c}^\mathrm{T} - w_i^\mathrm{T}\mathbf{d}_i^\mathrm{T})\mathbf{R}_\mathrm{d}^{-1}. \tag{A.8}$$

Finally, differentiation by λ yields

$$\frac{\partial L}{\partial \lambda} = \mathbf{w}^T\mathbf{R}\mathbf{w} - \mathbf{p}^T\mathbf{w} - \mathbf{w}^T\mathbf{p} + 1 - J_{\max} + z^2 = 0 \tag{A.9}$$

Since $z = 0$ and using (A.1), (A.2), and (A.3) for **w**, **R**, and **p** respectively we get

$$\begin{bmatrix}\mathbf{w}_\mathrm{a}^\mathrm{T} w_i^\mathrm{T} \mathbf{w}_\mathrm{d}^\mathrm{T}\end{bmatrix}\begin{bmatrix}\mathbf{R}_\mathrm{a} & \mathbf{a}_i & \mathbf{R}_\mathrm{c}^\mathrm{T}\\ \mathbf{a}_i^\mathrm{T} & r_i & \mathbf{d}_i^\mathrm{T}\\ \mathbf{R}_\mathrm{c} & \mathbf{d}_i & \mathbf{R}_\mathrm{d}\end{bmatrix}\begin{bmatrix}\mathbf{w}_\mathrm{a}\\ w_i\\ \mathbf{w}_\mathrm{d}\end{bmatrix} - \begin{bmatrix}\mathbf{p}_\mathrm{a}^\mathrm{T} p_i^\mathrm{T} \mathbf{p}_\mathrm{d}^\mathrm{T}\end{bmatrix}\begin{bmatrix}\mathbf{w}_\mathrm{a}\\ w_i\\ \mathbf{w}_\mathrm{d}\end{bmatrix} - \begin{bmatrix}\mathbf{w}_\mathrm{a}^\mathrm{T} w_i^\mathrm{T} \mathbf{w}_\mathrm{d}^\mathrm{T}\end{bmatrix}\begin{bmatrix}\mathbf{p}_\mathrm{a}\\ p_i\\ \mathbf{p}_\mathrm{d}\end{bmatrix} + 1 - J_{\max} = 0$$

Doing the matrix multiplication and after re-grouping the products we get

$$\begin{Bmatrix}\mathbf{w}_\mathrm{a}^\mathrm{T}\mathbf{R}_\mathrm{a}\mathbf{w}_\mathrm{a} + \mathbf{w}_\mathrm{a}^\mathrm{T}\mathbf{a}_i w_i + \mathbf{w}_\mathrm{a}^\mathrm{T}\mathbf{R}_\mathrm{c}^\mathrm{T}\mathbf{w}_\mathrm{d}\\ + w_i^\mathrm{T}\mathbf{a}_i^\mathrm{T}\mathbf{w}_\mathrm{a} + w_i^\mathrm{T} r_i w_i + w_i^\mathrm{T}\mathbf{d}_i^\mathrm{T}\mathbf{w}_\mathrm{d}\\ + \mathbf{w}_\mathrm{d}^\mathrm{T}\mathbf{R}_\mathrm{c}\mathbf{w}_\mathrm{a} + \mathbf{w}_\mathrm{d}^\mathrm{T}\mathbf{d}_i w_i + \mathbf{w}_\mathrm{d}^\mathrm{T}\mathbf{R}_\mathrm{d}\mathbf{w}_\mathrm{d}\end{Bmatrix} - \begin{Bmatrix}\mathbf{p}_\mathrm{a}^\mathrm{T}\mathbf{w}_\mathrm{a}\\ + p_i^\mathrm{T} w_i\\ + \mathbf{p}_\mathrm{d}^\mathrm{T}\mathbf{w}_\mathrm{d}\end{Bmatrix} - \begin{Bmatrix}\mathbf{w}_\mathrm{a}^\mathrm{T}\mathbf{p}_\mathrm{a}\\ + w_i^\mathrm{T} p_i\\ + \mathbf{w}_\mathrm{d}^\mathrm{T}\mathbf{p}_\mathrm{d}\end{Bmatrix} + 1 - J_{\max} = 0 \tag{A.10}$$

Using (A.8) and doing some simplification we end up with

$$Aw_i^2 + 2Bw_i + \mathrm{C} = 0 \tag{A.11}$$

where

$$\begin{aligned} A &= r_i - \mathbf{d}_i^{\mathrm{T}}\mathbf{R}_{\mathrm{d}}^{-1}\mathbf{d}_i \\ B &= -p_i + \mathbf{d}_i^{\mathrm{T}}\mathbf{R}_{\mathrm{d}}^{-1}\mathbf{p}_{\mathrm{d}} + (\mathbf{a}_i^{\mathrm{T}} - \mathbf{d}_i^{\mathrm{T}}\mathbf{R}_{\mathrm{d}}^{-1}\mathbf{R}_{\mathrm{c}})\mathbf{w}_{\mathrm{a}} \\ C &= 1 - J_{\max} - \mathbf{p}_{\mathrm{d}}^{\mathrm{T}}\mathbf{R}_{\mathrm{d}}^{-1}\mathbf{p}_{\mathrm{d}} - 2(\mathbf{p}_{\mathrm{a}}^{\mathrm{T}} - \mathbf{p}_{\mathrm{d}}^{\mathrm{T}}\mathbf{R}_{\mathrm{d}}^{-1}\mathbf{R}_{\mathrm{c}})\mathbf{w}_{\mathrm{a}} + \mathbf{w}_{\mathrm{a}}^{\mathrm{T}}(\mathbf{R}_{\mathrm{a}} - \mathbf{R}_{\mathrm{c}}^{\mathrm{T}}\mathbf{R}_{\mathrm{d}}^{-1}\mathbf{R}_{\mathrm{c}})\mathbf{w}_{\mathrm{a}} \end{aligned} \tag{A.12}$$

If the roots of this equation are real then the minimum and maximum values of w_i are

$$w^s{}_{i,\min} = \min\left\{ \left(-B \pm \sqrt{B^2 - AC}\right) \Big/ A \right\} \tag{A.13}$$

$$w^s{}_{i,\max} = \max\left\{ \left(-B \pm \sqrt{B^2 - AC}\right) \Big/ A \right\} \tag{A.14}$$

otherwise w_i has no solution.

A.2. Solutions to the Optimization Problems of (5.13) and (5.14)

Minimize/maximize w_i

Such that

$$\mathbf{w}^T\mathbf{R}\mathbf{w} - \mathbf{p}^T\mathbf{w} - \mathbf{w}^T\mathbf{p} + 1 \le J_{\max}$$

where $\mathbf{w} = [w_0\, w_1 \cdots w_k \cdots w_{N-1}]^{\mathrm{T}}$ and w_k : free for $k = 0,1,\ldots,N-1$.

Let $\mathbf{w}_i$ be the vector consisting of the coefficients of $\mathbf{w}$ excluding w_i. Thus

$$\mathbf{w}_i = [w_0\, w_1 \cdots w_{i-1}\, w_{i+1} \cdots w_{N-1}]^{\mathrm{T}}. \tag{A.15}$$

Let $\mathbf{c}_i$ be the $(i+1)$'th column of $\mathbf{R}$. Using the partitioning in (A.2)

$$\mathbf{c}_i = \begin{bmatrix} \mathbf{a}_i \\ \cdots \\ \mathbf{d}_i \end{bmatrix}. \tag{A.16}$$

Similarly, let $\mathbf{R}_i$ be the matrix consisting of the elements of $\mathbf{R}$ excluding the (i+1)'th column and rows, and $\mathbf{p}_i$ be the vector of elements of $\mathbf{p}$ excluding the (i+1)'th element. Thus, using the partitioning in (A.2) and (A.3)

$$\mathbf{R}_i = \left[\begin{array}{c:c} \mathbf{R}_{\mathrm{a}} & \mathbf{R}_{\mathrm{c}}^{\mathrm{T}} \\ \hdashline \mathbf{R}_{\mathrm{c}} & \mathbf{R}_{\mathrm{d}} \end{array}\right], \ \mathbf{p}_i = \begin{bmatrix} \mathbf{p}_{\mathrm{a}} \\ \cdots \\ \mathbf{p}_{\mathrm{d}} \end{bmatrix}. \tag{A.17}$$

Using the same Lagrange multiplier relaxation methodology of section A.1 we end up with

$$w_{i,\min} = \min\left\{ \left(-B \pm \sqrt{B^2 - AC}\right) \Big/ A \right\} \tag{A.18}$$

$$w_{i,\max} = \max\left\{ \left(-B \pm \sqrt{B^2 - AC}\right) \Big/ A \right\} \tag{A.19}$$

where

$$\begin{aligned} A &= r_i - \mathbf{c}_i^{\mathrm{T}} \mathbf{R}_i^{-1} \mathbf{c}_i \\ B &= -p_i + \mathbf{c}_i^{\mathrm{T}} \mathbf{R}_i^{-1} \mathbf{p}_i \\ C &= 1 - J_{\max} - \mathbf{p}_i^{\mathrm{T}} \mathbf{R}_i^{-1} \mathbf{p}_i \end{aligned} \quad . \tag{A.20}$$

The values r_i and p_i in (A.20) are the elements of matrix $\mathbf{R}$ and vector $\mathbf{p}$ as denoted in (A.2) and (A.3) respectively.

APPENDIX B: USER MANUAL

B.1. FRQ

The program FRQ generates a text file containing the frequency response characteristics of the filter. The synopsis of the program is

FRQ "[*fm*1 *fx*1 *fs*1 *hm*1 *hx*1; *fm*2 *fx*2 *fs*2 *hm*2 *hx*2; ...]" > *ide*.frq

where

*fm*1 : cutin frequency
*fx*1 : cutout frequency
*fs*1 : frequency step (frequency grid stepsize)
*hm*1 : minimum amplitude
*hx*1 : maximum amplitude

are the frequency response specifications of the first band. You can enter as many band specifications as you want. You should place a semi-colon after each band specification except the last. The band specifications *fm*1 and *fx*1 should have frequency values normalized to one. That is the minimum frequency value can be 0 and the maximum frequency value can be 1. Figure B.1 shows an example filter for having two bands. If the output is not directed to a file (here *ide*.frq) it is written to the screen. The output filename extension should always be ".frq". An example run of FRQ for a filter named m3

FRQ "[0 0.36 0.001 0.9826 1.0174; 0.5 1 0.001 -0.001 0.001]" > m3.frq

B.2. HMX

The program HMX is used to determine the boundary values of the coefficients given the filter length *N*, quantization wordlength *B*, and the frequency response characteristics in a file named "*ide*.frq". hmx is run from the command line as follows

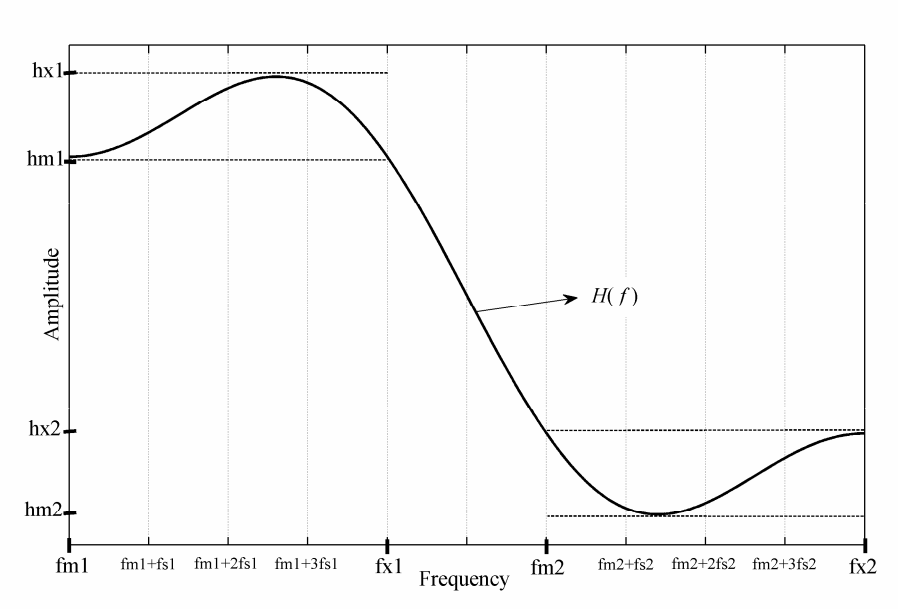

Figure B.1. Frequency band specifications as they are entered to the program FRQ

HMX [*ide*] [*N*] [*B*]

where the arguments are defined as follows:

ide : name of the filter
N : number of taps
B : wordlength

HMX looks for the frequency response characteristics file *ide*.frq. It should reside in the same directory where HMX is invoked. Otherwise the program will exit with an error.

The program generates an output text file named "*ide*_n*N*.hmx" in which the minimum and maximum values of the filter coefficients are written. An example output file is shown in Figure B.2. Lines beginning with an asterisk (*) are comment lines. Due to symmetry, only half of the coefficients' boundary values are written. The first column is the coefficient index (i in $h[i]$). The second is the minimum value of the coefficient ($h_{\min}[i]$) written in B-bit two's complement format. Similarly, the third is the maximum value of the coefficient ($h_{\max}[i]$) written in B-bit two's complement format. There are more than three columns in the actual output file which are only used for debugging purposes. They are not shown in Figure B.2.

```
********************
*
*  N = 41
*  B = 16
*
********************
*
0    1111111110100111  0000000000100110
1    1111111100111110  0000000000100001
2    1111111100101011  0000000000010111
3    1111111101110110  0000000001111111
4    0000000000100100  0000000100101000
5    0000000000010110  0000000100101001
6    1111111011000000  1111111111111000
7    1111111000101111  1111111100100001
8    1111111101001100  0000000011000100
9    0000000110000111  0000001010001000
10   0000000010010100  0000001000011010
11   1111110011111111  1111111010000101
12   1111110000000111  1111110100110100
13   0000000001000000  0000001000110110
14   0000010101001100  0000011000101111
15   0000000011101011  0000001100000010
16   1111011100011100  1111100001111101
17   1111010111011011  1111011110000100
18   0000100011100100  0000101100000010
19   0010011010100100  0010011111011101
20   0011010000110010  0011011010110010
```

Figure B.2. An example output file generated by HMX

An example invocation of HMX from the command line for a filter named “m3” is as follows

HMX m3 41 12

which generates the output file “m3_*n*41.hmx”.

B.3. GAM

This is the implementation of the GAM algorithm. It is a bundle which can solve 5 different linear-phase FIR optimization problems. The synopsis of the program is as follows:

GAM [*ide*] [*N*] [*B*] [*P*] [*O*] [*L*] [*S*]

ide : name of the filter

N : number of taps

B : wordlength

P : maximum SPT count per coefficient (used only for the optimization problem *S* = 0)

O : upper bound for the cost (If you don't need it give it high number)

L : refined value set size

S : type of optimization you want to do. It can be a number between 0 and 4.

- 0 : minimize the number of SPT terms in CSD notation
- 1 : minimize the number of ones in 2's complement notation
- 2 : minimize the number of ones in sign-magnitude notation
- 3 : minimize switching activity between adjacent coefficients in 2's complement notation
- 4 : minimize switching activity between adjacent coefficients in sign-magnitude notation

The program requires two files. The names of the files are not explicitly entered but derived from the arguments. These files are the frequency response specification file (*ide*.frq) generated by the program FRQ and the boundary value specification file (*ide*_n*N*.hmx) generated by the program HMX. The program will prompt an error message and exit in case the files cannot be found.

The program generates three output files for each solution found, namely "*ide*_n*N*b*B*c#_g#.log", "*ide*_n*N*b*B*c#_g#.m", and "*ide*_n*N*b*B*c#_g#.dat" where *ide*, *N*, and *B* are the input arguments described above. The number after 'c' in the file names is the cost of the solution (depends on the type of optimization chosen, for example if we have chosen to optimize the nonzero digits in CSD notation (*S*=0) it would be the total number of nonzero digits in the coefficients). Similarly, the number after 'g' is the solution index. The file with the extension ".m" is a MATLAB m-file consisting of the coefficient array in MATLAB format. The file with the extension ".dat" can be used as an input file to the sub-expression elimination program.

An example run of the program for a filter named "m3" with the parameters *N*=41, *B*=12, *P*=203, *O*=10000, *L*=2, and *S*=3 is

GAM m3 41 12 203 10000 2 3

B.4. CSE

This is the implementation of the sub-expression elimination algorithm of [8]. It is invoked from the command line as

CSE < *coefficient*.dat > *filter_netlist*.txt

where "*coefficients*.dat" is the input file consisting of the coefficients in CSD notation, and "*filter_netlist*.txt" file is the output file generated by the program consisting of a the netlist of the filter. An example "*coefficients*.dat" file is shown in Figure B.3. The first line in the file indicates the number of coefficients. The coefficient wordlength is written in the second line. The following lines consist of the coefficient values written in CSD notation. The letter N in the values corresponds to −1.

```
Tapno= 45
Wordlength= 12
000000000N0N
00000000N000
00000000N00N
00000000N000
000000000N01
000000000100
0000000100N0
00000010N0N0
00000010N010
000000010100
000000001000
0000000N0010
000000N0N001
00000N001000
00000N000001
00000N01000N
00000000N0N0
0000010N0100
000010000010
00010N010100
000100100N00
000101010000
0010N0N00000
000101010000
000100100N00
00010N010100
000010000010
0000010N0100
    .
    .
    .
```

Figure B.3. An example ".dat" file consisting of the filter coefficients in CSD notation

There are five different nodes in the output filter netlist file, namely the input, adders, tap adders, delay elements, and the output. The nodes are also listed in Table B.1 with an example for each. An example filter netlist file generated by CSE is shown in Figure B.4.

Table B.1. Type of nodes in the filter netlist file

Node abbreviation	Node	Example
in	: input	in
a	: adder	a2
t	: tap adder	t3
d	: delay element (register)	d5
o	: output	o41

```
*Solving the model iteratively for the suboptimal solution...
*adders:6

a2 =  (in << 0) - (in << 2)
a3 =  (in << 0) + (in << 2)
a4 =  (in << 0) - (in << 3)
a5 =  (in << 0) + (in << 6)
a7 =  (in << 0) + (in << 3)
a8 =  (in << 0) + (a2 << 2)

d0 =     - (a3 << 0)
t1 =  d0 - (in << 3)
d1 =  t1
t2 =  d1 - (a7 << 0)
d2 =  t2
t3 =  d2 - (in << 3)
d3 =  t3
t4 =  d3 + (a2 << 0)
d4 =  t4
t5 =  d4 + (in << 2)
d5 =  t5
t6 =  d5 - (a4 << 1)
     .
     .
     .
d42 =  t42
t43 =  d42 - (in << 3)
d43 =  t43
t44 =  d43 - (a3 << 0)
o44 =  t44
```

Figure B.4. An example filter netlist file generated by CSE

B.5. VHD

The program VHD generates a VHDL netlist from the filter netlist file generated by the program CSE. The synopsis of the program is as follows

VHD [*filename*] [input wordlength]

where the *filename* is the name of the file containing the filter netlist. It should have an extension of ".txt". That is the netlist file name should be "*filename*.txt". The input wordlength is the number of input bits of the filter. The program generates an output file named "*filename*.vhd" with its subcomponents containing the VHDL descriptions of the adders, subtractors, and registers used in the filter. An example run of VHD from the command line for a filter with eight bit input wordlength is as follows

VHD m3_n41b12c52_g2 8

which generates the output file "m3_n41b12c52_g2.vhd".

B.6. EQUGAM

The EQUGAM algorithm is implemented in MATLAB under the name equ_gam. The synopsis of the program is shown in Figure B.5. The program requires at least five arguments which are a unique identifier of the problem *prbname*, the channel coefficient vector h, signal-to-noise ratio in decibels *SNR*, the equalizer length M, and the coefficient wordlength B. The rest of the arguments are optional. The first optional argument is *jmax*. It refers to the maximum MSE value of the coefficients to be found. If omitted it defaults to the MSE value of the B-bit rounded MMSE coefficients for the given *SNR* and equalizer length M. The second optional argument is the maximum number of SPT terms allowed for each coefficient which is denoted as P. If omitted P is taken to be equal to B. The third optional argument is O which corresponds to the maximum total number of SPT terms int the coefficients. The next three optional arguments L, *SPTPredict* and *ValueOrder* determine the refined value set size, cost prediction strategy, and value selection strategies respectively.

The program generates three output files for each solution found, namely

*prbname*_s*SNR*n*M*b*B*c#_g#.log
*prbname*_s*SNR*n*M*b*B*c#_g#.m
*prbname*_s*SNR*n*M*b*B*c#_g#.dat

where *prbname*, *M*, and *B* are the input arguments described above. The number after 'c' in the file names is the cost of the solution (the total number of SPT terms in the coefficients). Similarly, the number after 'g' is the solution index. The file with the extension ".m" is a MATLAB m-file consisting of the coefficient array in MATLAB format. The file with the extension ".dat" can be used as an input file to the sub-expression elimination program CSE.

```
function [wgam] = equ_gam(prbname, h, SNR, M, B, varargin)
%
%
% [wgam] = equ_gam(prbname, h, SNR, M, B, varargin)
%
%
% wgam : optimum equalizer coefficient vector
%
%%%%%%%
%
% prbname : a unique identifier for the problem
%
% SNR  : signal-to-noise ratio in dB
% h    : channel coefficient vector
% M    : Number of equalizer coefficients
% B    : coefficient wordlength
%
%
% varargin : (1) jmax, (2) P, (3) L, (4) O, (5) SPTPredict,
%            (6) ValueOrder
%
% jmax : maximum error % default = the MSE of B-bit rounded MMSE
%        coefficients
%    P : maximum number of SPT terms allowed in CSD representation
%        of a coefficient % default = B
%    L : refined value set size % default = 2
%    O : maximum number of SPT terms
%
% SPTPredict = 0 : U_MIN
%            = 1 : U_MID % rounded to (B-1) bits % default = 1
%            = 2 : U_SUB_AVG
%            = 3+: U_AVG
%
% ValueOrder = 0 : order values by their SPTcount
%            = 1 : order values by their closeness to mid value
%          default = 1
```

Figure B.5. Synopsis of the MATLAB function equ_gam

An example run of the algorithm from the MATLAB command window is as follows

```
equ_gam('PROAKIS_B', [0.407 0.815 0.407]', 20, 11, 8)
```

where signal-to-noise ratio SNR = 20 dB, equalizer length M = 11, and coefficient wordlength B = 8.

REFERENCES

1. Rabaey, J., Digital Integrated Circuits, Prentice Hall, 1996

2. Lee, M. T. C., V. Tiwari, S. Malik, and M. Fujita, "Power analysis and minimization techniques for embedded DSP software," *IEEE Transactions on VLSI Systems*, pp. 123-135, June 1997.

3. Parhi, K. K., Digital Signal Processing Systems, New York: Wiley, 1999.

4. Mehendale, M., SD. Sherlekar, and G. Venkatesh, "Low power realization of FIR filters on programmable DSP's," *IEEE Transactions on VLSI Systems*, Vol. 6, pp. 546-553, Dec. 1998.

5. Pearson, D. N. and K. K. Parhi, "Low power FIR digital filter architectures," Proc. 1995 *IEEE International Symposium on Circuits and Systems* - ISCAS 95, pp. 231-234, Apr. 1995.

6. Hartley, R., "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Trans. Circuits Systs. II, Analog Digit. Signal Process*., vol. 43, pp. 677-688, Oct. 1996.

7. Pasko, R., P. Schaumont, V. Derudder, S. Vernalde, and D. Iuraekova, "A new algorithm for elimination of common subexpressions," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst*., vol. 18, pp. 58-68, Jan. 1999.

8. Yurdakul, A. and G. Dündar, "Fast and efficient algorithm for the multiplierless realisation of linear DSP transforms," *Circuits, Devices and Systems, IEE Proceedings*, vol 149, pp. 205-211, 2002.

9. Yao, C.-Y., H.-H. Chen, C.-Y. Chien, and C.-T. Hsu, "A high-level synthesis procedure for linear-phase fixed-point FIR filters with SPT coefficients," *International Journal of Electrical Engineering*, vol. 12, no. 1, pp. 75-84, 2005.

10. Vinod, A. P. and E. M-K. Lai, "On the implementation of efficient channel filters for wideband receivers by optimizing common subexpression elimination methods," *IEEE Trans. Computer-Aided Design Integr.* Circuits Syst., vol. 24, no. 2, pp. 295-304, Feb. 2005.

11. Yli-Kaakinen, J. and T. Saramäki, "A systematic algorithm for the design of multiplierless FIR filters," in Proc. *IEEE Int. Symp. Circuits and Systems*, Sydney, Australia, May 2001, vol. 2, pp. 185-188.

12. Gustafsson, O., H. Johansson, and L. Wanhammar, "An MILP approach for the design of linear-phase FIR filters with minimum number of signed-power-of-two terms," in Proc. *European Conf. Circuit Theory Design*, Espoo, Finland, Aug. 2001.

13. Zhao, Q.F. and Y. Tadokoro, "A simple design of FIR filters with power-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 566-570, May 1988.

14. Samueli, H., "An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1044-1047, July 1989.

15. Yao, C.-Y. and C.-J. Chien, "A partial MILP algorithm for the design of linear-phase FIR filters with SPT coefficients," *IEICE Trans. Fundamentals*, vol. E85-A, pp. 2302-2310, Oct. 2002.

16. Gustafsson, O. and L. Wanhammar, "Design of linear-phase FIR filters combining subexpression sharing with MILP," in Proc. *IEEE 2002 45th Midwest Symposium on Circuits and Systems*, Tulsa, OK, Aug. 2002, vol. 3, pp. 9-12.

17. Li, D., J. Song, and Y. C. Lim, “A polynomial-time algorithm for designing digital filters with power-of-two coefficients,” in Proc. *IEEE Int. Symp. Circuits and Systems*, Chicago, May 1993, vol. 1, pp. 84-87.

18. Chen, C. and A. N. Willson Jr., “A trellis search algorithm for the design of FIR filters with signed-powers-of-two coefficients,” *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process*., vol. 46, pp. 29-39, Jan. 1999.

19. Kodek, D. M., “Design of optimal finite wordlength FIR digital filters using integer programming techniques,” *IEEE Trans. Acoust., Speech, Signal Process*., vol. ASSP-28, pp. 304-308, June 1980.

20. Lim, Y. C. and S. R. Parker, “Discrete coefficient FIR digital filter design based upon an LMS criteria,” *IEEE Trans. Circuits Syst*., vol. CAS-30, pp. 723-739, Oct. 1983.

21. Lim, Y. C. and S. R. Parker, “FIR filter design over a discrete powers-of-two coefficient space,” *IEEE Trans. Acoust., Speech, Signal Process*., vol. ASSP-31, pp. 583-591, June 1983.

22. Lim, Y. C. and S. R. Parker, “Design of discrete-coefficient-value linear-phase FIR filters with optimum normalized peak ripple magnitude,” *IEEE Trans. Circuits Syst*. vol. 37, pp. 1480-1486, Dec. 1990.

23. Mehendale, M., S. D. Sherlekar, and G. Venkatesh, “Coefficient optimization for low power realization of FIR filters,” in Proc. *IEEE Workshop on VLSI Signal Processing*, pp. 352 - 361, 1995.

24. Masselos, K., P. Merakos, S. Theoharis, T. Stouraitis, and C. E. Goutis, “Power Efficient Data Path Synthesis of Sum-of-Products Computations,” *IEEE Transactions on VLSI Systems*, vol. 11, No. 3, pp. 446-450, June 2003.

25. Erdogan, A. T. and T. Arslan, "Low power implementation of linear-phase FIR filters for single multiplier CMOS based DSPs," in *IEEE ISCAS*, California, USA, pp. D425-D428, May 1998.

26. Gustafsson, O. and L. Wanhammar, "Design of linear-phase FIR filters with minimum Hamming distance," in *IEEE Nordic Signal Processing Symp*., Hutigruten, Norway, Oct. 4-7, 2002.

27. McClellan, J. H., T. W. Parks, and L. R. Rabiner, "A computer program for designing optimum FIR linear-phase digital filters," *IEEE Trans. Audio Electroacoust*., vol. AU-21, pp. 506-526, Dec. 1973.

28. Adams, J. W., "FIR Digital Filters with Least-Squares Stopbands Subject to Peak-Gain Constraints," *IEEE Transactions on Circuits and Systems*, vol. 39 No. 4, pp. 376-388, Apr. 1991.

29. Steiglitz, K., T. W. Parks, and J. F. Kaiser, "METEOR: A constraint-based FIR filter design program", *IEEE Trans. Signal Process*., vol. 40, pp. 1901-1909, Aug. 1992.

30. Jain, R. J., P. T. Yang, and T. Yoshino, "FIRGEN: A computer-aided design system for high performance FIR filter integrated circuits," *IEEE Trans. Signal Process*., vol. 39, pp. 1655-1668, July 1991.

31. Hawley, R. A., B. C. Wong, T. Lin, J. Laskowski, and H. Samueli, "Design Techniques for silicon compiler implementations of high-speed FIR digital filters," *IEEE J. Solid-State Circuits*, vol. 31, pp. 656-667, May 1996.

32. Fox, T. W. and L. E. Turner, "The design of peak constrained least squares FIR filters with finite precision coefficients," *IEEE Transactions on Circuits and Systems II*, Vol. 49, No. 2, pp. 151-154, Feb. 2002.

33. Suckley, D., "Genetic algorithm in the design of FIR filters," *IEE Proc*., vol. 138, pp. 234-238, Apr. 1991.

34. Cemes, R. and D. Ait-Boudaoud, “Genetic approach to design of multiplierless FIR filters,” *Electronic Letters*, vol. 24, pp. 2090-2091, Nov. 1993.

35. Wade, G., A. Roberts, and G. Williams, “Multiplier-less FIR filter design using a genetic algorithm,” *IEE Proc.-Vis. Image Signal Process*., vol. 141, pp. 175-180, June 1994.

36. Xu, D. J. and M. L. Daley, “Design of optimal digital filter using a parallel genetic algorithm,” *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process*., vol. 42, pp. 673-675, Oct. 1995.

37. Cen, L., “A hybrid genetic algorithm for the design of FIR filters with SPoT coefficients,” *Signal Processing 87*, pp. 528-540, Mar. 2007.

38. Haseyama, M. and D. Matsuura, “A filter coefficient quantization method with genetic algorithm included simulated annealing,” *IEEE Signal Process*. Lett., vol. 13, pp. 189-192, Apr. 2006.

39. Cho, N. I. and S. U. Lee, “Optimal design of finite precision FIR filters using linear programming with reduced constraints”, *IEEE Trans. Signal Process*., vol. 46, pp. 195-199, Jan. 98.

40. Wanhammar, L., *DSP Integrated Circuits*, Academic Press, 1999.

41. Kodek, D. M. and K. Steiglitz, “Filter-length word-length tradeoffs in FIR digital filter design,” *IEEE Trans. Acoust., Speech, Signal Process*., vol. ASSP-28, pp. 739-744, Dec. 1980.

42. Avenhaus, E., “On the design of digital filters with coefficients of limited word length”, *IEEE Trans. Audio Electroacoust*., vol. AU-20, pp. 206-212, Aug. 1972.

43. Chan, D. S. K. and L. R. Rabiner, “Analysis of quantization errors in the direct form for finite impulse response digital filters,” *IEEE Trans. Audio Electroacoust*., vol. AU-21, pp. 354-366, Aug. 1973.

44. Crochiere, R. E., “A new statistical approach to the coefficient word length problem for digital filters,” *IEEE Trans. Circuits Syst*., vol. CAS-22, pp. 190-196, Mar. 1975.

45. Niedringhaus, W. P., K. Steiglitz, and D. M. Kodek, “An easily computed performance bound for finite wordlength direct-form FIR digital filters,” *IEEE Trans. Circuits Syst*., vol. CAS-29, pp. 191-193, Mar. 1982.

46. Kodek, D. M., “Performance limit of finite wordlength FIR digital filters,” *IEEE Trans. Signal Process*., vol. 53, pp. 2462-2469, July 2005.

47. QSOPT Linear Programming Solver, http://www2isye.gatech.edu/~wcook/qsopt/, Aug. 2007.

48. Juan, J. and C. Svensson, “High-speed CMOS circuit technique,” *IEEE J. Solid-State Circuits*, vol. 24, pp. 62-70, Feb. 1989.

49. ILOG CPLEX, http://www.ilog.com/products/cplex/, Aug. 2007.

50. Qureshi, S., “Adaptive equalization,” *Proc. IEEE*, vol. 73, pp. 1349-1387, Sept. 1985.

51. Proakis, J. G., *Digital Communications*, 4th edition, McGraw-Hill, 2002.

52. Aoki, M., *Introduction to Optimization Techniques Fundamentals and Applications of Nonlinear Programming*, The Macmillan Company, 1971.

REFERENCES NOT CITED

Aktan, M. and G. Dündar, "Design of Digital Filters for Low Power Applications Using Integer Quadratic Programming," in *Proceedings of PATMOS Conference*, Leuven-Belgium, 2005

Aktan, M., A. Yurdakul, and G. Dündar, "An Algorithm for the Design of Low-Power Hardware Efficient FIR Filters," accepted for publication in *IEEE Trans. Circuits Syst. I, Reg. Papers I*, 2008.