1995

FPGA implementation of adaptive interference canceler for periodic signal.

Tony Wai Sing. Yuen
University of Windsor

National Library of Canada

Acquisitions and Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada

Direction des acquisitions et des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Canada

FPGA Implementation of Adaptive Interference Canceler for Periodic Signal

by

Tony Wai Sing Yuen

A Thesis
Submitted to the Faculty of Graduate Studies through the
Department of Electrical Engineering in Partial Fulfillment
of the Requirement for the Degree of
Master of Applied Science
at the
University of Windsor

Windsor, Ontario
December, 1995

ISBN 0-612-11039-7

Canada

Name TONY YUEN WAI SING YUEN

Dissertation Abstracts International and *Masters Abstracts International* are arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation or thesis. Enter the corresponding four-digit code in the spaces provided.

Electronics and Electrical
SUBJECT TERM

0544 UMI
SUBJECT CODE

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS
Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION
General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523
Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS
Language
- General 0679
- Ancient 0289
- Linguistics 0290
- Modern 0291

Literature
- General 0401
- Classical 0294
- Comparative 0295
- Medieval 0297
- Modern 0298
- African 0316
- American 0591
- Asian 0305
- Canadian (English) 0352
- Canadian (French) 0355
- English 0593
- Germanic 0311
- Latin American 0312
- Middle Eastern 0315
- Romance 0313
- Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY
Philosophy 0422
Religion
- General 0318
- Biblical Studies 0321
- Clergy 0319
- History of 0320
- Philosophy of 0322

Theology 0469

SOCIAL SCIENCES
American Studies 0323
Anthropology
- Archaeology 0324
- Cultural 0326
- Physical 0327

Business Administration
- General 0310
- Accounting 0272
- Banking 0770
- Management 0454
- Marketing 0338

Canadian Studies 0385
Economics
- General 0501
- Agricultural 0503
- Commerce-Business 0505
- Finance 0508
- History 0509
- Labor 0510
- Theory 0511

Folklore 0358
Geography 0366
Gerontology 0351
History
- General 0578
- Ancient 0579
- Medieval 0581
- Modern 0582
- Black 0328
- African 0331
- Asia, Australia and Oceania 0332
- Canadian 0334
- European 0335
- Latin American 0336
- Middle Eastern 0333
- United States 0337

History of Science 0585
Law 0398
Political Science
- General 0615
- International Law and Relations 0616
- Public Administration 0617

Recreation 0814
Social Work 0452
Sociology
- General 0626
- Criminology and Penology 0627
- Demography 0938
- Ethnic and Racial Studies 0631
- Individual and Family Studies 0628
- Industrial and Labor Relations 0629
- Public and Social Welfare 0630
- Social Structure and Development 0700
- Theory and Methods 0344

Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES
Agriculture
- General 0473
- Agronomy 0285
- Animal Culture and Nutrition 0475
- Animal Pathology 0476
- Food Science and Technology 0359
- Forestry and Wildlife 0478
- Plant Culture 0479
- Plant Pathology 0480
- Plant Physiology 0817
- Range Management 0777
- Wood Technology 0746

Biology
- General 0306
- Anatomy 0287
- Biostatistics 0308
- Botany 0309
- Cell 0379
- Ecology 0329
- Entomology 0353
- Genetics 0369
- Limnology 0793
- Microbiology 0410
- Molecular 0307
- Neuroscience 0317
- Oceanography 0416
- Physiology 0433
- Radiation 0821
- Veterinary Science 0778
- Zoology 0472

Biophysics
- General 0786
- Medical 0760

EARTH SCIENCES
Biogeochemistry 0425
Geochemistry 0996
Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES
Environmental Sciences 0768
Health Sciences
- General 0566
- Audiology 0300
- Chemotherapy 0992
- Dentistry 0567
- Education 0350
- Hospital Management 0769
- Human Development 0758
- Immunology 0982
- Medicine and Surgery 0564
- Mental Health 0347
- Nursing 0569
- Nutrition 0570
- Obstetrics and Gynecology 0380
- Occupational Health and Therapy 0354
- Ophthalmology 0381
- Pathology 0571
- Pharmacology 0419
- Pharmacy 0572
- Physical Therapy 0382
- Public Health 0573
- Radiology 0574
- Recreation 0575
- Speech Pathology 0460
- Toxicology 0383

Home Economics 0386

PHYSICAL SCIENCES
Pure Sciences
Chemistry
- General 0485
- Agricultural 0749
- Analytical 0486
- Biochemistry 0487
- Inorganic 0488
- Nuclear 0738
- Organic 0490
- Pharmaceutical 0491
- Physical 0494
- Polymer 0495
- Radiation 0754

Mathematics 0405
Physics
- General 0605
- Acoustics 0986
- Astronomy and Astrophysics 0606
- Atmospheric Science 0608
- Atomic 0748
- Electronics and Electricity 0607
- Elementary Particles and High Energy 0798
- Fluid and Plasma 0759
- Molecular 0609
- Nuclear 0610
- Optics 0752
- Radiation 0756
- Solid State 0611

Statistics 0463

Applied Sciences
Applied Mechanics 0346
Computer Science 0984
Engineering
- General 0537
- Aerospace 0538
- Agricultural 0539
- Automotive 0540
- Biomedical 0541
- Chemical 0542
- Civil 0543
- Electronics and Electrical 0544
- Heat and Thermodynamics 0348
- Hydraulic 0545
- Industrial 0546
- Marine 0547
- Materials Science 0794
- Mechanical 0548
- Metallurgy 0743
- Mining 0551
- Nuclear 0552
- Packaging 0549
- Petroleum 0765
- Sanitary and Municipal 0554
- System Science 0790

Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY
General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

Abstract

This thesis describes the design, implementation and testing of an adaptive digital interference canceler for periodic signals. This canceler uses the least mean-square (LMS) adaptive filtering technique to eliminate interference from an input signal. An advantage of the adaptive interference cancelling approach is that it requires no prior knowledge of both the signal and the interference characteristics. A digital design approach is used in this thesis because of its reliability and accuracy. The canceler is implemented using a Xilinx FPGA chip. A prototype design demonstrated that the canceler was able to reduce the input interference power by about 40%.

To my parents

&

my love

Agnes

Acknowledgements

I wish to express my gratitude to my thesis supervisor and advisor, Dr. H. K. Kwan, who suggested to me the present project and introduced me to the subjects of adaptive signal processing and FPGA technology. Dr. Kwan has also provided me with invaluable technical advice, encouragement, and an enriching research environment throughout the course of this thesis.

I would like to thank the other two members of my thesis committee, Prof. P. H. Alexander for serving as my Department Reader, and Dr. M. Wang for serving as my Outside Department Reader. Their comments are gratefully acknowledged. In particular, I also like to thank Prof. Alexander for the advice in the thesis presentation.

I would like to acknowledge Mr. J. Hochreiter for his technical assistance and Mr. A. Johns for his help in providing me some equipment.

I would also like to acknowledge my fellow graduate students, in particularly, Andy Hung and Jacky Luk for research interactions and friendship.

I sincerely thank my family and particularly my parents for their encouragement and support. Special thanks are expressed to my friend, Agnes Chung, for her patience and understanding.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis presents the design methodology for an adaptive interference canceler (AIC) for periodic signals. The canceler can reduce interference power from a corrupted signal based on the principles of least mean-square (LMS) adaptive filtering [1]. This approach has the advantage of requiring no prior knowledge of the interference signal because the filter has the ability to adjust its own parameters automatically. Implementation details and stimulation results are given for a Field Programmable Gate Array (FPGA) prototype. This chapter provides a brief introduction and motivation of the research work conducted. In addition, the outline of the thesis is described.

1.1 Motivation

Separating a desired signal from background interference is an important problem in signal processing. In many environments, such as in aircrafts, helicopters, and concert halls, the characteristics of interference changes frequently. It is known that the presence of high levels of acoustic interference in an audio signal significantly reduces the intelligibility of the signal. Although the use of gradient (noise cancelling) microphones and a physical barrier such as oxygen facemasks reduces the noise problem somewhat, the ambient interference levels encountered in environments such as high performance tactical aircrafts are often severe enough that the performance of vocoders and automatic speech recognition systems are seriously affected.

Traditionally, a fixed weight filter is used for interference cancellation. However, the design of fixed filter must be based on prior knowledge of both signal and interference. If the properties of the interference are unknown, the fixed weight filter for interference cancellation is not a suitable approach. A useful method that requires more than one input source is adaptive noise cancellation (ANC) proposed by Widrow *et al.*[1]. The ANC method uses a second sensor, commonly called the reference, to gain additional information about the background interference. If

the reference sensor contains only a correlated version of the interference component in the primary sensor, and an interference corrupted signal is applied in the primary, the filter minimizes the interference power while leaving the desired signal unaffected. If the interference in the reference microphone is related to the noise in the primary microphone by a linear filter, then it is possible to cancel completely the noise corrupting the desired signal.

In this thesis, an adaptive digital interference canceler is implemented to process the corrupted periodic signal directly using a third generation of programmable logic devices FPGA. The results of this thesis show that the adaptive digital interference canceler performs satisfactorily.

1.2 Thesis Organization

This thesis is divided into six chapters. Chapter 2 provides the background information on the adaptive noise cancellation and the LMS algorithm. Chapter 3 presents the background material to understand FPGA technology. It includes a brief description about the evolution of programmable devices, FPGA architectures and CAD flow for implementation. Design methodology of the adaptive interference canceler is in Chapter 4. The structure of

different arithmetic units are also described. In Chapter 5, the test results of a FPGA implementation are presented and compared with computer simulation results. Finally, Chapter 6 summarizes the tasks undertaken by this thesis and shows some directions that can be taken for future work.

Chapter 2

Background Theory

This chapter provides a brief overview of the background theory relevant to this thesis. The concept of adaptive noise cancellation is presented followed by an brief introduction of adaptive filters and the adaptive LMS algorithm.

2.1 Concept of Adaptive Noise Cancellation

One of the most useful applications of adaptive filtering is the method of Adaptive Noise Cancellation (ANC). It was proposed by Widrow *et al.*[8] in 1975. Fig. 2.1 illustrates the basic model of the ANC. The signal x at the primary input is composed of a desired component s and a noise

component n, whereas the signal r

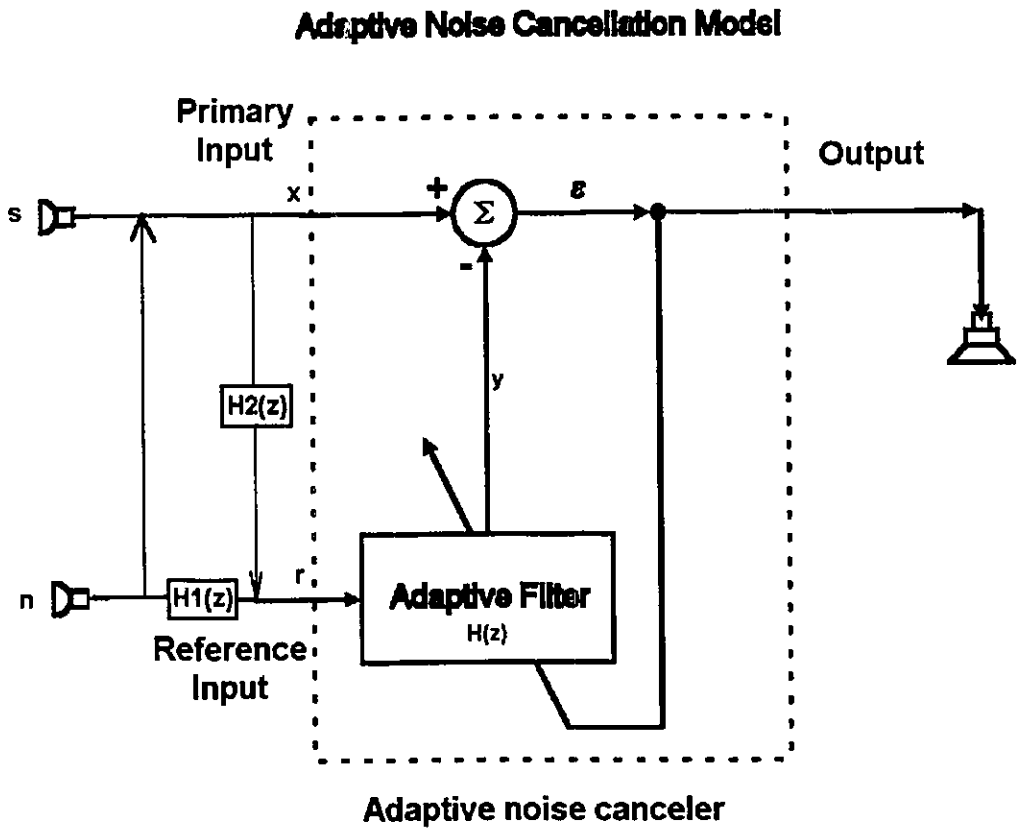


Figure 2.1 Model of Adaptive Noise Cancellation

at the reference input consists of a filtered version of the desired signal s' and noise signal n'. For simplicity, the discrete time index has been removed. The transfer functions $H_1(z)$ and $H_2(z)$ are assumed to be models of propagation channels for the noise and the signal in reference input. The r is filtered by $H(z)$ to produce the output y that is an estimate of the additive noise n. The estimated desired

signal ϵ is obtained by subtraction $\epsilon = X - y$ and it is also used to control the adaptive filter. If it is assumed that $|H_2(z)| \approx 0$ so that the signal r contains no signal s component and signal s is uncorrelated with both noise signal n and n', then the result of Widrow *et al.* shows that minimizing $E[\epsilon^2]$ will give the best least-square fit to the clean signal s. From Fig 2.1, the system output is

$$\epsilon = s + n - y \tag{2.1}$$

Squaring, one obtains

$$\epsilon^2 = s^2 + (n-y)^2 + 2s(n-y) \tag{2.2}$$

Taking expectations of both sides of (2.2), and realizing that s is uncorrelated with n and with y, yields

$$\begin{aligned} E[\epsilon^2] &= E[s^2] + E[(n-y)^2] + 2E[s(n-y)] \\ &= E[s^2] + E[(n-y)^2] \end{aligned} \tag{2.3}$$

The signal power $E[s^2]$ will be unaffected as the filter is adjusted to minimize $E[\epsilon^2]$. Accordingly, the minimum output power is

$$E_{min}[\epsilon^2] = E[s^2] + E_{min}[(n-y)^2] \tag{2.4}$$

Since minimizing $E[\epsilon^2]$ corresponds to minimizing $E[(n-y)^2]$, y is the best least-square estimate of noise n.

The success of the ANC is dependent on obtaining an

external reference noise input which satisfies the stated requirements. In some applications, the reference input contains signal components $|H_2(z)| \neq 0$. Assume the input signal and noise power spectra are $\Phi_{ss}(z)$ and $\Phi_{nn}(z)$ respectively. The spectrum of the reference input is related to the spectrum of the input *x as*,

$$\Phi_{rr}(z) = \Phi_{ss}(z)|H_2(z)|^2 + \Phi_{nn}(z)|H_1(z)|^2 \tag{2.5}$$

The cross-spectrum between the reference input and the primary inputs is identical to the cross-spectrum between the filter's input *r* and desired response *x*, which is

$$\Phi_{rx}(z) = \Phi_{ss}(z)H_2(z^{-1}) + \Phi_{nn}(z)H_1(z^{-1}) \tag{2.6}$$

When the adaptive process has converged, the unconstrained Wiener transfer function of the adaptive filter is

$$H(z) = \frac{\Phi_{ss}(z)H_2(z^{-1}) + \Phi_{nn}(z)H_1(z^{-1})}{\Phi_{ss}(z)|H_2(z)|^2 + \Phi_{nn}(z)|H_1(z)|^2} \tag{2.7}$$

The transfer function of the propagation path from the signal input to the noise-canceler output is $1-H_2(z)H(z)$ and that of the path from the noise input to the canceler output is $1-H_1(z)H(z)$. The spectrum of the signal component in the output is

$$\Phi_{ss\text{-}out}(z) = \Phi_{ss}(z)\ |1-H_2(z)H(z)|^2$$

$$= \Phi_{ss}(z) \left| \frac{[H_1(z)-H_2(z)]\Phi_{nn}(z)H_1(z^{-1})}{\Phi_{ss}(z)|H_2(z)|^2 + \Phi_{nn}(z)|H_1(z)|^2} \right|^2 \quad (2.8)$$

and that of the noise component is similarly

$$\Phi_{nn\text{-}out}(z) = \Phi_{nn}(z) \ |1-H_1(z)H(z)|^2$$

$$= \Phi_{nn}(z) \left| \frac{[H_2(z)-H_1(z)]\Phi_{ss}(z)H_2(z^{-1})}{\Phi_{ss}(z)|H_2(z)|^2 + \Phi_{nn}(z)|H_1(z)|^2} \right|^2 \quad (2.9)$$

The output signal-to noise density ratio is

$$\rho_{out}(z) = \frac{\Phi_{ss}(z)}{\Phi_{nn}(z)} \left| \frac{\Phi_{nn}(z)H_1(z^{-1})}{\Phi_{ss}(z)H_2(z^{-1})} \right|^2$$

$$= \frac{\Phi_{nn}(z)|H_1(z)|^2}{\Phi_{ss}(z)|H_2(z)|^2} \quad (2.10)$$

The output-to-signal density ratio can be expressed in terms of the signal-to-noise density ratio at the reference input, $\rho_{ref}(z)$. The spectrum of the signal component in the reference input is

$$\Phi_{ss\text{-}ref}(z) = \Phi_{ss}(z)|H_2(z)|^2 \quad (2.11)$$

and that of the noise component is similarly

$$\Phi_{nn\text{-}ref}(z) = \Phi_{nn}(z)|H_1(z)|^2 \quad (2.12)$$

The signal-to-noise density ratio at the reference input is

$$\rho_{ref}(z) = \frac{\Phi_{ss}(z)\,|H_2(z)|^2}{\Phi_{nn}(z)\,|H_1(z)|^2} \tag{2.13}$$

The output signal-to-noise density ratio is

$$\rho_{out}(z) = \frac{1}{\rho_{ref}(z)} \tag{2.14}$$

This equation is known as the signal-to-noise inversion principle. Assuming the adaptive solution to be unconstrained and the noise in the primary and reference inputs to be mutually correlated, the SNR of the ANC system output is the inverse of the SNR of the reference.

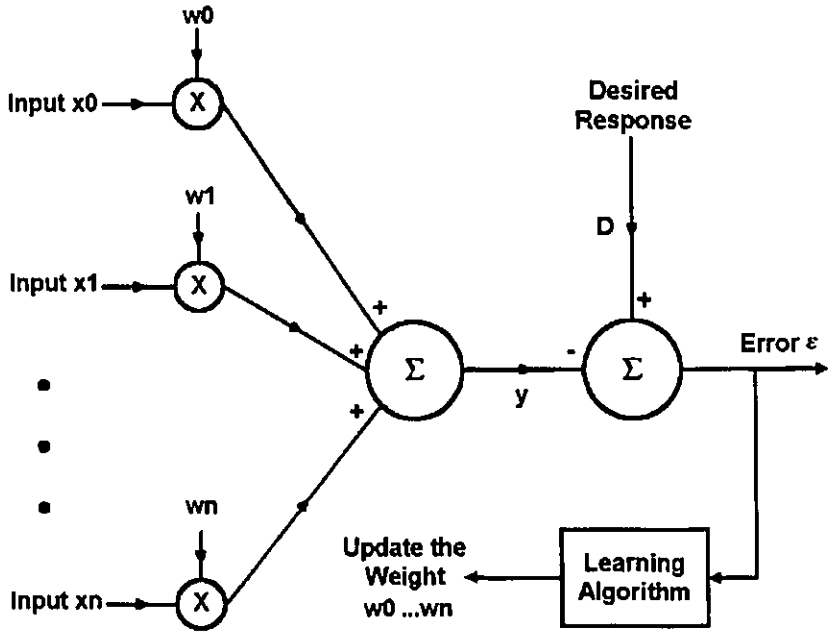


Figure 2.2 Multiple-input Adaptive FIR Filter with Desired Response and Error Signals

2.2 Adaptive Nonrecursive Filter

An adaptive nonrecursive or FIR filter, sometimes called an adaptive linear combiner, is fundamental to adaptive signal processing. It is the single most important element in adaptive or learning processes. Because of its nonrecursive structure, the adaptive FIR filter is relatively easy to understand. A general form of adaptive FIR filter is shown in Fig. 2.2. It consists of an input signal vector with elements x_0, x_1, x_2 ... x_n ($X = x_0$, x_1, x_2 ... x_n), a corresponding set of adjustable weights w_0, w_1, w_2 ... w_n ($W = w_0$, w_1, w_2 ... w_n), a summing unit, and a single output signal y. The procedure for adjusting the weights is called a "weight adjustment" or "adaptation" procedure. The weights are updated based on the ϵ and the learning algorithm.

There are two forms of the input vector X. First they may be considered to be simultaneous inputs from $n+1$ different signal sources. Second, the elements in vector X may be considered to *be* $n+1$ sequential samples for the same signal source. The structure of a single input FIR filter is shown in Fig. 2.3. This thesis focuses on a single input FIR filter.

For multiple input:

$$X_k = [\ x_{0k}\ x_{1k}\ x_{2k}\ \cdots\ x_{nk}\ ]^T \tag{2.15}$$

$$y_k = \sum_{l=0}^{n} w_{lk}\, x_{lk} \tag{2.16}$$

Corresponding to (2.16), we have a weight vector

$$W_k = [\ w_{0k}\ w_{1k}\ w_{2k}\ \cdots\ w_{nk}\ ] \tag{2.17}$$

Equation (2.17) can be expressed in vector notation

$$y_k = X_k^T\, W_k \tag{2.18}$$

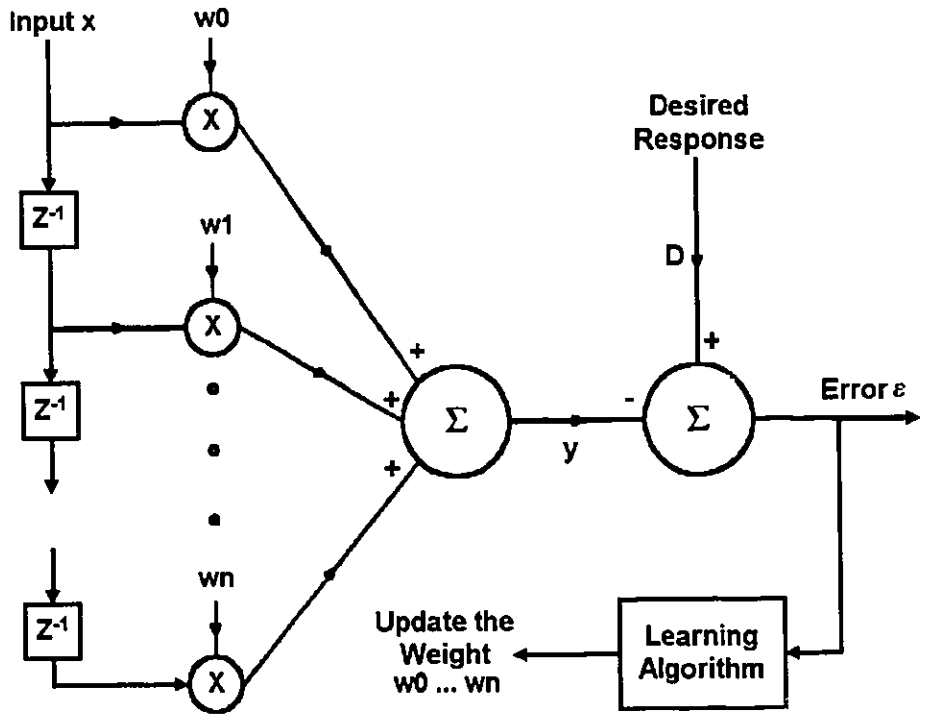


Figure 2.3 Single-input Adaptive Transversal Filter with Desired Response and Error Signal

For single input:

$$X_k = [\ x_k\ x_{k-1}\ x_{k-2}\ \cdots\ x_{k-n}\ ]^T \tag{2.19}$$

$$y_k = \sum_{l=0}^{n} w_{lk}\ x_{k-l} \tag{2.20}$$

Error signal is

$$\epsilon_k = D_k - y_k \tag{2.21}$$

where k is the time index.

2.3 Least-Mean-Squares Adaptive Algorithm

The purpose of the adaptive algorithm is to adjust the weights of the adaptive filter to minimize the mean-square error ϵ. For stationary input signals and a desired response, the mean-square error is a quadratic function of the weights with a single fixed minimum point [2],[3]. The least-mean-squares (LMS) algorithm [1],[3],[4],[5] is a widely used adaptive algorithm for finding close approximate solutions in real time. The algorithm does not require specific measurements of correlation functions and complex arithmetic operations. Accuracy is limited by statistical sample size because the weight values found are based on real-time measurements of input signals.

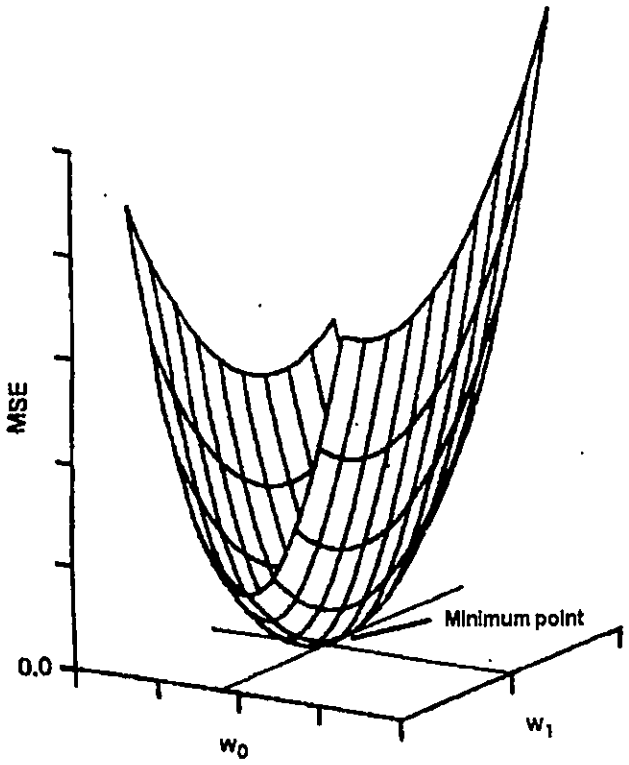


Figure 2.4 A Two-dimensional Quadratic Performance Surface of Adaptive Filter

The LMS is a gradient search algorithm using the steepest descent, the estimation of the gradient is based on the instantaneous value of the error. According to LMS, the "next" weight vector W_{k+1} is equal to the "present" weight vector W_k plus a change proportional to the negative gradient:

$$W_{k+1} = W_k - \mu \nabla_k \tag{2.22}$$

The parameter μ is the factor that controls stability and rate of convergence. ∇_k represents the true gradient at the k^{th}

iteration. The LMS algorithm estimates an instantaneous gradient ∇_k' by assuming that ϵ_k^2 is an estimate of the mean-square error and by differentiating ϵ_k^2 with respect to W. The formulas for the true and estimated gradients are given as follows:

$$\nabla_k \triangleq \begin{bmatrix} \partial E[\epsilon_k^2] / \partial w_0 \\ \vdots \\ \partial E[\epsilon_k^2] / \partial w_n \end{bmatrix} \tag{2.23}$$

$$\nabla_k' = \begin{bmatrix} \partial \epsilon_k^2 / \partial w_0 \\ \vdots \\ \partial \epsilon_k^2 / \partial w_n \end{bmatrix} = 2\epsilon_k \begin{bmatrix} \partial \epsilon_k / \partial w_0 \\ \vdots \\ \partial \epsilon_k / \partial w_n \end{bmatrix} = -2\epsilon_k X_k \tag{2.24}$$

To replace the true gradient of (2.23) with the estimated gradient in (2.24) gives the Widrow-Hoff LMS algorithm:

$$W_{k+1} = W_k + 2\mu\epsilon_k X_k \tag{2.25}$$

This algorithm is simple and easy to implement in a practical system without squaring, averaging and differentiation. It has been shown [9] that starting with an arbitrary weight vector, the algorithm will converge in the mean and will remain stable as long as the parameter μ is greater than zero but less than the reciprocal of the largest eigenvalue of the matrix R:

$$R = E[X_n X_n^T] \tag{2.26}$$

2.4 ANC without an External Reference Input

In some situations, no external reference input free of the signal is available. If an input signal is consisted of a multiple of sinusoidal signals and is corrupted by some broadband signals. It is possible to construct an ANC system without an external reference. A delayed version of the input signal is used as the reference in this case. The block diagram of the ANC without the external reference input is shown in Figure 2.5. The approach is to used a fixed delay Δ to cause the broadband signal components in the reference input to become decorrelated from those in the primary input.

$$E\{ s(n)*s(n-\Delta) \} \approx 0 \qquad (2.27)$$

The chosen delay Δ must be sufficient to decorrelate the broadband components. Because of the periodic nature of the periodic components, they will remain correlated with each other.

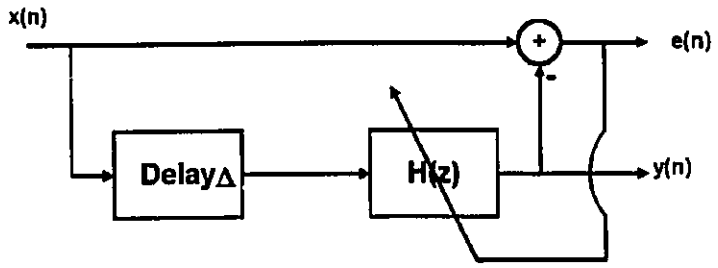


Figure 2.5 Block Diagram of ANC without External Reference Input

2.5 Summary

In this chapter, the basic concepts of the ANC, adaptive FIR filters and the LMS algorithm have been introduced. The ANC requires a reference signal to cancel the noise from a primary signal. It greatly improves the signal-to-noise ratio of noisy signals under certain conditions. The advantage of the ANC is that it requires little or no prior knowledge of signal or noise characteristics. Due to its simple structure, it is suitable for digital implementation as demonstrated in this thesis.

Chapter 3

Introduction to FPGA Technology

This chapter provides a brief introduction to FPGA technology. It begins by describing the evolution of programmable devices, and follows with a brief discussion on programming elements, FPGA examples and the related CAD flow for the implementation of circuits.

3.1 Evolution of Programmable Device

Digital system design has changed a great deal in the last ten years. Programmable devices, which are general-purpose chips that can be configured for different

applications, have become a key role in the design of digital hardware. A programmable Read-Only Memory (PROM) was the first type of programmable device used widely. It is a one-time programmable device that consists of an array of read-only cells. A pre-defined truth-table in the form of a logic circuit is stored in memory cells. By using the PROM's address lines as the circuit's inputs, the circuit's outputs are based on the stored bits. With this strategy, any truth-table function can be implemented. A PROM is suitable for computer memories in microcontroller based systems.

There are two basic versions of PROMs: mask-programmable and field programmable. The first is configured by the manufacturer. It provides a high speed of operation because connections within the device can be hardwired during the manufacturing process. In contrast, the speed of field-programmable devices is always slowed down by programmable switches. However, a field programmable chip can be programmed within days using inexpensive equipment. There are two enhanced versions of field programmable devices, the Erasable Programmable Read-Only Memory and Electrically Erasable Programmable Read-Only Memory. Both of them can be erased and reprogrammed many times.

The second type of programmable device is called a Programmable Logic Device (PLD) which is specifically designed for implementing logic circuits. Programmable Array Logic

(PAL) is the most basic form of PLD. It contains a programmable AND-plane followed by a fixed OR-plane. The AND-plane can give the product of inputs as an output and the OR-plane can generate the sum of product terms. Hence, PAL is ideal for implementation of sum-of-products form of logic functions. The outputs of the OR-gates can be registered by a flip-flop optionally. The Programmable Logic Array (PLA) is a variant of PAL in that both planes are programmable. Both types of PLDs allow high operation speed but they can only implement small logic circuits due to their simple structures. The logic capacity is about 5000 logic gates. They are also available in either mask-programmable or field-programmable versions.

Besides PLDs, Mask-Programmable Gate Arrays (MPGAs) are also widely used in industry. An MPGA consists of rows of transistors that can be interconnected to form a desired logic circuit. User specified connections are available between rows (to connect basic gates together) and within rows (to construct basic logic gates). All mask layers of the chip are pre-defined by the manufacturer, except for those that specify the final metal layers. These metal layers are customized to connect the transistors in the array during fabrication in order to form the desired circuit. The advantage of MPGAs is that they allow the implementation of much larger circuits. However, they lack of the instantaneous programmability and

have a large non-recurring engineering cost.

In order to take advantage of both PLDs and MPGAs, Field Programmable Gate Arrays (FPGAs) were introduced by Xilinx Company in 1985. FPGAs combine the programmability of PLDs and the scalable interconnection structure of MPGAs. Hence, present programmable devices have a much higher logic density.

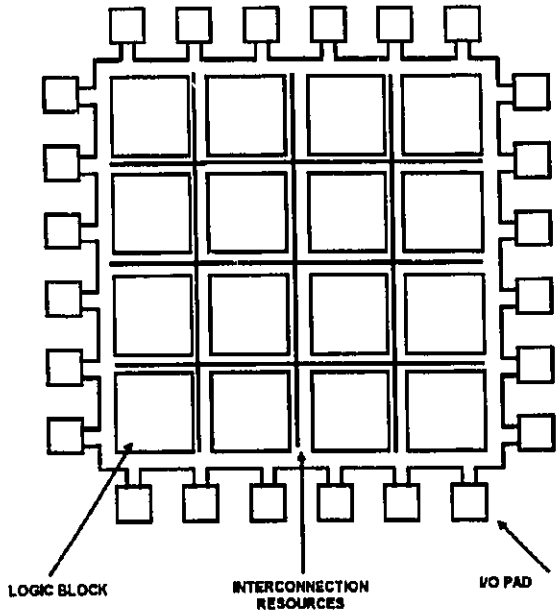


Figure 3.1 Conceptual FPGA

3.2 Structure of FPGA

FPGAs were introduced by the Xilinx Company in 1985. They are programmable devices like PALs, in that the interconnections between the elements are user-programmable without the use of an integrated circuit fabrication facility.

Also like an MPGA, an FPGA includes an array of uncommitted elements that can be interconnected in a general way. Since then, a number of companies, such as Actel, Advanced Micro Devices (AMD), Motorola, AT&T and Crosspoint Solutions, have introduced many different FPGAs.

FPGAs are a collection of functionally complete or universal logic elements placed in an interconnection framework. Some parts look like two dimensional cell arrays with little interconnection channels running horizontally and vertically between the cells. It is shown in Figure 3.1. Other parts look more like stacked logic gates with programmable arrays, similar to regular PLDs. The interconnect contains segments of wire, where the segments may vary in lengths. By the programmable switches, logic blocks can connect different wire segments, or one wire segment to another. Logic circuits are implemented in a FPGA by partitioning the logic into individual logic blocks and then interconnecting the blocks as required via the switches.

The heart of FPGAs is the programmable logic block. The structure and content of a logic block are called its *architecture*. The design of the logic block should be as versatile as possible. Some FPGA logic blocks can be as simple as 2-input NAND gates or have a more complex structure, such as look-up tables (Xilinx's FPGA [7]) and multiplexers (Actel's FPGA [8]). In some FPGAs, a logic block compares

with an entire PAL-like structure. Most logic blocks also contain some type of flip-flop, to aid in the implementation of sequential circuits.

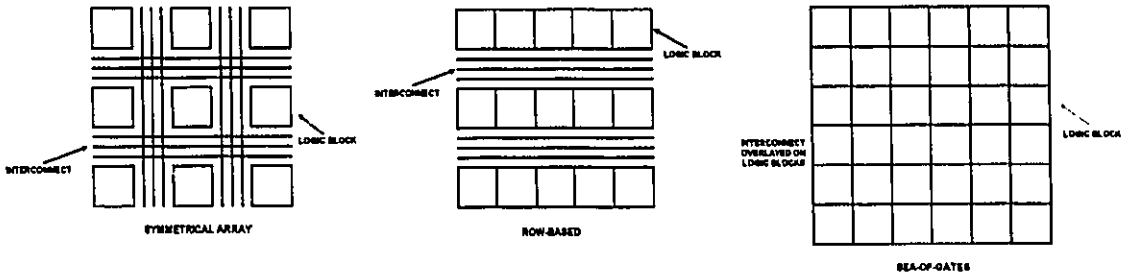


Figure 3.2 Three Classes of Commercial FPGA [6]

The structure and content of the interconnect in FPGA is called its *routing architecture*. The routing architecture includes programmable switches and wire segments. The programmable switches can be built using several methods including: pass-transistors controlled by static RAM cells, anti-fuses, EPROM transistors, and EEPROM transistors. As in the logic blocks, different companies provide different routing architectures. Some FPGAs offer a large number of simple connections between blocks but others provide less connections with more complex routes.

3.3 Programming Technologies

The instant programmability of FPGA depends on the programmable devices of a FPGA chip, including programmable logic blocks and *programmable elements*. It is important to understand how the programmable devices work. The term *programmable elements* refer to the entities that allow programmable connections between wire segments. Generally, a typical FPGA may contain more than 100,000 programmable elements. The design of programmable elements depends on the application in which a FPGA is used. For example, some programmable elements are non-volatile and some elements can be re-configured without being removed from the circuit board. No matter which design is used, the programming elements can be configured in one of two states: ON or OFF. Due to the huge number of programmable elements in a chip, the design of a programmable element should have the following properties:

- the chip area of programmable element should be as small as possible,
- the programmable element should have a low ON resistance and a very high OFF resistance,
- the programmable element should contribute low parasitance to the wiring resources to which it is attached,
- the programmable element should be reliably manufactured in large volume on a chip.

There are four types of programming elements that are currently used in commercial FPGAs. They are static RAM cells, anti-fuses, EPROM transistors, and EEPROM transistors. Each one of them has different properties and different fabrication procedures. The details of each of the programming elements will be described in the following.

3.3.1 Static RAM Programming Technology

Static RAM programming technology uses SRAM cells to control pass-transistors, transmission gates or multiplexers in programmable connections. Figure 3.3a shows the use of a static RAM cell to control a CMOS pass-transistor. Also the SRAM can control both the n-channel and p-channel transistors of a full transmission gate (Figure 3.3b). In the OFF state, the pass-gate forms a very high resistance between the two wires to which it is attached. In the ON state, it creates a low resistance connection between the two wires. Besides the above two approaches, several SRAMs can be used to control the select inputs on a multiplexer. It is shown in Figure 3.3c. This design is used to optionally connect one of several wires to a single input of a logic block.

Due to volatility of SRAM, static RAM FPGA must be re-configured when the power is applied to the chip. Hence, permanent storage, such as a ROM or a disk, is required to

store the programming information. Compared with other programming technology, this technology requires a relatively large chip area. This is because at least five transistors are needed for each RAM cell and the transistors for the pass-gate or multiplexers. The major advantage of this technology is that FPGAs using this technology can be reconfigured very quickly and the programmable elements can be produced using standard CMOS process technology. FPGA's produced by Algotronix, Concurrent Logic, Plessey Semiconductors, and Xilinx use static RAM programming technology.

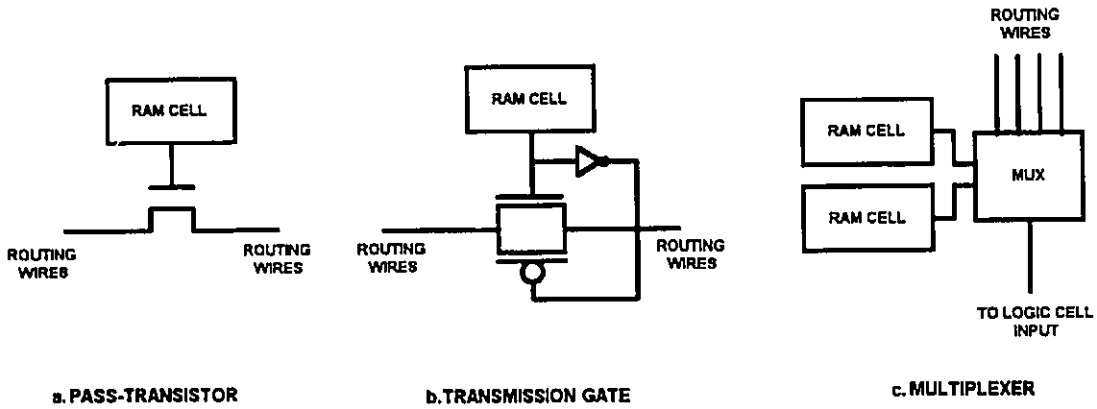


Figure 3.3 Static RAM Programming Technology [6]

3.3.2 Anti-fuse Programming Technology

Anti-fuse programming technology is used in FPGAs offered by Actel Corp., QuickLogic and Crosspoint Solutions. Although

the implementation of these elements is different from company to company, their function is the same. An anti-fuse element normally stays in a high-impedance state but can be "fused" into a low-impedance state when programmed by a high voltage.

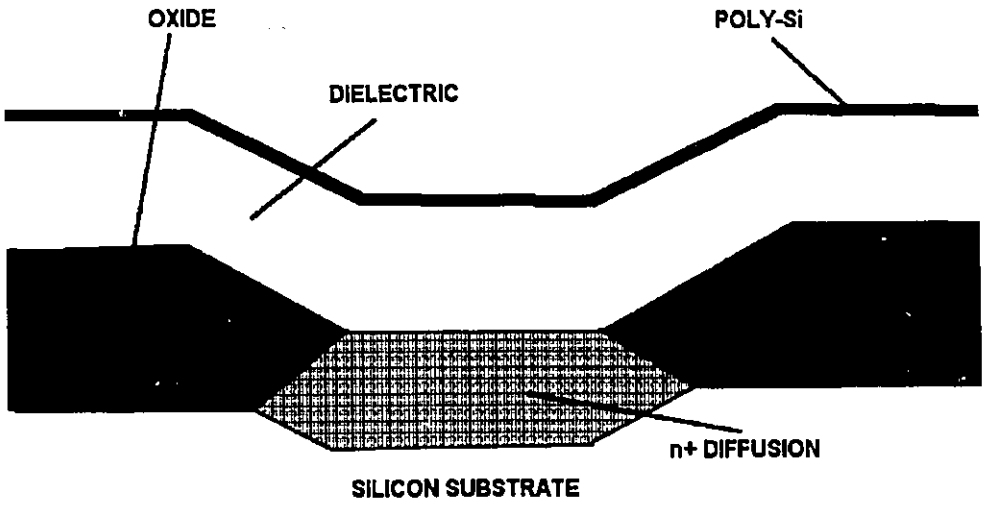


Figure 3.4 PLICE Anti-fuse Programming Technology [6]

PLICE [9] is an anti-fuse device used by Actel. It can be described as a square structure that consists of three layers: the top layer made of poly silicon, the middle layer, a dielectric (Oxygen-Nitrogen-Oxygen insulator), and the bottom layer, composed of positively-doped silicon (n+ silicon). This construction is shown in Figure 3.4. When a high voltage (18 V) is applied across the anti-fuse terminals and driving a current of about 5 mA through the device, enough

heat in the dielectric is generated to form a conductive link between the Poly-Si and n+ diffusion. Metal wires are connected to both the bottom layer and top layer. When it is in ON state, there is a resistance of about 300-500 ohms between the two metal wires. The PLICE anti-fuse requires special high-voltage transistors within FPGA to accommodate the necessary large voltages and currents, and three additional specialized masks to a normal CMOS process in fabrication.

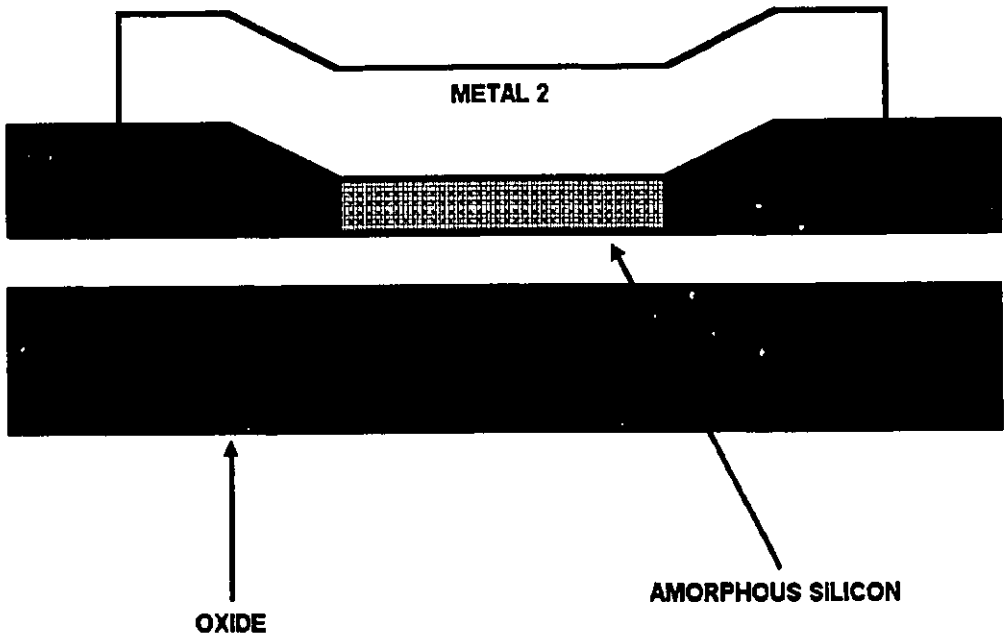


Figure 3.5 ViaLink Anti-fuse Programming Technology [6]

ViaLink [10] is an anti-fuse used by Quicklogic. It consists of a metal bottom layer, an alloy of amorphous

silicon for its middle layer, and a metal top layer. The structure of ViaLink anti-fuse is shown in Figure 3.5. When 10V is applied across its terminals, the amorphous silicon will create a conductive link between the bottom and top layers of metal. The resistance of the conductive layer is about 80 ohms between the two metal wires. This anti-fuse is manufactured using three extra masks above a normal CMOS process.

Compared to other programming technologies, anti-fuses require less chip area. However, they require large chip area for the high-voltage transistors that are needed to handle the high programming voltages and currents. Also, anti-fuse technology requires modification to the basic CMOS manufacture process.

3.3.3 EPROM & EEPROM Programming Technology

EPROM programming technology is used by Altera Corp [6] and Plus Logic. It consists of two gates, a floating gate and a select gate. The floating gate is located between the select gate and the transistor's channel, and it is not electrically connected to any circuitry. The structure of the EPROM programming circuitry is shown in Figure 3.6.

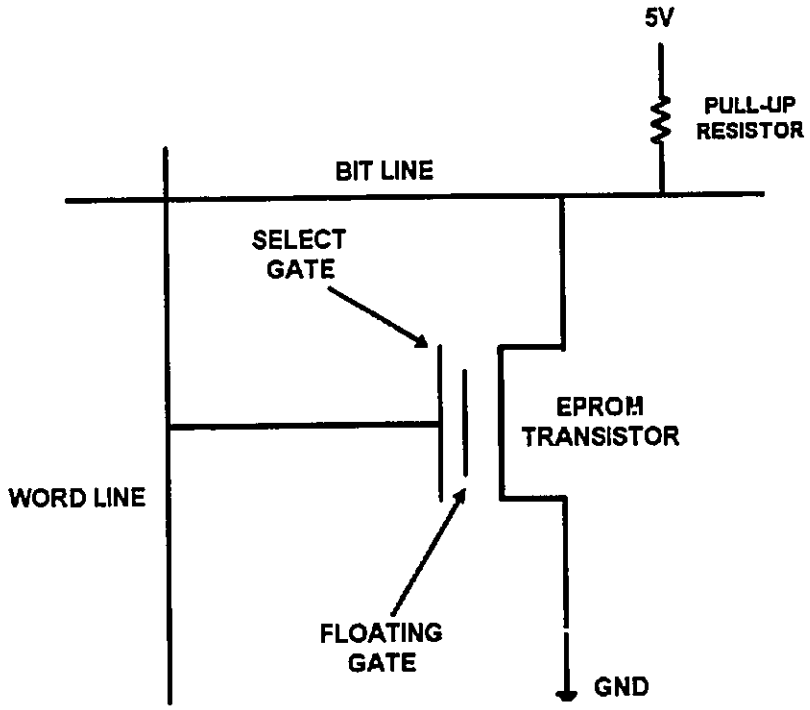


Figure 3.6 EPROM Programming Technology [6]

In its unprogrammed state, no charge exists in the floating gate and the transistor can be turned ON in the normal fashion by using the select gate. When the transistor is programmed by causing a large current to flow between the source and drain, a charge is trapped under the floating gate. This charge turns the transistor OFF permanently. Hence, the EPROM transistor can function as a programmable element. An EPROM transistor can be re-programmed by removing the trapped charge from the floating gate. By exposing the gate to ultraviolet light, the trapped charge is excited to the point where they can pass through the gate oxide into the substrate. One

advantage of the EPROM programming transistors is that they are reprogrammable and do not require external storage. However, the EPROM programming transistors cannot be reprogrammed in circuit.

Programming Technology	Volatile	Re-Prog	Chip-Area	R (ohm)	C (ff)
Static RAM Cells	yes	in circuit	large	1-2 k	10-20
PLICE Anti-fuse	no	no	small anti-fuse, large prog. tran.	300-500	3-5
ViaLink Anti-fuse	no	no	small anti-fuse, large prog. tran.	50-80	1.3
EPROM	no	out of circuit	small	2-4 k	10-20
EEPROM	no	in circuit	2x EPROM	2-4 k	10-20

Table 3.1 Characteristics of Programming Technologies [6]

The EEPROM approach is similar to the EPROM technology except that EEPROM transistors can be re-programmed in-circuit. The disadvantage of using EEPROM transistors is that they consume about twice the chip area as EPROM transistors and they require a voltage source for re-programming.

3.4 Applications of FPGA

FPGAs can be used in a wide variety of applications. They have two important advantages: FPGA have lower prototype costs, and shorter production times. Compared to MPGAs, FPGAs suffer from relatively low speed of operation, and lower logic density. This is because programmable switches in FPGA have significant propagation delay, and the programmable switches as well as associated programming circuity require a great deal of chip area. A direct comparison with MPGAs indicates that a typical circuit will be roughly three times slower and 8 to 12 times less dense if implemented in FPGA. However, it can still be used in almost all of the applications that currently use MPGAs, PLDs and small scale integration logic chips. Some of the applications are listed below:

Application-Specific Integrated Circuits (ASICs): An FPGA is a completely general medium for implementing digital logic. They are particularly suited for implementation of ASICs. A printer controller, a graphics engine and network transmitter/receiver are the typical examples of ASICs that can be effectively implemented by FPGA.

Implementation of Random Logic: Random logic circuitry is usually implemented using PALs. If the speed of the circuit

is not of critical concern, FPGA can be used for circuitry. At present, a FPGA circuit is equivalent to 10-20 PALs.

Prototyping: FPGAs are well suited for prototyping applications. The low cost of implementation and the short time needed to physically realize a given design, provide great advantages over more traditional approaches for building prototype hardware.

FPGA-Based Compute Engines: FPGAs have been widely used in computer hardware. These machines consist of a board which contains a number of FPGAs with the pins of neighboring chips connected. By loading different programs into the FPGA, the job can be run by hardware rather than software. This approach saves a lot of CPU time and, it provides high levels of parallelism.

Onsite Reconfiguration of Hardware: One of the most attractive feature of FPGAs is that they can be reprogrammable in-circuit. The function of the hardware can be changed within minutes, for example: computer equipment in a remote location that may have to be altered on site in order to correct a failure or perhaps a design error.

3.5 Xilinx's FPGA

Xilinx company introduced field-programmable gate arrays(FPGA) in 1985. Xilinx's FPGA consists of a two-dimensional array of programmable blocks, called *Configurable Logic Blocks (CLBs)*, with horizontal routing channels between rows of blocks and vertical routing channels between columns. It uses static RAM cells as programmable switches. There are three families of Xilinx FPGAs, called the XC2000, XC3000 and XC4000. Table 3.2 is a comparison among the families.

Family	# of CLBs	# of I/Os	Equivalent Gates	System Clock
XC2000	64-100	58-74	1200-1800	33 MHz
XC3000	64-480	64-176	2000-9000	80 MHz
XC4000	64-576	64-192	2000-13000	40 MHz

Table 3.2 Xilinx FPGA Logic Capacities [6]

3.5.1 Xilinx XC2000

The XC2000 CLB [7] consists of a four-input look-up table and a D flip-flop. The look-up table can generate any function of up to four variables or any two functions of three variables. Both of the CLB outputs can be combinational, or one output can be registered. The routing architecture of

XC2000 employs three types of routing resources: *Direct Interconnect*, *General Purpose Interconnect* and *Long Lines*. The Direct Interconnect provides connections from the output of a CLB to its right, top, and bottom neighbours. For connections that span more than one CLB, the General Purpose

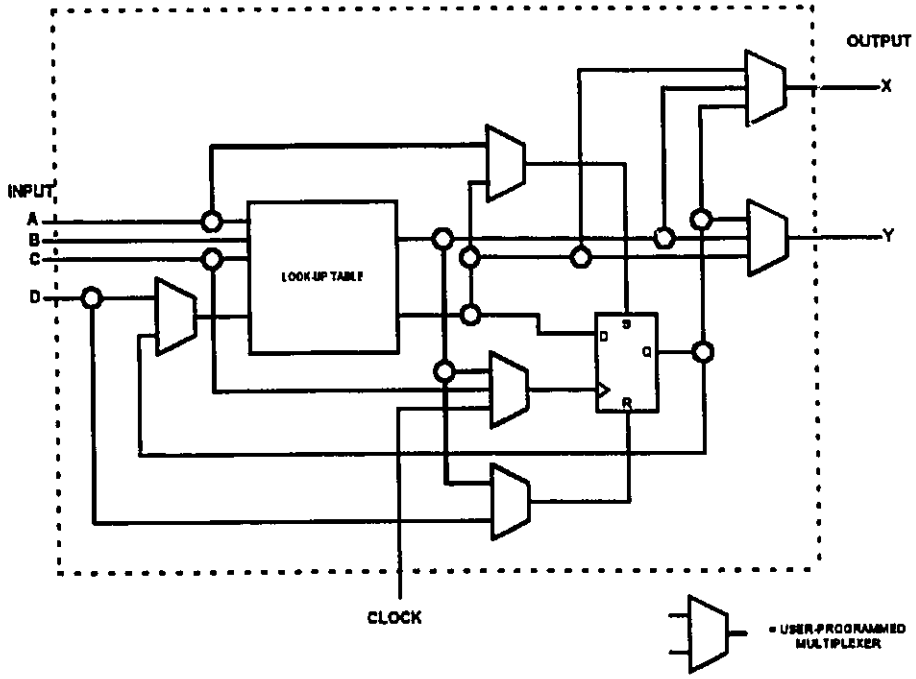


Figure 3.7 XC2000 CLB [6]

Interconnect provides horizontal and vertical wiring segments, with four segments per row and five segments per column. The general purpose interconnect will suffer from significant routing delays because the signal must pass through a routing switch at each switch matrix. Connections that are required to reach several CLBs with low skew can use the Long Lines,

which traverse at most one routing switch to span the entire length or width of the FPGA.

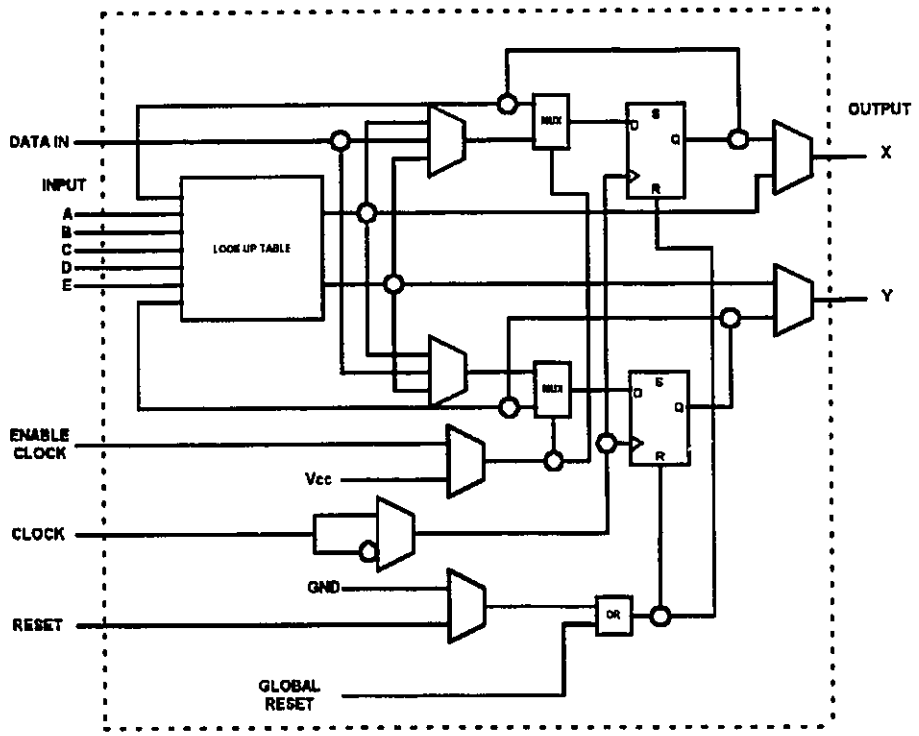


Figure 3.8 XC3000 CLB [6]

3.5.2 Xilinx XC3000

The XC3000 [7] is an enhanced version of the XC2000. It features a more complex CLB and more routing resources. The structure of XC3000 CLB is shown in Figure 3.8. The CLB consists of a look-up table that can implement any function of five variables, or any two functions of four variables. The CLB has two outputs, both of which may be either combinational

or registered. The routing architecture is similar to the XC2000. It has Direct Interconnect, General Purpose Interconnect and Long Lines. Each resource is enhanced; the Direct Interconnect can additionally reach the left neighbour of a CLB, the General Purpose Interconnect has extra wiring segments per row and there are more Long Lines.

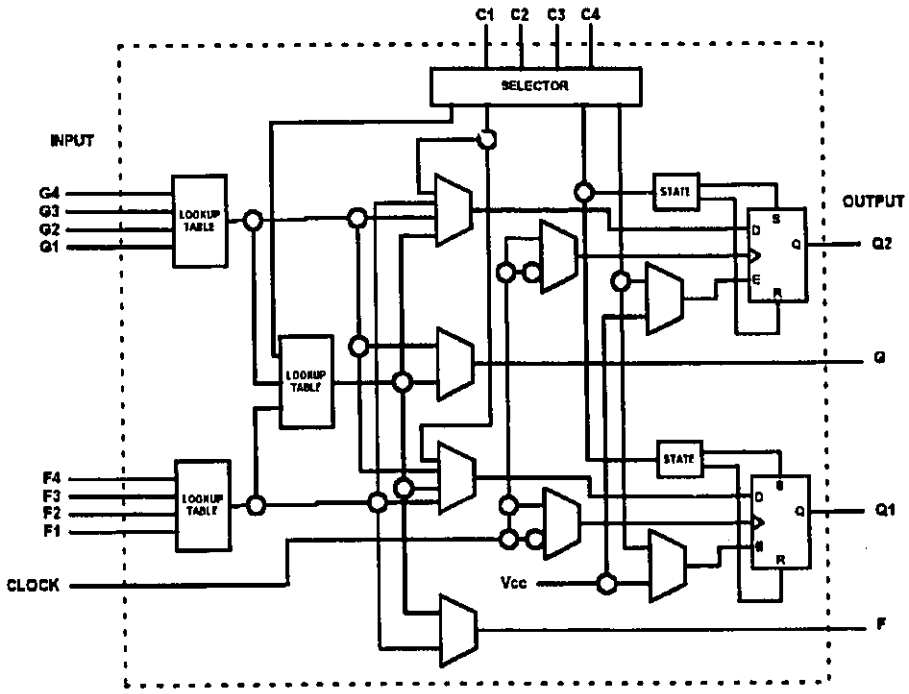


Figure 3.9 XC4000 CLB [6]

3.5.3 Xilinx XC4000

The XC4000 [7] consists of a two-stage arrangement of look-up tables that can implement two independent functions of

four variables, any single function of five variables or some functions of up to nine variables. The CLB has two outputs which may be either combinational or registered. The routing resource contains Single-length Lines and Double-length Lines and Long Lines. The Single-length Lines are intended for relatively short connections or those that do not have critical timing requirements. Double-length Lines are similar to the Single-length Lines except that each one passes through half as many switch matrices. This scheme offers lower routing delays for moderately long connections that are not appropriate for the low-skew Long Lines.

XC4000 has two extra features to improve performance: a fast carry logic and on-chip memory. Each CLB consists of high speed carry logic that can speed up addition operations. By using fast carry logic, a 16-bit adder requires nine CLBs and has a combinatorial delay of 20.5 ns. The XC4000 also includes on-chip static memory resources. The look-up table in a CLB can be configured as either a 16x2 or 32x1 bit array of Read/Write memory cells. On-chip RAM cells are very useful for designs such as DMA counters, LIFO stacks and FIFO buffers. In this thesis, the XC4000 is used.

3.6 FPGA Design Flow

To use FPGAs, one must have access to an efficient CAD

system. Figure 3.10 shows the design processing sequence for Xilinx FPGAs [11]. Generally the design flow can be divided into three steps: design capture, verification and implementation.

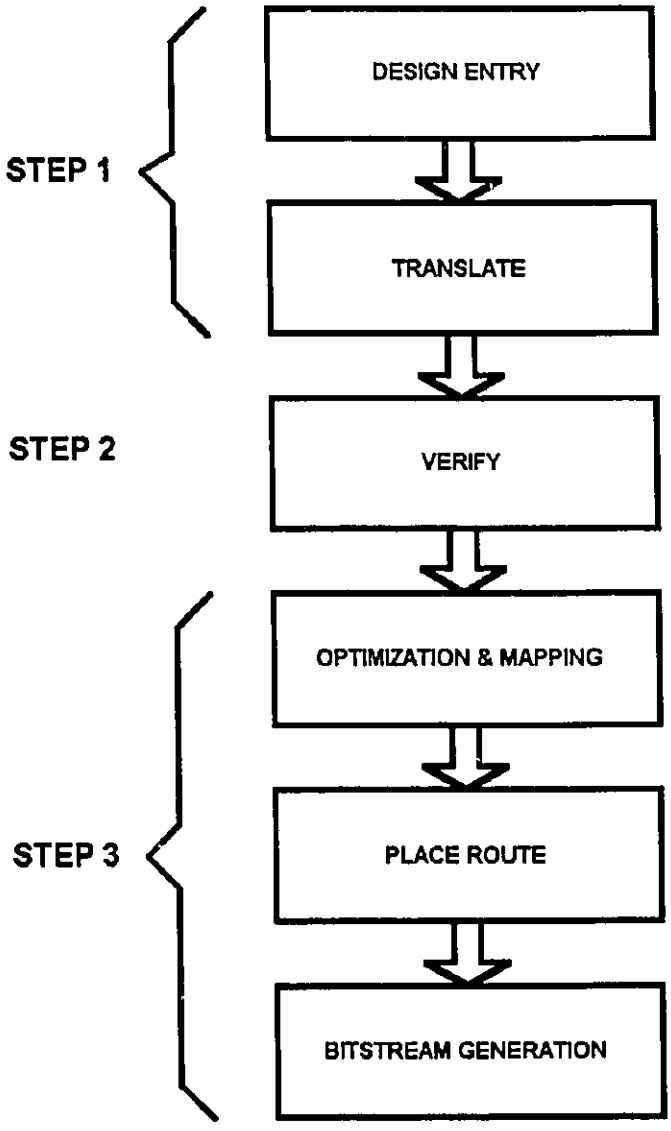


Figure 3.10 Xilinx Design CAD Flow [11]

The starting point for the design process is the design entry and capture. This step involves drawing a schematic using the schematic capture program "*VIEWlogic* [12]". Xilinx provides a lot of functional cells such as adders, registers and comparators that can speed up the design process. The design will automatically be translated to the Xilinx netlist format. The netlist passes to simulator "*VIEWsim*" for functional checking. Once this is verified, the design is compiled into a configuration bitstream to download into the FPGAs. During the compilation process, the CAD may attempt to minimize the total number of blocks required and the number of stages of logic blocks in time-critical paths. This process is called optimization and mapping, that are performed by an automatic place and route program, which assigns the FPGA's wire segments and chooses programmable switches to establish the required connections among the logic blocks. The choice of CLB and route depends on the choice of I/O pins, configuration of library cells and the length of the path. Upon successful completion of the placement and route steps, the bitstream can be downloaded to the FPGA chip for demonstration.

3.7 Summary

This chapter has provided a brief introduction to FPGA

technology. It covers most aspects of FPGAs including evolution of FPGAs, structure of FPGAs and design flow. The structure of Xilinx FPGAs was also briefly introduced.

Chapter 4

Adaptive Interference Canceler

This chapter provides a detailed description of an adaptive interference canceler (AIC) for periodic signals. The structure of AIC and design specifications will be discussed first, and then followed by the design of different functional circuits and arithmetic operators for the AIC. The software simulations and hardware testing results are discussed in Chapter 5.

4.1 Design of the AIC

The AIC plays an important role in interference filtering. The structure of the AIC affects the overall

system performance significantly. This section provides a description about the design of the AIC.

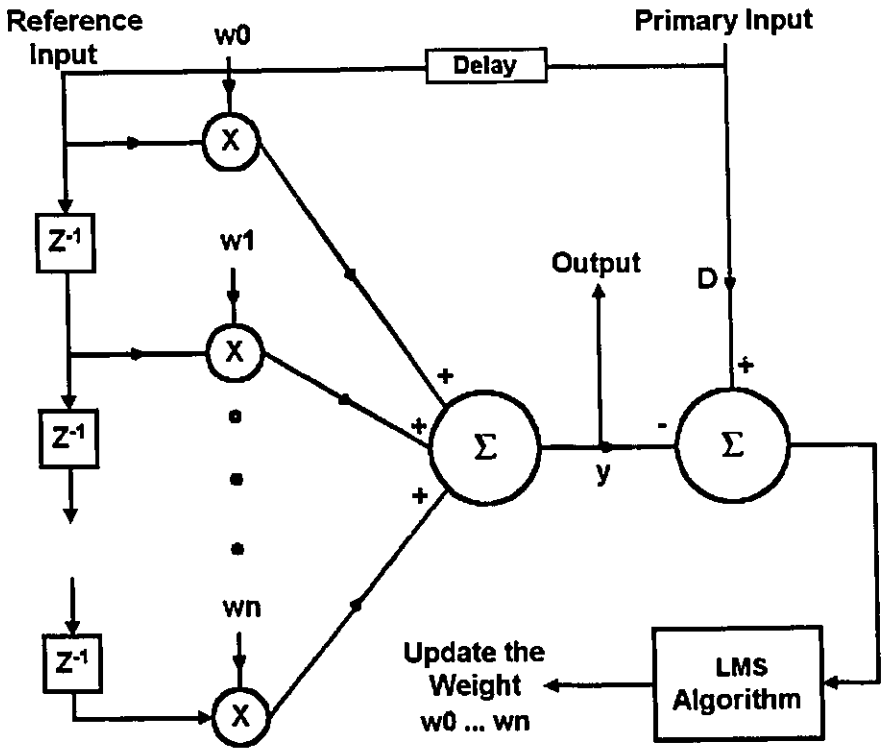


Figure 4.1 Structure of Adaptive Interference Canceler

The AIC is designed to extract a periodic signal from interference such as radio noise. In this case, no external reference input free of the signal is available. It might seem that the ANC could not be applied to reduce or eliminate this kind of interference. However, if a fixed delay is inserted in a reference input drawn directly from the primary input, as shown in Figure 4.1, the periodic signal can be

extracted. The number of delay elements must be large enough to cause the interference components in the reference input to become decorrelated from those in the primary input. The periodic signal will remain correlated with itself because of its periodic nature.

The AIC consists of an input, a transversal adaptive filter and an output. The weights are automatically updated by the LMS algorithm after each iteration. The structure of the AIC is illustrated in Figure 4.2. First the analog input is amplified and shifted up within 0 V to 4 V for the A/D conversion. Then the binary input signal is sent to the FPGA for signal extraction. The output of the FPGA is converted back to an analog signal as the AIC output.

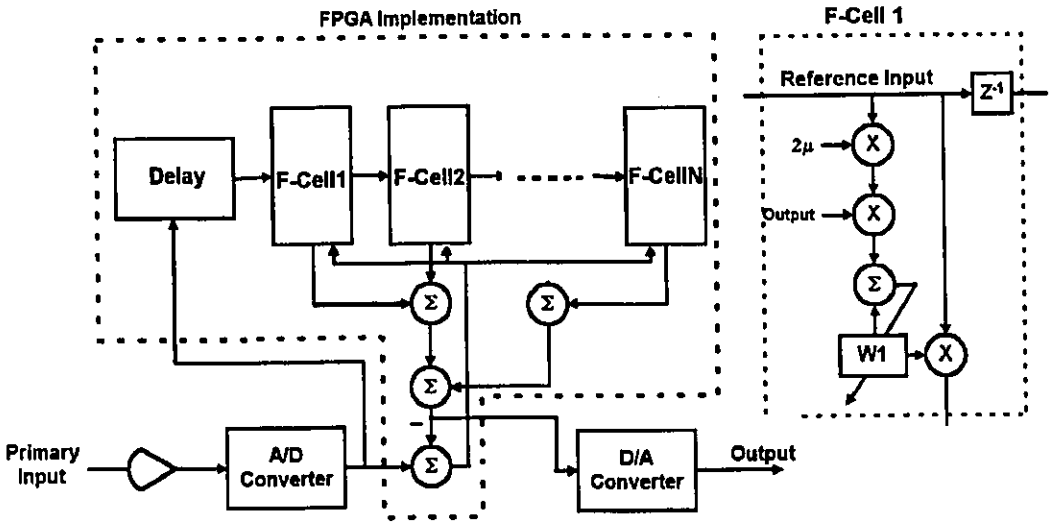


Figure 4.2 Structure of the AIC

The performance of the AIC depends on its precision, the

sampling rate of the A/D converter and the order of the adaptive filter. According to the sampling theorem, the sampling frequency should greater than twice of the frequency of sampling signal. For CD quality music, the sampling rate should be 44 kHz at 16 bit quantization. According to [13], 8 kHz at 8 bit quantization should be sufficient for speech signal. To reduce the round-off error resulting from arithmetic operation, arithmetic operations in the FPGA are performed at 16 bit precision. When the order of the filter is increased, the probability of cancelling out the interference is much higher. However, it requires more logic operators. In this thesis, the order of the filter depends mainly on the logic capacity of the FPGA chip so that its structure should be as flexible as possible. In Figure 4.2, the order of the AIC can be increased by adding a functional cell (F-Cell) in the schematic. It is a good approach for high operation speed and to allow for a future upgrade.

4.2 Design of Input & Output Circuits

The input circuit layout shown in Figure 4.3 is discussed in this section. It consists of an amplification circuit for an input signal before the A/D converter (ADC). When a signal is received from either a microphone or a line-in from another equipment, the magnitude of the signal is about 2 mV. It is

necessary to amplify the input signal before digital conversion. As the ADC only accepts a signal between 0 V and 5 V, the signal has to be shifted up to this range for conversion. The output circuit includes the D/A converter (DAC) only. Both conversions are performed by commercial IC packages. The detailed description of each process is presented in a later section.

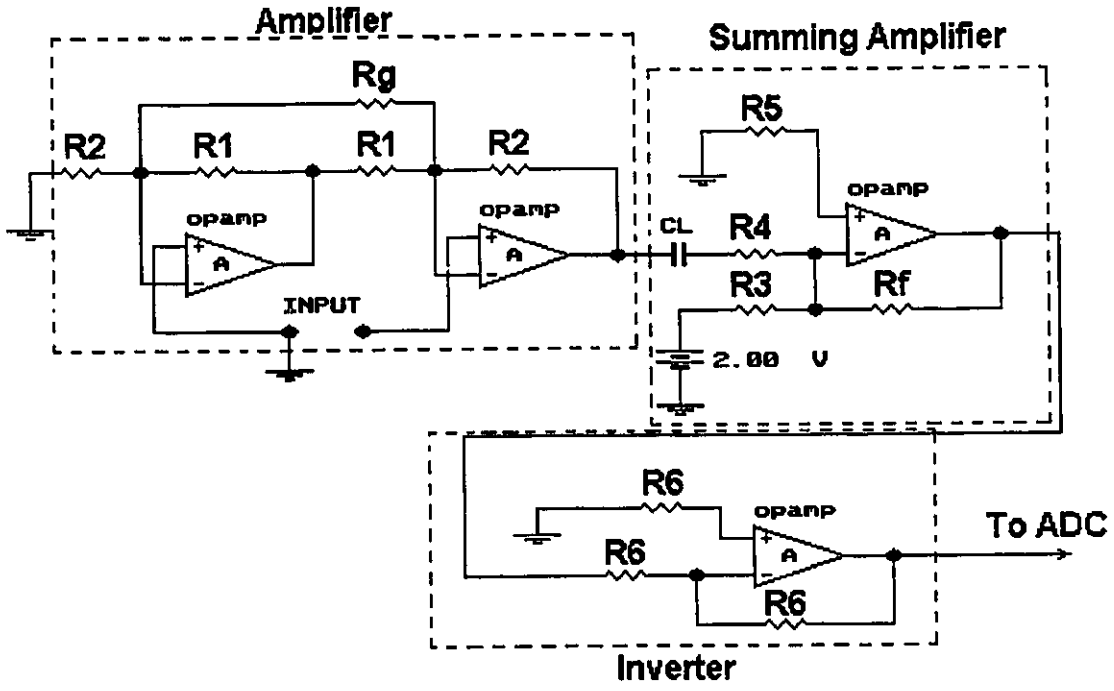


Figure 4.3 Input Circuit Layout

4.2.1 Input Amplification Circuit

To reduce the external common-mode noise from the input

signal, a differential-mode instrumentation amplifier (IA) is used. An IA is characterized by a low input offset voltage and drift, low noise voltage, adequate open-loop gain and adequate common-mode rejection ratio (CMRR). It uses one or more op-amps connected in a circuit which takes the difference between the two input voltages and yields an output that is proportional to this difference. Hence, most of the common-mode input voltage is rejected. For single op-amp configuration IA, the amplifier is dependent upon the equality of the resistor values for the circuit common-mode rejection, not the op-amp CMRR; but CMRR is the limiting factor in a circuit with perfect matching resistors. Therefore, the resistors must be precise and probably equal in value to within 0.1% or better [14]. The requirement of high precision resistors can be solved by using a dual op-amp configuration. Also it provides high input impedance and high CMRR. In this thesis, two op-amp configurations will be used. The circuit is shown in Figure 4.3. The gain of two op-amp IAs is shown below

$$A = 1 + \frac{R_2}{R_1} + 2 * \frac{R_2}{R_g} \qquad (4.1)$$

To suit various signal sources such as a microphone or a signal generator, a 10 kΩ variable resistor is used as R_g. In this thesis, R_1 and R_2 are equal to 1 kΩ and 10 kΩ

respectively. Hence, the minimum gain of the IA is 13 for this configuration. It should be enough to amplify various signals to 4 $V_{p\text{-}p}$; that is the full-scale analog voltage of the ADC.

After the input signals is amplified, it has to be shifted up 2 V to make its voltage within the range 0 V to 4 V, because the analog input range of ADC is 0 V to 4 V. A summing amplifier is used to add the amplified input signal to the 2 V DC signal. The output voltage is determined as

$$V_o = - \left(\frac{R_f}{R_3} V_1 + \frac{R_f}{R_4} V_2 \right) \qquad (4.2)$$

From eqn. 4.2, we notice that the polarity of the output is opposite to the inputs so that an unity-gain inverter is added to reverse the polarity of the output from summer. The circuit of the summing amplifier and unity-gain inverter is shown in Figure 4.3.

4.2.2 Successive-Approximation Analog-to-Digital Converter (SAC)

The successive-approximation ADC is one of the most widely used types of ADC. It has a fixed value of conversion time that is independent of the value of the analog input. The simplified block diagram of SAC is shown in Figure 4.4. The control logic adjusts the content of the register bit by

bit until the register data is the digital equivalent of the analog input within the resolution of the converter.

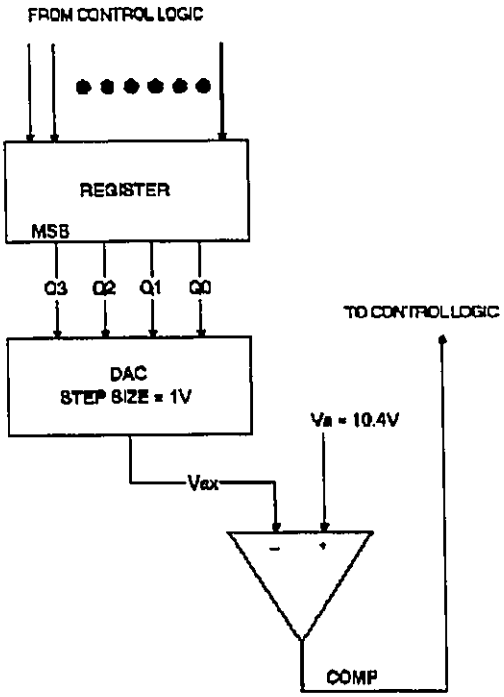


Figure 4.4 Simplified Block Diagram of Successive-approximation ADC [15]

A simple 4-bit converter with a step size of 1 V is chosen as an example of how the SAC works. Although the SAC used in this project is a 8-bit converter with a 15.6 mV step size, the operation will be the same. Clearly the four register bits feeding the DAC have weights of 8, 4, 2 and 1, respectively.

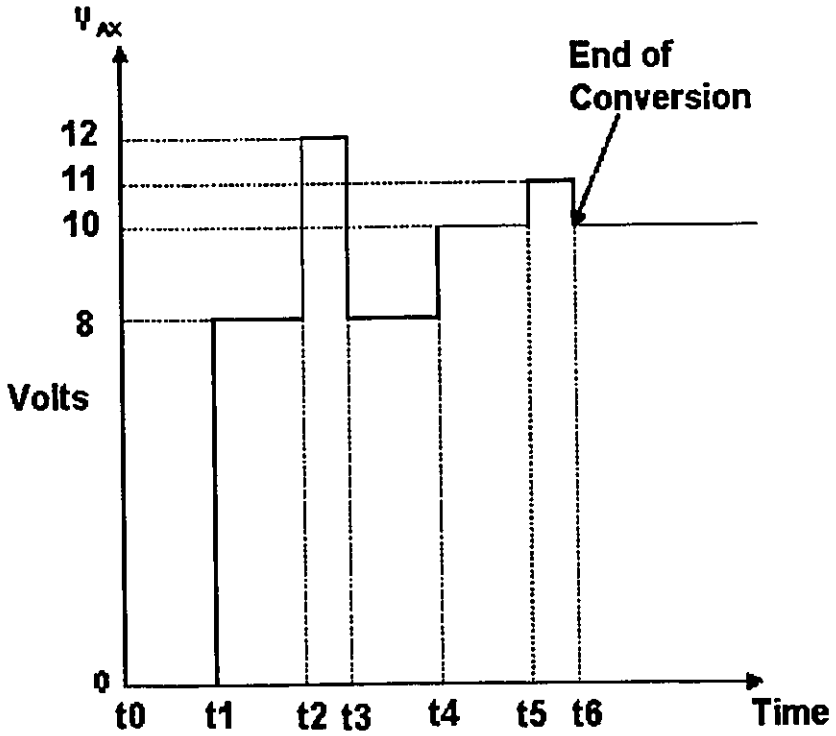


Figure 4.5 Timing Graph for SAC Operation [15]

Supposedly, the analog input is V_A = 10.4 V. At the beginning, the control logic clears all the register data to 0 so that Q_3=Q_2=Q_1=Q_0=0 ([Q] = 0000) and the DAC output V_{AX} = 0 V. This is indicated at time t_0 on the timing graph (Fig. 4.5). The comparator (COMP) output is high because of $V_{AX} < V_A$. At time t_1, the control logic sets the MSB of the register to 1 so that [Q] = 1000 and V_{AX} = 8 V. The COMP output is still high because of $V_{AX} < V_A$. This high signal tells the control logic that the setting of the MSB did not make V_{AX} exceed V_A, so that MSB is kept at 1. The control logic sets Q_2 to 1 to produce [Q] = 1100, V_{AX} = 12 V at time t_2. Since V_{AX}

> V_A, the COMP output goes low. The low signal lets the control logic clear Q_2 back to 0 at t_3 due to V_{AX} being too large. Therefore, the register content is still 1000. At t_4, the control unit sets Q_1 to 1 so that [Q] = 1010 and V_{AX} = 10 V. With $V_{AX} < V_A$, COMP output goes high and the control logic keeps Q_1 set at 1. At the final stage, the control logic sets LSB to 1 at t_5 so that [Q] = 1011 and V_{AX} = 11 V. Due to $V_{AX} > V_A$, COMP output moves to low to signal that V_{AX} is too large and the control logic clears Q_0 back to 0 at t_6. All the register bits have been treated so that the conversion is completed. The control logic sends out an EOC signal to provide notice that the digital equivalent of V_A is in the register now. For this example, we notice that V_{AX} is less than V_A. This is the characteristic of the successive-approximation method.

In this project, a 8-bit SAC chip ADC0804 [16] is used as an A/D converter. It is a 20-pin CMOS IC with operation voltage 5 V. The pin layout is shown in Figure 4.6. The chip can handle a full range 0 to 5 V analog input between pins 6 and 7 ($V_{in} = V_6 - V_7$). There is an on-chip clock generator which requires only an external resistor and capacitor to produce a frequency of $f = 1/(1.1RC)$. A 10 kΩ resistor and 150 pF capacitor are used in this project to generate 606 kHz internal clock frequency and the corresponding conversion time is approximately 100 μs. The connection for ADC0804 is shown

in Figure 4.7. The sampling frequency was controlled by FPGA at 8 kHz.

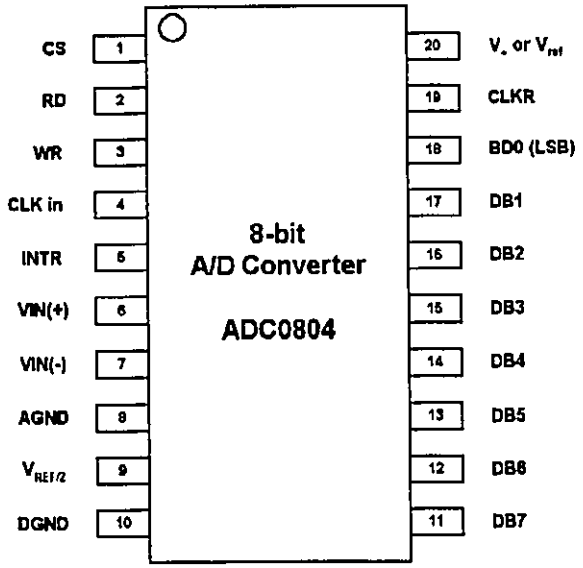


Figure 4.6 Pin Layout of ADC0804

4.2.3 Digital-to-Analog Converter (DAC)

A monolithic 8-bit high-speed current-output DAC0800 [17] is used in this project. It is a 16-pin CMOS IC with operation voltage 5 V. The DAC0800 features typical settling times of 100 ns and high compliance complementary current outputs to allow differential output voltages of 20 V_{p-p} with simple resistor loads. Full scale error of the DAC is ±1 LSB. The performance and characteristics of the device are essentially unchanged over the full ±4.5 V to ±18 V power

supply range. The pin layout of the DAC0800 is shown in Figure 4.8.

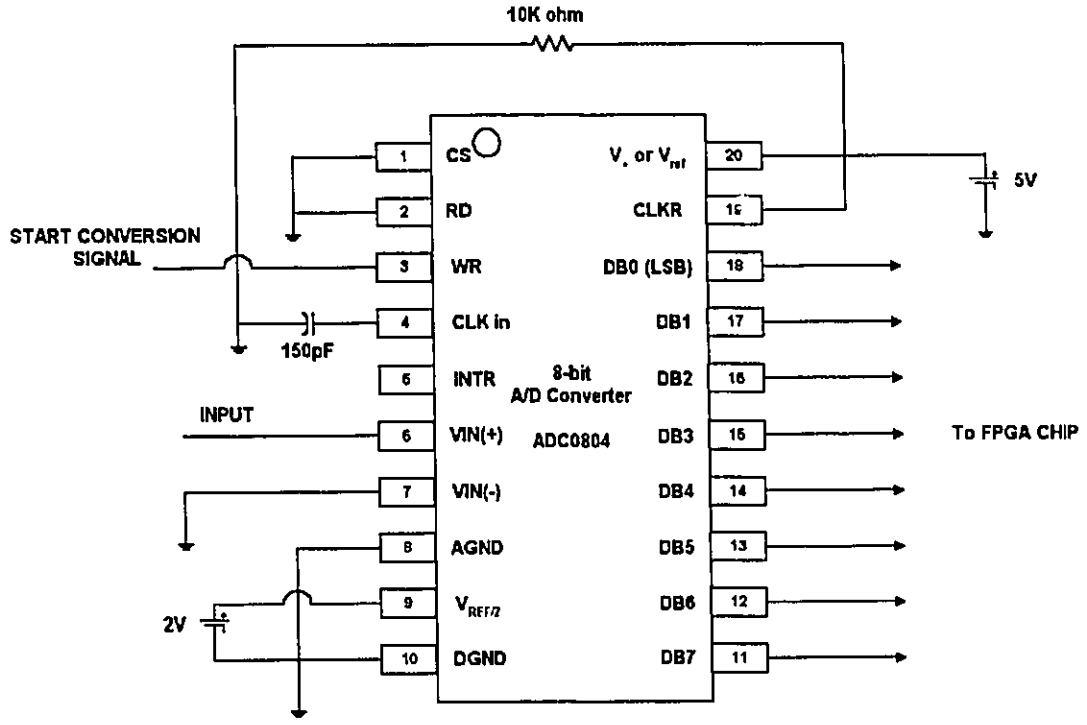


Figure 4.7 Typical Connection of ADC0804

In this thesis, the DAC is connected as a symmetrical offset binary operation. The output voltage is 10 V_{p-p} and symmetrical about ground under the condition $R_L = R'_L$ within ± 0.05%. The expression for the output is

$$E_o = V_{ref} \left(\frac{-255}{256} + \frac{2X}{256} \right) \qquad (4.3)$$

where X is the value of the 8-bit input code. Input and

output examples for the symmetrical offset binary operations are shown in Table 4.1. Figure 4.9 shows a connection for this operation mode.

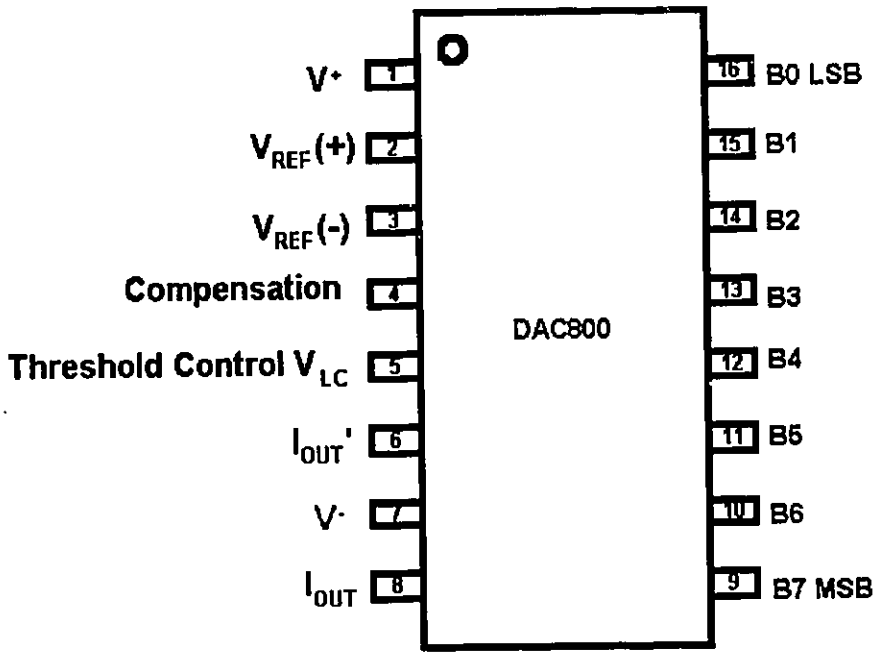


Figure 4.8 Pin Layout of DAC0800

4.3 FPGA Design of Arithmetic Units

Based on the AIC's structure, several arithmetic operations are involved in an adaptive filter such as sign binary addition, multiplication, unsign binary to sign binary conversion and vice versa. These operations are not available in FPGA library so that we have to design several functional cells for these operations based on FPGA library to minimize the number of logic block to be used. In this section, the

structure of functional blocks are described.

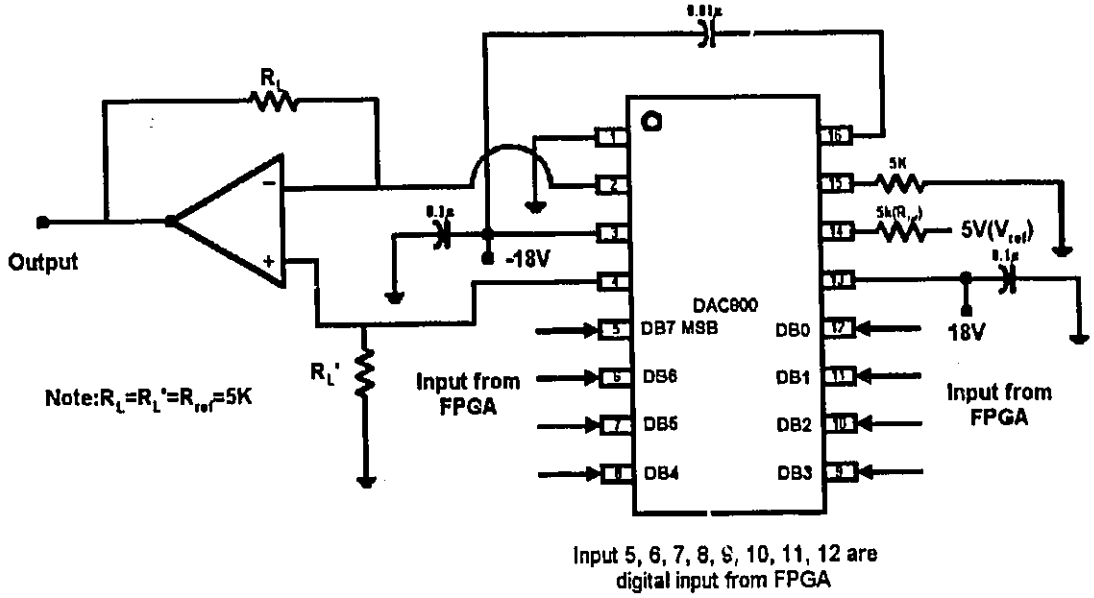


Figure 4.9 Connection of Symmetrical Offset Binary Operation

	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0	E_o
Pos. Full Scale	1	1	1	1	1	1	1	1	+4.98
Pos.Full Scale-LSB	1	1	1	1	1	1	1	0	+4.94
(+) Zero Scale	1	0	0	0	0	0	0	0	+0.02
(+) Zero Scale	0	1	1	1	1	1	1	1	-0.02
Neg.Full Scale+LSB	0	0	0	0	0	0	0	1	-4.94
Neg.Full Scale	0	0	0	0	0	0	0	0	-4.98

Table 4.1 Input & Output Example for Symmetrical Offset Binary Operation

Before we begin the following sections, it is necessary to mention that the sign-magnitude system is used in this thesis to represent a signed binary number. A signed binary number consists of a sign bit, which indicates the positive and negative nature of the stored binary number, and magnitude

bits. The number in Figure 4.10 includes a sign bit and eight magnitude bits.

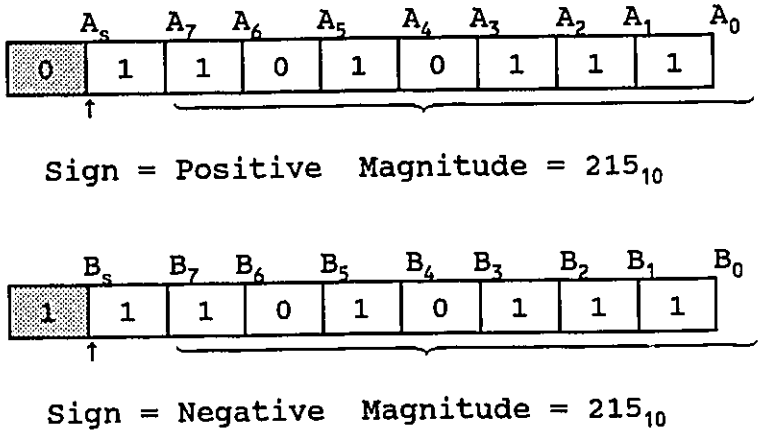


Figure 4.10 Representation of Signed Binary Number in Sign-magnitude Format

4.3.1 Signed Binary Adder

The addition of signed binary numbers actually involves binary subtraction and addition. We will now investigate how the operations are performed in a FPGA chip. In considering various cases, it is important to note that the sign bits of inputs determine the arithmetic operation of the functional cell.

Case I: Two Positive Numbers. The addition of two positive numbers is straightforward. Consider the addition of +4 and +7:

```
          sign
            ↓
     +7 ⇒ 0   0111 (augend)
+    +4 ⇒ 0   0100 (addend)
-------------------------
    +11 ⇒ 0   1011 (sum)
```

Note that the sign bit of sum always has the same sign as the augend and addend.

Case II: Two Negative Numbers. The addition of two negative numbers is straightforward. Consider the addition of -4 and -7:

```
          sign
            ↓
     -7 ⇒ 1   0111 (augend)
+    -4 ⇒ 1   0100 (addend)
-------------------------
    -11 ⇒ 1   1011 (sum)
```

As shown in case 1, the sign bit of sum always has the same sign as the augend and addend.

Case III: Positive Number and Smaller Negative Number. Consider the addition of +7 and -4.

```
          sign
            ↓
     +7 ⇒ 0   0111 (minuend)
+    -4 ⇒ 1   0100 (subtrahend)
-------------------------
     +3 ⇒ 0   0011
```

Magnitude subtraction is performed to get magnitude of the

answer and the sign bit of the result is the same as that of the biggest absolute magnitude number. From the above example, the answer is positive because the biggest absolute magnitude number is 7.

Case IV: Positive Number and Larger Negative Number. Consider the addition of -7 and +4.

```
          sign
            ↓
    -7 ⇒ 0  0111 (minuend)
+   +4 ⇒ 1  0100 (subtrahend)
-------------------------------
    -3 ⇒ 1  0011
```

With reference to Case III, the magnitude subtraction is performed to get the result's magnitude and the sign bit of the answer is the same as that of the biggest absolute magnitude number. Needless to say, the answer is negative from the above example.

We have to design a signed binary adder to accommodate all of the above cases. If the input binary numbers have the same sign, an addition is performed and the sign of sum is the same as the inputs. If input binary numbers are of different signs, a magnitude subtraction is performed and the sign of the answer is equal to that of the biggest absolute magnitude number. According to the Xilinx design library, adders and subtracters for unsigned binary numbers are available. Hence

we only need to design a comparison logic unit for the sign bit to determine the arithmetic operation. During the magnitude subtraction, if the absolute value of the minuend is less than that of subtrahend, the answer will be a negative number in a 2's complement format. An operation of inverse 2's complement of the answer is required under this case. The flowchart, the schematic of the signed binary adder and the ViewSim simulation results are shown in Figure 4.11 and Figure 4.12 respectively.

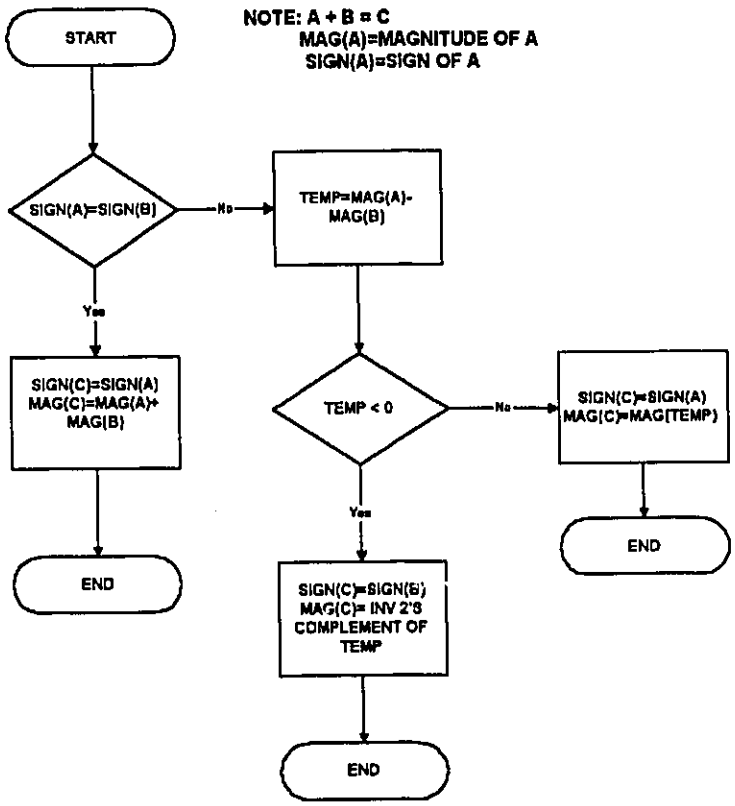
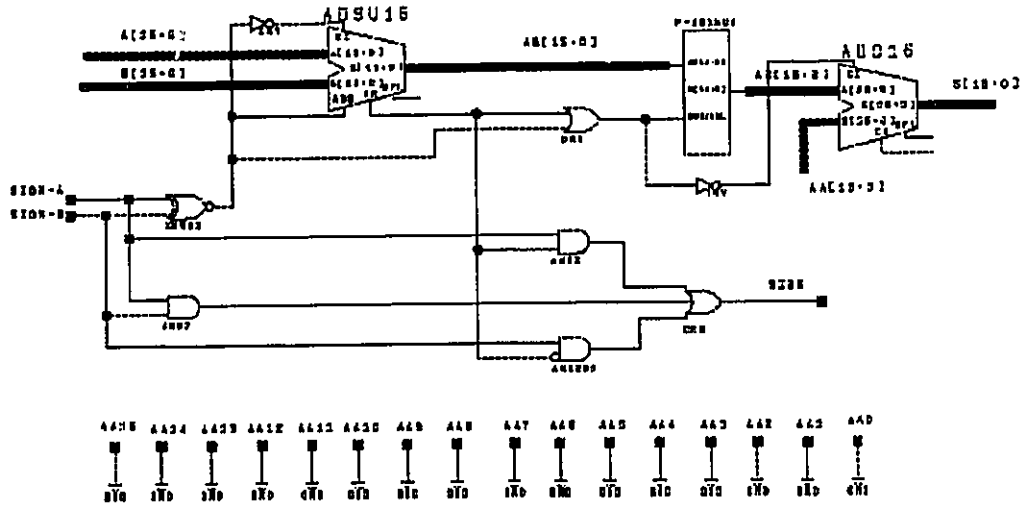
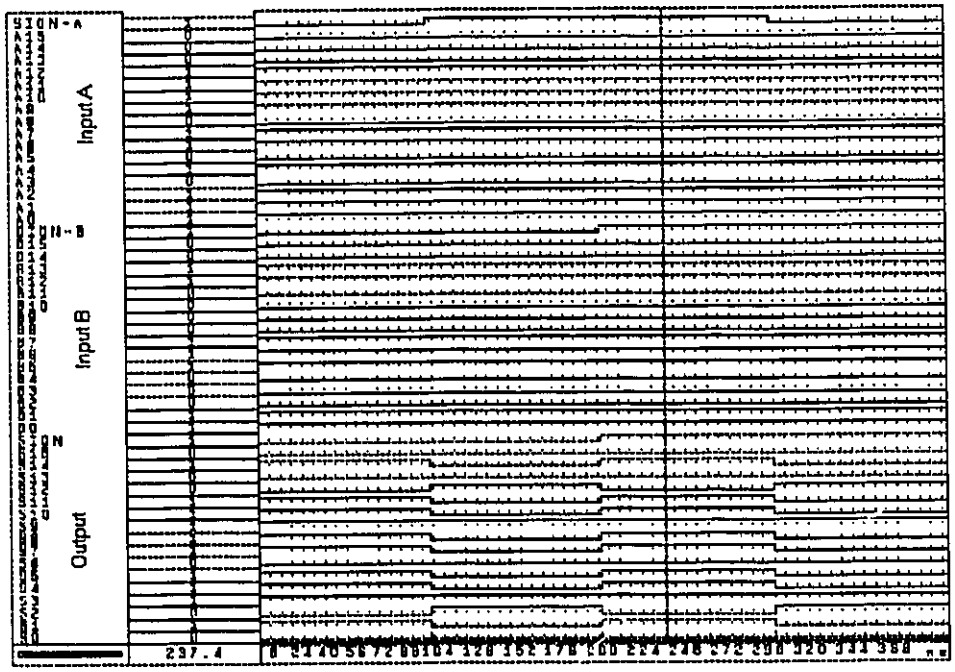


Figure 4.11 Flowchart of Signed Binary Adder



4.12a



4.12b

Figure 4.12 (a) Schematic of Signed Binary Adder; (b) Simulation Results of Signed Binary Adder

4.3.2 Binary Multiplier

The multiplication of two binary numbers is done in the same manner as the multiplication of two decimal numbers. The process is actually simpler, since the multiplier digits are either 0 or 1. Therefore, we are always conduct the multiplication by using 0 or 1 and no other digits. The following example illustrates for unsigned binary number.

```
          1001   ⇐ (multiplicand)
          1011   ⇐ (multiplier)
      ------------
          1001
         1001      (partial product)
        0000
  +    1001
      ------------
       1100011   ⇐ (product)
```

First, the LSB of the multiplier is examined; in our example it is a 1. This 1 multiplier the multiplicand to produce 1001, which is written down as the first partial product. Next, the second bit of the multiplier is examined. It is a 1, and so 1001 is written for the second partial product. Note that this second partial product is shifted one place to the left relative to the first one. The third bit of the multiplier is 0, and 0000 is written as the third partial product; again, it is shifted one place left relative to second partial product. The forth multiplier bit is 1, and so the last partial product is 1001, which again is shifted one

position to the left. The four partial products are then summed to produce the final product.

Multiplication of two signed binary numbers is almost the same as two unsigned binary numbers. Basically, the magnitude of product is unchanged with either signed or unsigned inputs. The sign of the product is depended on the sign of the inputs only. The truth-table of signed multiplication is shown below

SIGN(A)	SIGN(B)	SIGN(A*B)
0	0	0
0	1	1
1	0	1
1	1	0

Table 4.2 Truth-table of Signed Multiplication

According to Table 4.2, the correct output will be obtained by an exclusive-OR operation. The 8-bit multiplier requires 11 clocks cycles to finish the computation. The flowchart of the multiplier is shown in Figure 4.13. The schematic of the multiplier and typical signals during a multiplication are shown in Figure 4.14.

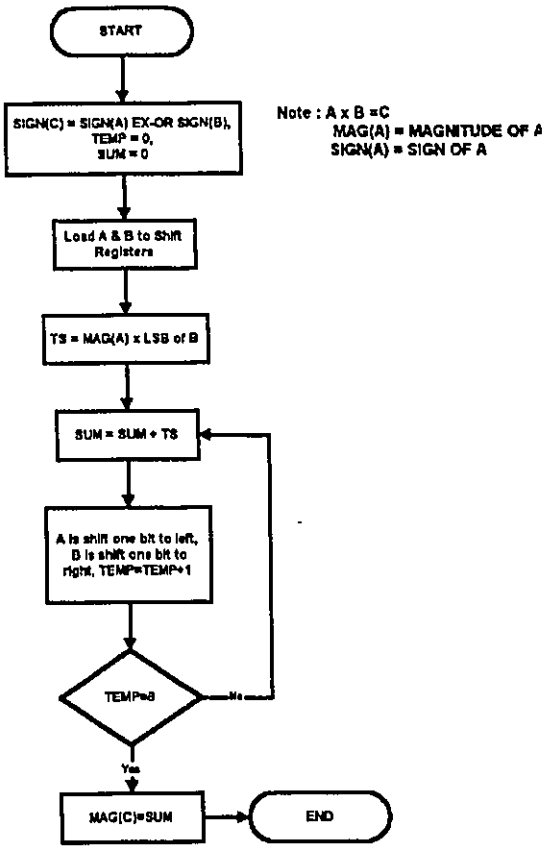
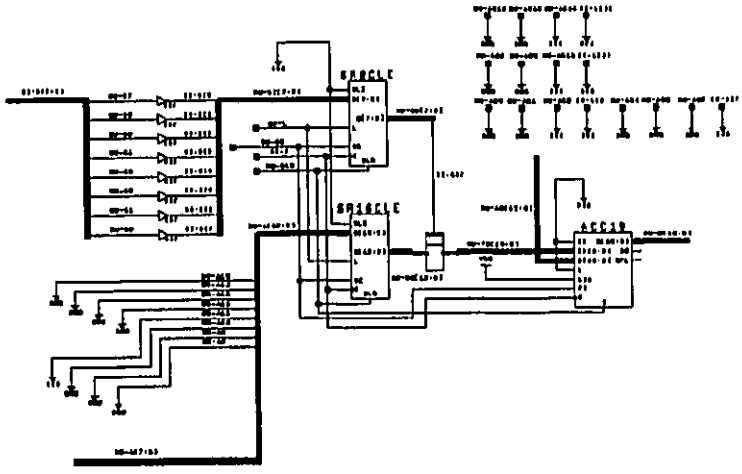
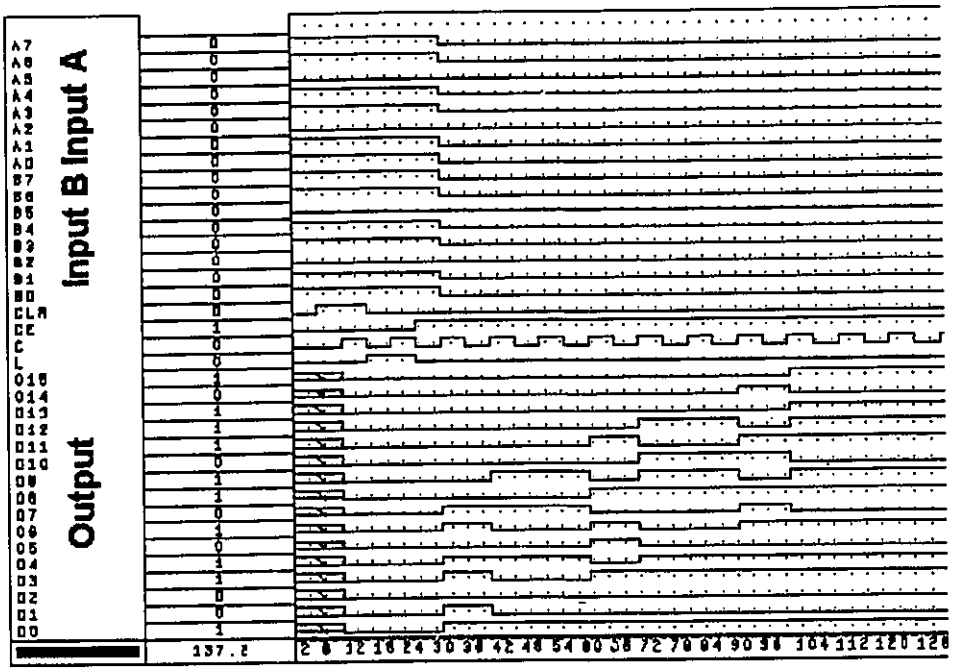


Figure 4.13 Flowchart of Binary Multiplier



4.14a



4.14b

Figure 4.14 (a) Schematic of Multiplier;
(b) Typical Signals During Multiplication

4.3.3 Unsigned Binary to Signed Binary Converter

The output of the ADC is 8-bit unsigned numbers with full-scale analog input range 0-4 V. Due to the input signals being shifted up by 2 V before the ADC, the first 127 levels (0-2 V) are negative components of the inputs. The 128-255 levels (2-4 V) represent positive components of the inputs. It is necessary to convert the unsigned binary inputs to the signed binary numbers before the adaption in the FPGA. From 128-255 level, the difference between the magnitude of an unsigned binary number and a signed binary number is the MSB.

In this case, we only take the inverse of the MSB and leave the rest of the bits unchanged in conversion. From 0-127 levels, 2's complement is performed on last seven bits to get the magnitude of the negative signed number. In both cases, the sign bit of a signed binary number is equal to the inverse of the MSB of an unsigned binary number. Figure 4.15 and Figure 4.16 show the flowchart, the schematic and the simulation results of the unsigned-to-signed binary converter, respectively.

Decimal Value	Level	Unsigned Binary Level	Signed Binary level
+2.0000	255	11111111	0-01111111
+1.9843	254	11111110	0-01111110
:	:	:	:
:	:	:	:
+0.0078	128	10000000	0-00000000
-0.0078	127	01111111	1-00000001
:	:	:	:
:	:	:	:
-1.9834	1	00000001	1-01111111
-2.0000	0	00000000	1-10000000

Table 4.3 Relationship between Different Numerical Formats

4.3.4 Signed Binary to Unsigned Binary Converter

The DAC only takes 8-bit unsigned binary numbers so that the output of the FPGA has to be converted back to unsigned binary numbers. This is a reverse process of unsigned-to-signed binary conversion. The MSB of an unsigned binary

number is equal to the inverse of the sign bit of a signed binary number. For positive sign numbers, the rest of the bits are unchanged. For negative binary numbers, the last seven bits come from the inverse 2's complement of the magnitude of signed binary numbers. The flowchart of the signed-to-unsigned binary converter is shown in Figure 4.17, and its schematic and its simulation results are displayed in Figure 4.18.

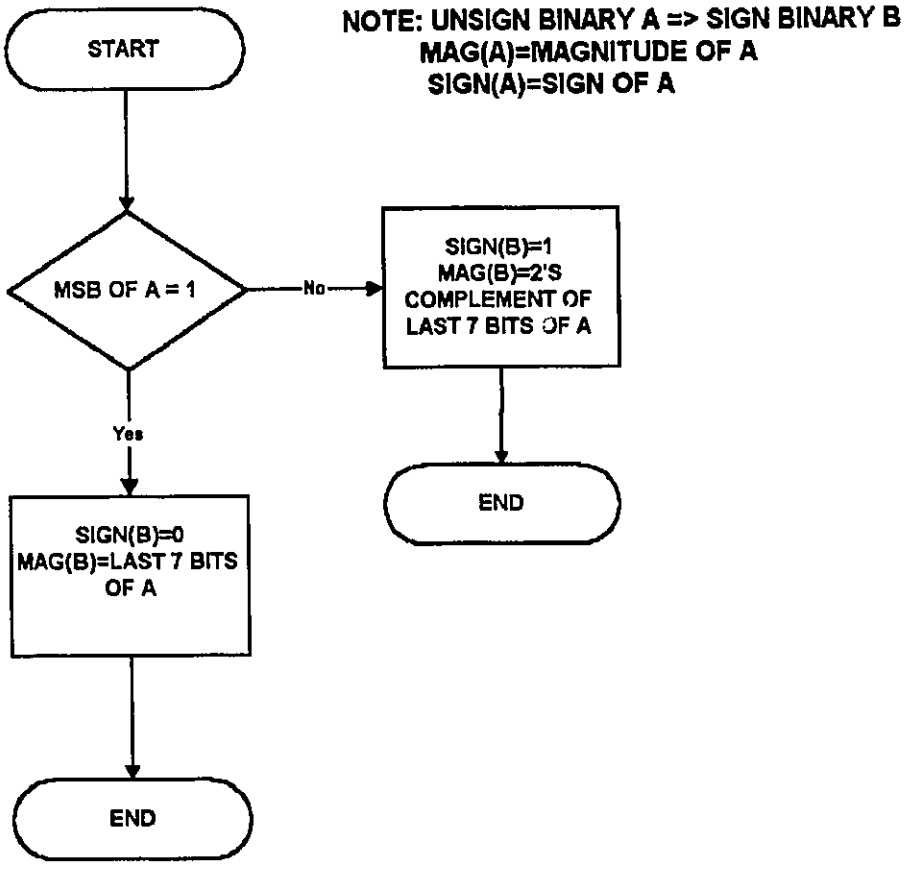
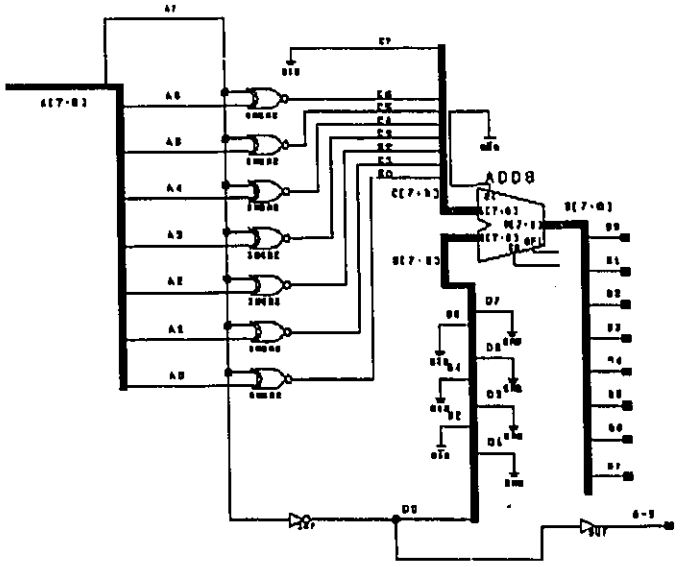
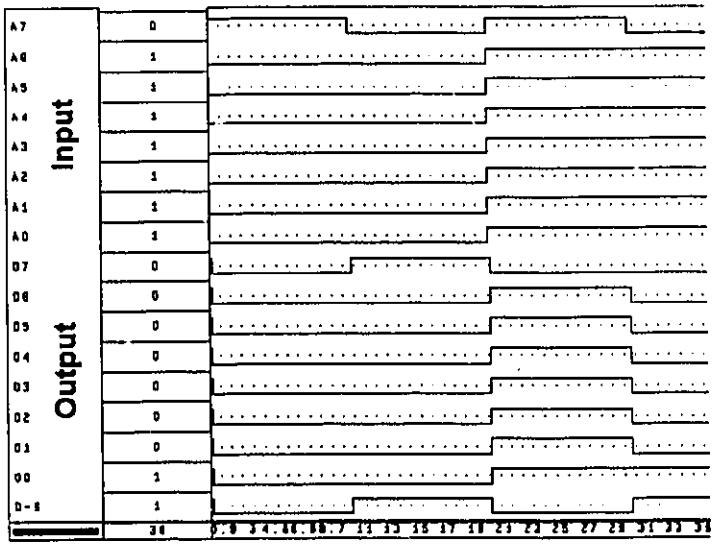


Figure 4.15 Flowchart of Unsigned Binary to Signed Binary Converter



4.16a



4.16b

Figure 4.16 (a) Schematic;
(b) Simulation Results of Unsigned Binary to Signed Binary Converter

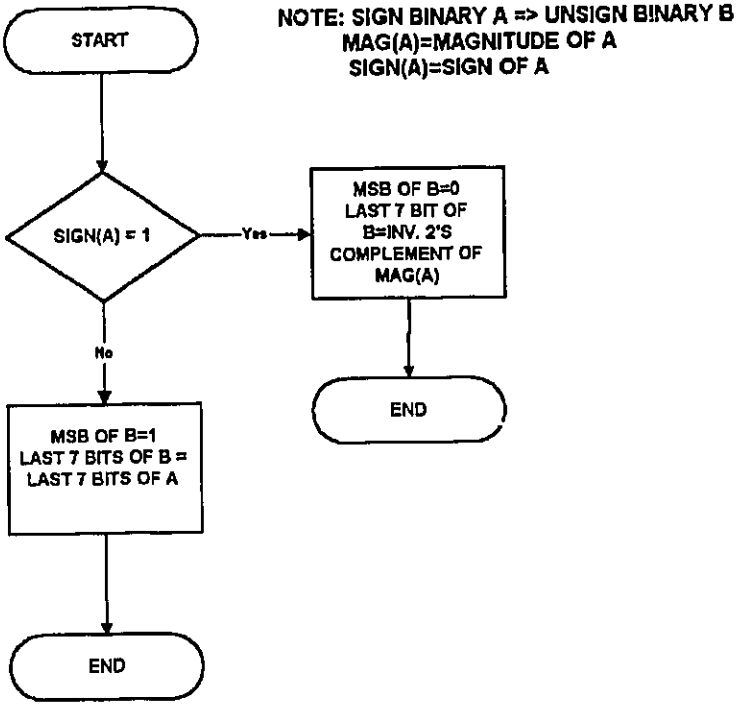
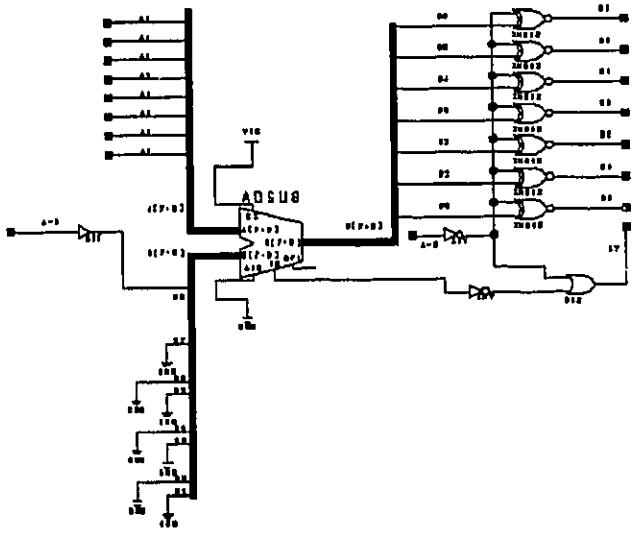
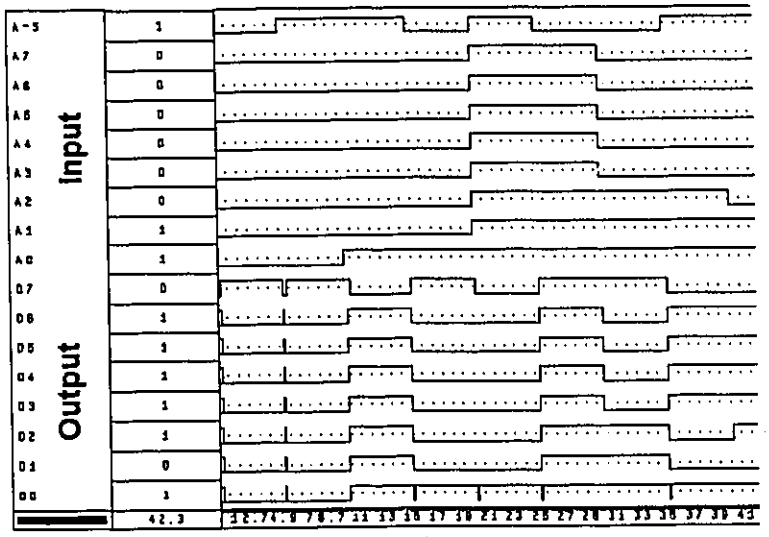


Figure 4.17 Flowchart of Signed Binary to Unsigned Binary Converter



4.18a



4.18b

Figure 4.18 (a) Schematic;
(b) Simulation Results of Signed Binary to Unsigned Binary Converter

4.3.5 Divisor

In this project, a binary division is just shifting a binary number to the right. If the number is shifted one bit to the right, it is equivalent to dividing by 2^1 in decimal. The following formula shows the relationship between the binary right shift and the decimal division

$$Ans = X / 2^Y \tag{4.4}$$

where Y is the number of right shifted bit.

For example:

$$\begin{array}{rcl} 24 & - & 11000 \\ \div\ 2^2 & - & \text{shift 2 bit right} \\ \hline 6 & - & 110 \end{array}$$

This division suffers from two main problems. First, it can only divide by 2^Y. Other divisions cannot be calculated by this method. Second, a truncation occurs when X is not a multiple of 2^Y. The accuracy of the computation is greatly reduced under this condition.

For example:

$$\begin{array}{rcl} 24 & - & 11000 \\ \div\ 2^4 & - & \text{shift 4 bit right} \\ \hline 1.5 & - & 00001 \end{array}$$

However, the main advantage is its simple structure.

4.3.6 16-bit binary number to 8-bit binary number

Multiplication of two 8-bit numbers will yield an answer in 16-bit format. It is necessary to convert the answer back into 8-bit format in order to perform additions or subtractions with other 8-bit numbers.

Let

$$A \times B = C$$

where A, B and C are binary number.

If the multiplication is computed in decimal arithmetic, the equation is as follows:

$$\underset{\text{8 bit format}}{(A * 2/256)} * \underset{\text{8 bit format}}{(B * 2/256)} = A*B*(2/256)^2 = \underset{\text{16 bit format}}{C*(2/256)^2}$$

$$\underset{\text{16 bit format}}{C*(2/256)^2} \longrightarrow \underset{\text{8 bit format}}{C/2^7 * 2/256}$$

For example:

Decimal			binary	
0.5	≈ 0.5/2*256	= 64	=	1000000
* 0.3	≈ 0.3/2*256	= 38.4	=	100110
0.15				100110000000 (16 bit format)
			Shift 7 bit right →	10011 (8 bit format)

Verify:

$$10011 = 19/259*2 = 0.148 \approx 0.15$$

In the division, this computation suffers from truncation error.

4.4 Structure of the AIC

Since the performance of the AIC depends on the order of the filter, it is better to have the order of the filter as high as possible. However, the logic capacity of FPGA (Xilinx 4013MQ208) is limited. Direct implementation of the design (Figure 4.2) is not a suitable approach in this condition.

The Queue structure of the AIC is introduced to increase the order of the AIC while providing satisfactory operation speed. The comparison between the direct implementation and the queue structure implementation is shown in Table 4.4.

4.4.1 Queue Structure of the AIC

The concept of the queue structure comes from banking. Clients queues up for available counters inside a bank. If there are 10 counters, the bank can deal with ten clients at a time. For a bank with two counters, it can accomplish the same amount of work if the tellers work five times as fast as a bank with 10 counters.

Inputs and the arithmetic processor are like the clients and the counter of a bank respectively. Inputs are loaded to the processor one at a time. The temporary output will be stored in the accumulator or data register. After all the inputs are processed, the final output is obtained. The main disadvantage of this structure is that it requires a higher clock rate than that of the direct implementation.

The AIC consists of two clocks, two processor units, a control unit and many data registers. The first clock (8 MHz) is for the processor which performs multiplications and additions. The second one (500 kHz) is for the control unit of the AIC which supervises the input and the output of the AIC

and associated processor. Multiplexers are used to regulate the input data to the processors. Each processor includes dual computation paths. Hence they can handle two input data in the same time interval. The block diagram and the schematic of the AIC are shown in Figure 4.19 and Figure 4.20 respectively.

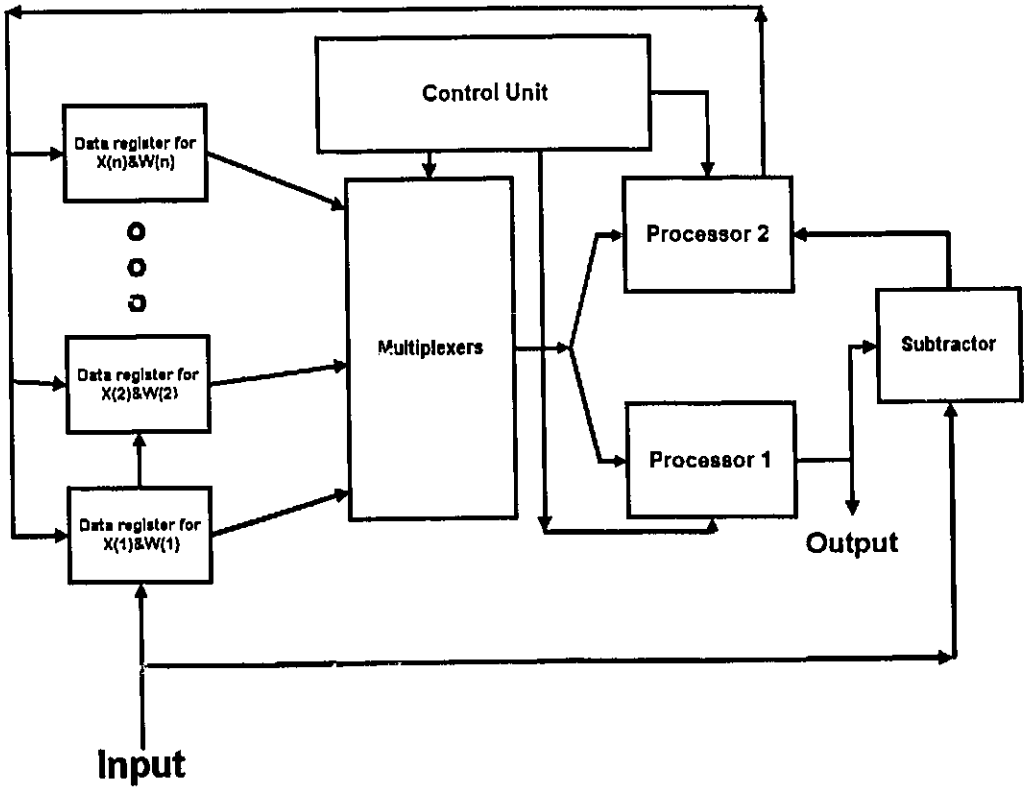


Figure 4.19 Block Diagram of AIC

4.4.2 Processors

There are two types of processor in the design. The

first one, shown in Figure 4.19, is for computation of canceler output. It consists of two multipliers and a sign binary adder. The processor can manage two input data at a time, and the output connects to an accumulator. After 14 clock cycles, the output of the canceler is available from the output of accumulator.

PROCESSOR2 is for computation of new weight. It consists of two multipliers, two sign binary adders and two divisors. With each clock cycle, it can calculate two updated weights and store them in data registers. Both processors require 15 internal clock cycles to finish the computation. The block diagram and the schematic of processors are shown in Figure 4.21, Figure 4.22, Figure 4.23 and Figure 4.24.

	Direct Implementation	Queue Structure
Order	4	28
No. of Processors	4	2
No. of Clock Cycles	35	400 **

** - Based on Single Clock Configuration

Table 4.4 Comparison between Direct Implementation and Queue Structure Implementation

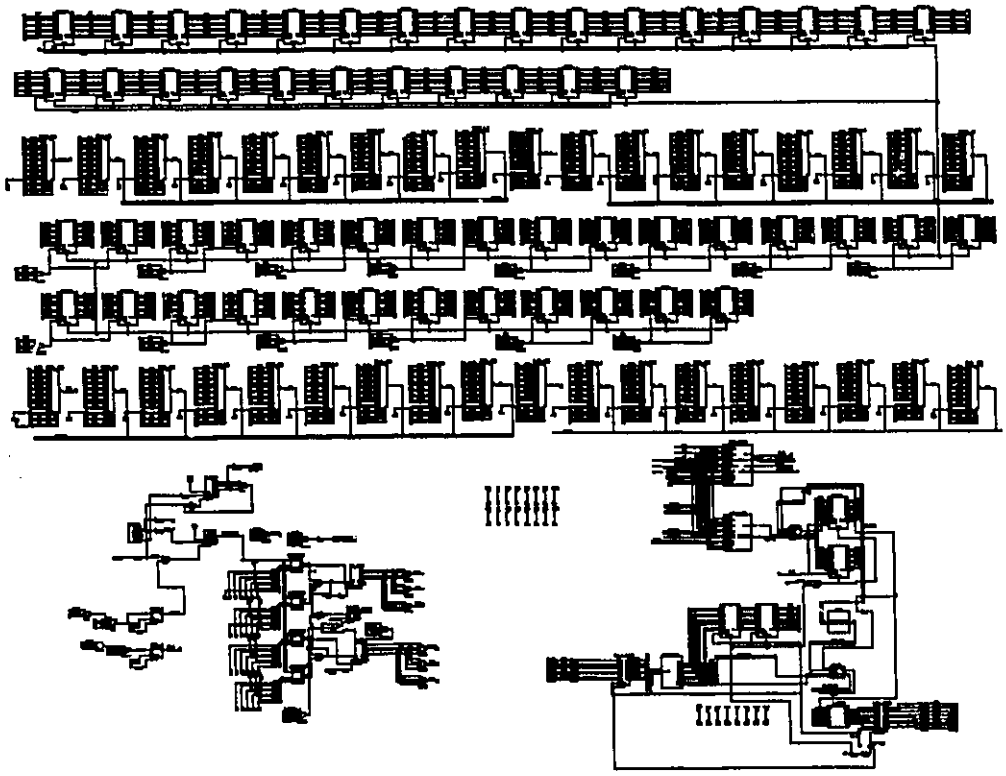


Figure 4.20 Schematic of AIC

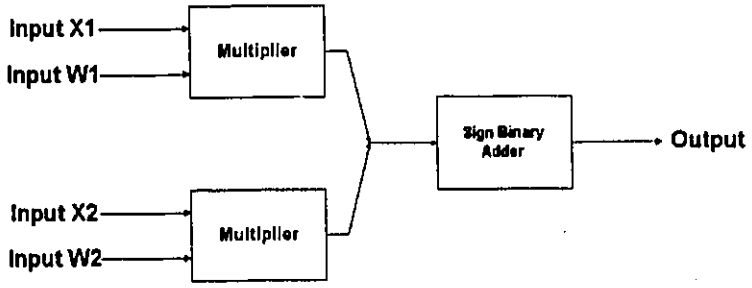


Figure 4.21 Block Diagram of PROCESSOR1

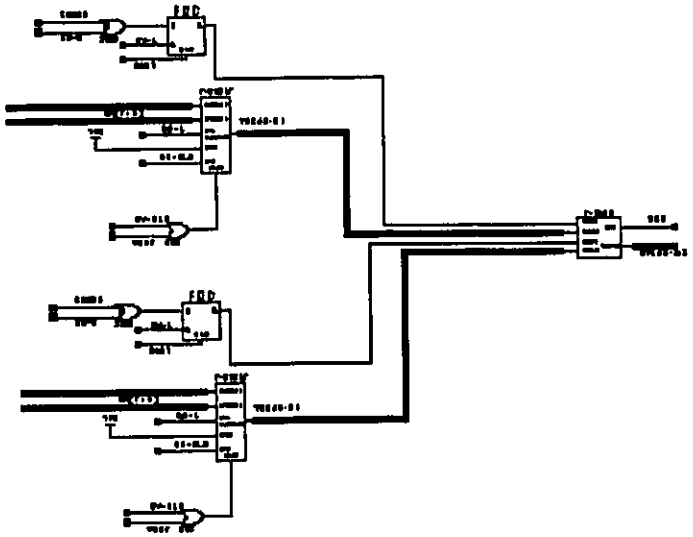


Figure 4.22 Schematic of PROCESSOR1

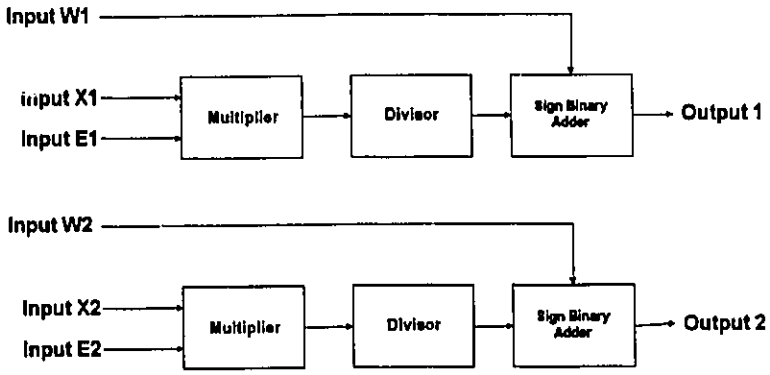


Figure 4.23 Block Diagram of PROCESSOR2

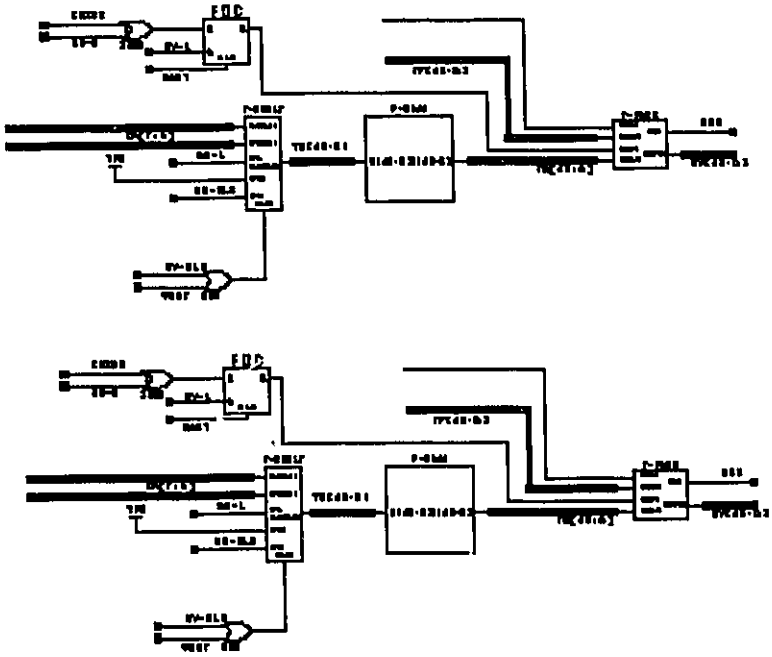


Figure 4.24 Schematic of PROCESSOR2

4.4.3 Control Unit

The control unit controls the data flow in AIC. It consists of three counters. The COUNTER1 guides the input and the output of the AIC. The second and third counters supervise the data input to the processors and binary multiplications respectively. The flowchart of the AIC is shown in Figure 4.25.

At the beginning, the AIC sends a START signal to the ADC to start digital conversion. Then the COUNTER2 is reset and begins counting. According to the value of the COUNTER2, different data is loaded to the processors. The operations in

the processors have to be finished before the next data loaded. Hence, the clock (CLOCK2) for the processors must be faster than the clock (CLOCK1) for the control unit. From previous section, processors require 15 clock cycles to finish the operations. After obtaining output, the COUNTER2 is reset again and the data is loaded to the PROCESSOR2. When the adaptive operation is finished, the output is sent to the output register and the COUNTER1 is reset for the next computation. The arithmetic operation of the AIC has to finish within 125μs (1/8000 Hz) to maintain the data input rate.

In this thesis, the CLOCK1 is 500 kHz and the CLOCK2 is 8 MHz. The CLOCK2 is sixteen times faster than the CLOCK1. Each data requires two clock cycles in the adaptive processes. For 28 data, they require 56 clock cycles in the arithmetic operations plus 5 clock cycles for the I/O operations. Due to the dual computation path of the processors, the required clock cycles for the processes reduce to 33 (66 μs). The schematic of the control unit and the signal during the adaptive operation are shown in Figure 4.26 and Figure 4.27 respectively.

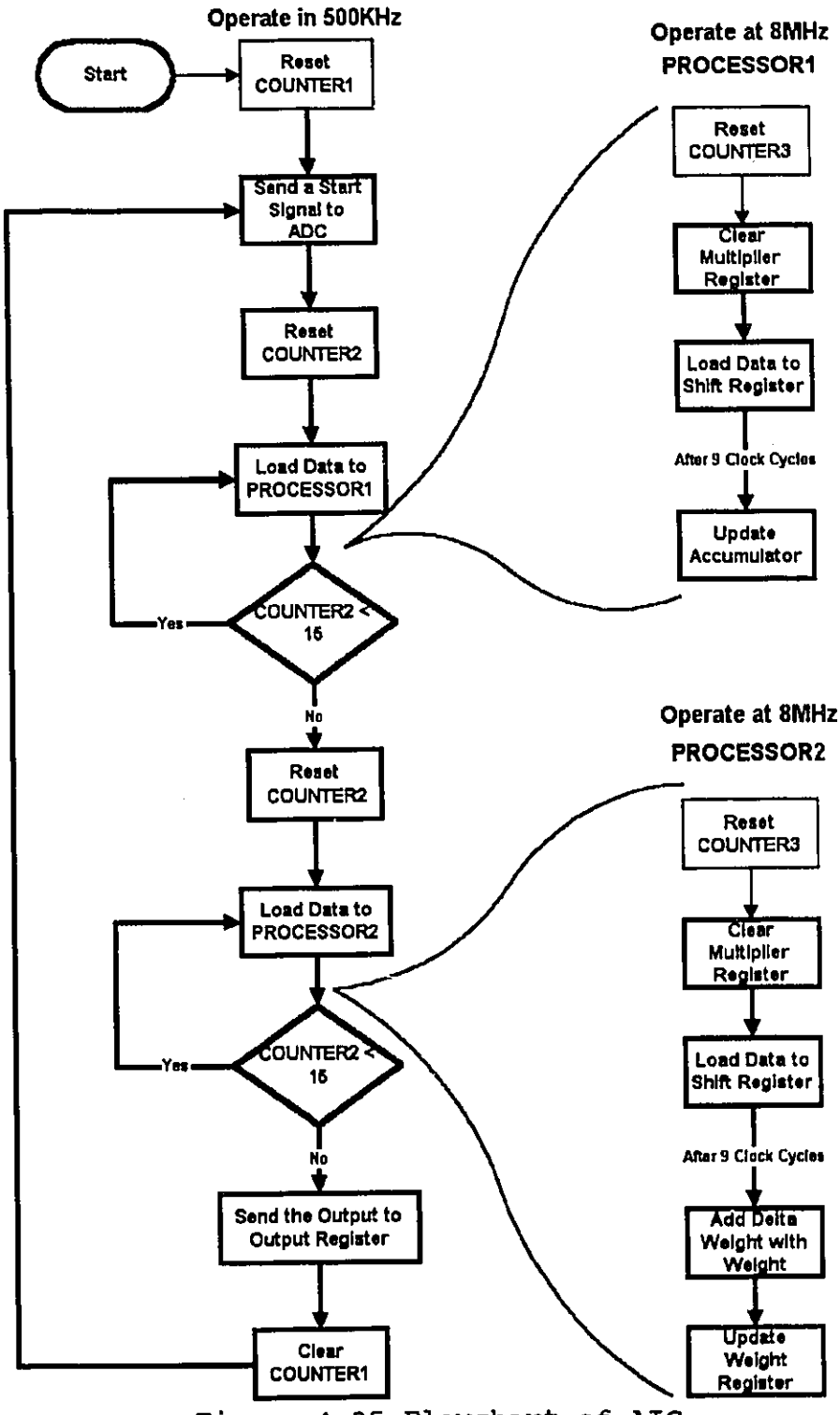


Figure 4.25 Flowchart of AIC

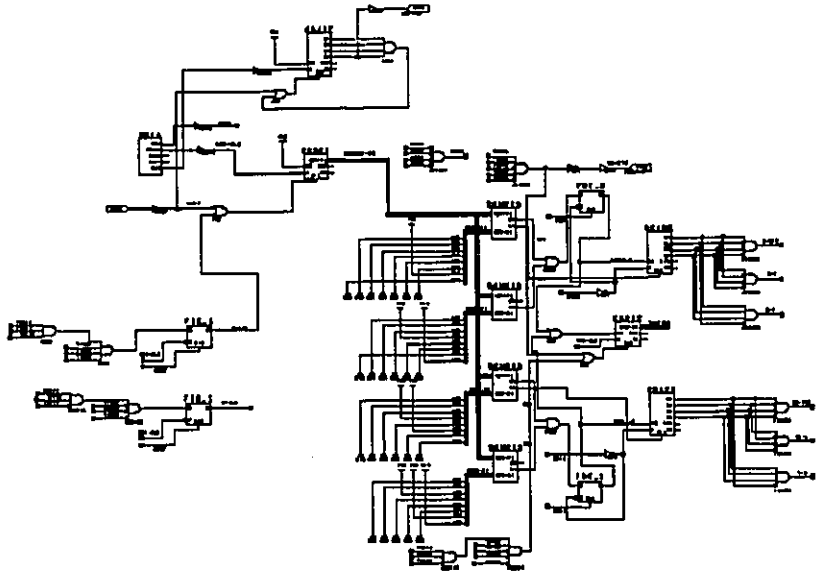


Figure 4.26 Schematic of Control Unit

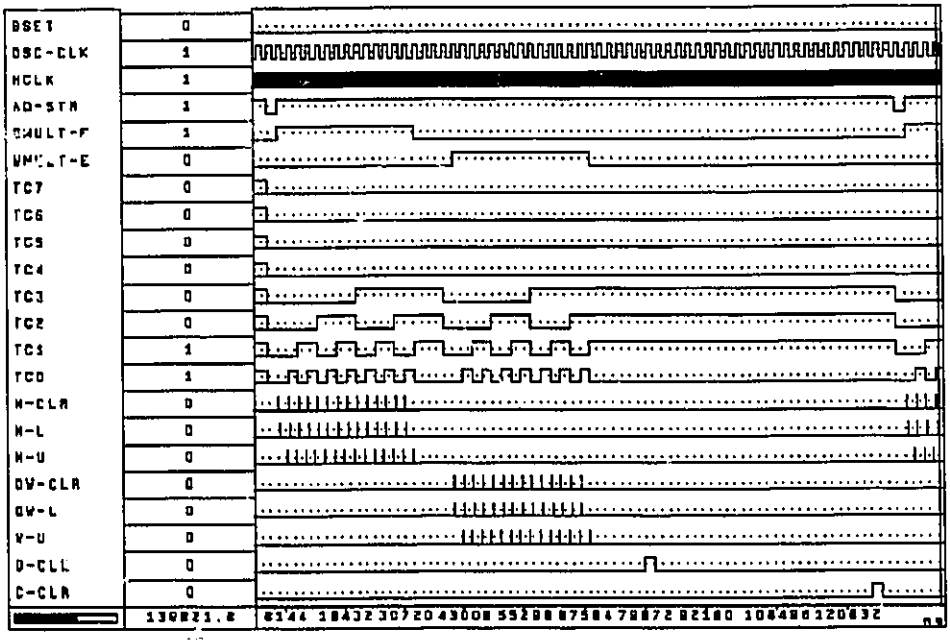


Figure 2.27 Control Signal During the Adaptive Process

4.5 Summary

This chapter presents the design of the adaptive interference canceler which is used to extract a periodic signal from radio interference. By increasing the order of the filter and the precision in the ACD, DAC and computation, the efficiency of the AIC is increased. The trade-off is that it requires more logic gates to implement the filter. In this thesis, 8-bit resolution ADC and DAC are used in the input and the output. 16-bit precision is used in computation inside the FPGA. The design of functional cells are based on the Xilinx library so that they can be as simple as possible. The AIC design offers a solution for higher filter order implementation than that of a direct approach.

Chapter 5

Hardware Implementation & Test Results

The adaptive interference canceler (Figure 4.19), described in the previous chapter, is laid out in Viewlogic Workview and implemented in an FPGA chip. The test results are recorded and verified with the simulation results. It is shown experimentally that the canceler suffers from insufficient filter order as well as insufficient word length for the input and output signal.

5.1 Simulations

The simulations of the canceler are divided into two levels: the Fortran language and ViewSim simulations. The

Fortran simulation consists of two parts. The first one is to examine the mathematical integrity of the proposed canceler and the second one is to verify the correctness of the AIC architecture. ViewSim simulations check for layout mistakes and provide timing information about the circuit.

5.1.1 Design Verification

In the Fortran simulation, the signal flow diagram of the canceler is converted into a set of arithmetic equations to simulate the AIC functional blocks. The accuracy of the Fortran simulation is found by comparing the simulated output with the expected output. The expected output is to be free from the interference from the input signal.

Simulations are also performed in Fortran and ViewSim to verify the architectural accuracy of the AIC. Several steps are taken to ensure the hardware designs are error free. Firstly, each arithmetic unit, described in the previous chapter, is simulated in Fortran in the form of a binary arithmetic calculation. Each test vector in the Fortran simulation is compared with the expected answer. Then, the canceler is constructed by connecting different arithmetic units after the simulated results of the computing units are validated. Arithmetic confirmation of the AIC is complete when the test vector outputs from both simulations match.

The validated architectural design of the AIC is then laid out in Viewlogic Workview. Furthermore, the timing and functional simulation of the layouts are performed in the ViewSim simulator. The ViewSim output test vectors are matched with those generated from functional simulation in Fortran to ensure no error is introduced during the schematic editing.

Simul-ation	Order of AIC	No. Input Delay	% of Signal w.r.t max. P-to-P Input	Learning Parameter	% of output interference w.r.t input interference
1	16	2	80	0.005	50
2	32	2	80	0.005	35
3	64	2	80	0.005	30
4	128	2	80	0.005	30
5	32	0	80	0.005	40
6	32	1	80	0.005	30
7	32	3	80	0.005	27
8	32	2	60	0.005	40
9	32	2	40	0.005	35
10	32	2	20	0.005	35
11	32	2	80	0.05	50
12	32	2	80	0.0005	25

Table 5.1 Results of Different Configuration Simulations

5.1.2 Simulation Results

Simulations were run to obtain the outputs of the AIC according to the AIC's structure described above. The simulation was divided into four categories: order of filter, input delay, percentage of interference with respect to input signal and learning parameter. Each category varied one parameter in simulation. Same test signal, consists an interference and three sinusoidal signals with frequencies 300 Hz, 700 Hz and 800 Hz, is used in simulations.

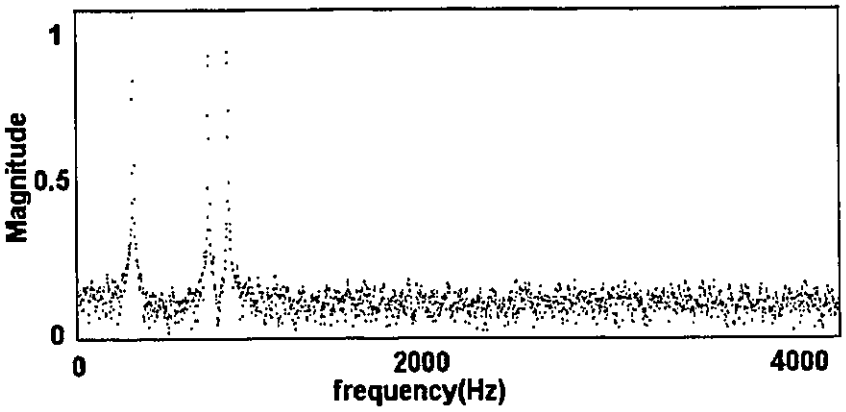


Figure 5.1 Frequency Spectrum of Test Input

From the simulation results, roughly 70% of the interference can be removed when the order of the AIC is increased to 64. There is not much improvement in performance when the order of canceler goes up to 128. This is because a 64^{th} order canceler is enough to converge to a minimum error

point for the test input used. If the test vector consists of more sinusoidal signals, a 128^{th} order canceler should out perform a 64^{th} order filter.

Also, the performance of the AIC improves as the number of input delay elements is increased. The delays cause the interference components in the reference input to become decorrelated from those in the primary input. It will reach its maximum performance when the number of delays reaches the optimal.

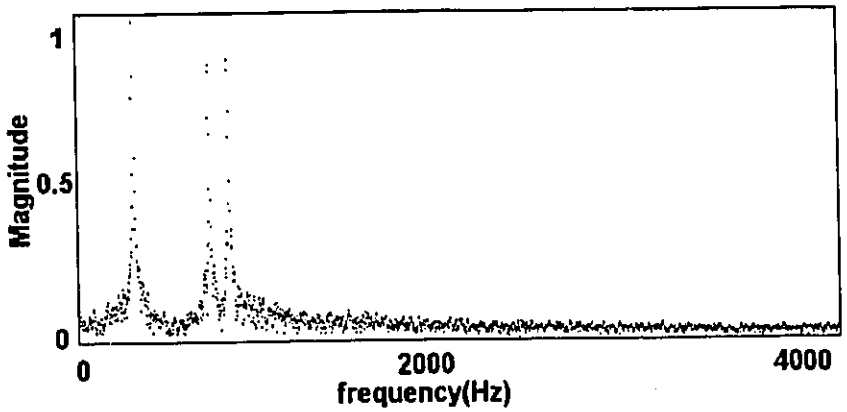


Figure 5.2 Output of Canceler in Simulation 4

Regarding the learning parameter (LP), a large value of the LP gives a fast convergence rate in the AIC but it has a potential problem of being over-sensitive. Sometimes the AIC cannot converge to a minimum error point because of being over-sensitive. As in simulation 11 and 12, simulation 11 suffers from being over-sensitive. Although simulation 12

takes more time to converge, it can remove more interference components than simulation 11.

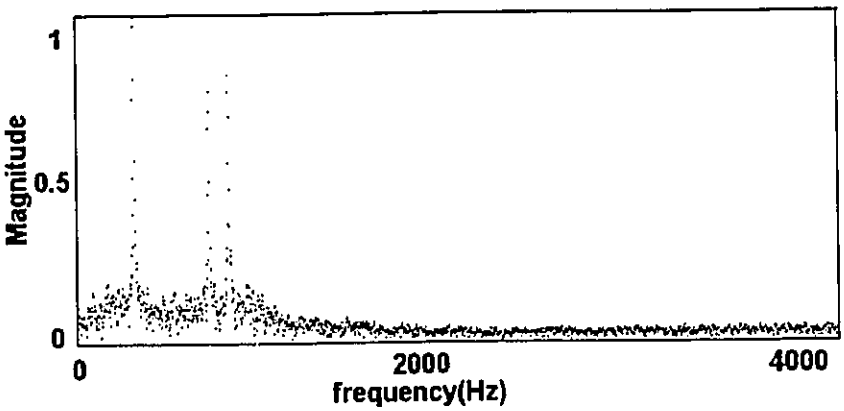


Figure 5.3 Output of Canceler in Simulation 7

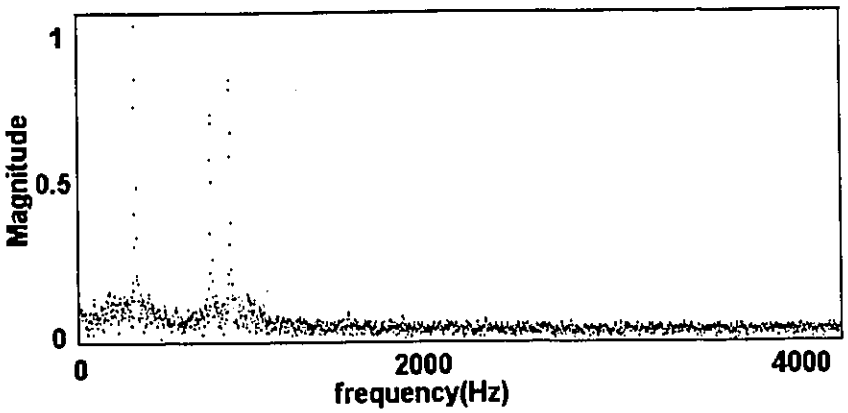


Figure 5.4 Output of Canceler in Simulation 12

Finally, the percentage of interference in the input signal does not affect the performance of the canceler much. For a 32^{th} order AIC, about 67% of interference can be removed from input signal in various conditions. The plots of the

frequency spectrum of various input signals are included in Appendix A.

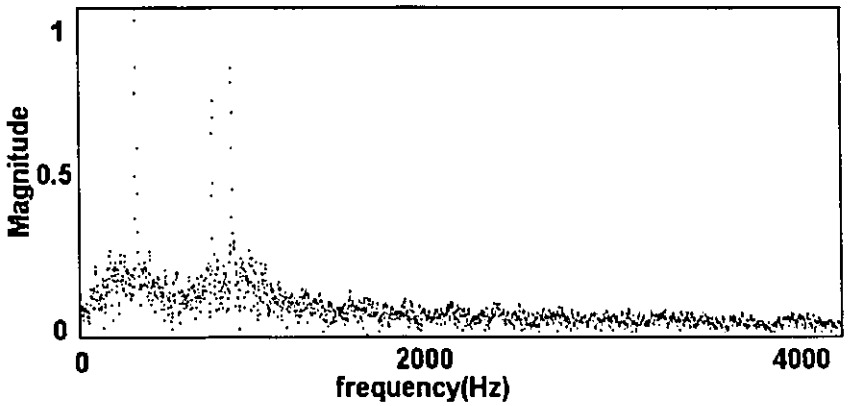


Figure 5.5 Output of Canceler in Simulation 8

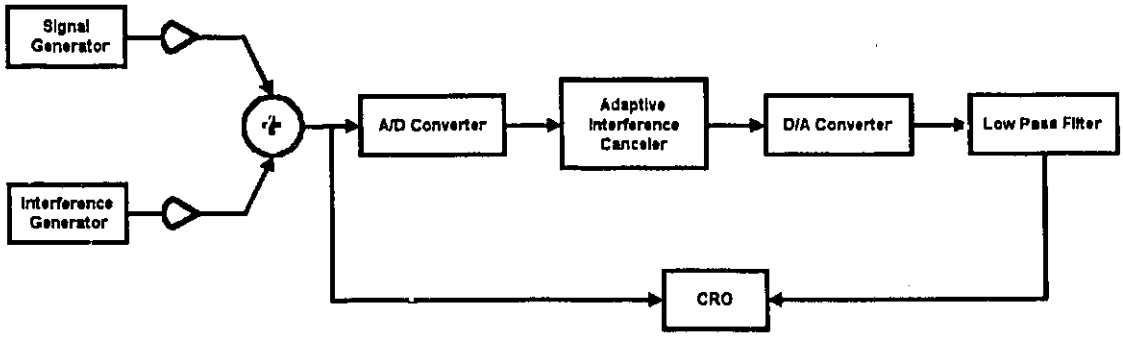


Figure 5.6 Connections of Equipment

5.2 Hardware Testing

The hardware test of the adaptive interference canceler consists of the real time cancellation of an analog input

signal. The input of the canceler connects to a signal generator and the output of the canceler connects to a CRO. The basic connections of the equipment used for testing the AIC are shown in Figure 5.6. The amplitude were obtained by introducing different sine waves with radio interference as analog inputs to the A/D converter.

The canceler occupied 80% (packed CLBs reports in PPR output file) of the CLBs and used 64% of the available flip-flops. Due to the limited routing resources of FPGA chip, the order of canceler cannot exceed 28. The input data rate of the AIC is set to 8 kHz, it is identical to the sampling rate of the A/D converter in order to maintain a consistent rate of data output. Figure 5.7 shows the AIC demonstration board.

5.2.1 Test Results

During prototype testing, a 600 Hz 3.0 V_{p-p} sine wave with 20% interference w.r.t. the sine wave were inputed to the A/D converter. The test result is shown in Figure 5.8. The output signal (the lower one in Figure 5.8) was a sine wave which was approximately 90° out of phase with the input signal. By observation of the output waveform, the canceler cleared some of the input interference from the input signal.

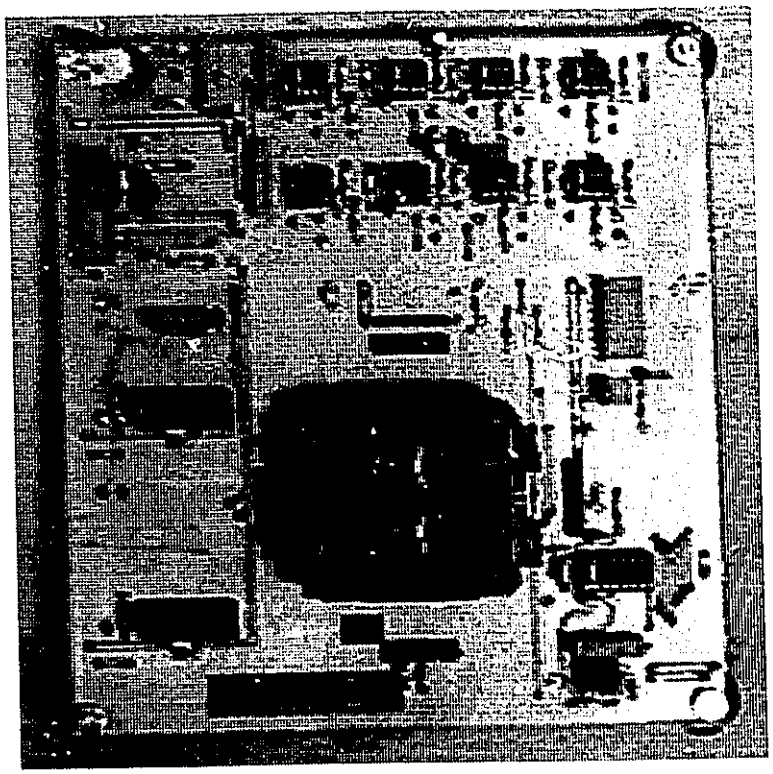


Figure 5.7 AIC Demonstration Board

Due to the AIC design based on digital arithmetic, the structure and the performance of the AIC can be accurately simulated by a computer program. To investigate the performance of the canceler, a Fortran simulation is performed based on integer arithmetic which has the same schematic design as the AIC.

Two different test vectors, a 300 Hz and a 700 Hz sine wave with interference, were used in the simulations. The frequency spectrum of the AIC output due to each of the two test vectors are plotted in Figure 5.9 and Figure 5.10. By

comparison of the frequency spectrum between the input signal and the output signal, the magnitude of interference components were reduced by about 40% in both cases. The canceler can reduce the magnitude of interference regardless of what the input frequency is. This cannot be done with a fixed weight filter. However, the results from the hardware simulation did not resemble the results of software simulation. This deviation between the outputs comes from not enough word length in the signal paths and the truncation errors in binary computation.

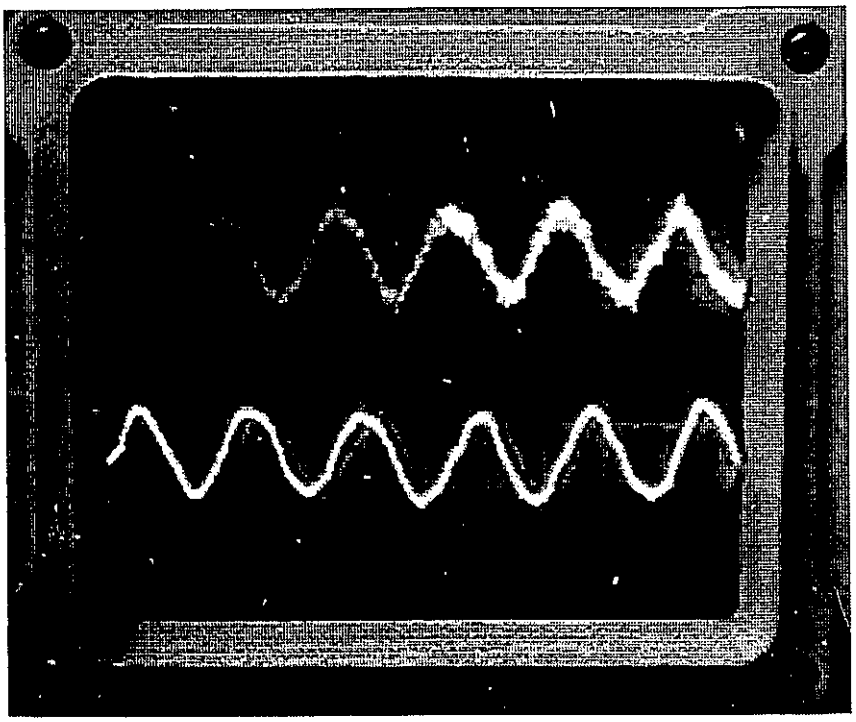
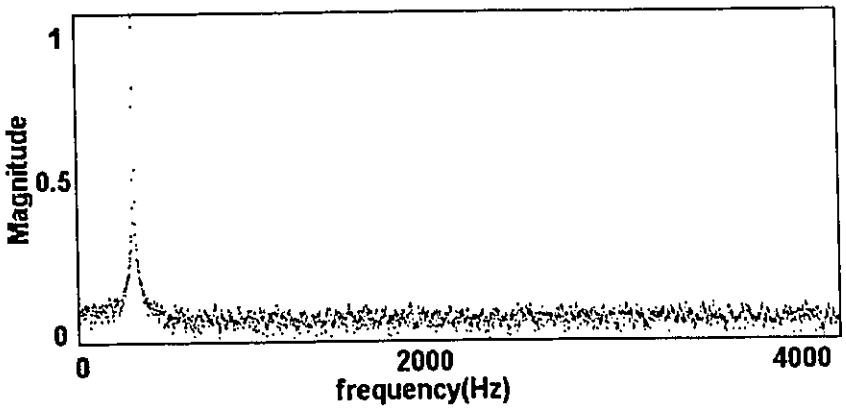
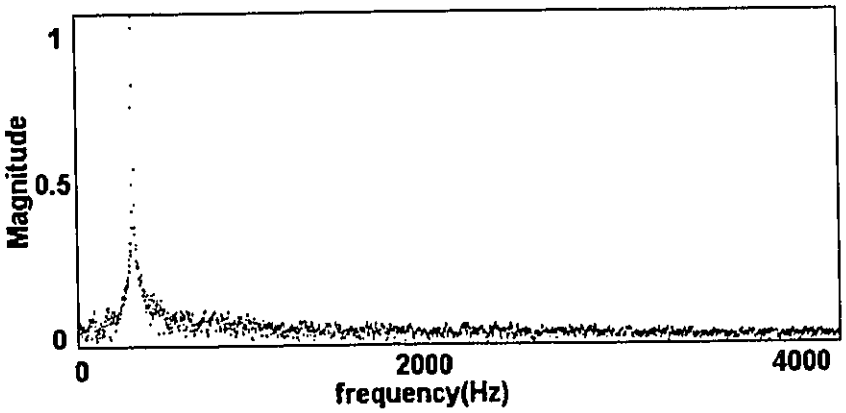


Figure 5.8 Test Result of AIC

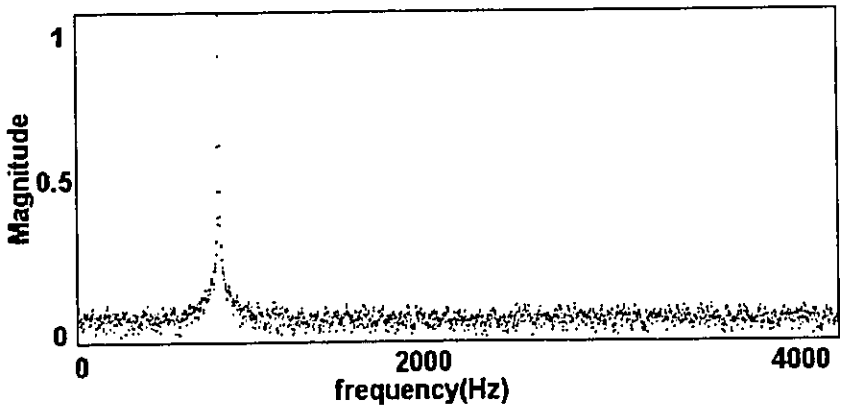


5.9a

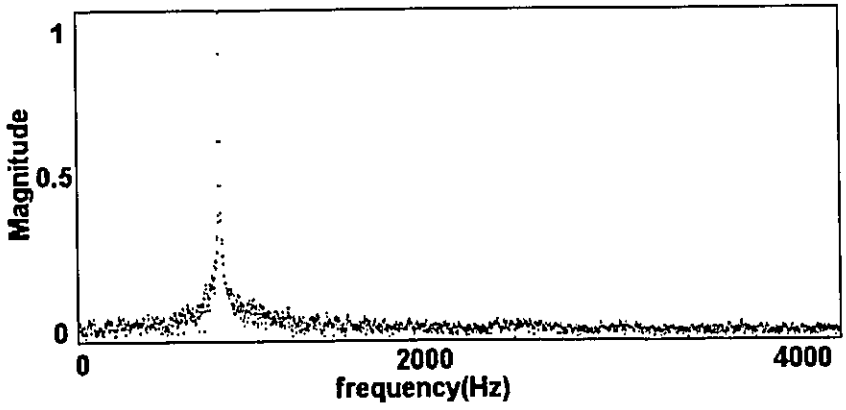


5.9b

Figure 5.9 Hardware Simulation 1.
(a) Frequency Spectrum of Input Signal.
(b) Frequency Spectrum of Output Signal.



5.10a



5.10b

Figure 5.10 Hardware Simulation 2.
(a) Frequency Spectrum of Input Signal.
(b) Frequency Spectrum of Output Signal.

5.3 Summary

The adaptive interference canceler designed in chapter 4 was implemented in hardware using an FPGA chip. Simulation results verified the functionality and the architecture of the canceler. The test results show that the prototype reduced

some of the input interference from the input signal. According to hardware simulations and obtained output waveform, the canceler can reduce the magnitude of input interference to about 40%. To summarize, the 28^{th} order adaptive interference canceler is a successful hardware implementation in FPGA and has proved the usefulness of adaptive filtering.

Chapter 6

Conclusion

The design and the implementation of the adaptive interference canceler (AIC) for interference to a periodic signal using a FPGA has been described in the previous chapters. Using the queue structure AIC can increase the canceler's order from 4 to 28 compared to the direct implementation method (Figure 4.2). The final design of the AIC occupied 80% of the available CLBs and 64% of the available flip-flops in an XC4013 FPGA and it was able to run at an 8 kHz data sampling rate.

6.1 Summary

The design and implementation of an adaptive interference canceler has been presented in this thesis. A brief introduction to the concepts of the adaptive noise cancellation is given in Chapter 2. The advantage of the AIC is that it does not require a reference for the input signal and interference. Due to its simple structure, it is suited for digital implementation.

Chapter 3 presents background material to understand the FPGA technology. FPGA is a programmable device which can be configured within a short period of time. This chapter begins by analyzing the evolution of programmable devices and then proceeds with a discussion on major FPGA architectures, and the CAD flow for implementation circuits.

The adaptive interference canceler design is presented in Chapter 4. This single input canceler generates its reference signal by adding delays in the primary input. Due to the periodic nature of the input signal, the interference component of the reference signal is decorrelated from that in the primary input. The performance of the canceler depends on the order of canceler and the precision of arithmetic. A queue structure canceler is introduced to increase the canceler's order. It is found that the order can subsequently be increased from 4 to 28.

The adaptive interference canceler designed in Chapter 4 was implemented in hardware using an FPGA as described in Chapter 5. Simulation results have verified the functionality and the architecture of the canceler. The successful hardware implementation of the adaptive interference canceler showed that the output signal contains a noticeable reduction of interference compared to that of the input signal. According to hardware simulation, the canceler can remove 40% of interference from the input signal.

6.2. Future Work

In this thesis, the idea of interference cancellation is used to increase the S/N ratio of a corrupted periodic signal. However, this technique can also be extended to other signal processing applications such as interference cancellation for audio signals and echo cancellation in telephone lines.

On the other hand, although the canceler implemented in a Xilinx FPGA is feasible, this is not the best approach in achieving the highest performance. An alternative approach of using a custom layout to build this canceler can provide higher order and operation speed than those by FPGA implementation. It is possible to extend this thesis work by creating a chip of an adaptive interference canceler using CMOS4S process. As well, some other FPGA vendors can be

investigated where perhaps a different FPGA would be more suitable than Xilinx parts in realizing this canceler.

References

[1] B. Widrow *et al.*, "Adaptive Noise Canceling: Principles and Applications", *Proc. IEEE*, vol. 63, pp.1962-1716, Dec. 1975.

[2] Bernard Widrow, Samuel D.Stearns, *Adaptive Signal Processing*, Prentice-Hall, 1985.

[3] Bernard Widrow, John M. McCool, Michael G. Larimore and C.Richard Johnson, "Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter", *Proc. IEEE*, vol. 64, No. 8, pp.1151-1162, Aug, 1976.

[4] Claude S. Lindquist, *Adaptive & Digital Signal Processing with Digital Filtering Applications*, Steward & Son, 1989.

[5] Marvin R. Sambur, "Adaptive Noise Canceling for Speech Signals", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-26, No. 5, pp.419-423, Oct, 1978.

[6] Stephen D. Brown, Robert J. Francis, Jonathan Rose, Zvonko G. Vranesic, *Field Programmable Gate Arrays*, Prentice Hall, 1992.

[7] *The Programmable Logic Data Book*, Xilinx Co., 1994.

[8] Jesse H. Jenkins, *Designing with FPGAs and CPLDs*, Prentice Hall, 1994.

[9] E. Hamdy, J. McCollum, S. Chen, S. Chiang, S. Eltoukhy, J. Chang, T. Speers and A. Mohsen, "Dielectric Based Antifuse for Logic and Memory ICs", *International Electron Devices Meeting Technical Digest*, pp. 786-789, 1988.

[10] J. Birkner, A. Chan, H.T. Chua, A. Chao, K. Gordon, B. Kleinman, P. Kolze, R. Wong, "A Very High-Speed Field Programmable Gate Array Using Metal-To-Metal Anti-Fuse Programmable Elements", *New Hardware Product Introduction at CICC'91 Custom Integrated Circuits Conference 91*, May 1991.

[11] *The XACT Development System Reference Guide*, Xilinx Co., 1994.

[12] *Viewlogic Interface User Guide*, Xilinx Co., 1994.

[13] Steven F. Boll, Dennis C. Pulsipher, "Suppression of Acoustic Noise in Speech using Two Microphone Adaptive Noise Cancellation", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, No. 6, pp.752-753, Dec, 1980.

[14] Sergio Franco, *Design with Operational Amplifiers and Analog Integrated Circuits*, Mcgraw-Hill, 1998.

[15] Ronald J. Tocci, *Digital Systems Principle and Applications*, Prentice-Hall Inc., 1991.

[16] *ADC804LCN Data Sheet*, National Semiconductor, May 1989.

[17] *DAC0800LCN Data Sheet*, National Semiconductor, March 1989.

Appendix A

Plots of Simulation Results

Functional Simulations

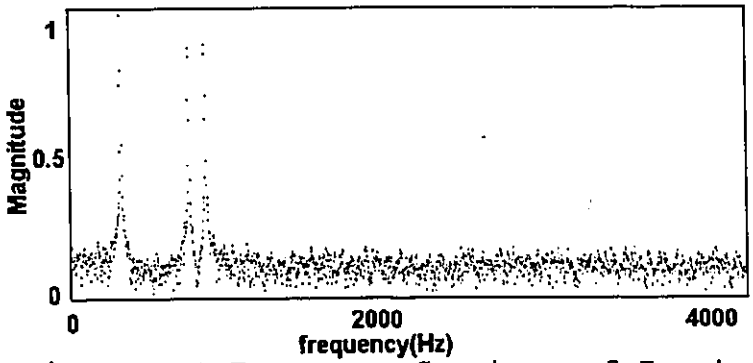


Figure A.1 Frequency Spectrum of Input Signal for Simulation 1-7,11-12

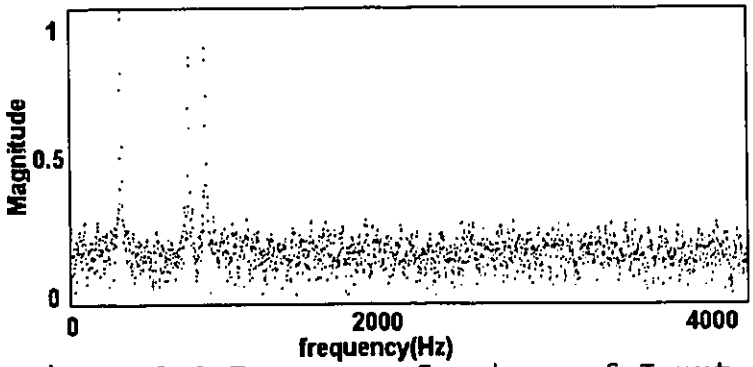


Figure A.2 Frequency Spectrum of Input Signal for Simulation 8

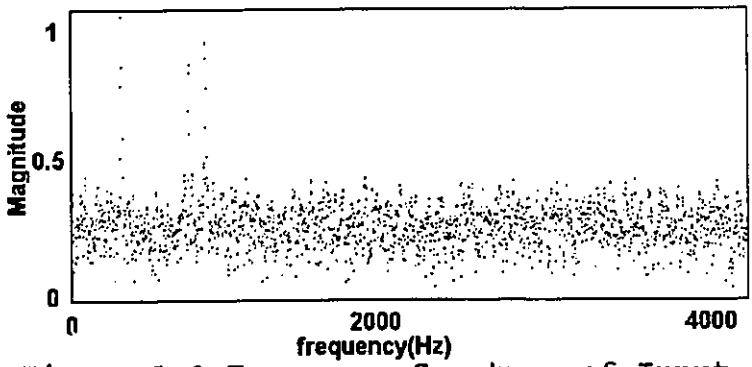


Figure A.3 Frequency Spectrum of Input Signal for Simulation 9

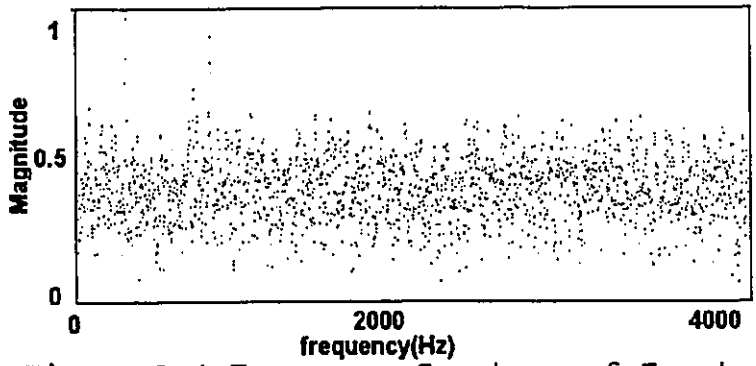


Figure A.4 Frequency Spectrum of Input Signal for Simulation 10

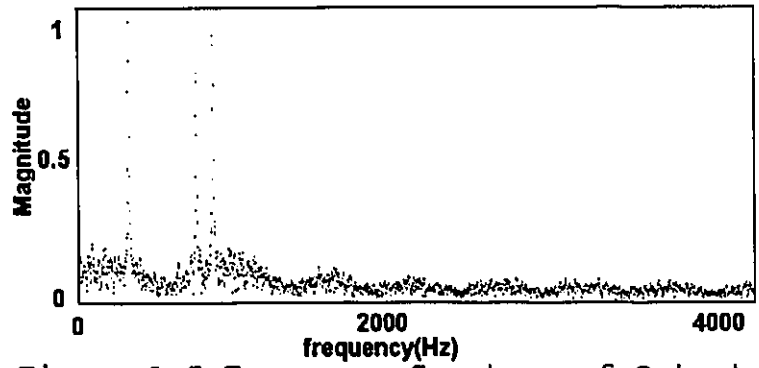


Figure A.5 Frequency Spectrum of Output Signal for Simulation 1

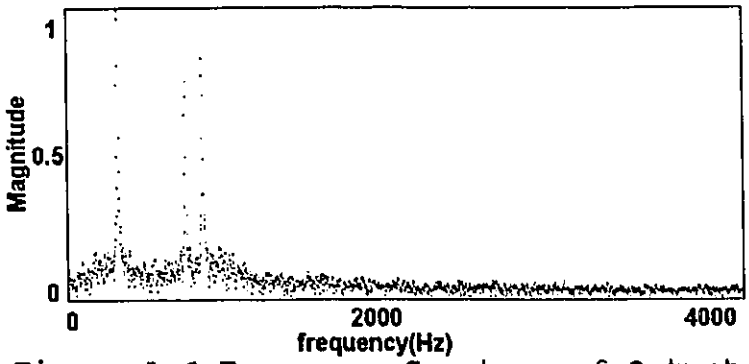


Figure A.6 Frequency Spectrum of Output Signal for Simulation 2

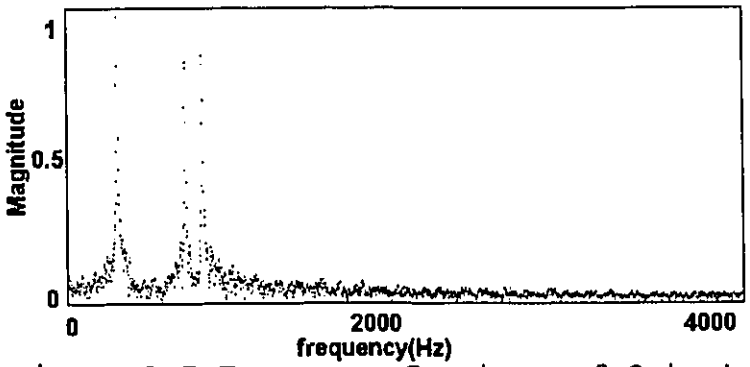


Figure A.7 Frequency Spectrum of Output Signal for Simulation 3

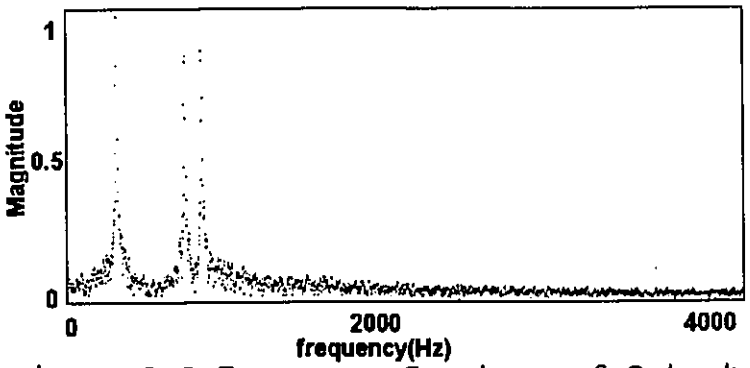


Figure A.8 Frequency Spectrum of Output Signal for Simulation 4

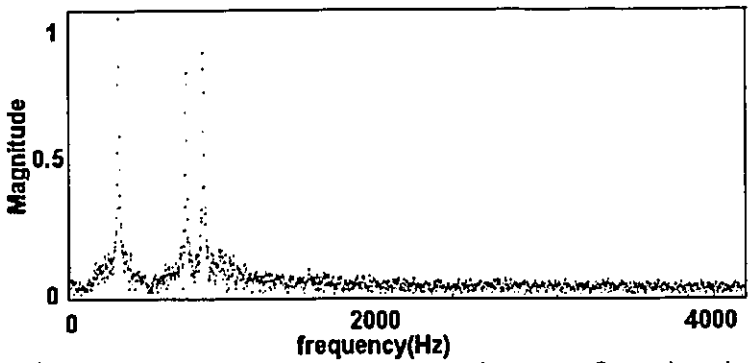


Figure A.9 Frequency Spectrum of Output Signal for Simulation 5

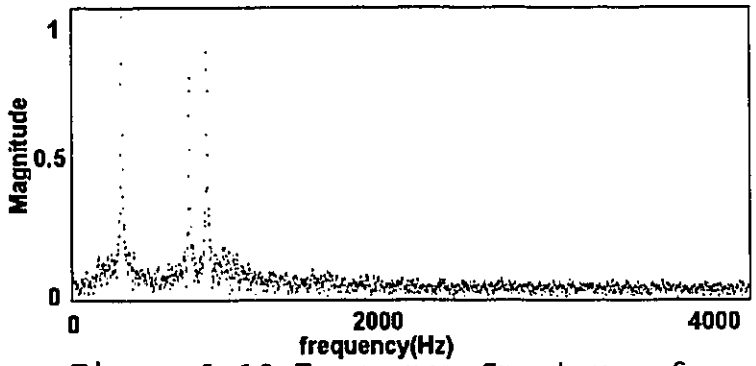


Figure A.10 Frequency Spectrum of Output Signal for Simulation 6

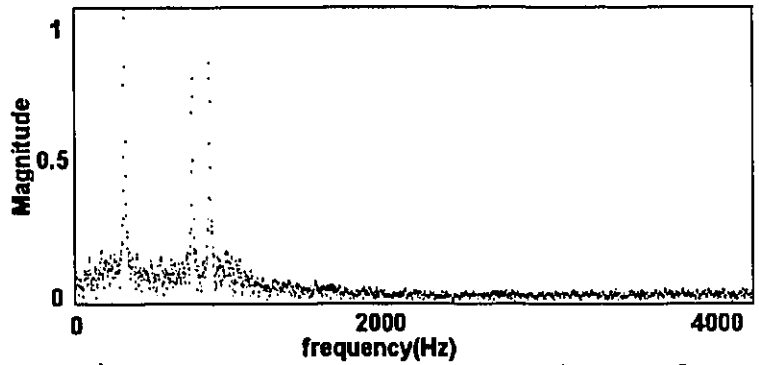


Figure A.11 Frequency Spectrum of Output Signal for Simulation 7

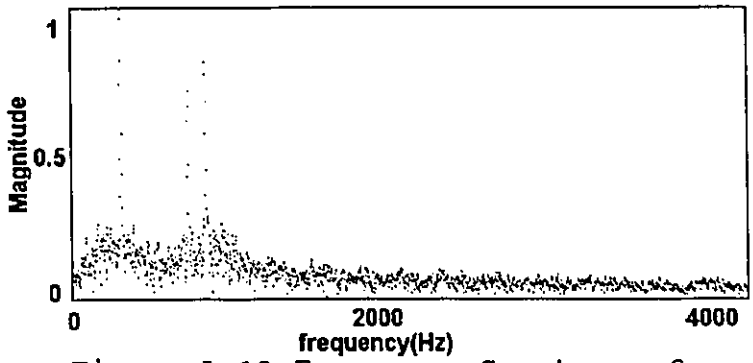


Figure A.12 Frequency Spectrum of Output Signal for Simulation 8

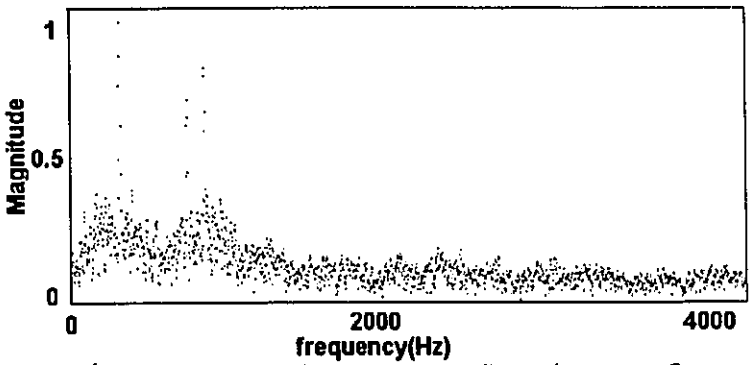


Figure A.13 Frequency Spectrum of Output Signal for Simulation 9

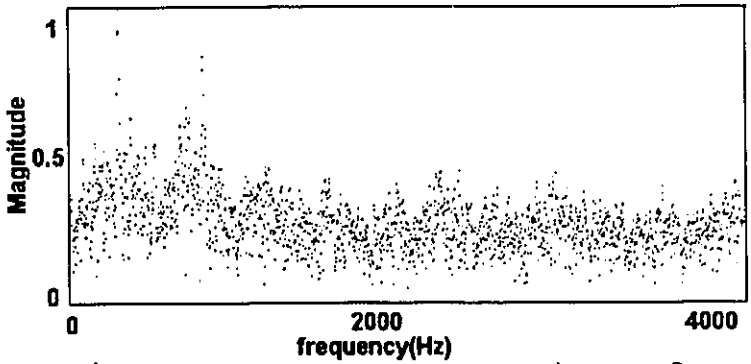


Figure A.14 Frequency Spectrum of Output Signal for Simulation 10

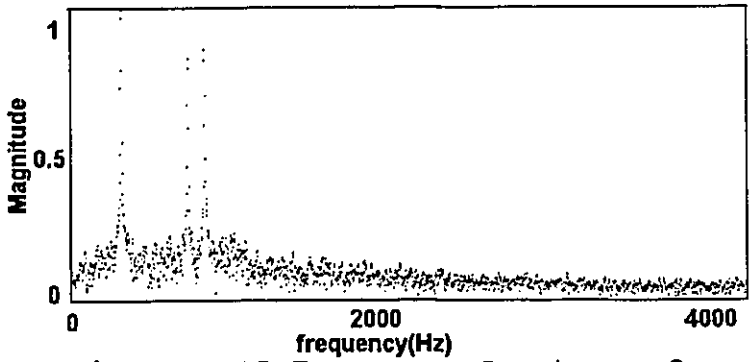


Figure A.15 Frequency Spectrum of Output Signal for Simulation 11

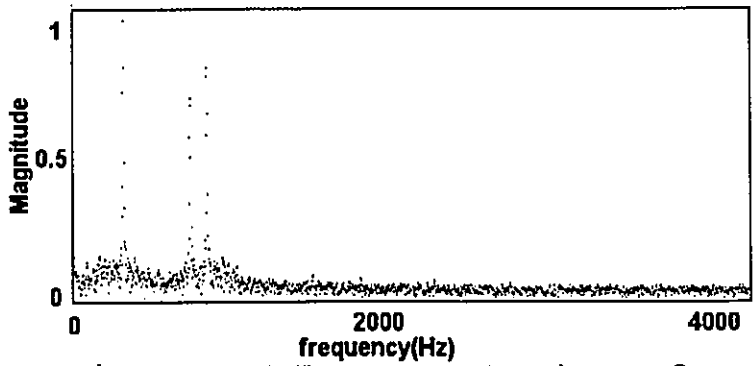
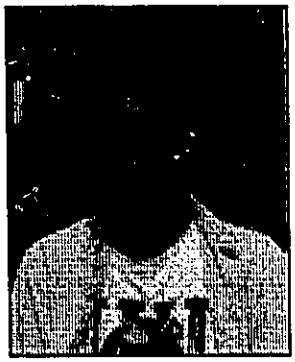


Figure A.16 Frequency Spectrum of Output Signal for Simulation 12

Vita Auctoris

Name: Tony Wai Sing Yuen



Place of Birth: Hong Kong

Year of Birth: 1966

Education: Chinese YMCA Secondary School, Hong Kong
1985-1987

W.D. Lowe Secondary School, Windsor,
Ontario, 1987-1988

University of Windsor, Windsor, Ontario
1988-1992 B.A.Sc.

University of Windsor, Windsor, Ontario
1992-1995 M.A.Sc.