VLSI Low-Power Digital Signal Processing

by

Emad N. Farag

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical Engineering

Waterloo, Ontario, Canada, 1997

0-612-22199-7

Abstract

This thesis reports on new high-level low-power design techniques for digital signal processing for wireless portable systems. Through proper choice and optimization of an algorithm or an architecture, significant power dissipation saving is achieved. Up to an order of magnitude, with little or no degradation in speed or SNR performance, is achieved.

At the heart of these techniques is the minimization of the computational complexity, by the elimination of redundant and irrelevant computations. Redundant computations are extra computations that can be eliminated by applying appropriate transformations to an architecture or an algorithm without changing its functionality. Irrelevant computations are unnecessary computations that can be eliminated by optimizing the datapath width.

The elimination of redundant computations has been applied to the design of a division algorithm. The division algorithm generates the quotient in the minimum signed-digit representation. Hence, the number of addition/subtraction operations is minimized.

A subband coding image compression algorithm with a simplified filtering structure that requires only addition and subtraction operations has been developed. This simplified filtering structure reduces the power dissipation by 23 times. A new vector quantization algorithm, having a simplified decoding structure, has also been developed for this subband coding algorithm.

The increased flexibility and functionality of signal processing in the digital domain is pushing digital signal processing more and more into the arena of high-speed analog signals. To be able to do this high-speed high-resolution analog-to-digital converters are required. Sigma-Delta A/D converters have been known for their high-resolution capabilities using low-precision components.

Parallelism by 4x of analog signal processors is applied to the design of a band-pass Sigma-Delta modulator. The speed of the modulator is increased without increasing the speed requirement of the individual building blocks.

The elimination of redundant and irrelevant computations has been employed in the design of the decimation filter. The decimation filter consists of two parts, the Sinc decimator and a lowpass decimation filter. In the Sinc decimator, the computational redundancy is minimized. The datapath width of the Sinc decimator is optimized to eliminate irrelevant computations.

The lowpass decimation filter employs multiplication minimization, and operation interleaving to reduce the power dissipation. Furthermore, the lowpass decimation filter is designed to be resolution-programmable, allowing the deactivation of the blocks corresponding to the least significant bits when a lower resolution is sufficient. The decimation filter has been designed in a $0.5\mu m$, 3.3 Volt CMOS technology.

Eliminating the pre-filter multiplier substantially reduces the power dissipation of a digital channel selection algorithm. The pre-filter multiplier has been substituted by less computationally complex operators, such as multiplexers and XOR gates. The frequency spectrum is divided into four overlapping frequency bands. This reduces the filter sharpness requirements, and hence contributes to the power saving. This algorithm achieves up to an order of magnitude saving in power dissipation.

Acknowledgements

It is by the grace and power of God the Almighty that I was able to complete this work. All things were made by Him; and without Him was not anything made that was made.

Looking back at the past three years I realize that this work could not have been successfully completed without the supervision, guidance, assistance, encouragement and support of others. First, and foremost, I would like to thank my supervisor Professor Mohamed I. Elmasry, for his valuable suggestions, for his guidance, encouragement and support throughout the program. His assistance has been of great value and is greatly appreciated.

I would also like to express my thanks and gratitude to the members of the Wireless Circuits and Systems department at Bell Laboratories, Lucent Technologies. Particularly, I would like to thank Dr. Ran-Hong Yan, the department head, for his valuable consultations, and for his support during the my internship program there. I would also like to thank Peng-Wen Ong, Eric H. Westerwick, and Donald D. Shugard for their consultations and assistance with the CAD tools.

I would also like to thank Professor M. Anwarul Hasan for his valuable consultations and his encouragement. Thanks is also due to Phil Regier and Anil Rana, the system administrators at the University of Waterloo and at Bell Laboratories respectively, for their valuable computer assistance and support.

I am also deeply indebted to all the members of my family, for their encouragement and genuine support throughout my education life. Especially, I would like to thank my father, my mother, my brothers, Naguib and Maged, and my fiancée, Irene. None of this could have been achieved without their enthusiasm.

The help of all my friends and colleagues both past and current is gratefully acknowledged. Special thanks is due to Essam S. Tony of the University of Waterloo, and Stephan ten Brink of the University of Stuttgart, Germany, for being good friends and for their valuable discussions. I would also like to thank my colleagues at the VLSI Research Group for their useful discussions and for their inspiring ideas.

This research is supported in part by MICRONET and by an Ontario Graduate Scholarship (OGS). This support is greatly appreciated.

But most of all, and above all, thanks to The LORD who makes all things possible.

This work is dedicated to the memory of my grandparents

Dr. Naguib Farag

Dr. Zaki Iskander

Mrs. Fayqua Farag, and

Mrs. Mary Guirguis

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The state-of-the-art in modern telecommunications is most fascinating and intriguing. Technical specialists are competing to digest modern techniques in order to be capable of providing due service to the inspired and ambitious users, to whom technology offers new areas and fields yet to be ventured for the service of mankind.

The last few years witnessed the widespread of portable equipment from cellular phones to multimedia portable terminals. However, these mobile equipments are constrained in computational capability due to battery limitations and size limitations [1]. Over the last 30 years battery capacity has increased by a factor of 2 to 4, while the computational power of digital IC's increased more than 4 orders of magnitude [2]. The energy density of the Ni Cd batteries used in portable terminals is 20 Watt-Hour/Pound [3]. Battery capacity isn't expected to increase dramatically over the next few years. New battery technology such as Nickel-Metal-Hydrite is expected to have a capacity of no more than 30–35 Watt-Hour/Pound.

With the increase in market demand for new capabilities and functionality in mobile equipments, new approaches are required to reduce the power dissipation and hence prevent the battery size from growing in tandem with computational complexity.

Until recently power consumption was not a high priority issue in the design of VLSI systems. Performance (speed) and cost (area) were the two metrics that governed the design of VLSI systems [4]. However, the need for longer battery life in future portable terminals has added power dissipation to the metrics that should be considered when designing a VLSI system.

Mobile applications are not the only factor driving the need for lower power dissipation. Power dissipation in cutting-edge immobile equipment has reached a limit where any further increase in power dissipation will lead to significant increase in the cost of packaging and the cooling system. The addition of a heat sink could increase the component cost by $5–$10 [5]. In addition, large power dissipation leads to lower component reliability. Every $10^{o}C$ increase in temperature doubles the component failure rate [2].

Finally, there are the economical and environmental advantages of reducing the power dissipation. A study in 1993, [6] showed that the 60 million personal computers in the USA dissipated $2 Billion of electricity per year, and that they indirectly produced as much CO_2 as 5 million cars. In 1993 personal computers accounted for 5% of the commercial electricity demand, this is expected to increase to 10% by the year 2000.

Low-power design is finding its way into numerous applications. From portable communication products such as cellular phones, cordless phones and pagers, to portable consumer products such as camcoders and portable CDs, to laptop and notebook computers, to sub-GHz processors for high performance workstations. Design for low power is essential in all these applications.

Reducing the power dissipation can be done at the various levels of the design process, starting at the algorithmic and architectural levels and going down to the circuit and device levels [5]. The power minimization problem at each one of these levels has different characteristics and meets different challenges. At the higher design levels, the designer faces alternative choices with little information about the design parameters of the lower layers. At the lower design levels, the number of parameters is limited making the low-power design problem easier.

However, implementation of the low-level low-power design techniques requires greater investment and longer time to implement than the high-level low-power design techniques. Consider, for example, process scaling as a technique to reduce power dissipation. This requires a greater investment and a longer time to implement than changing the algorithm as a means of reducing the power dissipation.

Despite their great potential for reducing the power dissipation, high-level techniques are the least investigated techniques. Selecting the suitable algorithm and mapping it to the appropriate architecture can have a great influence on the minimization of power dissipation [7] [8]. Eliminating redundant and irrelevant computations has a substantial effect on the reduction of the power dissipation.

Future portable terminals are required to handle multimedia information — speech, video and data [8]. Because of the limited bandwidth allocated to mobile systems, compression/decompression of information is required in mobile terminals. Compression algorithms and in particular video compression algorithms demand large computation capability [9] which in turn leads to high power dissipation. The desire to have multimedia portable equipment has motivated work towards low-power implementations of video compression algorithms.

Increased public demand for higher performance, better quality of service, and system interoperability, has motivated the idea of software radio [10]. In software radio, the digitization of the received/transmitted radio signal is performed as electrically close to the antenna as possible. The signal processing is done digitally after that, using a general purpose programmable hardware.

Software radios require wideband (high-speed) high-resolution analog-to-digital converters [11]. They also require high DSP horse-power (up to 10 GFLOPS/s) [12]. The desire to have a portable software radio has motivated work towards lowering the power of wideband high-resolution A/D converters. It has also motivated work towards the development of DSP algorithms with lower computational complexity to be used in the software radio.

The objective of this thesis is to investigate, develop, design and implement low-power techniques for portable wireless terminals at the architectural and the algorithmic levels. The low-power techniques are applied to video compression algorithms used in multimedia portable terminals. Low-power algorithms are also developed to lower the power dissipation in software radios.

1.1 Thesis Contributions

1. Analysis of high-level low-power design tradeoffs. Three examples of such analysis are given in sections 3.6.2, 3.8 and 3.9.1 of chapter 3. These are; the use of carry save adders in FIR filters, the use of the Gray code number system, and the use of higher-order radix in the division algorithm.

2. A new division algorithm that minimizes the number of addition/subtraction operations required to generate the quotient. This algorithm is presented in section 3.9.2 of chapter 3.

3. A new low-power subband coding image compression algorithm developed in chapter 4. The filtering structure for the proposed subband coding algorithm, requires only addition/subtraction operations, this significantly reduces the power dissipation. A novel vector quantization coding algorithm having a simplified decoding architecture has been developed in chapter 4.

4. A novel bandpass Sigma-Delta modulator, along with its switched-capacitor implementation, presented in chapter 5. Parallelism by 4x of analog signal processors is applied to the design of the bandpass Sigma-Delta modulator. This increases the speed of the modulator without increasing the speed requirement of the individual building blocks. A switched-capacitor circuit with a minimum number of operational amplifiers is also given for the proposed modulator architecture.

5. The design of a decimation filter incorporating several low-power design techniques such as; operation minimization, multiplier elimination, and block deactivation. The decimation filter is resolution-programmable, allowing the deactivation of the blocks corresponding to the least significant bits, when a lower resolution is sufficient. The design of the decimation filter is given in chapter 5.

6. The design of a resolution-programmable multiplier-accumulator (MAC) array. The interleaving of the adder in the multiplier array reduces the power dissipation. The resolution of the MAC array is programmable allowing the deactivation of the blocks corresponding to the least significant bits when a lower resolution is sufficient. The design of the MAC array is given in chapter 6.

7. A novel digital channel selection algorithm with no pre-filter multiplier. The

channel selection algorithm uses lowpass, highpass and bandpass filters. The basic filter is the lowpass filter. Other filters are implemented using the lowpass filter and simple logic gates such as multiplexers and XOR gates. The channel selection is done in stages. The elimination of the pre-filter multiplier reduces the power dissipation. The design of the digital channel selection algorithm is given in chapter 7.

1.2 Thesis Outline

The thesis consists of eight chapters and two appendices. Chapters 2 and 3 are a survey of wireless architectures and standards, and low-power design techniques. Chapters 4 – 7 present the main contributions of this thesis for the high-level low-power design of multimedia wireless terminals. A person interested in power-efficient design of multimedia terminals can proceed directly to chapter 4.

After the introduction, which provides for the motivation and a brief description of the thesis, chapter 2 deals with wireless communication systems. It talks about the different standards for voice and data wireless communications. The transceiver architecture is reviewed in this chapter. The emerging software radio architecture is also presented in this chapter.

In chapter 3, low-power design techniques are explored. In this chapter, the sources of power dissipation are investigated, and power estimation methods are considered. Low-power design has recently captured the attention of many researches. A survey of low-power techniques employed in the design of portable equipment is presented in this chapter. Also in this chapter, the application of low-power techniques to the design of certain algorithms and architectures is investigated.

In chapter 4, a new low-power subband coding image compression algorithm is presented. Subband coding is a technique in which the video signal is divided into subbands and each subband is allocated a number of bits according to the information it carries and its spectral importance. Tradeoffs between the computational complexity (power dissipation) and the signal-to-noise ratio (SNR) performance of the subband coding algorithm are considered, and an algorithm with low computational complexity is presented. Finally, the performance of this algorithm is evaluated.

In chapter 5, a high-speed high-resolution A/D converter is presented. In the first part of this chapter, a novel bandpass Sigma-Delta modulator architecture is developed. In this architecture parallelism by 4x of analog signal processors is applied to the design of the bandpass Sigma-Delta modulator. The switched-capacitor implementation of the proposed architecture is also presented in this chapter. In the second part of the chapter, several low-power design techniques are applied to the design of the decimation filter. These techniques include, operation minimization, multiplier elimination and block deactivation. The design of the decimation filter in a $0.5\mu m$, 3.3 Volt CMOS technology is also presented in this chapter.

In chapter 6, the design of a new resolution-programmable multiplier-accumulator (MAC) array is presented. The multiplier of the MAC array is based on the modified Booth algorithm. The accumulator's input and output are in the sum-carry representation. The effect of interleaving the adder in the multiplier array on reducing the power dissipation is discussed in this chapter. To further reduce the power dissipation a block deactivation architecture is developed, where the cells corresponding to the least significant bits are deactivated when a smaller resolution is sufficient. The design of the MAC array in a $0.5\mu m$, 3.3 Volt CMOS technology is also presented in this chapter.

In chapter 7, the design of a novel power-efficient digital channel selection algorithm is presented. In software radio, a block of channels is digitized, the channel selection is performed in the digital domain. Conventionally, channel selection is done by a multiplier followed by a lowpass filter. The multiplier operates at a high sampling rate and hence, it dissipates a large amount of power. A novel digital channel selection algorithm is developed that eliminates the pre-filter multiplier, this can reduce power dissipation by up to an order of magnitude.

Chapter 8 contains the summary of the research, along with the major contributions of this dissertation. Also contained in this chapter are the conclusions reached after conducting this research. Finally, future directions in research for power minimization at the algorithmic and architectural levels for future portable wireless terminals are discussed.

Appendix A presents the SPW^{TM} simulation model of the bandpass Sigma-Delta modulator. Appendix B presents an analysis for the Sinc decimator.

Chapter 2

Wireless Communication Systems

The last decade witnessed an explosion in the development and the commercialization of wireless communication products, such as cellular phones, cordless phones, pagers, wireless LAN and WAN terminals, etc. This development was fueled by the acceptance of the wireless communication standards, the advancement in wireless circuit design techniques and high-speed monolithic IC technology, as well as the development of new wireless system architectures.

New services and features are now being envisioned for future mobile communication systems. These systems will allow users to have access to information databases and to communicate in any form of media — voice, video, images or data — at any time and in any place [1] [13]. Increased public demand for better performance, higher quality of service and lower costs has led to the development of new wireless communication techniques and new wireless transceiver architectures.

In the last few years, there has been a shift from analog to digital communication techniques. This shift led to enhanced performance and lower cost. Table 2.1 compares the analog communication techniques to the digital ones. In the future,

Table 2.1: Comparison of analog and digital communication techniques.

Criterion	Analog Communication Techniques	Digital Communication Techniques
Technology	Discrete/Hybrid	Monolithic
Modulation	FM	QPSK, GMSK ...
Access	FDMA	TDMA, CDMA
Noise resilience	Low	High
Capacity (Spectral efficiency)	Low	High

this digitization trend is expected to continue moving into the front-end of the transceiver, and eventually leading to a true software radio.

There exists numerous standards for wireless communication systems throughout the world. These standards regulate the spectrum usage and define the key parameters of the different wireless systems, such as the cellular, cordless and wireless data systems. In section 2.1, we review the standards used in analog and digital cellular systems, as well as wireless data systems.

Traditionally, the superheterodyne principle has been used in the design of wireless receivers since its discovery by Armstrong in the 20's [14] [15] [16]. With consumers demanding more functionality and enhanced performance and with the advancement of the IC technology, interest has been growing in direct conversion receivers [17], as well as software radios [10] [18]. These architectures are examined in sections 2.2 and 2.3.

Table 2.2: Major analog cellular standards.

Standard	Up link (MHz)	Down link (MHz)	Channel spacing (KHz)	Number of channels	Region
AMPS	824 — 849	869 — 894	30	832	Americas Africa Australia Far East
TACS	890 — 915	935 — 960	25	1000	Europe Africa Far East
NMT 900	890 — 915	890 — 915	12.5	1999	Europe Africa Asia
NTT	925 — 940 915 — 918.5 922 — 925	870 — 885 860 — 863.5 867 — 870	25/6.25 6.25 6.25	600/2400 560 480	Japan

2.1 Land Mobile Wireless Systems Standards

Cellular system design was pioneered by Bell Laboratories in the 70's [19]. The first generation of cellular systems used analog frequency modulation. Frequency Division Multiple Access was used to divide the spectrum between the different users. Table 2.2 gives the salient features of the major analog cellular standards [19].

The desire for larger system capacity and better performance, coupled with the

Table 2.3: Digital cellular standards.

Criterion	IS-136 (DAMPS)	IS-95	GSM	PDC
Up link	824 — 849	824 — 849	890 — 915	940 — 956*
Down link	869 — 894	869 — 894	935 — 960	810 — 826*
Channel spacing	30 KHz	1.25 MHz	200 KHz	25 KHz
Number of channels	832	20	124	1600
Multiple access	TDMA	CDMA	TDMA	TDMA
Modulation	$\pi/4$ DQPSK	QPSK	GMSK	$\pi/4$ DQPSK
Channel bit rate	48.6 Kb/s	1.2288 Mb/s	270.833 Kb/s	42 Kb/s

* More spectrum is allocated around 1.5 GHz.

advancement of digital integrated circuit design and low bit rate speech coding algorithms led to the emergence of the second generation cellular systems which use digital modulation techniques [20]. Digital cellular systems are more efficient in their spectral usage than analog cellular systems. There exists different digital cellular standards, Table 2.3 gives the salient features of these standards [21].

Each standard defines the control signals, and the minimum performance requirements for the mobile terminal and the base station. For the DAMPS standard, the mobile terminal is required to satisfy the following minimum requirements [22]:

Adjacent channel selectivity :

Assigned channel = -107 dBm

Adjacent channel = -94 dBm

Error rate = 3%

Alternate channel selectivity :

Assigned channel = -107 dBm

Second adjacent channel = -65 dBm

Error rate = 3%

Intermodulation Spurious Response Attenuation :

Assigned channel = -107 dBm

Unmodulated RF @ $\pm$ 120 KHz = -45 dBm, OR

Modulated RF @ $\pm$ 240 KHz = -45 dBm

Error rate = 3%

Spurious Response Interference :

Assigned Channel = -107 dBm

Undesired RF = -52 dBm

Undesired RF modulated in cellular band, unmodulated elsewhere.

Error rate = 3%

Except within 90 KHz of the assigned channel.

In addition to the cellular standards for wireless voice networks, there exists standards for wireless data networks [19] [23]. Wireless data networks are classified into: wide-area mobile data networks, and wireless local area networks (WLAN).

Wide area mobile data networks are low-speed networks. Several standards exist for wide area mobile data networks, such as; Advanced Radio Data Information Service (ARDIS), Mobitex, and Cellular Digital Packet Data (CDPD). The salient features of these standards are given in Table 2.4.

WLANs are high-speed networks but they have a limited coverage area. WLAN use spread spectrum in the unlicensed ISM bands [24]. WLAN standards include IEEE 802.11 in North America and HIPERLAN in Europe.

Table 2.4: Wireless data network standards.

Criterion	ARDIS	Mobitex	CDPD
Up link	806 — 824	896 — 901	824 — 849
Down link	851 — 869	935 — 940	869 — 894
Channel spacing	25 KHz	12.5 KHz	30 KHz
Number of channels	720	480	832
Modulation	FSK	0.3-GMSK	0.5-GMSK
Channel bit rate	19.2 Kb/s	8.0 Kb/s	19.2 Kb/s

2.2 Wireless Transceiver Architectures

A wireless transceiver consists of a transmitter and a receiver. The duplexer is used for directing each of the transmit/receive signals to its intended path. The incoming RF signal from the antenna is directed to the receiver, while the transmitted signal is directed from the transmitter to the antenna with no coupling to the receiver.

The transmitter converts the baseband signal to the RF carrier frequency. It can do so using a single-stage quadrature modulator [25]. This is shown in Figure 2.1. A filter is needed after the power amplifier. This filter removes any out of band frequency components due to the nonlinearity of the power amplifier or spurious frequencies from the oscillator. A transmitter can also have multi-stage mixing.

Figure 2.2 shows the block diagram of the conventional wireless receiver. This receiver is a two-stage superheterodyne receiver [26] [27]. Typically the first IF stage down converts the frequency by an order of magnitude. The first IF stage is used for image rejection. The second IF stage is used for channel selectivity. The first IF stage frequency is about 90 MHz. The second IF stage frequency is

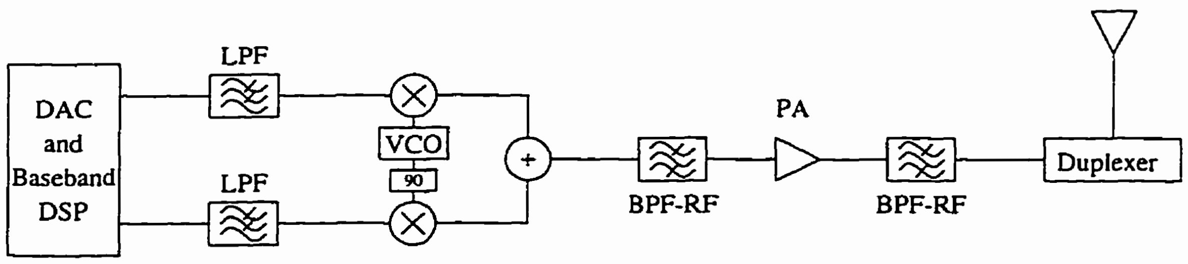


Figure 2.1: A single-stage up conversion transmitter.

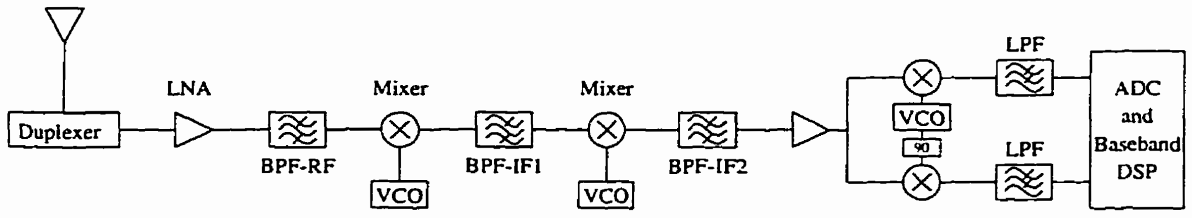


Figure 2.2: A Two-stage superheterodyne receiver.

455 KHz [17].

In the superheterodyne receiver, the IF frequency is fixed. The design of the channel selection filter is easier in the IF band than in the RF band [17], because the center frequency of the IF filter is fixed and the relative filter bandwidth with respect to the center frequency is larger in the IF band than in the RF band.

The complexity of having a two-stage IF superheterodyne receiver can be eliminated by using the direct conversion receiver [28] — [34]. The direct conversion receiver is also referred to as the homodyne receiver when the local oscillator is synchronized in phase with the incoming RF signal carrier [17]. The homodyne receiver converts the RF signal to baseband directly, eliminating the need for intermediate IF stages. Figure 2.3 shows the block diagram of the direct conversion (homodyne) receiver. Direct conversion receivers enable the highest level of intergration and require the least amount of tuning [35].

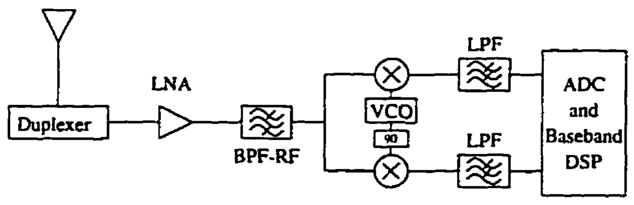


Figure 2.3: Direct conversion (homodyne) receiver.

An alternative direct conversion receiver architecture [36] uses an external band-pass filter with a 90° phase splitter. The same VCO signal is fed to both mixers in this case.

Despite its simplicity, direct conversion receivers have several drawbacks [17] [34] compared to the superheterodyne receiver. These include, the possibility of mismatch between the I and Q paths, carrier leakage and DC feed-through, sensitivity to the 1/f noise, limited dynamic range, and finally, back radiation of the receiver's local oscillator signal.

Having different standards creates a need for a multistandard transceiver architecture. In [37], the authors consider using a zero-IF receiver and an image rejection receiver to achieve multistandard operation for DECT, GSM/DCS1800 and INMARSAT M. Another way to achieve a multistandard receiver is to use software radio. Software radio is introduced in the next section.

2.3 Software Radio

In software radio [10] [18], the received/transmitted radio signal is digitized as electrically close to the antenna as possible. The signal processing is done digitally after that, using general purpose programmable hardware. Performing the radio, IF and baseband functions in programmable digital hardware increases the flexibility of the transceiver. Although software radios use digital techniques, digital radios are generally not software radios. The key difference is the total programmability of software radios, including programmable RF bands, multiple access modes, and modulation schemes.

Software radio was conceived in the 70's. However, technology limitations prevented it from being implemented. The first operational digital high frequency communications system was built in 1980 [38]. It was used by the military. That system occupied many racks, it dissipated a large amount of power, and it only had a bandwidth, for simultaneous coverage, of 750 KHz, with a dynamic range of 60 dB.

Today the US military is in phase II of developing its software radio - Speakeasy [18] [39]. Speakeasy is a programmable multi-band multi-mode radio (MBMMR) that operates in the HF to the UHF bands, from 2 MHz to 2 GHz. Speakeasy emulates 15 existing military radios. It supports 9 modulation schemes, and 4 digital audio coding algorithms. It also supports multiple internetworking protocols, multiple interfaces, multiple forward error correction codes and multiple information security (INFOSEC) algorithms.

For civil applications, software radio is used in cutting-edge base stations. The design of portable terminals is a compromise between low-power and high-performance, this involves a tradeoff between analog ICs, low-power ASICs, DSP cores and embedded microprocessors [10]. However, as low-power techniques and design methodologies emerge, digital signal processing will gradually replace analog signal processing in the wireless portable terminal.

There are numerous advantages to increasing the portion of the radio that is implemented digitally. These include relaxing the analog components requirements.

Digital implementations tend to be compact and inexpensive for large volume production. One of the most important advantages is the ability to program digital structures to meet the communication needs of different networks using a single hardware platform.

The access, modulation and coding schemes used in a software radio are programmable, making it possible to reprogram the transceiver if any of these schemes change. Channel selection, propagation channel characterization, antenna steering and power level adjustment are all done under software control [10]. In the transmit mode, the software radio characterizes the available channels, steers the transmit beam in the right direction, selects the appropriate power level and than transmits the signal. In the receive mode, the software radio analyzes the received spectrum, in frequency, time and space. It identifies the interferers and nulls them. It estimates the multi-path propagation channel model and adaptively equalizes the received signal. The signal is then demodulated and decoded.

Software radio is characterized by its modular, open architecture allowing constant upgrades as the technology advances. The software radio architecture, as shown in Figure 2.4, consists of three subsystems. The real-time channel processing subsystem is where all the radio functions are performed. This subsystem must have isochronous performance, which means that the input samples must be processed during a limited time duration. The environment management subsystem constantly characterizes the radio environment. This information is used by the channel processing subsystem for better transmission and reception. The environment management subsystem has near real-time operation. The software tools subsystem provides incremental service enhancements. This subsystem allows, defining, prototyping, testing and delivering these service enhancements.

To implement software radio the entire spectrum of a particular standard should

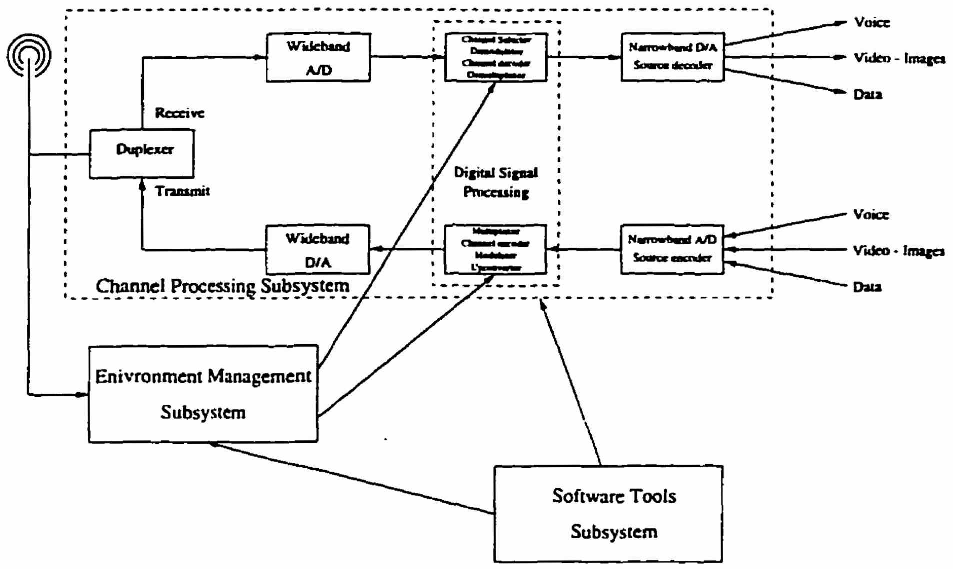


Figure 2.4: The software radio architecture.

be digitized. This is 25 MHz for GSM, IS-136 and IS-95, as given in Table 2.3. To digitize a 25 MHz bandpass signal, bandpass sampling is used [11]. To satisfy the Nyquist sampling criteria, the sampling frequency should be at least twice the bandwidth. Assuming the sampling frequency is 2.5 times the bandwidth, then a sampling frequency of 62.5 MSa/s is required. To meet the requirements of the different wireless standards the A/D converter is required to have over 20 bits resolution (this will be shown later in Chapter 5). This is the first bottleneck facing software radio, i.e. high-speed high-resolution analog-to-digital conversion. In chapter 5, the design of an A/D converter that has a sampling frequency of 1.25 MHz and a programmable resolution up to 20 bits is examined.

In addition to the high-speed, high-resolution A/D converter, software radio also requires high DSP horsepower. Typically Software Radio requires up to 10 GFLOPS/s [12]. Such high processing power is beyond the capabilities of todays DSPs. This is the second bottleneck facing software radio. In chapter 7, a digital channel selection algorithm, that can be employed in software radios to reduce the computational complexity required for digital channel selection, is presented.

2.4 Chapter Summary

In this chapter, the salient features of analog and digital cellular standards, as well as the wireless data standards were presented. The superheterodyne principle has been commonly used in conventional transceivers. The advancement of the IC technology, and the demand for enhanced performance has led to the emergence of new architectures, such as the homodyne architecture and the software radio.

Software radio, which allows more functionality and programmability, is currently being used for military applications and in cutting-edge base stations. For the mobile terminals, new low-power techniques need to be developed to make software radio a power-efficient architecture that can compete with the superheterodyne and homodyne radio architectures.

Chapter 3

Low-Power Design Techniques

3.1 Introduction

In this chapter, the techniques used to lower the power dissipation at the architectural and algorithmic design levels are investigated. These are the higher design levels, as opposed to the device and circuit levels, the lower design levels.

The organization of this chapter is as follows, section 3.2 talks about the sources of power dissipation in CMOS circuits and the parameters they depend on. In section 3.3, the model used in the estimation of the power dissipation is presented. In section 3.4, low-power examples for: wireless portable systems, digital signal processors, video compression algorithms and microprocessors are presented. Low-power techniques, used at the device and circuit levels, are presented in section 3.5.

The quadratic dependency of the power dissipation on the voltage makes voltage reduction an effective way to reduce the power dissipation. This is examined in section 3.6. However, reducing the voltage leads to longer delays. Techniques used to maintain a constant throughput with voltage scaling are also considered in section 3.6. Section 3.7 demonstrates the effect of pipelining and parallelism, by applying these techniques to three different architectures of the discrete cosine transform.

Reducing the switching activity is another degree of freedom in reducing the power dissipation. In section 3.8, the effect of the Gray code number system on reducing the switching activity is considered. In section 3.9, the effect of reducing the number of block iterations on the power dissipation of the division algorithm is considered. Two examples demonstrate this. First, a higher order radix is used. Second, a division algorithm is developed, requiring a minimum number of add/sub operations. In section 3.10, reducing the computational complexity of the vector quantization algorithm is considered.

3.2 Sources of Power Dissipation

The power dissipated in an electronic system depends on the implementation technology and the circuit style used. Current mode BJT and NMOS have DC (static) power dissipation, while CMOS almost has no DC power dissipation, making its power dissipation lower than the two former technologies. The CMOS style is the most commonly used style for the implementation of VLSI systems.

In CMOS circuits, there are three sources of power dissipation [40]:

1. Switching power dissipation.
2. Short-circuit-current power dissipation.
3. Leakage-current power dissipation.

The most dominate of these is the switching power dissipation which is given by [41]:

$$P_{switching} = \alpha_{0\to 1} C_L V_{DD}^2 f_{clk} \tag{3.1}$$

The **switching power dissipation,** as seen from the previous equation, depends on four parameters. The switching activity factor $\alpha_{0\to 1}$, the load capacitance C_L, the supply voltage V_{DD} and the clock frequency f_{clk}. Of these, the supply voltage has the greatest effect on the switching power dissipation because of the quadratic dependence. In a well designed CMOS circuit, the switching power dissipation accounts for 90% of the power dissipation.

The switching activity factor $\alpha_{0\to 1}$, the probability of a zero-one transition, depends on:

1. Logic function. For example, a NAND gate with equi-probable and independent inputs has
$$\alpha_{0\to 1} = \frac{3}{16},$$
while an XOR gate, with equi-probable and independent inputs has
$$\alpha_{0\to 1} = \frac{1}{4}.$$

2. Logic style. Dynamic logic has higher switching activity than static logic, because the output is precharged at the end of each cycle. However, dynamic logic is glitch free. The logic style also influences the capacitances.

3. Signal statistics. The higher the correlation between the successive samples the lower the switching activity.

4. Circuit topology. e.g. chain structure versus tree structure. A Chain structure has lower switching activity, but higher glitching power.

The **short-circuit-current power dissipation,** unlike the switching power dissipation, depends on the rise and fall times of the input signal. To minimize the effect of the short-circuit power dissipation it is desirable to have equal input and output edge times [42]. In this case, the power dissipation is less than 10% of the total dynamic power dissipation.

The **leakage-current power dissipation** is due to:

1. Reverse-bias diode leakage current. This is in the order of 25 μA for a 1 million transistor chip. Hence, it represents a negligible component of the power dissipation [41].

2. Subthreshold current. Associated with this is the subthreshold slope S_{th}, which is the voltage required to reduce the subthreshold current by an order of magnitude [43]. The absolute minimum of S_{th} is about 60 mv/(decade current) at room temperature. This can be achieved using Silicon-On-Insulator (SOI) technology [44]. Lowering the subthreshold voltage increases this component.

Of these three power dissipation components, the switching power dissipation component is the most dominant [41]. Hence, this is the component we usually seek to minimize, especially at the architectural and algorithmic levels.

3.3 Estimating the Power Dissipation

Power estimation can be a complex task. Not only does it require knowledge about the technological parameters of the system under consideration such as the operating voltage, the physical capacitance, the circuit style, etc., but it also requires

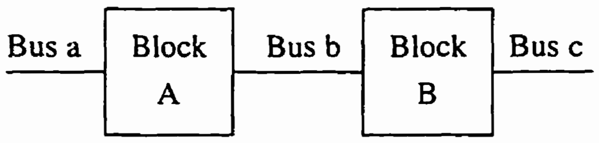


Figure 3.1: A simplified system consisting of two building blocks and the interconnection busses.

detailed knowledge of the signal statistics such as the data activity and the signal correlations.

The aim of power estimation is to find the average power dissipated in a system based on a certain model. Power estimation becomes more inaccurate as the degree of model abstraction increases. Hence, the most accurate power estimators are the circuit simulators [45]. However, circuit simulators are slow and require complete and specific information about the inputs [46].

Gate-level probabilistic techniques have been proposed, ranging from simple techniques [47] which assume a zero-delay gate model and thus don't calculate the glitching power which can be as high as 70% [48], to more elaborate techniques [49] [50] that not only consider the effect of glitching, but they also take into account the effect of temporal and spatial correlations [46].

The research done on power estimation at the higher abstraction levels is still limited [51] [52]. At the architecture level, the system is described in terms of interconnected operators (adders, multipliers, etc.) and memory blocks (registers, ROMs, etc.). These building blocks, as they will be called from now on, are interconnected by busses. Figure 3.1, shows a simplified architecture consisting of: two building blocks, one interconnecting bus (bus b), one input bus (bus a), and one output bus (bus c).

The total power dissipated in such an architecture is the sum of the power dissipated in the building blocks and the power dissipated in the busses. The power dissipated by a building block depends on:

1. Block activity factor (β) (number of executions per second).
2. Output signal activity factor (α).
3. Normalized block energy E_n. The normalized energy is the energy dissipated by the building block per execution when the signal switching activity factor is one.

The total power dissipated by the building blocks is given by:

$$P_{BB} = \sum_{n \in \Re} \alpha_n \beta_n E_n \tag{3.2}$$

Where $\Re$ is the set of all building blocks. The power dissipated by a bus depends on:

1. Signal activity factor (α).
2. Length of bus (ℓ).
3. Capacitance per unit length (C).

The total power dissipated by the busses is given by:

$$P_{BU} = K \sum_{n \in \aleph} C_n \alpha_n \ell_n \tag{3.3}$$

Where $\aleph$ is the set of all busses. K is some constant that depends on the operating voltage. When determining ℓ_n the dimensions of the building blocks should be taken into consideration.

3.4 Low-Power Examples of Portable Systems

There are numerous low-power techniques used by researchers and designers to lower the power dissipation of portable systems. Some of these techniques, used for the design of wireless portable systems, digital signal processors, video compression algorithms and microprocessors are presented in this section.

The first low voltage, very low current integrated circuits were developed about 25 years ago for the watch [53]. However, for other electronic systems power dissipation was only an afterthought. During the last decade, this has began to change. There has been great interest in the implementation of low-power, small size portable communicators for voice, video, images and data information as well as low-power note-book and lap-top computers [8] [54] [55] [56].

In cellular systems, a considerable fraction of battery energy is used for transmission. Reducing the cell size not only increases the spectrum efficiency through frequency reuse but it also allows operation at lower transmission power levels. This in turn leads to longer battery life. Currently mobile phones operate in a cell of several hundred meters radius, and transmit power in the order of 0.1–1 Watt [57].

The Viterbi decoder, used in CDMA cellular applications, presented in [58], employs various low-power techniques. The squared Euclidean measure has been substituted by a non-squared Euclidean measure. This reduces the complexity of the branch metric unit and the word-length of the path metric unit. The Viterbi decoder presented uses minimum sized processing units. To meet the throughput requirement, parallelism and pipelining are employed. To reduce spurious transitions on high-capacitance busses, gated control signals are used for controlling the multiplexers connected to these busses.

Surviving-path memory management [59] is one of the operations required in the

Viterbi decoder. There are two techniques for surviving-path memory management: exchange register and trace back. In [60], the effect of hybrid techniques on reducing the power dissipation is considered.

In a receiver, the matched filter is positioned between the RF section and the baseband section. Hence, it can be implemented in digital or in analog technology. The effect of each implementation on the power dissipation is considered in [61]. In turns out that for slow matched filters, with a large number of taps and high precision, the digital implementation is more power efficient than the analog one.

By lowering the supply voltage from 5 volts to 1.5 volts, the power dissipation of different digital filters has been lowered by 8–11 times [62]. Architectural transformations such as parallelism, associativity, distributivity, commutativity, operation substitution and bit width optimization were used to maintain a constant throughput, lower the glitching activity, and reduce the interconnect capacitance.

In [63], a low-voltage low-power DSP is designed. The operating speed of the DSP is 63 MHz at 1 Volt. The power dissipation at this voltage and speed is 17.0 mW. During active operation of the DSP, power saving is realized by the use of locally gated clocks. Global gating is also available and it is controlled by three power-down instructions. The memory is divided into 8 arrays, only one array is activated during each memory access. A multi-level threshold voltage, V_T, is used. High V_T is used in the 6 transistors of the memory cells to lower the standby current. Low V_T is used in the peripheral circuitry to allow high-speed operation at 1 Volt.

To lower the power dissipation in the lowpass interpolation and decimation filters, the filter order is adapted according to the input and output signal characteristics [64]. This avoids the use of higher order filters when a lower order is sufficient. Powering-down control is used in the ALU when a lower order is sufficient. The saving in power dissipation achieved for the decimation and interpolation filters is 42% and 21% respectively.

A variable threshold voltage scheme is used in [65], to lower the standby power dissipation in a low V_T CMOS technology. It also mitigates the effect of fluctuations in V_T on the system delay. The threshold voltage is controlled by changing the substrate voltage V_{BB}. For the NMOS, in the active mode $V_{BB} = -0.5$ Volt, and $V_T = 0.1$ Volt. In the standby mode, $V_{BB} = -3.3$ Volt and $V_T = 0.5$ Volt.

A low-power subband video compression algorithm decoder is presented in [66]. Parallelism of the subband algorithm has been exploited to achieve an excess throughput that can be traded for lower power by reducing the supply voltage. Off chip memory is avoided to eliminate the high power consumption of external memory access. An asymmetric wavelet filter is used in the lowpass and highpass filters, the filter uses 3-2 adders in its implementation. For the high-frequency bands, the data is zero run-length encoded to reduce the number of external inputs by almost a factor of 4. At 1.0 Volt, the decoder is capable of operating at a 3.2 MHz real time video rate, and dissipates 1.2 mW.

Unlike standard Vector Quantization (VQ) decoders which require codebook storage, the Pyramid Vector Quantization (PVQ) decoder relies on intensive arithmetic computations [67]. Several low-power techniques have been used in the implementation of that PVQ decoder. The architecture is divided into four independent processing blocks, each block is separated by a FIFO. Each processing block operates as long as it has data to process and its output FIFO is not full, otherwise, it enters into the standby mode by gating its clock. The critical path of the vector decoder is optimized to improve throughput and hence allow lower voltage operation. The FIFO uses a pointer based scheme for better energy-efficiency.

With microprocessors' speed approaching 300 MHz, for the DEC Alpha 21164, the power dissipation can reach 50 Watts [3] [68]. Various low-power techniques have been employed to lower the power dissipation in microprocessors. In [69], a low-power RISC processor that dissipates less than 2 Watts is presented. The processor uses a 64-bit common bus for floating point as well as integer instruction execution, this reduces the number of functional elements. The number of instruction cache access is reduced by half. The number of I/O transactions is minimized. Data and instruction caches are partitioned into four banks with only one bank active at a time. The dynamic nodes are charged to $V_{DD} - V_T$, rather than to V_{DD}. Through software programming, the system clock can be reduced to 25% of its value.

In [70], low-power techniques were applied to the Alpha 21064 microprocessor [71]. These techniques were able to reduce the power dissipation by over 50 times. Such low-power techniques include; the lowering of the internal power supply to 1.5 Volts. Reduced functionality, the floating point unit and the branch history table were eliminated. Process scaling from $0.75\mu m$ to $0.35\mu m$, this reduced the total switched capacitance. The microprocessor dissipates 450 mW. It has two power down modes. During the idle mode the internal clock is stopped, power dissipation drops to 20 mW. During the sleep mode the internal power supply is switched off, the current drops to $50\mu A$.

In [72], another low-power microprocessor is presented that dissipates less than two watts. This processor features several low-power modes. In the "halt" mode, the processor stops its internal clock. The system can also change the input frequency. During the shutdown state, the system disconnects the processor from the V_{DD}, the register contents are saved in the memory. At power up, the system returns the registers to their previous state.

In [73], another low-power microprocessor is presented that dissipates 3 Watts. The processor has dynamic as well as static power management modes. The dynamic power management disables blocks that are not required to operate during a cycle. Dynamic power management can give up to 30% power saving. There are three static power management modes; Doze, Nap and Sleep. This processor uses an H-Tree clock distribution network over a set of distributed buffers to minimize active power.

3.5 Reducing the Power Dissipation at the Device and Circuit Levels

There are several techniques, during each level of the design process, to reduce the power dissipation. At the device level, the following techniques lead to lower power dissipation:

- **Silicon-On-Insulator (SOI) technology,** this leads to lower leakage currents and lower parasitic capacitances [74] [75].
- **Place and route optimization.** Assign signals with high switching activities to short wires. Also, assign global signals, such as the clock, to layers with low capacitance per unit length.
- **Transistor sizing.** Increasing (W/L) decreases the transistor delay which allows a decrease in voltage to maintain a constant throughput. However, increasing the transistor size increases the capacitance and hence the power dissipation. Hence, an optimum transistor size for minimum power dissipation exists. If the interconnect capacitance is C_p, and the transistor input

capacitance is C_i. It is found that, for small C_p/C_i, the optimum transistor size is the minimum size, otherwise there is an optimum size that gives minimum power dissipation [41].

- **Using submicron devices.** This reduces the parasitic capacitances and allows the use of lower supply voltage, with minimum effect on the delay, for a velocity-saturated device [76].
- **Reducing the subthreshold voltage.** This allows a reduction in the operating voltage, with minimum effect on the delay. But this leads to larger subthreshold currents. Hence, a compromise is required. Some designs use a multi-threshold voltage technology [77]. While others use a variable threshold voltage [65].

At the circuit and logic levels, the following techniques can be used to reduce the power dissipation:

- **Reduce gate capacitance,** for example complementary pass-transistor logic has a lower input capacitance than conventional CMOS logic [78].
- **Reduced logic swing** [41] by making $V_H = V_{DD} - V_T$. However, this has two disadvantages:
  1. Low noise-margin-high (NM_H).
  2. Following gate can dissipate static power.
- **Low-power support circuitry**
  - Level converting circuit [41].
  - High efficiency low-voltage DC/DC converter [79].

- **Logic level power-down.** Modifying the circuits to allow power-down of unused logic blocks. This adds some overhead but can be beneficial if there are certain blocks that are not used for a large portion of the time [41] [80].
- **Multi-threshold circuit technology.** This allows the optimization of low-voltage circuits for high-speed and low-power [63] [81].
- **Scaled multi-buffer stages.** This compromises speed and power for gates driving large capacitive loads [3] [82].

3.6 Low-Voltage Low-Power Operation

The switching power is proportional to the square of the voltage, thus a quadratic reduction in power dissipation is achieved by lowering the supply voltage. However, the delay increases with the reduction of the supply voltage [1]. There are certain techniques used to keep the throughput constant despite the longer delay of the various building blocks. In this section, some of these techniques are investigated.

Figure 3.2.a shows the relative increase in delay as the supply voltage is scaled down. Figure 3.2.b shows the relative decrease in power dissipation as the voltage is scaled down. Both figures were obtained for a CMOS inverter gate loaded by a 1 pF load and using $0.8\mu m$ BiCMOS technology.

From Figure 3.2, it can be notice that at low-voltage the rate of increase of the delay exceeds the rate of decrease of the power dissipation. This usually places a limit on the extent of using voltage scaling techniques.

In this section, two methods which allow the use of lower supply voltage without reducing the throughput are presented. These are:

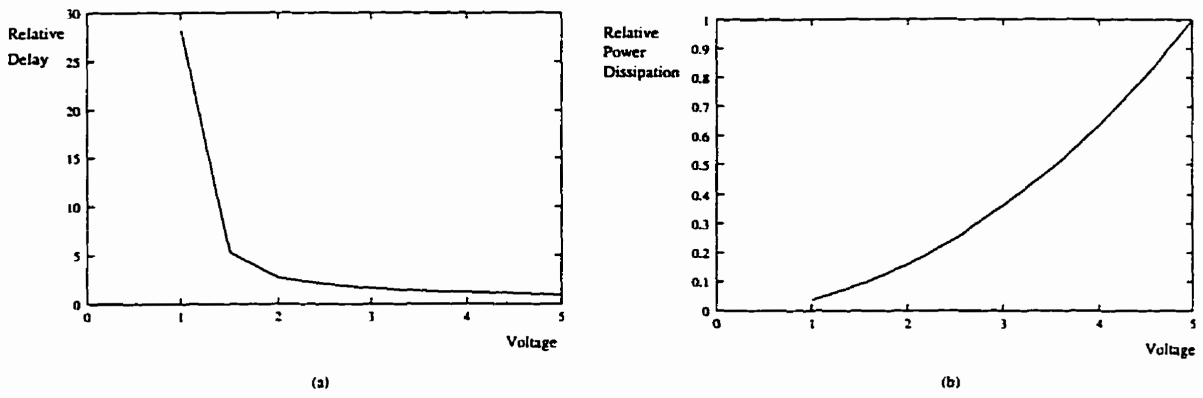


Figure 3.2: The effect of reducing the supply voltage in CMOS circuits, for a $0.8\mu m$ BiCMOS technology:

(a) Relative delay versus supply voltage.
(b) Relative power dissipation versus supply voltage.

1. Pipelining and parallelism.

2. Using carry save adders.

3.6.1 Pipelining and Parallelism at the Architecture Level

The architecture level, is the level in which operators (functional units) act on sets of logic values grouped into words. The manner in which these operators are interconnected or sequenced in time can have an influence on the performance of the architecture in terms of throughput, power dissipation and/or area, without affecting the actual functionality of the architecture.

For example, consider an architecture consisting of two cascaded operators as shown in Figure 3.3.a. Pipelining [83] this architecture, as shown in Figure 3.3.b, gives an alternative architecture with the same functionality as the original architecture, but with different performance. It is possible to operate the pipelined

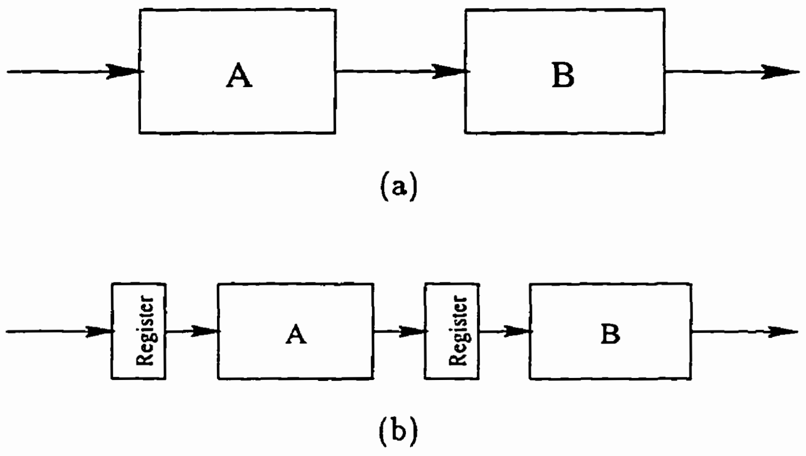


Figure 3.3: Two cascaded operators:

(a) Nonpipelined architecture.
(b) Two-stage pipelined architecture.

architecture at a lower supply voltage and at the same throughput as the original architecture. Thus through pipelining, it is possible to preserve the functionality and the throughput of the system but lower its power dissipation [1], at the cost of latency and extra overhead.

It is also possible, through parallelism, to decrease the power dissipation of the system [1]. In parallelism, the datapath is repeated N times, where N is the degree of parallelism. Like pipelining, the reduction of power dissipation in parallelism is due to the reduction in voltage, while the throughput is kept constant. Figure 3.4 illustrates parallelism.

The merits of pipelining over parallelism are:

1. Smaller area.

2. Reduced logic depth. Hence, less power due to glitches.

On the other hand, the disadvantage of pipelining over parallelism is the unbal-

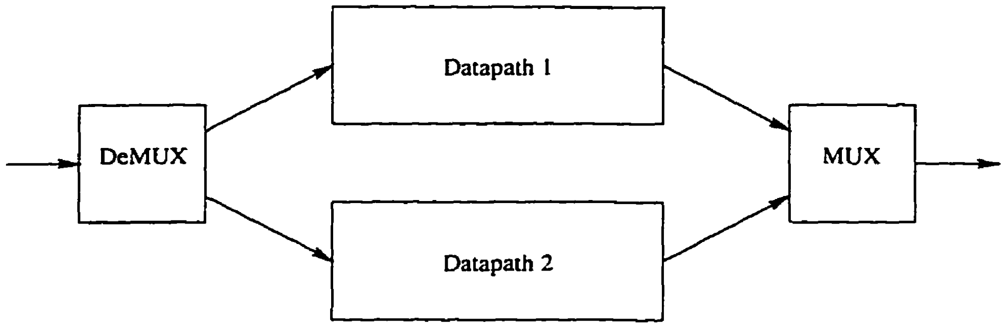


Figure 3.4: A two-datapath parallel system.

anced pipe-stage delay problem [83]. To overcome this, we can combine parallelism and pipelining together as illustrated in the following example.

Example: Combining Pipelining and Parallelism

Consider two cascaded operators A and B as shown in Figure 3.3.a. Assume that:

$$D(A) = 1 \text{ unit}$$

$$D(B) = 2 \text{ units}$$

$$D(AB) = 3 \text{ units}$$

Where D(X) means the delay of operator X.

Pipeline this architecture as shown in Figure 3.3.b. Neglect the register delay with respect to that of the operator. The effective delay of the pipelined system is determined by the delay of the slowest stage which is 2 units in this case. It is possible now to reduce the voltage of the pipelined system to make the throughput of that system equal to the throughput of the non-pipelined one. The power dissipation, from Figure 3.2, is reduced by about 56%.

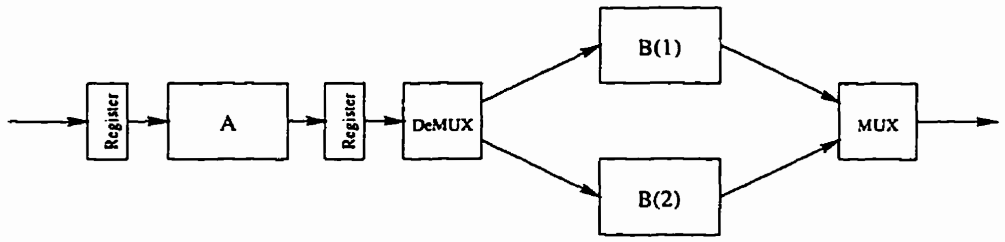


Figure 3.5: Combining parallelism with pipelining to balance pipestage delays.

While the power was reduced by more than half of its value, yet we didn't make full use of pipelining due to the unbalanced pipestage delays. It is possible, by combining parallelism with pipelining, to balance the pipestage delays and hence achieve a larger reduction in power dissipation. Figure 3.5, shows a system that combines parallelism with pipelining. In this case, the effective system delay is 1 unit, allowing an 89% reduction in power dissipation, while maintaining the same throughput as the non-pipelined system.

There is a limit to pipelining and parallelism beyond which no improvement in power dissipation is possible, this is determined by:

1. The extra overhead required for pipelining and parallelism. This is represented by the pipeline registers for pipelining, and by the multiplexers, demultiplexers and the extra wiring capacitance for parallelism.

2. At low-voltage, the rate of increase of delay exceeds the rate of decrease of power dissipation.

The concept of parallelism can also be applied to memory accesses, where several bytes are accessed in parallel instead of accessing them sequentially. Parallelism in memory access is possible only if the data access pattern is sequential in nature [41].

3.6.2 Carry Save Adder

Addition is the most frequent operation performed by a digital signal processor, whether explicitly or within other operations such as multiplication. Hence, if we are able to use faster adders, the delay of the critical path can be reduced which allows the use of lower supply voltage.

The delay of the adder is mainly due to the propagation of the carry from the least significant bit position to the most significant bit position. Consider an $N-$ bit adder, the delay of the ripple-carry adder [84], which is the most power-efficient adder [85] compared to other adders at the same voltage, is proportional to N. Adders, such as the carry-lookahead adder and conditional sum adder, have a delay proportional to $\log(N)$ [84], hence it is possible to lower the supply voltage of these adders and get a delay equal to that of the ripple-carry adder. The lowering of the supply voltage in these adders leads to a lower power dissipation than that of the ripple-carry adder [86].

However, it will be much better if the carry propagation can be eliminated all together. This will be specially useful when we have a cascade of adders, which is usually the case in a finite impulse response (FIR) filter. In this case, by using a carry save adder [84] for all adders and using a ripple-carry adder (or any other fast adder) for adding the sum and carry words of the last carry save adder, the delay can be significantly reduced, hence a lower supply voltage can be used and lower power dissipation is achieved.

The choice of the adder depends on the following parameters;

1. The number of adders in the adder chain M.

2. The number of bits per word N.

3. The ratio between the sum delay T_s and the carry delay T_c.

4. The ratio between the adder power P_A and the load power P_L.

5. The relation between delay and power versus voltage (Figure 3.2).

The power-optimum adder is not necessary the fastest adder. In some cases [87], the carry save adder architecture is faster than the ripple carry adder architecture, but the latter is more power efficient even after voltage scaling to maintain a constant throughput. This is due to the extra hardware required for the carry save adder architecture [87]. In general, optimization for high throughput is different from optimization for low-power [51].

3.7 Pipelining and Parallelism of the Discrete Cosine Transform

Discrete cosine transform (DCT) is frequently used in video compression [9] [88]. In this section, three different architectures for DCT are considered. the effect of pipelining and parallelism on reducing the power dissipation of each architecture is also considered.

The mathematical formula for the one dimensional DCT (1D-DCT) is given by:

$$Y_k = \sum_{i=0}^{N-1} X_i C_k \cos \frac{(2i+1)k\pi}{2L} \tag{3.4}$$

for $k = 0, 1, \ldots, N-1$

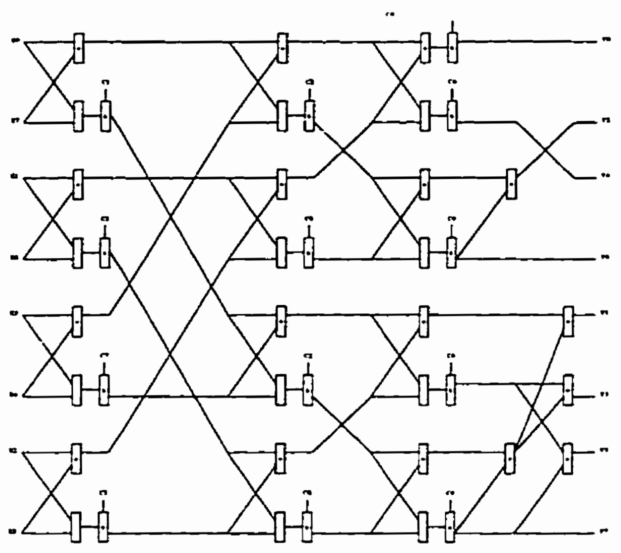


Figure 3.6: Multiplier architecture for an 8-point 1D-DCT.

where,

$$C_k = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } k = 0 \\ 1 & \text{otherwise} \end{cases}$$

Direct implementation of this algorithm requires N^2 multiplications and $N(N-1)$ additions, this not only increases the power dissipation, but the area and/or delay as well.

3.7.1 Three Alternative Architectures

The Multiplier Architecture

While the direct implementation of an 8-point 1D-DCT requires 64 multiplications and 56 additions, various algorithms have been proposed that require a fewer number of additions and multiplications. As an example, the algorithm given in [89] and shown in Figure 3.6, requires only 29 add/sub blocks and 13 multipliers.

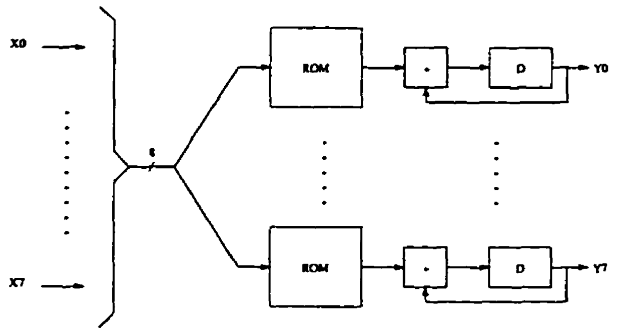


Figure 3.7: Pure ROM architecture for an 8-point 1D-DCT.

The Pure ROM Architecture

The idea of this implementation is to replace the multiplication operation with addition and a look-up ROM table. This process is known as distributed arithmetic [9] [90].

We can use distributed arithmetic to implement the 8-point 1D-DCT. The block diagram for this architecture is shown in Figure 3.7. Eight 256-word ROMs are required for this architecture.

The Mixed ROM Architecture

The 8-point 1D-DCT can be expressed as the product of an 8×8 matrix by an eight element column vector. However, through algebraic manipulation [9] [89], this matrix can be broken down into two 4×4 matrices, as given by the following equations:

$$\begin{bmatrix} Y_0 \\ Y_2 \\ Y_4 \\ Y_6 \end{bmatrix} = \begin{bmatrix} c_4 & c_4 & c_4 & c_4 \\ c_2 & c_6 & -c_6 & c_2 \\ c_4 & -c_4 & -c_4 & c_4 \\ c_6 & -c_2 & c_2 & -c_6 \end{bmatrix} \begin{bmatrix} X_0 + X_7 \\ X_1 + X_6 \\ X_2 + X_5 \\ X_3 + X_4 \end{bmatrix} \tag{3.5}$$

$$\begin{bmatrix} Y_1 \\ Y_3 \\ Y_5 \\ Y_7 \end{bmatrix} = \begin{bmatrix} c_1 & c_3 & c_5 & c_7 \\ c_3 & c_7 & -c_1 & c_5 \\ c_5 & -c_1 & -c_7 & c_3 \\ c_7 & -c_5 & c_3 & -c_1 \end{bmatrix} \begin{bmatrix} X_0 - X_7 \\ X_1 - X_6 \\ X_2 - X_5 \\ X_3 - X_4 \end{bmatrix} \tag{3.6}$$

where,

$$c_i = \frac{1}{2\cos(\frac{i\pi}{16})}$$

In this case, the number of words per ROM is only 16 words (16 times less than the pure ROM architecture), but some overhead has been incurred in the adders required to calculate the address of the ROMs. Figure 3.8 shows a block diagram for this architecture.

3.7.2 Reducing Power Through Pipelining and Parallelism

Pipelining

As the voltage decreases, to decrease the power dissipation, the overall delay of the datapath increases (an undesirable side effect). To counteract this increase in delay, we can use pipelining [1]. The rational of using pipelining here is that the increase in delay accompanying the decrease in voltage is balanced by dividing the datapath into smaller pipestages and keeping the maximum delay of any pipestage,

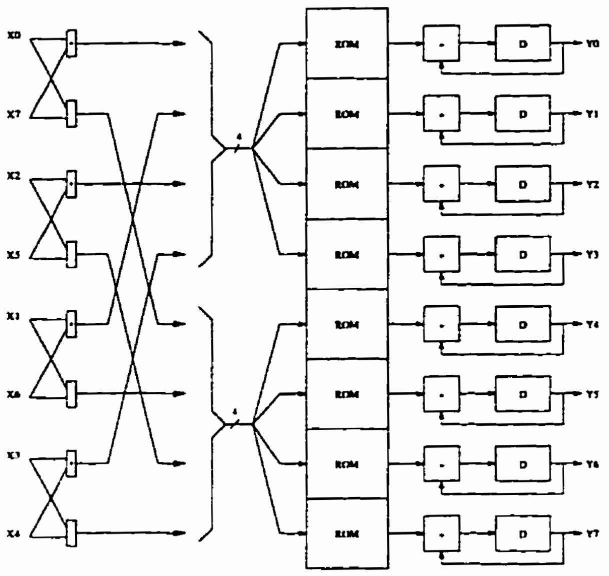


Figure 3.8: Mixed ROM architecture for an 8-point 1D-DCT.

at the lower voltage, equal to the overall delay of the datapath without pipelining, at the higher voltage.

As a first order approximation, make the following assumptions:

- Neglect the effect of overhead caused by the pipeline registers, whether in terms of increased capacitance or increased delay.
- Assume that the pipeline can be perfectly balanced. All pipestages have the same delay.
- The delay of any stage is inversely proportional to the applied voltage.

To maintain the same maximum throughput when dividing the datapath into N pipestages, each pipestage operates at a voltage V/N. The power dissipation in this case is approximately given by:

$$P_{pl} = \frac{P_s}{N^2} \tag{3.7}$$

Where, P_s is the power dissipation before pipelining, and P_{pl} is the power dissipation after pipelining and reducing the voltage.

Parallelism

Parallelism can also be used to keep the throughput of the system constant as the voltage decreases [1]. To compensate the increase in datapath delay as the voltage decreases, we replicate the datapath N times. The input samples are split among the N datapaths. The outputs of the N datapaths are then multiplexed onto a single output stream. This allows the system to maintain its throughput, while each datapath is operating at a lower rate and a lower voltage.

Unlike pipelining, parallelism greatly increases the area. Making approximations similar to those made in the analysis of pipelining, we can show that the power dissipation of a system consisting of N parallel datapaths and having the same throughput rate is approximately given by:

$$P_{pr} = \frac{P_s}{N^2} \tag{3.8}$$

Where, P_s is the power dissipation for a single datapath system, and P_{pr} is the power dissipation for a system consisting of N parallel datapaths and having the same maximum throughput as the single datapath system.

3.7.3 Performance Evaluation

The Multiplier Architecture

Table 3.1 gives the Spice simulation results of the overall delay and power dissipation of the multiplier architecture of Figure 3.6 in a $0.8\mu m$ BiCMOS technology, the

Table 3.1: Delay and power dissipation at 5 MHz for the multiplier implementation with no pipelining.

Voltage	Delay	Power
5	101.56 nS	$13.49mW$
4	132.27 nS	$7.64mW$
3.3	162.53 nS	$4.99mW$

power dissipation is evaluated at a frequency of 5 MHz. Notice that the power dissipation is reduced as the voltage decreases, but this is at the expense of the increased delay (decreased throughput). To maintain the same throughput rate at the lower voltage we use pipelining or parallelism.

Figure 3.9 shows the approximate relation between the power dissipation and the number of pipestages, given by Equation 3.7. Also shown are the simulation results obtained from pipelining the multiplier architecture. Everything has been normalized to the case of a single stage system operating at the same throughput.

It is also possible to maintain the same throughput while reducing the voltage by applying parallelism. Figure 3.10 shows the approximate relation between the power dissipation and the number of datapaths, given by Equation 3.8. Also shown are the simulation results obtained from increasing the degree of parallelism of the multiplier architecture. Everything has been normalized to the case of a single datapath system operating at the same throughput.

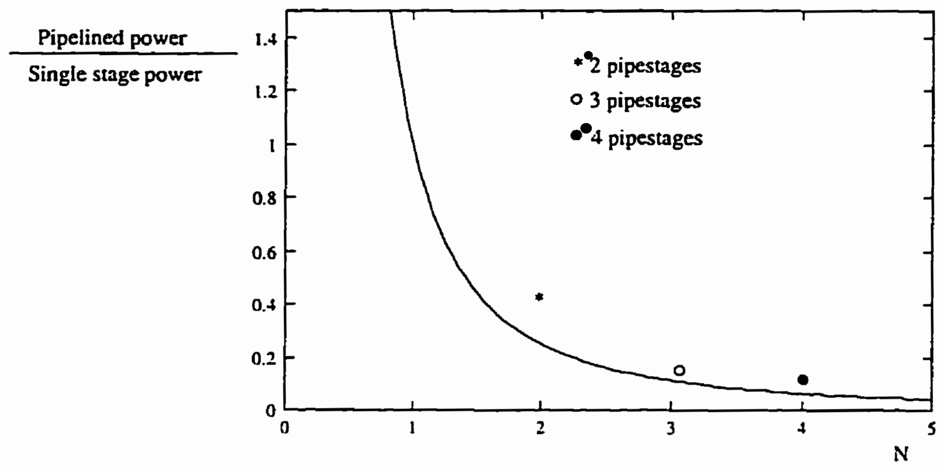


Figure 3.9: The effect of pipelining on reducing the power dissipation, while maintaining a constant throughput.

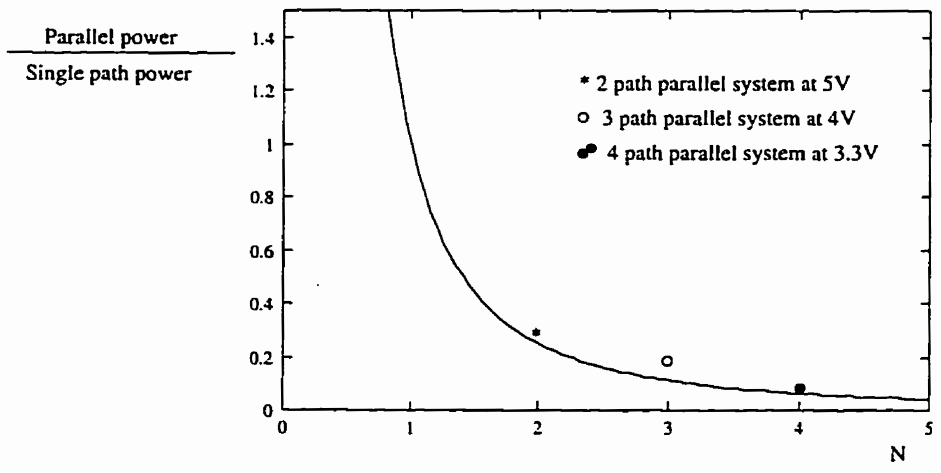


Figure 3.10: The effect of parallelism on reducing the power dissipation, while maintaining a constant throughput.

Table 3.2: Total delay and power dissipation at 5 MHz for the pure ROM architecture.

Voltage	Delay	Power
5	86.5 ns	30.4 mW
4	111.4 ns	18.3 mW
3.3	137.1 ns	10.5 mW

The ROM Architectures

Two different types of ROM architectures were considered. First, the pure ROM architecture, which requires eight 256-word ROMs is considered. The Spice simulation results of the overall delay and power dissipation for this architecture are given in Table 3.2. All eight ROMs have the same address, hence a single ROM address decoder was used.

The second type of ROM architecture considered is the mixed ROM architecture. This architecture requires eight 16-word ROMs. But it requires eight extra adder/subtractor units. The simulation results of the overall delay and power dissipation for this architecture are given in Table 3.3. The eight ROMs can be divided into two groups, the ROMs of each group have the same address. Hence, two ROM address decoders were used.

Three architectural alternatives, in addition to the single stage alternative, were considered for each of the pure and mixed ROM implementations:

- Two stage pipeline.
- Two stage pipeline with two parallel adders per path, Figure 3.11.

Table 3.3: Total delay and power dissipation at 5 MHz for the mixed ROM architecture.

Voltage	Delay	Power
5	79.5 ns	30.1 mW
4	103.7 ns	18.1 mW
3.3	129.1 ns	11.0 mW

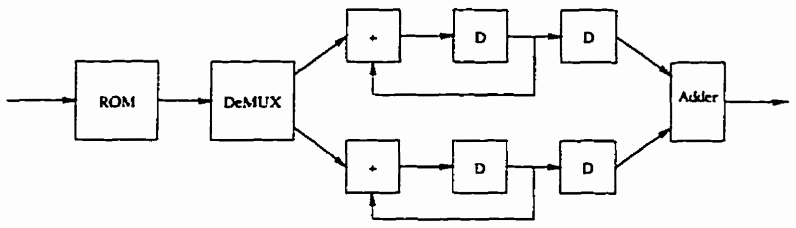


Figure 3.11: Using two parallel adders in the second pipestage.

- Two stage pipeline with three parallel adders per path, Figure 3.12.

Tables 3.4 and 3.5 show the effect of pipelining and parallelism on reducing the power dissipation in the pure ROM and mixed ROM implementations. The advantage of using parallelism with pipelining is to balance the pipestage delays.

In terms of power dissipation, the multiplier architecture has the lowest power dissipation, while the pure ROM and the mixed ROM architectures have higher power dissipations. In terms of speed, the multiplier architecture is the slowest and the mixed ROM architecture is the fastest.

Pipelining the ROM architectures provides only a modest reduction in power dissipation, or a modest increase in throughput if the voltage is kept constant. This is because of the unbalanced pipestage delays. The delay of the second pipestage is 2–2.5 times the delay of first pipestage. Parallelism in the second pipestage is used

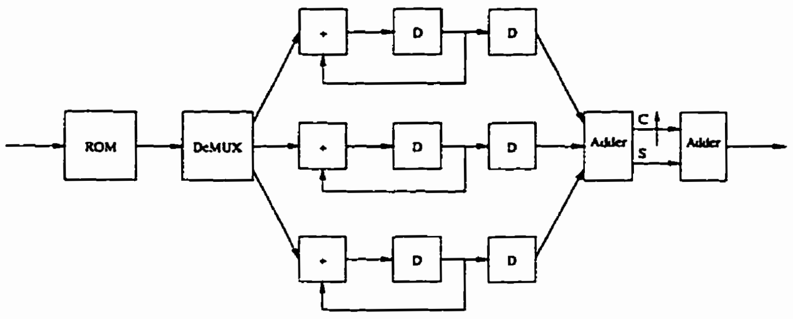


Figure 3.12: Using three parallel adders in the second pipestage.

Table 3.4: Reducing the power dissipation by pipelining and parallelism in the pure ROM implementation.

Delay(ns)	Architecture	Power (mW)
85 - 90	Single stage	30.4 (@ 5V)
	2 stage 1 add/path	21.9 (@ 4V)
65 - 70	2 stage 1 add/path	36.1 (@ 5V)
	2 stage 2 add/path	14.5 (@ 3.3V)
52 - 57	2 stage 2 add/path	24.6 (@ 4V)
	2 stage 3 add/path	14.5 (@ 3.3V)
42 - 46	2 stage 2 add/path	40.6 (@ 5V)
	2 stage 3 add/path	24.9 (@ 4V)

Table 3.5: Reducing the power dissipation by pipelining and parallelism in the mixed ROM implementation.

Delay(ns)	Architecture	Power (mW)
65 - 70	2 stage 1 add/path	35.7 (@ 5V)
	2 stage 2 add/path	15.0 (@ 3.3V)
50 - 55	2 stage 2 add/path	24.4 (@ 4V)
	2 stage 3 add/path	16.0 (@ 3.3V)
41 - 43	2 stage 2 add/path	40.2 (@ 5V)
	2 stage 3 add/path	25.8 (@ 4V)

to alleviate this problem as shown in Figure 3.11 and Figure 3.12.

Using a two stage pipeline, and three times parallelism in the second stage of the pure ROM architecture, reduces the power dissipation by 52% and increases the throughput by 38%, when lowering the voltage from 5 Volts to 3.3 Volts. For the mixed ROM architecture, a two stage pipeline, with three times parallelism in the second stage, achieves a 47% saving in the power dissipation, and increases the throughput by 36%.

3.8 Effect of the Number System on the Switching Activity

In this section, I demonstrate how the choice of the number system can reduce the switching activity. Two number systems are considered, the two's complement number system and the Gray code number system. The Gray code number system has the advantage that any two adjacent numbers differ in one bit only. So that, by coding correlated samples in Gray code we can effectively reduce the switching activity [91].

Assume positive samples, hence instead of considering two's complement representation, unsigned binary representation is considered. Each sample is represented by an N bit binary word, each binary word is assigned an integer n. The range of n is:

$$n = 0 \ldots 2^N - 1$$

The binary word is the unsigned binary representation of n, or the Gray code of n depending on which representation is used. Let i be the integer representing the current sample and j be the integer representing the previous sample. Assume that the conditional probability distribution is given by:

$$P(i/j) = \begin{cases} \frac{M+1-|i-j|}{(M+1)^2} & |i-j| \leq M \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

M is a factor which describes the correlation between the successive samples. The larger the value of M, the less correlated the samples are. Figure 3.13 shows the conditional probability distribution of x_n given x_{n-1}, for different values of M.

Using the probability distribution given in Equation 3.9, we can derive the switching activity for different values of M. This is given in Table 3.6. α_g is the switching activity for the Gray code representation, while α_u is the switching activity for the unsigned binary representation.

Notice that as the correlation between the successive samples is reduced, the effectiveness of the Gray code in reducing the switching activity becomes less. Another factor in determining the power-optimum number system is the complexity

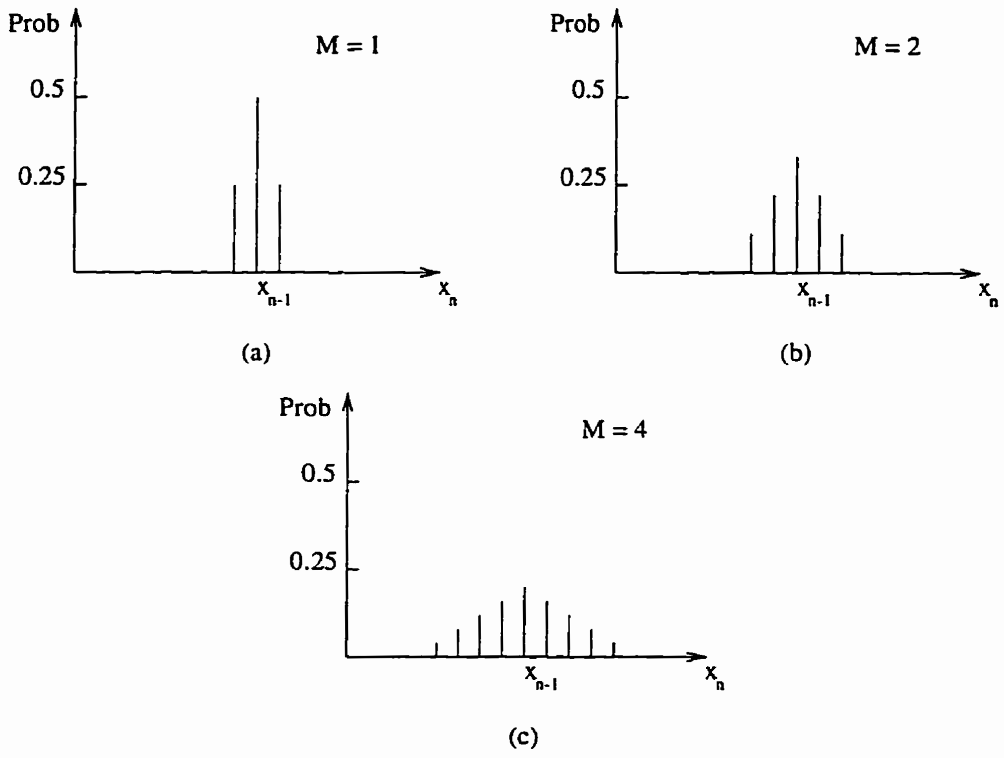


Figure 3.13: Conditional probability distribution between successive samples for different values of M. x_{n-1} is the value of the previous sample. The x-axis represents the value of the current sample.

Table 3.6: Switching activity of the Gray code and the unsigned binary representations for correlated samples.

M	α_g	α_u	$\frac{\alpha_u}{\alpha_g}$ $(N >> 1)$
1	$\alpha_g = \frac{1}{2N}$	$\alpha_u = (1 - \frac{1}{2^N})\frac{1}{N}$	2
2	$\alpha_{GC} = \frac{8}{9N}$	$\alpha_u = (\frac{12}{9} - \frac{16}{9}\frac{1}{2^N})\frac{1}{N}$	1.5
3	$\alpha_g = \frac{9}{8N}$	$\alpha_u = (\frac{13}{8} - \frac{5}{2}\frac{1}{2^N})\frac{1}{N}$	1.44
4	$\alpha_g = \frac{32}{25N}$	$\alpha_u = (\frac{44}{25} - \frac{74}{25}\frac{1}{2^N})\frac{1}{N}$	1.38
5	$\alpha_g = \frac{17}{12N}$	$\alpha_u = (\frac{23}{12} - \frac{35}{9}\frac{1}{2^N})\frac{1}{N}$	1.35
6	$\alpha_g = \frac{76}{49N}$	$\alpha_u = (\frac{100}{49} - \frac{212}{49}\frac{1}{2^N})\frac{1}{N}$	1.32

of the operator and hence the energy dissipated per single execution of the operator. It is quite possible that the choice of a number system to lower the switching activity will lead to higher operator energy. Hence, a compromise is required in choosing the power-optimum number system.

As an example, consider the addition operator. The adder operates repeatedly on correlated successive samples. Two number systems are considered:

1. The unsigned binary number system.

2. The Gray code number system.

Four factors are considered in determining the optimum number system:

1. The correlation factor ρ.

2. The number of bits per word N.

3. The operator energy ratio.

4. The relative load capacitance.

The effect of the correlation factor and the number of bits per word on the switching activity is given by Table 3.6 and shown in Figure 3.14. The operator energy ratio depends on the details of the circuits and the implementing technology for each number system adder. Hence, this ratio can vary from one implementation to the other. Yet, a first order estimation is required. This was done by writing a VHDL description for each adder, and then synthesizing this design and estimating its power dissipation using the COMPASS tools. For a single operator execution. the energy dissipated in an unloaded Gray code adder is about double that dissipated in an unsigned binary representation adder.

Let α_u be the switching activity of the unsigned binary number representation. Let α_g be the switching activity of the Gray code number representation. Let P_u be the normalized power dissipated by the unsigned binary adder. Let P_g be the normalized power dissipated by the Gray code adder, and let P_l be the normalized power dissipated by the adder load. The total power dissipated by each adder and its load is given by:

$$P_U = \alpha_u(P_u + P_l) \tag{3.10}$$

$$P_G = \alpha_g(P_g + P_l) \tag{3.11}$$

for the unsigned binary adder and the Gray code adder respectively.

The objective is to find the boundary at which the two adders dissipate the same amount of power. On one side of this boundary, the unsigned binary representation has lower power dissipation. While on the other side, the Gray code representation has lower energy dissipation. The Equation of this boundary is given by:

$$R_G = S_G + (S_G - 1)R_L \tag{3.12}$$

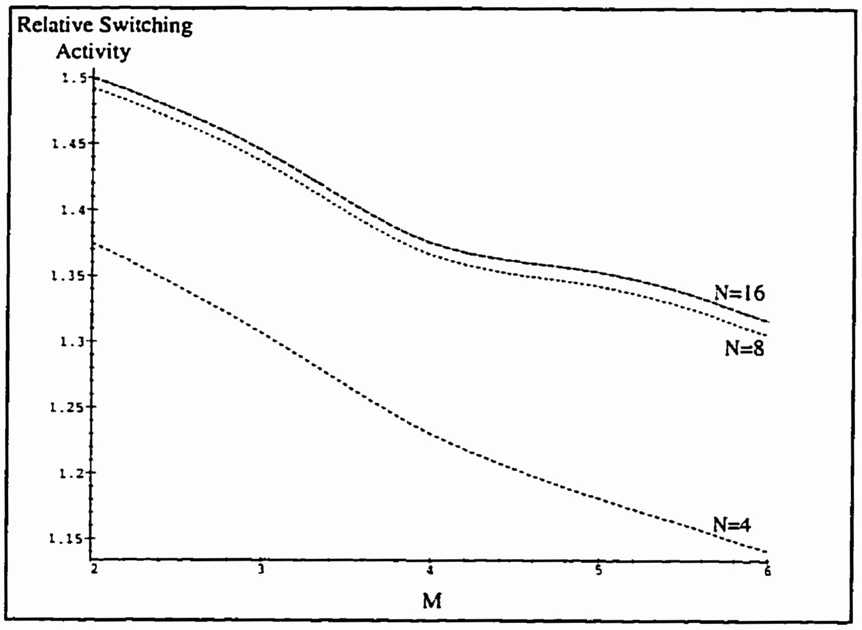


Figure 3.14: The ratio between the switching activity of the unsigned binary representation and the Gray code representation versus the correlation factor M, for different word length N.

where,

$R_G = \frac{P_g}{P_u}$,

$R_L = \frac{P_l}{P_u}$, and

$S_G = \frac{\alpha_u}{\alpha_g}$.

Figure 3.15 shows the relation between R_G and R_L for different values of N and M. N is the number of bits per word, while M is a factor related to the correlation between the successive samples. For a certain R_L, a value of R_G higher than that given by the curves means that the unsigned binary representation is more power-efficient than the Gray code representation and vice versa.

3.9 Reducing the Number of Iterations

In this section, the effect of reducing the number of block iterations per output on the power dissipation is examined. Two examples are given. In the first example, the number of block iterations per output for a division algorithm is reduced. This is done by using higher-order radix. The effect of this on power dissipation is illustrated.

In the second example, a division algorithm is developed which reduces the number of add/sub operations required per output [92]. The effect of this on reducing the power dissipation is illustrated.

3.9.1 Higher Radix Division Algorithms

In the division process [93] the dividend X is divided by the divisor D to generate the quotient Q. The division operation is expressed as:

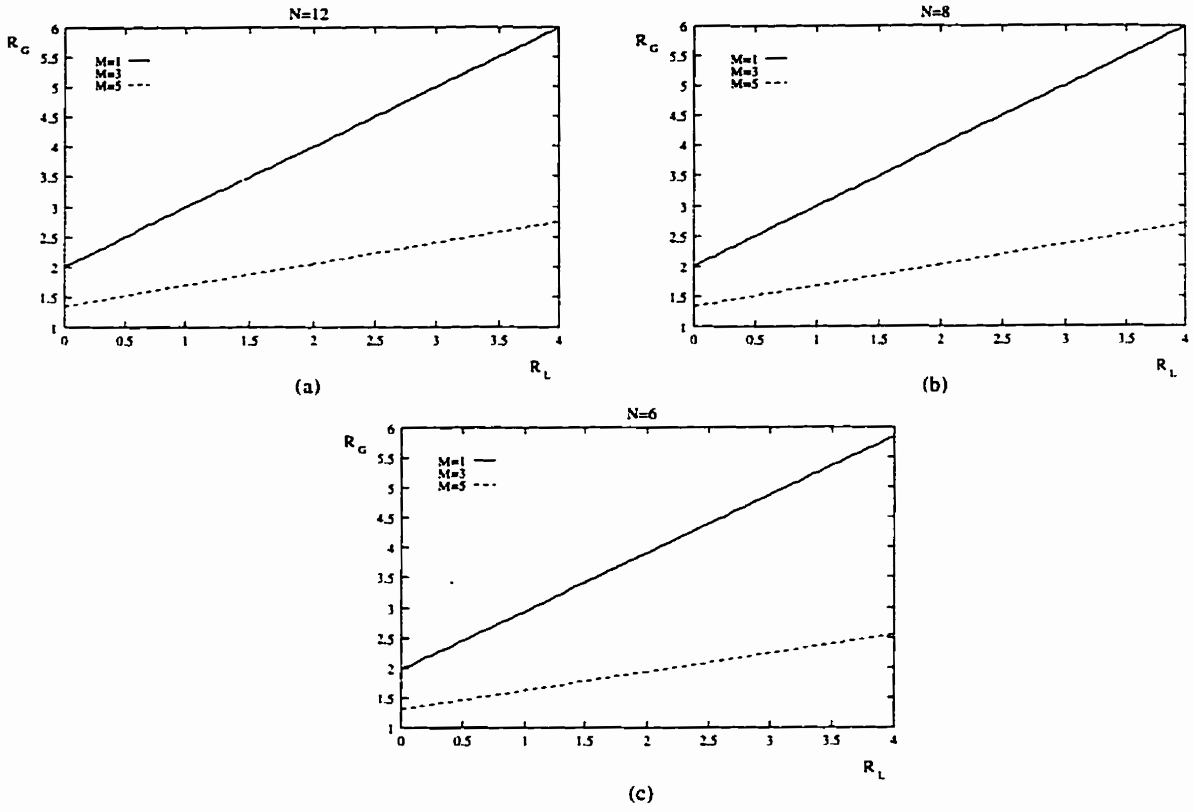


Figure 3.15: The relation between the relative Gray code adder power and the relative load power, for equal power dissipation in the Gray code and unsigned binary adders. N is the number of bits per word. M is a factor related to the correlation between successive samples. The larger the value of M, the less the correlation between successive samples.

$$\frac{X}{D} = Q = q_1 r^{-1} + q_2 r^{-2} + \ldots + q_n r^{-n} \tag{3.13}$$

In Equation 3.13, r is the radix order. For the purpose of this discussion, r is a power of 2, $r = 2^m$, where m is a positive integer. $q_i \in \{0, 1, \ldots, r-1\}$. The purpose of the division algorithm is to find the quotient digits $q_1, q_2, \ldots q_n$. In the digit-recurrence division algorithm, the quotient digits $q_1, q_2, \ldots, q_n$ are obtained sequentially starting with q_1.

In the process of obtaining the quotient in the two's complement representation, an intermediate quotient in a redundant signed-digit representation is first obtained. For the intermediate quotient, $q_i \in \{-a, \ldots, -1, 0, 1, \ldots, a\}$. Where, $a < r$. a is an integer.

During iteration $j+1$, the quotient digit q_{j+1} is generated by the Quotient Digit Selection (QDS) unit:

$$q_{j+1} = \mathrm{QDS}(P(j), D) \tag{3.14}$$

A new partial remainder is generated according to the following equation,

$$P(j+1) = rP(j) - Dq_{j+1}, \tag{3.15}$$

with

$$P(0) = X \tag{3.16}$$

Where

$P(j)$ is the partial remainder after j iterations.

$j = 0 \ldots (n-1)$.

r is the radix of the number system.

D is the divisor.

Figure 3.16 shows the block diagram of the digit-recurrence division algorithm. Several points need to be clarified about this block diagram. The adder used is a carry save adder which generates two outputs the sum and the carry of the partial remainder. The QDS requires the partial remainder in two's complement format. However, when using a redundant quotient-digit set the accuracy of the partial remainder required by the QDS is limited. Hence, the carry propagate adder (CPA) is of limited accuracy, adding only a few of the most significant bits of the sum and carry of the partial remainder. The quotient is generated in signed-digit format [94] (a redundant quotient-digit set), hence a module is required to convert it to the two's complement format. This is done by On-The-Fly (OTF) module [95].·

For the same accuracy, the number of iterations n depends on the radix used. Assume that n is 12 for a radix 2 algorithm, then n is 6 for a radix 4 algorithm, 4 for a radix 8 algorithm, 3 for a radix 16 algorithm and so on. However, reducing the block activity factor, i.e. the number of iterations per second, doesn't necessary lead to lower power dissipation. Higher radix blocks are more complex and hence, dissipate more power.

For low-power design, the objective is to minimize the power dissipation after meeting the throughput requirement. There are two conflicting factors that need to be taken into consideration:

- The block activity factor. This decreases as the radix order increases.
- The normalized energy per block. This increases as the radix order increases.

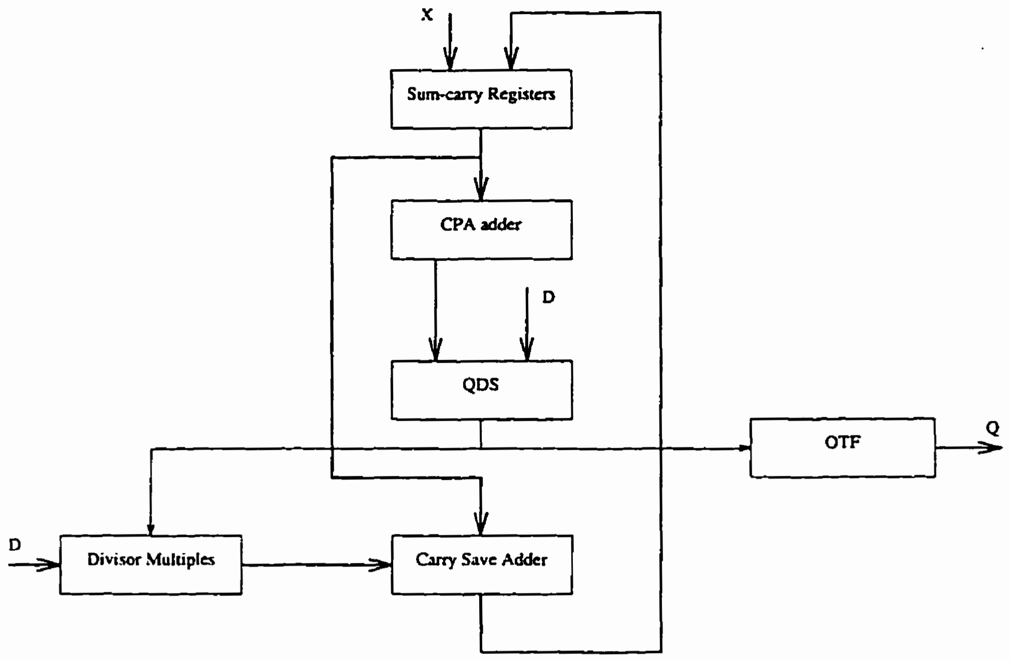


Figure 3.16: Block diagram of a digit-recurrence division algorithm.

In addition, the effect of voltage scaling, if permissible by the technology, has to be taken into account.

Table 3.7 shows the parameters of each radix-dependent block. For the QDS, the numbers shown determine the number of integer bits and the required fractional accuracy for the shifted partial remainder rP and the divisor D. The actual number of bits the QDS requires from D is one less than the accuracy of D, because D is always in the range $[0.5, 1)$. The range of the signed-digit quotient digits is $[a, -a]$. a is chosen to minimize the computation done in the Divisor Multiples module.

Using the data of Table 3.7, the relative speed and relative power dissipation of each block in Figure 3.16 can be estimated. The word *relative* means relative to radix 2. Tables 3.8 and 3.9 show this data. For the CPA, the power dissipation is proportional to the number of bits added. The delay is also proportional to

Table 3.7: Parameters of the radix-dependent blocks of Figure 3.16.

Radix	a	CPA	QDS	
			rP	D
2	1	4	$3+1$	1 (-1)
4	2	7	$3+4$	4 (-1)
8	6	8	$4+3$	5 (-1)
16	10	11	$5+6$	6 (-1)

the number of added bits. The QDS is a combinational logic book, the power dissipation is estimated according to [96]:

$$P \propto \frac{2^{n+1}}{3n(n+m)} H_o(H_i + 2H_o) \tag{3.17}$$

where,

n and m are the number of inputs and outputs respectively.

H_i and H_o are the entropies of the input and output respectively.

Assuming that all inputs and outputs are equally probable

$$P \propto \frac{2^{n+1}}{3n(n+m)} m(n+2m) \tag{3.18}$$

The delay of the QDS is proportional to the radix. For the Divisor Multiples (DM), this module generates one or two outputs (in case of radix 8 and 16) from the divisor D through shifting. In the case of radix 8 and 16, these two outputs should be added, hence a two-level CSA is required.

The OTF consists of two parts. Two registers N bits each, and a combinational logic block. The relative power dissipation between these two blocks has to be

Table 3.8: Relative power dissipation for the different blocks of Figure 3.16.

Radix	SCR	CPA	QDS	CSA	DM	OTF
2	1	1	1	1	1	$1+1$
4	1	1.75	35	1	2	$1+2$
8	1	2	88	2	6	$1+3$
16	1	2.75	2400	2	8	$1+4$

Table 3.9: Relative delay for the different blocks of Figure 3.16.

Radix	SCR	CPA	QDS	CSA	DM	OTF
2	1	1	1	1	1	$1+1$
4	1	1.75	2	1	1	$1+2$
8	1	2	3	2	1	$1+3$
16	1	2.75	4	2	1	$1+4$

determined, so when comparing the relative power dissipation and delay for the different radices the two blocks are compared independently.

To evaluate the performance of each radix implementation, the relative power dissipation and the relative delay of each building block for the Radix 2 division algorithm has to be known. This was done through computer simulation. The results obtained are shown in Table 3.10.

Using the data given in Tables 3.8, 3.9 and 3.10, the power dissipation and the throughput of any radix division algorithm relative to the radix 2 division algorithm can be calculated. If it is also possible to scale the voltage to get equal throughput, the power dissipation of any radix division algorithm relative to the radix 2 division

Table 3.10: Relative power dissipation and delay for the different blocks of Figure 3.16 for a radix 2 division algorithm.

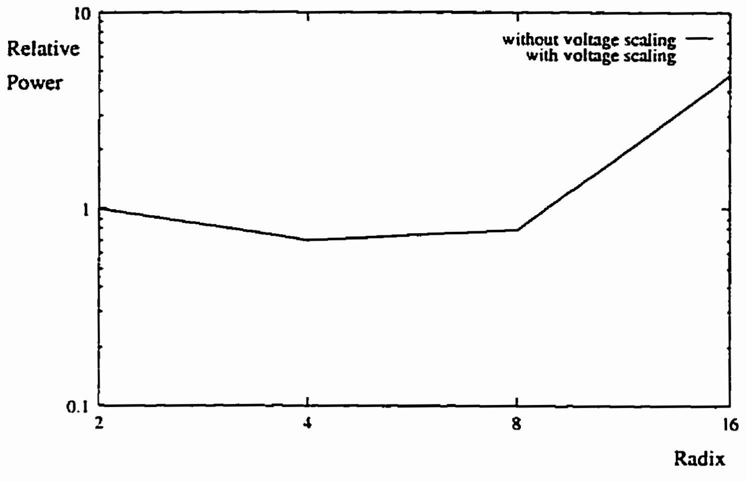
	SCR	CPA	QDS	CSA	DM	OTF
Power	7.2	1	0.17	4.9	2.2	$7.9 + 0.3$
Delay	1.7	1	1.2	0.85	1.3	$1.07 + 1.03$

algorithm can be calculated with help of Figure 3.2. Figure 3.17 shows the relative throughput and power dissipation with and without voltage scaling.

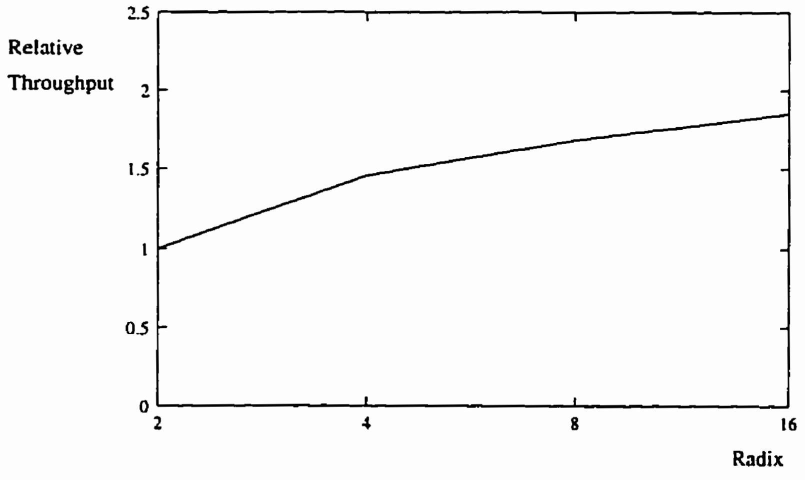
Notice from Figure 3.17.a that as the radix order increases the power initially decreases. This is because the reduction in the block activity is greater than the increase in the normalized block power. As the radix order increases more, the increase in the normalized block power exceeds the decrease in the block activity factor. Hence, the over all power dissipation increases.

Another factor, that can be taken into consideration, is the increase in throughput as the radix order increases. This allows a reduction in voltage to equalize the throughputs, reducing the power dissipation of the higher radix systems. Even after taking into account the effect of voltage scaling, radix 16 has higher power dissipation than radix 2. However, with voltage scaling the minimum-power radix has shifted from 4 to 8.

The reason for the high power dissipation of radix 16 is the complexity of the QDS module. This is because a is limited to 10, so as to limit the number of output words from the Divisor Multiples unit to 2, which only requires one extra CSA level. Allowing a to go up to 14 greatly reduces the complexity and power dissipation of the QDS, but increases the power dissipation in the Divisors Multiples unit and in the CSA. The choice of one alternative over the other depends on the relative



(a)



(b)

Figure 3.17: The effect of the radix on the power dissipation and throughput of the division algorithm.

(a) Relative power dissipation with and without voltage scaling.

(b) Relative throughput.

power dissipation between the various modules.

3.9.2 Minimizing Add/Sub Operations in Division

In the multiplication process, the minimum number of add/sub operations occurs when the multiplier is in the minimal signed-digit (SD) representation (having the minimum number of non-zero digits). One can thus expect the minimum number of add/sub operations required in the division operation to occur when the resultant quotient has the minimal SD representation. In the following discussion, we concentrate on how to generate this minimal SD quotient.

Consider the division operation:

$$Q = \frac{X}{D} \tag{3.19}$$

where $X < D$. Based on the values of X, we can proceed to calculate the partial remainder as follows [97] [98]:

1. If $0 \leq X < \frac{1}{2}D$, then
$$Q = 0 + y \tag{3.20}$$
where, $0 \leq y < \frac{1}{2}$.
To get the shifted partial remainder, we only have to multiply X by 2.

2. If $\frac{1}{2}D \leq X < \frac{3}{4}D$, then
$$Q = \frac{1}{2} + y \tag{3.21}$$
where, $0 \leq y < \frac{1}{4}$.
1/2 requires only one digit for its representation. To get the shifted partial remainder, we subtract $D/2$ from X and multiply the remainder by four.

3. If $\frac{3}{4}D \leq X < D$, then

$$Q = \frac{3}{4} + y \tag{3.22}$$

where, $0 \leq y < \frac{1}{4}$. Alternatively,

$$Q = 1 - z \tag{3.23}$$

where, $0 \leq z < \frac{1}{4}$.

The second representation is preferred over the first one, because 1 can be represented by one digit, while $\frac{3}{4}$ needs two digits (i.e. $0.75_{10} = 0.11_2$). In this case, to get the shifted partial remainder, we subtract D from X and multiply the remainder by four.

It is clear from the above discussion that X is first compared with $\frac{3}{4}D$ and $\frac{1}{2}D$. Depending on this comparison result one of the following two actions is taken:

- **Either** a subtraction operation is performed and the resulting remainder is multiplied by 4 (shift to the left by 2) to get the new partial remainder.
- **Or** no subtraction is required and the partial remainder is multiplied by 2 (shift to the left by 1) to get the new partial remainder.

Now a division algorithm can be formulated as follows:

If $|r_i| < D/2$

$$q_i = 0$$

$$r_{i+1} = 2r_i$$

else if $\frac{1}{2}D \leq |r_i| < \frac{3}{4}D$

$$q_i = 0 \quad q_{i+1} = \begin{cases} 1 & r_i > 0 \\ \bar{1} & r_i < 0 \end{cases}$$

$$r_{i+2} = 4(r_i - q_{i+1}\frac{D}{2})$$

else if $\frac{3}{4}D \leq |r_i|$

$$q_i = \begin{cases} 1 & r_i > 0 \\ \bar{1} & r_i < 0 \end{cases} \qquad q_{i+1} = 0$$

$$r_{i+2} = 4(r_i - q_i D). \tag{3.24}$$

As it can be seen in the above algorithm, each iteration is divided into two steps, each of which can be performed in a clock cycle, when implemented in VLSI. In the first step, the comparison is performed. The addition (subtraction), if required, is performed in the second step. As a result, a one step iteration produces one quotient digit, while a two step iteration produces two quotient digits. This technique has two advantages. First, it allows the use of a shorter clock cycle. Second, the division period is independent of the number of non-zero quotient digits. Figure 3.18 shows a block diagram of the proposed division algorithm.

Now we want to show that the obtained SD quotient contains the minimum number of nonzero digits. For a minimal SD quotient in the canonical form, any two adjacent digits should contain at least one zero digit [84]. However, for the quotient representation obtained by the algorithm presented here, it is possible to get two adjacent 1's or two adjacent $\bar{1}$'s. Consider the case of two adjacent 1's. This is obtained when we have a partial remainder with a value in the range $[\frac{1}{2}D, \frac{3}{4}D)$, and the following partial remainder is in the range $[\frac{3}{4}D, D)$. In this case, the resultant sequence of digits is 0 1 1 0. Changing this to canonical form, it becomes $10\bar{1}0$, which also contains two nonzero bits. Thus, the SD representation obtained for the quotient is a minimal SD representation. The average number of zeros, in the quotient, is 66.7%.

By exploiting the redundancy of the signed-digit representation it is possible to

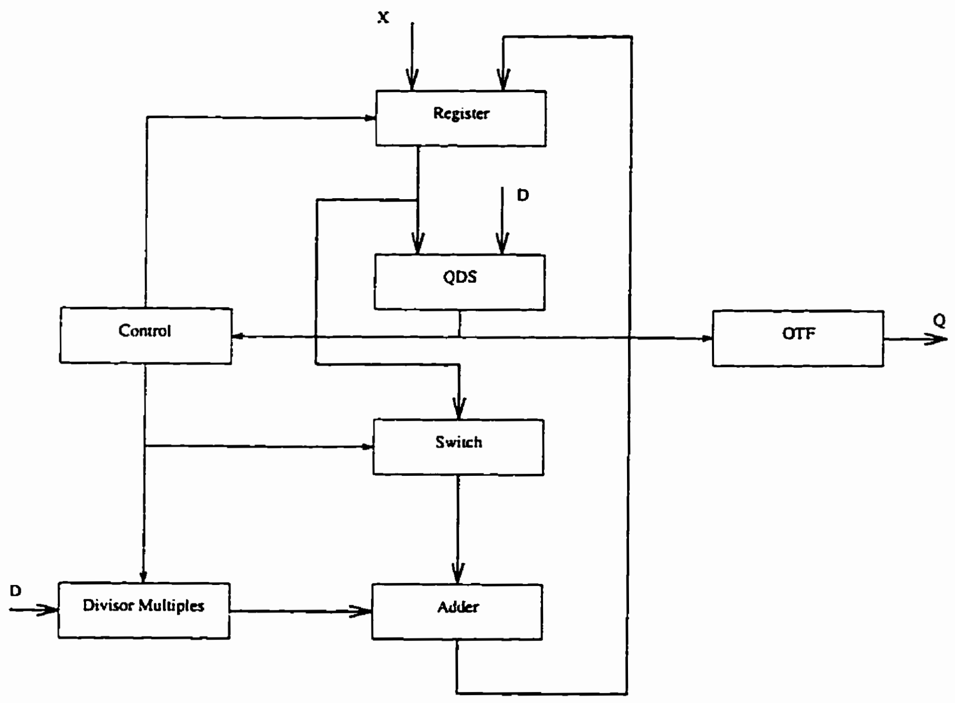


Figure 3.18: Minimum Add/Sub Division Algorithm.

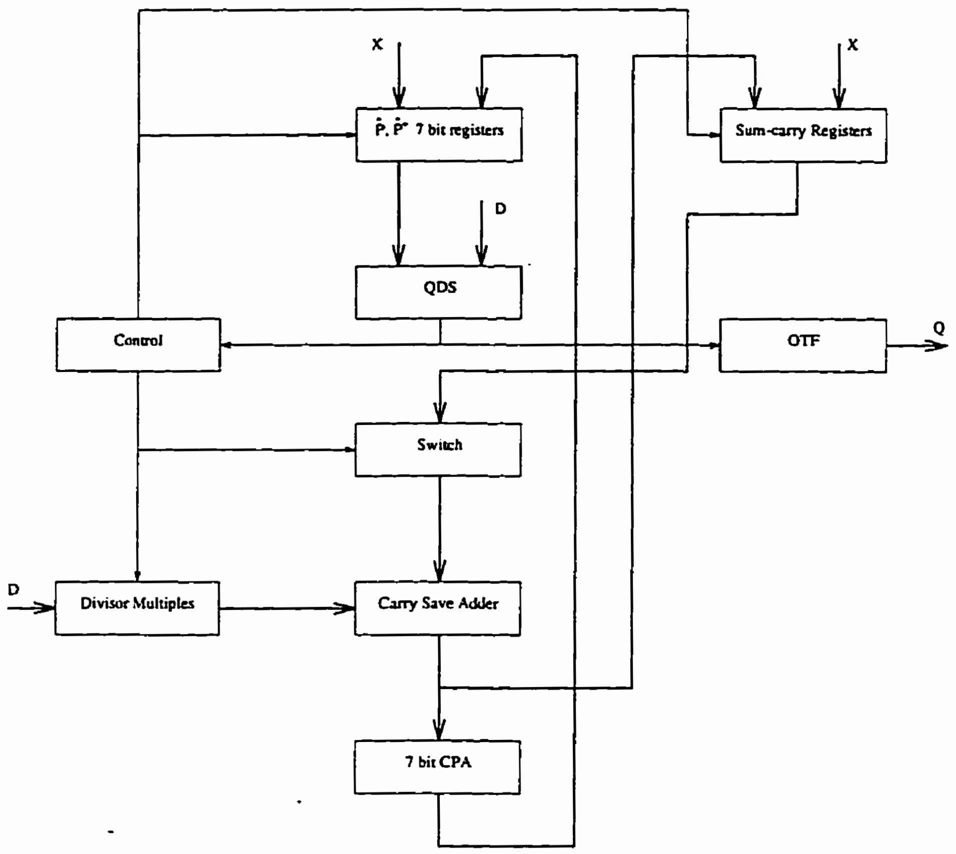


Figure 3.19: Minimum add/sub division algorithm using a limited precision QDS and a CSA.

use a limited precision QDS which has a lower complexity, and to use a CSA adder. This leads to lower power dissipation and to faster operation [92]. Figure 3.19 shows the block diagram of the modified algorithm.

The updating of the $\hat{P}$ and $\hat{P+}$ registers proceeds as follows:

1. If an addition operation occurred in the last cycle, the $\hat{P}$ and $\hat{P+}$ registers are loaded with the values produced by the CPA.

2. If no addition operation occurred in the last cycle, $\hat{P}$ and $\hat{P+}$ are updated as

follows:

if $S_o^{<8>}C_o^{<8>} = 11$

$\hat{P}_n^{<1:6>} <= \hat{P}+_o^{<2:7>}$

$\hat{P}_n^{<7>} <= 0$

$\hat{P}+_n^{<1:6>} <= \hat{P}+_o^{<2:7>}$

$\hat{P}+_n^{<7>} <= 1$

if $S_o^{<8>}C_o^{<8>} = 01$ or 10

$\hat{P}_n^{<1:6>} <= \hat{P}_o^{<2:7>}$

$\hat{P}_n^{<7>} <= 1$

$\hat{P}+_n^{<1:6>} <= \hat{P}+_o^{<2:7>}$

$\hat{P}+_n^{<7>} <= 0$

if $S_o^{<8>}C_o^{<8>} = 00$

$\hat{P}_n^{<1:6>} <= \hat{P}_o^{<2:7>}$

$\hat{P}_n^{<7>} <= 0$

$\hat{P}+_n^{<1:6>} <= \hat{P}_o^{<2:7>}$

$\hat{P}+_n^{<7>} <= 1$

where,

S and C are the sum and carry registers respectively.

Subscripts n and o denote the new value and the old value of the registers respectively.

The superscript is the bit position with respect to the partial remainder.

In order to find the average number of zeros in the quotient, due to the difficulty in solving this problem analytically or even numerically, we had to restore to simulations. The simulation has been performed on half-a-million randomly generated numbers. The average percentage of zeros in the quotient has been found to be in the order of 65%, which is quite close to the previous percentage of 66.7%.

The power dissipation of the proposed division algorithm is compared to that of Radix 2 and Radix 4 division algorithms [93]. Figure 3.16 shows the block diagram of the Radix 2 and Radix 4 division algorithms [93], used in the comparisons. The Radix 2 division algorithm generates one bit per iteration. The number of iterations is equal to the number of digits in the quotient. Each iteration requires one addition and one QDS operation.

The Radix 4 division algorithm generates two digits per iteration. The number of iterations is thus half the number of iterations required for the Radix 2 division algorithm, reducing the number of additions and QDS operations by 50%. However, the reduction in power dissipation is less than 50%, because of the increase in complexity as explained previously.

The proposed division algorithm reduces the number of addition/subtraction operations to 33% of the number required by the Radix 2 division algorithm. Compared to the Radix 4 division algorithm it is reduced by 33%. However, there is an increase in the number of QDS operations for the proposed algorithm over that of the Radix 4 algorithm.

The power, speed and area performance of the different division algorithms have been compared using computer simulation. A VHDL description has been written for each module of these algorithms. The VHDL files have then been synthesized in a $0.8\mu m$ 5 Volt Standard Cell CMOS technology. Through simulation, the per-

Table 3.11: Performance comparisons of the different division algorithms. The power is measured at a speed of 15 M division operation per second.

Features	Radix 2 Algorithm	Radix 4 Algorithm	Proposed Algorithm
Area (μm)	1600	1900	2000
Power (mW)	13.2	8.5	7.2
Speed (M words/s)	1.10	1.45	1.40

formance of the synthesized algorithms has been measured. Table 3.11 gives the simulation results for the division algorithms considered.

The power saving in the proposed division algorithm over that of the Radix 4 division algorithm is less than 33%, this is because some units, which operate during the first clock cycle, operate for 67% of the time. It has been found through computer simulation that the power saving for the proposed algorithm over the Radix 4 algorithm is 15%.

When the power dissipation of the proposed division algorithm is compared to that of the Radix 2 division algorithm, the power saving is found to be 45%.

3.10 Reducing the Computational Complexity: Vector Quantization Example

Vector quantization [88] is a compression technique. It exploits the correlation that exists between successive samples by quantizing a group of successive samples together. The encoder of a vector quantizer finds the representation vector closest to the quantized vector. The index of this vector is transmitted to the decoder. The decoder uses a look-up table or through computations finds the value of the representation vector.

The search process performed by the encoder, requires large computational capabilities to be performed exactly. However, approximate search algorithms exist, which greatly reduce the computational complexity with a much less degradation in performance. In this section, Full-Search Vector Quantization (FSVQ) and Tree-Structured Vector Quantization (TSVQ) [88] are considered.

Consider a vector X consisting of 8 samples. Each sample consists of 6 bits. This vector is to be approximated to the closest representation vector in set $\{C_i\}$ of 64 representation vectors. That is, the compression ratio is 8:1. The closest representation vector to vector X is the one having the minimum square error

$$E_i = \sum_{j=0}^{7}(X_j - C_{ij})^2$$

Full-Search Vector Quantization, requires the calculation of the square error for every representation vector, then comparing the square errors to find the index of the representation vector with minimum square error. This always finds the nearest neighbour, however, it requires great computational capability as can be seen from Table 3.12.

In Tree-Structured Vector Quantization , the search is performed in stages [88]. During each stage, a subset of the representation vectors is eliminated from consideration by a relatively small number of operations. In general, consider a tree n stages deep and having a branching factor m (the number of branches leaving a node) as shown in Figure 3.20. Each node in the tree has m vectors corresponding to each one of its m branches, the branch whose vector gives the minimum square

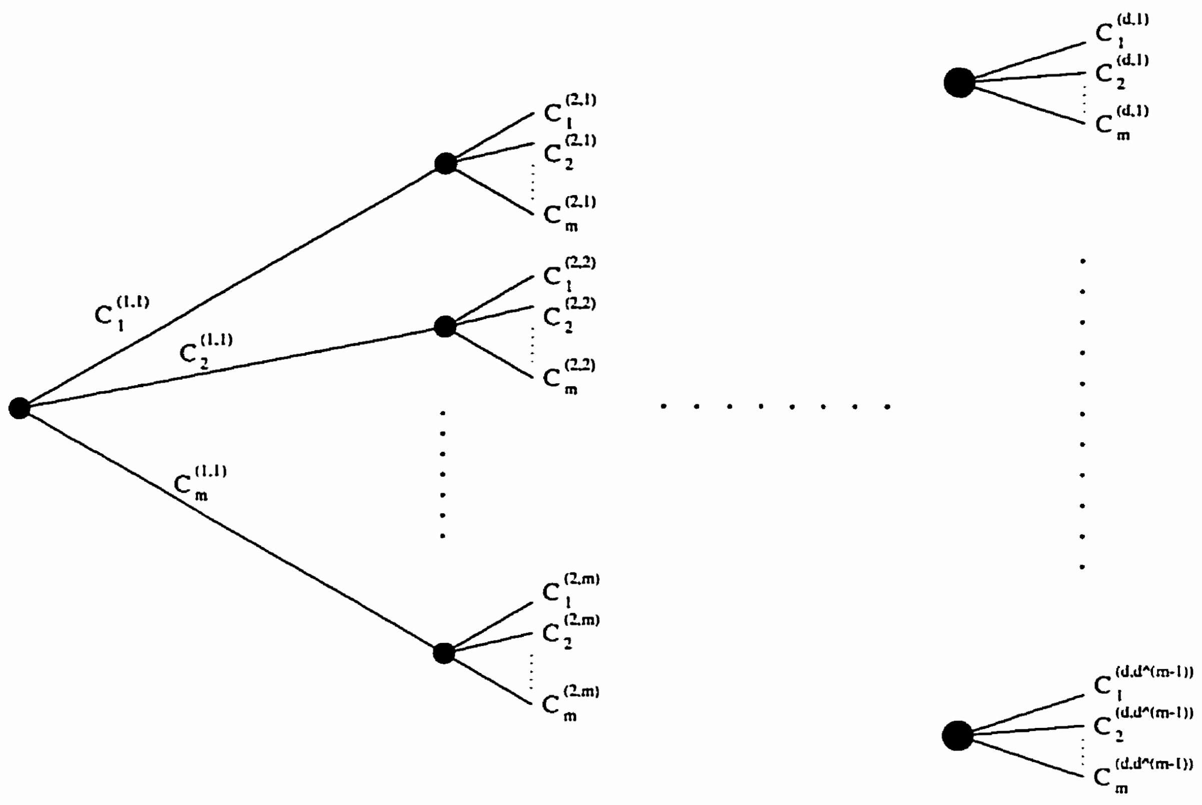


Figure 3.20: Tree-Structured Vector Quantization.

error, is chosen. The representation vectors corresponding to all the other branches are eliminated.

The total number of representation vectors is given by

$$N = m^n \tag{3.25}$$

The computational complexity of this algorithm is proportional to $m \log_m(N)$ [88]. Minimum computational complexity is achieved at $m = e$. But m has to be an integer and preferably a power of two. Hence, $m = 2$ or 4 is an optimum choice for minimum computational complexity. However, higher values of m give better performance but at the expense of greater computational complexity [88] [99]. Ta-

Table 3.12: Computational complexity and memory requirement of VQ encoding algorithms. The VQ algorithm encodes a 64-level eight-sample vector into one of 64 representation vectors.

Algorithm	# of ADD/SUB	# of Mult.	# of Mem. access	ROM size
FSVQ	960	512	512	512 Byte
TSVQ $m = 2$	180	96	96	1008 Byte
TSVQ $m = 4$	180	96	96	672 Byte
TSVQ $m = 8$	240	128	128	576 Byte

ble 3.12 compares the computational requirements for various TSVQ algorithms and that of the FSVQ algorithm.

Notice from Table 3.12, that while the computational complexity decreases the ROM size increases which can lead to an increase in the power dissipation. Another factor in choosing the vector quantization algorithm is the performance of the algorithm, generally the lower the computational complexity, the lower the performance. However, for TSVQ the degradation in performance can be quite small [88]..

3.11 Chapter Summary

Reducing the power dissipation of CMOS circuits is a necessity for the design of portable systems. In this chapter, the sources of power dissipation in CMOS circuits, the methods of estimating the power dissipation, and examples of low-power electronic systems were considered.

Reduction of the power dissipation can be achieved at the various design levels. Some of the techniques used to lower the power dissipation at the device and circuit levels were presented in this chapter. At the architectural and algorithmic levels, there is great opportunity to further lower the power dissipation.

For example, through architectural changes such as pipelining and parallelism or the use of fast adders, it is possible to increase the speed of the architecture and hence reduce the voltage to maintain the same throughput and lower the power dissipation. The effect of pipelining, parallelism, and a combination of both were considered for three different architectures of the discrete cosine transform (DCT). One of these architectures uses a fast DCT algorithm [89]. The other two depend on the use of distributed arithmetic [90].

The choice of the number system can influence the power dissipation. The Gray code representation and its power dissipation relative to the unsigned binary representation was considered. Reducing the block activity factor is another way to reduce the power dissipation. This can be done by the choice of a higher radix, or by the choice of a number representation that minimizes the number of operations required per output. Finally, the effect of using approximate search algorithms, for vector quantization, such as the tree search vector quantization algorithm [88] was considered.

Chapter 4

Subband Coding: A Low-Power Design

4.1 Introduction

The increase in demand for mobile telecommunication systems and the limited bandwidth allocated to these systems has forced research for innovative techniques to increase the spectral efficiency of mobile systems. Some of these techniques are related to the architecture of the mobile network [100] [101]. While others are based on the compression of the user information transmitted across the mobile network.

Power efficiency is of utmost importance when designing compression algorithms for mobile terminals. Compression algorithms and in particular video compression algorithms demand great computational capability [9], which in turn leads to greater power dissipation. The desire to have multimedia portable equipment has motivated work towards low-power implementations of video compression algorithms [102].

In this chapter, the design of a low-power subband coding image compression algorithm is investigated. Section 4.2 is a brief overview of video/image compression algorithms. In section 4.3, the basics of the subband image compression algorithm are reviewed.

In section 4.4, the effect of performance-power tradeoff for subband coding is considered. The structure of the analysis/synthesis filter system used in the low-power subband coding image compression algorithm is developed. A filtering structure with a small number of taps is used. The statistical properties of each subband signal is obtained. This is required to calculate the number of bits allocated to each subband. A power-efficient vector quantization algorithm, along with the architecture used to implement it are also developed in this section.

Finally, in section 4.5, the performance of the new subband coding algorithm and its power dissipation are evaluated and compared to those of conventional subband coding image compression algorithms.

4.2 Video Compression Algorithms

The information transmitted over the mobile network can be divided into three categories, data, audio and video (and image). Each type of information has its own characteristics and the corresponding compression algorithm should satisfy certain requirements. For data, it is important that the compression algorithm introduces no errors, a lossless compression algorithm must be used in this case. For audio and video, some noise is tolerable. The amount and spectral content of this noise depends on the characteristics of the human auditory and visual systems.

The compression algorithm needs to take into account the type of correlation (redundancy) in the signal to be compressed signal. Audio signals have one dimensional correlation (temporal redundancy). Images have two dimensional correlations (spatial redundancy). While video signals have both spatial redundancy as well as temporal redundancy.

The target of image/video compression is to reduce the bit-rate required to transmit the signal, while maintaining its quality. A digital picture at TV resolution requires about one million bytes without compression. Hence direct use of digital transmission or storage will not be efficient. Current image/video compression standards offer from 1/10 to 1/50 compression ratios without affecting the image quality.

The following values, give the relation between the amount of compression and the quality of the video signal [103]:

- 0.25–0.5 bpp (bit per pixel) moderate to good quality. Adequate for some applications.
- 0.5–0.75 bpp good to very good quality. Adequate for many applications.
- 0.75–1.5 bpp excellent quality. Adequate for most applications.
- 1.5–2.0 bpp usually indistinguishable from original. Adequate for most demanding applications.

Any video compression algorithm can be divided into three parts [104], as shown in Figure 4.1:

1. Signal processing. This is required to prepare the signal for quantization so that it can give better performance. For example:

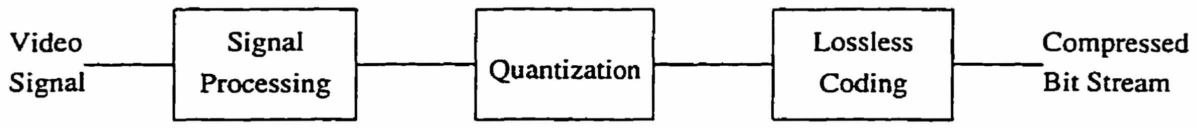


Figure 4.1: General block diagram of a image/video compression algorithm.

- DCT for JPEG [103]
- Motion Compensated DCT (MC-DCT) for H.261, MPEG-1 and MPEG-2 [105] [106].
- Subband filtering for subband coding of images [107].

2. Quantization. This is where all the lossy compression occurs. The quantization can be:

   - Scaler quantization.
   - Vector quantization [88]. The complexity of the vector quantization algorithm used is a function of the required SNR, the required compression ratio and the allowed system complexity which is determined by factors such as cost and power dissipation.

3. Lossless coding. This is where lossless compression occurs. Examples of this type of coding include, zero-runlength coding and Huffman coding.

In the signal processing part, we convert two correlated random variables into two new random variables with little or no correlation between them. This can be done by:

1. Linear prediction [88].
2. Orthogonal transformation. e.g. DCT [103].

3. Subband filtering and wavelet transform [108] [109].

Discrete cosine transform (DCT) based compression algorithms such as JPEG and MPEG are computationally intensive. A two-dimensional DCT algorithm requires in the order of $N^2 \log_2 N$ multiplications [9] [110]. Distributed arithmetic [90] can also be used in the implementation of the DCT [111] [112]. Distributed arithmetic architectures dissipate more power but have a higher throughput [113].

Subband coding has the potential of having a low computational complexity and hence low computational power dissipation. This is at the expense of some degradation in the performance of the algorithm. In this chapter, subband coding image compression algorithms are investigated. An implementation with lower complexity and hence lower power dissipation is presented.

4.3 Subband Coding for Image Compression

Subband coding was originally introduced for digital speech coding in [114]. One of the operations required to transmit speech digitally is quantization. Direct quantization introduces noise which is spread equally over most of the speech spectrum. However, the quantization noise is not equally detectable at all frequencies. Dividing the signal into subbands and quantizing each one of these subbands independently offers greater control over the spectrum of the quantization noise. Frequency bands with higher subjective importance are coded with higher resolution than other bands. Subband coding was considered for video and image applications in [109] [107] [115].

The basic idea of subband coding is to decompose the image into several subbands using a filter bank and to quantize and code the subbands instead of the original image. The quantization characteristics of the various bands can be made to match the psycho-visual characteristics of the human visual system. Such that the higher spatial frequency components are quantized with a larger quantization step size than the lower ones.

Subband coding, unlike DCT-based coding techniques, doesn't introduce blocking artifacts. This is because DCT-based systems process separately adjacent blocks, while subband systems process overlapping blocks of the signal. Subband coding allows bit allocation according to the spectral importance of each band. Subband coding systems consist of two parts:

1. The analysis/synthesis filter banks.
2. The coding system which determines how the subbands are quantized and coded. Examples of coding systems include:
   - Predictive coding (DPCM) [107].
   - VQ within subbands.
   - VQ across subbands.
   - Predictive VQ.

Analysis/synthesis filter banks [115] — [120] divide the signal into subbands and then reconstruct the signal from its subband components. To be useful for image applications the filters have to be two dimensional. A 4-band 2D-filter bank is shown in Figure 4.2. The output of each bank corresponds to a certain part of the 2D spectrum as shown in Figure 4.3. After filtering, decimation by a factor of two in each dimension is required, this makes each subband signal a fullband one at the lower sample rate.

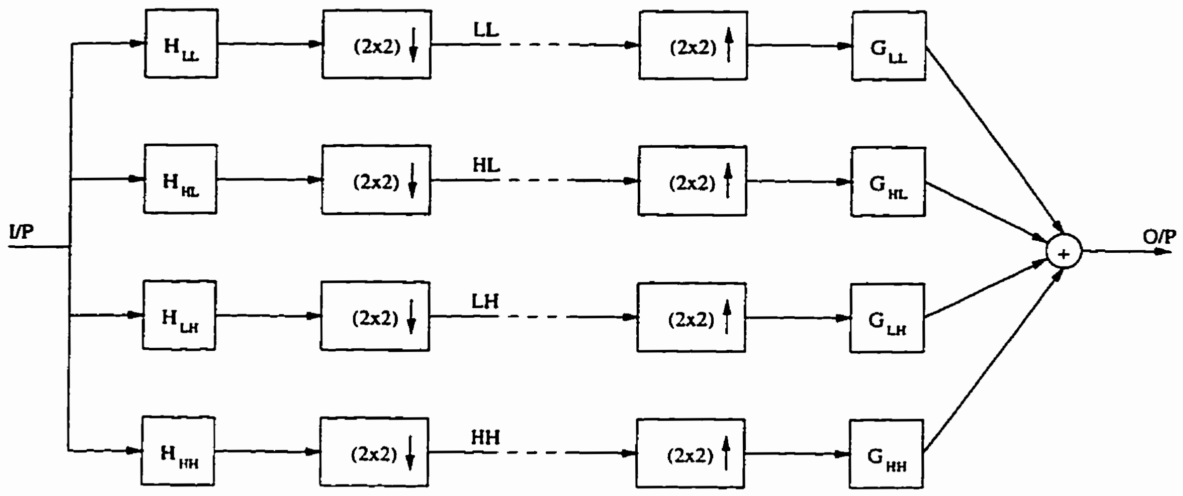


Figure 4.2: A 2D, 4-band, 1-level analysis/synthesis system.

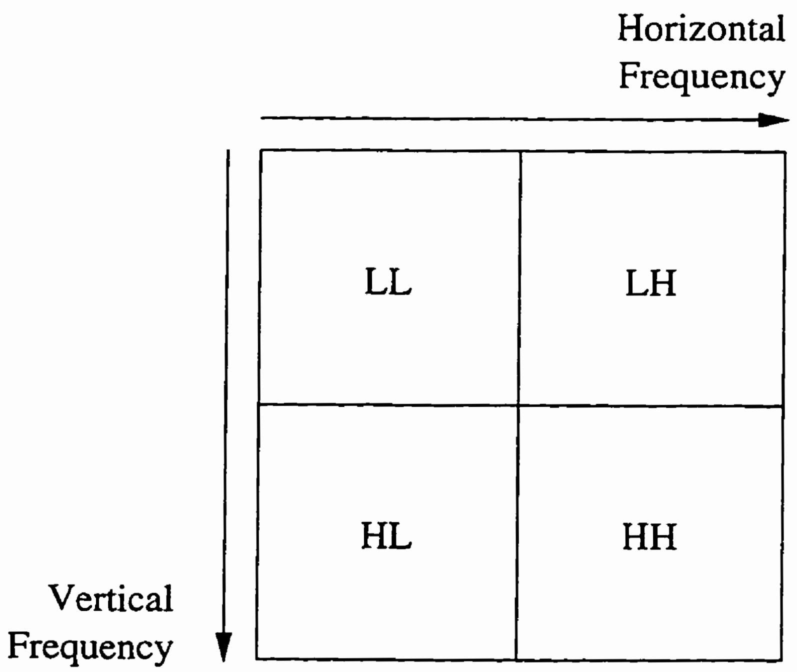


Figure 4.3: Frequency partitioning among the different bands.

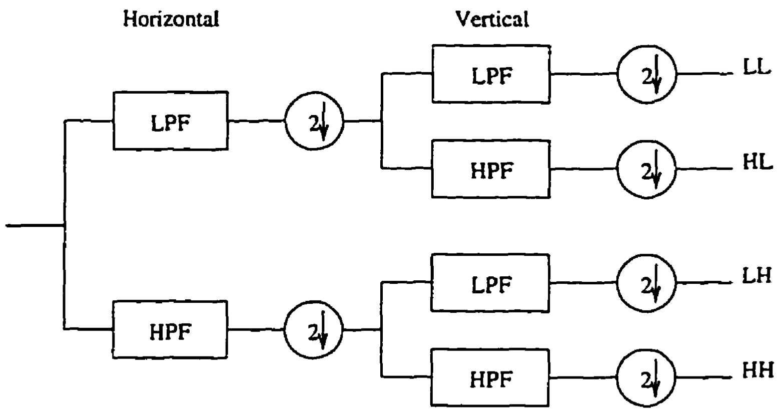


Figure 4.4: Block diagram of a one-level 2D subband analysis filter bank.

2D filter banks can be implemented using separable filters [115] which have the advantage of computation simplicity, while they lack the directional capability of nonseparable 2D filter banks. Separable 2D filter banks are implemented using a two-level 1D filter bank as shown in Figure 4.4 [109].

It is possible to cascade the filter banks and continue the frequency band division process to any desired degree. It is common to do the frequency band division to the low frequency band and leave the high frequency bands undivided [109], because the low frequency subband is correlated and contains most of the energy.

The filter banks used in subband image coding have to satisfy the following criteria:

1. Alias-free operation. This is not guaranteed due to the impossibility of implementing ideal lowpass or highpass filters.
2. Perfect reconstruction, this involves:
    - Avoiding amplitude distortion.
    - Avoiding phase distortion.

It has been shown [116] [121] that the following equations are necessary and sufficient conditions to satisfy the above criteria:

1. To remove alias distortion:

$$H_0(e^{jw}) = H_1(-e^{jw}) \tag{4.1}$$

therefore,

$$h_0(n) = (-1)^n h_1(n) \tag{4.2}$$

and

$$G_0(e^{jw}) = -G_1(-e^{jw}) \tag{4.3}$$

therefore

$$g_0(n) = -(-1)^n g_1(n) \tag{4.4}$$

2. To remove amplitude and phase distortions:

$$H_0(e^{jw})G_0(e^{jw}) - H_0(-e^{jw})G_0(-e^{jw}) = e^{-jw\delta} \tag{4.5}$$

If we take,

$$H_0(e^{jw}) = G_0(e^{jw}) \tag{4.6}$$

then,

$$H_0^2(e^{jw}) - H_0^2(-e^{jw}) = e^{-jw\delta} \tag{4.7}$$

Where, $H_0(e^{jw})$ and $H_1(e^{jw})$ are the frequency response of the lowpass and highpass filters on the analysis side respectively. While, $G_0(e^{jw})$ and $G_1(e^{jw})$ are the frequency response of the lowpass and highpass filters on the synthesis side respectively.

Consider now the case of a symmetric FIR filter with N taps,

$$h_0(n) = h_0(N-1-n) \tag{4.8}$$

N must be even, and the distortion free condition becomes,

$$|H_0(e^{jw})|^2 + |H_0(-e^{jw})|^2 = 1 \tag{4.9}$$

An analysis/synthesis filter bank satisfying Equations 4.1, 4.3, 4.6 and 4.9 produces an output which is an exact replica of the input except for a delay.

The other part of the subband image coding system is the quantization part [107] [109] [122]. To quantize the subbands efficiently, the statistical characteristics of the subband signals have to be investigated. The image signal exhibits a great deal of correlation in both the vertical and horizontal directions. In fact, the autocorrelation function (acf) [123] can be approximated by a separable negative exponential function [109]

$$R_x(m,n) = \sigma_x^2 \rho_v^{|m|} \rho_h^{|n|} \tag{4.10}$$

The low frequency subband acf can be fitted to this negative exponential distribution, while the degree of correlation in the higher frequency subbands becomes weaker [109] [122]. Hence, it has been suggested [122] [124] to use DPCM for the lower frequency subband and to use PCM for the other subbands.

The probability distribution function of the prediction error for the lower frequency subband and of the actual samples for the other subbands was found to follow the Generalized Gaussian pdf [109]. This pdf is given by

$$p(x) = a \exp(-|bx|^{\gamma}) \tag{4.11}$$

Where,

$a = \frac{b\gamma}{2\Gamma(\frac{1}{\gamma})}$

$b = \frac{1}{\sigma}\sqrt{\frac{\Gamma(\frac{3}{\gamma})}{\Gamma(\frac{1}{\gamma})}}$

$\Gamma(.)$ is the Gamma function. The value of γ depends on the subband. For the prediction error of the low frequency subband, $\gamma = 0.75$. For the other subbands, $\gamma = 0.5$. Knowing the variance and probability distribution of each subband we can allocate bits to the subbands [125] in accordance and determine the quantization intervals using Max-Lloyd algorithm [88] [126].

Knowing the behaviour of the signal statistically allows the investigation of tradeoffs in computational complexity for the sake of lower power dissipation with minimum effect on performance. Furthermore, knowing the statistics of the signal allows an estimation of the switching activity and hence an estimation of the power dissipation.

4.4 Performance-Power Tradeoff for Subband Coding

Any subband image compression algorithm consists of two subsystems. The analysis/synthesis subsystem and the coding subsystem. The complexity of the anal-

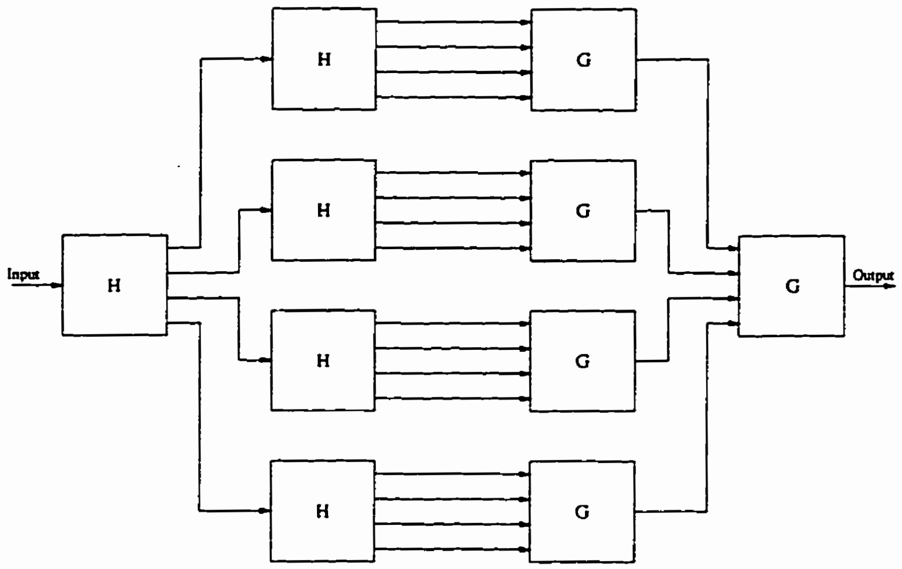


Figure 4.5: A 16-subband 2-level analysis/synthesis system.

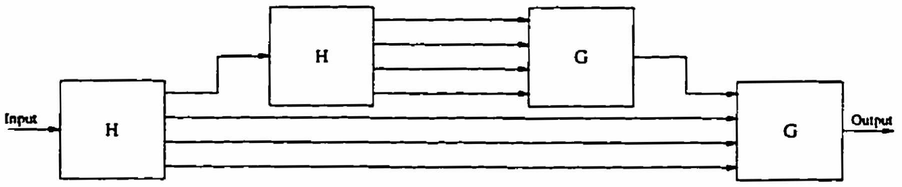


Figure 4.6: A 7-subband 2-level analysis/synthesis system.

ysis/synthesis subsystem depends on the number of filter bank levels and on the length (number of taps) of the filters used.

To lower the complexity and hence the power dissipation, it is desirable to decrease the number of filter bank levels used. In [126], the optimum SNR was found to be for a two-level system consisting of 16 subbands, shown in Figure 4.5. A two-level system consisting of only 7 subbands, with the lower frequency subband of the first level being the only level divided into smaller subbands, as shown in Figure 4.6, has a 1 dB degradation over the 16 subband system [126].

The hardware complexity of the analysis/synthesis system shown in Figure 4.6

is 60% lower than that of Figure 4.5. To compare the computational complexity, we have to consider the rate at which each filter bank operates at. The filter banks in the second level operate at quarter the speed of the filter banks in the first level due to decimation. Hence, the reduction in computational complexity for the 7-subband system over that of the 16-subband system is 37.5%.

The length of the filters also determines the complexity of the analysis/synthesis subsystem. In [126], it was found that the improvement in the SNR for FIR filters with more than 8 taps doesn't exceed 1 dB, and the improvement in SNR for filters with more than 12 taps doesn't exceed 0.2 dB. This indicates that it is possible to achieve good SNR performance with reasonable length filters.

There are several ways to do coding in the subband image compression system. Using scaler quantization [107] [122] is the least complex scheme and hence it is expected to have the least power dissipation. Vector quantization coding [7] [127] [128] [129] has greater complexity but with superior performance.

4.4.1 The Analysis/Synthesis Filter

For video applications, the analysis/synthesis filter banks have to be two dimensional. A two dimensional filter bank can be decomposed into two levels of one dimension filter banks as shown in Figure 4.4. For perfect reconstruction, Equations 4.1, 4.3, 4.6, and 4.9 have to be satisfied. To reduce the design complexity and hence the computational power dissipation, the filters should have the smallest number of taps. Take M, the number of taps, to be two. The set of filters satisfying the prefect reconstruction conditions [116] [117] are given by:

$$H_0 = \frac{1}{2}(1 + e^{-jw}) \tag{4.12}$$

$$H_1 = \frac{1}{2}(1 - e^{-jw}) \tag{4.13}$$

$$G_0 = (1 + e^{-jw}) \tag{4.14}$$

$$G_1 = (e^{-jw} - 1) \tag{4.15}$$

These filters simply find the sum and difference between two successive samples. Using the average and half the difference at the receiver side it is possible to reconstruct the original signal perfectly in the absence of quantization error. What makes this filter intuitively pleasing is that video signals are highly correlated and hence most of the energy is compact into the average component making that of half the difference component quite small.

Another advantage of these filters, from the power dissipation point of view, is that it needs no multipliers. Multiplication by half is just a shift right operation which can be accomplished by the reodering of the datapath.

Figure 4.7 shows the frequency spectrum of both the lowpass and the highpass filters. Notice that, the over simplified structure has lead to a poor frequency response. However, it remains to be seen if through the use of efficient coding we can compensate the poor frequency response. Remember that the filter banks in themselves introduce no distortion. The distortion is actually produced during quantization.

Figure 4.8 shows the simplified subband coding algorithm. Initially, the image is divided up into partitions each containing 4 samples $S_0 \ldots S_3$. These samples are

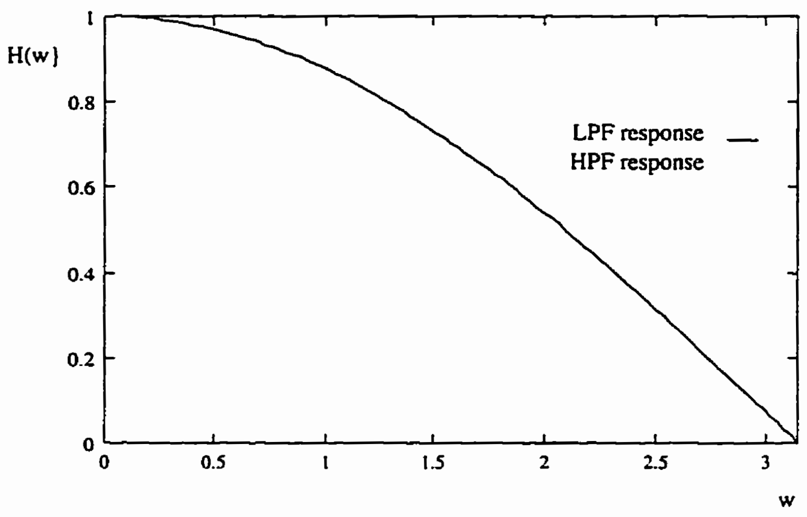


Figure 4.7: Frequency spectrum of the simplified filters.

transformed into the variables A, B_1, B_2 and B_3 according to the following set of equations:

$$A = \frac{(S_0 + S_1) + (S_2 + S_3)}{4} \tag{4.16}$$

$$B_1 = \frac{(S_0 + S_1) - (S_2 + S_3)}{4} \tag{4.17}$$

$$B_2 = \frac{(S_0 - S_1) + (S_2 - S_3)}{4} \tag{4.18}$$

$$B_3 = \frac{(S_0 - S_1) - (S_2 - S_3)}{4} \tag{4.19}$$

The samples corresponding to the variable A are partitioned into groups of 4. These samples are than transformed into the variables C, D_1, D_2 and D_3 according to the following set of equations:

$$C = \frac{(A_0 + A_1) + (A_2 + A_3)}{4} \tag{4.20}$$

$$D_1 = \frac{(A_0 + A_1) - (A_2 + A_3)}{4} \tag{4.21}$$

$$D_2 = \frac{(A_0 - A_1) + (A_2 - A_3)}{4} \tag{4.22}$$

$$D_3 = \frac{(A_0 - A_1) - (A_2 - A_3)}{4} \tag{4.23}$$

4.4.2 Statistical Properties of the Subband Coded Signal

The image signal is a highly correlated signal. Hence, it is expected that the energy content in the low frequency part of the spectrum to be much higher than that in the high frequency part. Figure 4.10 shows the statistical distribution of the first level subband signals of the aeroplane, shown in Figure 4.9. The low frequency subband (LL1) is further decomposed into four subband signals the statistical distribution of which are shown in Figure 4.11.

Table 4.1 gives the variance for each subband of the two-level decomposed image. For LL2, the variance of the successive sample difference is given. For that subband, the adjacent samples are still correlated hence DPCM is used to encode it [122]. For the other subbands, PCM or vector quantization is used depending on the number of bits allocated to that subband. The vector quantization algorithm is explained in the next section.

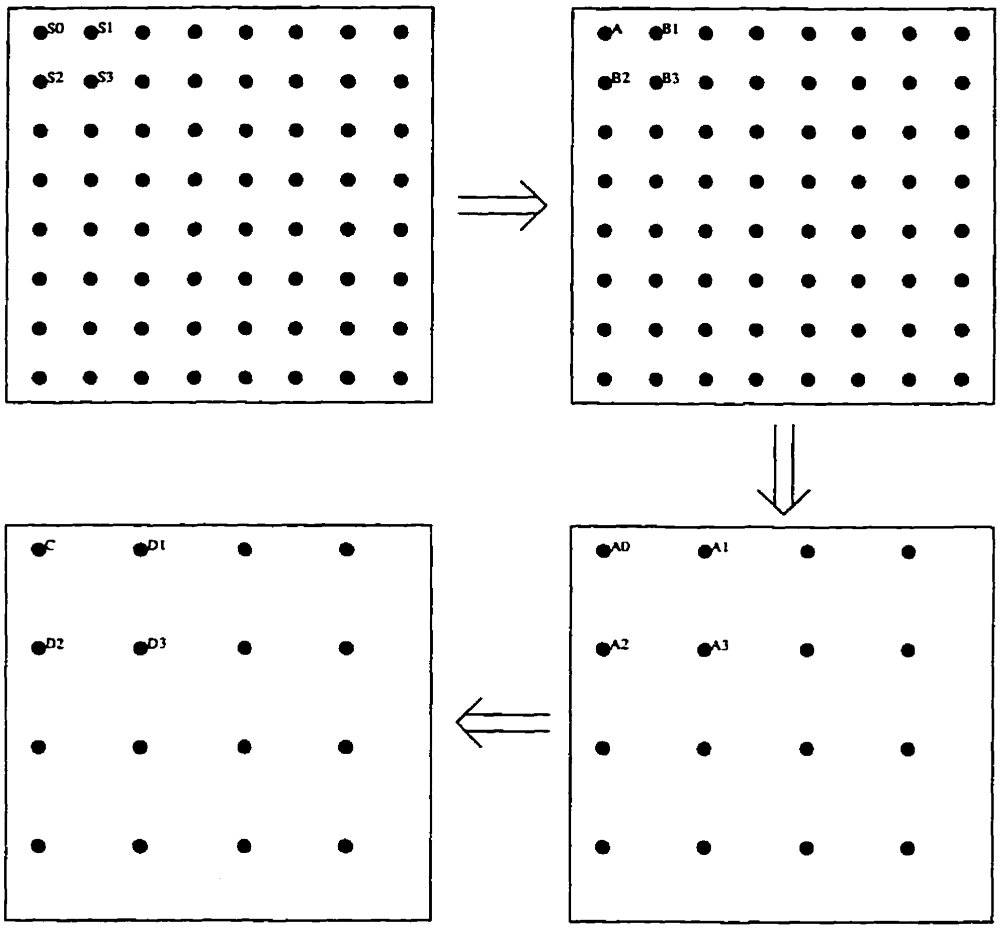


Figure 4.8: Simplified subband coding algorithm.

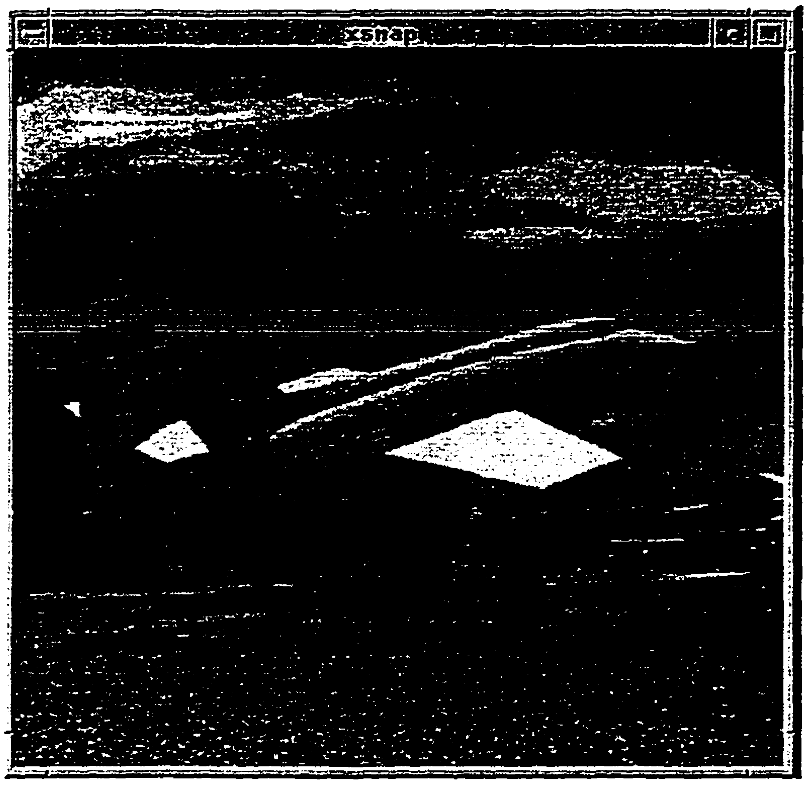
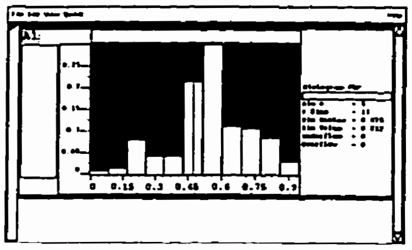


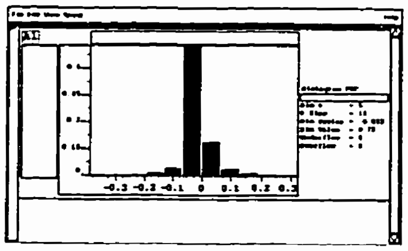
Figure 4.9: Aeroplane: The image used in subband coding.

Table 4.1: Variance for the two-level subband image compression system.

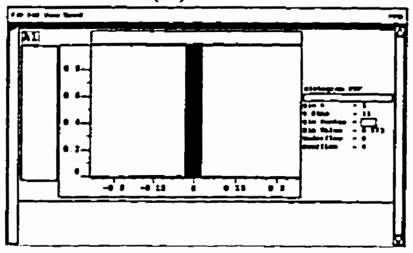
Subband	Variance
HL1	0.00158
LH1	0.000713
HH1	0.000129
LL2	0.00248
HL2	0.00396
LH2	0.00191
HH2	0.000286



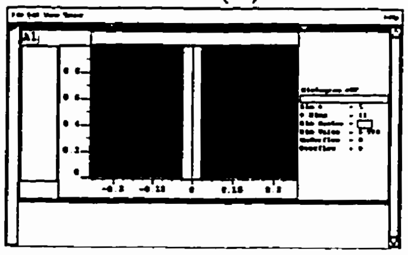
(a)



(b)



(c)



(d)

Figure 4.10: Statistical distribution of level one suband signals.

(a) LL1. (b) HL1.
(c) LH1. (d) HH1.

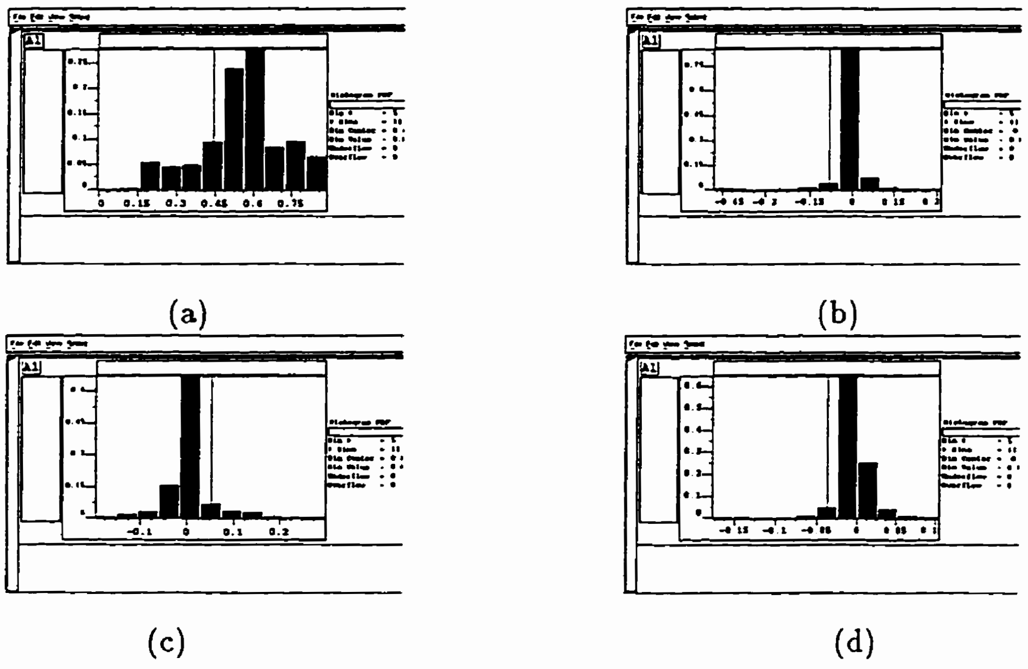


Figure 4.11: Statistical distribution of level two subband signals.

(a) LL2. (b) HL2.
(c) LH2. (d) HH2.

4.4.3 The Vector Quantization Algorithm

Vector quantization groups samples together, codes are assigned to the most likely patterns in the sequence of samples in such a way that the mean square error (MSE) is minimized. The vector quantization algorithm needs to be designed to minimize the power dissipation during decoding. The decoding of most VQ algorithms requires a memory lookup table [88]. However, memory access has large power dissipation [7]. Hence, to be power-efficient, the decoding of the VQ algorithm should be done in a computational way, with the least amount of computations.

The algorithm considered here is a simplified modification of the pyramid vector quantization algorithm (PVQ) [7]. In the case of subband coding, the samples of the upper frequency subbands are usually around the zero except near the edges. For a vector of length N, assume that at most two samples are nonzero. Each nonzero sample is encoded using two bits. The total number of possible ways in which two and only two samples out of N can be nonzero is given by:

$$\frac{N(N-1)}{2} \tag{4.24}$$

In addition, there are $N+1$ alternatives in which only one sample or no samples are nonzero. Assuming that $N = 2^n$, it can be shown that the total number of bits required to encode one vector is $2n + 3$. Hence, the number of bits per sample is given by:

$$\frac{2n+3}{2^n} \tag{4.25}$$

Clearly, increasing n leads to higher compression. In the following, it will be explained how the information of one vector is encoded into the $2n+3$ bits, in such a way that its decoding requires minimum computation.

Assume that two samples are nonzero, let i be the most significant of these samples and let j be the least significant. Most significant and least significant refers to the position of the sample in the vector. Divide the $2n+3$ bits required for each vector into three parts, the first part is $n-1$ bits long, assume these to be the most significant bits, and denote them by L_1. The second part is n bits long, and is denoted by L_2. The third and final part is 4 bits long and is denoted by L_3. If

$$i = k$$

then j can range from 0 to $k-1$, that is j can assume any one of k values. If,

$$i = 2^n - k$$

then j can range from 0 to $2^n - k - 1$, that is j can assume any one of $2^n - k$ values. It is thus evident that the two previous values of i are complementary in the sense that the total values j can take in both cases is 2^n. Hence, it is reasonable to group these two values together. The value of i is determined by L_1. In this case:

$$i = L_1 + 1 \tag{4.26}$$

or,

$$i = OC(L_1) \tag{4.27}$$

Where OC is the one's complement operation assuming n bits. To determine which value of i to choose, we have to look at L_2. If,

$$L_2 + OC(L_1) \geq 2^n \tag{4.28}$$

then,

$$i = OC(L_1) \tag{4.29}$$

and

$$j = L_2 + OC(L_1) \tag{4.30}$$

else

$$i = L_1 + 1 \tag{4.31}$$

and

$$j = L_2 \tag{4.32}$$

The field L_3 determines the values of the two nonzero samples, two bits per sample.

The above equations are valid for $L_1 = 0, 1, \ldots, (2^{n-1} - 2)$. However, if $L_1 = 2^{n-1} - 1$ the following set of equations are used instead:

1. If the most significant bit of L_2 is zero.

$$i = 2^{n-1} \tag{4.33}$$

   and

$$j = L_2 \tag{4.34}$$

2. If the most significant bit of L_2 is one. In other words, the most significant n bits of the decoded word are one. Then there is only one or no non-zero samples.
   - If the second most significant bit of L_2 is one. All the N samples are zeros.
   - If the second most significant bit of L_2 is zero. One and only one sample of the vector is nonzero. The address of this sample is determined by the $n-2$ least significant bits of L_2, and the two most significant bits of L_3. In this case, the two least significant bits of L_3 determine the value of the nonzero sample.

Now that the algorithm has been described, it has to be seen how it can be mapped into a power-efficient architecture. Assume that the word to be decoded of length $2n+3$ bits is stored in register R which has three fields R_1, R_2 and R_3, corresponding to L_1, L_2 and L_3 respectively. Let R_N be the vector corresponding to the N samples. Initially, the samples of this vector are set to zero.

Let A_0 be the ANDing of the $n+2$ most significant bits of R, and let A_1 be the ANDing of the $n+1$ most significant bits of R. If A_0 is high, the N samples of the vector are all zeros, which is the value already contained in R_N. All the other functional units are deactivated in this case.

If A_1 is high, while A_0 is low, then one and only one sample is nonzero. The address of this sample is determined by the $n-2$ least significant bits of R_2 and the two most significant bits of R_3. While the value of the nonzero sample is determined by the two least significant bits of R_3. In addition to the AND operation, one ROM access is required to determine the value of the nonzero sample. One mux operation is required to determine the address of the nonzero sample. Finally, one write operation into a bank of N registers is required.

Finally, if A_1 and A_0 are both zeros, there will be two nonzero samples and their addresses have to be calculated. This process requires, two ROM accesses, two ADD operations, one INV operation, three MUX operations, and two write operations into a bank of N registers. The architecture required to implement the proposed vector quantization decoding algorithm is shown in Figure 4.12

4.5 Performance of the Subband Coding Algorithm

Two cases of the subband algorithm are considered. The first is the one-level subband algorithm. The second is the two-level subband algorithm, the second level is the decomposition of the low-frequency subband of the first level.

Table 4.2 gives the variance of each subband, the theoretical number of bits that should be allocated to each subband [88], and the actual number of bits allocated to each subband for a one-level subband system. For the low frequency subband, the variance shown is that of the differential signal. The bit rate is 1.025 bits/pixel. The overall peak signal-to-noise ratio using the aeroplane was found to be 24 dB. This is about 8 - 9 dB lower than subband coding systems using a FIR filter having more than 8 taps [126].

For the one-level subband coding image compression algorithm, DPCM is used to encode the signal of subband LL1. The proposed VQ algorithm with $n = 4$ is used to encode subband HL1. The proposed VQ algorithm with $n = 5$ is used to encode subband LH1.

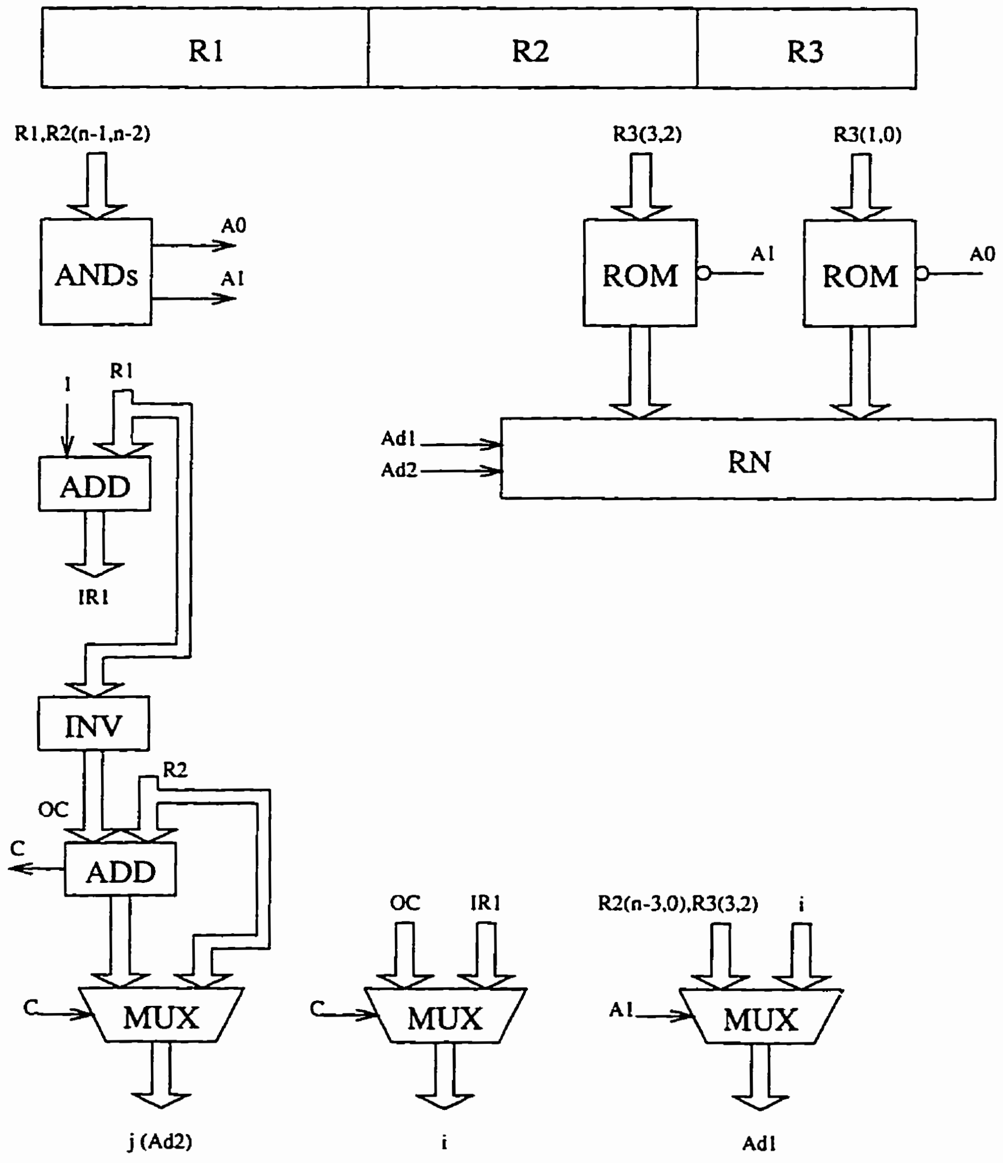


Figure 4.12: The architecture of the proposed VQ decoding algorithm.

Table 4.2: Variance and bit allocation for a one-level subband system.

Subband	Variance	Bits Required	Bits Used
LL1	0.00764	3.896	3
HL1	0.00158	1.622	0.69
LH1	0.000713	0.474	0.41
HH1	0.000129	-1.992	0

Table 4.3 gives the variance of each subband, the theoretical number of bits that should be allocated to each subband [88], and the actual number of bits allocated to each subband for a two-level subband system. For the low-frequency subband, the variance shown is that of the differential signal. The bit rate is 1.016 bits/pixel. The overall peak signal-to-noise ratio using the aeroplane was found to be 28 dB. Note that, the use of a two-level subband gave a 4 dB improvement in the SNR over the one-level subband. However, this is still 4 - 5 dB lower than subband coding systems using a FIR filter having more than 8 taps.

For the two-level subband coding image compression algorithm, DPCM is used to encode the signal of subband LL2. PCM is used to encode the signals of subbands HL2 and LH2. The proposed VQ algorithm with $n = 3$ is used to encode the signal of subband HL1. The proposed VQ algorithm with $n = 4$ is used to encode the signal of subband LH1.

Figure 4.13 shows the aeroplane after passing through a two-level subband compression/decompression system. The quality of the signal is lower than that of other subband image compression algorithms. But a large reduction has been achieved in the power dissipation. The proposed system requires only one addition/subtraction

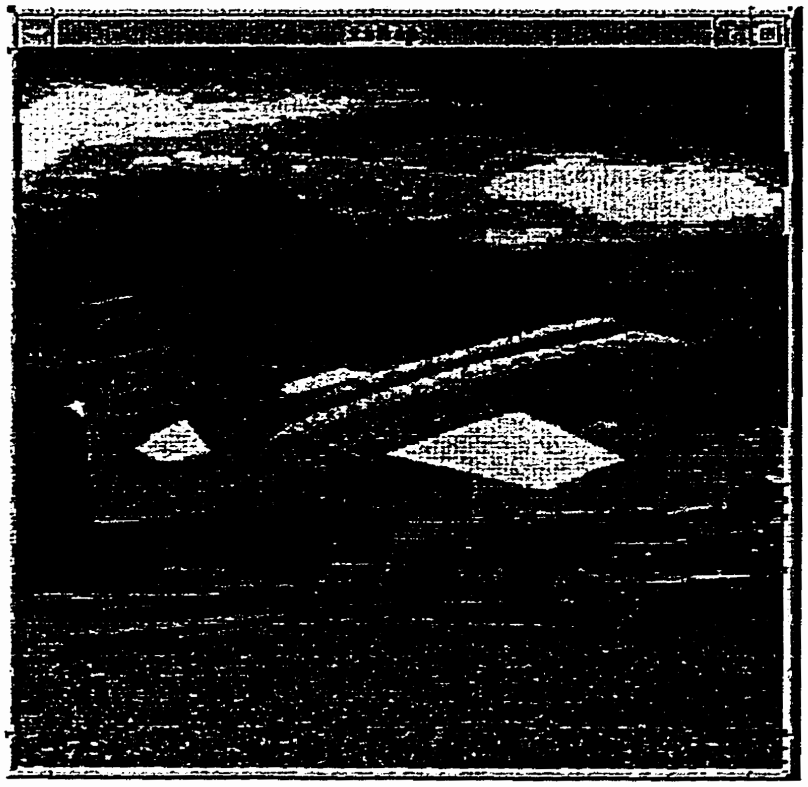


Figure 4.13: Aeroplane: The effect of the proposed two-level subband coding image compression algorithm.

Table 4.3: Variance and bit allocation for a two-level subband system.

Subband	Variance	Bits Required	Bits Used
HL1	0.00158	2.118	1.125
LH1	0.000713	0.970	0.6875
HH1	0.000129	-1.497	0
LL2	0.00647	4.152	4
HL2	0.00396	3.442	3
LH2	0.00191	2.391	2
HH2	0.000286	-0.348	0

operation for each 1D filter, while a subband system using 8 tap FIR filters usually requires 7 addition/subtraction operations and 4 multiplication operations (assuming a symmetric filter) for each 1D filter. Assuming that the 2D filter is separable, then each 2D filter consists of six 1D filters, as shown in Figure 4.4.

For the one-level subband coding system, the proposed filtering structure requires 2 ADD/SUB operations per sample. While a filtering structure based on an 8 tap FIR filter requires 14 ADD/SUB and 8 MULT operations per sample. For a two-level subband coding system, the proposed filtering structure requires 2.5 ADD/SUB operations per sample. While a filtering structure based on an 8 tap FIR filter requires 17.5 ADD/SUB and 10 MULT operations per sample. Assuming that the multiplier dissipates 4 times the adder power. The proposed filtering structure dissipates 23 times less power than a filtering structure using an 8 tap FIR filter [126].

The simulation results of the vector quantization algorithm, for the two-level

Table 4.4: The power dissipation of the proposed VQ decoding algorithm and that of a memory-based VQ algorithm. Both have the same compression factor and are designed in a 0.5μm, 3.3 Volt CMOS technology. Each sample is 4 bits. The power dissipation is calculated at a speed of 1 Vector per μs.

Samples per vector	Compression factor	Memory based		Proposed algorithm	Power
		Memory size	Power	Power	reduction
8	32/9	0.5 K $\times$ 32	0.65 mW	89 μW	7.3 times
16	64/11	2 K $\times$ 64	2.82 mW	99 μW	23 times
32	128/13	8 K $\times$ 128	17.6 mW	112 μW	156 times

subband coding image compression algorithm, show that, 40% - 50% of the vectors were zero, 15% - 25% had only one nonzero sample and 30% - 40% had two nonzero samples. The decoding algorithm thus requires 2 AND operations, 0.7 ADD operation, 0.35 INV operation, 1.25 MUX operation, 0.9 ROM access and 0.9 register write operations per vector. It should be noted that number of operations per vector is independent of the vector dimension. In PVQ the number of operations per vector is proportional with the vector dimension and it turns out to be much larger than that of the new algorithm.

Table 4.4 compares the power dissipation of the proposed VQ decoding algorithm, to that of a memory-based VQ decoding algorithm. The power dissipation of the proposed algorithm varies slightly as the vector size grows larger. This is because the number of operations required is independent of the vector size, but the datapath width of the operators increases logarithmically with the vector size. On the other hand, the size of the memory in a memory-based VQ decoding algorithm increases approximately with the cube of the vector size.

4.6 Chapter Summary

A subband coding image compression algorithm with low computational complexity has been developed in this chapter. The chapter starts with an overview of image compression algorithms and in particular subband coding image compression.

The use of a simplified filtering structure is one of the distinct features of the new subband coding algorithm. The analysis/synthesis filter system is a two-level system, with the lower frequency subband of the first level being the only one divided into smaller subbands. Addition and subtraction are the only operations used in the filter, no multiplication is required.

The statistical properties of each subband are evaluated to determine the number of bits allocated to each subband. A vector quantization algorithm which avoids the need of large look-up tables for subband decoding was developed. The filtering structure used reduces the computational power dissipation by 23 times. The reduction in computational complexity is achieved at the expense of a 4–5 dB degradation in the SNR performance, for a subband image compression algorithm employing a two-level analysis/synthesis system.

Chapter 5

A/D Converter for Software Radio

5.1 Introduction

Intricate signal processing of real world analog signals often requires signal conversion into the digital domain. Conversion makes feasible the use of either conventional digital computers or special purpose digital signal processors. This increases the systems flexibility and programmability.

Software radios require high-speed high-resolution A/D converters. To achieve a resolution of up to 20 bits a Sigma-Delta A/D converter [130] [131] [132] is used. The Sigma-Delta A/D converter is composed of a Sigma–Delta modulator, followed by a decimation filter [133] [134] [135], which digitally transforms a low-resolution oversampled signal into a high-resolution Nyquist-rate sampled signal. Figure 5.1 shows the block diagram of a bandpass Sigma-Delta A/D converter.

Figure 5.2 shows a typical receiver where the digitization is done after the first

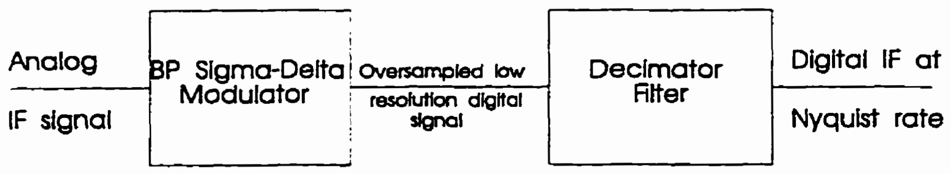


Figure 5.1: Bandpass Sigma-Delta A/D converter.

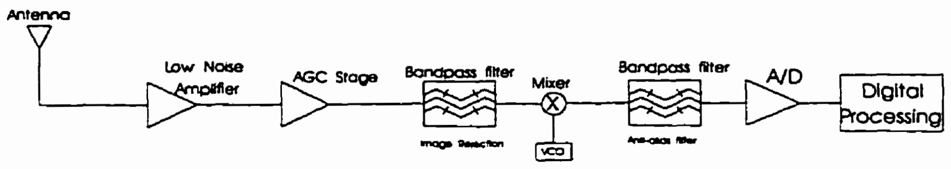


Figure 5.2: Digital IF receiver architecture.

IF (Intermediate Frequency) stage [136]. The bottle neck of this architecture is the A/D (analog-to-digital) converter. Not only does this A/D operate at a high speed (in the MHz), but it requires high resolution as well (12 - 20 bits).

Parallelism by 4x of analog signal processors is applied to the design of a bandpass Sigma-Delta modulator. The speed of the modulator is increased without increasing the speed requirement of the individual building blocks. Several architectures are considered in terms of their resilence to implementation details such as mismatch and gain errors. A switched-capacitor circuit is also given for the proposed modulator.

Several high-level low-power design techniques have been incorporated in the design of the decimation filter. These include; operation minimization, multiplier elimination, operation interleaving and block deactivation. Analysis and simulation results indicate that these techniques can achieve a 4 times reduction in power dissipation. A novel memory access algorithm is employed in the design of the lowpass filter. An interleaved multiplier-accumulator array is used in the lowpass filter. In this chapter, the effect of optimizing the datapath width of the Sinc decimator on its numerical accuracy is also considered. The decimation filter designed has a programmable resolution, that varies from 12 to 20 bits. The entire decimation filter has been designed in a $0.5\mu m$, 3.3 Volt CMOS technology.

The organization of this chapter is as follows. In the next section, the resolution requirement of the A/D converter is determined. In section 5.3, a novel architecture that applies parallelism by 4x of analog signal processors to the design of a bandpass Sigma-Delta modulator is presented. In section 5.4, the performance of the proposed Sigma–Delta architecture is evaluated for different configurations, in terms of their resilence to implementation details such as mismatch and gain errors. In section 5.5, a switched-capacitor implementation of the proposed architecture with minimum number of operational amplifiers is presented. In section 5.6, the decimation filter architecture is presented. The decimation filter designed is composed of a Sinc decimator and a lowpass decimation filter (LPDF). In section 5.7, the design of the Sinc decimator is investigated, and operation minimization is applied to minimize the power dissipation by eliminating redundant computations. In section 5.8, the effect of optimizing the datapath width on the numerical accuracy and power dissipation of the Sinc decimator is considered, this eliminates irrelevant computations. In section 5.9, the low-power design of the LPDF filter is investigated. The VLSI design of the decimation filter in a $0.5\mu m$, 3.3 Volt CMOS technology is given in section 5.10.

5.2 The Resolution Requirement

The resolution requirement of the A/D converter and hence the resolution requirement of the decimation filter varies according to the strength of the received signal as well as the background noise and interference. Simulation results for a system based on the DAMPS standard [22], indicate that for a digital receiver digitizing an IF signal of bandwidth 0.96 MHz (32 TDMA channels), the maximum required dynamic range for the A/D converter is 20 bits. Simulation results also indicate that when the input signal is strong enough a dynamic range of 10 bits is sufficient.

The total dynamic range required can be expressed as:

$$DR_{total} = f(DR_{AGC}, DR_{inter}) \tag{5.1}$$

DR_{AGC} is the required dynamic range due to the variation in the strength of the received input signal. The strength of the input signal can vary by up to 120 dB. This necessitates the use of automatic gain control (AGC) in the conventional receiver. DR_{inter} is the dynamic range required so that the digital stages following the A/D converter can distinguish the desired channel from any interference. The digitized signal includes more than one channel, the desired channel is then selected digitally, the dynamic range of the A/D converter should be enough to be able to perform this channel selection in the following digital stages. The DAMPS standard [22] specifies that the receiver should operate properly when an interference 55 dB greater than the desired signal exists 90 KHz or more from the desired signal.

Assume that the receiver consists of an AGC amplifier followed by the A/D converter as shown in Figure 5.3.a. The dynamic range of the A/D converter in this case is 55 dB ($\approx$ 10 bits). The AGC amplifier is required to have a gain variation of $120 - 55 = 65$ dB.

Without the AGC amplifier, Figure 5.3.b, the required dynamic range of the A/D converter is 120 dB ($\approx$ 20 bits). However, the high resolution is not required in all cases. In fact, if the input signal is large enough, which corresponds to the AGC amplifier of Figure 5.3.a having a gain G_{min}, a 12 bit A/D converter is all

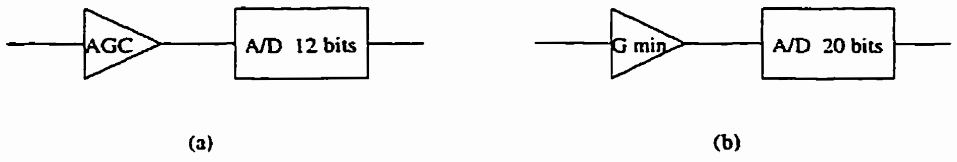


Figure 5.3: A/D resolution requirement: (a) With AGC amplifier. (b) Without AGC amplifier.

that is required. If the input signal is weak, the AGC amplifier of Figure 5.3.a will have a gain G_{max} ($= G_{min} + 65$ dB), which corresponds to a 20 bit A/D converter in Figure 5.3.b.

The reason for having an A/D converter with variable resolution is to save power when the lower resolution is sufficient. This leads to the concept of Automatic Resolution Control (ARC) where the resolution of the A/D converter is varied as opposed to AGC where the gain of the amplifier is controlled by the level of the input signal. The required resolution is determined by the digital stages following the A/D converter. In section 5.6 of this chapter, the design of a decimation filter, to be used with the Sigma-Delta modulator with resolution varying between 12 to 20 bits, is examined.

5.3 A Parallel Bandpass Sigma–Delta Modulator

Sigma-Delta modulation has been commonly used in high resolution analog-to-digital converters because of the ability to shape noise away from the desired band. Moreover, Sigma-Delta modulators require a two-level quantizer to achieve a high-resolution Nyquist-rate sampled-stream.

Sigma-Delta modulation has commonly been used for lowpass signals [132] [137]. However, the signal digitized at the IF stage is a bandpass signal. The signal can

Table 5.1: Dynamic range versus OSR for a second and a third order LPSD.

OSR	2nd Order LPSD	3rd Order LPSD
32	62 dB (10 bits)	84 dB (14 bits)
64	77 dB (12 bits)	105 dB (17 bits)
128	92 dB (15 bits)	126 dB (21 bits)

be subsampled with no loss of information due to aliasing. In this case, a bandpass Sigma-Delta modulator (BPSD) [138] — [142] is used instead of a lowpass Sigma-Delta modulator (LPSD).

The bandpass Sigma-Delta A/D modulators presented in the literature so far have been one channel A/D modulators, with bandwidth 30 kHz (for DAMPS systems) [141], or bandwidth 200 kHz (for GSM systems) [142]. As the digitized signal bandwidth increases, more than one channel is digitized and then the desired channel is filtered out digitally. This increases the sufficient dynamic range required to meet the standard's interference rejection criteria. To operate at a high sampling rate, parallelism by 4x of analog signal processors is applied to the design of the bandpass Sigma-Delta modulator. This increases the overall speed of the modulator without increasing the speed requirement of the individual building blocks.

The dynamic range of the Sigma-Delta A/D converter depends on the order of the Sigma-Delta modulator, as well as the oversample ratio (OSR). Table 5.1 gives the output dynamic range of the Sigma-Delta modulator for different oversampling ratios, for second-order and third-order lowpass Sigma-Delta modulators. A fourth-order BPSD is equivalent, in its dynamic range performance, to a second-order LPSD. A sixth-order BPSD is equivalent to a third-order LPSD.

To subsample a bandpass signal and ensure that spectrum overlap doesn't occur, the sampling frequency has to satisfy the following inequality [11]:

$$\frac{2f_h}{k} \leq f_s \leq \frac{2f_l}{k-1} \tag{5.2}$$

where,

f_s is the sampling frequency.

f_h is the highest frequency in the bandpass signal.

f_l is the lowest frequency in the bandpass signal.

k is an integer satisfying the following inequality:

$$1 \leq k \leq \frac{f_c}{\mathrm{BW}} + 0.5 \tag{5.3}$$

f_c is the band center frequency,

$$f_c = \frac{f_h + f_l}{2} \tag{5.4}$$

BW is the bandwidth,

$$\mathrm{BW} = f_h - f_l \tag{5.5}$$

According to Inequality 5.2, the sampling frequency depends on both, the bandwidth and the band position of the bandpass signal. Figure 5.4 [143], shows the valid sampling frequency as a function of the bandwidth BW and the center frequency f_c of the bandpass signal. Generally, the sampling frequency is given by [136]:

$$f_s = \frac{4f_c}{2k-1} \tag{5.6}$$

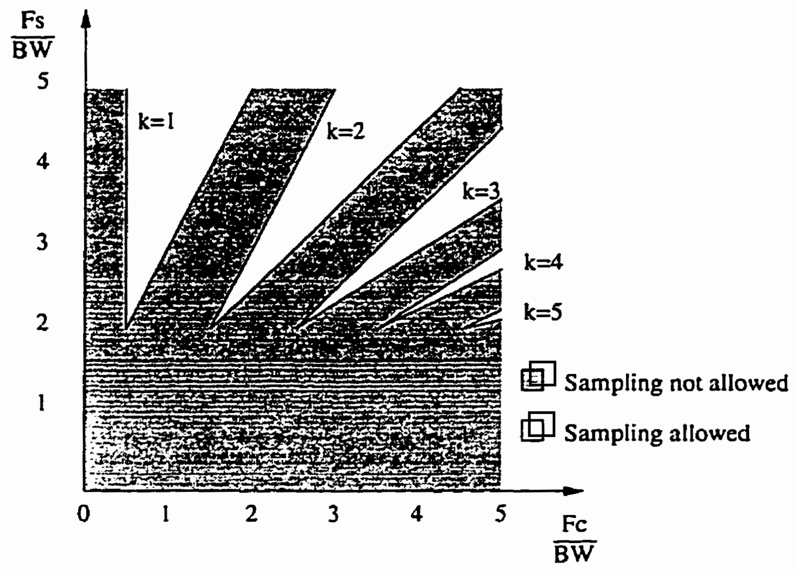


Figure 5.4: Valid bandpass sampling rate regions.

It is clear that the samples can be divided into four groups: G0, G1, G2 and G3, equally spaced in time. Groups G0 and G2 sample the in-phase channel, while groups G1 and G3 sample the quadrature-phase channel. The division of the samples into four groups suggests, that we can replace the bandpass Sigma-Delta modulator with four lowpass Sigma-Delta modulators as shown in Figure 5.5.b. Each one of these lowpass modulators would operate at quarter the speed of the bandpass modulator relaxing the circuits speed requirements. It would, however, suffer the impact of component mismatch and a loss in the dynamic range. This loss in dynamic range is 12 dB for a fourth-order bandpass Sigma-Delta modulator, and could be avoided using a cross-coupled architecture.

The conventional second-order bandpass Sigma-Delta modulator, shown in Figure 5.6.a, can be split into two branches, one for the in-phase channel and the other for the quadrature-phase channel. This is shown in Figure 5.6.b. Consider one of the branches of Figure 5.6.b. It is desirable to split that branch into two branches,

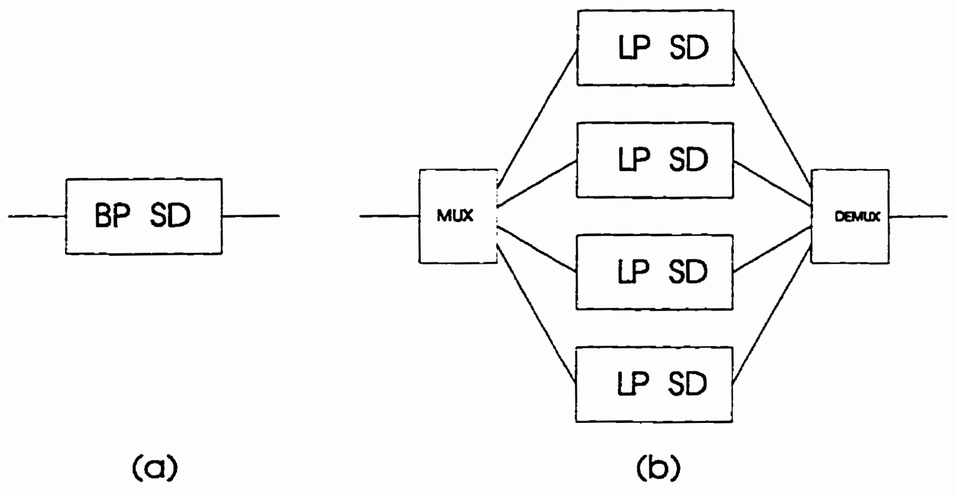


Figure 5.5: Using four lowpass Sigma-Delta modulators to implement a bandpass Sigma-Delta modulator.

each operating at half rate, while maintaining the overall transfer function and hence the signal-to-noise ratio is maintained.

$H(z)$ can be expressed as a sum of an even function and an odd one:

$$H(z) = \frac{z^{-1}}{1+z^{-1}} = \frac{-z^{-2}}{1-z^{-2}} + \frac{z^{-1}}{1-z^{-2}} \tag{5.7}$$

Notice that, $H_e(z)$ (the even function) and $H_o(z)$ (the odd one) are similar with the exception of a delay. This means that the common filtering can be done before the split or after the subtraction. The two possible solutions are shown in Figure 5.7 [144] [145]. Note that, both figures show a single channel (I or Q) bandpass Sigma-Delta modulator.

The same concept can be extended to higher order bandpass Sigma-Delta modulators. Figure 5.8 shows the extension of this analog parallelism to a single channel fourth-order bandpass Sigma-Delta modulator [145]. Without the cross-coupling

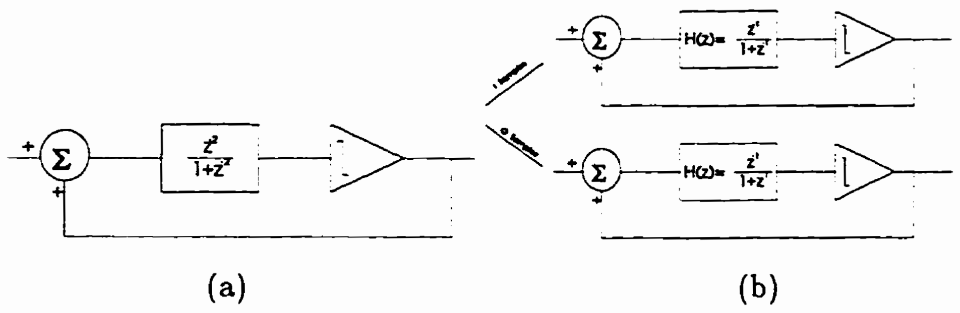


Figure 5.6: Second-order bandpass Sigma-Delta modulator (a) conventional (b) with separate IQ branches

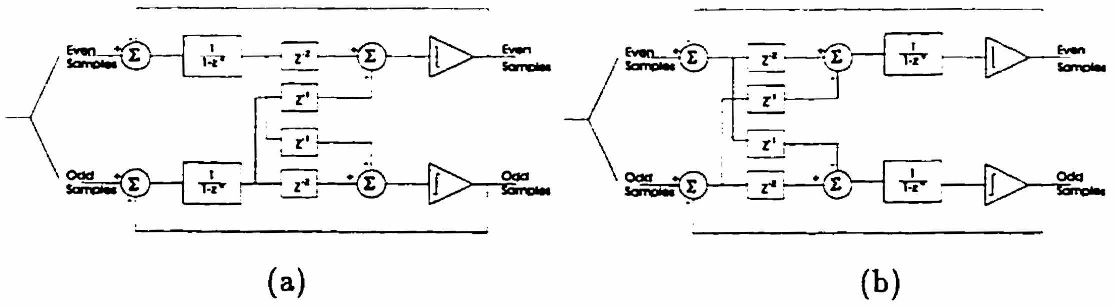


Figure 5.7: A single-channel second-order bandpass Sigma-Delta modulator with two cross-coupled branches and common filtering done: (a) before splitting (b) after subtraction.

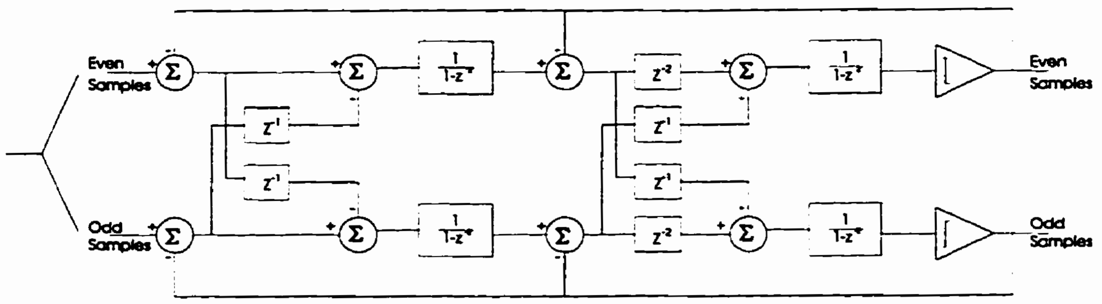


Figure 5.8: A single-channel fourth-order bandpass Sigma-Delta modulator with two cross-coupled branches.

shown in Figure 5.7 and Figure 5.8, the SNR of the Sigma-Delta modulator is degraded by $6N$ dB, where, N is the order of the equivalent lowpass Sigma-Delta modulator.

5.4 The Performance of the Parallel Sigma–Delta Modulator

The two split-branch architectures for the single-channel second-order bandpass Sigma-Delta modulator, shown in Figure 5.7, have the same linearized transfer function as the conventional single-channel second-order bandpass Sigma-Delta modulator. However, in the presence of mismatch, the response of each modulator becomes different.

The errors considered are assumed to occur in the even/odd-sample integrator block, which ideally should have a transfer function:

$$\frac{1}{1-z^{-2}} \tag{5.8}$$

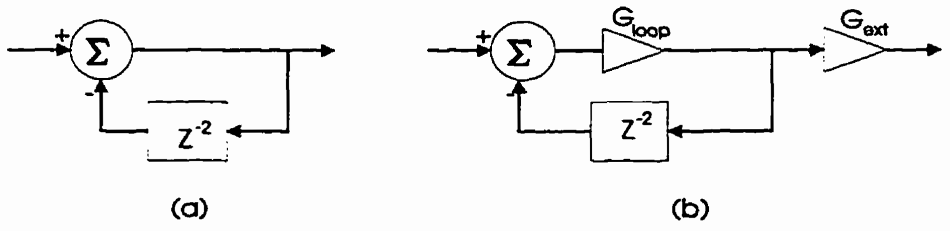


Figure 5.9: (a) An ideal integrator (b) Integrator with mismatch.

The mismatch is modeled by the elements G_{loop} and G_{ext} as shown in Figure 5.9.b. Ideally, both elements should be 1. However, practically G_{loop} can be slightly less than 1, while G_{ext} can be slightly greater than or less than one. G_{loop} is the leakage of the integrator, caused by the finite gain, A, of the operational amplifier in the integrator [130]:

$$G_{loop} = 1 - \frac{1}{A} \tag{5.9}$$

The $G_{loop}G_{ext}$ product is the gain of the integrator.

The simulation of the proposed Sigma-Delta modulator in different configurations, along with the simulation of the conventional Sigma-Delta modulator was performed using SPW$^{\text{TM}}$. Details of the simulation model are given in appendix A. In this section, the obtained results are presented.

Notice that, G_{ext} has no effect on the performance of the Sigma-Delta modulator of Figure 5.7.b, because the gain doesn't effect the signal's polarity and hence the operation of the comparator. However, this is only true in Figure 5.7.a if G_{ext} of the even and odd branches are equal. But if there is a discrepancy between them, it would degrade the performance. Figure 5.10 shows the degradation in SNR due to discrepancy in the value of G_{ext} between the even and the odd branches of the Sigma-Delta modulator given in Figure 5.7.a. A 2% difference in G_{ext} would lead

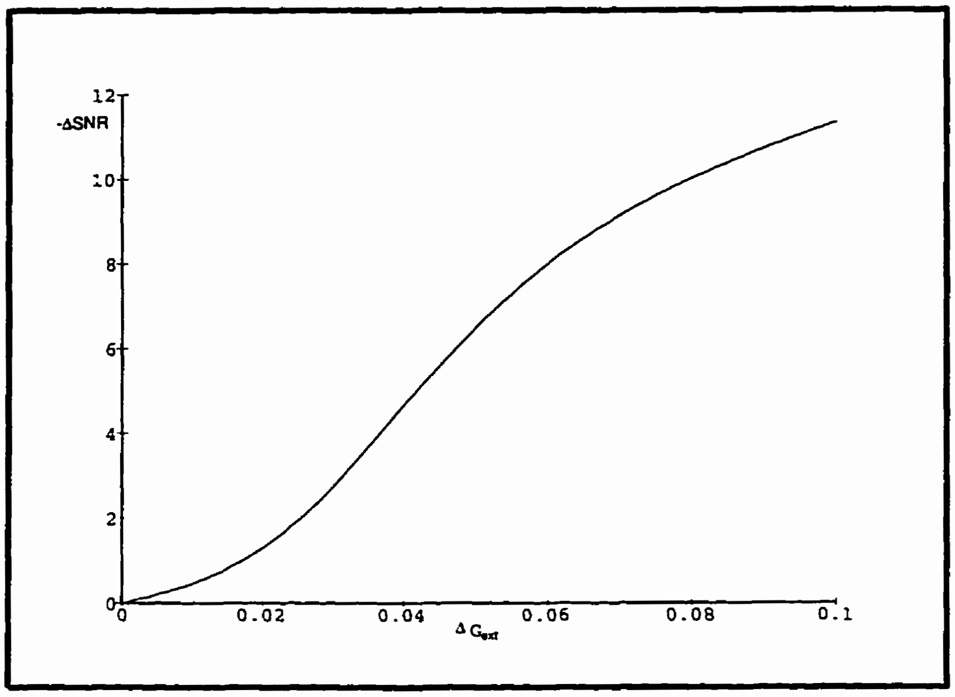


Figure 5.10: Degradation in SNR due to mismatch in the value of G_{ext} between the even and the odd branches of the Sigma-Delta modulator shown in Figure 5.7.a

to a 1.5 dB degradation in SNR.

Figure 5.11 shows the effect of non-unity in G_{loop} on the conventional Sigma-Delta modulator of Figure 5.6.

Figure 5.12 shows the effect of mismatch and non-unity in G_{loop} on the SNR performance of the bandpass Sigma-Delta modulator of Figure 5.7.b. Notice that, the SNR performance is very close to that of the conventional Sigma-Delta modulator given in Figure 5.11.

Figure 5.13 shows the effect of mismatch and non-unity in G_{loop} on the SNR performance of the bandpass Sigma-Delta modulator of Figure 5.7.a. Notice the substantial degradation in performance with mismatch.

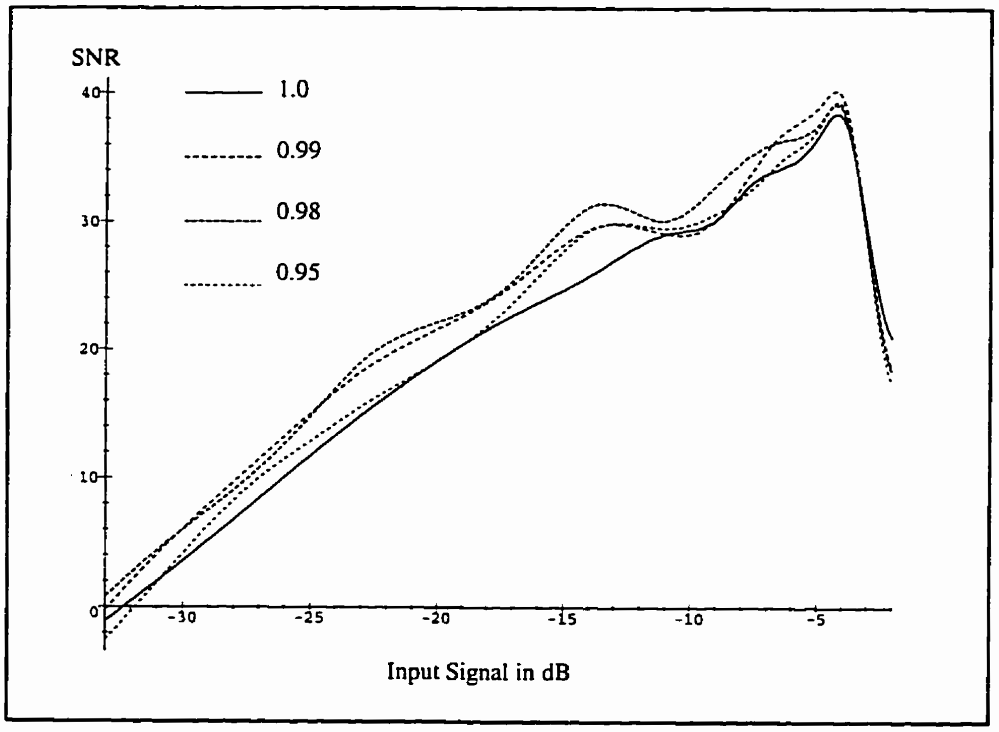


Figure 5.11: The effect of non-unity in G_{loop} on the SNR of the conventional Sigma-Delta modulator.

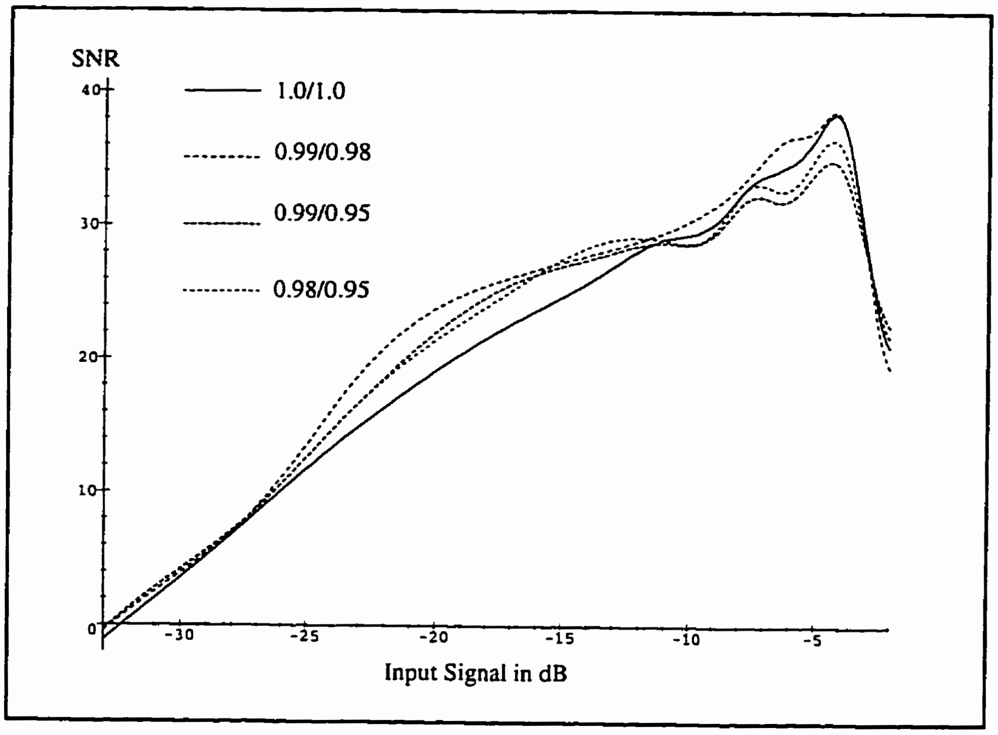


Figure 5.12: The effect of mismatch and non-unity in G_{loop} on the SNR performance of the bandpass Sigma-Delta modulator of Figure 5.7.b.

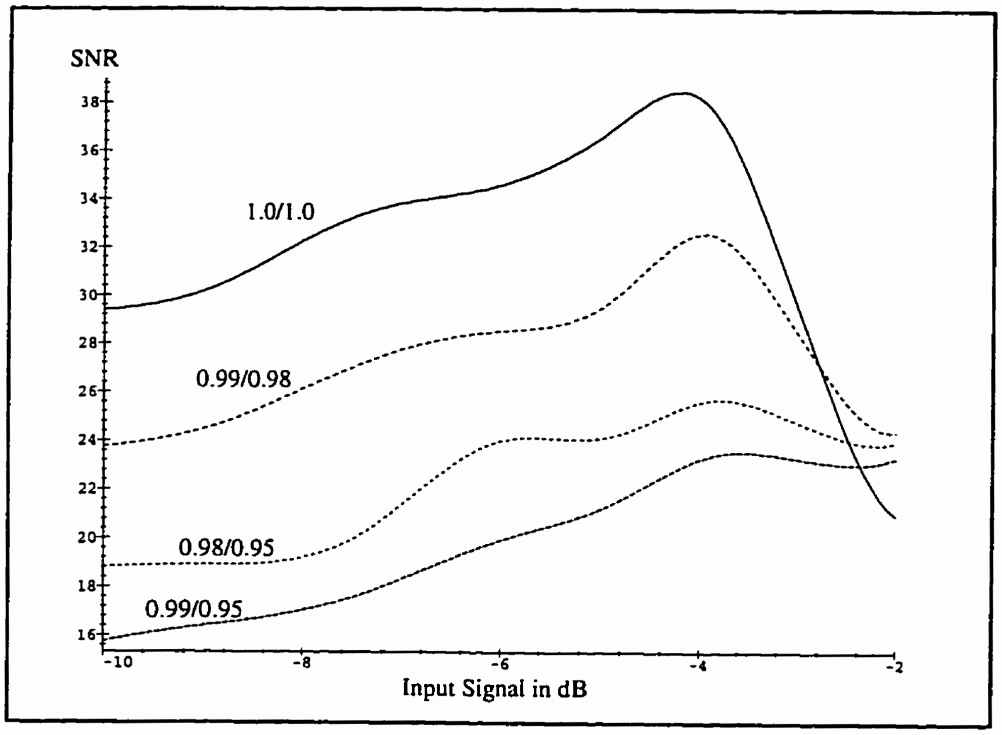


Figure 5.13: The effect of mismatch and non-unity in G_{loop} on the SNR performance of the bandpass Sigma-Delta modulator of Figure 5.7.a.

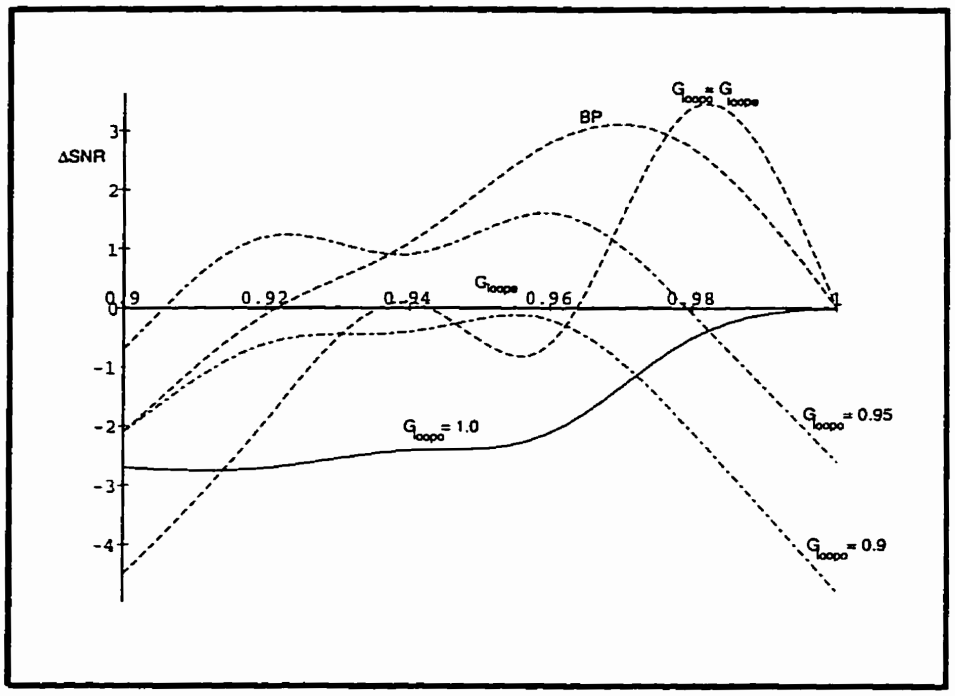


Figure 5.14: Change in SNR due to mismatch in the value of G_{loop} for the conventional single-channel second-order bandpass Sigma-Delta modulator, and for the bandpass Sigma-Delta modulator of Figure 5.7.b.

Figure 5.14 shows the effect of mismatch and non-unity in G_{loop} on the conventional bandpass Sigma-Delta modulator (Figure 5.6) and the bandpass Sigma-Delta modulator of Figure 5.7.b. A 10% degradation in G_{loop} could lead up to 4.5 dB degradation in SNR.

Discrepancy in the value of G_{loop} between the even and the odd branches of the bandpass Sigma-Delta modulator given in Figure 5.7.a can cause severe distortion to the signal as shown in Figures 5.15 and 5.16.

The results presented so far demonstrate that the Sigma-Delta modulators of Figures 5.7.a and 5.7.b, even though they have identical behaviour under ideal conditions, yet their performance is differently affected by parameter mismatch.

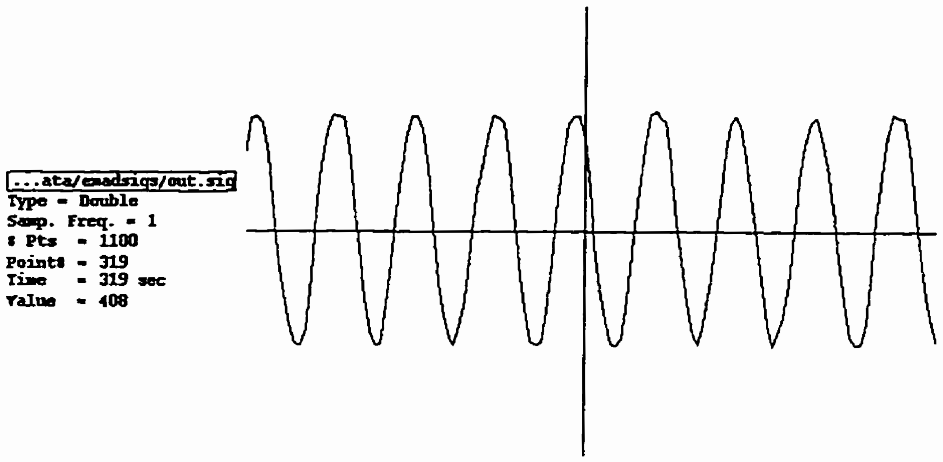


Figure 5.15: Distortion in the output signal of the bandpass Sigma-Delta modulator shown in Figure 5.7.a due to $G_{loopc} = 0.99$ and $G_{loopo} = 0.95$.

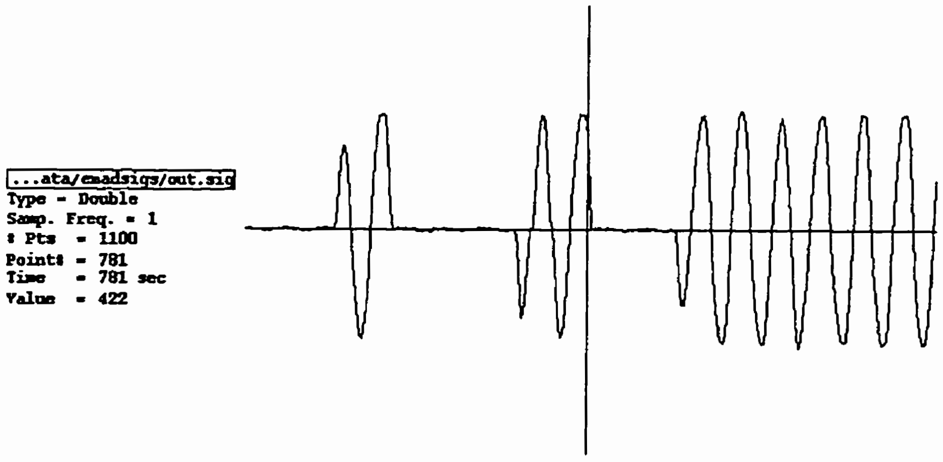
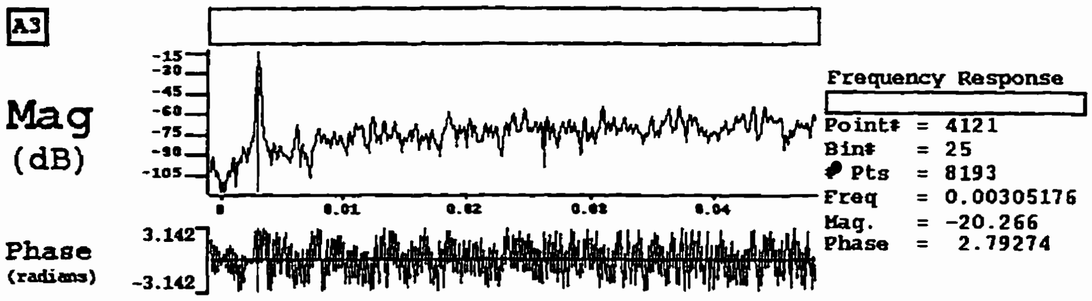


Figure 5.16: Distortion in the output signal of the bandpass Sigma-Delta modulator shown in Figure 5.7.a due to $G_{loopc} = 1.0$ and $G_{loopo} = 0.95$.



```
Point# = 4121
Bin#   = 25
# Pts  = 8193
Freq   = 0.00305176
Mag.   = -20.266
Phase  =  2.79274
```

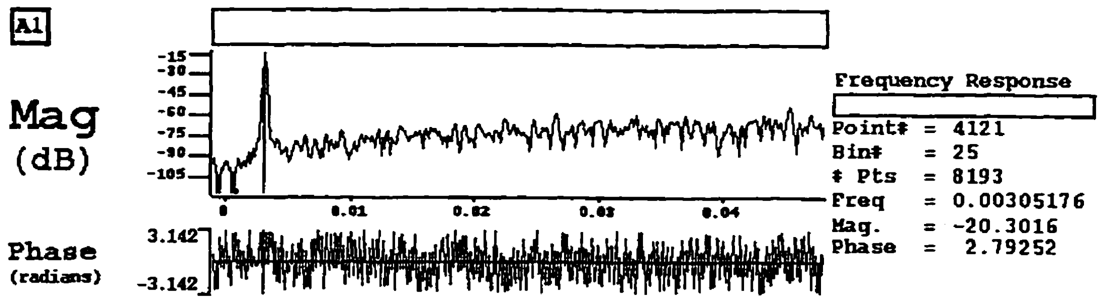
Figure 5.17: Frequency spectrum of the Sigma-Delta modulators of Figure 5.7 having $G_{loope}/G_{loopo} = 1.0/1.0$.

Mismatch can cause the architecture of Figure 5.7.a to become unstable and it causes unacceptable signal distortion. On the other hand, the effect of mismatch on the architecture given in Figure 5.7.b is quite small, and its SNR performance is comparable to that of the conventional second-order bandpass Sigma-Delta modulator.

The frequency spectrum at the output of the proposed Sigma-Delta modulators of Figure 5.7, for different values of G_{loope}/G_{loopo}, is shown in Figures 5.17 — 5.21. The injected sinusoidal signal has an amplitude of 0.5, and a frequency of 0.0031. The sampling frequency is 1.0. Notice that the mismatch increases the low-frequency quantization noise substantially for the modulator of Figure 5.7.a. While it has a negligible effect on the low-frequency quantization noise of the modulator of Figure 5.7.b.

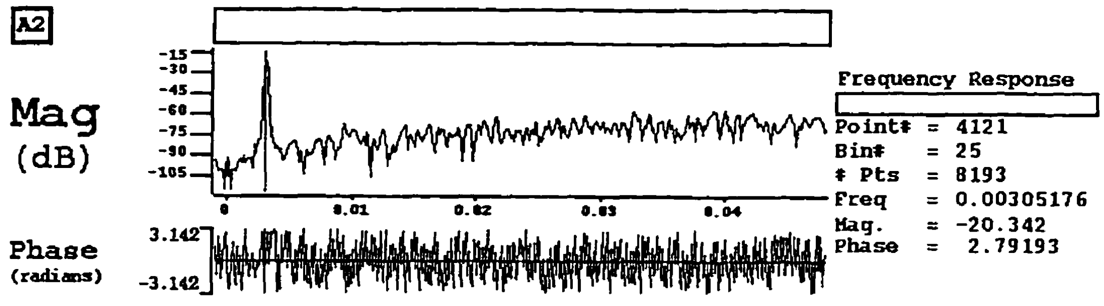
Similar analysis for the fourth-order bandpass Sigma-Delta modulator show that placing the integrator after the subtractor (as shown in Figure 5.8) significantly reduces the degradation in SNR due to mismatch.

Figure 5.22 shows the effect of mismatch and non-unity in G_{loop} of the first stage on the SNR performance of a fourth-order bandpass Sigma-Delta modulator,



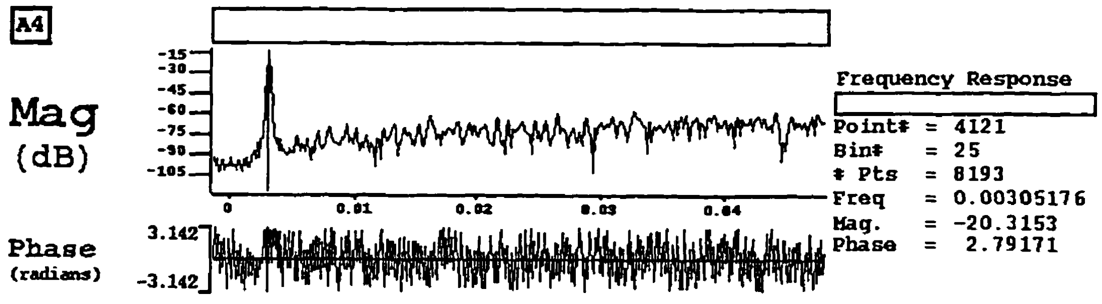
```
Point# = 4121
Bin#   = 25
# Pts  = 8193
Freq   = 0.00305176
Mag.   = -20.3016
Phase  =  2.79252
```

Figure 5.18: Frequency spectrum of the Sigma-Delta modulators of Figure 5.7 having $G_{loope}/G_{loopo} = 0.99/0.99$.



```
Point# = 4121
Bin#   = 25
# Pts  = 8193
Freq   = 0.00305176
Mag.   = -20.342
Phase  =  2.79193
```

Figure 5.19: Frequency spectrum of the Sigma-Delta modulators of Figure 5.7 having $G_{loope}/G_{loopo} = 0.98/0.98$.



```
Point# = 4121
Bin#   = 25
# Pts  = 8193
Freq   = 0.00305176
Mag.   = -20.3153
Phase  =  2.79171
```

Figure 5.20: Frequency spectrum of the Sigma-Delta modulator of Figure 5.7.b having $G_{loope}/G_{loopo} = 0.99/0.98$.

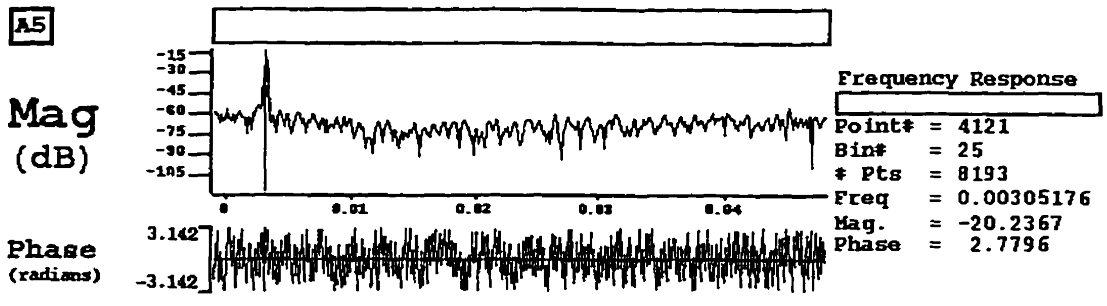


Figure 5.21: Frequency spectrum of the Sigma-Delta modulator of Figure 5.7.a having $G_{loope}/G_{loopo} = 0.99/0.98$.

having the integrator after the subtractor in the first stage. Notice the negligible degradation in SNR performance due to mismatch.

Figure 5.23 shows the effect of mismatch and non-unity in G_{loop} of the second stage on the SNR performance of a fourth-order bandpass Sigma-Delta modulator, having the integrator after the subtractor in the second stage. Notice the negligible degradation in SNR performance due to mismatch.

Figure 5.24 shows the effect of mismatch and non-unity in G_{loop} of the first stage on the SNR performance of a fourth-order bandpass Sigma-Delta modulator, having the integrator before the branch splitting in the first stage. Notice the substantial degradation in SNR performance due to mismatch.

Figure 5.25 shows the effect of mismatch and non-unity in G_{loop} of the second stage on the SNR performance of a fourth-order bandpass Sigma-Delta modulator, having the integrator before the branch splitting in the second stage. Notice the slight degradation in SNR performance due to mismatch.

Figures 5.22 — 5.25 indicate that, if the gain of the operational amplifier of the integrator is 37 dB with a ± 3 dB mismatch, the degradation in the SNR with the integrator after the subtractor (Figure 5.8) is about 1 dB, while the degradation in

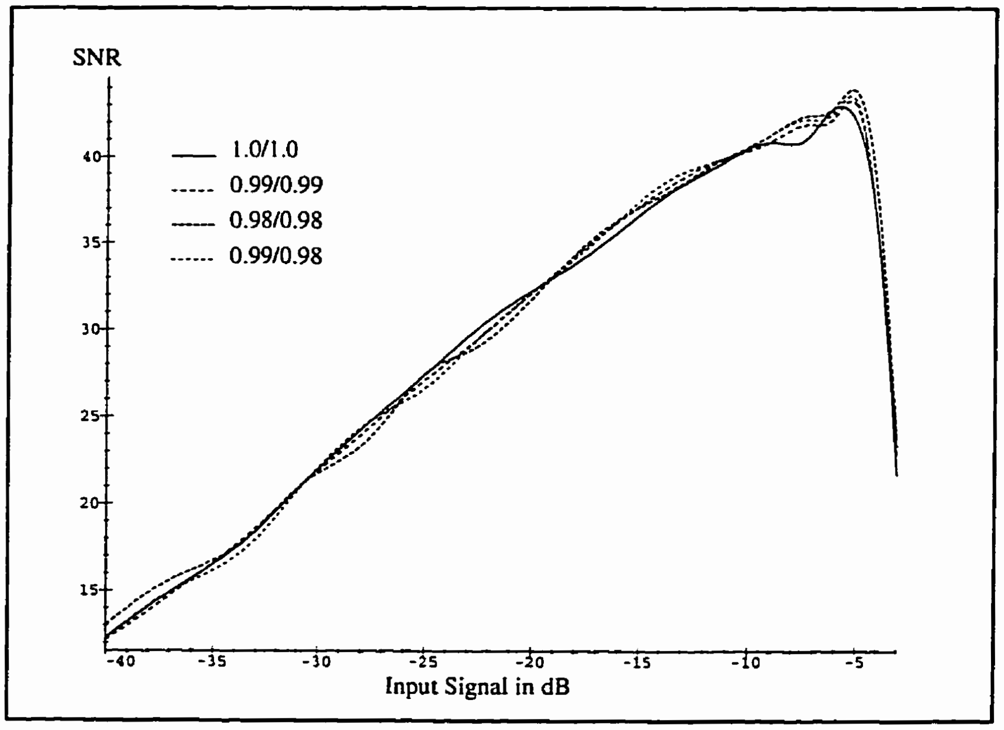


Figure 5.22: The effect of mismatch and non-unity in G_{loop} of the first stage on the SNR performance of a single-channel fourth-order bandpass Sigma-Delta modulator, having the integrator after the subtractor in the first stage.

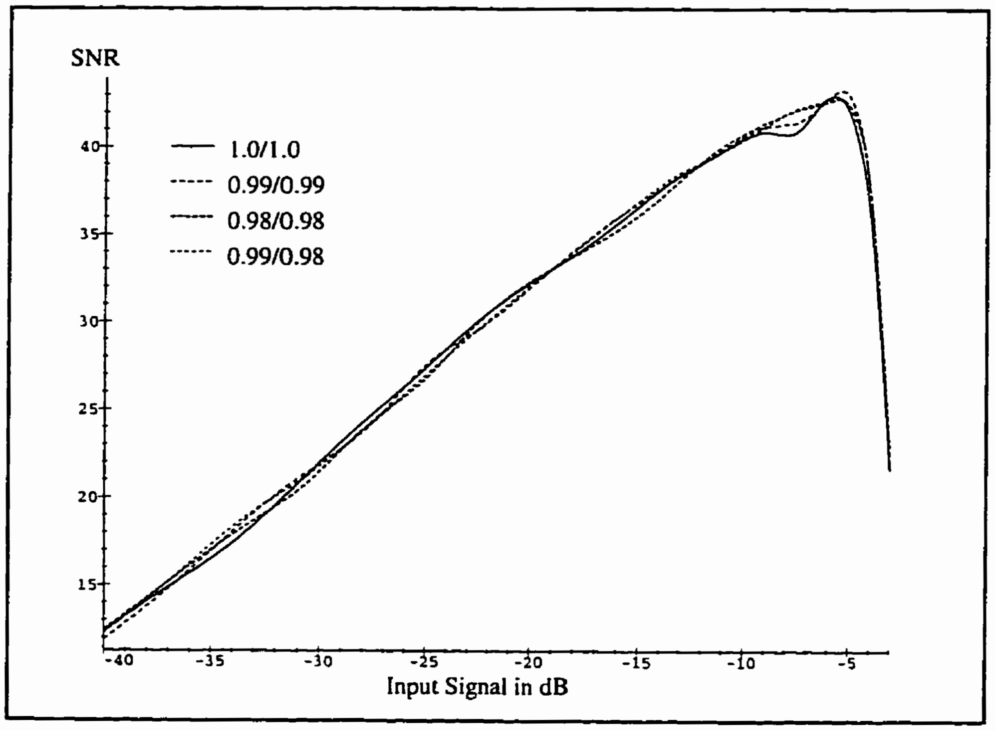


Figure 5.23: The effect of mismatch and non-unity in G_{loop} of the second stage on the SNR performance of a single-channel fourth-order bandpass Sigma-Delta modulator, having the integrator after the subtractor in the second stage.

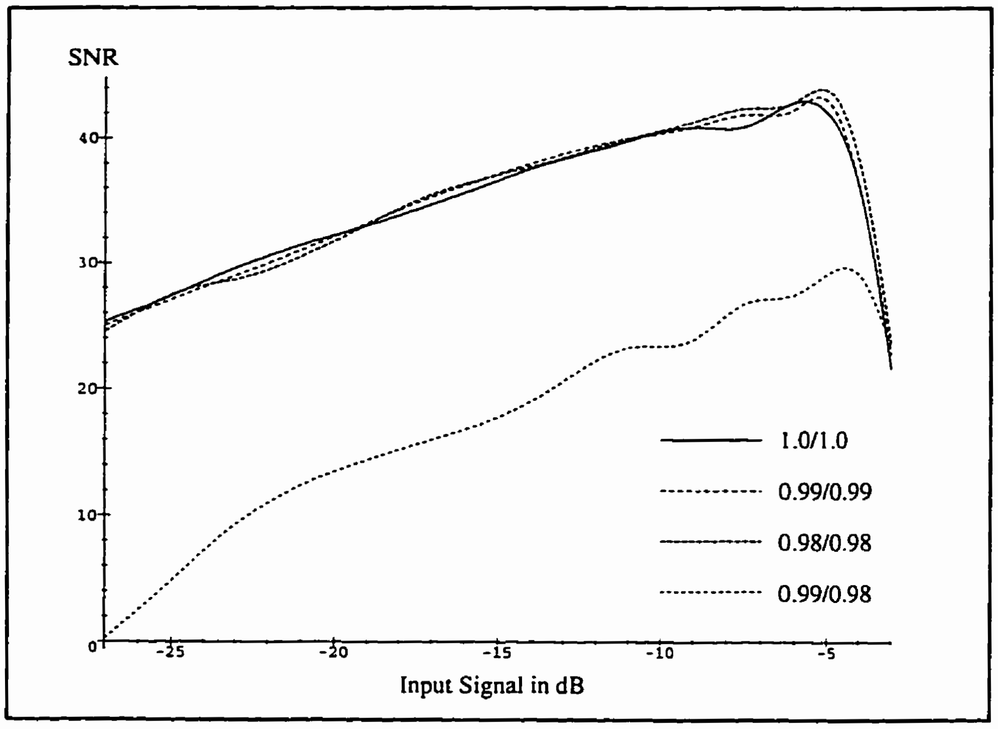


Figure 5.24: The effect of mismatch and non-unity in G_{loop} of the first stage on the SNR performance of a single-channel fourth-order bandpass Sigma-Delta modulator, having the integrator before the branch splitting in the first stage.

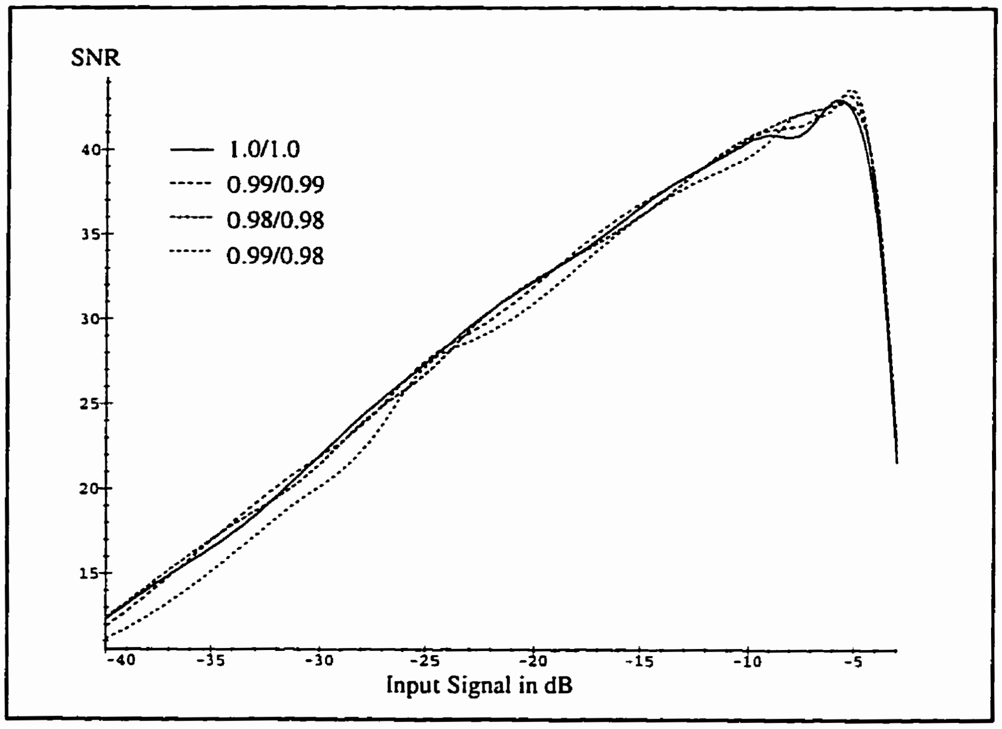


Figure 5.25: The effect of mismatch and non-unity in G_{loop} of the second stage on the SNR performance of a single-channel fourth-order bandpass Sigma-Delta modulator, having the integrator before the branch splitting in the second stage.

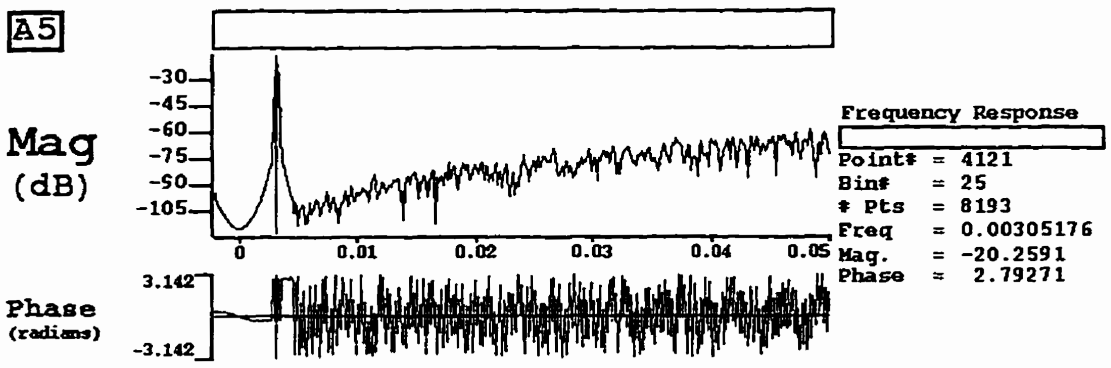


Figure 5.26: Frequency spectrum of a single-channel fourth-order bandpass Sigma-Delta modulator, having $G_{loope}/G_{loopo} = 1.0/1.0$ for the first and second stages.

the SNR with the integrator before the branch splitting is about 25 dB.

The frequency spectrum at the output of the proposed Sigma-Delta modulator of Figure 5.8 and its variants for different values of G_{loope}/G_{loopo}, is shown in Figures 5.26 — 5.30. The injected sinusoidal signal has an amplitude of 0.5, and a frequency of 0.0031. The sampling frequency is 1.0. Notice that the mismatch increases the low frequency quantization noise substantially for the architectures having the integrator placed before the branch splitting. Notice also that the mismatch in G_{loop} of the first stage causes a greater increase in the noise than the mismatch in G_{loop} of the second stage. Mismatch has a negligible effect on the low-frequency quantization noise of the architectures having the intergrator placed after the subtractor.

Another advantage of placing the integrator after the subtractor is that it is possible, in a switched-capacitor implementation, to use the same operational amplifier for integration and addition (subtraction), thus reducing the number of operational amplifiers required to implement the modulators. The switched capacitor implementation is explained in the next section.

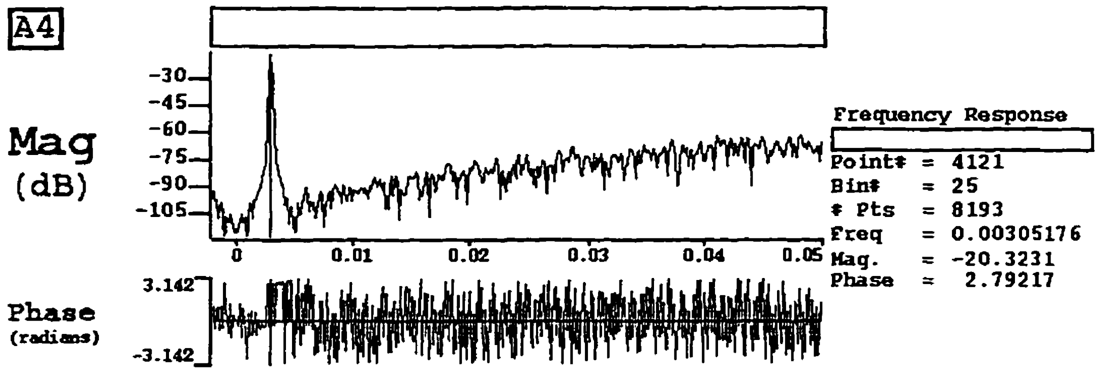


Figure 5.27: Frequency spectrum of a single-channel fourth-order bandpass Sigma-Delta modulator, having the integrated placed after the subtractor in the first stage, and having $G_{loope}/G_{loopo} = 0.99/0.98$ for the first stage, and $G_{loope}/G_{loopo} = 1.0/1.0$ for the second stage.

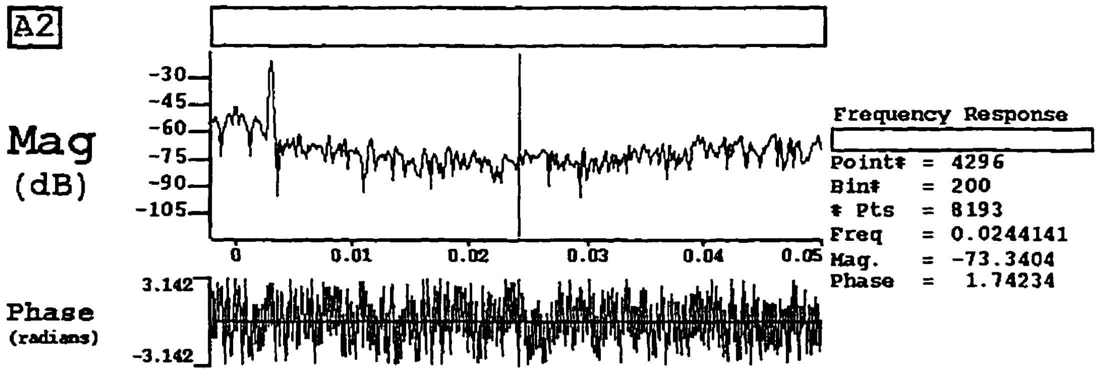
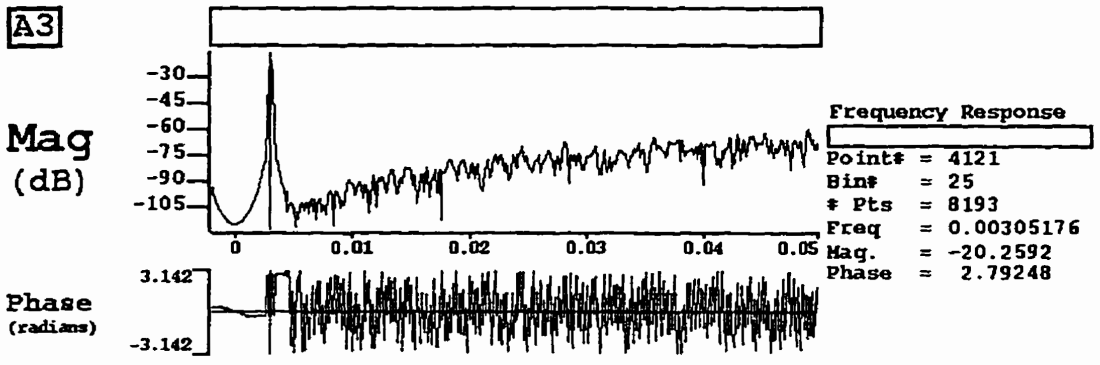
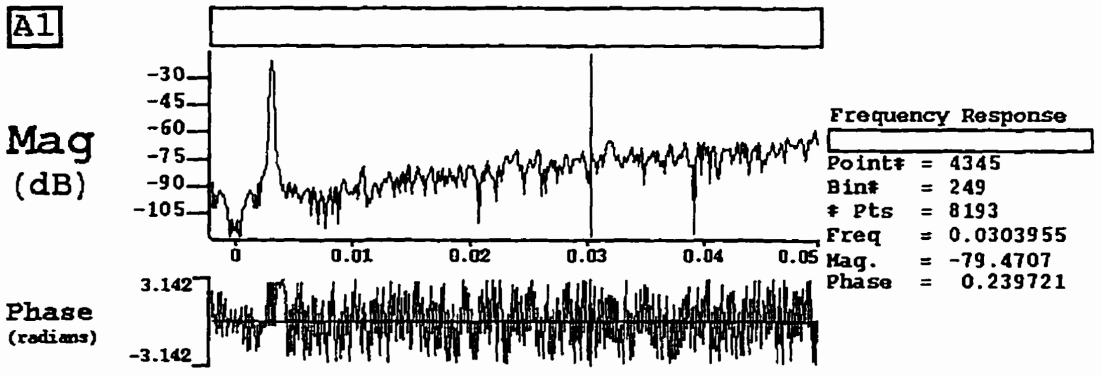


Figure 5.28: Frequency spectrum of a single-channel fourth-order bandpass Sigma-Delta modulator, having the integrated placed before the branch splitting in the first stage, and having $G_{loope}/G_{loopo} = 0.99/0.98$ for the first stage, and $G_{loope}/G_{loopo} = 1.0/1.0$ for the second stage.



```
Point# = 4121
Bin#   = 25
# Pts  = 8193
Freq   = 0.00305176
Mag.   = -20.2592
Phase  =  2.79248
```

Figure 5.29: Frequency spectrum of a single-channel fourth-order bandpass Sigma-Delta modulator, having the integrated placed after the subtractor in the second stage, and having $G_{loope}/G_{loopo} = 1.0/1.0$ for the second stage, and $G_{loope}/G_{loopo} = 0.99/0.98$ for the second stage.



```
Point# = 4345
Bin#   = 249
# Pts  = 8193
Freq   = 0.0303955
Mag.   = -79.4707
Phase  =  0.239721
```

Figure 5.30: Frequency spectrum of a single-channel fourth-order bandpass Sigma-Delta modulator, having the integrated placed before the branch splitting in the second stage, and having $G_{loope}/G_{loopo} = 1.0/1.0$ for the first stage, and $G_{loope}/G_{loopo} = 0.99/0.98$ for the second stage.

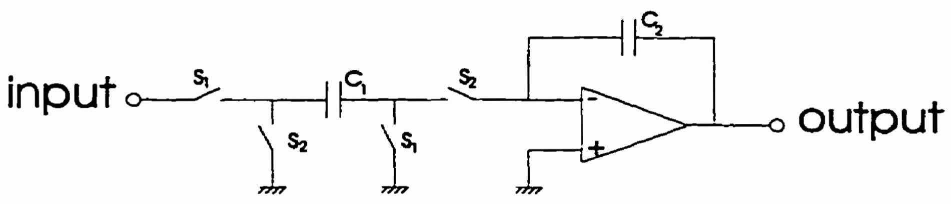


Figure 5.31: Switched-Capacitor Integrator.

5.5 Switched-Capacitor Architecture

The switched-capacitor implementation for the Sigma-Delta modulators of Figures 5.7.b and 5.8 is developed in this section [145]. The switched-capacitor integrator used in these implementations is given in Figure 5.31 [142]. Notice that this architecture introduces a half cycle delay between the input and the output.

In this section the word cycle refers to the duration between two consecutive even samples (or odd samples). In this case, a delay of one cycle is z^{-2} in the z-domain, and a delay of half a cycle is z^{-1}. The even and odd samples at the input as well as those at the output of the comparator are held for an entire cycle. The even and odd samples are staggered by half a clock cycle. In the switched capacitor implementations given in this section the delays are implemented by proper timing of the switches.

First, consider the implementation of the single-channel second-order bandpass Sigma-Delta modulator given in Figure 5.7.b. Since the integrator introduces a delay of half a cycle (z^{-1}), this delay should be included with each integrator. Also, the four adders are combined into two adders. Figure 5.32 shows the modified modulator.

The modified architecture of Figure 5.32 requires only two operational amplifiers for its implementation. The delays are implemented by proper timing of the

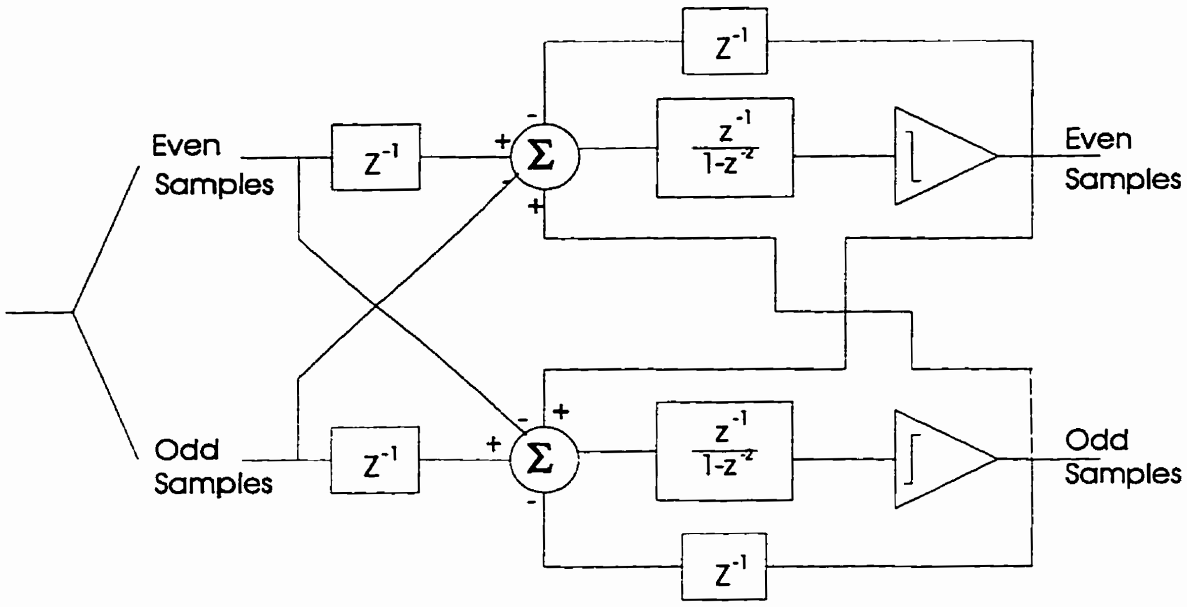


Figure 5.32: A modified single-channel second-order bandpass Sigma-Delta modulator with two cross-coupled branches.

switches. Figure 5.33 shows the switched-capacitor implementation for the single-channel second-order bandpass Sigma-Delta modulator.

For the single-channel fourth-order bandpass Sigma-Delta the modulator of Figure 5.8 is modified to include a delay (z^{-1}) with each integrator. However, the modified architecture contains non-causal blocks as shown in Figure 5.34.

It is possible to manipulate the blocks around to retain the causality of each block. The new modified architecture is shown in Figure 5.35. This modified architecture requires four operational amplifiers for its implementation. The delays are obtained by proper timing of the switches. Figure 5.36 shows the switched-capacitor implementation for the single-channel fourth-order bandpass Sigma-Delta modulator.

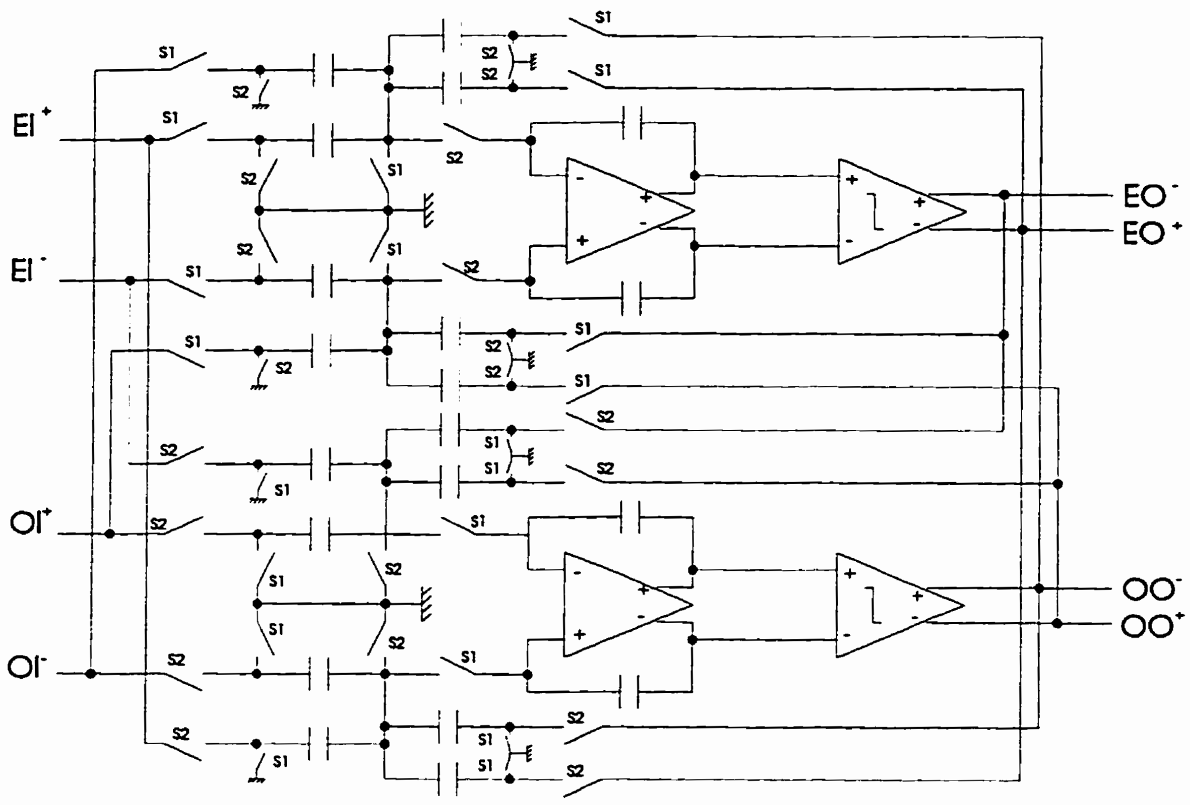


Figure 5.33: The switched capacitor implementation of the single-channel second-order bandpass Sigma-Delta modulator with two cross-coupled branches.

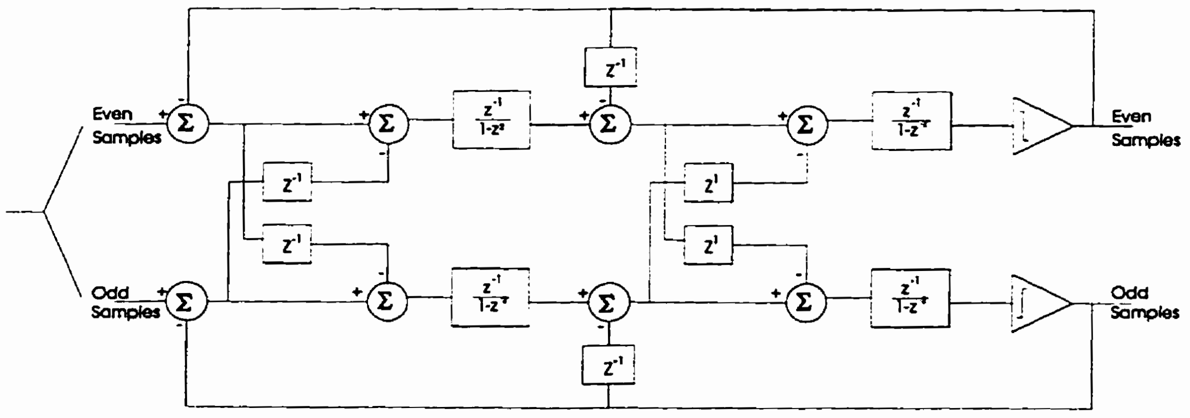


Figure 5.34: A modified single-channel fourth-order bandpass Sigma-Delta modulator with two cross-coupled branches, having non-causal blocks.

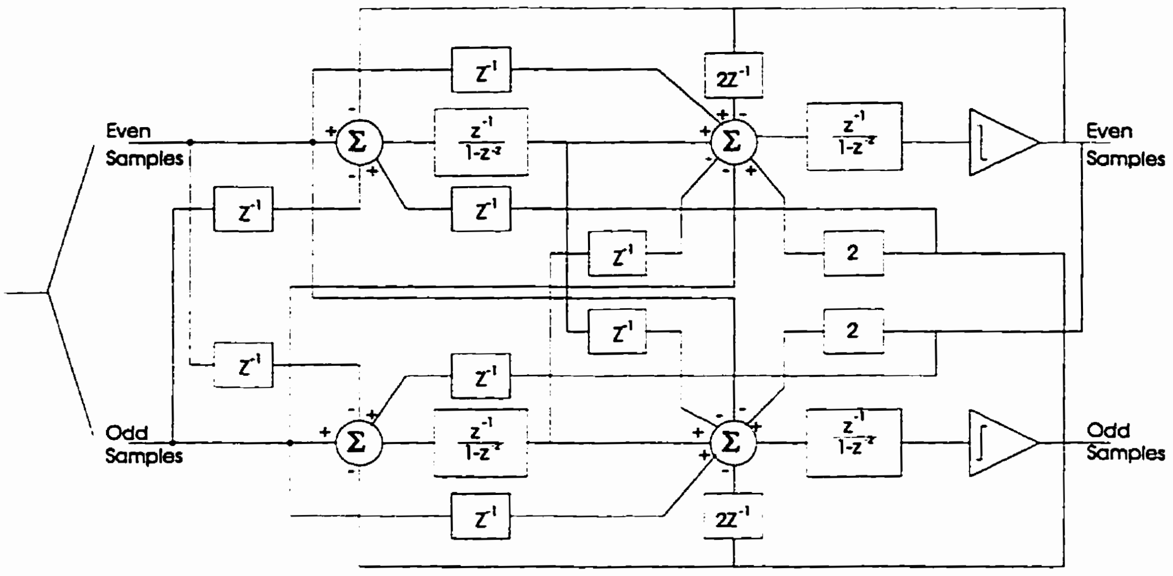


Figure 5.35: A modified single-channel fourth-order bandpass Sigma-Delta modulator with two cross-coupled branches, having no non-causal blocks.

5.6 The Decimation Filter Architecture

The decimation filter (Figure 5.37) consists of two parts; the Sinc decimator and the lowpass decimation filter (LPDF). The Sinc decimator is characterized by its simple structure, requiring only addition operations which makes it a power-efficient structure. The order of the Sinc decimator used depends on the order of the Sigma-Delta modulator, and is given by [135]:

$$\text{Order of Sinc} = \text{Order of LPSD} + 1 \tag{5.10}$$

For a Sinc decimator with order N and a decimation factor of M, the transfer function (before down-sampling is given by):

$$H(e^{jw}) = (1 + e^{-jw} + e^{-2jw} + \ldots + e^{-(M-1)jw})^N \tag{5.11}$$

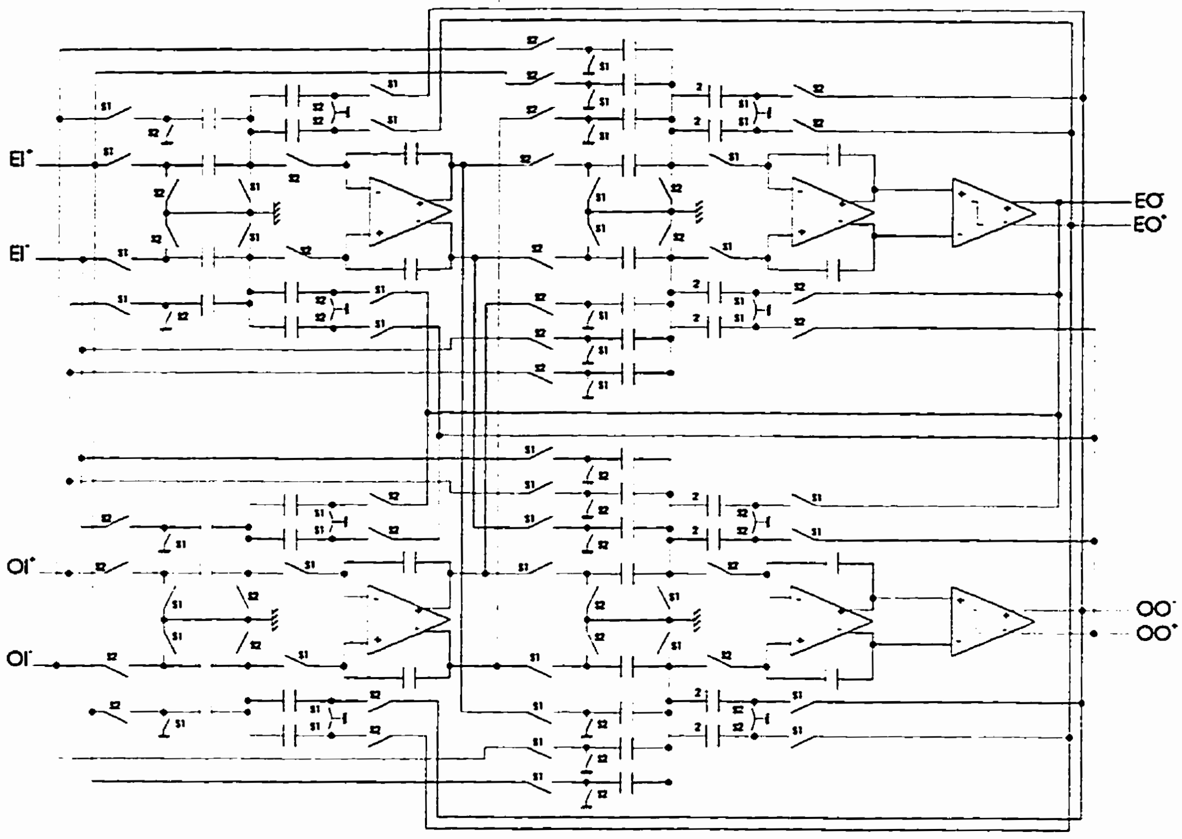


Figure 5.36: The switched capacitor implementation of the single-channel fourth-order bandpass Sigma-Delta modulator with two cross-coupled branches.

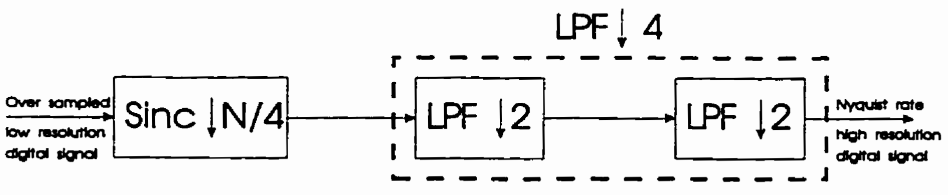

LPF ↓ 4
Over sampled low resolution digital signal
Sinc ↓N/4
LPF ↓2
LPF ↓2
Nyquist rate high resolution digital signal

Figure 5.37: The decimation filter.

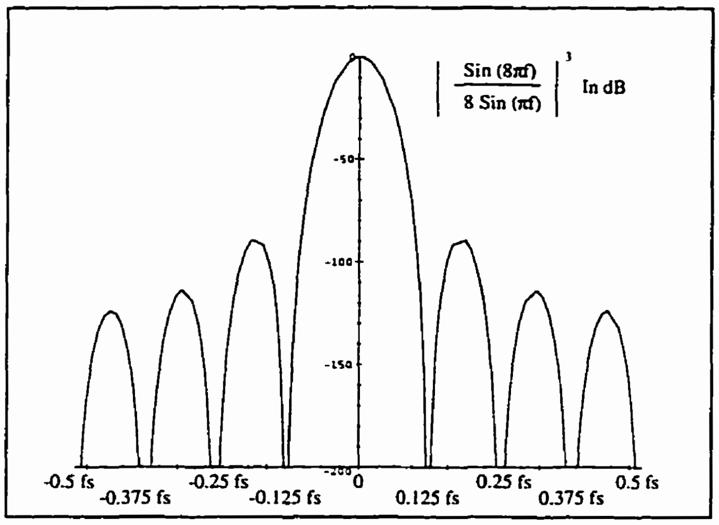


Figure 5.38: The transfer function of a Sinc decimator having, $M = 8$, and $N = 3$.

Figure 5.38 shows the transfer function for a Sinc decimator having $M = 8$, and $N = 3$. Notice the zeros of the Sinc decimator at frequencies: $\pm f_s/8$, $\pm f_s/4$, $\pm 3f_s/8$, and $\pm f_s/2$. When the frequency spectrum is folded three times, around $f_s/4$, $f_s/8$, and $f_s/16$, the zeros of the folded spectrum fall on the spectrum at $f_s = 0$. This minimizes the out-of-band noise added to the low frequency spectrum due to folding.

Due to the variable-resolution requirement of the A/D converter, the order of the Sinc, N, and the decimation factor, M, can vary. N can be 3 or 4 for a fourth-order or a sixth-order BPSD respectively. M can be 8, 16 or 32.

Due to its gradual transition from the passband to the stopband, the Sinc decimator can't be used in the entire decimation process. The last stage of decimation is done using an LPDF, which does decimation by a factor of four [133]. The LPDF is built as a two stage LPDF each doing decimation by a factor of two. Due to the variable-resolution requirement of the A/D converter, this filter is designed to operate with variable resolution, and hence reduce power dissipation when operating at the lower resolution [146] [147], by eliminating irrelevant computations. A novel memory access algorithm is employed in the LPDF. An interleaved multiplier-accumulator array is also used in the LPDF [148].

5.7 Power Efficient Sinc Decimator Architecture

Several architectures were considered for the implementation of the Sinc decimator [149]. In this section, the power dissipation of four architectures that implement an n th order Sinc decimator that does decimation by a factor of 2^m are compared. The metric for the power dissipation comparison is the number of operations required to generate a single output. The most power-efficient Sinc decimator is the one that requires the least number of computations to generate a single output, and thus eliminates redundant computations.

5.7.1 First Architecture

In the first implementation, the Sinc decimator is divided into m stages. Each stage is an n th order Sinc decimator that does decimation by a factor of 2. Figure 5.39 shows the implementation of such a decimator. Assuming that k is the resolution of the input to the Sinc decimator, the resolution at the output of the Sinc decimator is $k + mn$ bits. Notice that in this case, the resolution after each Sinc stage increases by n bits. Also notice that each Sinc stage operates at double the speed of the following Sinc stage because of down-sampling by 2.

Assuming that $n = 3$. Each Sinc stage in Figure 5.39, having i bits at its input,

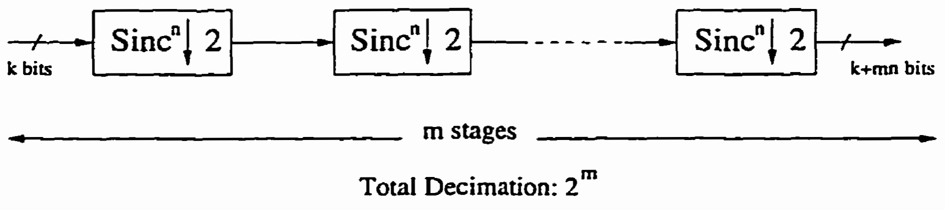


Figure 5.39: First architecture of a Sinc decimator.

requires a total of $4i + 2$ additions to generate a single output. Hence, the total number of one-bit additions per output for a third order Sinc is:

$$\text{One-bit additions per output} = (4k + 14)(2^m - 1) - 12m \qquad (5.12)$$

Assuming that $n = 4$. Each Sinc stage in Figure 5.39, having i bits at its input, requires a total of $5i + 4$ additions to generate a single output. Hence, the total number of one-bit additions per output for a fourth order Sinc is:

$$\text{One-bit additions per output} = (5k + 24)(2^m - 1) - 20m \qquad (5.13)$$

5.7.2 Second Architecture

In this implementation, the Sinc decimator is divided into n stages. Each stage is a Sinc decimator of the first order and having 2^m taps. The decimation by a factor 2^m is done after the last stage. Figure 5.40.a shows a block diagram of this implementation.

A Sinc decimator of the first order and having 2^m taps is simply the moving sum (or average) of the last 2^m samples. The new output can be obtained from the previous output by adding sample $x(k)$ and subtracting sample $x(k - 2^m)$. This is

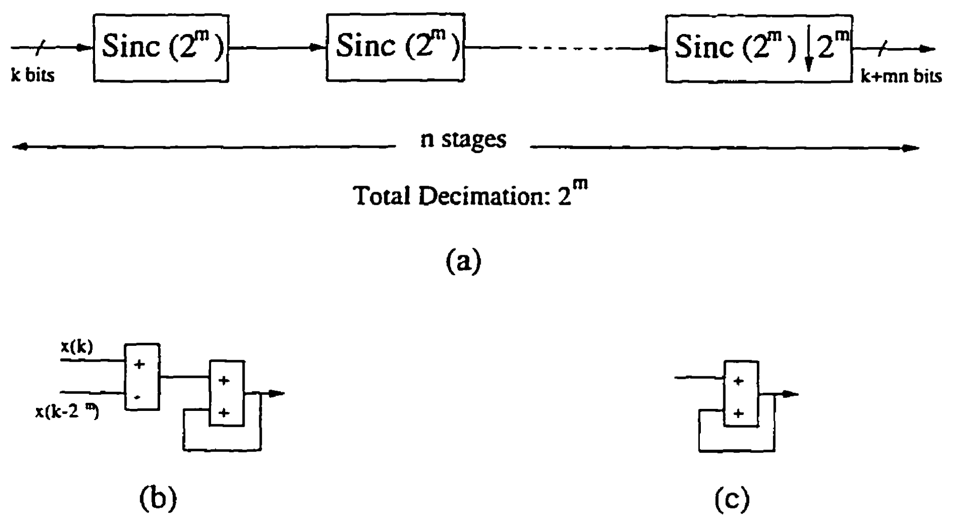


Figure 5.40: Second architecture of a Sinc decimator. (a) Block diagram. (b) First $n-1$ Sinc stages. (c) Last Sinc stage.

shown in Figure 5.40.b. Since the last Sinc stage is followed by a down-sampler, the last stage is simply an accumulate and dump. It accumulates 2^m samples than the output is cleared and the accumulation starts again. This is shown in Figure 5.40.c.

The resolution at the input is k bits, the resolution after each stage increases by m bits. Hence, the resolution at the output is $k+mn$ bits. The number of one-bit additions per output for this architecture is given by:

$$\text{One-bit additions per output} = [(2n-1)k + m((n-1)^2+n)]2^m \qquad (5.14)$$

5.7.3 Third Architecture

The transfer function of the Sinc decimator $[\text{Sinc}^n(2^m)]$ can be expressed as:

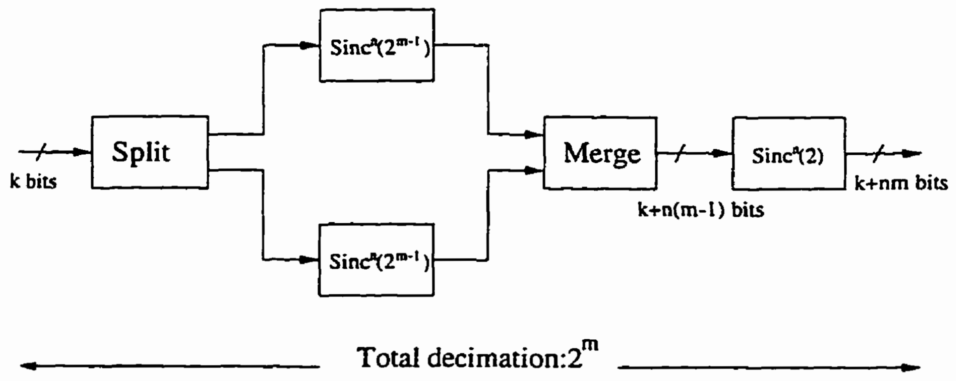


Figure 5.41: Third architecture of a Sinc decimator.

$$H(z) = \frac{1}{M^n}\left(\frac{1-z^{-M}}{1-z^{-1}}\right)^n = \frac{1}{(M/2)^n}\left(\frac{1-z^{-M}}{1-z^{-2}}\right)^n \frac{1}{2^n}\left(\frac{1-z^{-2}}{1-z^{-1}}\right)^n \tag{5.15}$$

Where, $M = 2^m$. The even and old samples are split into two branches. Each branch is filtered using a sinc$^n(2^{m-1})$ filter, the filtered signals are then merged and filtered using a sinc$^n(2)$ filter. The block diagram of this implementation is shown in Figure 5.41.

The decimation is distributed throughout the blocks of Figure 5.41. To show how this can be achieved, considered the implementation of the Sinc decimator [Sinc^3(16)]. The implementation of this filter is shown in Figure 5.42.a. The output rate is 1/16 the input rate. The filter Sinc^3(2) is a four tap filter. It requires four inputs (two from the even branch and two from the odd branch) to generate a single output. To avoid unnecessary computations, the filters Sinc^3(8) of the even and odd branches should generate two outputs for every 16 inputs. This is why there are two outputs for each filter.

Figure 5.42.b shows a computationally efficient method to generate the two outputs of Sinc^3(8) filter. Each output is at 1/8 the input rate. The number of

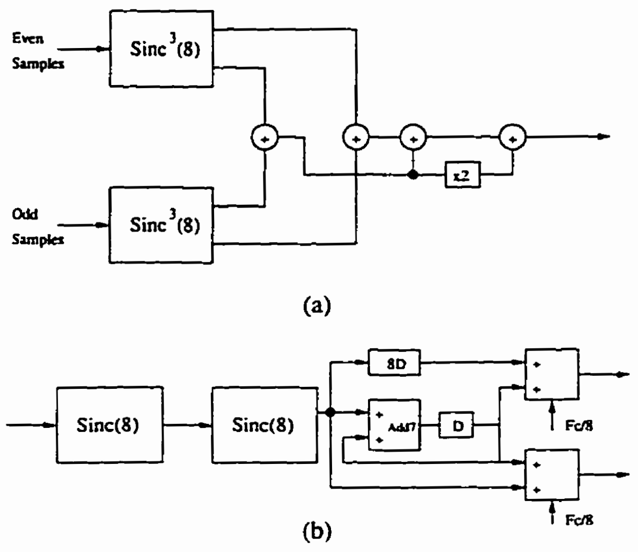


Figure 5.42: (a) Block diagram of Sinc^3(16) filter. (b) Block diagram of Sinc^3(8) filter generating two outputs every 8 inputs.

one-bit additions required per output, for a third-order Sinc decimator, is given by:

$$\text{One-bit additions per output} = (5k + 7(m-1))2^m + 7k + 21m - 14 \qquad (5.16)$$

For a fourth-order Sinc decimator, the number of one-bit additions required per output is given by:

$$\text{One-bit additions per output} = (7k + 13(m-1))2^m + 9k + 36m - 32 \qquad (5.17)$$

5.7.4 Fourth Architecture

The transfer function for the Sinc decimator [Sinc^n(2^m)] is given by:

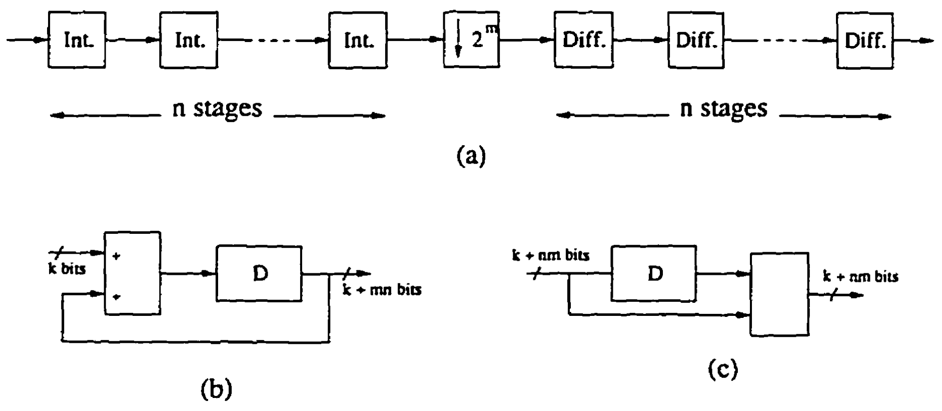


Figure 5.43: Fourth architecture of a Sinc decimator. (a) Block diagram. (b) Integrator stage. (c) Differentiator stage.

$$H(z) = \frac{1}{M^n}\left(\frac{1-z^{-M}}{1-z^{-1}}\right)^n \tag{5.18}$$

Where, $M = 2^m$. According to Equation 5.18, the Sinc decimator can be implemented as a cascade of n integrators followed by a 2^m down-sampler and then followed by n differentiators [133]. The block diagram of such an architecture is given in Figure 5.43.a. Figure 5.43.b gives the implementation of the integrator stage. While Figure 5.43.c gives the implementation of the differentiator stage. To prevent overflow, the datapath width of the integrators and the differentiators has to be:

$$k + mn \text{ bits} \tag{5.19}$$

The number of one-bit addition operations required per output is given by:

$$\text{One-bit additions per output} = (2^m + 1)(k + nm)n \tag{5.20}$$

Table 5.2: Number of additions per output for the Sinc decimators of Figures 5.39 — 5.43

Sinc Filter	Order	Number of additions per output
Figure 5.39	3	$(4k+14)(2^m-1)-12m$
Figure 5.39	4	$(5k+24)(2^m-1)-20m$
Figure 5.40	n	$[(2n-1)k+m((n-1)^2+n)]2^m$
Figure 5.41	3	$(5k+7(m-1))2^m+7k+21m-14$
Figure 5.41	4	$(7k+13(m-1))2^m+9k+36m-32$
Figure 5.43	n	$(2^m+1)(k+nm)n$

This architecture has the advantage that the down-sampling need not be a power of two, it can be any integer. In Figure 5.43, the down-sampling was chosen a power of two for the sake of comparison with the other Sinc architectures.

5.7.5 Comparison of the Sinc Decimator Architectures

Table 5.2 gives the number of one-bit additions required to generate a single output for each Sinc architecture. Figures 5.44 and 5.45 show the number of one-bit addition operations per Sinc output, for each of the four Sinc architectures, for a third-order and a fourth-order Sinc respectively, with $k = 1$.

From Figure 5.44 and 5.45, it can be seen that the first architecture requires 3 to 5 times less one-bit addition operations than the other three architectures. Architecture 3, in which the even and odd samples are filtered separately and then merged together and filtered again, is more computationally efficient, especially for higher decimation factors, than a decimator which filters all the samples together

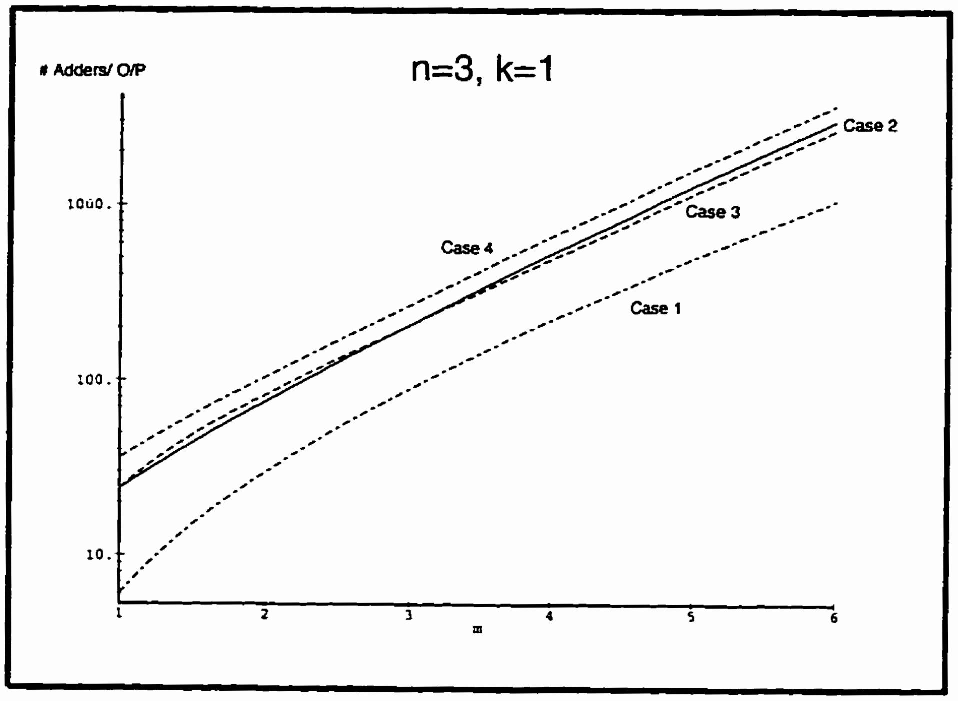


Figure 5.44: The number of one-bit addition operations required to generate a single output for the different implementations of a third-order Sinc decimator. 2^m is the decimation factor of the Sinc decimator.

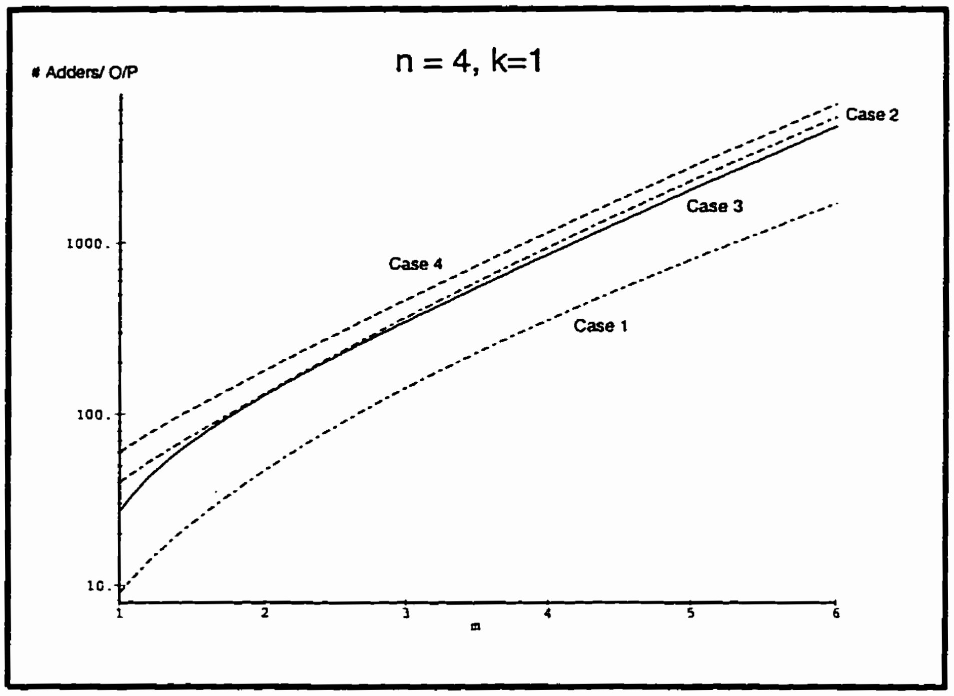


Figure 5.45: The number of one-bit addition operations required to generate a single output for the different implementations of a fourth-order Sinc decimator. 2^m is the decimation factor of the Sinc decimator.

(architecture 2), this is because it removes some of the redundant computations.

In terms of the decimation factor programmability, the fourth architecture is the most easily programmable. However, this architecture is the least computationally efficient. Notice that, the integrators of this architecture operate at a high-rate and at a high-resolution. For architecture 1, the decimation factor must be a power of 2, however, this architecture is the most computationally efficient, hence it was the implementation used in realizing the Sinc decimator.

5.8 Sinc Decimator Numerical Accuracy

Optimizing the datapath width without degrading the output numerical accuracy plays a central role in achieving a power-efficient architecture. In the previous section, four possible architectures for a Sinc decimator [$\text{Sinc}^n(2^m)$] were considered. In this section, the effect of reducing the datapath width of the internal operators, and the corresponding reduction in computational complexity, on the numerical accuracy (signal-to-noise ratio) at the output of the Sinc decimator is considered [149].

The architectures considered in this analysis are the first and the fourth Sinc architectures given in the previous section. The first architecture was found to be the most computation efficient architecture, while the fourth architecture is the most flexible in terms of programmability. The Sinc decimator considered in this analysis has the following parameters:

$$k = 1$$

$$m = 3$$

$$n = 3$$

The resolution at the output should be 10 bits. However, this resolution is more than what is sufficient. A third-order Sinc decimator is used to decimate the oversampled output of a second-order (lowpass equivalent) Sigma-Delta modulator. Decimating the oversampled signal by a factor of 8 ($m = 3$) achieves an SNR of 32 dB. This is equivalent to 5-6 bits resolution. The output resolution can be lowered from 10 bits with little impact on the output numerical accuracy. Thus eliminating any irrelevant computations.

Figure 5.46 shows the spectrum at the output of a third-order Sinc decimator based on the first architecture given in Figure 5.39. The Sinc decimator is connected to the output of a second-order lowpass Sigma-Delta modulator. The injected sine wave into the Sigma-Delta modulator has an amplitude of 0.5 and a frequency of 0.0031 relative to the sampling frequency at input of Sigma-Delta modulator. The output sampling frequency of the Sinc decimator $F_s' = \frac{f_s}{8}$. The SNR at the output of the Sinc decimator, where is the noise is limited to the frequency band $[-\frac{f_s'}{4}, \frac{f_s'}{4}]$ is given by 40.2 dB. Theoretically, this SNR should have been 41 dB.

The output of the Sinc decimator can have any value between -1.00000000_2 to 1.00000000_2. This requires 1 integer bit, 1 sign bit and 8 fraction bits. The integer bit is required only to represent 1.00000000_2. If we can eliminate this value by approximating it to 0.11111111_2, we will require only 9 bits, 1 sign and 8 fraction bits.

The question now is where do we do this approximation? We can do it after the first stage by approximating 1.00_2 to 0.11_2. In this case, the maximum output of the Sinc decimator is 0.11000000_2. Or we can do it after the second stage by approximating 1.00000_2 to 0.11111_2. In this case, the maximum output of the Sinc

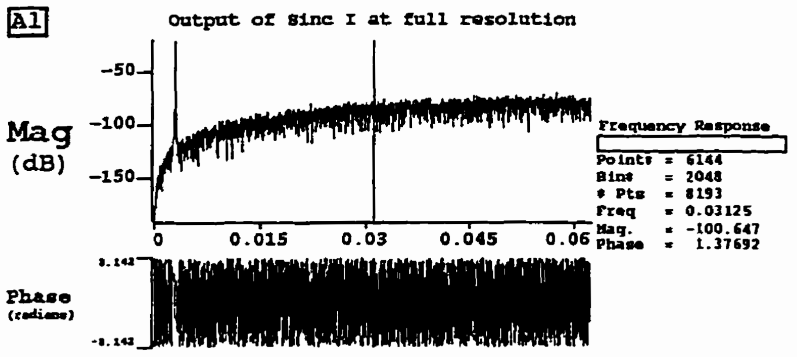


Figure 5.46: The output frequency spectrum of the Sinc decimator given in Figure 5.39, having $k = 1$, $m = 3$, and $n = 3$, operating at full resolution as shown in Figure 5.39. The input is the output of a second-order lowpass Sigma-Delta modulator, having an input sine wave of amplitude 0.5 and a frequency 0.0031 the sampling frequency.

decimator is 0.11111000_2. Finally, we can do the approximation at the output of third and final stage.

The advantage of doing this approximation in an earlier stage is to reduce the datapath width in the following stages. The disadvantage being lower SNR at the Sinc output. Figure 5.47 shows the frequency spectrum at the output of the Sinc decimator, under the same conditions that have been previously explained, and with the approximation done after the first stage. Figure 5.48 shows the same frequency spectrum but with the approximation done after the second stage.

The reduction in computational complexity resulting from approximating the sampled signal after the second stage, by the elimination of the integer bit, is 4.5%. The degradation in the SNR at the output of the Sinc decimator, evident from comparing Figure 5.48 to Figure 5.46, is negligible. While the reduction in the computational complexity resulting from approximating the sampled signal after

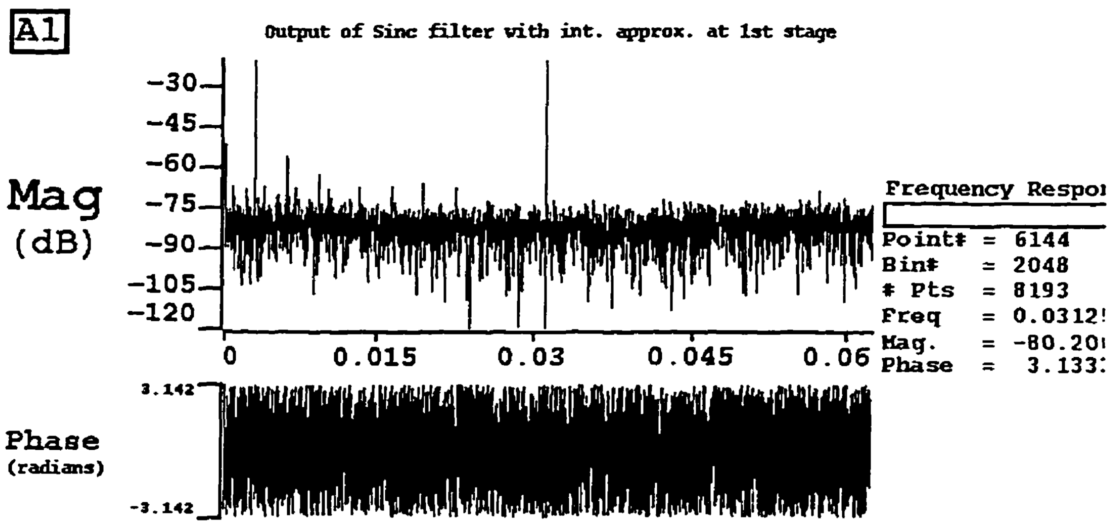


Figure 5.47: The output frequency spectrum of the Sinc decimator given in Figure 5.39, having $k = 1$, $m = 3$, and $n = 3$, the sampled signal is approximated after the first stage by eliminating the integer bit. The input is the output of a second-order lowpass Sigma-Delta modulator, having an input sine wave of amplitude 0.5 and a frequency 0.0031 the sampling frequency.

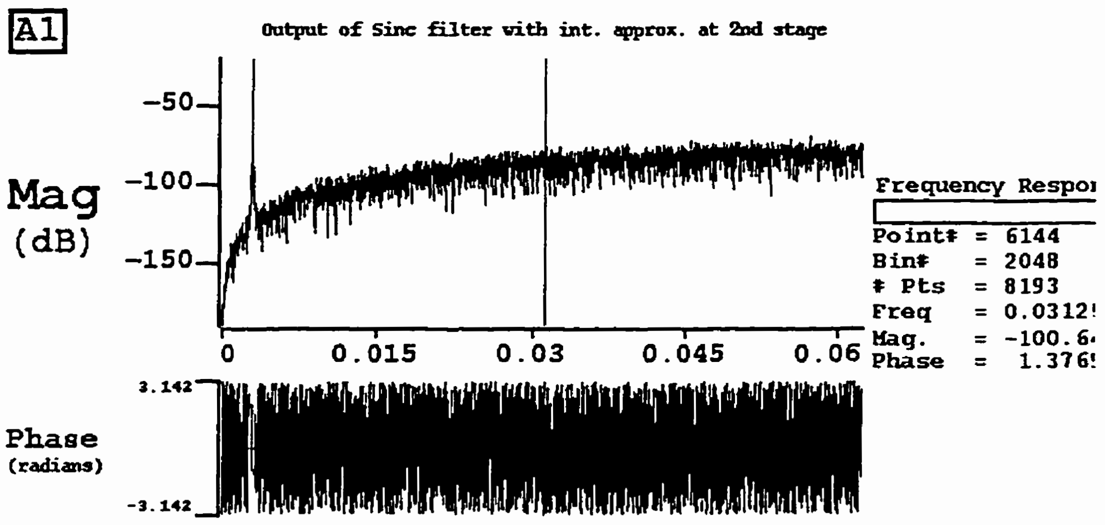


Figure 5.48: The output frequency spectrum of the Sinc decimator given in Figure 5.39, having $k = 1$, $m = 3$, and $n = 3$, the sampled signal is approximated after the second stage by eliminating the integer bit. The input is the output of a second-order lowpass Sigma-Delta modulator, having an input sine wave of amplitude 0.5 and a frequency 0.0031 the sampling frequency.

the first stage, by the elimination of the integer bit, is 13.5%. However, in this case, the degradation in the SNR at the output of the Sinc decimator, evident from comparing Figure 5.47 to Figure 5.46, is substantial.

To further reduce the datapath width at the output of the Sinc decimator, it is possible to eliminate the least significant fraction bits. This elimination can be done after any Sinc stage. The earlier it is preformed, the more the reduction in the computational complexity and the lower the SNR performance at the output of the Sinc decimator. Notice that, when the least significant bit is eliminated after the last stage, there is no saving in the computational complexity of the Sinc decimator. However, the computational complexity of the following stage, which is the LPDF, is reduced due to the lower datapath width.

Two methods for fraction bit elimination were considered. The first is truncation. The second is alternate up/down rounding, were the sample is rounded up for one sample and rounded down for the next. Table 5.3 gives the SNR performance at the output of Sinc decimator for the two bit-elimination methods and with rounding performed after the first, second and third Sinc stages. The noise calculated at the output of the Sinc decimator is limited to the frequency band $[-\frac{f_s'}{4}, \frac{f_s'}{4}]$.

Notice, that when performing fraction bit elimination after the first or second stages, alternate up/down rounding has substantially better performance than truncation. Notice also, that fraction bit elimination after the first stage leads up to 7 dB degradation in the SNR, when alternate up/down rounding is used. While bit elimination after the second stage leads only to a 3 dB degradation in the SNR, which is equivalent to half a bit. Hence, it is more appropriate to use. If we perform, fraction bit elimination after the second and the third stages, and integer bit elimination after the second stage, the SNR at the output of the Sinc decimator is

Table 5.3: The effect of fraction bit elimination on the SNR at the output of a Sinc decimator based on the first architecture. The input signal is the output of a second-order Sigma-Delta modulator having an input sine wave of an amplitude 0.5 and a frequency 0.0031 the sampling frequency.

Bit elimination method	Bit elimination stage	SNR	Saving in CC*
Truncation	1	15.6 dB	13.5 %
	2	32.7 dB	4.5 %
	3	39.7 dB	0
Up/down rounding	1	33.4 dB	13.5 %
	2	37.3 dB	4.5 %
	3	39.7 dB	0

* CC = computational complexity

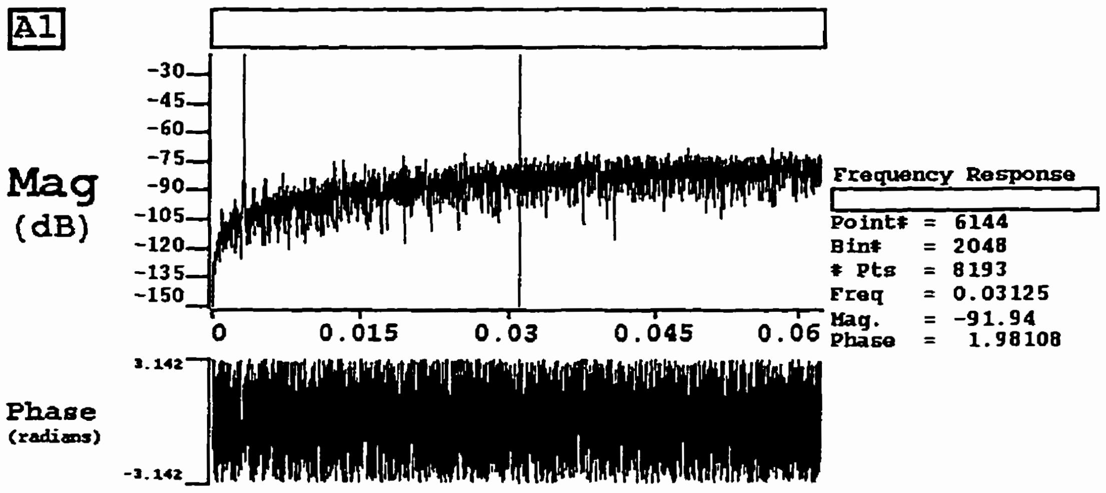


Figure 5.49: The output frequency spectrum of the Sinc decimator given in Figure 5.39, having $k = 1, m = 3$, and $n = 3$, and having a resolution 4 bits after the first stage, 5 bits after the second stage and 7 bits at the output of the Sinc decimator. The input to the Sinc decimator is the output of a second-order low-pass Sigma-Delta modulator, having an input sine wave of amplitude 0.5 and a frequency 0.0031 the sampling frequency.

36.4 dB. This is equivalent to a 3.8 dB (just over half a bit) reduction from the SNR of the full resolution case, while the resolution at the output has dropped by 3 bits from 10 to 7 bits. The computational complexity of the Sinc decimator is reduced by 9% in this case. The frequency spectrum at the output of the Sinc decimator when the output resolution is 7 bits is shown in Figure 5.49.

Now consider the effect of reducing the numerical accuracy on the output of a Sinc decimator based on the fourth architecture shown in Figure 5.43. When the output is at full resolution and each intergrator and differentiator is operating at the full resolution (10 bits), the output spectrum is identical to that of the first architecture shown in Figure 5.46. The output sample word has 1 integer bit, 1 sign

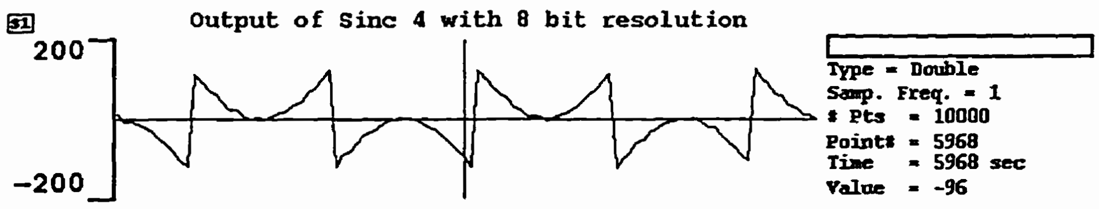


Figure 5.50: The output of a Sinc decimator based on the fourth architecture, having $k = 1, m = 3$, and $n = 3$, and having integrators and differentiators with a datapath width of 8 bits. The input to the Sinc decimator is the output of a second-order lowpass Sigma-Delta modulator, having an input sine wave of amplitude 0.5 and a frequency 0.0031 the sampling frequency.

bit and 8 fraction bits. If we eliminate the integer bit, by using 9 bit intergrators and differentiators. The output of the Sinc decimator is not affected, except if the output should have been $+1.00000000_2$, which is interpreted as -1.00000000_2 (a full scale error). For this output to occur, an input pattern consisting of 22 consecutive ones is required (see appendix B). This input pattern rarely occurs, and if we slightly limit the amplitude of the input signal it will never occur.

If we try to further reduce the datapath width of the integrators and differentiators to 8 bits, by removing a most significant bit. The output signal will be distorted as shown in Figure 5.50.

If the resolution of the first integrator is 8 bits, and the remainder of the datapath is 9 bits. The output signal will be substantially distorted. This is evident from the frequency spectrum shown in Figure 5.51.

The resolution of any stage of a Sinc decimator based on the fourth architecture cannot fall below 9 bits (by removing most significant bits) without substantially degrading the output performance. So far we tried to reduce the resolution by the elimination of the most significant bits. However, if we try to reduce the

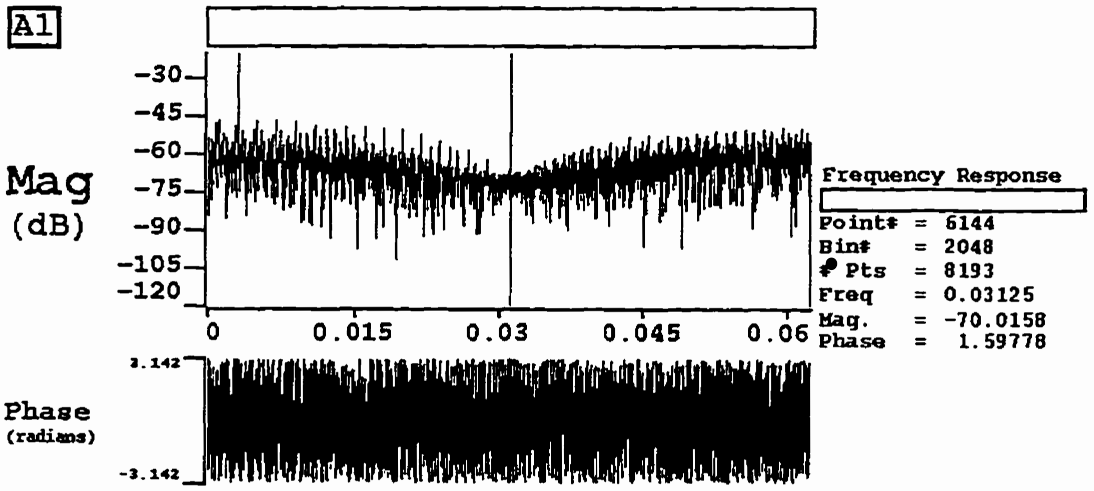


Figure 5.51: The output frequency spectrum of a Sinc decimator based on the fourth architecture, having $k = 1, m = 3$, and $n = 3$, and having the first integrator of resolution 8 bits, and the remainder of the datapath with resolution 9 bits. The input of the Sinc Decimator is the output of a second-order lowpass Sigma-Delta modulator, having an input sine wave of amplitude 0.5 and a frequency 0.0031 the sampling frequency.

Table 5.4: The effect of fraction bit elimination on the SNR at the output of a Sinc decimator based on the fourth architecture. The input signal to the Sinc decimator is the output of a second-order lowpass Sigma-Delta modulator having an input sine wave of an amplitude 0.5 and a frequency 0.0031 the sampling frequency. The differentiators have same resolution as that of the last integrator stage.

Bits per integrator stage			SNR	Saving in CC*
9	9	9	40.2 dB	10 %
9	9	8	37.1 dB	14 %
9	9	7	32.1 dB	18 %
9	9	6	25.9 dB	22 %
9	8	8	24.4 dB	17 %
9	7	7	21.3 dB	24 %

* CC = computational complexity. This is relative to the full resolution case were each integrator has 10 bits.

resolution by the elimination of the least significant bits. The degradation in the output performance is more graceful. The results of these simulations are shown in Table 5.4.

The simulation results that have been presented for a Sinc decimator based on the fourth architecture indicate that, reducing the computational complexity by the elimination of the most significant bit results in a reduction in the computational complexity by 10%, with a negligible degradation in the SNR performance at the output of the Sinc decimator. The reduction of the computational complexity, in this architecture, is greater than the corresponding reduction in computational complexity of a Sinc decimator based on the first architecture when the most significant bit is eliminated after the second Sinc stage.

When the output has 8 bits resolution (eliminating the least significant bit in addition to the most significant bit), and with the first and second integrators having a 9 bit resolution, the output SNR is 37.1 dB. The reduction in the computational complexity, due to the elimination of the least significant bit only, is 4%. Note that, this case is comparable to that of a Sinc decimator based on the first architecture were the approximation is performed after the second stage. In that case, the output SNR is 37.3 dB, and the reduction in computational complexity is 4.5%.

If we further reduce the numerical accuracy of the third integrator stage to 7 bits, the output SNR is reduced by 5 dB (almost 1 bit) to 32.1 dB. In this case, the degradation in performance is too large to make it a practical solution. However, if we choose the resolution of the third integrator stage to be 8 bits, and perform the approximation from 8 bits to 7 bits at the output of the Sinc decimator, by dropping the least significant bit. In this case, the SNR at the output of the Sinc decimator is 35.8 dB. The overall reduction in computational complexity is 14%. The reduction in the numerical accuracy from the full resolution case is 4.5 dB (0.75 bits). While the datapath width of the output of Sinc decimator has been reduced from 10 bits to 7 bits.

Even though the reduction in the computational complexity achieved in a Sinc decimator based on the fourth architecture (14%) is greater than that achieved in a Sinc decimator based on the first architecture (9%). Yet, the computational complexity (number of one-bit addition operations required per output) of the first architecture is still 2.8 times lower than that of the fourth architecture operating at the lower numerical accuracy.

5.9 Power Efficient Lowpass Filter Design

The lowpass decimation filter (LPDF) is based on the multiply-accumulate architecture. This architecture is shown in Figure 5.52. The filter designed is a linear phase FIR filter, it has symmetric coefficients. The filter architecture has been modified [147] to take into account the symmetry of the LPDF coefficients, by reading two values from the memory adding them together, and then multiplying them by the desired coefficient. This reduces the number of multiplications by 50%. However, the number of RAM reads remains unchanged. By simulation of the different blocks it was found that, the RAM dissipates 35% of the total LPF power dissipation. Hence, the power saving achieved by eliminating multiplications due to filter symmetry is 33%.

The number of multiplications is further reduced by using a halfband filter [120]. Halfband filters have half of their coefficients zero. However, this does not reduce the number of multiplications by 50% because halfband filters are usually longer (have more delay units and hence more taps) than non-halfband filters that provide the same out-of-band attenuation.

To give the same stop band attenuation, halfband filters are designed 25%-30% longer than their corresponding non-halfband filters. Hence, the designed halfband filters require 30%–35% less multiplications (and RAM reads) than the non-halfband filters. Hence, the power saving achieved by halfband filters is 30%-35%.

Figure 5.53 is the timing diagram for the LPDF. Every two consecutive RAM reads (one from the A stream and the other from the B stream) are added and multiplied by a single ROM coefficient. During the RAM's read state, the output of the LPDF is being accumulated. At the end of the read state, the LPDF output

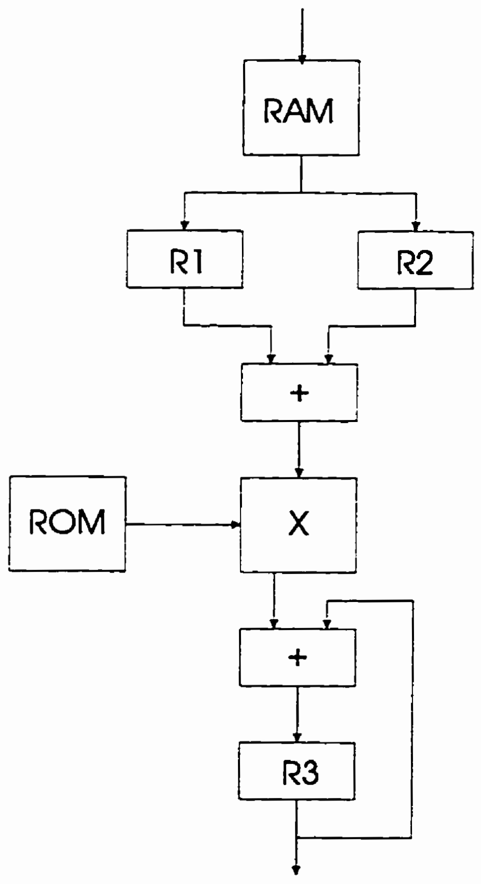


Figure 5.52: Conventional multiply-accumulate filter architecture.

is generated, the RAM changes to the write state, and two samples are written into the RAM. The RAM than changes back to the read state. Every read cycle, the samples read are advanced two memory locations. The fact that two samples are written into the RAM every time a single sample is generated at the output means that the LPDF does down-sampling by a factor of two.

The samples are written into the RAM at non-uniform periods (two samples are written every read/write cycle). While, the input samples to the LPDF are at uniform periods. To achieve synchronization, the synchronization block shown in Figure 5.54 is used [147]. This synchronization block allows the memory to operate between two quasi-synchronous blocks. It eliminates the need for an interrupt every time an input sample is available. This simplifies the design of the control unit of the LPDF.

Figure 5.55 and Figure 5.56 show the timing diagrams for the synchronization block for two cases. In the first case, Figure 5.55, clkI doesn't change state during the RAM write state (when R/W $= 0$). Notice that when $w = 1$, the output to the RAM is always the earliest sample found in the two registers R1 and R2. When $w = 0$, the output to the RAM is the latest sample found in the two registers R1 and R2.

In the second case, Figure 5.56, clkI changes state during the RAM write state. In this case, the value of clkI, during the moment R/W changes from 1 to 0 (read to write state), is stored in clkI'. This is necessary to avoid the neglection of the samples stored in R2, for the case clkI changes from 0 to 1 when R/W $= 0$, this is the case shown in Figure 5.56. Or the neglection of the samples stored in R1, for the case clkI changes from 1 to 0 when R/W $= 0$.

The multiplier-accumulator (MAC) used in the LPDF, has been designed such

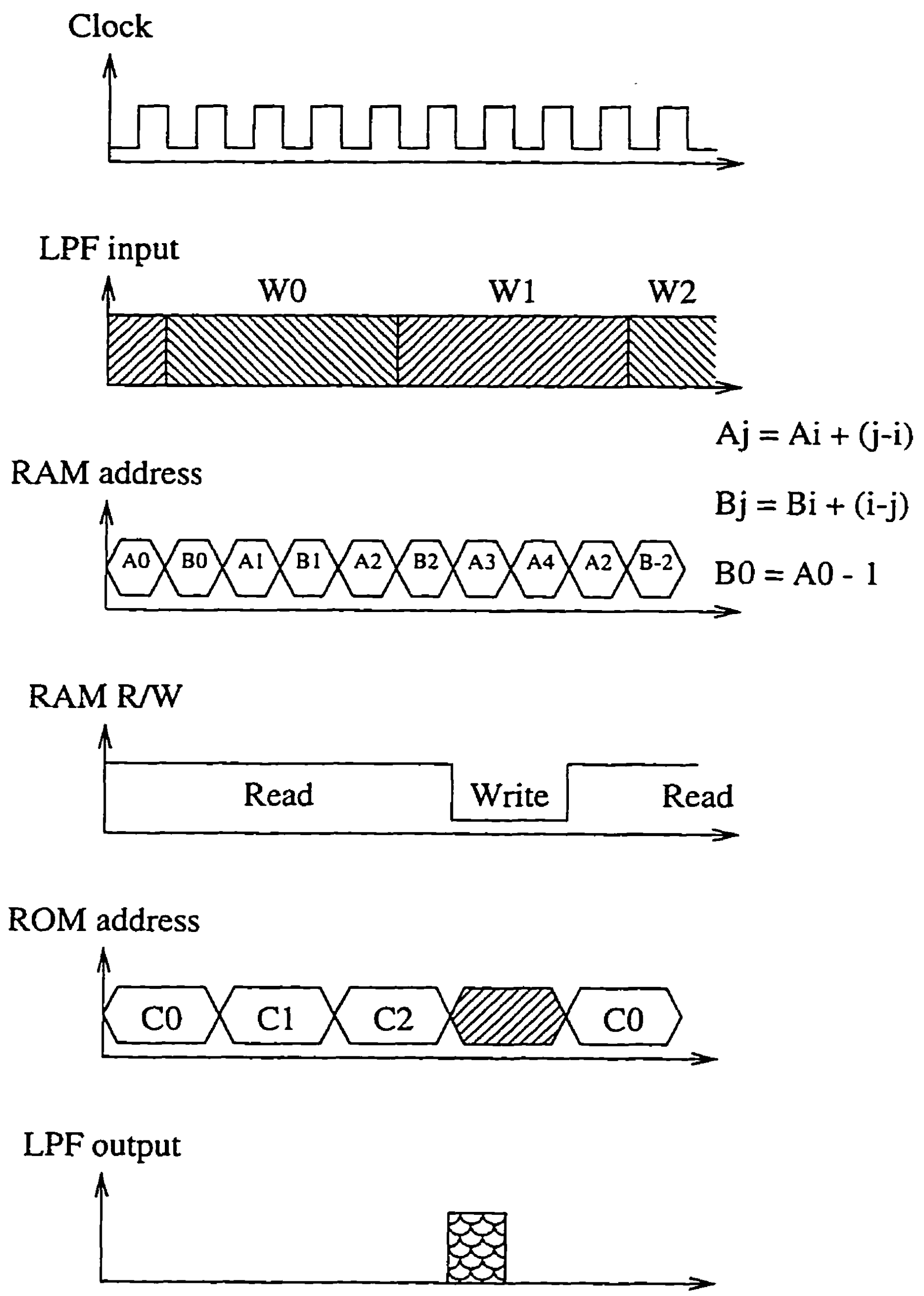


Figure 5.53: Lowpass filter timing diagram.

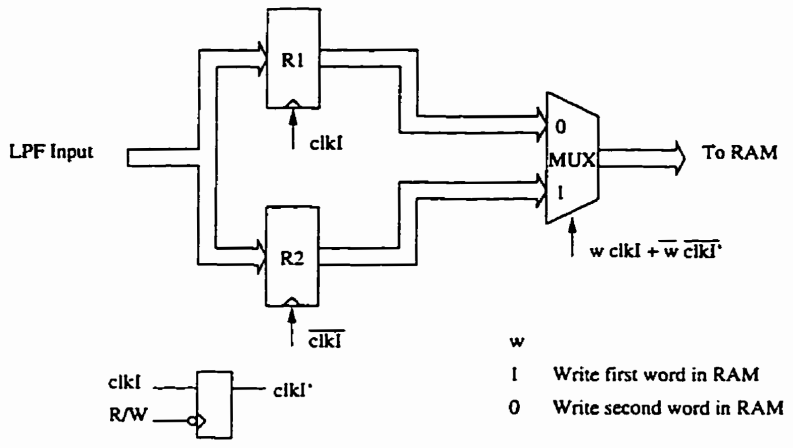

R1
R2
LPF Input
clkI
$\overline{\text{clkI}}$
0
MUX
1
To RAM
w clkI + $\overline{\text{w}}$ $\overline{\text{clkI'}}$
clkI
R/W
clkI'
w
1 Write first word in RAM
0 Write second word in RAM

Figure 5.54: Synchronization Block.

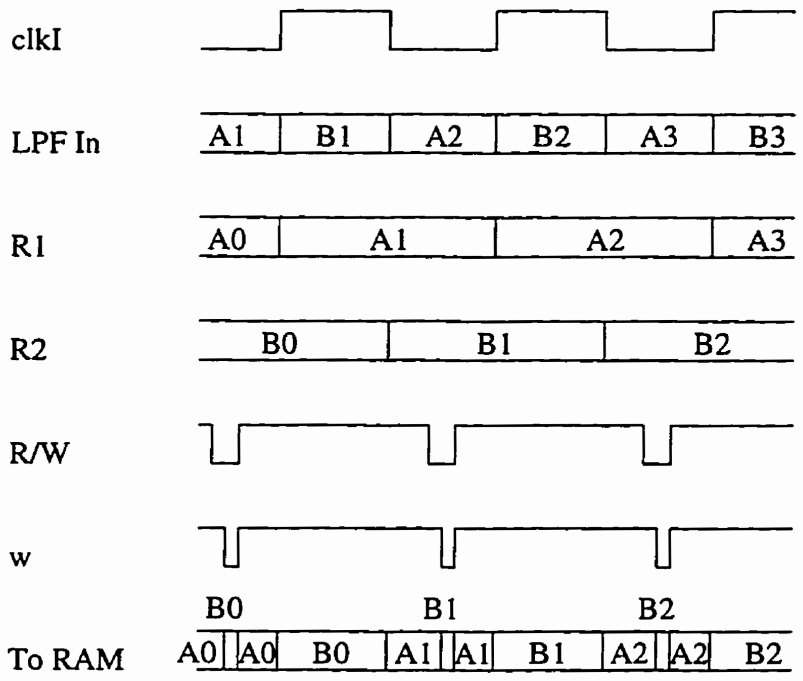


Figure 5.55: Synchronization block timing diagram.

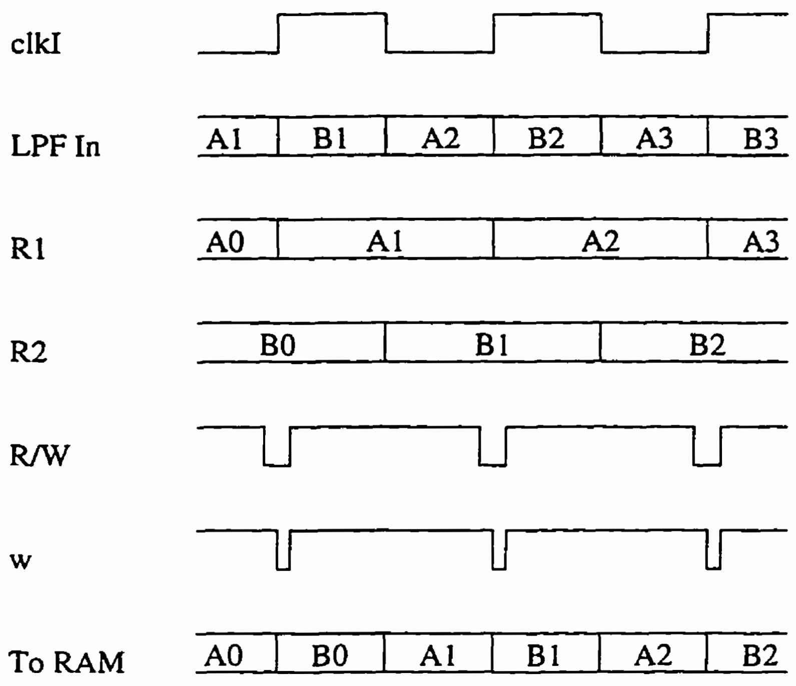


Figure 5.56: Synchronization block timing diagram.

that the adder is interleaved in the multiplier array. Figures 6.7 and 6.8 shows the architecture used to implement the MAC [148]. For a 20 $\times$ 20 multiplier-accumulator, this structure achieves a 17% reduction in power dissipation.

5.9.1 Variable Resolution Lowpass Architecture

To reduce the power dissipation when a lower resolution is sufficient, the datapath is divided into parallel units as shown in Figure 5.57 [146] [147]. The blocks corresponding to the least significant bits are deactivated when a lower resolution is sufficient. In Figure 5.57, the datapath is shown consisting of 3 parallel units, making its resolution one of three values: 12, 16 or 20 bits.

The division of the datapath into parallel units increases the overhead in terms

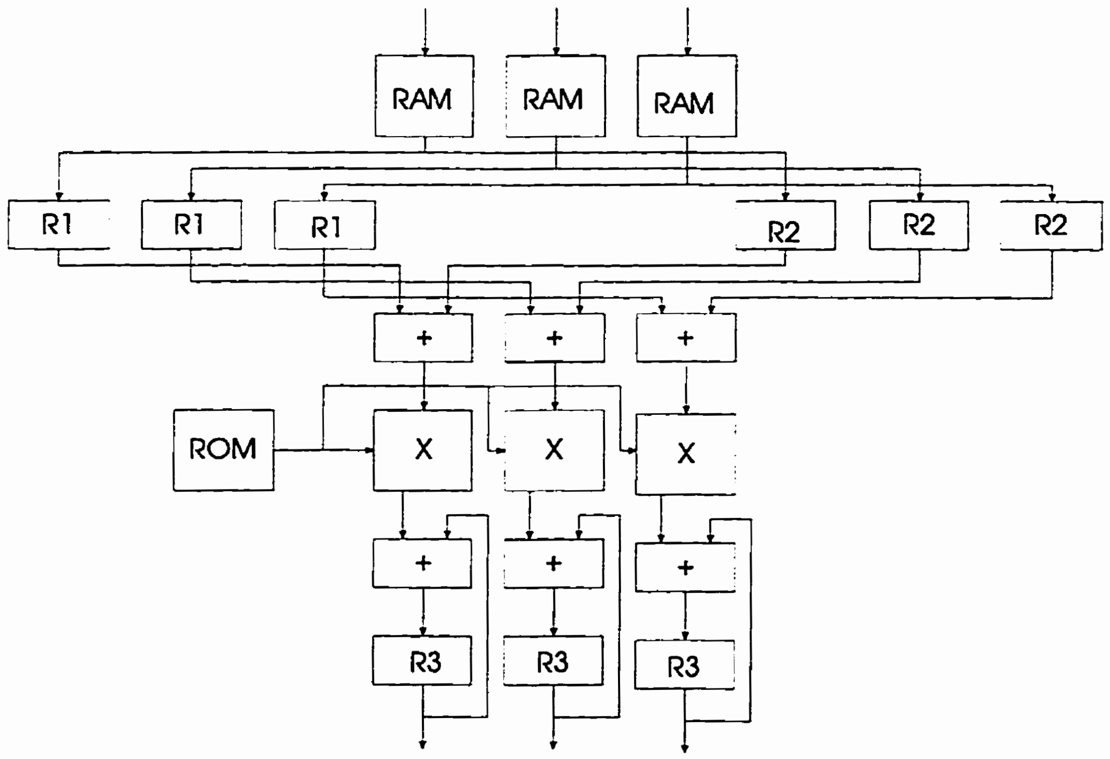


Figure 5.57: The modified filter architecture.

of power dissipation (as well as area). It is the purpose of this section to analyze this block deactivation technique, to determine the amount of power saving that can be achieved using it, and to determine the optimum number of parallel units the datapath is divided into for a certain application and technology.

The datapath of the LPF consists of functional blocks as shown in Figure 5.57. The power dissipation of each functional block F can be divided into two components, the first component is a constant independent of the datapath width, while the second component is proportional to the datapath width:

$$P_F = P_{0F} + P_{1F}N \tag{5.21}$$

Equation 5.21 is valid for a multiplier unit only if the width of the multiplicand is variable, while the width of the multiplier is fixed.

Table 5.5 gives the power components for each functional block in the datapath, in addition to the power components of the entire datapath. These numbers are for a $0.5\mu m$, 3.3 Volt CMOS technology [150]. For the multiplier unit, it is assumed that the multiplier has a fixed width of 20 bits. The total power dissipation is for the architecture given in Figure 5.52, which has; 1 RAM, 1 multiplier, 2 adders and 3 registers. Assume that the total power components are P_{0T} and P_{1T} for the fixed and width-dependent power components respectively. Thus, if the datapath is not divided into parallel units the total power dissipation is given by:

$$P_T = P_{0T} + P_{1T}N \tag{5.22}$$

If the datapath is divided into M parallel units. The power dissipation when the full accuracy of the datapath is required is:

Table 5.5: Power components for the individual functional blocks and the entire datapath of the architecture shown in Figure 5.52.

Functional Unit	P_{0F}	P_{1F}
RAM	$250\mu W$/MHz	$55\mu W$/MHz
Multiplier	$70\mu W$/MHz	$60\mu W$/MHz
Adder	0	$5\mu W$/MHz
Register	0	$5\mu W$/MHz
Total	$320\mu W$/MHz	$140\mu W$/MHz

$$P_T = P_{0T}M + P_{1T}N \tag{5.23}$$

Notice that, the power dissipation in this case has increased by $(M-1)P_{0T}$. In the low resolution case, the power dissipation is given by:

$$P_T = P_{0T} + P_{1T}N_L \tag{5.24}$$

In this case, the power dissipation has decreased by $(N - N_L)P_{1T}$. N_L is the datapath width in the low resolution case. The amount of power saving is dependent on the percentage of time the datapath is required to operate at each resolution, as well as the ratio between the two power components.

Suppose that the datapath can take one of M resolution. The lowest resolution, N_L bits, has a probability α. The remaining resolutions:

$$N_L + \frac{N-N_L}{M-1} \times 1, N_L + \frac{N-N_L}{M-1} \times 2, \ldots, N \tag{5.25}$$

are equi-probable and have a probability:

$$\frac{1-\alpha}{M-1} \tag{5.26}$$

Using this probability distribution it is possible to find the average power dissipation:

$$P_{avg} = P_{0T}[1 + \frac{M}{2}(1-\alpha)] + P_{1T}[N_L + \frac{M}{2(M-1)}(N-N_L)(1-\alpha)] \tag{5.27}$$

Define Pr, the relative power, to be the ratio between the power dissipation of the system with parallel datapath units, to that dissipated in a single datapath system. ρ is defined as:

$$\rho = \frac{P_{0T}}{P_{1T}} \tag{5.28}$$

Therefore,

$$\mathrm{Pr} = \frac{\rho}{\rho+N}[1 + \frac{M}{2}(1-\alpha)] + \frac{1}{\rho+N}[N_L + \frac{M}{2(M-1)}(N-N_L)(1-\alpha)] \tag{5.29}$$

To minimize Pr, the optimum number of units, the datapath should be divided into, is given by:

$$M_{opt} = \sqrt{\frac{N-N_L}{\rho}} + 1 \tag{5.30}$$

For the design parameters of the lowpass decimation filter being designed, M_{opt} is 3. Figures 5.58 and 5.59 show the relationship between the relative power dissipation and M, the number of parallel units the datapath is divided into for different

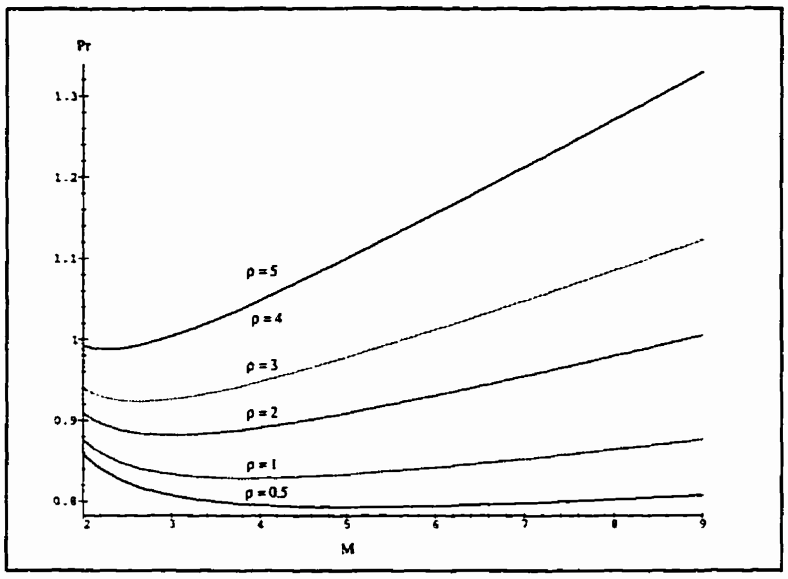


Figure 5.58: Relative power dissipation, Pr, of a lowpass filter using block deactivation versus M, for $\alpha = 0.4$, $N = 20$ and $N_L = 12$. M is the number of parallel datapath units.

values of α and ρ. N was taken to be 20 and N_L to be 12. Figure 5.60 shows the relative power dissipation, for a 3 parallel-unit datapath, versus γ and δ. γ is the probability that a 12 bit resolution is sufficient. δ is the probability that a 16 bit resolution is sufficient.

From the results shown in Figures 5.58 — 5.60, we can make the following conclusions:

- The optimum value of M (which makes Pr minimum) depends on ρ. As ρ decreases, M_{opt} increases.
- For the total power components, P_{0T} , P_{1T}, given in Table 5.5, $\rho = 2.29$, we find that M_{opt} is 3, which is the value used in the design of the LPF decimation

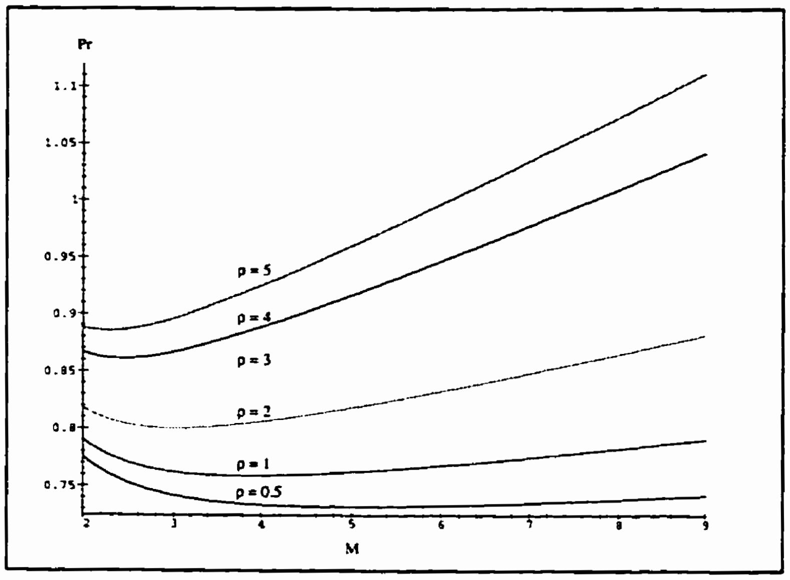


Figure 5.59: Relative power dissipation, Pr, of a lowpass filter using block deactivation versus M, for $\alpha = 0.6$, $N = 20$ and $N_L = 12$. M is the number of parallel datapath units.

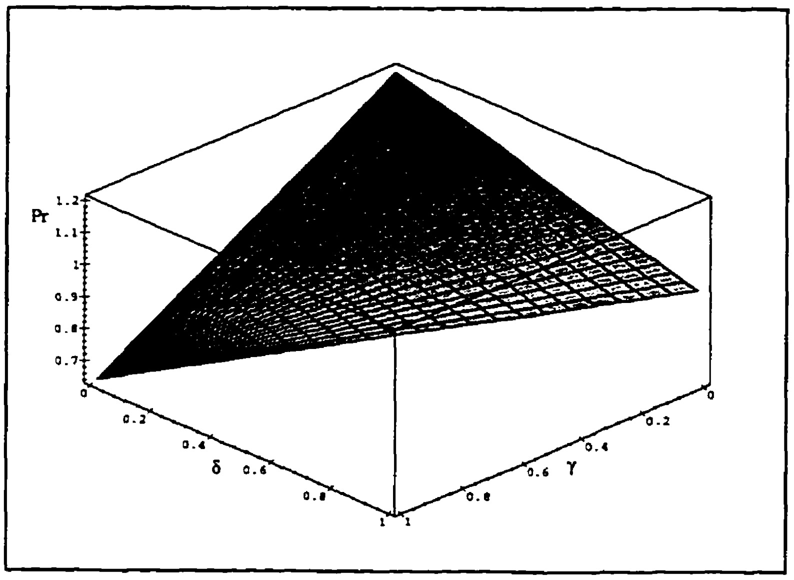


Figure 5.60: Relative power dissipation, Pr, of a lowpass decimation filter consisting of a three-unit parallel datapath versus γ and δ, for $M = 3$ and $\rho = 2.29$. The datapath resolution can be: 12, 16, or 20 bits. γ is the probability that a 12 bit resolution is sufficient. δ is the probability that a 16 bit resolution is sufficient.

filter.

- The maximum reduction in power dissipation that can be achieved using this algorithm is 35%. This is achieved when a 12 bit resolution is sufficient for most of the time.

The analysis of the block deactivation technique presented so far assumes that the multiplier unit has a fixed multiplier width of 20 bits. If the width of the multiplier as well as the width of the multiplicand are allowed to vary, which is the case in the multiplier-accumulator array design in chapter 6, the power saving that can be achieved with the block deactivation technique, presented in this section, is even greater.

Figure 5.61 shows the relative power dissipation, for a three-unit parallel datapath, versus γ and δ. γ is the probability that a 12 bit resolution is sufficient. δ is the probability that a 16 bit resolution is sufficient. The maximum reduction in power dissipation that can be achieved is 40%. This is achieved when a 12 bit resolution is sufficient for most of the time.

In this section, several low-power techniques have been employed in the design of the lowpass decimation filter. The power saving each technique can achieve is shown in Table 5.6. The total reduction in power dissipation is about 4 times.

5.10 VLSI Implementation of the Decimation Filter

The decimation filter is designed to generate a 12 - 20 bits resolution sampled signal from a 1 - 2 bits resolution oversampled signal. The first stage of the deci-

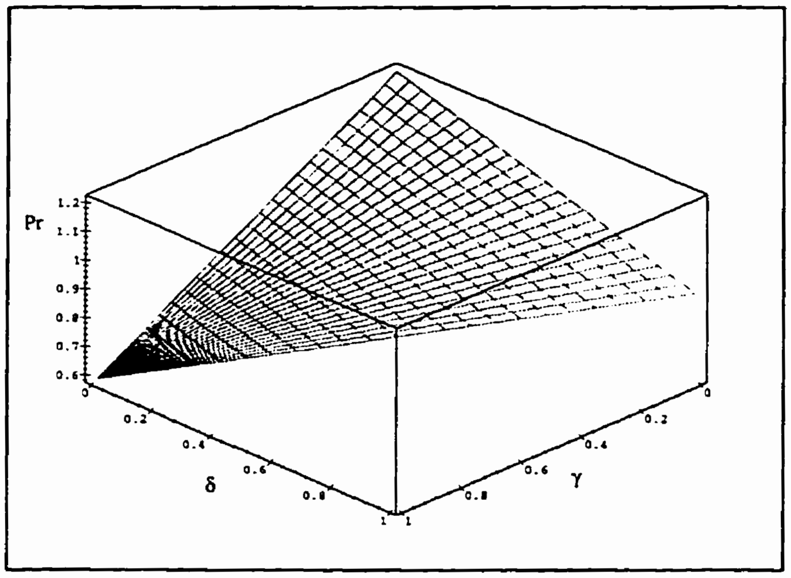


Figure 5.61: Relative power dissipation, Pr, of a lowpass decimation filter consisting of a three-unit parallel datapath, versus γ and δ, for $M = 3$. The lowpass decimation filter uses the programmable multiplier-accumulator array designed in chapter 6. The power dissipation of the other components are as given in Table 5.5. The datapath resolution can be: 12, 16, or 20 bits. γ is the probability that a 12 bit resolution is sufficient. δ is the probability that a 16 bit resolution is sufficient.

Table 5.6: Power savings for high-level low-power techniques used in the design of the lowpass decimation filter.

Low power technique	Power saving
Symmetric filter coefficients	33%
Halfband filter	30 - 35%
Interleaved MAC array	17%
Datapath division	10 - 40%
TOTAL	**65 – 78%**

mation filter is the Sinc decimator. The order of the Sinc decimator, as well as its decimation factor, can be programmed.

The Sinc decimator can be a third-order Sinc decimator. This accepts a one bit sampled signal from a fourth-order bandpass Sigma-Delta modulator. The Sinc decimator can also be a fourth-order Sinc decimator. This accepts a two-bit sampled signal from a sixth-order bandpass Sigma-Delta modulator. The decimation factor of the Sinc decimator can be 8, 16 or 32.

The second stage of the decimation filter is the lowpass decimation filter. The lowpass decimation filter consists of two stages in cascade, each stage does decimation by a factor of two. Each lowpass stage is a linear phase stage. Each lowpass decimation filter can be programmed to be halfband filter or a non-halfband filter. The output resolution of each lowpass stage can be programmed to be 12, 16 or 20 bits. The number of taps required in the first LPF stage is 15 taps, while the number of taps required in the second LPF stage is 47 taps.

Figure 5.62 shows a block diagram of the decimation filter. The decimation

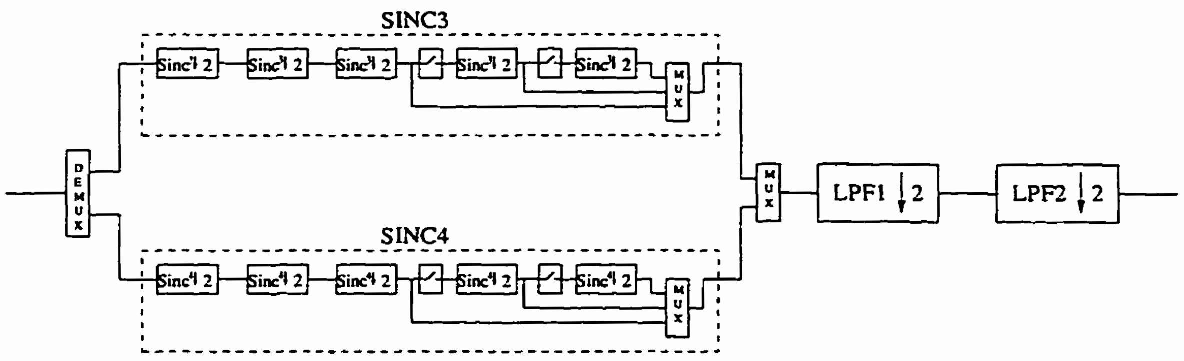


Figure 5.62: Block diagram of the designed decimation filter.

Table 5.7: The number of transistors required for each stage of the lowpass decimation filter.

SINC3	9,000
SINC4	14,000
LPF1	82,000
LPF2	82,000

filter was designed in a $0.5\mu m$, 3.3 Volt CMOS technology. The total number of transistors required for the decimation filter is 187 K. Table 5.7 shows the number of transistors required for each stage of the decimation filter. The total area of the decimation filter is 7.4mm $\times$ 6.4mm. The VLSI layout of the decimation filter is shown in Figure 5.63.

5.11 Chapter Summary

Parallelism by 4x of analog signal processors is applied to the design of a bandpass Sigma-Delta modulator. The speed of the modulator is increased without increas-

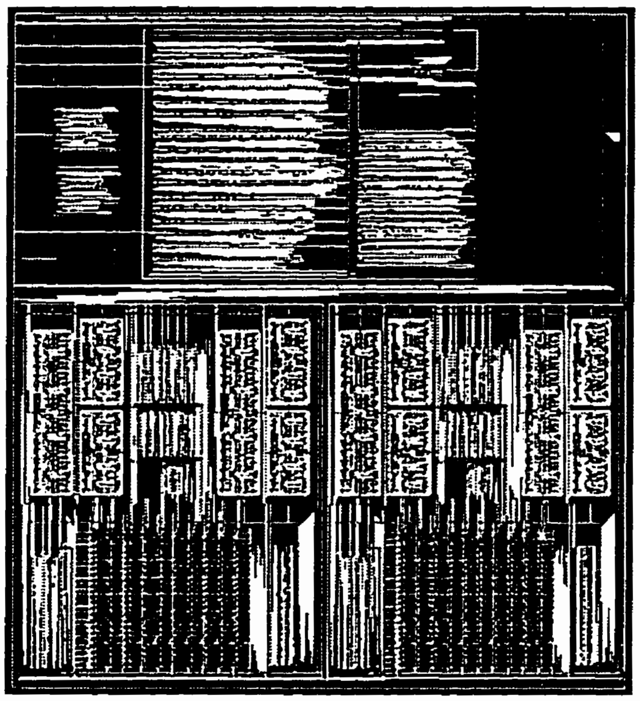


Figure 5.63: VLSI layout of the decimation filter.

ing the speed requirement of the individual building blocks. Several architectures were considered in terms of their resilence to implementation details such as mismatch and gain errors. A switched-capacitor circuit has also been designed for the proposed modulator.

Several high-level low-power design techniques have been incorporated into the design of the decimation filter. These include; operation minimization, multiplier elimination and block deactivation. These techniques lower the computational complexity by eliminating redundant and irrelevant computations. In the case of the Sinc decimator, eliminating redundant computations involves using architectures that avoid unnecessary computations due to decimation. Reducing the datapath width of the Sinc decimator, eliminates irrelevant computations. These are the computations that have little effect on the output numerical accuracy, because the numerical accuracy is limited by some other block in the system. In this case, the other block is the Sigma-Delta modulator.

The lowpass decimation filter also eliminates irrelevant computations, by using a block deactivation technique that avoids unnecessary computations when a lower resolution is sufficient. The other low-power techniques used in the lowpass decimation filter include operation interleaving and multiplier elimination. The decimation filter, with a programmable resolution that varies from 12 to 20 bits, has been designed in a $0.5\mu m$, 3.3 Volt CMOS technology.

Chapter 6

Low-Power Multiplier-Accumulator Array

Several low-power design techniques have been applied to the design of a power-efficient multiplier-accumulator (MAC) array. The addition operation has been interleaved into the multiplier array. This can achieve up to 27% saving in power dissipation. The MAC array is designed to have a programmable resolution so that the blocks corresponding to the least significant bits can be deactivated when a lower resolution is sufficient, this achieves up to 50% saving in power dissipation. The multiplier-accumulator has been designed in a $0.5\mu m$, 3.3 Volt CMOS technology.

6.1 Introduction

The rapid development in integrated circuit technology has led to the emergence of powerful, faster and smaller digital signal processors. Many of the functions that were performed in the analog domain are now being performed digitally. Digital processing not only improves quality, but it enhances the performance as well by allowing more programmability.

There is a trend to continue with the digital processing to higher speed analog signals. To be able to do this power-efficient DSP architectures are required. At the heart of a digital signal processor is the multiply-accumulate operation.

In this chapter, the design of a power-efficient multiplier accumulator (MAC) is investigated. Several low-power techniques have been incorporated into the design of this MAC. The addition operation has been interleaved into the multiplier array, this achieves a power saving of up to 27% for a 10×10 array. The MAC is designed to have a programmable resolution so that the blocks corresponding to the least significant bits can be deactivated when a lower resolution is sufficient. This achieves a power saving of up to 50%.

The multiplier of the MAC array is based on the modified Booth algorithm. The accumulator's input and output are in the sum-carry representation.

In section 6.2, the modified Booth algorithm multiplier is presented. The sign extension algorithm and the multiplier array architecture are developed in that section as well. In section 6.3, the multiplier-accumulator array is developed. The resolution-programmable MAC array is developed in section 6.4. The MAC array has been designed in a $0.5\mu m$, 3.3 Volt CMOS technology, this is discussed in section 6.5.

6.2 The Modified Booth Algorithm Multiplier

There are three types of multipliers [84]. Parallel multipliers [151] generate the partial products concurrently and sum them using a multi-operand adder. Sequential multipliers generate the partial products sequentially and accumulate them to the previously summed partial products. Array multipliers [152] — [155] are made up of an array of identical cells that generate the partial product and do the summation. Array multipliers have a regular structure making them suitable for VLSI implementation.

The modified Booth algorithm multiplier [156] [157] is used for multiplying two's complement operands. The number of partial products generated is half the number of multiplier bits. The multiplier is divided into overlapping three-bit groups. Each three-bit group generates a single partial product, this is done according to Table 6.1 [3]. The partial product can be -2, -1, 0, 1, 2 times the multiplicand. The multiplication factor is determined by calculating [84]:

$$\mathrm{MR}_{2i-1} + \mathrm{MR}_{2i} - 2\mathrm{MR}_{2i+1} \tag{6.1}$$

6.2.1 Sign Extension

Each partial product is shifted two bits from the previous partial product. This shifting process requires sign extension [3]. In the following we examine one way in which sign extension is done. Let the first partial product be

$$A_n A_{n-1} \ldots A_1 A_0$$

Where, A_n is the sign bit. Therefore, the value of A is given by:

$$A = -A_n 2^n + \sum_{i=0}^{n-1} A_i 2^i \tag{6.2}$$

Table 6.1: Partial product generation. PP is the partial product, MD is the multiplicand, and MR_i is the i th bit of the multiplier.

MR_{i+1}	MR_i	MR_{i-1}	PP	×2	×1	s
0	0	0	0 MD	0	0	0
0	0	1	1 MD	0	1	0
0	1	0	1 MD	0	1	0
0	1	1	2 MD	1	0	0
1	0	0	-2 MD	1	0	1
1	0	1	-1 MD	0	1	1
1	1	0	-1 MD	0	1	1
1	1	1	0 MD	0	0	0

Let the second partial product be

$$B_{n+2}B_{n+1}\dots B_3B_2$$

In this case, B_{n+2} is the sign bit. Therefore, the value of B is given by:

$$B = -B_{n+2}2^{n+2} + \sum_{i=2}^{n+1} B_i 2^i \tag{6.3}$$

To add these two numbers taking into account the sign of each, the following modifications are done to A and B. For A, complement the sign bit ($\overline{A}_n = 1 - A_n$), add 2^n. The number obtained NA, which is in unsigned binary representation, is related to A by:

$$\text{NA} = 2^{n+1} + A \tag{6.4}$$

Note that, this number has the same value as A to a resolution of $n+1$ bits.

For B, complement the bits B_{n+2} and B_{n+1}, add $2^{n+2}B_{n+1}$. The number obtained BN, which is in unsigned binary representation, is related to B by:

$$\mathrm{NB} = 2^{n+3} + B - 2^{n+1} \tag{6.5}$$

Adding NA and NB as unsigned binary numbers, we get:

$$NA + NB = 2^{n+3} + A + B \tag{6.6}$$

Which equals $A + B$ to a resolution of $n+3$ bits. This sign extension process is shown in Figure 6.1. The sign extension for the remaining partial products is identical to that of B. The sum of the partial products using this sign-extension is given by:

$$P = 2^{m+n-1} + \mathrm{MR} \times \mathrm{MD} \tag{6.7}$$

Where, m is the number of multiplier bits, n is the number of multiplicand bits. Thus, the product obtained is accurate to $m+n-1$ bits. To make the product accurate to $m+n$ bits, the $(m+n)$th bit is complemented. This is equivalent to subtracting 2^{n+m-1}.

6.2.2 Partial Product Generation

The multiplier is divided into overlapping three-bit groups. Each group is encoded into three bits $\times 2$, $\times 1$, and s, as given in Table 6.1. Figure 6.2 shows the logic circuit of the row-decoder used for encoding each three-bit group. If $\times 2 = 1$, the

$$\begin{array}{cccc}
2^{n+2} & 2^{n+1} & 2^{n} & 2^{n-1} \\
 & & -a_n & a_{n-1} \\
-b_{n+2} & b_{n+1} & b_n & b_{n-1}
\end{array}
\qquad\qquad
\begin{array}{cccc}
2^{n+2} & 2^{n+1} & 2^{n} & 2^{n-1} \\
 & & 1 & \\
b_{n+1} & & \bar{a}_n & a_{n-1} \\
\bar{b}_{n+2} & \bar{b}_{n+1} & b_n & b_{n-1}
\end{array}$$

(a) (b)

Figure 6.1: Sign extension. (a) Signed-digit representation. (b) Equivalent unsigned binary representation.

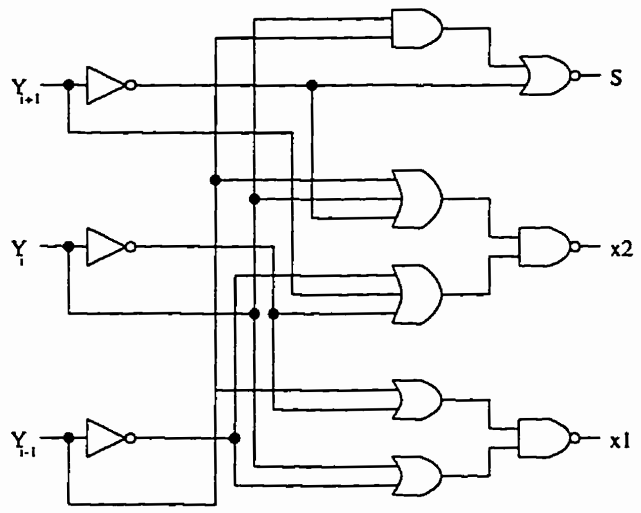


Figure 6.2: The row-decoder.

magnitude of the partial product is twice the magnitude of the multiplicand. If $\times 1 = 1$, the magnitude of the partial product equals that of the multiplicand. If $s = 1$, the partial product and the multiplicand have different signs. In terms of $\times 2$, $\times 1$ and s, the partial product is expressed as:

$$\text{PP} = (-1.s + 1.\bar{s})(2.\text{x2} + 1.\text{x1})\text{MD} \tag{6.8}$$

The i th bit of the partial product, PP_i, is generated according to the following equation:

$$\mathrm{PP}_i = (\mathrm{MD}_i.\mathrm{x1} + \mathrm{MD}_{i-1}.\mathrm{x2}) \oplus s \tag{6.9}$$

For $i = 0 \ldots n$. Where, MD_i is the i th bit of the multiplicand, $MD_{-1} = 0$ and $MD_n = MD_{n-1}$. n is the number of multiplicand bits. To correctly generate the two's complement of the multiplicand when $s = 1$, s should be added to the least significant bit of the partial product. Figure 6.3 shows the Booth multiplier array based on Equation 6.9 and the sign extension algorithm given in Figure 6.1 [158]. The array cells of Figure 6.3 are given in Figure 6.4. The logic circuit diagram of the row-decoder (RD) of Figure 6.3 is given in Figure 6.2. The logic circuit diagram of the partial product generator (PPGen) of Figure 6.4, described by Equation 6.9, is given in Figure 6.5.

The array multiplier of Figure 6.3 generates the product in the sum-carry representation. To get the two's complement product, the sum product word (PS) and the carry product word (PC) need to be added.

The resolution of the product is the sum of the resolution of the multiplier m and the multiplicand n. Assume that both the multiplier and the multiplicand are integers, then the value of the multiplier falls in the range $[-2^{m-1}, 2^{m-1} - 1]$. While the value of the multiplicand falls in the range $[-2^{n-1}, 2^{n-1} - 1]$. Thus the product falls in the range:

$$[-2^{m+n-2} + \min(2^{m-1}, 2^{n-1}), 2^{m-n-2}]$$

With the exception of 2^{m+n-2}, the other values can be represented using $m+n-1$ bits. The product, 2^{m+n-2}, is obtained when MR $= -2^{m-1}$ and MD $= -2^{n-1}$. If

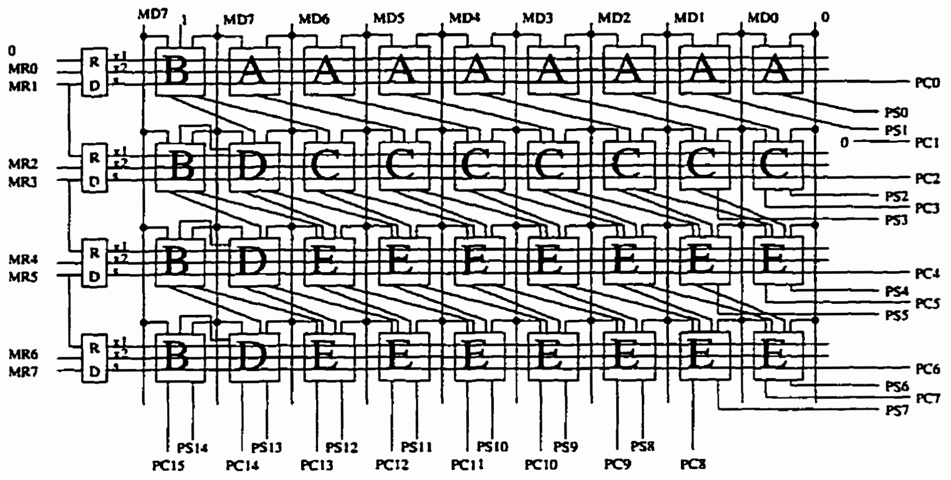


Figure 6.3: The Modified Booth Algorithm Array Multiplier.

this case is avoided, then $m + n - 1$ bits will be sufficient to represent the product. Thus, the carry from the left most cell in the last row of Figure 6.3 can be neglected.

The array multiplier of Figure 6.3 is the basis of the multiplier accumulator array (MAC array) and the programmable MAC array developed in the following sections.

6.3 The Multiplier-Accumulator Array

The multiplier-accumulator (MAC) accepts three operands, two operands are multiplied and accumulated (added) to the third operand. Figure 6.6 shows a block diagram for the MAC. In the MAC array [159] [160], the adder is interleaved into the array multiplier. The MAC array presented here is unique to other implementations in that [158]:

1. The accumulator input and output are in the sum-carry representation.

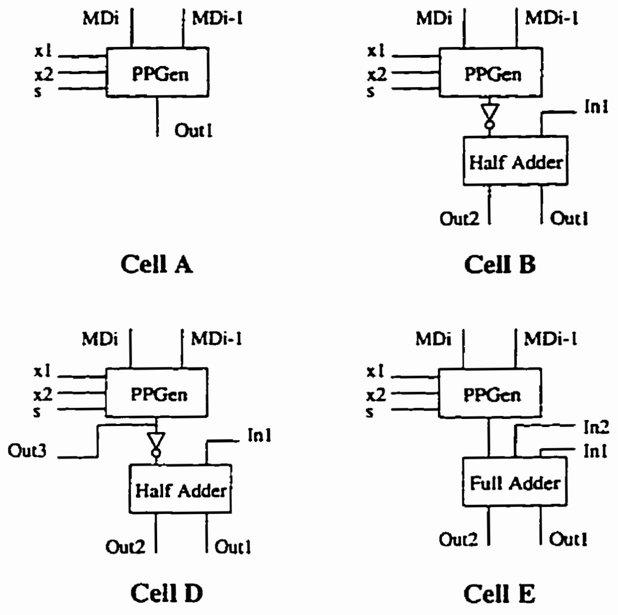


Figure 6.4: The array cells of the array multiplier given in Figure 6.3.

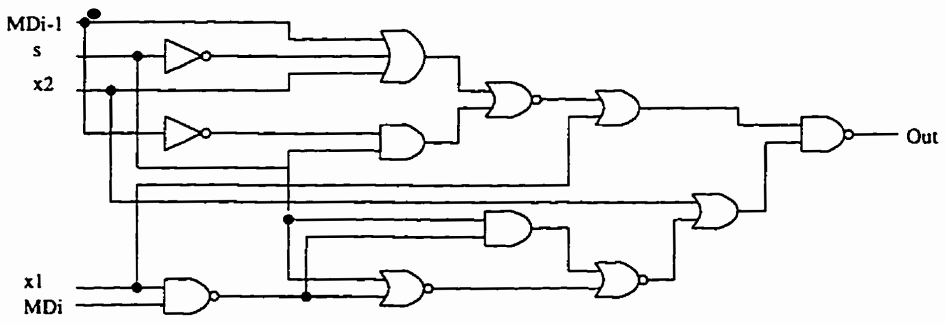


Figure 6.5: Partial product bit generator.

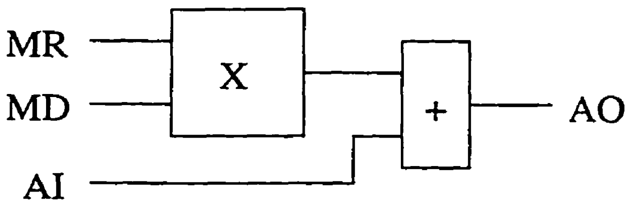


Figure 6.6: Block diagram of a multiplier-accumulator (MAC).

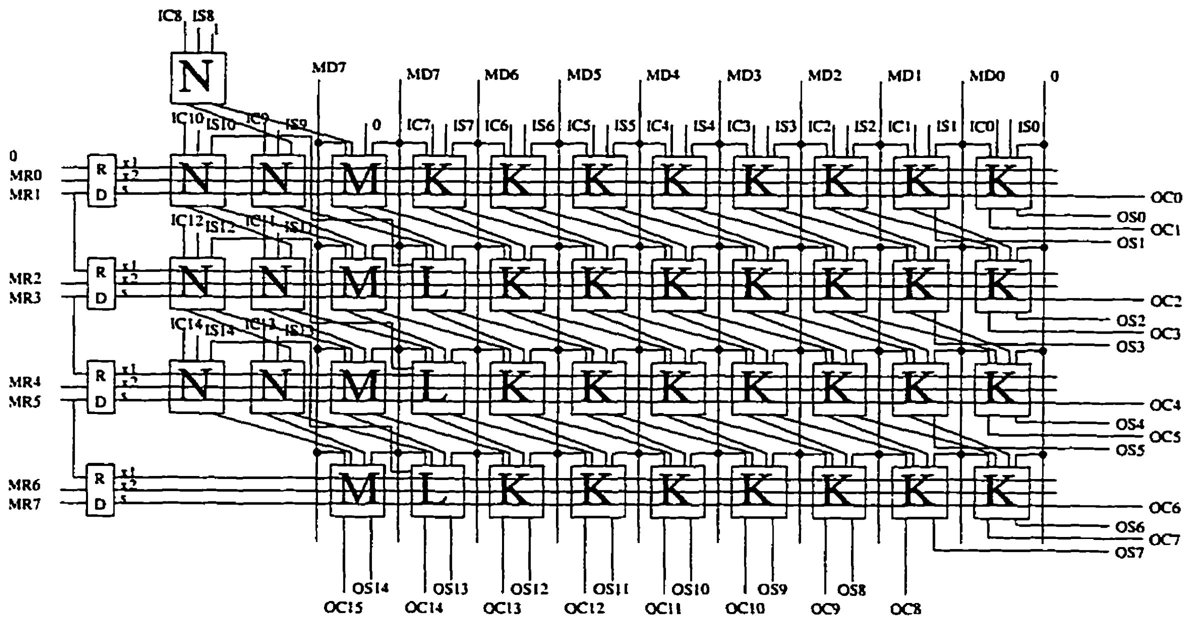


Figure 6.7: The Multiplier-Accumulator Array.

2. The array multiplier used is based on the modified Booth algorithm, and it uses the sign extension algorithm developed in section 6.2.1.

Figure 6.7, a modification of Figure 6.3, is the multiplier accumulator array. The multiplier and the multiplicand are in the two's complement representation, the accumulator input (AI) and output (AO) are in sum-carry representation. The array cells of Figure 6.7 are given in Figure 6.8.

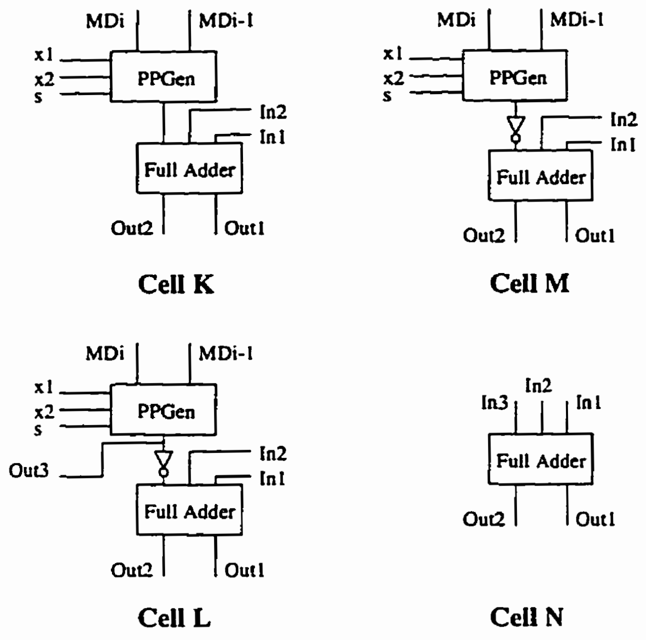


Figure 6.8: The array cells of the MAC array given in Figure 6.7.

The area, speed and power dissipation of the MAC array, given in Figure 6.7, are compared to that of a MAC with a separate multiplier/adder as shown in Figure 6.6. The basic components of each architecture are, the row-decoder (RD), the partial product generator (PPGen), the full adder (FA) and the half adder (HA). The multiplier has m bits, the multiplicand has n bits and the accumulator input and output have $m + n - 1$ bits. Table 6.2, gives the number of basic components required by the MAC array and the separate multiplier/adder MAC.

Notice that the MAC array has no half adders. However, one of the full adders could have been a half adder, but it was chosen to be a full adder to make the array regular.

For the separate multiplier/adder MAC, the adder has to add four operands, the sum and carry of the accumulator input and the sum and carry generated by the multiplier. A two level carry save adder is used to do this. One of the full

Table 6.2: Number of basic components for the MAC array and the separate multiplier/adder MAC.

Basic Component	MAC Array	Separate Multiplier/Adder MAC
RD	$\frac{m}{2}$	$\frac{m}{2}$
PPGens	$\frac{m(n+1)}{2}$	$\frac{m(n+1)}{2}$
FA	$\frac{m(n+1)}{2} + (m-1)$	$(n-1)\left(\frac{m}{2}-2\right) + 2(m+n-1) - 1$
HA	0	m + n - 2

adders of the separate multiplier/adder MAC could have been a half adder, but it was chosen to be a full adder for regularity.

Comparing the number of components used in the MAC array versus that used in the separate multiplier/adder MAC, it turns out that the latter uses $m+n-2$ extra half adders.

The critical path delay of the MAC array in terms of the basic components is given by:

$$D(\text{RD}) + D(\text{PPGen}) + \frac{m}{2}D(\text{FA})$$

$D(X)$ is the delay of the basic component X. X can be RD, PPGen, FA or HA. For the separate multiplier/adder MAC, the array multiplier of Figure 6.3 has a critical path delay of:

$$D(\text{RD}) + D(\text{PPGen}) + (\frac{m}{2} - 2)D(\text{FA}) + 2D(\text{HA})$$

The CSA has a delay of $2D(\text{FA})$. Hence, the entire critical path delay of the separate multiplier/adder MAC is:

$$D(\text{RD}) + D(\text{PPGen}) + \frac{m}{2}D(\text{FA}) + 2D(\text{HA})$$

which is greater than the delay of the MAC array by $2D(\text{HA})$.

Table 6.3 gives the area, delay and power dissipation/MHz of the basic components used in building the MAC. These values are based on CMOS standard cell libraries [150] [161] — [164]. The delay and power dissipation of the basic components designed in a $0.5\mu m$, 3.3 Volt CMOS technology were obtained through simulations. For the other CMOS technologies, the delay and power dissipation obtained in Table 6.3 are based on the data sheet parameters of each technology. Table 6.4 gives the W/L for the N-transistor and P-transistor of each CMOS technology.

The delay-power of the MAC array and the separate multiplier/adder MAC using different CMOS technologies is shown in Figure 6.9 and Figure 6.10 for a 10×10 bit and 20×20 bit multiplier respectively. The following observations can be made:

- The MAC array has a lower power-delay product than the separate multiplier/adder MAC of the same technology.
- The lower the resolution of the MAC, the greater the reduction of the delay-power product of the MAC array over the separate multiplier/adder MAC.

The MAC array reduces the power dissipation, delay and area relative to the separate multiplier/adder MAC. The relative reduction of the power dissipation, delay and area for a $0.5\mu m$, 3.3 Volt CMOS technology is shown in Figures 6.11, 6.12 and 6.13 respectively.

Table 6.3: Area in μm^2, delay in ns and power in μW/MHz for the basic components of the MAC architecture in different CMOS standard cell technologies.

Library	Component	Area μm^2	Delay ns	Power/MHz μW/MHz
0.6μm 5 Volt	RD	3873	1.41	12.0
	PPGen	4401	2.48	14.0
	FA	3345	1.99	6.0
	HA	1936	0.80	4.0
0.6μm 3.3 Volt	RD	3873	2.11	5.3
	PPGen	4401	3.53	6.2
	FA	3345	2.84	2.7
	HA	1936	1.19	1.8
0.5μm 5 Volt	RD	1728	1.13	7.8
	PPGen	1964	1.69	9.1
	FA	1414	1.58	3.9
	HA	864	0.68	2.7
0.5μm 3.3 Volt	RD	1728	1.14	3.4
	PPGen	1964	1.68	4.0
	FA	1414	1.57	1.7
	HA	864	0.72	1.2
0.35μm 5 Volt	RD	1093	0.77	7.4
	PPGen	1243	1.07	6.3
	FA	1044	1.08	3.2
	HA	497	0.49	2.1
0.35μm 3.3 Volt	RD	1088	0.74	3.4
	PPGen	1236	1.11	2.9
	FA	1039	0.90	1.5
	HA	495	0.48	1.0

Table 6.4: W/L for the different CMOS technologies.

CMOS Technology	N-Transistor W/L	P-Transistor W/L
0.6μm 5 Volt	12.8/0.7	18.1/0.8
0.6μm 3.3 Volt	12.8/0.7	18.1/0.8
0.5μm 5 Volt	7.8/0.6	10.6/0.6
0.5μm 3.3 Volt	7.8/0.5	10.6/0.6
0.35μm 5 Volt	6.18/0.48	8.42/0.48
0.35μm 3.3 Volt	6.12/0.36	8.38/0.4

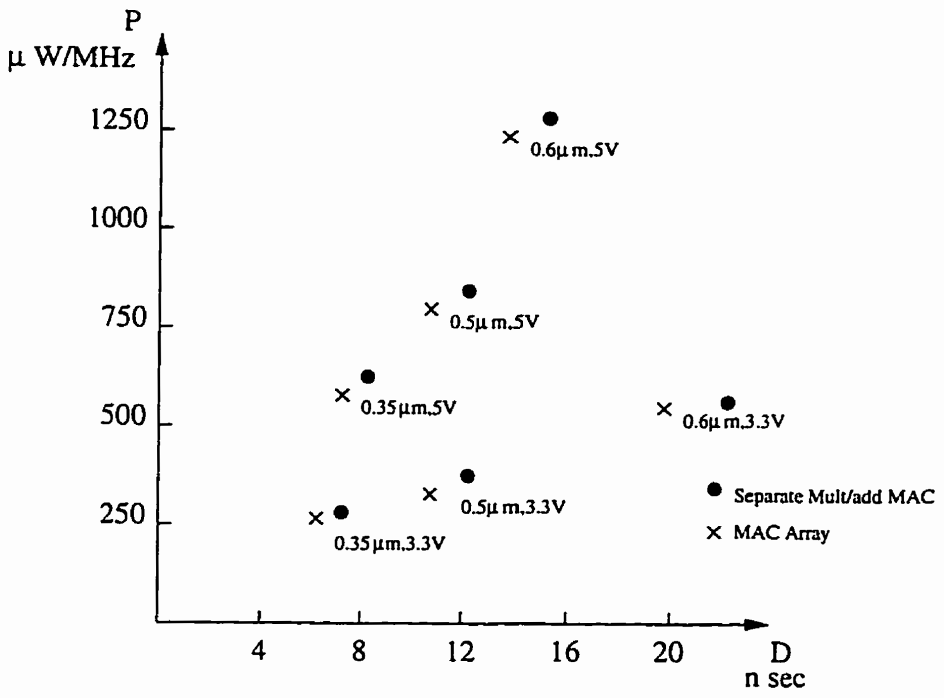


Figure 6.9: The delay-power relationship of the MAC array and the separate multiplier/adder MAC having a 10 × 10 bit multiplier implemented in different CMOS technologies.

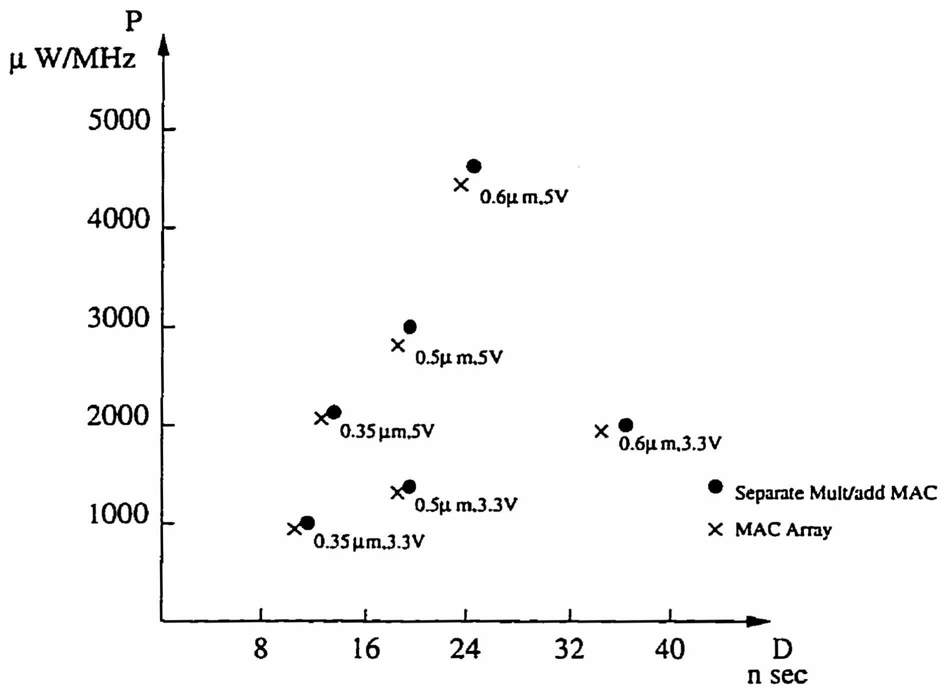


Figure 6.10: The delay-power relationship of the MAC array and the separate multiplier/adder MAC having a 20 × 20 bit multiplier implemented in different CMOS technologies.

For a $0.5\mu m$, 3.3 Volt CMOS, at a resolution of 20 bits, the saving in power dissipation when using the MAC array is about 3.6%. The saving in area is about 4.5%. While the increase in speed is about 7.7%. The increase in speed can lead to further reduction in the power dissipation [1] by reducing the voltage to maintain the same throughput. To maintain the same throughput the voltage is reduced by 7.2% [150]. This corresponds to a further 13.9% reduction in power dissipation. Hence, at a 20 bit resolution, the total reduction in power dissipation, achieved using the MAC array, is 17%.

At a resolution of 10 bits, the saving in power dissipation when using the MAC array is about 6.5%. The saving in area is about 8.2%. While the increase in speed is about 13.5%. The increase in speed can lead to further reduction in the power dissipation [1] by reducing the voltage to maintain the same throughput. To maintain the same throughput the voltage is reduced by 11.9% [150]. This corresponds to a further 22.4% reduction in power dissipation. Hence, at a 10 bit resolution, the total reduction in power dissipation, achieved by using the MAC array, is 27.4%.

6.3.1 Analysis of the Computational Efficiency of the MAC Array

The difference between the MAC array and the separate multiplier/adder MAC, is that the addition operation has been interleaved into the multiplier array of the former. This provides an opportunity to merge some blocks together. The multiplier array, Figures 6.3 and 6.4, use half adders in some of its cells. Whereas, the MAC array, Figures 6.7 and 6.8, uses only full adders. This is where the elimination of redundant computations occur.

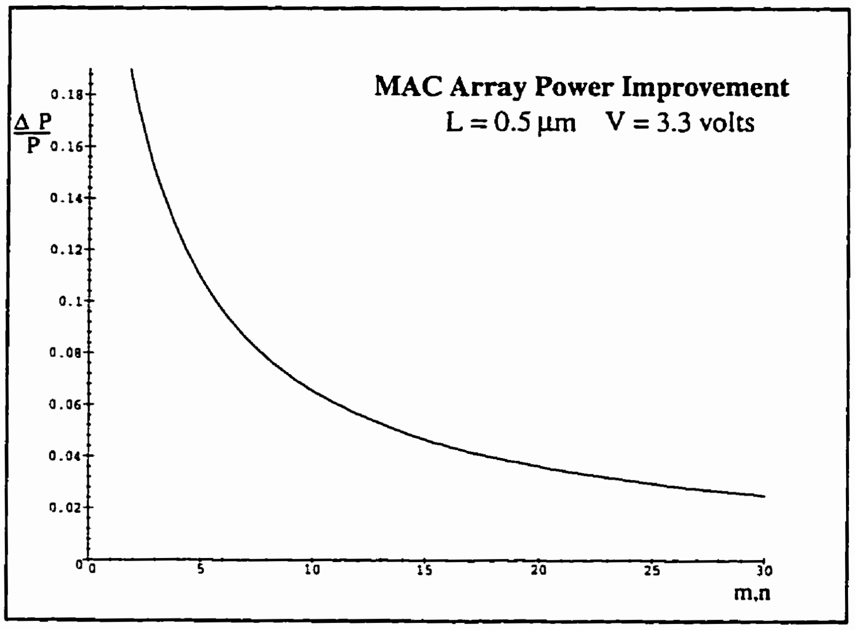


Figure 6.11: The relative decrease in the power dissipation of the MAC array of Figure 6.7 over that of the separate multiplier/adder MAC. Both designed in a $0.5\mu m$, 3.3 Volt CMOS technology.

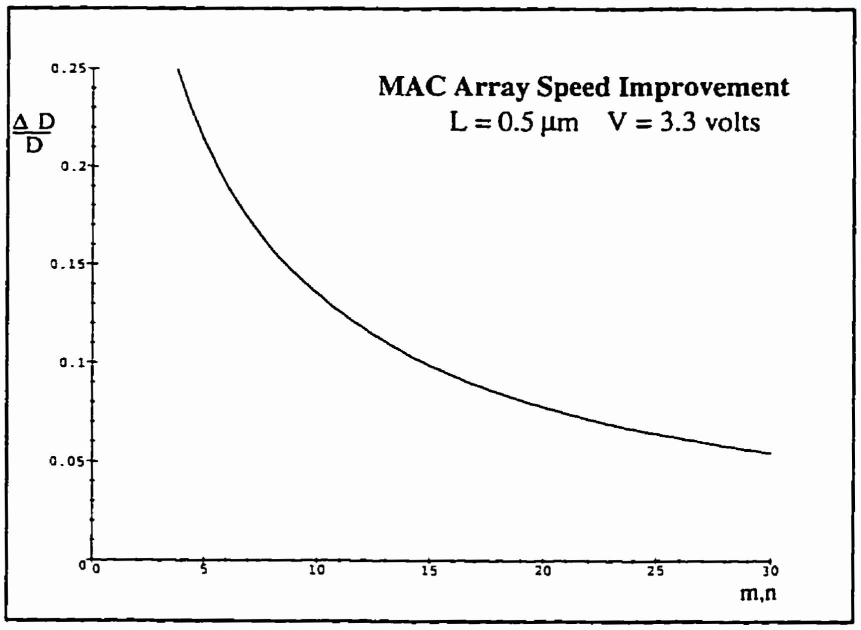


Figure 6.12: The relative decrease in the delay of the MAC array of Figure 6.7 over that of the separate multiplier/adder MAC. Both designed in a $0.5\mu m$, 3.3 Volt CMOS technology.

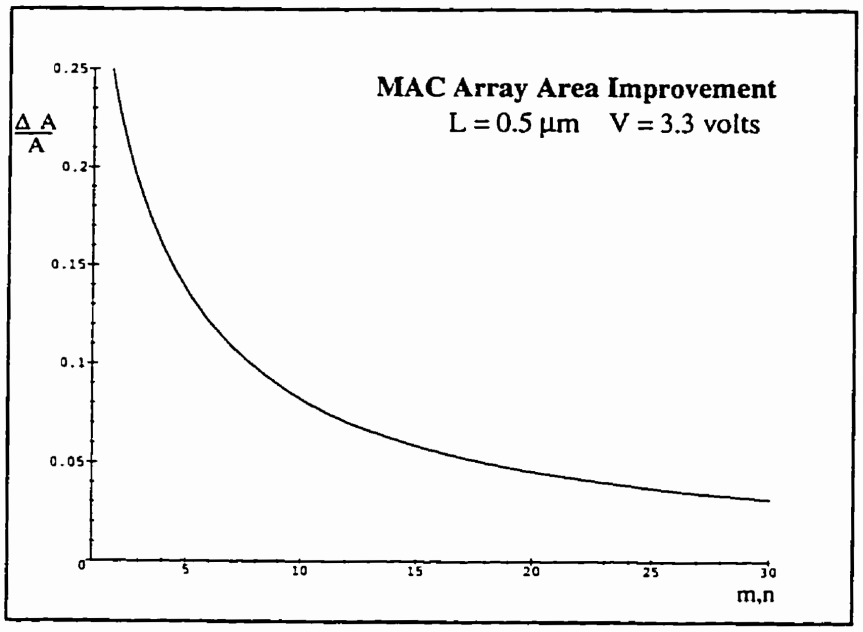


Figure 6.13: The relative decrease in the area of the MAC array of Figure 6.7 over that of the separate multiplier/adder MAC. Both designed in a $0.5\mu m$, 3.3 Volt CMOS technology.

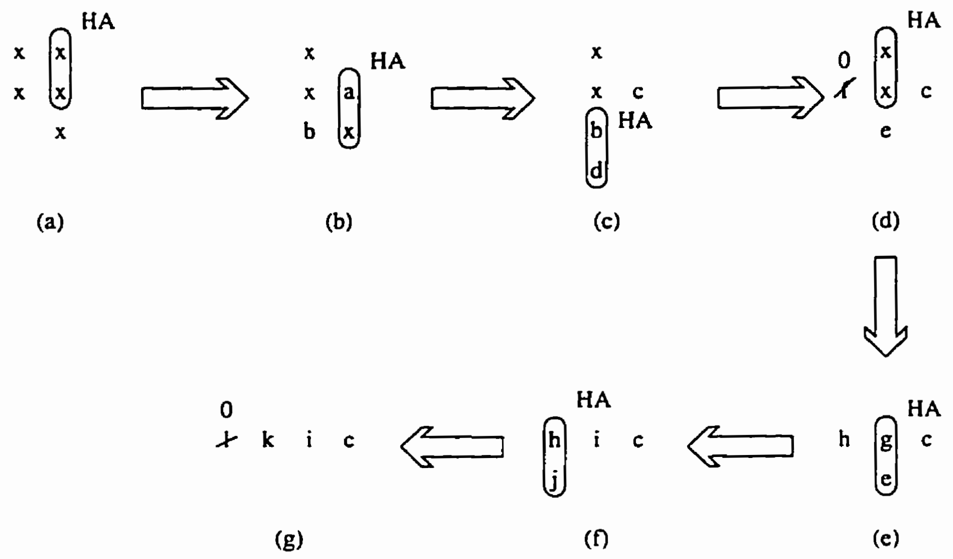


Figure 6.14: Adding 5 bits using half adders only. This requires 6 half adders.

Half adders have two inputs and two outputs. Hence the number of bits that remain to be added after the half adder remains the same. Consider the scenario shown in Figure 6.14, where 5 bits need to be added. After the application of the first HA, the number of bits that remain to be added is still 5 bits. However, there is redundancy in this case (Figure 6.14.b), because a and b cannot be 1 simultaneously. After the second HA, there is still 5 bits to be added. Again, there is redundancy in that c and d cannot be 1 simultaneously. After the third HA, there is only 4 bits. The carry out, f, of this HA is always zero.

To reduce the number of bits from 5 bits to 4 bits it took three half adders. This could have been accomplished using one full adder as shown in Figure 6.15. Similarly, to go from 4 bits, as shown in Figure6.14.d, to 3 bits as shown in Figure 6.14.g, three half adders are required.

It takes three half adders to add three bits together, an operation that can be done using one full adder. Figure 6.16 shows the implementation of the full adder

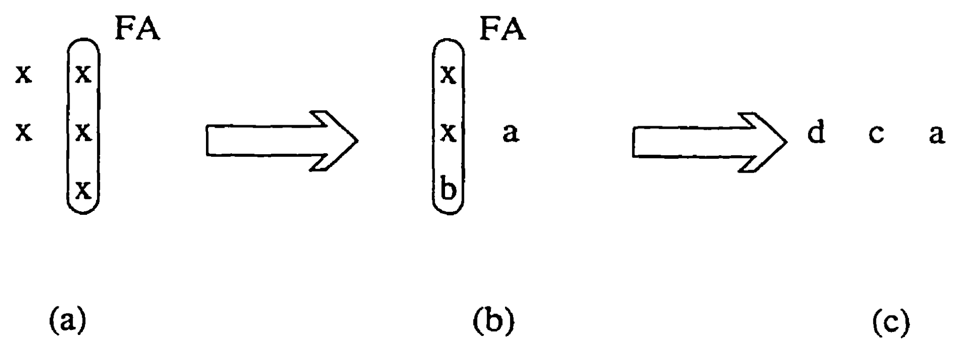


Figure 6.15: Adding 5 bits using full adders. This requires 2 full adders.

using three half adders.

Consider now the scenario shown in Figure 6.17, where the bits x and z are to be added together. Two scenarios are considered. In the first, the bits x are added using a half adders. The resulting bits y are then added to bits z using full adders and half adders. The output is in the sum carry representation. This implementation requires a total of 9 full adders and 6 half adders.

In the second implementation, the x bits and the z bits are added directly using full adders and half adders to give the same sum carry representation as that generated by the first implementation. This implementation requires only 9 full adders and 2 half adders. Thus, achieving a saving of 4 half adders.

When the adder is interleaved into the multiplier array, the half adders are merged into the full adders required for the addition process. Hence, eliminating redundant computations and achieving a saving in the power dissipation.

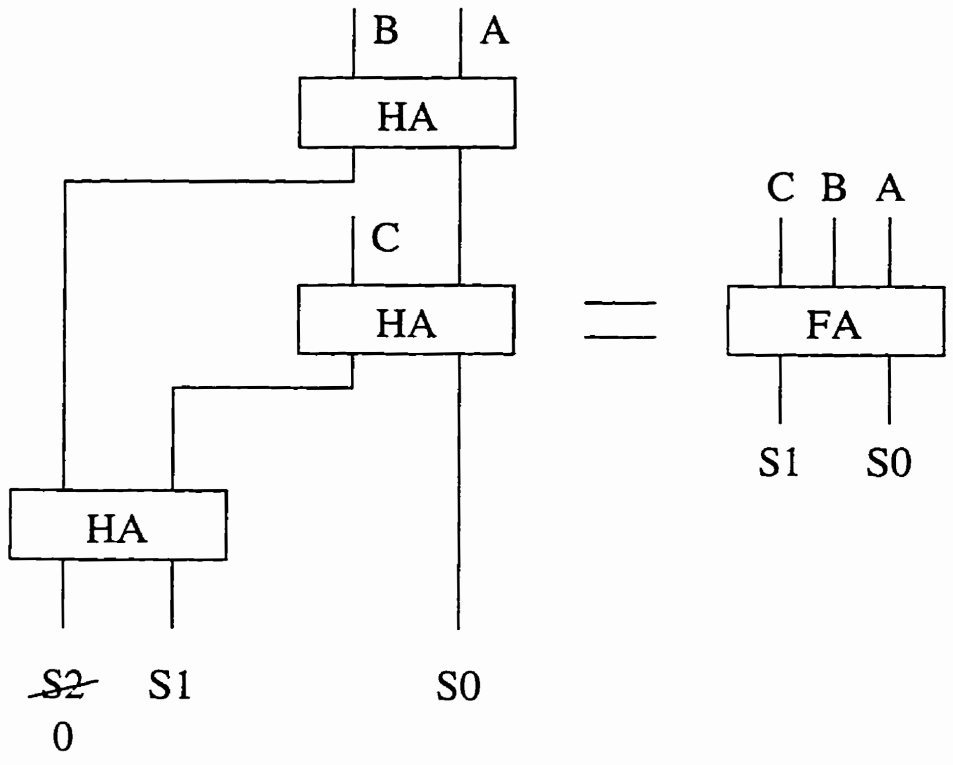


Figure 6.16: Merging three half adders into a single full adder.

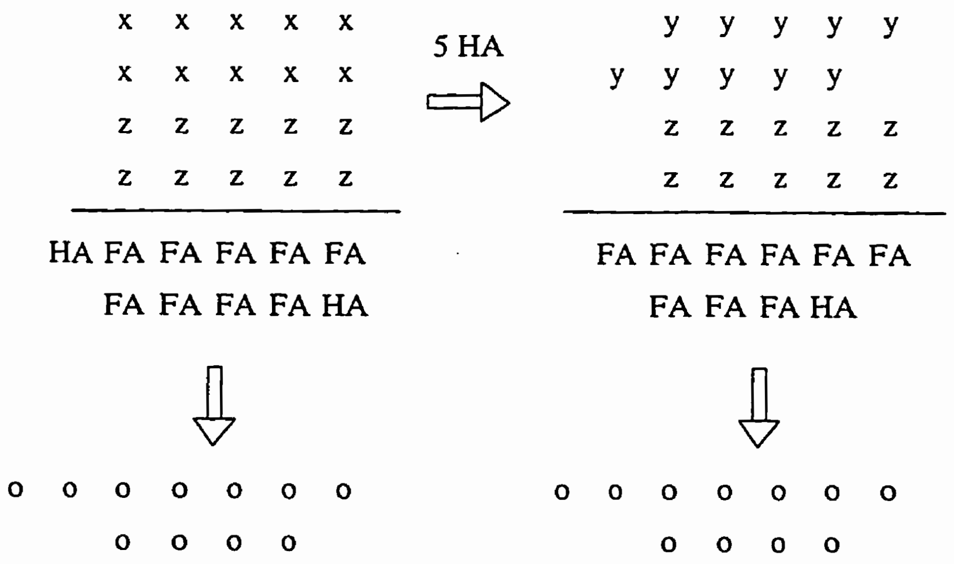


Figure 6.17: Binary number addition using half adders and full adders.

6.4 Programmable MAC Array

To reduce the power dissipation of the MAC when a lower resolution is sufficient the unused cells of the MAC array – those corresponding to the least significant bits – are deactivated, making the resolution of the MAC array programmable [158]. This eliminates irrelevant computations. Irrelevant computations are those that affect the output of the unit under consideration, the MAC array in this case, but whose elimination doesn't affect the over all SNR performance of the system because the SNR is limited by some other system considerations, or by another unit in the system.

For the MAC array of Figure 6.7, the multiplicand comes from the bottom, and the multiplier comes from the left side. When the resolution of the multiplicand is reduced, the cells of the right-hand-side columns are deactivated. When the resolution of the multiplier is reduced, the cells of the top rows are deactivated. When a certain cell is deactivated, the inputs are bypassed to the output. This is done through the use of the bypass logic. The bypass logic is an overhead in the programmable MAC array.

The carry output of each cell shifts one column only, while the sum output shifts two columns. This fact is taken into account when designing the bypass logic. The dotted line in Figure 6.18 represents the bypass path for a deactivated cell.

Extra logic is required for the bypass. This extra logic leads to larger area. It also increases the power dissipation when the full resolution of the MAC is used. At the lower resolution the power dissipation decreases. The amount of power saving depends on the percentage of time the MAC spends at each resolution.

In addition to the bypass logic, extra logic is also required to prevent any signal changes on the input lines from reaching the deactivated cells. The blocking logic

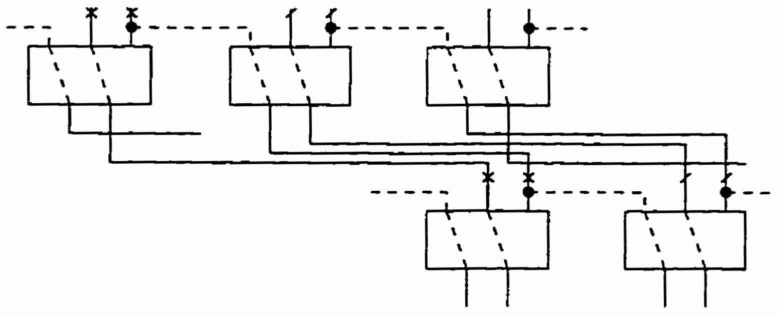


Figure 6.18: The bypass path for a deactivated cell.

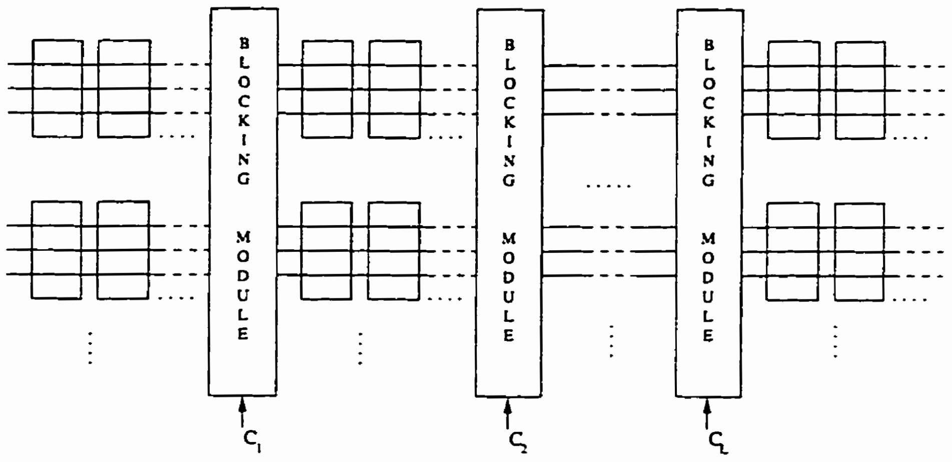


Figure 6.19: The blocking modules for the recoded multiplier signals.

for the recoded multiplier signals is shown in Figure 6.19. The resolution of the multiplicand determines the value of the control signals C_1, C_2 ... C_L. When the multiplicand has minimum resolution, n_n, the recoded multiplier bits are blocked at the left most blocking module. When the multiplicand has maximum resolution, n_x, the recoded multiplier bits pass through all the blocking modules. Similar blocking logic is required for the multiplicand.

Assume the number of multiplicand bits is given by:

$$n = n_n + \frac{l}{L}(n_x - n_n) \tag{6.10}$$

where,

$L + 1$ is the number of resolutions the multiplicand can take.

n_x is the maximum resolution of the multiplicand.

n_n is the minimum resolution of the multiplicand.

$l = 0, 1, \ldots, L.$

Similarly, the number of multiplier bits is given by:

$$m = m_n + \frac{k}{K}(m_x - m_n) \tag{6.11}$$

where,

$K + 1$ is the number of resolutions the multiplier can take.

m_x is the maximum resolution of the multiplier.

m_n is the minimum resolution of the multiplier.

$k = 0, 1, \ldots, K.$

In general, the power dissipation of the programmable MAC array was found to be given by:

$$\text{Power} = P_2 nm + P_1 m + P_0 + P_3\left[m_x(3 + n_x) - m_n(3 + n_n)\right] + P_4 nk + P_5 ml \tag{6.12}$$

where,

Table 6.5: Power coefficients, in μW/MHz, of the programmable MAC used in Equation 6.12, for a $0.5\mu m$, 3.3 Volt CMOS technology.

CMOS Technology	P_0	P_1	P_2	P_3	P_4	P_5
$0.5\mu m$ 3.3V	-1.7	6.25	2.85	0.45	0.45	0.7

P_0 — P_5 are technology dependent coefficients.

The first three terms of Equation 6.12 are the power dissipation terms of a regular MAC array. The fourth term is due to the bypass logic. The last two terms are due to the input blocking logic. The values of P_0 — P_5 for a 0.5 μm, 3.3 Volt CMOS technology are given in Table 6.5. Equation 6.12 along with the power coefficients of Table 6.5 were initially derived analytically. Later they were verified through simulations.

The amount of power saving is dependent on the percentage of time the multiplicand and the multiplier are required to operate at each resolution. The multiplier can take one of $K+1$ resolutions. Assume the lowest resolution m_n has a probability α. The remaining resolutions:

$$m_n + \frac{m_x - m_n}{K} \times 1, m_n + \frac{m_x - m_n}{K} \times 2, \ldots, m_x \tag{6.13}$$

are equi-probable and have a probability of:

$$\frac{1-\alpha}{K} \tag{6.14}$$

Similarly, the multiplicand can take one of $L+1$ resolutions. Assume the lowest resolution n_n has a probability β. The remaining resolutions:

$$n_n + \frac{n_x - n_n}{L} \times 1, n_n + \frac{n_x - n_n}{L} \times 2, \ldots, n_x \tag{6.15}$$

are equi-probable and have a probability of:

$$\frac{1-\beta}{L} \tag{6.16}$$

The MAC array designed is required to have a maximum resolution of 21 bits for the multiplicand and 20 bits for the multiplier. The power dissipation of the nonprogrammable MAC array, implemented in a $0.5\mu m$, 3.3 Volt CMOS technology, is $1320\mu W$/MHz. To reduce the power dissipation when a lower resolution is sufficient, the MAC array is designed to have a programmable resolution for both the multiplicand and the multiplier. The resolution of the multiplicand can be; 21, 17, or 13 bits. The resolution of the multiplier can be; 20, 16, or 12 bits.

The power dissipation ratio between that of the programmable MAC array and that of the non-programmable MAC array depends on the probability distribution parameters α and β. This power dissipation ratio is shown in Figure 6.20. A power saving of up to 50% can be achieved.

To further investigate the effect of the probability distribution on the power saving that can be achieved using the programmable MAC array, assume the following probability distributions:

$$\text{Prob(Multiplicand is 13 bits)} = \text{Prob(Multiplier is 12 bits)} = \gamma \tag{6.17}$$

$$\text{Prob(Multiplicand is 17 bits)} = \text{Prob(Multiplier is 16 bits)} = \delta \tag{6.18}$$

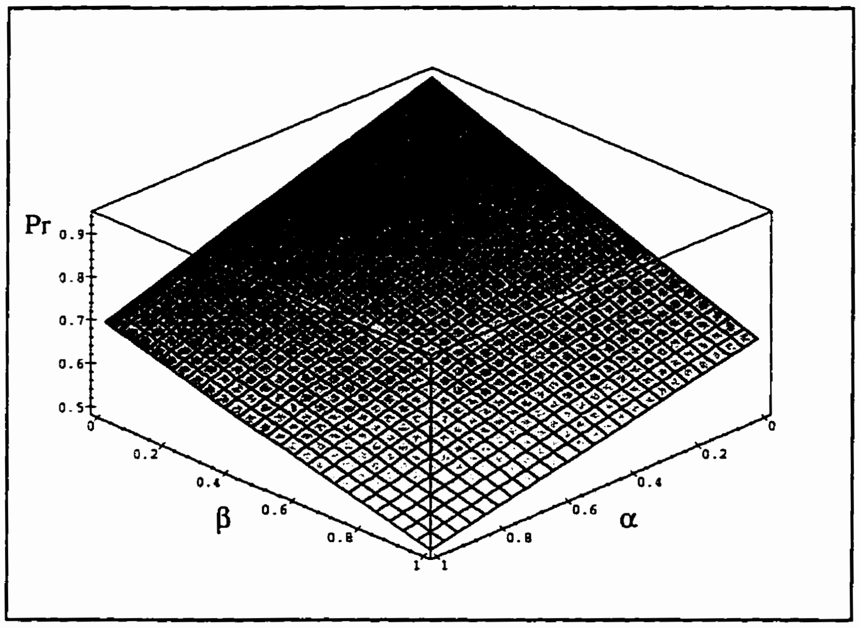


Figure 6.20: The power dissipation ratio between the programmable MAC array having $K = 2$, $L = 2$ and the non-programmable MAC array. α and β are the resolution probability distribution parameters, from Equations 6.13, 6.14, 6.15 and 6.16.

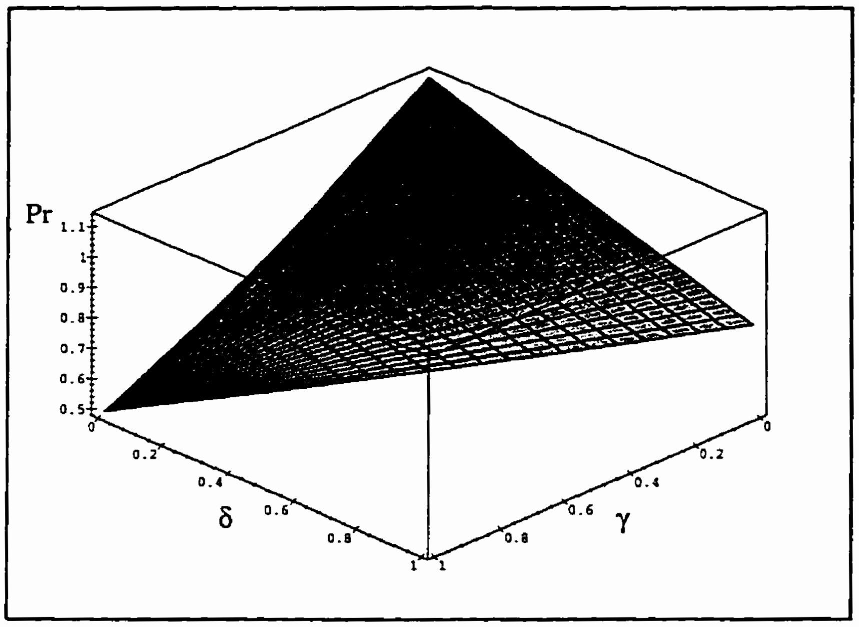


Figure 6.21: The power dissipation ratio between the programmable MAC array having $K = 2$, $L = 2$ and the non-programmable MAC array. γ and δ are the resolution probability distribution parameters, from Equations 6.17, 6.18 and 6.19.

$$\text{Prob(Multiplicand is 21 bits)} = \text{Prob(Multiplier is 20 bits)} = 1 - \gamma - \delta \qquad (6.19)$$

The relatived power dissipation for different values of γ and δ, as defined by Equations 6.17 and 6.18, is given in Figure 6.21. This power dissipation ratio varies between 0.5 (when $\gamma = 1$ and $\delta = 0$), to 1.1 (when $\gamma = \delta = 0$).

6.5 VLSI Implementation of the Programmable MAC Array

The programmable MAC array has been designed with $L = 2$ and $K = 2$. The allowed resolutions for the multiplier are; 20, 16, and 12 bits. The allowed resolutions for the multiplicand are; 21, 17 and 13 bits.

The programmable MAC array has been designed in a $0.5\mu m$, 3.3 Volt CMOS technology. It has over 17000 transistors. It occupies an area $1815\mu m \times 1542\mu m$. Simulation results show that the programmable MAC array can run at a speed up to 42 MHz. Simulation results also show that the average power dissipation of the programmable MAC array, assuming that all resolutions are equi-probable is 1.0 mW/MHz. This is 24% lower than the power dissipation of the corresponding non-programmable MAC array. Figure 6.22 shows the VLSI layout of the programmable MAC array.

6.6 Chapter Summary

A multiplier-accumulator array has been designed in a 0.5 μm, 3.3 Volt CMOS technology. The multiplier of the MAC array is based on the modified Booth algorithm. The accumulator's input and output are in the sum-carry representation.

To reduce the power dissipation, the adder was interleaved in the multiplier array. This eliminates redundant computations. The MAC array achieved up to 17% saving in power dissipation for a 20×20 array, and up to 27.4% saving in power dissipation for a 10×10 array.

To further reduce the power dissipation a block deactivation architecture was

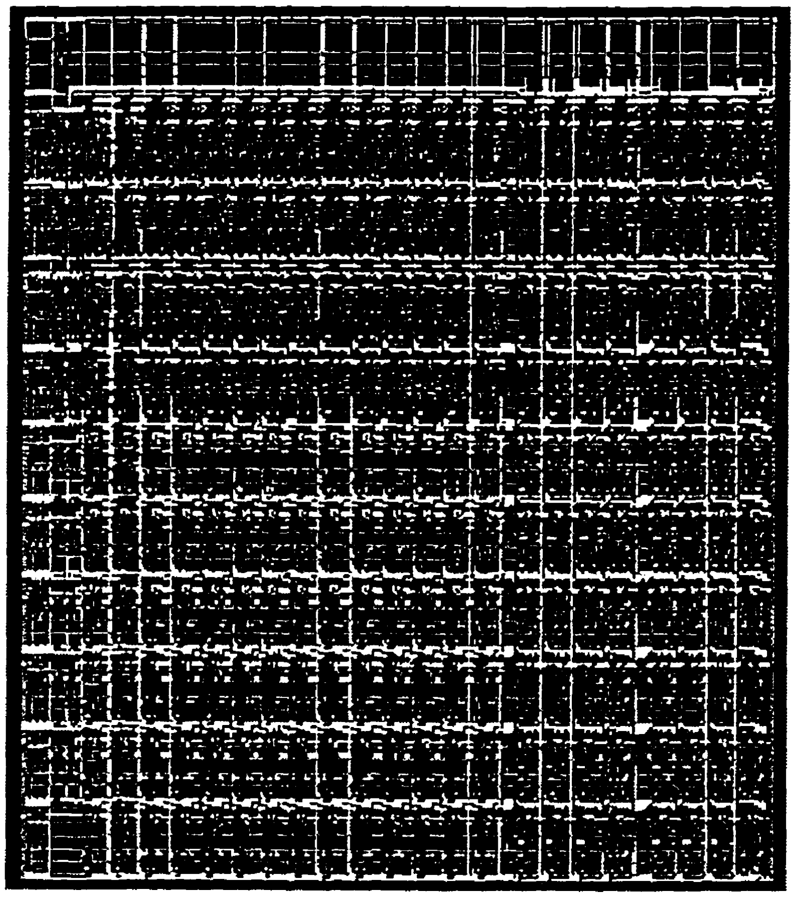


Figure 6.22: The VLSI layout of a programmable MAC array designed in a $0.5\mu m$, 3.3 Volt CMOS technology. The total area of the MAC array is $1815\mu m \times 1542\mu m$.

developed, where the cells corresponding to the least significant bits are deactivated when a smaller resolution is sufficient. This eliminates redundant computations. The power dissipation saving achieved using this architecture can be up to 50%. However, a more conservative estimate, when all the resolutions are equi-probable, puts the power dissipation saving at 24%.

Chapter 7

Digital Channel Selection

7.1 Introduction

In software radio, the IF as well as the baseband functions are done digitally. These functions require high DSP horsepower, which can be as high as 10 GFLOPS/s [12]. This in turn leads to high power dissipation.

In software radio, the digitized signal represents a block of channels. The channel to be selected is then filtered digitally. In this chapter, I examine ways to reduce the computational complexity and hence the power dissipation of the digital channel selection algorithm. The salient low-power features of this digital channel selection algorithm are [165]:

1. The pre-filter multiplier has been eliminated.
2. The channel selection filtering is performed in stages. During each stage the sampled signal is decimated by a factor of 2.
3. The multipliers in the filters have been replaced by adders.

In this chapter, the effect each of these has on the power dissipation of the digital channel selection algorithm is demonstrated.

In section 7.2, the conventional channel selection algorithm is considered, and its computational power dissipation is computed. The effect of performing channel selection in stages on the computational power dissipation is also considered in this section.

In section 7.3, an algorithm for eliminating the pre-filter multiplier is considered. However, this algorithm requires sharp filtering stages. In section 7.4, the algorithm of section 7.3 is further developed to relax the filter sharpness requirement. The implementation of this algorithm is considered, and its computational power dissipation is computed.

Finally, in section 7.5, the computational power dissipation of the novel digital channel selection algorithm is compared to that of other digital channel selection algorithms.

7.2 The Conventional Channel Selection Algorithm

In software radio, the digitized signal represents a block of adjacent channels, having a spectrum as shown in Figure 7.1. This signal is called the composite digital signal, it is denoted by C. C is a complex discrete-time baseband signal, given by;

$$C(nT_s) = I(nT_s) + jQ(nT_s) \tag{7.1}$$

T_s is the sampling frequency. C is composed of $M = 2^m$ frequency division mul-

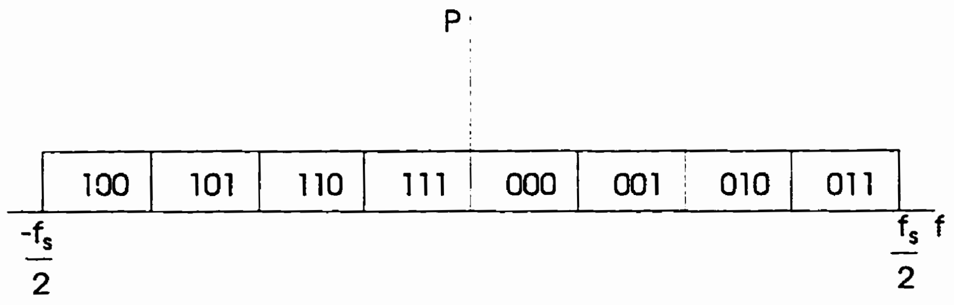


Figure 7.1: The frequency spectrum of a baseband signal consisting of eight channels.

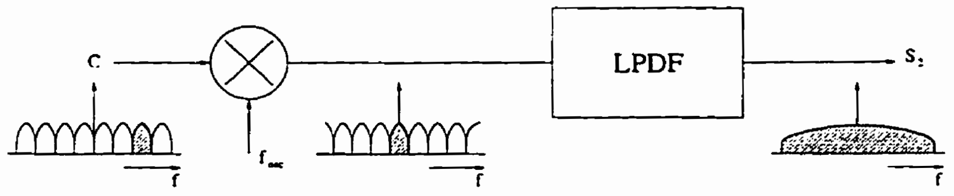


Figure 7.2: The conventional channel selection algorithm.

tiplexed channels, each denoted by S_n. Each of these channels occupies a frequency band df. The center frequency of channel S_n is given by:

$$f_n = (n + \frac{1}{2})\text{df} \tag{7.2}$$

Where, $n = -\frac{M}{2}, \ldots, \frac{M}{2} - 1$. n can be represented in a two's complement binary format. Figure 7.1 shows the binary channel-numbering for the case $M = 8$.

To select channel S_n, the composite signal is multiplied by a sinusoidal signal of frequency $-f_n$. This shifts channel S_n to be centered around the zero frequency. A lowpass decimation filter (LPDF) selects channel S_n and rejects the remaining channels. This channel selection process is shown in Figure 7.2.

The algorithm of Figure 7.2 requires:

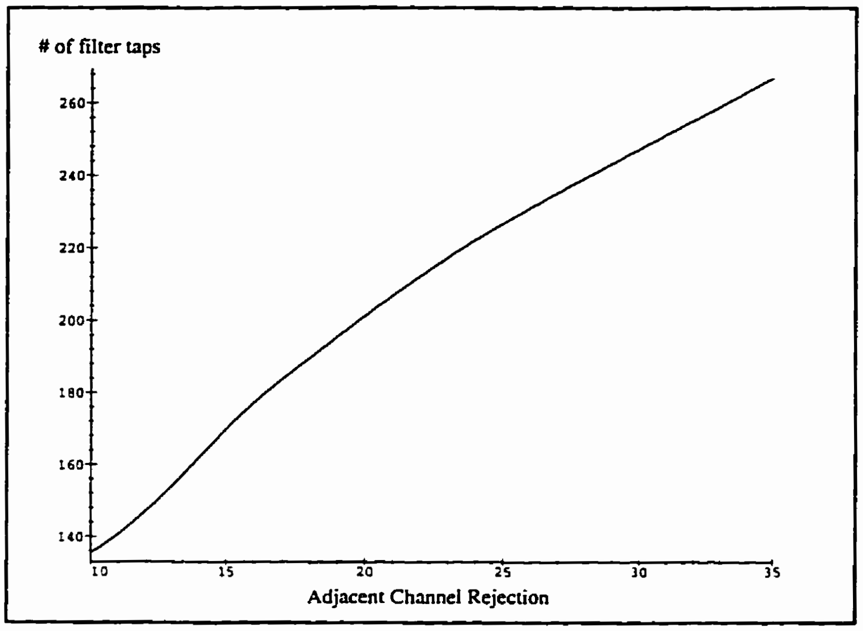


Figure 7.3: Number of filter taps vs. adjacent channel rejection, for the digital selection of 1 out of 32 channels.

1. A frequency synthesizer, to generate the selected channel frequency.
2. A pre-filter multiplier. This multiplier operates at the high sampling rate, and its operands have high resolution.
3. A sharp LPDF which has a large number of taps to provide sufficient adjacent channel rejection. The number of taps required to achieve a certain channel rejection ratio is shown in Figure 7.3. However, using a polyphase filter structure [120], the LPDF operates at the lower output rate.

In the DAMPS standard, the adjacent channel selectivity is 13 dB [22]. DQPSK is the modulation scheme used in IS-136, to achieve a bit error rate of 10^{-2}, $\frac{E_b}{N_o}$

Table 7.1: Computational power dissipation for the conventional digital channel selection algorithm shown in Figure 7.2, and designed in a $0.5\mu m$, 3.3 Volt CMOS technology.

Criteria	Pre-Filter Multiplier	LP Decimation Filter	Total
Speed	960 KHz	30 KHz	30 KHz
Operations	4 MULT(20×20) 2 ADD(20)	 216MULT(20×16) 428 ADD(20)	128 MULT(20×20) 216MULT(20×16) 492 ADD(20)
Power	5.09 mW	7.45 mW	12.55 mW

should be 6 dB [166], assuming that 50% of the noise is due to adjacent channel interference than the filter is required to achieve a 22 dB adjacent channel rejection. From Figure 7.3, this requires a 215 tap filter to digitally select one out of 32 channels. Through simulation, it was found that the filter coefficients should have a resolution of 16 bits to achieve the sufficient channel rejection required for non-adjacent channels [22].

The computational power dissipation for a conventional digital channel selection algorithm used in the selection of one out of 32 30 KHz channels, as specified by the DAMPS standard [22], is given in Table 7.1. The sample rate of the composite digital signal is 960 KHz (complex sample rate). The sample rate at the output of the LPDF of Figure 7.2 is 30 KHz (complex sample rate). The values given in Table 7.1 are for a $0.5\mu m$, 3.3 Volt CMOS technology [150]. The lowpass decimation filter is a linear phase filter, this reduces the number of multiplications by half.

To reduce the lowpass filter sharpness and hence reduce the number of taps and lower the filter computational power dissipation, the lowpass decimation filter is implemented as a cascade of lowpass decimation filters. Each filter decimates the sampled signal by a factor of 2. The number of filtering stages required is $\log_2 M$, for a filter selecting one out of M channels.

Assuming $M = 32$, 5 cascaded filtering stages are required. The first three stages don't require sharp filters, hence a Sinc decimator is used. For the last two stages, a LPF is used. Table 7.2 gives the sufficient image channel rejection each filtering stage is required to achieve in order to obtain a BER of 10^{-2} for the DAMPS standard. Also given in Table 7.2 is the order of the Sinc decimator and the number of the LPF taps necessary to achieve the required image channel rejection.

The number of operations required per stage per output, power dissipation per stage and the total computational power dissipation including that of the pre-filter multiplier are also given in Table 7.2. The values given in this table are for a $0.5\mu m$, 3.3 Volt CMOS technology.

The total computational power dissipation given in Table 7.2 is that of the filter and the pre-filter multiplier. The power saving achieved is 45%, over the conventional channel selection algorithm of Table 7.1. However, now the dominate source of computational power dissipation is that of the pre-filter multiplier which is 5.09 mW. This represents 75% of the total computational power dissipation. In the next section, a digital channel selection algorithm that eliminates the pre-filter multiplier is examined.

Table 7.2: Computational power dissipation for a lowpass decimation filter implemented as cascaded stages in a $0.5\mu m$, 3.3 Volt CMOS technology.

Criteria	1st stage	2nd stage	3rd stage	4th stage	5th stage
Image rejection	64 dB	64 dB	64 dB	51 dB	22 dB
Order/# of tapes	3	4	5	13	32
Speed	480 KHz	240 KHz	120 KHz	60 KHz	30 KHz
Operations	8 ADD(20)	10 ADD(20)	12 ADD(20)	14MULT(20×12) 24 ADD(20)	32MULT(20×9) 62 ADD(20)
Power dissipation	0.192 mW	0.120 mW	0.072 mW	0.744 mW	0.688 mW
Total power dissipation	6.91 mW				

7.3 Pre-Filter Multiplier Elimination

According to Table 7.2, the elimination of the pre-filter multiplier achieves 75% saving in the computational power dissipation. To achieve this, filtering is performed in stages. During each filtering stage half of the channels are eliminated. Thus, the number of filtering stages is $\log_2 M$.

The channel selection in this algorithm [165] depends on the use of highpass or lowpass filters to select the desired channel. Selecting one channel out of 2^m channels requires m filtering stages. The first $m-1$ stages are composed of a highpass/lowpass filter and a factor two down-sampler. In the last stage we do frequency shifting followed by lowpass filtering.

The selection of a highpass filter or a lowpass filter depends on the location of the channel to be selected. If that channel falls in one of the bands $[-\frac{f_s}{2}, -\frac{f_s}{4}]$ or

$[\frac{f_s}{4}, \frac{f_s}{2}]$ a highpass filter is used to select the desired channel. If the channel to be selected falls in the band $[-\frac{f_s}{4}, \frac{f_s}{4}]$ a lowpass filter is used.

Channel Selection Example

Figure 7.4 is an example showing the selection of one channel out of 16 channels. The channel to be selected is channel 6 (0110). Since the number of channels is halfed after each filtering stage, the number of bits required in numbering the channels is reduced by one bit after each stage.

The selection process is performed as follows. Channel 6 (the desired channel) falls in the highpass region. A highpass filter is used, the highpass filter selects channels -8, -7, -6, -5, 4, 5, 6, and 7. The highpass filter is followed by a factor 2 down-sampler, leading to the second frequency spectrum in Figure 7.4. Notice that, only three bits are required to number the remaining channels shown in the second frequency spectrum of Figure 7.4. The most significant bit has been removed. Furthermore, notice that because of the highpass filtering channel 6 has moved from the positive frequency region to the negative one.

Channel 6 now lies in the lowpass region $[-\frac{f_s}{4}, \frac{f_s}{4}]$. A lowpass filter is used for selecting channels -8, -7, 6 and 7. The lowpass filter is followed by a factor 2 down-sampler. Again the most significant bit is removed leaving only two bits. A highpass filter followed by a factor 2 down-sampler is used to select channels -7 and 6.

Channel -7 is the image of channel 6. The final selection process involves a frequency shift and lowpass filtering. The frequency shift is performed by multiplying the received signal by $e^{-j\frac{\pi}{2}n}$, n is the discrete time.

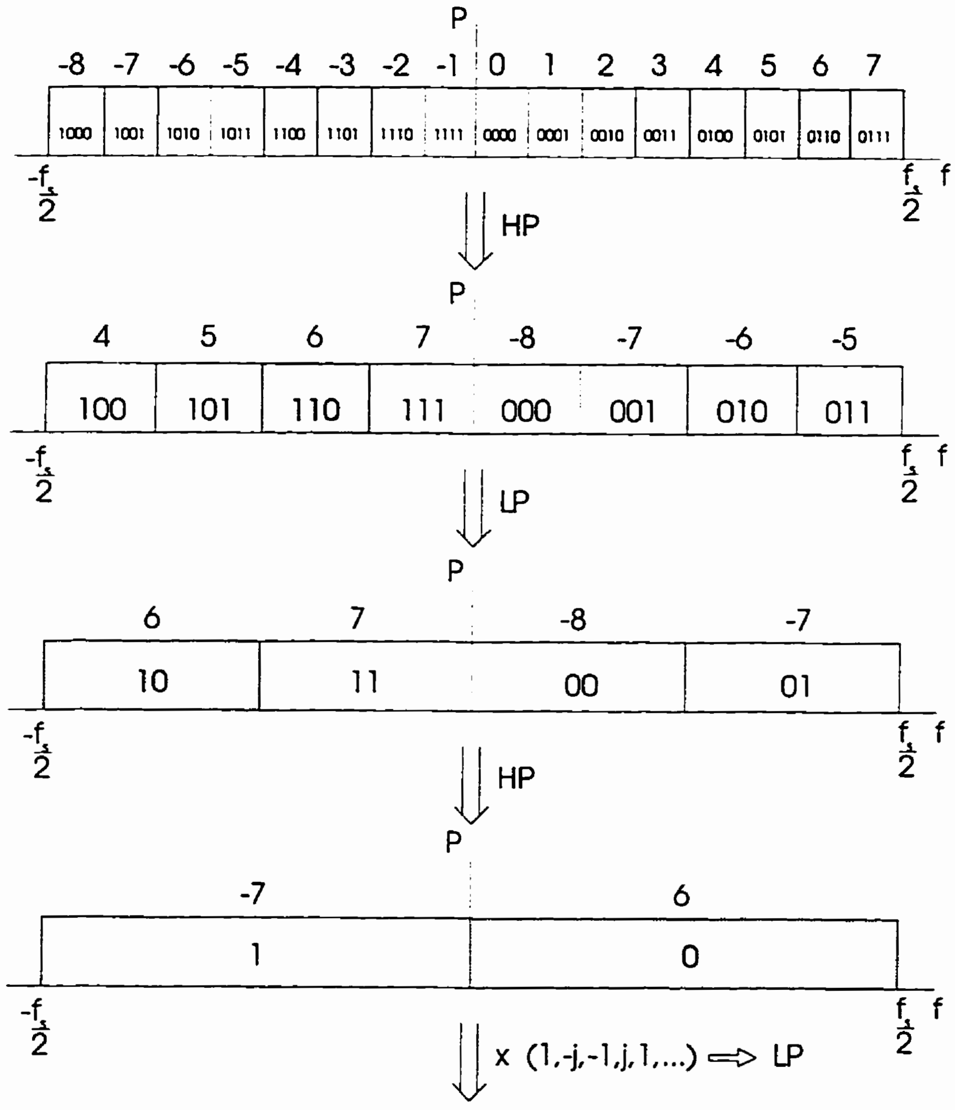


Figure 7.4: Channel selection example using lowpass and highpass filters only.

7.3.1 The Channel Selection Algorithm

The number of stages required to select one channel out of 2^m channels is m stages. During the first $m-1$ stages a lowpass filter or a highpass filter is used followed by a factor 2 down-sampler. The last stage is a frequency shift by $\frac{f_s}{4}$ or $-\frac{f_s}{4}$, followed by a lowpass filter and a factor 2 down-sampler. The required operation is determined from the binary representation of the desired channel. Assume the desired channel has the binary representation:

$$b_{m-1}b_{m-2}\dots b_1b_0 \tag{7.3}$$

Define x_i to be:

$$x_i = b_{m-i} \oplus b_{m-i-1} \tag{7.4}$$

For $i = 1 \dots m-1$.

If $x_i = 1$ use a highpass filter during stage i. Otherwise, use a lowpass filter.

In stage m, the sign of the frequency shift is determined by the least significant bit b_0. If $b_0 = 1$, the frequency shift is $\frac{f_s}{4}$. Otherwise, it is $-\frac{f_s}{4}$.

The frequency spectrum is divided into two non-overlapping bands; the lowpass band and the highpass band. The fact that two bands are non-overlapping necessitates the use of sharp lowpass or highpass filters. This in turn leads to filters having a large number of taps, and hence an adverse effect of the computational power dissipation. In the next section, an algorithm that has overlapping selection bands is presented. This relaxes the sharpness requirement of the filters, which in turn leads to filters with a fewer number taps and hence lower power dissipation.

7.4 Filter Sharpness Relaxation

The algorithm of the previous section is modified such that, the frequency spectrum, which extends from $-\frac{f_s}{2}$ to $\frac{f_s}{2}$, is divided into four overlapping frequency bands [165]. Each frequency band has a bandwidth of $\frac{f_s}{2}$. Any channel lies in two frequency bands. In one band it will be closer to its center, in the other band it will be closer to its edge. The channel is selected by the band in which it is closer to its center, see Figure 7.5. The advantage of doing this is to relax the sharpness requirement of the used filters, and hence lower the computational power dissipation.

The channel selection algorithm requires the use of lowpass and bandpass filters. A highpass filter is only required during the first channel selection stage. The bandpass filter is centered around $\frac{f_s}{4}$ or $-\frac{f_s}{4}$ and it has a bandwidth of $\frac{f_s}{2}$. The bandpass filter is implemented as a frequency shift of $-\frac{f_s}{4}$ or $\frac{f_s}{4}$ followed by a lowpass filter. The highpass filter is implemented as a frequency shift of $\frac{f_s}{2}$ followed by a lowpass filter.

The type of filter used is determined by the frequency band in which the signal lies. This is shown in Figure 7.5. BPN is the negative frequency bandpass filter centered around $-\frac{f_s}{4}$. While, BPP is the positive frequency bandpass filter centered around $\frac{f_s}{4}$.

Channel Selection Example

Figure 7.6 is an example showing the selection of one channel out of 16 channels. The channel to be selected is channel 6 (0110). The channel selection process proceeds as follows; channel 6 lies near the center of the highpass band. A highpass filter is used, the highpass filter selects channels -8, -7, -6, -5, 4, 5, 6, and 7. The highpass filter is followed by a factor two down-sampler, leading to the second

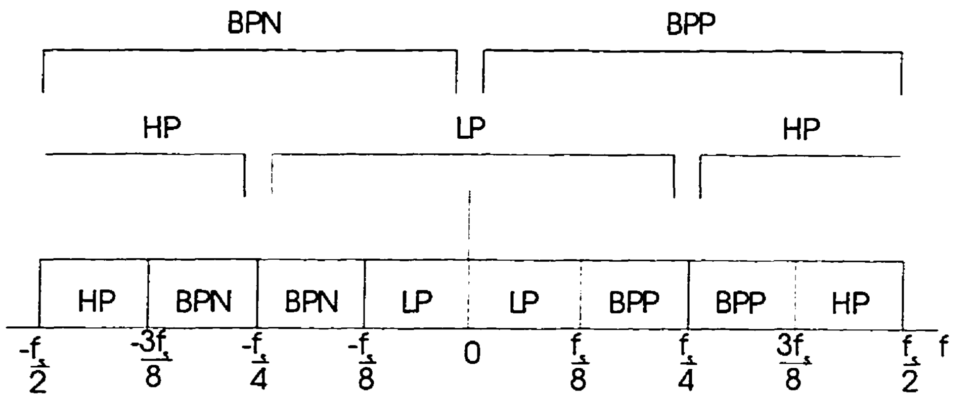


Figure 7.5: Filter used in each frequency band.

frequency spectrum in Figure 7.6. Notice that, only three bits are required to number the remaining channels. The most significant bit is removed.

Channel 6 now lies near the center of the negative frequency bandpass band. A negative frequency bandpass filter is used selecting channels; 4, 5, 6, and 7. The negative frequency bandpass filter is followed by a factor 2 down-sampler. The most significant bit is removed and the second most significant bit is inverted in this case.

Channel 6 now lies in the lowpass band. If we were to select channel 6 with channel 5 using a lowpass filter, then this lowpass filter will be sharp and it will require many taps. Instead, the spectrum is shifted by $-\frac{f_s}{8}$, a lowpass filter is used for selecting channel 6 in two stages. An implementation with reduced computational complexity, for this shifting process, is presented in section 7.4.2.

7.4.1 The Channel Selection Algorithm

The channel selection process, previously described for the selection of channel 6, is formulated in the following algorithm for the selection of any channel [165]:

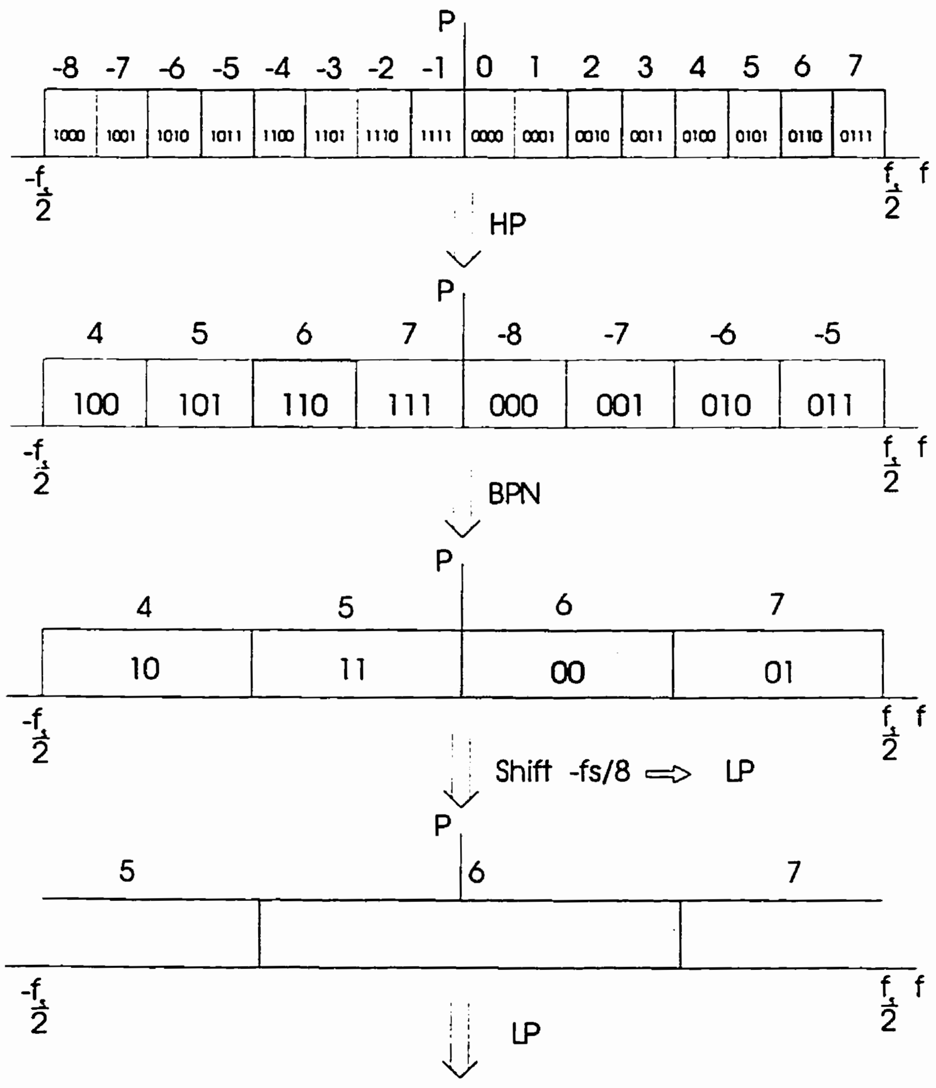


Figure 7.6: Channel selection example using lowpass, highpass and bandpass filters.

For the first stage

If $(b_{m-1} \oplus b_{m-2})\ (\overline{b_{m-2} \oplus b_{m-3}}) = 1$

Use a highpass filter.

$a_{m-2} = b_{m-2}$.

else if $(\overline{b_{m-1} \oplus b_{m-2}})\ (\overline{b_{m-2} \oplus b_{m-3}}) = 1$

Use a lowpass filter.

$a_{m-2} = b_{m-2}$.

else if $b_{m-1}\ (b_{m-2} \oplus b_{m-3}) = 1$

Use a negative frequency bandpass filter.

$a_{m-2} = \bar{b}_{m-2}$.

else if $\bar{b}_{m-1}\ (b_{m-2} \oplus b_{m-3}) = 1$

Use a positive frequency bandpass filter.

$a_{m-2} = \bar{b}_{m-2}$.

For the next $m-3$ stages

The condition $(a_{m-i} \oplus b_{m-i-1})\ (\overline{b_{m-i-1} \oplus b_{m-i-2}}) = 1$ can not occur.

If $(\overline{a_{m-i} \oplus b_{m-i-1}})\ (\overline{b_{m-i-1} \oplus b_{m-i-2}}) = 1$

```
    Use a lowpass filter.
    $a_{m-i-1} = b_{m-i-1}$.
else if $a_{m-i}\ (b_{m-i-1} \oplus b_{m-i-2}) = 1$
    Use a negative frequency bandpass filter.
    $a_{m-i-1} = \bar{b}_{m-i-1}$.
else if $\bar{a}_{m-i}\ (b_{m-i-1} \oplus b_{m-i-2}) = 1$
    Use a positive frequency bandpass filter.
    $a_{m-i-1} = \bar{b}_{m-i-1}$.
Where, $i = 2 \ldots m-2$
```

For stage $m-1$

```
if $a_1\ \bar{b}_0 = 1$
    Shift the spectrum by $\frac{3f_s}{8}$.
    Use a lowpass filter.
else if $a_1\ b_0 = 1$
    Shift the spectrum by $\frac{f_s}{8}$.
    Use a lowpass filter.
```

else if $\bar{a}_1 \ \bar{b}_0 = 1$

Shift the spectrum by $-\frac{f_s}{8}$.

Use a lowpass filter.

else if $\bar{a}_1 \ b_0 = 1$

Shift the spectrum by $-\frac{3f_s}{8}$.

Use a lowpass filter.

For stage m

Always use a lowpass filter.

7.4.2 Algorithm Implementation

The algorithm just described requires four types of filters, a lowpass filter, a highpass filter, a negative frequency bandpass filter, and a positive frequency bandpass filter. Each one of these filters has a bandwidth of $\frac{f_s}{2}$. The four filters can be implemented using a lowpass filter preceded by a multiplier that does the appropriate frequency shifting.

For the highpass filter, the frequency shift is $\frac{f_s}{2}$. This is performed by multiplying the incoming samples by a sequence of $1, -1, 1, -1, 1 \ldots$. This is shown in Figure 7.7.a.

For the negative frequency bandpass filter, the frequency shift is $\frac{f_s}{4}$. This is performed by multiplying the incoming samples by a sequence of $1, j, -1, -j, 1, j \ldots$. This is shown in Figure 7.7.b.

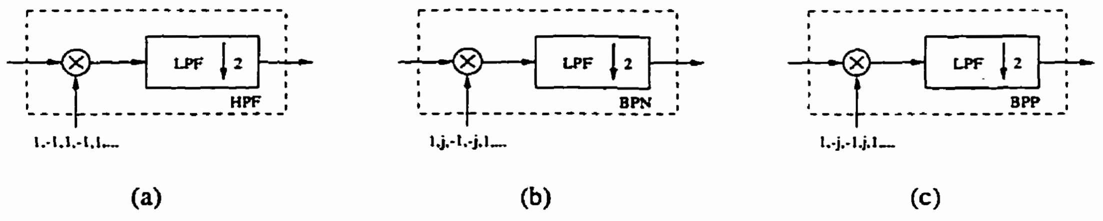


Figure 7.7: Implementation of: (a) HP filter (b) BPN filter (c) BPP filter using a multiplier and a LP filter.

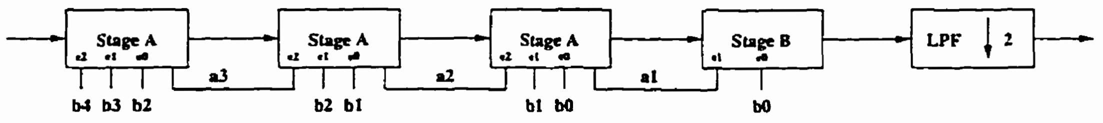


Figure 7.8: The proposed digital channel selection algorithm for the selection of one out of 32 channels. Stage A is a $\{0, \pm 1, 2\}\frac{f_s}{4}$ frequency shifter followed by a lowpass filter. Stage B is a $\pm\{1, 3\}\frac{f_s}{8}$ frequency shifter followed by a lowpass filter. $b_4b_3b_2b_1b_0$ is the binary representation of the selected channel number.

For the positive frequency bandpass filter, the frequency shift is $-\frac{f_s}{4}$. This is performed by multiplying the incoming samples by a sequence of $1, -j, -1, j, 1, -j \ldots$. This is shown in Figure 7.7.c.

The multiplier coefficients in these implementations are: $1, -1, j, \text{and} -j$. This doesn't require the use of a full multiplier, all that is required is multiplexers and simple logic gates. This achieves a dramatic saving in the power dissipation.

The implementation of the digital channel selection algorithm for the selection of one channel out of 32 channels is shown in Figure 7.8 [165]. This algorithm has five stages. The first three stages have a block diagram as shown in Figure 7.9. The fourth stage has a block diagram as shown in Figure 7.10. The fifth stage is a lowpass decimation filter.

In stage $m-1$ a frequency shift of $\pm\{1, 3\}\frac{f_s}{8}$ is required. To eliminate the use

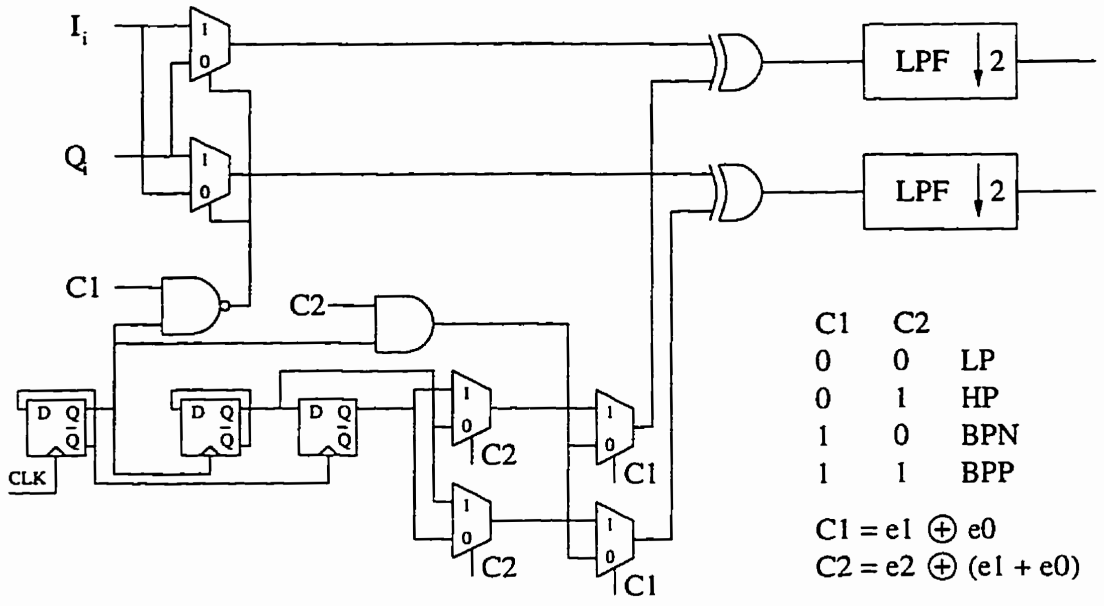


Figure 7.9: Implementation of a $\{0, \pm 1, 2\}\frac{f_s}{4}$ frequency shifter followed by a lowpass filter.

of a multiplier for this frequency shift operation, we exploit the fact that the filter following the frequency shifter is a decimation filter, that decimates the signal by a factor of 2. This filter is implemented as a polyphase filter [120] that separates the input sample stream into even and odd sample streams. The even samples are always multiplied by ± 1, or $\pm j$. While the odd samples are multiplied by $\frac{1}{\sqrt{2}}[\pm 1 \pm j]$. The multiplier is replaced by adders, multiplexers and other logic gates. The multiplier coefficient ($\frac{1}{\sqrt{2}}$) is hidden in the filter coefficients. The implementation of the frequency shifter followed by the filter is shown in Figure 7.10. Details of this implementation are explained below.

Assume a multiplier is used to provide the desired frequency shift, the multiplier coefficients are:

$$e^{\pm j\{1,3\}\frac{f_s}{8}n}.$$

Table 7.3: Frequency-shift-multiplier output.

Discrete time	$-\frac{f_s}{8}$	$-\frac{3f_s}{8}$	$\frac{3f_s}{8}$	$\frac{f_s}{8}$
0	$I_i + jQ_i$	$I_i + jQ_i$	$I_i + jQ_i$	$I_i + jQ_i$
1	$\frac{I_i+Q_i}{\sqrt{2}} - j\frac{I_i-Q_i}{\sqrt{2}}$	$-\frac{I_i-Q_i}{\sqrt{2}} - j\frac{I_i+Q_i}{\sqrt{2}}$	$-\frac{I_i+Q_i}{\sqrt{2}} + j\frac{I_i-Q_i}{\sqrt{2}}$	$\frac{I_i-Q_i}{\sqrt{2}} + j\frac{I_i+Q_i}{\sqrt{2}}$
2	$Q_i - jI_i$	$-Q_i + jI_i$	$Q_i - jI_i$	$-Q_i + jI_i$
3	$-\frac{I_i-Q_i}{\sqrt{2}} - j\frac{I_i+Q_i}{\sqrt{2}}$	$\frac{I_i+Q_i}{\sqrt{2}} - j\frac{I_i-Q_i}{\sqrt{2}}$	$\frac{I_i-Q_i}{\sqrt{2}} + j\frac{I_i+Q_i}{\sqrt{2}}$	$-\frac{I_i+Q_i}{\sqrt{2}} + j\frac{I_i-Q_i}{\sqrt{2}}$
4	$-I_i - jQ_i$	$-I_i - jQ_i$	$-I_i - jQ_i$	$-I_i - jQ_i$
5	$-\frac{I_i+Q_i}{\sqrt{2}} + j\frac{I_i-Q_i}{\sqrt{2}}$	$\frac{I_i-Q_i}{\sqrt{2}} + j\frac{I_i+Q_i}{\sqrt{2}}$	$\frac{I_i+Q_i}{\sqrt{2}} - j\frac{I_i-Q_i}{\sqrt{2}}$	$-\frac{I_i-Q_i}{\sqrt{2}} - j\frac{I_i+Q_i}{\sqrt{2}}$
6	$-Q_i + jI_i$	$Q_i - jI_i$	$-Q_i + jI_i$	$Q_i - jI_i$
7	$\frac{I_i-Q_i}{\sqrt{2}} + j\frac{I_i+Q_i}{\sqrt{2}}$	$-\frac{I_i+Q_i}{\sqrt{2}} + j\frac{I_i-Q_i}{\sqrt{2}}$	$-\frac{I_i-Q_i}{\sqrt{2}} - j\frac{I_i+Q_i}{\sqrt{2}}$	$\frac{I_i+Q_i}{\sqrt{2}} - j\frac{I_i-Q_i}{\sqrt{2}}$

Where, n is the discrete time. These coefficients are periodic with a period $N = 8$. Table 7.3, gives eight consecutive samples of the multiplier output for the four different frequency shifts. The input sample is $I_i + jQ_i$.

Notice from Table 7.3, that the even samples require only multiplication by ± 1, in addition to I_i/Q_i interchange, which is equivalent to multiplying by $\pm j$. The odd samples always require multiplication by $1/\sqrt{2}$, in addition to subtraction or addition operations. The multiplication operation is hidden in the coefficients of the odd sample branch of the polyphase filter [120]. Hence, no extra multiplications are required to implement the frequency shift.

Now that the sharpness of the filters is not critical, in the initial filtering stages Sinc decimators are used eliminating the need for multipliers. However, in the last two stages a sharper lowpass filter is used. Table 7.4 gives the sufficient image channel rejection required to achieve a BER of 10^{-2} for a DAMPS system. Also given in this table is the Sinc decimator order and the number of LPF taps required

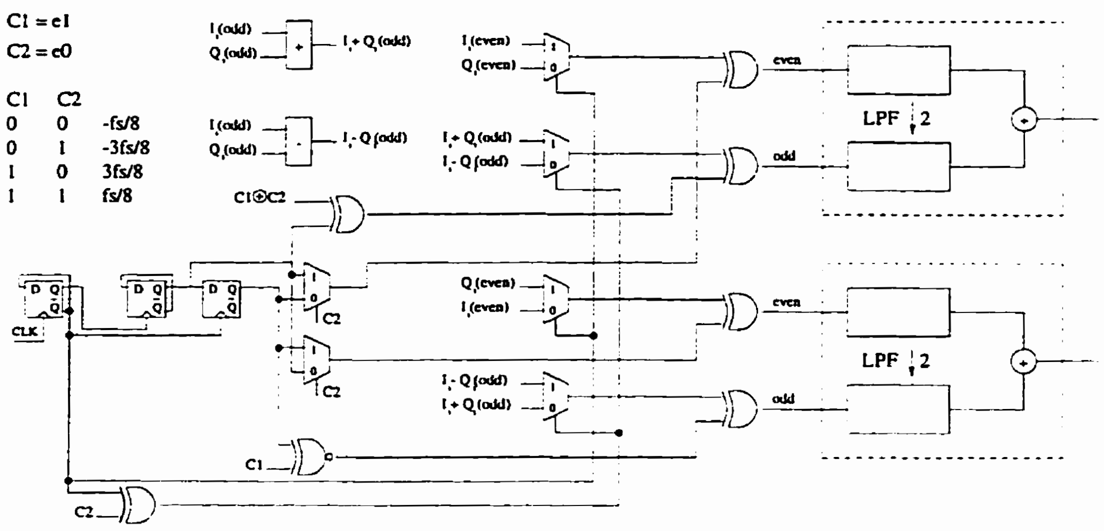


Figure 7.10: Implementation of a $\pm\{1,3\}\frac{f_s}{8}$ frequency shifter followed by a lowpass filter.

to satisfy this requirement.

The number of operations required per stage per output, the power dissipation per stage, and the total power dissipation are also shown in Table 7.4[1]. The values given in this table are for a $0.5\mu m$, 3.3 Volt CMOS technology.

For a digital channel selection algorithm that selects one out of 32 channels, eliminating the pre-filter multiplier achieves an 81% saving in power dissipation

[1]An eighth order Sinc filter with a decimation factor of two has the transfer function:

$$H(z) = 1 + 8z^{-1} + 28z^{-2} + 56z^{-3} + 70z^{-4} + 56z^{-5} + 28z^{-6} + 8z^{-7} + z^{-8} \qquad (7.5)$$

Direct implementation of Equation 7.5 requires 18 addition operations. Using the properties of arithmetic operators, such as commutativity, associativity and common factoring, Equation 7.5 can be reformated as:

$$H(z) = (1 + z^{-8}) + 8(z^{-1} + z^{-7}) + (8 - 1)[4(z^{-2} + z^{-6}) + 8(z^{-3} + z^{-5}) + (8 + 2)z^{-4}] \qquad (7.6)$$

Now the Sinc filter, according to Equation 7.6, requires only 10 addition/subtraction operations.

Table 7.4: Computational power dissipation for the novel digital channel selection algorithm shown in Figure 7.8, and designed in a $0.5\mu m$, 3.3 Volt CMOS technology.

Criteria	1st stage	2nd stage	3rd stage	4th stage	5th stage
Image rejection	64 dB	64 dB	64 dB	51 dB	22 dB
Order/# of tapes	8	8	8	13	32
Speed	480 KHz	240 KHz	120 KHz	60 KHz	30 KHz
Operations	20 ADD(20) 4 MUX(20) 4 XOR(20)	20 ADD(20) 4 MUX(20) 4 XOR(20)	20 ADD(20) 4 MUX(20) 4 XOR(20)	14MULT(20×12) 26 ADD(20) 4 MUX(20) 4 XOR(20)	32MULT(20×9) 62 ADD(20)
Power dissipation	0.549 mW	0.275 mW	0.137 mW	0.759 mW	0.688 mW
Total power dissipation		2.407 mW			

when compared to the conventional channel selection algorithm of Table 7.1, and 65% saving in power dissipation when compared to the channel selection algorithm of Table 7.2.

7.5 Digital Channel Selection Algorithms: Comparison

In this section, the power dissipation of three channel selection algorithms are compared.

Architecture A. This is the architecture given in Figure 7.2. In this architecture a pre-filter multiplier is used to shift the channel to be selected to be centered around the zero frequency. A single stage polyphase filter is then used to select the desired channel.

Architecture B. This architecture is also given in Figure 7.2. In this architecture a pre-filter multiplier is used to shift the channel to be selected to be centered around the zero frequency. A multi-stage filter is then used to select the desired channel. Each stage does decimation by a factor 2.

Architecture C. This architecture is described in section 7.4. This architecture eliminates the pre-filter multiplier. Channel selection is achieved through an algorithm that determines the type of filter (LP, HP, BPN, or BPP) based on the channel to be selected, see section 7.4.1.

Figure 7.11 gives the computational power dissipation versus the number of channels, for the three different architectures of the digital channel selection algorithm. Notice that, the increase in computational power dissipation of architecture A as the number of channels from which one channel is selected increases, exceeds

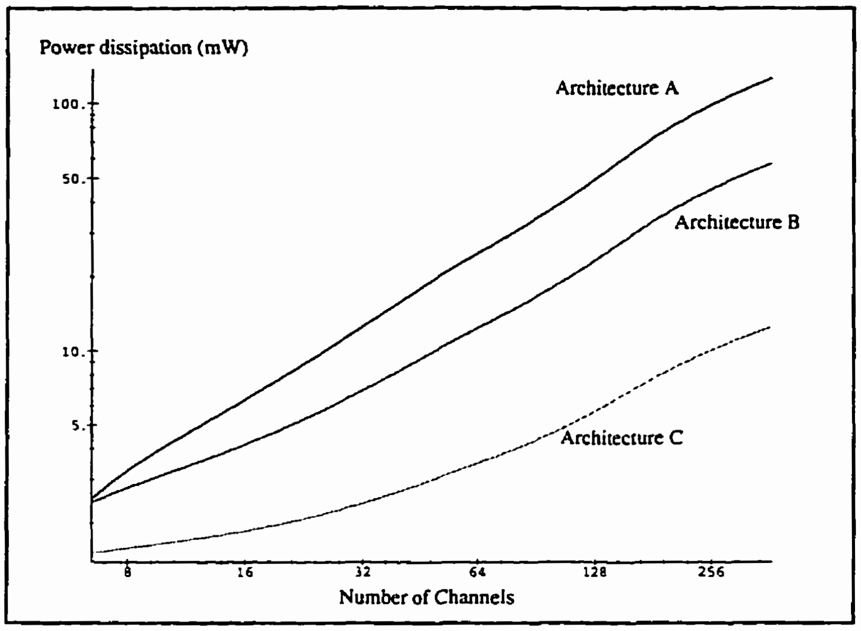


Figure 7.11: Computational power dissipation for digital channel selection algorithms.

the rate of computational power dissipation increase of architecture B which in turn exceeds the rate of increase of computational power dissipation of architecture C.

Figure 7.12 gives the relative computational power dissipation of architecture A and architecture B relative to that of architecture C. Again as the number of channels form which one channel is selected increases, the efficiency of architecture C in saving power over architectures A and B becomes more apparent. When selecting one out of 256 channels, architecture C can achieve an order of magnitude saving in computational power dissipation over that of architecture A. Architecture C also lowers the computational power dissipation by 4.5 times over that of architecture B.

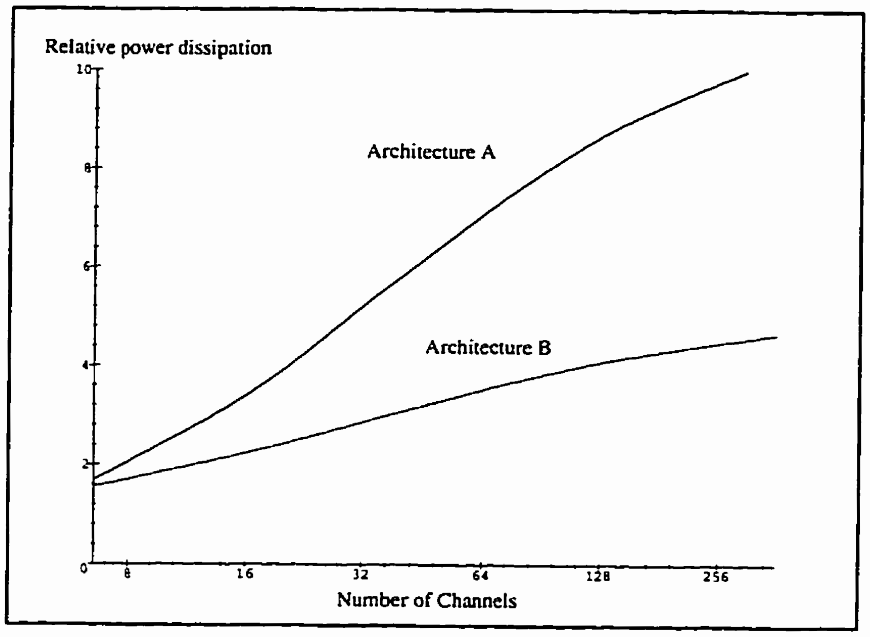


Figure 7.12: Relative computational power dissipation between conventional digital channel selection algorithms and proposed algorithm.

7.6 Chapter Summary

A novel digital channel selection algorithm with no pre-filter multiplier has been developed in this chapter. The algorithm performs channel selection in stages. During each stage, half the channels are rejected, and the other half is selected. The algorithm employs a basic lowpass filter, which can also perform bandpass and highpass filtering functions by using simple logic gates.

The use of overlapping frequency bands to perform the channel selection, relaxes the filter sharpness requirement. Thus permitting the use of Sinc filters in the initial stages. Sinc decimators require only addition operation. This greatly reduces the computational power dissipation.

The computational power dissipation of the novel digital channel selection algorithm is compared to that of other channel selection algorithms. The reduction in computational power dissipation can be up to an order of magnitude.

Chapter 8

Summary, Conclusions and Future Directions

8.1 Summary

The research reported in this dissertation is a study of high-level (algorithmic and architectural) techniques to lower the power dissipation in portable wireless terminals.

The first part of this dissertation is a survey of the wireless communication systems; architectures and standards, this is presented in chapter 2, and of low-power techniques, this is presented in the first half of chapter 3.

In the second part of this dissertation, low-power techniques for portable wireless terminals at the architectural and the algorithmic levels were investigated, developed, designed and evaluated. The low-power techniques are applied to video compression algorithms used in multimedia portable terminals. Low-power algorithms are also developed to lower the power dissipation in software radios.

The first contribution of this dissertation is the analysis of high-level low-power design techniques, to show the limit to which some of these design techniques can be applied effectively, to reduce the power dissipation. Three examples of such analysis are given in chapter 3. These are; the use of carry save adders in FIR filters, the use of the Gray code number system, and the use of higher-order radix for the division operation.

The second contribution of this dissertation is a new division algorithm that minimizes the number of addition/subtraction operations required to generate the quotient. This algorithm is presented in chapter 3.

The third contribution of this dissertation is a new low-power subband coding image compression algorithm developed in chapter 4. Subband coding is a technique in which the video signal is divided into subbands and each subband is allocated a number of bits according to the information it carries and its spectral importance. The filtering structure for the proposed subband coding image compression algorithm, requires only addition/subtraction operations, this greatly reduces the power dissipation. A novel vector quantization coding algorithm having a simplified decoding architecture has been developed in chapter 4.

The fourth contribution of this dissertation is a novel bandpass Sigma-Delta modulator, along with its switched capacitor architecture, presented in chapter 5. Parallelism by 4x of analog signal processors is applied to the design of the bandpass Sigma-Delta modulator. This increases the speed of the modulator without increasing the speed requirement of the individual building blocks. A switched-capacitor circuit with a minimum number of operational amplifiers is also given for the proposed modulator architecture.

The fifth contribution of this dissertation is the design of a decimation filter incorporating several low-power design techniques such as; operation minimization, multiplier elimination and block deactivation. The decimation filter is resolution-programmable, allowing the deactivation of the blocks corresponding to the least significant bits, when a lower resolution is sufficient. The design of the decimation filter is given in chapter 5.

The sixth contribution of this dissertation is the design of a resolution programmable multiplier-accumulator (MAC) array. The interleaving of the adder in the multiplier array reduces the power dissipation. The resolution of the MAC array is programmable allowing the deactivation of the blocks corresponding to the least significant bits when a lower resolution is sufficient. The design of the MAC array is given in chapter 6.

The seventh contribution of this dissertation is a novel digital channel selection algorithm with no pre-filter multiplier. The channel selection algorithm uses lowpass, highpass and bandpass filters. The basic filter is the lowpass filter. Other filters are implemented using the lowpass filter and simple logic gates such as multiplexers and XOR gates. The channel selection is done in stages. The elimination of the pre-filter multiplier reduces the power dissipation by up to an order of magnitude. The design of the digital channel selection algorithm is given in chapter 7.

8.2 Conclusions

The widespread use of portable wireless terminals, and the increase in consumer demand for more capabilities and functionality, coupled with the limited energy supply of batteries is driving research into the low-power design arena.

The promising results obtained from this dissertation indicate that the proper choice of an algorithm or architecture can achieve up to an order of magnitude, or even more in some cases, of savings in the power dissipation. High-level low-power design techniques are easier and faster to implement than low-level low-power design techniques.

The high-level techniques presented in this dissertation to reduce the power dissipation include; multiplier elimination, operation minimization, operation interleaving and block deactivation.

At the heart of these techniques is the minimization of the computational complexity by the elimination of redundant and irrelevant computations. Redundant computations are those whose elimination has no effect on the operation of the architecture. This is generally achieved by proper encoding of the input and output data, or by modifying the architecture. For example, multiplication by 7 requires two addition operations:

$$7A = 4A + 2A + A \tag{8.1}$$

However, encoding 7 in sign-digit representation $7 = 100\bar{1}$, eliminates one addition operation and substitutes the other one by subtraction:

$$7A = 8A - A \tag{8.2}$$

This is an example of redundant computations elimination. Redundant computations also occur in decimation filters, where there is redundancy in the intermediate stages, due to the decimation at the output of the filter. Through proper choice of an architecture, the redundancy in the intermediate stages is minimized.

Irrelevant computations are those whose elimination generally affects the operation of the architecture. But under certain circumstances, this effect is irrelevant.

For example, reducing the datapath width of an operator when the SNR is limited by some other system considerations, or by another unit in the system, is an example of irrelevant computations elimination.

In chapter 7, a novel digital channel selection algorithm was presented that eliminates the pre-filter multiplier. The multiplier was replaced by less computationally complex operators, such as adders, multiplexers and XOR gates. This algorithm filters the desired channel in stages. During each stage, the signal is decimated by a factor of 2. To further reduce the computational complexity, multiplication is substituted by addition in the first filtering stages. This reduces the computational power dissipation by up to an order of magnitude. This is an example of redundant computations elimination.

The lowpass decimation filter presented in chapter 5, was a halfband filter with symmetric coefficients. This reduced the number of multiplications required per output sample, which in turn led to a power saving of over 50%.

The effect of operation minimization on reducing the power dissipation was demonstrated in section 3.9.2 of chapter 3. A novel division algorithm was presented in that section that produces the quotient in a minimal sign-digit representation. Simulation results indicate that the new algorithm achieves a 15% saving in power dissipation when compared to a radix 4 division algorithm. This is an example of redundant computations elimination.

Operation minimization was also considered in chapter 5, for the design of the Sinc decimator. The fact that the output of the Sinc filter is decimated, offers opportunity to minimize the number of one-bit addition operations required per Sinc decimator output. Of the different Sinc decimator architectures considered in chapter 5, the most computationally efficient implementation reduces the number of one-bit addition operations by up to five times when compared to other Sinc decimator architectures. This is also an example of redundant computations elimination.

Besides multiplier elimination and operation minimization, operation interleaving is another technique to lower the power dissipation. Operation interleaving allows two (or more) operators to be interleaved into a single operator. In chapter 6, the effect of interleaving an adder and a multiplier in a single array was presented. For a 10×10 multiplier-accumulator array, operation interleaving achieves a 27.4% saving in power dissipation.

Wireless portable terminals are required to have very high sensitivity and selectivity, in order to operate in severe radio environments. In software radio, this leads to high resolution A/D converters and high resolution digital signal processors in the digital IF portion of the receiver. However, in many cases the level of received signal is strong enough to obviate the need for such high resolution. In this case, block deactivation is used to deactivate the blocks corresponding to the least significant bits.

Block deactivation was applied to the design of the lowpass decimation filter of chapter 5, and also to the design of the multiplier-accumulator (MAC) array of chapter 6. The block deactivation technique presented can achieve a power saving of up to 50% for the MAC array and up to 40% for the lowpass decimation filter. This is an example of irrelevant computations elimination.

Reducing the datapath width without degrading the output numerical accuracy plays a central role in achieving a power-efficient architecture. A Sinc decimator of order $N + 1$ is used to decimate the output signal of the Sigma-Delta modulator of order N. Each octave of oversampling in the Sigma-Delta modulator increases the resolution by $N + 0.5$ bits. However, each octave of decimation by the Sinc decimator increases the datapath width by $N + 1$ bits. Half a bit more than what is required.

For a third-order Sinc decimator having a decimation factor of 8, it is possible to reduce the output datapath width from 10 bits to 7 bits, at the expense of a 3 - 4 dB degradation in the SNR (which is equivalent to just over half a bit). The saving in computational complexity is 9–14%. This is an example of irrelevant computations elimination.

The choice of an algorithm can have the greatest impact on the power dissipation and the system performance. In many cases, it is possible to trade a slight degradation in algorithm performance, for a larger reduction in power dissipation. The subband image compression algorithm presented in chapter 4 has low computational complexity. The filtering structure reduces the computational complexity 23 times. The vector quantization algorithm used in the decoding of the high frequency subbands achieves a large reduction in computational complexity over the other vector quantization algorithms such as table-lookup vector quantization decoding algorithms. This reduction in computational complexity is achieved at the expense of a 4 – 5 dB reduction in the SNR.

The choice of a particular low-power technique depends on the design parameters of the system being design. Using an algorithm or an architecture to reduce one component of the power dissipation may lead to an increase in another component of the power dissipation. In chapter 3, several such examples where considered, one of which was the Gray code number system. The Gray code number system reduces the switching activity for successively correlated samples, when compared to the two's complement number system. However, the energy per operation is higher for the Gray code number system. The effect of the Gray code number system on reducing the power dissipation depends on several factors such as, the correlation coefficient, the load capacitance, and the energy per operation. In chapter 3, the limits of applying the Gray code to reduce the power dissipation were presented.

In chapter 5, a parallel bandpass Sigma–Delta modulator is proposed. The basic advantage of the proposed architecture is that each block operates at a lower speed (because of the parallelism) than that of the conventional bandpass Sigma-Delta modulator for the same oversampling ratio. This makes it suitable for high-speed Sigma-Delta modulators, such as those used in RF applications, for high-speed analog-to-digital conversion.

Another advantage of the switched capacitor architecture, given in chapter 5, is in the reduced number of operational amplifiers required. A fourth-order bandpass Sigma-Delta modulator typically requires four operational amplifiers [140] [141]. The bandpass Sigma-Delta modulator presented in chapter 5 uses a factor of four parallelism (for both I and Q channels), while the number of operational amplifiers required is eight, only increased by a factor of two.

8.3 Future Directions

8.3.1 Low-Power Digital Radio

Future mobile wireless terminals will be able to perform, in the digital domain, many of the functions that are currently being done in the analog domain. An example of this is digital channel selection which was considered in chapter 7. New power-efficient algorithms for these operations, suitable for digital implementations, need to be developed.

Furthermore, software radios allow new features which are not available in current transceivers, such as the ability to change coding scheme, modulation scheme, and even multi-access scheme, based on the radio channel environment. This topic needs further investigation. New algorithms need to be developed to determine which coding/modulation/multi-access schemes can lead to optimum system performance for a certain radio channel environment.

We have presented an A/D converter architecture with a sampling rate of 1.25 MS/s. This captures only part of the 25 MHz cellular band. To capture the entire cellular band an A/D converter with a sampling rate of 62.5 MS/s (1.25 times the Nyquist-rate) is required, and with a 20+ bits resolution. More research is required in this area to achieve power-efficient high-resolution A/D converters operating at such a rate.

8.3.2 Low-Power Multimedia

New applications are likely to emerge as wireless multimedia terminals become popular. New power-efficient algorithms and architectures need to be developed for these applications. For example, to enable video conferencing, a duplex video link needs to be established. The vector quantization algorithm presented in chapter 4, is power-efficient for video decompression. For video compression a power-efficient search algorithm is required. Thus, a multimedia terminal has to be able to perform video compression as well as video decompression in a power-efficient manner.

There are enormous opportunities for future research to reduce the power dissipation in wireless multimedia terminals at the system level. Future wireless terminals are required to handle different types of information such as speech, video and data. Each type of information has its own requirements in terms of the acceptable delay and the probability of error. For example, voice packets must have a low delay, while delay is not critical for data packets. On the other hand, voice packets can tolerate a small amount of transmission errors, while data packets can not tolerate such errors. Based on this, a strategy for the power-efficient transmission of packets needs to be established. In an unfavourable radio environment, transmitted packets are likely to suffer transmission errors, which leads to retransmission requests for data packets leading to a high loss of valuable power. To minimize data packet retransmission, data packets can be transmitted with a higher power level or use a more elaborate error correction code, both these lead to higher power dissipation, but can lead to a power efficient solution in the sense that they avoid data retransmissions. The optimum error coding algorithm, and the optimum power transmission level for the most power-efficient transmission strategy of information packets need to be determined for different radio channel environments.

8.3.3 Low-Power CAD

Traditionally, high-level synthesis systems have been associated with the optimization of area and speed [167] [168]. Recently, high-level synthesis systems have started to address the optimization of power dissipation as well [51] [169] [170]. However, automated tools for synthesizing power optimum algorithms are still not available. Further research needs to be carried out in this area.

An integral part of a high-level low-power synthesis systems is fast but accurate high-level power estimation. The majority of the literature deals with power estimation at the transistor, switch or gate levels [45] [47] [49]. Power estimation at the register transfer level (RTL) and the architectural level is starting to capture the attention of researchers [52] [96] [171] [172]. However, further research is still required for power estimation at the architectural level. Power estimation at the algorithmic level is the least researched segment of the power estimation process. This area has the greatest research potential because of the dramatic impact algorithm selection has on the minimization of the power dissipation.

There is a need for a low-power CAD system, that explores the algorithmic design space to determine the most power-efficient algorithm, for a certain implementation technology, and under specific constraints. It is envisioned that such a CAD system will consist of three parts, the input part, the processing part and the output part. The input part consists of all the information that needs to be known before processing can start. This information consists of:

- Basic building block parameters. This is a library that contains information such as delay, power dissipation and area about the basic building blocks. The basic building blocks include adders, multipliers, multiplexers, memory elements, etc.
- Design constraints. The design constraints include throughput, SNR, compression ratio and total area.
- Low-power techniques knowledge-based system. This is a database of the low-power techniques and transformations, and the extent by which they can reduce the power dissipation.
- Algorithm knowledge-based system. For example in a software radio CAD system, this is a database of the software radio algorithms and the tradeoffs that can reduce the computational complexity and the effect of that on the performance of the algorithm.

The processing part, which is the set of design rules, makes use of all of this input information and the builtin knowledge-based systems and it determines the algorithm and its corresponding architecture that minimize the power dissipation. Finally, the output part is the power optimum design.

Most of the low-power techniques presented in this dissertation rely on the minimization of the computational complexity by eliminating the redundant computations. But what is the minimum computation required for a certain architecture? And how can we reach this limit? These questions can be answered for each architecture individually. However, there is no automated design methodology that can transform any general architecture into an architecture with minimum computations. This topic needs further research.

8.3.4 New Low-Power Techniques

The break up of power dissipation in digital ICs used in portable terminals is given by the pie-chart of Figure 8.1[1]. It is interesting to note that the power dissipated by the clock and its associated circuits is about 50% of the total power dissipation. Asynchronous circuits don't require a clock and thus have a potential of achieving a saving in power dissipation of up to 50%.

Whereas synchronous circuits have been around for many years and are fully understood, the application of asynchronous circuits in low-power has just started recently [173] [174]. Further research is needed in this area. It is envisioned that future systems might be some type of hybrid system using both asynchronous and synchronous circuits.

[1]From the presentation of Dr. Deo S. Singhat at the University of Waterloo, on Tuesday the 1st of April 1997

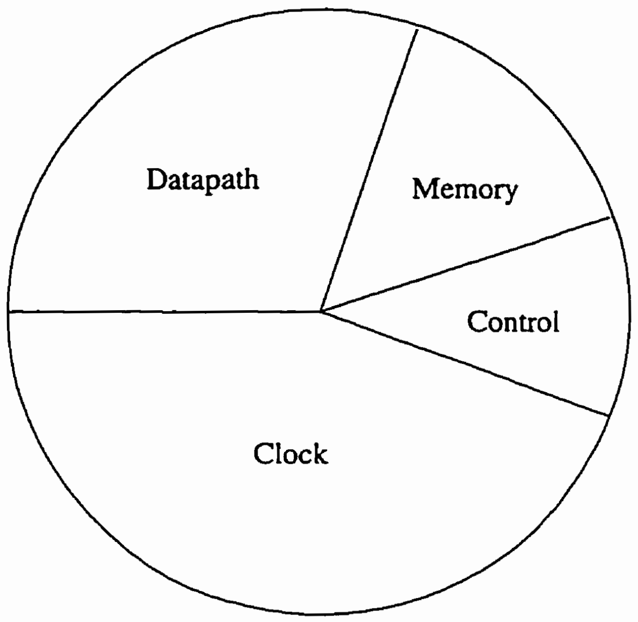


Figure 8.1: Pie-chart of the distribution of the power dissipation in portable terminals.

Low-power design is vital for the survival of the telecommunication and computer industries. As the functionality of portable terminals increases and as traditional low-power techniques exhaust, new low-power techniques must be developed and integrated into future portable terminals.

Appendix A

The Simulation of the Sigma-Delta Modulator

The simulation of the Sigma-Delta modulator was performed using SPW$^{\text{TM}}$ (Signal Processing WorkSystem ®). The Signal Processing WorkSystem is an integrated framework for developing discrete-time signal processing systems and communication protocols [175]. SPW was used to model and simulate different Sigma-Delta modulator architectures with the purpose of verifying the functionality and determining the effect of implementation details such as mismatch and gain errors on the different architectures.

SPW consists of several modules. The main modules used for modeling and simulating the Sigma-Delta modulator are [175]:

1. The Block Diagram Editor also called Designer BDE. This is where the discrete-time signal processing system is created, edited and wired together.

2. The Signal Flow Simulator. This is the tool that simulates the operation of

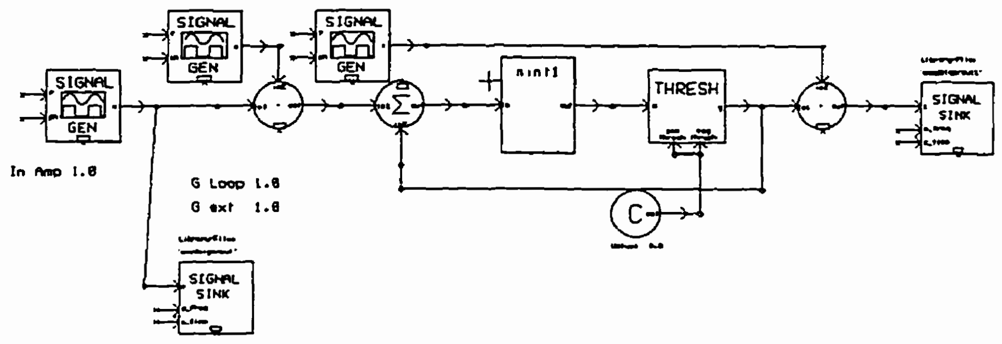


Figure A.1: A second-order bandpass Sigma-Delta modulator modeled in SPW.

the discrete-time signal processing system designed using SPW block diagram editor.

3. The Signal Calculator^{TM}. This is used for creating input signals and analyzing output signals.

Figure A.1 shows the block diagram of a single channel (I or Q) bandpass Sigma-Delta modulator. The input signal generated by SIGNAL GEN is a sinusoidal wave with a low frequency. Since the bandpass sampler samples the bandpass signal at double the carrier frequency for the I or Q channels. Therefore, to get a sampled stream identical to that at the output of the bandpass sampler, the low frequency sinusoidal wave is multiplied by a 1, -1, 1, -1, ... stream.

Figures A.2—A.7 show the SPW model of the novel parallel bandpass Sigma-Delta modulator in different configurations. The block diagrams of the block *nint1* of Figure A.1, and of the block *int2* of Figures A.2 — A.7, are shown in Figures A.8 and A.9 respectively.

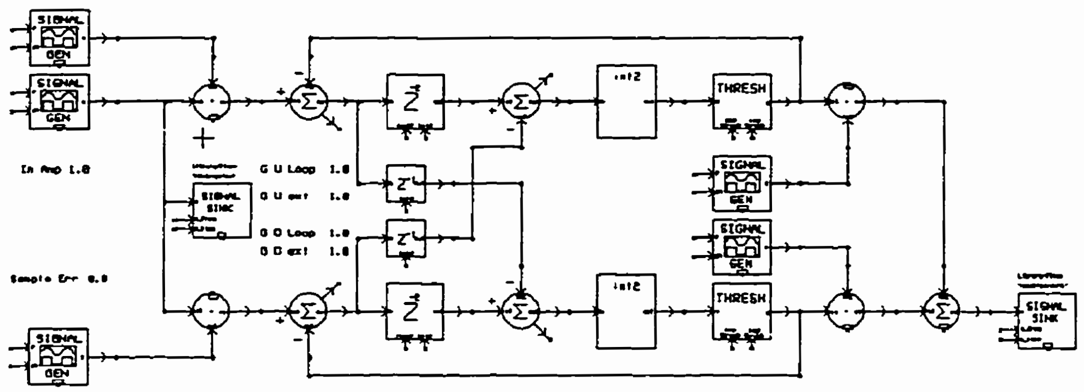


Figure A.2: SPW model of a single-channel second-order bandpass Sigma-Delta modulator, with two cross-coupled branches, and common filtering done after subtraction.

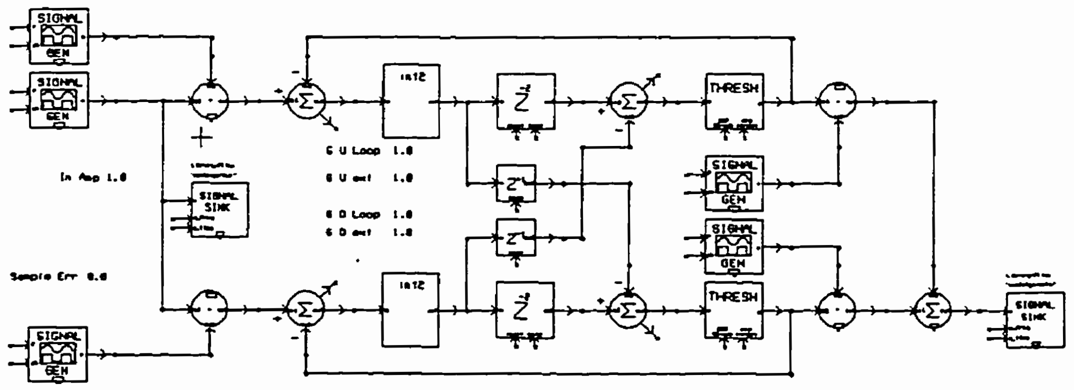


Figure A.3: SPW model of a single-channel second-order bandpass Sigma-Delta modulator, with two cross-coupled branches, and common filtering done before branch splitting.

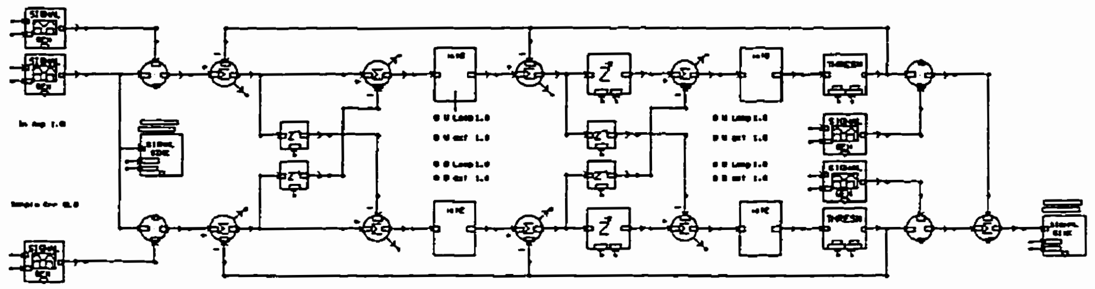


Figure A.4: SPW model of a single-channel fourth-order bandpass Sigma-Delta modulator, with two cross-coupled branches, and common filtering done after subtraction in the first and second stages.

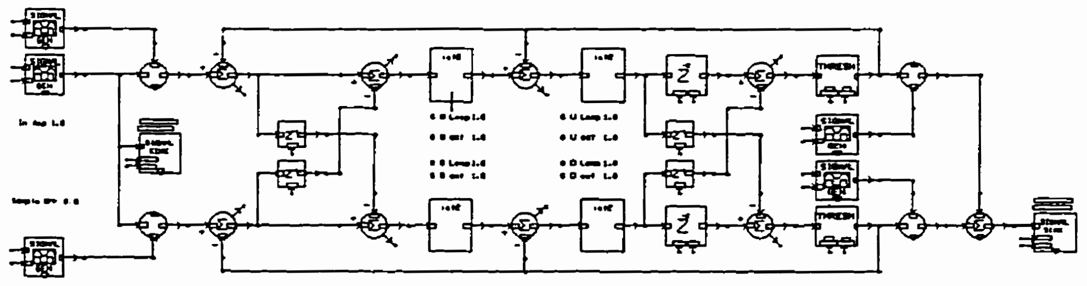


Figure A.5: SPW model of a single-channel fourth-order bandpass Sigma-Delta modulator, with two cross-coupled branches, and common filtering done after subtraction in the first stage and before branch splitting in the second stage.

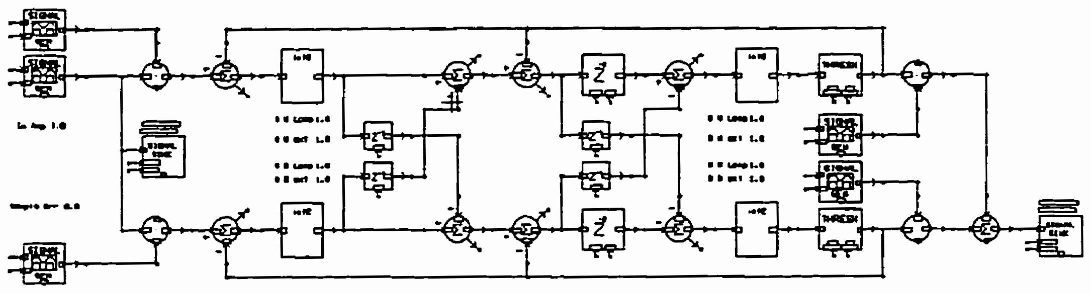


Figure A.6: SPW model of a single-channel fourth-order bandpass Sigma-Delta modulator, with two cross-coupled branches, and common filtering done before branch splitting in the first stage and after subtraction in the second stage.

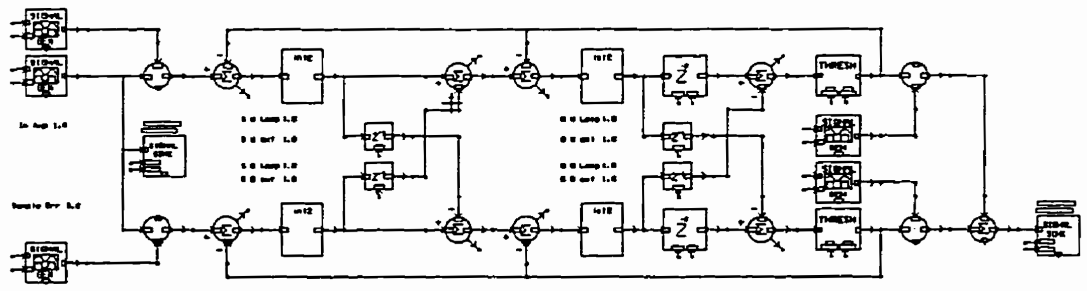


Figure A.7: SPW model of a single-channel fourth-order bandpass Sigma-Delta modulator, with two cross-coupled branches, and common filtering done before branch splitting in the first and second stages.

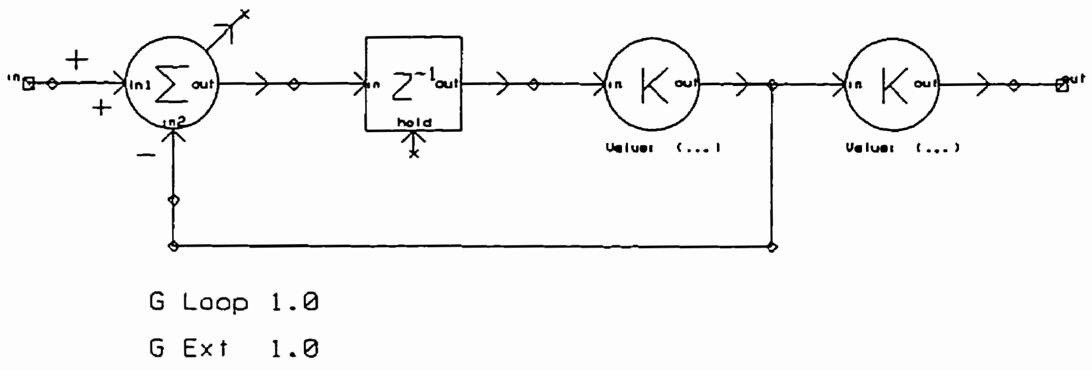


Figure A.8: SPW model of the block *nint1* of Figure A.1.

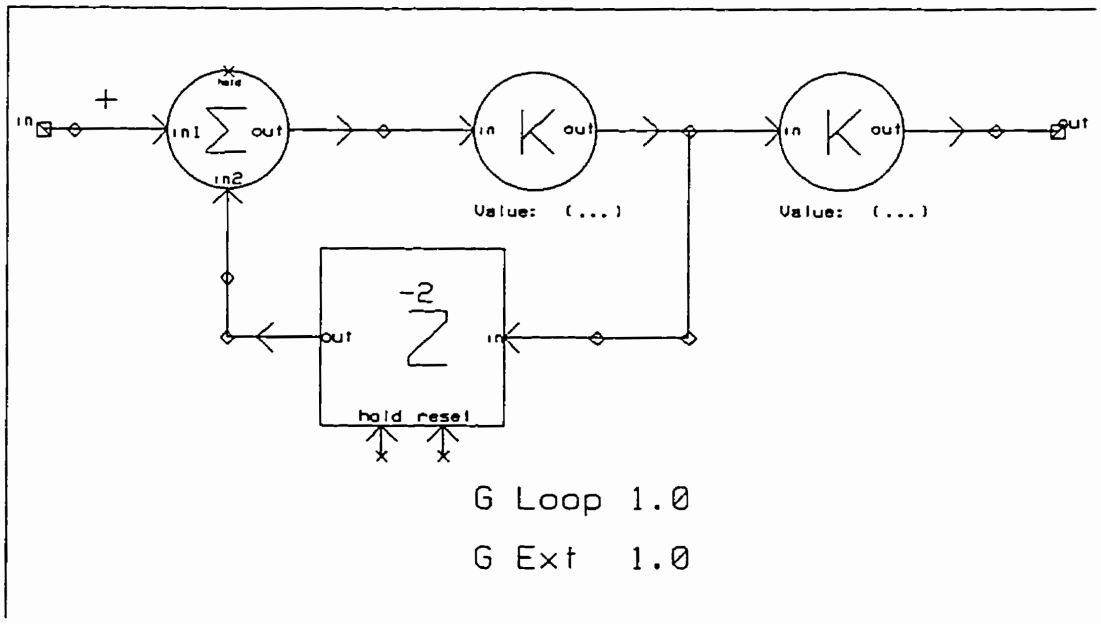


Figure A.9: SPW model of the block *int2* of Figures A.2 — A.7.

Appendix B

Sinc Decimator Analysis

The transfer function for the Sinc decimator [Sinc$^n(2^m)$] is given by:

$$H(z) = \frac{1}{M^n}\left(\frac{1-z^{-M}}{1-z^{-1}}\right)^n \tag{B.1}$$

Where, $M = 2^m$. According to Equation B.1, the Sinc decimator can be implemented as a cascade of n integrators followed by a 2^m down sampler and then followed by n differentiators [176]. This architecture has been previously explained in section 5.7.4, and is shown in Figure 5.43.

The purpose of the following analysis is to find the effect of each input sample i_b, on an output sample O_B. This analysis shows that even though the Sinc decimator consists of integrators which are infinite impulse response blocks, yet the response of the entire decimator of Figure 5.43.a has finite impulse response. Furthermore, we determine the maximum Sinc decimator output and the input sequence necessary to give this maximum output.

Before proceeding with the analysis, the format of the input and output samples need to be defined. The input samples, which can be represented by k bits, are bipolar samples having values:

$$i_s : \ 2^k - 1, \ 2^k - 3, \ \ldots\ldots - (2^k - 3), \ -(2^k - 1)$$

These samples are reinterpreted as unsigned binary values, through the following mapping:

$$i_b = \frac{1}{2}(i_s + 2^k - 1) \tag{B.2}$$

The output samples, O_B of the Sinc decimator are unsigned binary integers with resolution $k + mn$ bits. When we divide this by 2^{mn}, as required in Equation B.1, the output samples, O'_B, become unsigned binary numbers with k integer bits and mn fraction bits. To obtain the actual output samples O_S from O'_B. The following mapping is used:

$$O_S = 2O'_B - (2^k - 1) \tag{B.3}$$

O_S is a signed binary number with one sign bit, k integer bits and $mn - 1$ fraction bits.

The objective of this analysis is to find the relation between i_b and O_B, for a Sinc decimator having $k = 1$, $m = 3$, and $n = 3$. Figure B.1 shows the progression of an input sample at discrete time $T = 1$ through the Sinc decimator, to its output. Each integrator has a delay of one unit. The down-sampler decimates the signal at discrete time d, $d + 8$, $d + 16$, ... etc. Where $4 \leq d \leq 11$.

For an output sample at discrete time T. The latest input sample that can contribute to this output sample is at discrete time $T - 3$. The earliest input

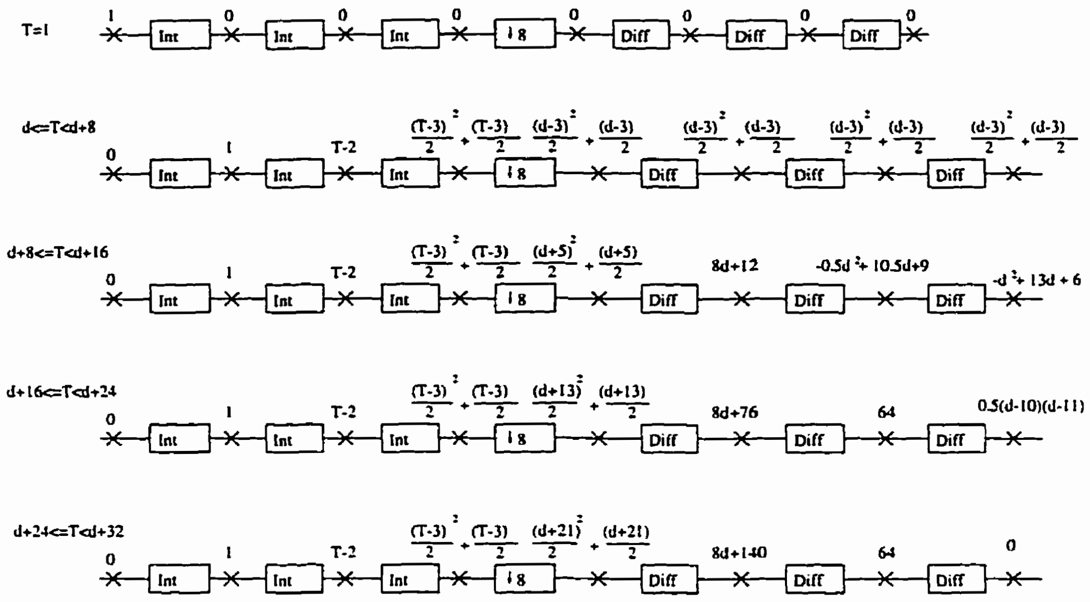


Figure B.1: The progression of an input sample through a Sinc decimator having $k = 1$, $m = 3$ and $n = 3$.

sample that can contribute to this output sample is at discrete time $T - 24$. Notice the finite impulse response of the Sinc decimator, despite the use of infinite impulse response integrators. Furthermore, the contribution of each input sample to the output sample is given in Table B.1. If we add all these contributions together, which means we have 22 successive ones at the input, the output will be 512. This is the maximum output.

Because any decimation filter is generally, a linear time-variant system, having 22 successive ones at the input of the Sinc decimator is not a sufficient condition to obtain an output of 512. The latest of these 22 successive ones must be 3 discrete time units earlier than the sampling time of the down-sampler.

Table B.1: The effect of an input sample to the Sinc decimator at discrete time $T-n$ on the output of the Sinc decimator at discrete time T. The Sinc decimator is $\text{Sinc}^3(8)$.

Discrete time	Contribution to output at T	Discrete time	Contribution to output at T
T	0	$T-13$	48
$T-1$	0	$T-14$	48
$T-2$	0	$T-15$	46
$T-3$	1	$T-16$	42
$T-4$	3	$T-17$	36
$T-5$	6	$T-18$	28
$T-6$	10	$T-19$	21
$T-7$	15	$T-20$	15
$T-8$	21	$T-21$	10
$T-9$	28	$T-22$	6
$T-10$	36	$T-23$	3
$T-11$	42	$T-24$	1
$T-12$	46	$T-25$	0

Bibliography

[1] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid State Circuits*, vol. SC-27, pp. 473–484, Apr. 1992.

[2] J. M. Rabaey and M. Pedram, *Low power design methodologies.* Boston: Kluwer Academic Publishers, 1996.

[3] A. Bellaour and M. I. Elmasry, *Low-Power Digital VLSI Design: Circuits and Systems.* Boston: Kluwer Academic Publishers, 1995.

[4] P. Michel, U. Lauther, and P. Duzy, *The Synhesis Approach to Digital System Design.* Boston: Kluwer Academic Publishers, 1992.

[5] D. Singh, J. M. Rabaey, M. Pedram, F. Catthoor, S. Rajgopal, N. Sehgal, and T. J. Mozdzen, "Power conscious CAD tools and methodologies: A perspective," *Proceedings of the IEEE*, vol. 83, pp. 570–594, Apr. 1995.

[6] B. Nadel, "The green machine," *PC Magazine*, vol. 12, pp. 110–145, May 25 1993.

[7] T. H. Meng, B. M. Gorgon, E. K. Tsern, and A. C. Hung, "Portable video-on-demand in wireless communication," *Proceedings of the IEEE*, vol. 83, pp. 659–679, Apr. 1995.

[8] S. Sheng, A. Chandrakasan, and R. W. Brodersen, "A portable multimedia terminal," *IEEE Communications Magazine*, pp. 64–75, Dec. 1992.

[9] P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI architectures for video compression — A survey," *Proceedings of the IEEE*, vol. 83, pp. 220–246, Feb. 1995.

[10] J. Mitola, "The software radio architecture," *IEEE Communications Magazine*, pp. 26–38, May 1995.

[11] J. A. Wepman, "Analog-to-digital converters and their applications in radio receivers," *IEEE Communications Magazine*, pp. 39–45, May 1995.

[12] R. Baines, "The DSP bottleneck," *IEEE Communications Magazine*, pp. 46–54, Mar. 1995.

[13] G. H. Heilmeier, "Personal communications: Quo vadis," in *IEEE International Solid-State Circuits Conference*, pp. 24–26, 1992.

[14] E. H. Armstrong, "A new system of short wave amplifications," *Proceedings of the Institute of Radio Engineers*, vol. 9, pp. 3–27, 1921.

[15] E. H. Armstrong, "The superheterodynce — its origin, development and some recent improvements," *Proceedings of the Institute of Radio Engineers*, vol. 12, pp. 539–552, 1924.

[16] W. Gosling, *Radio Receivers*. Cambridge, England: Peter Peregrinus Ltd., 1986.

[17] A. A. Abidi, "Low-power radio-frequency ic's for portable communications," *Proceedings of the IEEE*, vol. 83, pp. 544–569, Apr. 1995.

[18] R. J. Lackey and D. W. Upmal, "Speakeasy: The military software radio," *IEEE Communications Magazine*, pp. 56–61, May 1995.

[19] J. E. Padgett, C. G. Gunther, and T. Hatorri, "Overview of wireless personal communications," *IEEE Communications Magazine*, pp. 28–41, Jan. 1995.

[20] D. J. Goodman, "Second generation wireless information networks," *IEEE Transactions on Vechicular Technology*, vol. 40, pp. 366–374, 1991.

[21] L. E. Larson, *RF and Microwave Circuit Design for Wirless Communications.* Boston: Artech House, 1996.

[22] IS-137, "800 MHz TDMA cellular - radio interface - minimum performance standards for mobile stations," 1994.

[23] K. Pahlavan and A. H. Levesque, "Wireless data communications," *Proceedings of the IEEE*, vol. 82, pp. 1398–1430, Sept. 1994.

[24] M. J. Marcus, "Recent U.S. regulatory decisions on civil uses of spread spectrum," in *GLOBECOM*, vol. 1, (New Orleans, Louisiana), pp. 16.6/1–3, Dec. 1985.

[25] T. Tsukkahara, M. Ishikawa, and M. Muraguchi, "A 2V 2GHz Si-bipolar direct conversion quadrature modulator," in *IEEE International Solid-State Circuits Conference*, (San Francisco, California), pp. 40–41, Feb. 1994.

[26] F. E. Terman, *Radio Engineers' Handbook.* New York: MaGraw-Hill, 1943.

[27] T. Okanobu, H. Tomiyama, and H. Arimoto, "Advanced low voltage single chip radio IC," *IEEE Transactions on Consumer Electronics*, vol. 38, pp. 465–475, Aug. 1992.

[28] H. Tsurami and T. Maeda, "Design study on direct conversion receiver front-end for 280 MHz 900 MHz, and 2.6 GHz band radio communication systems," in *IEEE Vechicular Technology Conference*, (St. Louis, Missori), pp. 457–462, May 1991.

[29] J. Sevenhans, A. Vanwelsenaers, J. Wenin, and J. Baro, "An integrated Si bipolar RF transceiver for a zero IF 900 MHz GSM digital radio front-end of a hand portable phone," in *Custom Integrated Circuits Conference*, (San Diego, California), pp. 7.7/1–4, May 1991.

[30] G. Schultes, A. L. Scholtz, E. Bonek, and P. Veith, "A new incoherent direct conversion receiver," in *IEEE Vechicular Technology Conference*, (Orlando, Florida), pp. 668–674, May 1990.

[31] G. Shultes, E. Bonek, P. Weger, and W. Herzog, "Basic performance of a direct conversion DECT receiver," *Electronics Letter*, vol. 26, pp. 1746–1748, 1990.

[32] A. Bateman and D. Haines, "Direct conversion transceiver design for compact low-cost portable mobile radio terminals," in *IEEE Vechicular Technology Conference*, vol. 1, pp. 57–62, 1989.

[33] P. Estabrook and B. B. Lusignan, "The design of a mobile radio receiver using a direct conversion architecture," in *IEEE Vechicular Technology Conference*, vol. 1, (San Francisco, California), pp. 63–72, May 1989.

[34] R. W. A. Bateman, D.M. Haines, "Linear transceiver architectures," in *IEEE Vechicular Technology Conference*, (Philadelphia, Pennsylvania), pp. 478–484, June 1988.

[35] F. Piazza and Q. Huang, "A 170MHz RF front-end for ERMES pager applications," in *IEEE International Solid-State Circuits Conference*, (San Francisco, California), pp. 324–325, Feb. 1995.

[36] C. Takahashi *et al.*, "A 1.9GHz Si direct conversion receiver IC for QPSK modulation systems," in *IEEE International Solid-State Circuits Conference*, (San Francisco, California), pp. 138–139, Feb. 1995.

[37] A. Fernandez-Duran *et al.*, "Zero-IF receiver architecture for multisandard compatible radio systems," in *IEEE Vehicular Technology Conference*, vol. 2, (Atlanta, Georgia), pp. 1052–1056, Apr. 1996.

[38] J. Kennedy and M. C. Sullivan, "Direction finding and "smart antennas" using software radio architectures," *IEEE Communications Magazine*, pp. 62–68, May 1995.

[39] "SPEAKeasy Home, http://troi.web.rl.af.mil:8001/Technology/Demos/SPEAKEASY/."

[40] N. H. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective.* Reading, Massachusetts: Addison-Wesley Publishing Company, 1993.

[41] A. P. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proceedings of the IEEE*, vol. 83, pp. 498–523, Apr. 1995.

[42] H. J. M. Veendrick, "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *IEEE Journal of Solid State Circuits*, vol. SC-19, pp. 468–473, Aug. 1984.

[43] E. S. Yang, *Micro-Electronic Devices.* New York: McGraw Hill, 1988.

[44] J.-P. Collinge, *Silicon-on-Insulator Technology: Materials to VLSI.* Boston: Kluwer Academic Publishers, 1991.

[45] S. M. Kang, "Accurate simulation of power dissipation in VLSI circuits," *IEEE Journal of Solid State Circuits*, vol. SC-21, pp. 889–891, Oct. 1986.

[46] F. N. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Transactions on VLSI Systems*, vol. 2, pp. 446–455, Dec. 1994.

[47] M. A. Cirit, "Estimating dynamic power consumption of CMOS circuits," in *IEEE International Conference on Computer-Aided Design*, (Santa Clara, California), pp. 534–537, Nov. 1987.

[48] A. Shen, A. Ghosh, S. Devadas, and K. Keutzer, "On average power dissipation and random pattern testability of CMOS combinational logic networks," in *IEEE/ACM International Conference on Computer-Aided Design*, (Santa Clara, California), pp. 402–407, Nov. 1992.

[49] F. N. Najm, "Transition density: A new measure of activity in digital circuits," *IEEE Transactions on Computer-Aided Design*, vol. 12, pp. 310–323, Feb. 1993.

[50] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of average switching activity in combinational and sequential circuits," in *IEEE/ACM*

International Conference on Computer-Aided Design, (Santa Clara, California), pp. 253–259, Nov. 1992.

[51] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen, "Optimizing power using transformations," *IEEE Transactions on Computer-Aided Design*, vol. 14, pp. 12–31, Jan. 1995.

[52] P. E. Landman and J. M. Rabaey, "Power estimation for high level synthesis," in *Proceedings of the European Conference on Design Automation*, (Paris, France), pp. 304–308, Feb. 1993.

[53] E. A. Vittoz, "Low-power design: Ways to approach the limits," in *IEEE International Solid-State Circuits Conference*, (San Francisco, California), pp. 14–18, Feb. 1994.

[54] D. C. Cox, "Universal digital portable radio communications," *Proceedings of the IEEE*, vol. 75, pp. 436–477, Apr. 1987.

[55] A. Matsuzawa, "Low-voltage and low-power circuit design for mixed analog/digital systems in portable equipments," *IEEE Journal of Solid State Circuits*, vol. 29, pp. 470–480, Apr. 1994.

[56] A. P. Chandrakasan, A. Burstein, and R. W. Brodersen, "A low-power chipset for a portable multimedia I/O terminal," *IEEE Journal of Solid State Circuits*, vol. 29, pp. 1415–1428, Dec. 1994.

[57] T. Barber, P. Carvey, and A. Chandrakasan, "Designing for wireless LAN communications," *Circuits & Devices Magazine*, pp. 29–33, July 1996.

[58] I. Kang and A. N. Willson, "A low-power state-sequantional Viterbi decoder for CDMA digital cellular applications," in *IEEE International Circuits and Systems Conference*, vol. 4, (Atlanta, Georgia), pp. 272–276, May 1996.

[59] R. Cypher and C. B. Shung, "Generalized traceback techniques for survivor memory management in Viterbi decoder," in *GLOBECOM*, vol. 2, (Houston, Texas), pp. 1318–1322, Dec. 1993.

[60] E. Boutillon and N. Demassieux, "High speed low power architecture for memory management in a Viterbi decoder," in *IEEE International Circuits and Systems Conference*, vol. 4, (Atlanta, Georgia), pp. 284–287, May 1996.

[61] M. D. Hahm, E. G. Friedman, and E. L. Titlebaum, "Analog vs. digital: A comparison of circuit implementations for low-power matched filters," in *IEEE International Circuits and Systems Conference*, vol. 4, (Atlanta, Georgia), pp. 280–283, May 1996.

[62] A. P. Chandrakasan, M. Potkonjak, J. Rabaey, and R. W. Brodersen, "HYPER-LP: A system for power minimization using architectural transformations," in *IEEE International Conference on Computer-Aided Design*, (Santa Clara, California), pp. 300–303, Nov. 1992.

[63] W. Lee *et al.*, "A 1V DSP for wireless communications," in *IEEE International Solid-State Circuits Conference*, (San Francisco, California), pp. 92–93, Feb. 1997.

[64] C. J. Pan, "A low-power digital filter for decimation and interpolation using approximate processing," in *IEEE International Solid-State Circuits Conference*, (San Francisco, California), pp. 102–103, Feb. 1997.

[65] T. Kuroda *et al.*, "A 0.9V 150MHz 10mW 4mm^2 2-D discrete cosine transform core processor with variable-threshold-voltage scheme," in *IEEE International Solid-State Circuits Conference*, (San Francisco, California), pp. 166–167, Feb. 1996.

[66] B. M. Gordon, T. H. Meng, and N. Chaddha, "A 1.2mW video-rate 2D color subband decoder," in *IEEE International Solid-State Circuits Conference*, (San Francisco, California), pp. 290–291, Feb. 1995.

[67] E. K. Tsern and T. H. Meng, "A low-power video-rate pyramid VQ decoder," in *IEEE International Solid-State Circuits Conference*, (San Francisco, California), pp. 162–163, Feb. 1996.

[68] W. J. Bowhill *et al.*, "A 300MHz 64b quad-issue CMOS RISC microprocessor," in *IEEE International Solid-State Circuits Conference*, (San Francisco, California), pp. 182–183, Feb. 1995.

[69] N. K. Yeung *et al.*, "The design of a 55SPECint92 RISC processor under 2W," in *IEEE International Solid-State Circuits Conference*, (San Francisco, California), pp. 206–207, Feb. 1994.

[70] J. Montanaro *et al.*, "A 160-MHz, 32-b, 0.5-w CMOS RISC microprocessor," *IEEE Journal of Solid State Circuits*, pp. 1703–1714, Nov. 1996.

[71] D. W. Dobberpuhl *et al.*, "A 200-MHz 64-b dual-issue CMOS microprocessor," *IEEE Journal of Solid State Circuits*, vol. 27, pp. 1555–1567, Nov. 1992.

[72] R. Bechade *et al.*, "A 32b 66MHz 1.8W microprocessor," in *IEEE International Solid-State Circuits Conference*, (San Francisco, California), pp. 208–209, Feb. 1994.

[73] D. Pham *et al.*, "A 3.0W 75SPECint92 85SPECfp92 superscalar RISC microprocessor," in *IEEE International Solid-State Circuits Conference*, (San Francisco, California), pp. 212–213, Feb. 1994.

[74] J. M. C. Stork, "Technology leverage for ultra-low power information systems," *Proceedings of the IEEE*, vol. 83, pp. 607–618, Apr. 1995.

[75] M. Ino *et al.*, "0.25μm CMOS/SIMOX gate array LSI," in *IEEE International Solid-State Circuits Conference*, (San Francisco, California), pp. 86–87, Feb. 1996.

[76] M. Kakumu and M. Kinugawa, "Power-supply voltage impact on circuit performance for half and lower submicrometer CMOS LSI," *IEEE Transactions on Electron Devices*, vol. 37, pp. 1902–1908, Aug. 1990.

[77] S. Mutoh *et al.*, "1 V high-speed digital circuit technology with 0.5-μm multi-threshold CMOS," in *IEEE International ASIC Conference and Exhibit*, (Rochester, New York), pp. 186–189, Sept. 1993.

[78] K. Yano *et al.*, "A 3.8-ns CMOS 16 $\times$ 16-b multiplier using complementary pass-transistor logic," *IEEE Journal of Solid State Circuits*, vol. 25, pp. 388–395, Apr. 1990.

[79] A. J. Stratakos, S. R. Sanders, and R. W. Bordersen, "A low-voltage CMOS DC-DC converter for a protable battery operated system," in *IEEE Power Electronics Specialists Conference*, pp. 619–626, 1994.

[80] M. Alidina, J. Monterio, S. Devadas, A. Ghosh, and M. Papaefthmiou, "Precomputation-based sequential logic optimization for low power," in *International Workshop in Low Power Design*, 1994.

[81] T. Douseki *et al.*, "A 0.5V SIMOX-MTCMOS circuit with 200ps logic gate," in *IEEE International Solid-State Circuits Conference*, (San Francisco, California), pp. 84–85, Feb. 1997.

[82] S. R. Vemuru and A. R. Thorbjornsen, "Variable-taper CMOS buffer," *IEEE Journal of Solid State Circuits*, vol. 26, pp. 1265–1269, Sept. 1991.

[83] J. L. Hennessy and D. A. Patterson, *Computer Architecture A Quantative Approach*. San Mateo, California: Morgan Kaufmann Publishers, Inc., 1990.

[84] I. Koren, *Computer Arithmetic Algorithms*. Englewood Cliffs: Prentice Hall, 1993.

[85] T. K. Callaway and E. E. Swartzlander, "Estimating the power consumption of CMOS adders," in *IEEE Symposium on Computer Arithmetic*, (Windsor, Ontario, Canada), pp. 210–216, June 1993.

[86] T. K. Callaway and E. E. Swartzlander, "Otimizing arithmetic elements for signal processing," in *VLSI Signal Processing Workshop*, pp. 91–100, 1992.

[87] E. N. Farag and M. I. Elmasry, "Using carry save adders to reduce power dissipation," in *Eighth International Conference on Microelectronics*, (Cairo, Egypt), pp. 173–176, Dec. 1996.

[88] A. Gersho and R. M. Gray, *Vector quantization and signal compression*. Kluwer Academic Publishers, 1992.

[89] B. G. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-32, pp. 1243–1245, Dec. 1984.

[90] S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," *IEEE ASSP Magazine*, pp. 4–19, July 1989.

[91] C.-L. Su, C.-Y. Tsui, and A. M. Despain, "Low power architecture design and compilation techniques for high-performance processors," in *Compcon*, (San Francisco, California), pp. 489–498, Mar. 1994.

[92] E. N. Farag and M. I. Elmasry, "Low-power subband coding algorithm," in *IEEE International Conference of Acoustics, Speech and Signal Processing*, vol. 4, (Atlanta, Georgia), pp. 2116–2119, May 1996.

[93] M. D. Ercegovace and T. Lang, *Division and Square Root Digit-Recurrence Algorithms and Implementations.* Kluwer Academic Publishers, 1994.

[94] A. Avizienis, "Signed digit number representation for fast parallel arithmetic," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 389–400, Sept. 1961.

[95] M. D. Ercegovace and T. Lang, "On-the-fly conversion of redundant into conventional representation," *IEEE Transactions on Communications*, vol. C-36, pp. 895–897, July 1987.

[96] F. N. Najm, "Towards a high-level power estimation capability," in *Proceedings of 1995 International Symposium on Low Power Design*, pp. 87–92, Apr. 1995.

[97] E. N. Farag, M. A. Hasan, and M. I. Elmasry, "Low-power radix 2 division algorithm with minimum add/sub operations," in *SPIE Advanced Signal Processing Algorithms, Architectures and Implementations VI*, vol. 2846, (Denver, Colorado), pp. 39–50, Aug. 1996.

[98] E. N. Farag, M. A. Hasan, and M. I. Elmasry, "Minimizing add/sub operations in a radix 2 division algorithm," *IEEE Transactions on Computers.*

[99] D. Wong, B. H. Juang, and A. H. Gray, "An 800 bit/s vector quantization LPC vocoder," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-30, pp. 770–779, Oct. 1982.

[100] E. Farag, M. Saleh, N. Elnady, and M. Elmasry, "Structure and network control of a hierarchical mobile network architecture," in *Phoenix Conference on Computers and Communications*, pp. 671–677, 1995.

[101] E. N. Farag, M. I. Elmasry, M. N. Saleh, and N. M. Elnady, "A two-level hierarchical mobile network: Structure and network control," *International Journal of Reliability, Quality and Safety Engineering*, vol. 3, pp. 325–351, Dec. 1996.

[102] B. M. Gordon and T. H. Meng, "A low power subband video decoder architecture," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. II–409–412, Apr. 1994.

[103] G. Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, vol. 34, pp. 30–44, Apr. 1991.

[104] W. Li and Y.-Q. Zhang, "Vector-based signal processing and quantization for image and video compression," *Proceedings of the IEEE*, vol. 83, pp. 317–335, Feb. 1995.

[105] M. Liou, "Overview of the px64 kbit/s video coding standard," *Communications of the ACM*, vol. 34, pp. 59–63, Apr. 1991.

[106] D. LeGall, "MPEG: A video compression standard for multimedia applications," *Communications of the ACM*, vol. 34, pp. 46–58, Apr. 1991.

[107] J. W. Woods and S. O'Neil, "Subband coding of images," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 34, pp. 1278–1288, Oct. 1986.

[108] W. Li and Y.-Q. Zhang, "A study of non-separable subband filters for video coding," in *SPIE Visual Communications and Image Processing*, vol. 1818, Part 1, (Boston, Massachusetts), pp. 233–240, Nov. 1992.

[109] J. Woods, *Subband Image Coding.* Boston: Kluwer Academic Publishers, 1991.

[110] N. I. Cho and S. U. Lee, "Fast algorithm and implementation of 2-D discrete cosine transform," *IEEE Transactions on Circuits and Systems*, vol. 38, pp. 297–305, Mar. 1991.

[111] M.-T. Sun, T.-C. Chen, and A. M. Gottlieb, "VLSI implementation of a 16 × 16 discrete cosine transform," *IEEE Transactions on Circuits and Systems*, vol. 36, pp. 610–617, Apr. 1989.

[112] Y.-H. Chan and W.-C. Siu, "On the realization of discrete cosine transform using the distributed arithmetic," *IEEE Transactions on Circuits and Systems*, vol. 39, pp. 705–712, Sept. 1992.

[113] E. N. Farag and M. I. Elmasry, "Low-power implementation of discrete cosine transform," in *Sixth Great Lakes Symposium on VLSI*, (Ames, Iowa), pp. 174–177, Mar. 1996.

[114] R. E. Crochiere, S. A. Webber, and J. L. Flanagan, "Digital coding of speech in sub-bands," *The Bell System Technical Journal*, vol. 55, pp. 1069–1085, Oct. 1976.

[115] M. Vetterli, "Multidimensional subband coding: Some theory and algorithms," *Signal Processing*, vol. 6, pp. 97–112, Apr. 1984.

[116] P. P. Vaidyanathan, "Quadrature mirror filter banks, M-Bank extensions and perfect-reconstruction techniques," *IEEE ASSP Magazine*, vol. 3, pp. 4–20, July 1987.

[117] M. Vetterli and D. LeGall, "Perfect reconstruction FIR filter banks: Some properties and factorizations," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, pp. 1057–1071, July 1989.

[118] P. P. Vaidyanathan, "Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial," *Proceedings of the IEEE*, vol. 77, Dec. 1989.

[119] G. Karlsson and M. Vetterli, "Theory of two-dimensional multirate filter banks," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, pp. 925–937, June 1990.

[120] P. P. Vaidyanathan, *Multirate systems and filter banks.* Englewood Cliffs, N.J.: Prentice Hall, 1993.

[121] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing.* Englewood Cliffs, NJ: Prentice Hall, 1983.

[122] H. Gharavi and A. Tabatabai, "Sub-band coding of digital images using two-dimensional quadrature mirror filtering," *IEEE Transactions on Circuits and Systems*, vol. 35, pp. 207–214, Feb. 1988.

[123] N. S. Jayant and P. Noll, *Digital Coding of Waveforms.* Englewood Cliffs, NJ: Prentice Hall, 1984.

[124] A. K. Al-Asmari and R. E. Ahmed, "VLSI architecture foe HDTV sub-band coding using GQMFs' filterbanks," *IEEE Transactions on Consumer Electronics*, vol. 41, pp. 1–11, Feb. 1995.

[125] P. H. Westerink, J. Biemond, and D. E. Boekee, "An optimal bit allocation algorithm for sub-band coding," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 757–760, Apr. 1988.

[126] P. H. Westerink, J. Biemond, and D. E. Boekee, "Evaluation of image sub-band coding schemes," in *Proceedings of the European Signal Processing Conference*, pp. 1149–1152, Sept. 1988.

[127] P. H. Westerink, J. Biemond, and D. E. Boekee, "Sub-band coding of images using predictive vector quantization," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1378–1381, Apr. 1987.

[128] P. H. Westerink, D. E. Boekee, J. Biemond, and J. W. Woods, "Subband coding of images using vector quantization," *IEEE Transactions on Communications*, vol. COM-36, pp. 713–719, June 1988.

[129] W. Li and Y.-Q. Zhang, "A study of vector transform coding of subband-decomposed images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, pp. 383–391, Aug. 1994.

[130] P. M. Aziz, H. V. Sorensen, and J. V. D. Spiegel, "An overview of sigma-delta converters," *IEEE Signal Processing Magazine*, pp. 61–84, Jan. 1996.

[131] A. M. Thurston, "Sigma-delta IF A-D converters for digital radios," *GEC Journal of Research*, vol. 12, pp. 76–85, Feb. 1995.

[132] J. C. Candy, B. A. Wooley, and O. J. Benjamin, "A voiceband codec with digital filtering," *IEEE Transactions on Communications*, vol. COM-29, pp. 815–830, June 1981.

[133] B. P. Brandt and B. A. Wooley, "A low-power area efficient digital filter for decimation and interpolation," *IEEE Journal of Solid State Circuits*, vol. 29, pp. 679–687, June 1994.

[134] R. Schreier and W. M. Snelgrove, "Decimation for bandpass sigma-delta analog-to-digital conversion," in *IEEE Internation Conference on Circuits and Systems*, vol. 3, pp. 1801–1804, May 1990.

[135] J. C. Candy, "Decimation for sigma delta modulation," *IEEE Transactions on Communications*, vol. COM-34, pp. 72–76, Jan. 1986.

[136] H. Meyr and R. Subramanian, "Advanced digital receiver principles and technologies for PCS," *IEEE Communications Magazine*, pp. 68–78, Jan. 1995.

[137] M. Rebeschini, N. R. V. Bavel, P. Rakers, R. Greene, J. Caldwell, and J. R. Haug, "A 16-b 160-kHz CMOS A/D converter using sigma-delta modulation," *IEEE Journal of Solid State Circuits*, vol. 25, pp. 431–440, Apr. 1990.

[138] R. Schreier and M. Snelgrove, "Bandpass sigma-delta modulation," *Electronics Letter*, vol. 25, pp. 1560–1561, Nov. 1989.

[139] S. Jantzi, R. Schreier, and M. Snelgrove, "Bandpass sigma-delta analog-to-digital conversion," *IEEE Transactions on Circuits and Systems*, vol. 38, pp. 1406–1409, Nov. 1991.

[140] S. A. Jantzi, W. M. Snelgrove, and P. F. Ferguson, "A fourth-order bandpass sigma-delta modulator," *IEEE Journal of Solid State Circuits*, vol. 28, pp. 282–291, Mar. 1993.

[141] L. Longo and B.-R. Horng, "A 15b 30kHz bandpass sigma-delta modulator," in *IEEE International Solid-State Circuits Conference*, (San Francisco, California), pp. 226–227, Feb. 1993.

[142] F. W. Singor and W. M. Snelgrove, "Switched-capacitor bandpass delta-sigma A/D modulation at 10.7 MHz," *IEEE Journal of Solid State Circuits*, vol. 30, pp. 184–192, Mar. 1995.

[143] A. J. Coulson, "A generalization of nonuniform bandpass sampling," *IEEE Transactions on Signal Processing*, vol. 43, pp. 694–704, Mar. 1995.

[144] E. N. Farag, R.-H. Yan, and M. I. Elmasry, "A novel parallel architecture for a switched-capacitor bandpass $\Sigma - \Delta$ modulator," in *Midwest Symposium on Circuits and Systems*, (Sacramento, California), Aug. 1997.

[145] E. N. Farag, R.-H. Yan, and M. I. Elmasry, "A parallel bandpass $\Sigma - \Delta$ modulator: Architecture and performance," *IEEE Transactions on Circuits and Systems*.

[146] E. N. Farag, R.-H. Yan, and M. I. Elmasry, "A programmable power-efficient decimation filter for software radios," in *International Symposium on Low Power Electronics and Design*, (Monterey, California), Aug. 1997.

[147] E. N. Farag, R.-H. Yan, and M. I. Elmasry, "Decimation filters for software radios: A power efficient design," *IEEE Transactions on Circuits and Systems.*

[148] E. N. Farag, R.-H. Yan, and M. I. Elmasry, "Power-efficient multiplier-accumulator design for FIR filters," in *IEEE Canadian Conference on Electrical and Computer Engineering*, vol. 1, (St. John's, Newfoundland, Canada), pp. 27–30, May 1997.

[149] E. N. Farag, R.-H. Yan, and M. I. ELmasry, "Decimation filters: Low-power design and optimization," in *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, (Victoria, British Columbia, Canada), Aug. 1997.

[150] A. Microelectronics, "HL400C 3 volt $0.5\mu m$ CMOS standard-cell library," 1995.

[151] C. S. Wallace, "A suggestion for parallel multipliers," *IEEE Transactions on Electronic Computers*, vol. EC-13, pp. 14–17, Feb. 1964.

[152] K. Hwang, *Computer Arithmetic: Principles, Architecture and Design.* John Wiley & Sons, 1979.

[153] J. C. Majithia and R. Kital, "An interative array for multiplication of signed binary numbers," *IEEE Transactions on Computers*, vol. C-20, pp. 214–216, Feb. 1971.

[154] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplier," *IEEE Transactions on Computers*, vol. C-22, pp. 1045–1047, Dec. 1973.

[155] A. K. Kwentus, H.-T. Hung, and A. N. Willson, "An architecture for high-performance/small area multipliers for use in digital filtering," *IEEE Journal of Solid State Circuits*, vol. 29, pp. 117–121, Feb. 1994.

[156] A. D. Booth, "A signed binary multiplication technique," *Quart. J. Mech. Appl. Math.*, vol. Volume 4 Part 2, pp. 236–241, 1951.

[157] N. Ling, "A high-speed parallel array multiplier," *International Journal of Mini and Microcomputers*, vol. 14 No. 3, pp. 139–146, 1992.

[158] E. N. Farag, R.-H. Yan, and M. I. Elmasry, "A programmable power-efficient multiplier accumulator array," *IEEE Transactions on Circuits and Systems.*

[159] D. Poornaiah, R. Haribabu, and M. Ahmad, "Design and VLSI implementation of a novel concurrent 16-bit multiplier-accumulator for DSP applications," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. 385–388, 1993.

[160] M. Ahmad and D. Poornalah, "Design of an efficient VLSI inner-product processor for real-time DSP applications," *IEEE Transactions on Circuits and Systems*, vol. 36, pp. 324–329, Feb. 1989.

[161] A. Microelectronics, "HS600C 5 volt lp600c 3 volt CMOS standard-cell libraries," 1994.

[162] A. Microelectronics, "HS500C 5 volt $0.5\mu m$ CMOS standard-cell library," 1995.

[163] L. T. Microelectronics group, "HS350C 5 volt $0.35\mu m$ CMOS standard-cell library," 1996.

[164] L. T. Microelectronics group, "HL350C 3 volt $0.35\mu m$ CMOS standard-cell library," 1996.

[165] E. N. Farag, R.-H. Yan, and M. I. Elmasry, "A novel digital channel selection algorithm with no pre-filter multiplier," *IEEE Transactions on Acoustics, Speech and Signal Processing.*

[166] K. Feher, *Advanced Digital Communications Systems and Signal Processing Techniques.* Englewood Cliffs, New Jersey: Prentice Hall, 1987.

[167] B. S. Haroun and M. I. Elmasry, "Architectural synthesis for DSP silicon compilers," *IEEE Transactions on Computer-Aided Design*, vol. 8, pp. 431–447, 1989.

[168] J. Rabaey, C. Chu, P. Hoang, and M. Potkonjak, "Fast prototyping of datapath-intensive architectures," *IEEE Design & Test of Computers*, pp. 40–51, June 1991.

[169] S. Wuytack, F. V. Catthoor, and H. J. D. Man, "Transforming set data types to power optimal structures," *IEEE Transactions on Computer-Aided Design*, vol. 15, pp. 619–629, June 1996.

[170] L. Goodby, A. Orailoglu, and P. M. Chau, "Microarchitectural synthesis of performance-constrained, low-power VLSI designs," in *IEEE International Conference on Computer Design*, pp. 323–326, 1994.

[171] P. E. Landman and J. M. Rabaey, "Activity-sensitive architectural power analysis," *IEEE Transactions on Computer-Aided Design*, vol. 15, pp. 571–587, June 1996.

[172] D. Marculescu, R. Marculescu, and M. Pedram, "Information theoretic measures for power analysis," *IEEE Transactions on Computer-Aided Design*, vol. 15, pp. 599–610, June 1996.

[173] K. van Berkel *et al.*, "A fully asynchronous low-power error corrector for the DCC player," *IEEE Journal of Solid State Circuits*, vol. 29, pp. 1429–1439, Dec. 1994.

[174] S.-J. Jou and I.-Y. Chung, "Low-power self-timed circuit design technique," *Electronics Letters*, vol. 33, pp. 110–111, 16th Jan 1997.

[175] Alta Group$^{\mathrm{TM}}$ of Cadence Design Systems, Inc., *Signal Processing WorkSystem Designer/BDE User's Guide.* 1996.

[176] S.-S. Hang and R. Jain, "Decimation filter compiler for oversampling A/D applications," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 5, pp. 537–540, 1992.

Publications and Patents Resulting from this Research

Published

1. E. N. Farag, R.-H. Yan, and M. I. Elmasry, "Power-efficient multiplier-accumulator design for FIR filters," in *IEEE Canadian Conference on Electrical and Computer Engineering*, vol. 1, (St. John's, Newfoundland, Canada), pp. 27–30, May 1997.

2. E. N. Farag and M. I. Elmasry, "Using carry save adders to reduce power dissipation," in *Eighth International Conference on Microelectronics*, (Cairo, Egypt), pp. 173–176, Dec. 1996.

3. E. N. Farag and M. I. Elmasry, "Low-power subband coding algorithm," in *IEEE International Conference of Acoustics, Speech and Signal Processing*, vol. 4, (Atlanta, Georgia), pp. 2116–2119, May 1996.

4. E. N. Farag, M. A. Hasan, and M. I. Elmasry, "Low-power radix 2 division algorithm with minimum add/sub operations," in *SPIE Advanced Signal Processing Algorithms, Architectures and Implementations VI*, vol. 2846, (Denver, Colorado), pp. 39–50, Aug. 1996.

5. E. N. Farag and M. I. Elmasry, "Low-power implementation of discrete cosine transform," in *Sixth Great Lakes Symposium on VLSI*, (Ames, Iowa), pp. 174–177, Mar. 1996.

Accepted

6. E. N. Farag, R.-H. Yan, and M. I. Elmasry, "A novel parallel architecture for

a switched-capacitor bandpass $\Sigma - \Delta$ modulator," in *Midwest Symposium on Circuits and Systems*, (Sacramento, California), Aug. 1997.

7. E. N. Farag, R.-H. Yan, and M. I. Elmasry, "A programmable power-efficient decimation filter for software radios," in *International Symposium on Low Power Electronics and Design*, (Monterey, California), Aug. 1997.

8. E. N. Farag, R.-H. Yan, and M. I. Elmasry, "Decimation filters: Low-power design and optimization," in *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, (Victoria, British Columbia, Canada), Aug. 1997.

Submitted

9. E. N. Farag, M. A. Hasan, and M. I. Elmasry, "Minimizing add/sub operations in a radix 2 division algorithm," submitted to *IEEE Transactions on Computers*.

10. E. N. Farag, R.-H. Yan, and M. I. Elmasry, "A parallel bandpass $\Sigma - \Delta$ modulator: Architecture and performance," submitted to *IEEE Transactions on Circuits and Systems*.

11. E. N. Farag, R.-H. Yan, and M. I. Elmasry, "Decimation filters for software radios: A power efficient design," submitted to *IEEE Transactions on Circuits and Systems*.

12. E. N. Farag, R.-H. Yan, and M. I. Elmasry, "A programmable power-efficient multiplier accumulator array," submitted to *IEEE Transactions on Circuits and Systems*.

13. E. N. Farag, R.-H. Yan, and M. I. Elmasry, "A novel digital channel selection algorithm with no pre-filter multiplier," submitted to *IEEE Transactions on Acoustics, Speech and Signal Processing.*