Finite WordLength Design and VLSI Implementation of Wave Digital Filters

By

A. R. Mirzai

A Thesis Submitted for the Degree of

Doctor of Philosophy

School of Electrical Engineering and

Applied Physics

The Centre for Information Engineering

The City University

October 1986

I dedicate this work to my [redacted] [redacted] [redacted] for her love, patience, encouragements and understanding and to our [redacted] [redacted] [redacted] who has given a new meaning to our lives.

CONTENTS

List of Tables (i)

List of Illustrations (ii)

Acknowledgements (ix)

Declaration (x)

Abstract (xi)

Abbreviations (xii)

Chapter 1 : Digital Filters and Array Processors 1

1.1.0- Introduction 1

1.2.0- Theory of LTI Digital Filters 4

- 1.2.1- Sampled Signals 4
- 1.2.2- Z-Transform 5
- 1.2.3- Digital Filter Configurations 8
- 1.2.4- Digital Filter Realizations 9
- 1.2.5- Design of Digital Filters from Prototype Analogue Filters 13
- 1.2.6- Finite Wordlength Effects 20
- 1.2.7- Summary 23

1.3.0- Wave Digital Filters (WDFs) 24

- 1.3.1- Introduction 24
- 1.3.2- Theory and Design of WDFs (I) 26
- 1.3.3- Theory and Design of WDFs (II) 35
- 1.3.4- Comments on WDF Design Approaches 41
- 1.3.5- Review of Hardware Implementation of WDFs 41

1.4.0- VLSI Array Processing 44

- 1.4.1- Introduction 44
- 1.4.2- Systolic Arrays 46
- 1.4.3- Wavefront Arrays 48
- 1.4.4- Programmable Array Processors 49
- 1.4.5- Comparison of Systolic and Wavefront Arrays 51

1.5.0- Objectives in this Thesis 52

Chapter 2 : Optimization and Systolic Implementation

2.1.0- Introduction 54
2.2.0- Use of optimization techniques in the Design of Digital Filters 56
2.2.1- Introduction 56
2.2.2- Characteristics of an Optimization Algorithm 58
2.2.3- Error Criterion 60
2.2.4- Statement of the Design Problem 61

2.3.0- Direct Search Methods for the Design of WDFs 65
2.3.1- Introduction 65
2.3.2- Hooke and Jeeves' Method 65
2.3.3- Example 68

2.4.0- Development of some Basic Systolic Arrays 74
2.4.1- Introduction 74
2.4.2- The Basic Systolic Arrays 75

2.5.0- Universal Systolic Array 89
2.5.1- Initialization of the Universal Systolic Array 89

2.6.0- Summary 95

Chapter 3 : Unit Element and Lattice WDFs 97

3.1.0- Introduction 97
3.2.0- 2-Port Systolic Adaptors 100
3.2.1- Single Board 2-port Systolic Adaptor 103
3.2.2- CMOS Implementation of a 2-Port Systolic Adaptor 104

3.3.0- Unit Element WDFs (UEWDFs) 108
3.3.1- Basic Theory 108
3.3.2- Finite WordLength Design of UEWDFs 112

3.4.0- Lattice WDFs (LTWDFs) 115
3.4.1- Introduction 115
3.4.2- Basic Theory 116

3.4.3- Finite WordLength Design of LTWDFs 118

3.5.0- Implementation of Complete Filters 121

3.5.1- UEWDFs 121

3.5.2- LTWDFs 123

3.6.0- Description of UEFD.SO and LTFD.SO Programs 126

3.7.0- Analysis and Simulation Results 131

3.7.1- Example 3.1 132

3.7.2- Example 3.2 136

3.7.3- Example 3.3 138

3.7.4- Example 3.4 143

3.7.5- Example 3.5 143

3.7.6- Example 3.6 146

3.8.0- Discussion and comments 153

Chapter 4 : LC-ladder WDFs 155

4.1.0- Introduction 155

4.2.0- 3-port Systolic Adaptors 157

4.2.1- 3-port Parallel Systolic Adaptors 157

4.2.2- 3-port Serial Systolic Adaptors 160

4.2.3- Universal Systolic Adaptor 163

4.2.4- CMOS Implementation of the adaptors 166

4.3.0- LC-ladder WDFs (LCWDFs) 169

4.3.1- Introduction 169

4.3.2- Direct LC-ladder WDFs 169

4.3.2- LCWDFs with Inserted Unit Elements 174

4.3.4- Finite WordLength Design of WDFs (FWLD) 181

4.4.0- Hardware Implementation of Systolic LCWDFs 183

4.5.0- Description of LCFD.SO Program 184

4.6.0- Analysis and Simulation Results 187

4.6.1- Example 4.1 188

4.6.2- Example 4.2 190

4.6.3- Example 4.3 192

4.6.4- 3-port and Universal Systolic Adaptors Simulation Results 194

4.7.0- Discussion and Comments 201

Chapter 5 : Frequency Transformations 204

5.1.0- Introduction 204

5.2.0- High-Pass WDFs 207

5.3.0- Band-Pass WDFs 216

5.4.0- Band-Stop WDFs 222

5.5.0- Synthesis of WDFs 226

5.6.0- Summary 229

Chapter 6 : Conclusions and Comments 230

6.1.0- Summary 230

6.2.0- Further Work 234

6.3.0- Practical Applications of WDFs 237

References 238

Bibliography 245

Appendix 'A' : Analysis of WDFs 247

Appendix 'B' : Program Listings 256

Appendix 'C' : Scattering Matrix 307

Appendix 'D' : List of Author's Publications 310

LIST OF TABLES

1.1- Wave realisation of some basic elements. 27

2.1- 2's complement subtraction 86

3.1- Estimation of number of transistors and speed of the 2-port systolic adaptor. 107

3.2- Specifications of the filter design examples. 131

4.1- Estimation of number of transistors and speed of the systolic adaptors. 168

4.2- Kuroda's Transformation. 175

4.3- Specifications of the filter design examples. 186

5.1- Effects of Low-pass to Band-pass transformation. 221

LIST OF ILLUSTRATIONS

1.1- Sampling process. 6

1.2- Frequency spectrums of a typical signal. 6

1.3- Convolution-Summation property. 6

1.4- Typical lowpass specifications. 10

1.5- Direct realisation of a digital filter. 10

1.6- Cononic direct form. 12

1.7- Cascade form. 12

1.8- Parallel form. 12

1.9- Bilinear transformation. 15

1.10- Frequency warping effect. 15

1.11- LC-ladder of a) odd, and b) even order. 17

1.12- Block representation of a doubly terminated network. 17

1.13- Passband characteristics of a lossless filter. 17

1.14- Practical use of a digital filter. 21

1.15- a) 2-port interconnection. 30
b) Schematic representation of a 2-port adaptor. 30
c) One possible realisation of a 2-port adaptor. 30

1.16- a) N-port parallel interconnection. 32
b) Schematic representation of a 3-port parallel adaptor. 32
c) One possible realisation of a 3-port parallel adaptor. 32

1.17- a) N-port serial interconnection. 34
b) Schematic representation of a 3-port serial adaptor. 34
c) One possible realisation of a 3-port serial adaptor. 34

1.18- a) A typical LC-ladder filter. 36
b) corresponding WDF derived from Fig. 1.18a. 36

1.19- General representation of a 2-port network. 36

1.20- Digital equivalent of a series capacitor. 40

1.21- a) A typical analogue filter. 41
b) Corresponding WDF derived from (a). 41

1.22- Inner- product cell. 47

2.1- Features of an optimization procedure. 59

2.2- Flow-chart of an Exploratory move. 67

2.3- Flow-chart of a Pattern move. 69

2.4- a) Systolic array to implement R1. 77
b) Basic cell in Fig. 2.4a. 79
c) Evaluation of $r1_2$. 79

2.5- a) Modified array to allow signed number to be used. 82
b) Modified basic cell. 82

2.6- a) Systolic array to implement R2. 84
b) Basic cell. 84

2.7- a) Systolic array to implement R3. 85
b)Basic cell. 85

2.8- a), b) and c) Schematic representation of the systolic arrays. 88
d) Triangular arrangment of latches. 88

2.9- a) Universal systolic array. 90
b) Basic cell of universal systolic array. 91
c) Schematic representation of universal systolic array. 91

3.1- Schematic representation of, 100
a) a 2-port adaptor. 100
b) a 2-port systolic adaptor. 100

3.2- Logic diagram of the basic cell. 102

3.3- Photograph of a 2-port systolic adaptor hardware. 105

3.4- Cascade transmission line filter. 109

3.5- a) Unit element WDF. 109
b) schematic representation. 109

3.6- UEWDF with full delay. 111

3.7- A typical UEWDF frequency response. 111

3.8- A 2-port network. 111

3.9- Lattice analogue filter. 117

3.10- Wave realisation of Fig. 3.9. 117

3.11- LTWDF with A2=0 117

3.12- a) Nth order LTWDF using 2-port adaptor. 119a
b) First degree all-pass section. 119a
c) Second degree all-pass section. 119a

3.13- LTWDF realisation using first and second degree all-pass sections. 120

3.14- 7th order UEWDF using 2-port systolic adaptor. 122

3.15- Multiplexed UEWDF using a 2-port systolic adaptor. 124

3.16- Nth order bireciprocal LTWDF. 125

3.17- a) Main menu of UEFD.SO program. 127
b) Main menu of LTFD.SO program. 127

3.18- Main menu of ANAWDF program. 129

3.19- Main menu of SIMWDF program. 129

3.20- Frequency responses for example 3.1. 134

3.21- Impulse responses for example 3.1.
a) With synthesis coefficients. 135
b) With FWLD program coefficients. 135

3.22- Frquency responses for example 3.2. 137

3.23- Impulse responses for example 3.2 with synthesis coefficients. 139

3.24- a) 0.08 HZ Sinewave. 140
b) 0.30 HZ Sinewave. 140
c) Input. 140
d) Output. 140

3.25- Frequency responses for example 3.3. 141

3.26- Impulse responses for example 3.3.
a) With synthesis coefficients. 142
b) With FWLD program coefficients. 142

3.27- Frequency responses for example 3.4. 144

3.28- Impulse responses for example 3.4 with synthesis coefficients. 145

3.29- Step responses for example 3.4. 145

3.30- Frequency responses for example 3.5. 147

3.31- Impulse responses for example 3.5 with FWLD program coefficients. 148

3.32- a) 0.10 HZ Sinewave. 149
b) 0.40 HZ Sinewave. 149
c) Input. 149
d) Output. 149

3.33- Frequency responses for example 3.6. 151

3.34- Impulse responses for example 3.6.
a) With synthesis coefficients. 152
b) With FWLD program coefficients. 152

4.1- Schematic representation of,
a) a 3-port parallel adaptor. 158
b) a 3-port parallel systolic adaptor. 158

4.2- Logic diagram of the basic cell in a 3-port parallel systolic adaptor. 159

4.3- Schematic representation of,
a) a 3-port serial adaptor. 161
b) a 3-port serial systolic adaptor. 161

4.4- Logic diagram of the basic cell in a 3-port serial systolic adaptor. 162

4.5- Logic diagram of the basic cell in a universal systolic adaptor. 164

4.6- Schematic representation of universal systolic adaptor. 165

4.7- a) 3rd order LC-ladder filter. 170
b) Corresponding WDF. 170

4.8- a) 3-port parallel adaptor with one multiplier. 173
b) 3-port serial adaptor with one multiplier. 173

4.9- Process of Kuroda's transformation. 176

4.10- WDF corresponding to Fig. 4.9d. 177

4.11- a) 4th order LC-ladder filter. 179
b) Use of UEs for Kuroda's transformation. 179
c) LC-ladder filter with inserted UEs. 179
d) Corresponding WDF using 3-port parallel adaptors only. 179

4.12- Sample example from the synthesis program. 180

4.13- Main menu of the LCFD.SO program. 186

4.14- Frequency responses for example 4.1. 190

4.15- Frequency responses for example 4.2. 191

4.16- Frequency responses for example 4.3. 193

4.17- Sample examples from the 3-port parallel systolic WDF adaptor program. 195

4.18- Sample examples from the 3-port serial systolic WDF adaptor program. 196

4.19- Sample results from the universal systolic adaptor program simulating,
a) a 2-port adaptor. 198
b) a 3-port parallel adaptor. 199
c) a 3-port serial adaptor. 200

5.1- Two approaches to the design of frequency selective digital filters. 206

5.2- General block representation of a WDF. 206

5.3- The output and the complementery output of a,
a) UEWDF. 208

b) LCWDF. 208
c) LTWDF. 208

5.4- a) Low-Pass UEWDF. 212
b) Corresponding High-Pass filter. 212

5.5- a) 3rd order Low-Pass LC-ladder filter. 214
b) Corresponding 3rd order High-Pass filter. 214

5.6- High-Pass LCWDF realised using only 3-port parallel adaptors. 214

5.7- High-Pass LCWDF realised using only 3-port serial adaptors. 215

5.8- Wave realisation of eqn. 5.4. 216

5.9- a) 5th order Low-Pass UEWDF. 218
b) Corresponding 10th order High-Pass UEWDF with f_0=0.2. 218

5.10- a) Band-Pass UEWDF corresponding to Fig.5.4a. 219
b) Multi-band UEWDF. 219

5.11- a) 3rd order Low-Pass LC-ladder filter. 221
b) Corresponding 6th order Band-Pass LC-ladder filter. 221

5.12- Wave realisation of eqn. 5.6. 222

5.13- a) 5th order Low-Pass UEWDF. 224
b) Corresponding 6th order Band-Pass LC-ladder filter. 224

5.14- a) Band-Stop UEWDF corresponding to Fig.5.4a. 225
b) Multi-band UEWDF. 225

5.15- a) 12th order LC-ladder filter with UEs. 229
b) Corresponding WDF. 229

A1.1- Kth section of a UEWDF. 247

A1.2- Nth order UEWDF. 247

A2.1- a) First degree all-pass section. 251

b) Second degree all-pass section. 251

A2.3- LTWDF realisation using first and second degree all-pass sections. 251

A3.1- Kth section of a LCWDF. 255

C.1- a) General 2-port network. 308
b) Corresponding WDF. 308

ACKNOWLEDGMENTS

My deepest appreciation and thanks goes to my supervisor and colleague Dr. Stuart. S. Lawson for his encouragement, guidance and helpful advice over the past three years.

I wish to thank [redacted] [redacted] [redacted] [redacted] and [redacted] [redacted] for their academic assistance and help.

Thanks are due to [redacted] [redacted] [redacted] for his careful review of the entire thesis and I would like to acknowledge many stimulating discussions with other staff members and research students in the Centre for Information Engineering, in particular [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted].

I also wish to acknowledge the financial support from the Science Engineering Research Council (SERC).

Finally, I am indebted to my parents and my family for their encouragement, support and patience during my stay in this country.

DECLARATION

I grant powers of discretion to the University Librarian to allow this thesis to be copied in whole or in part without further reference to me. This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement.

ABSTRACT

Electronic filters are one of the basic elements in a communication system. In recent years, digital filters have attracted much attention due to many reasons, such as stability, flexibility, speed, cost, etc. One major problem with digital filters is the effects of finite wordlength. Wave Digital Filters (WDFs) were first introduced by Fettweis in 1971 to reduce these effects. However, the main drawback of WDFs is the hardware complexity when compared with the conventional cascade of second order sections. In general, the implementation of WDFs depends on how efficient the 2-port, 3-port parallel and 3-port serial adaptors are implemented. Therefore, one way of approaching the hardware complexity of WDFs is to consider the VLSI implementation of WDF adaptors.

In this thesis, bit-level systolic arrays are developed for the implementation of WDF adaptors. The systolic arrays developed are very suitable for the VLSI implementation of WDFs. A 2-port prototype systolic adaptor has been constructed and tested fully to prove the correctness of the design. Also, a universal systolic adaptor is designed which can be programmed to realise 2-port, 3-port parallel and 3-port serial adaptors. The number of transistors required to implement the adaptors in CMOS technology and the speed of the adaptors has also been estimated.

Also in this thesis, a complete software package has been developed which can be used for the synthesis and finite wordlength design of WDFs based on three well known reference filters, i.e unit element, lattice and LC-ladder filters. Software tools are also developed for the analysis and simulation of the filters designed. The simulation program allows the simulation of the systolic WDFs.

Many examples have been considered to illustrate the performance of the design programs and the systolic WDFs. From the example, it will be shown that the finite wordlength design program can be used to minimize the number of bits used to represent the filter coefficients. Also, it will be seen that a small reduction in the number of bits for the coefficients would exponentially reduce the complexity, and consequently the number of transistors, of a systolic WDF.

ABBREVIATIONS

A/D	Analogue to Digital converter.
Ap	Pass band ripple.
As	Stop band Loss.
BPF	Band-Pass Filter.
BSF	Band-Stop Filter.
CMOS	Complementary Metal Oxide Semiconductor.
D/A	Digital to Analogue converter.
EPROM	Erasable Programmable Read Only Memory.
FA	Full Adder.
FIR	Finite Impulse Response.
FIRST	Fast Implementation of Real-time Signal Transforms.
Fp	Pass band edge frequency.
Fs	Stop band edge frequency.
FWLD	Finite WordLength Design.
HPF	High-Pass Filter.
IIR	Infinite Impulse Response.
I/O	Input/Output.
LCWDF	LC-ladder WDF.
LPF	Low-Pass Filter.
LSB	Least Significant Bit.
LSI	Large Scale Integration.
LTI	Linear Time Invariant.
LTWDF	LaTtice WDF.
MNB	Maximum Number of Bits.
MSB	Most Significant Bit.
NBC	Number of Bits for Coefficients.
NBS	Number of Bits for Signal.
VLSI	Very Large Scale Integration.
UEWDF	Unit Element WDF.
USA	Universal Systolic Adaptor.
WDF	Wave Digital Filter.

CHAPTER ONE

DIGITAL FILTERS AND ARRAY PROCESSORS

1.1.0- Introduction

Filters are signal processors that enhance some frequency components in a signal while attenuating the others. In analouge filters, the signals may be continuous functions of time for example current or voltage waveforms. These signals are called continuous-time signals. On the other hand, the input to a digital filter is represented by a sequence of values available only at discrete intervals of time. These type of signals are referred to as discrete-time signals. Therefore, a digital filter is a digital signal processor that converts a sequence of numbers, called input samples, into another sequence of numbers called output samples. This computational process may be that of low-pass filtering, band-pass filtering, interpolation, etc.

Digital filters can also be used to process continuous signals by means of A/D and D/A converter, as will be shown in section 1.2.6 (Fig. 1.14). There are a number of reasons for considering the filtering of a continuous signal using digital techniques,

1) The frequency response of a digital filter can be made as close as possible to the ideal response.

2) Adaptive filtering and linear phase characteristics are possible.

3) Also digital filters are programmable and very flexible.

4) When designed, digital filters are very stable and they are not subjected to ageing.

5) The cost of analogue filters is static while, with advances in digital hardware, the cost of digital filters is decreasing rapidly.

This chapter presents a concise introduction to the fundamental techniques involved in the design and implementation of digital filters. The chapter is divided into four sections. In the first section, we begin with a review of the sampling process of a continuous signal. Also we introduce the concept of the Z-transform as applied to linear time-invariant discrete systems and discuss methods by which a continuous filter design can be translated into a digital filter design. There are a number of problems associated with the realization of digital filters, such as coefficient accuracy, quantization and rounding, etc, which affect the frequency response of the filter. These problems and their solutions are also discussed.

As a solution to these problems, the concept of Wave Digital filters was introduced in 1971 by Fettweis. In section 2, we briefly review the basic theory, design and implementation of WDFs. In section 3, the concept of VLSI array processing is introduced. Some features of suitable architectures for VLSI implementation are discussed and systolic and wavefront arrays are described and compared. Finally in section 4, we outline the main objectives of this thesis.

1.2.0- <u>Theory of LTI Digital Filters [1]</u>

1.2.1- Sampled Signals

The sampling process can be thought of as the impulse modulation of a continuous input signal (Fig. 1.1). The input, $x(t)$, is sampled every T seconds to produce the output signal, $x_s(t)$. From Fig. 1.1, $x_s(t)$ is given by,

$$x_s(t) = x(t).\delta_T(t) \qquad (1.1)$$

where $\delta_T(t) = \Sigma\ \delta(t-nT) \qquad (1.2)$

and $\delta(t-nT)$ is the Dirac delta function. Substituting eqn. 1.2 into eqn. 1.1, we have,

$$x_s(t) = x(t)\ \Sigma\ \delta(t-nT) \qquad (1.3)$$

or $x_s(t) = \Sigma\ x(nT).\delta(t-nT) \qquad (1.4)$

assuming $x(t)=0$ for $t<0$ and $x(t)$ is only known at $t=nT$. Eqn. 1.4 represents the time-domain characteristics of the sampled output. In order to see the frequency characteristics, we need to apply the Laplace transform to eqn. 1.4. $\delta_T(t)$ can be expressed as a Fourier series as shown below,

$$\delta_T(t) = (1/T)\ \Sigma\ e^{jn\omega_s t} \qquad n=(-\infty,\infty) \qquad (1.5)$$

where ω_s is the sampling frequency in Rad/sec.

Substituting (1.5) into (1.1), we have,

$$x_s(t) = (1/T)\ \Sigma\ x(nT)e^{jn\omega_s t} \qquad (1.6)$$

Now taking the Laplace transform of (1.6) and using the shifting theorem we obtain,

$$X_s(s) = (1/T)\ \Sigma\ X(s-jn\omega_s) \qquad (1.7)$$

Substituting $s=j\omega$ into (1.7) results,

$$X_s(j\omega) = (1/T)\ \Sigma\ X[j(\omega-n\omega_s)] \qquad (1.8)$$

Therefore the sampling process has resulted in a frequency spectrum which is a periodic function with a period of ω_s. Fig. 1.2 shows the specturm of a typical signal, x(t), and the corresponding $X_s(s)$ for two cases, $f>2f_{min}$ and $f<2f_{min}$ where f is the sampling frequency and f_{min} is the maximum frequency component in the input signal. In the first case, the input can be reconstructed since the frequency spectrum of $x_s(t)$ can be recognised (Fig. 1.2b). In the second case, the signal cannot be recovered since the spectra of the signal are overlapping (Fig. 1.2c). This effect is known as aliasing and its effect can be reduced by bandlimiting the input signal (Fig. 1.2d), and/or increasing the sampling frequency. From Fig. 1.2, it can be deduced that the sampling frequency has to be at least equal to $2f_{min}$ in order to be able to recover the signal. This is referred to as the Nyquist sampling theorem.

1.2.2- Z-Transform

In continuous-time domain, filters are described using sets of linear differential equations and the Laplace transform can be used to describe the frequency characteristics of the filters. Digital filters are however described using linear difference equations and the z-transform provides information about the

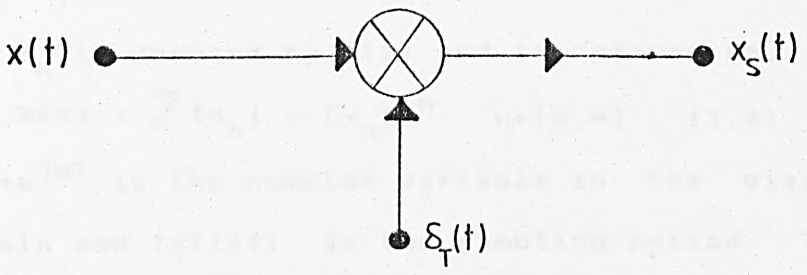


Fig. 1.1 _ Sampling process.

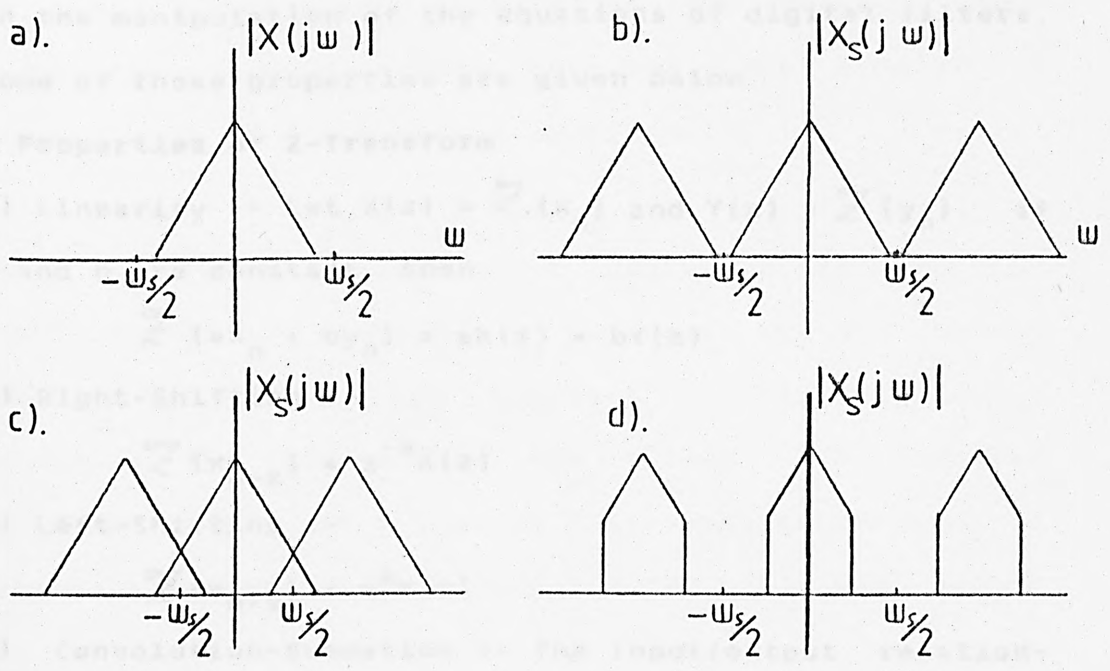
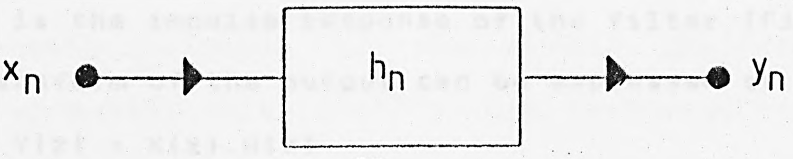


Fig. 1.2 _ Frequency spectrums of a typical signal.



$$y_n = \sum_{i=0}^{\infty} h_n \; x_{n-i}$$

Fig.1.3 _ Convolution _ Summation property.

frequency response of the filters. The z-transform of a sequence x_n is denoted by $X(z)$ and is defined as,

$$X(z) = \mathcal{Z}\{x_n\} = \Sigma x_n z^{-n} \qquad n=[0,\infty] \qquad (1.9)$$

where $z=e^{j\omega T}$ is the complex variable in the discrete-time domain and $T=(1/f)$ is the sampling period. The z-transform has a number of properties which are useful in the manipulation of the equations of digital filters. Some of these properties are given below.

- **Properties of Z-Transform**

a) Linearity :- Let $X(z) = \mathcal{Z}\{x_n\}$ and $Y(z) = \mathcal{Z}\{y_n\}$. If a and b are constant, then

$$\mathcal{Z}\{ax_n + by_n\} = aX(z) + bY(z)$$

b) Right-Shifting :-

$$\mathcal{Z}\{x_{n-k}\} = z^{-k}X(z)$$

c) Left-Shifting :-

$$\mathcal{Z}\{x_{n+k}\} = z^{k}X(z)$$

d) Convolution-Summation :- The input/output relationship of a digital filter can be expressed using the convolution-summation as described below :-

$$y_n = x_n * h_n$$

where h_n is the impulse response of the filter (Fig 1.3). The z-transform of the output can be expressed as :-

$$Y(z) = X(z).H(z)$$

e) Multiplication by A^n :-

$$\mathcal{Z}\{A^n x_n\} = X(A^{-1}z)$$

1.2.3- Digital filter Configurations

- **Transfer Functions**

As mentioned earlier, linear differential equations are used to describe analogue filters, while linear difference equations are used for digital filters. The linear difference equations express the output samples of the digital filter in terms of the present input sample and a number of past input and output samples. A typical form of a difference equation is,

$$y_n = \Sigma a_i x_{n-i} + \Sigma b_j y_{n-j} \qquad (1.10)$$

where x_n is the present input sample, x_i is the ith input sample and similarly y_j represent the jth output sample. A_i and b_j are constant coefficients which determine the response of the filter. The transfer function, G(z), of a digital filter may be obtained by taking the z-transform of eqn. 1.10. This will result in,

$$G(z) = \frac{Y(z)}{X(z)} = \frac{\Sigma a_i Z^{-i}}{1-\Sigma b_j Z^{-J}} \qquad (1.11)$$

$$i = (0,N) \quad \text{and} \quad j = (1,M)$$

The frequency response of the filter can then be obtained by substituting $z=e^{j\omega T}$. The frequency response of a digital filter described by eqn. 1.10 would have an infinite impulse response and filters of this type are called recursive, or IIR, filters. If we, however, set $b_j=0$, then the filter would become finite impulse response, or FIR.

The design of digital filters involves approximation and synthesis methods which are used to find the values of the a_i and b_j coefficients for a given set of specifications. These specifications include passband edge frequency, stopband edge frequency, minimum loss in the stopband and maximum ripple in the passband. Fig. 1.4 illustrates typical lowpass specifications and a typical response which meets these specifications.

1.2.4- Digital filter realisations

Eqn. 1.10 suggest that there are three basic elements required to realise a digital filter. First, some form of storage is needed to store the input and output samples, secondly digital multipliers are needed to multiply a constant with a sampled signal and finally digital adders are needed to add two samples together. There are many digital filter configurations which can be designed to realise the difference equation 1.10. Each configuration has properties which may or may not be desirable depending on the particular applications. Here we briefly consider three form of realisations.

- **Direct and Canonical Forms of Realisation**

The simplest form of realisation of eqn. 1.10 would be to implement the difference equation directly. The output, y_n, is obtained by adding the present sample inputs, x_n, with the past sample inputs, x_{n-k}, and

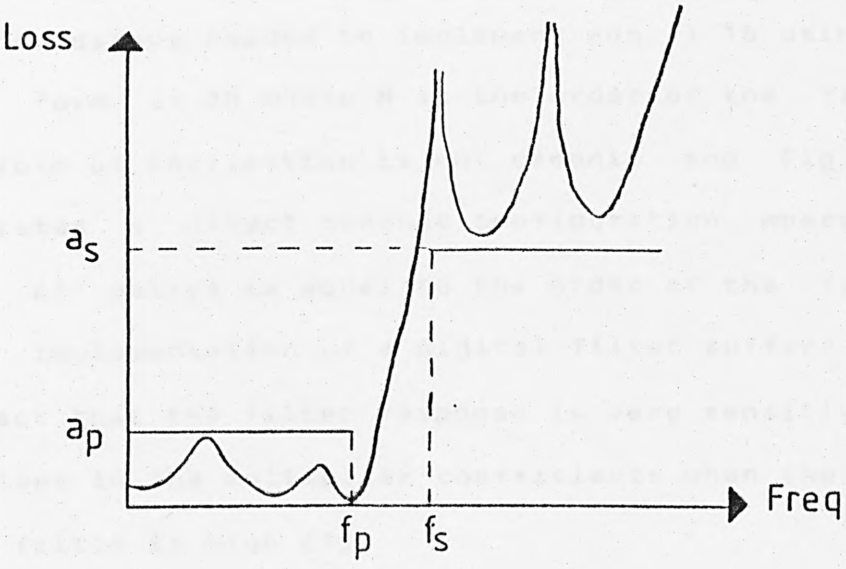


Fig. 1.4 _ Typical Lowpass specifications.

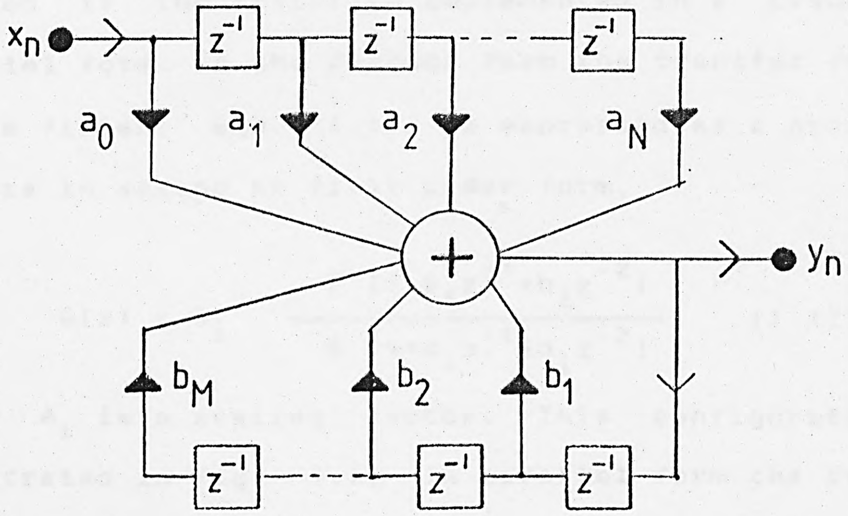


Fig. 1.5 _ Direct realisation of a Digital filter.

ouputs, y_{n-k}, where k=1,2,...,N assuming that M=N. This configuration is shown in Fig. 1.5. As can be seen, the number of delays needed to implement eqn. 1.10 using the direct form is 2N where N is the order of the filter. This form of realisation is not canonic and Fig. 1.6 illustrates a direct canonic configuration where the number of delays is equal to the order of the filter. Direct implementation of a digital filter suffers from the fact that the filter response is very sensitive to variations in the multiplier coefficients when the order of the filter is high [1].

- Cascade and Parallel forms of Realisation

The sensitivity in the direct form realisation can be reduced if the filter is implemented in a cascade or parallel form. In the cascade form the transfer function of the filter, eqn. 1.11, is expressed as a product of factors in second or first order form,

$$G(z) = A_i \quad \frac{\Pi\ (1+a_i z^{-1}+b_i z^{-2})}{\Pi\ (1+c_i z^{-1}+d_i z^{-2})} \qquad (1.12)$$

where A_i is a scaling factor. This configuration is illustrated in Fig. 1.7. In parallel form the transfer function is expressed as a sum of partial fractions, as shown below,

$$G(z) = A_0 + \Sigma\ [B_i \frac{a_i+b_i z^{-1}}{1+c_i z^{-1}+d_i z^{-2}}] \qquad (1.13)$$

This form of realisation is shown in Fig. 1.8. As

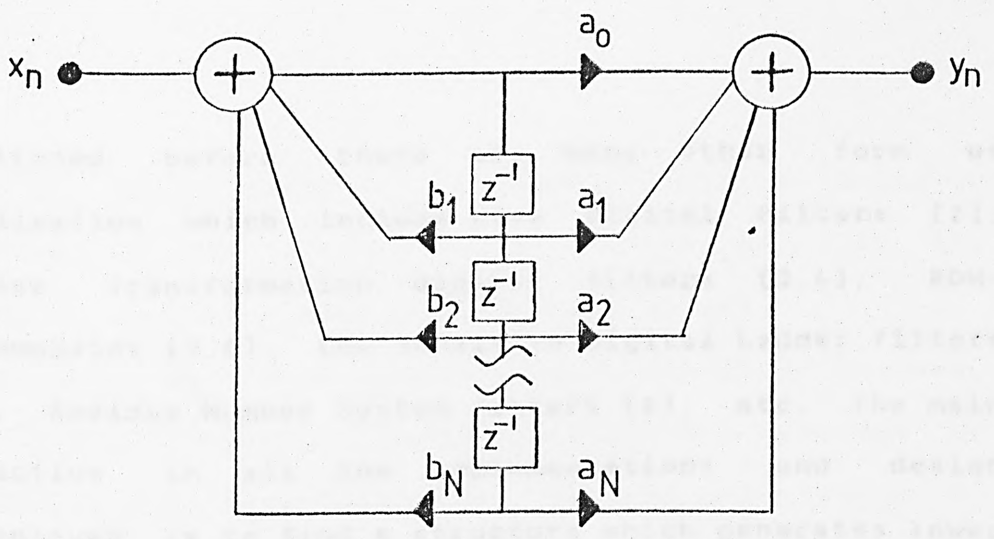


Fig. 1.6 _ Canonic direct form .

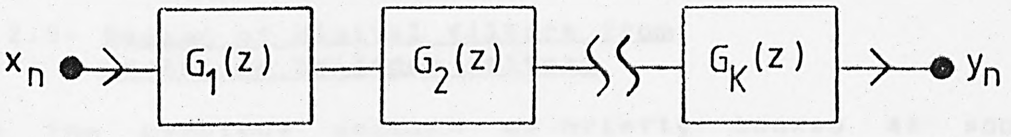


Fig. 1.7 _ Cascade form .

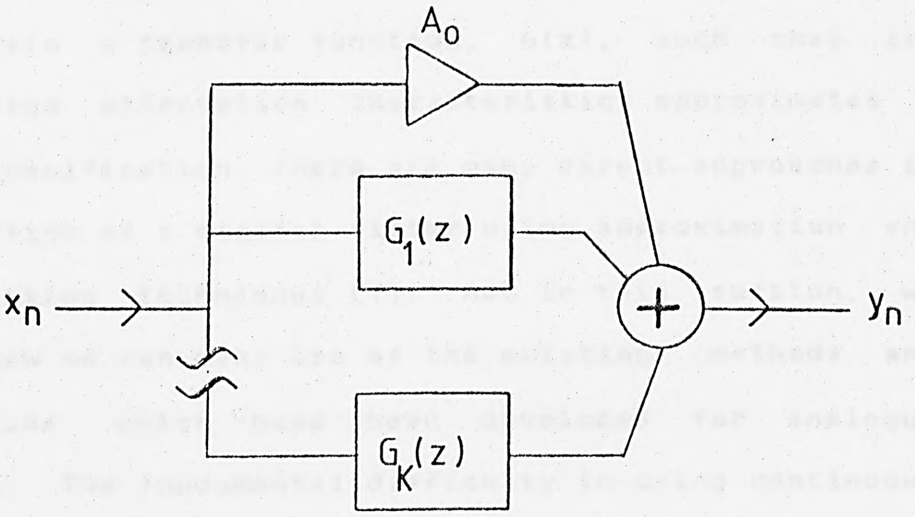


Fig. 1.8 _ Parallel form .

mentioned before there are many other form of realisation which include Wave Digital Filters [2], Linear Transformation digital filters [3,4], ROM-ACCumulator [5,6], Low sensitive Digital Ladder filters [7], Residue Number System filters [8], etc. The main objective in all the implementations and design techniques is to find a structure which generates lower noise than the other structures. In this thesis, we consider the WDF approach and the reasons for this choice will be discussed at relevant points throughout the remainder of this chapter.

1.2.5- Design of digital filters from prototype analogue filters

In the previous section we briefly looked at some possible realisation forms to implement a digital filter. Now we address ourselves to the question of how to obtain a transfer function, G(z), such that its associated attenuation characteristics approximates a given specification. There are many direct approaches in the design of a digital filter using approximation and optimization techniques [1]. But in this section, we show how we can make use of the existing methods and techniques which have been developed for analogue filters. The fundamental difficulty in using continuous methods in the discrete-time domain is the fact that a suitable transformation is required to transform the continuous transfer function, H(s), which is a rational function in s (complex variable), into a transfer function in the z-plane, G(z), which has all the desirable frequency domain properties of H(s).

There are a number of transformation which can be used, such as the standard z-transform (sometimes referred to as impulse invariant transform), matched z-transform or the bilinear transform. Here we only look at the bilinear transform which will be used later in this thesis. The reader is referred to [1] for more details on the bilinear transformation and the other transformations available.

- Bilinear Transformation

The bilinear transformation is a mapping from the s-plane into the z-plane and it is described as,

$$s = \frac{1- z^{-1}}{1 + z^{-1}} \qquad (1.14)$$

with its inverse as,

$$z = \frac{1 + s}{1 - s} \qquad (1.15)$$

This transformation maps the entire left-hand side of the s-plane into the inside of a unit circle in the z-plane and the right-hand side to the outside of the unit circle as illustrated in Fig. 1.9. Therefore any stable transfer function in the s-plane, H(s), can be mapped into a corresponding transfer function in the z-plane,

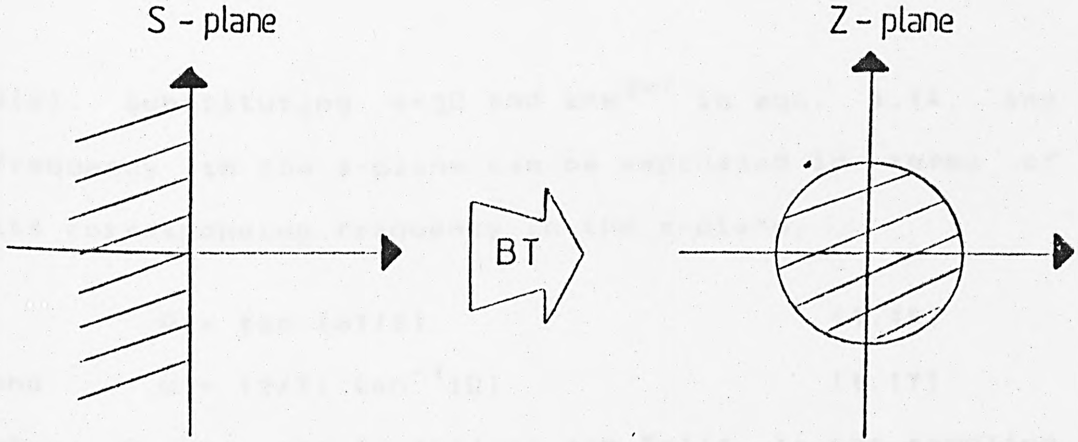


Fig. 1.9 _ Bilinear Transformation.

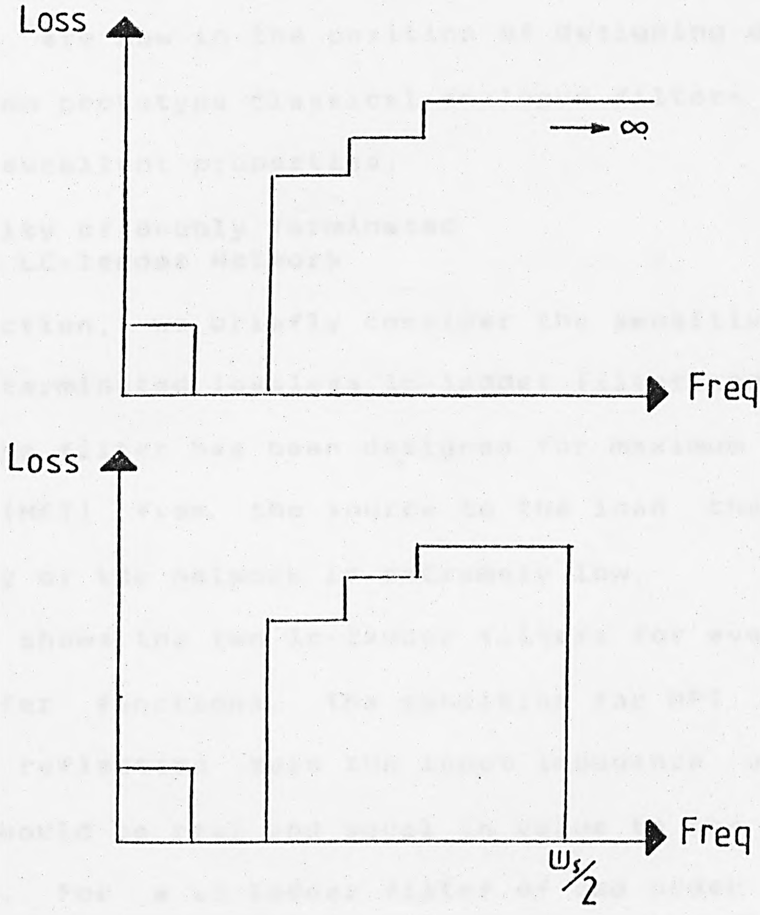


Fig. 1.10 _ Frequency warping effect.

G(z). Substituting $s=j\Omega$ and $z=e^{j\omega T}$ in eqn. 1.14, any frequency in the s-plane can be expressed in terms of its corresponding frequency in the z-plane,

$$\Omega = \tan(\omega T/2) \qquad (1.16)$$

and $$\omega = (2/T)\tan^{-1}(\Omega) \qquad (1.17)$$

where Ω and ω are in Rad/sec and $T=1/f$ is the sampling period. Note that the relationship in eqn. 1.16 is not a linear relationship. This fact results in a frequency "warping" near to the half sampling frequency (Fig. 1.10). We are now in the position of designing digital filters from prototype classical analogue filters which have many excellent properties.

- Sensitivity of Doubly Terminated Lossless LC-ladder Network

In this section, we briefly consider the sensitivity of a doubly terminated lossless lc-ladder filter and show that if the filter has been designed for maximum power transfer (MPT) from the source to the load then the sensitivity of the network is extremely low.

Fig. 1.11 shows the two lc-ladder filters for even and odd transfer functions. The condition for MPT states that at a reflection zero the input impedance of the network should be real and equal in value to the source resistance. For a lc-ladder filter of odd order (Fig. 1.11a) at $\Omega=0$ the input impedence of the network is equal to R1 (Fig. 1.12) and all we need to do is to make

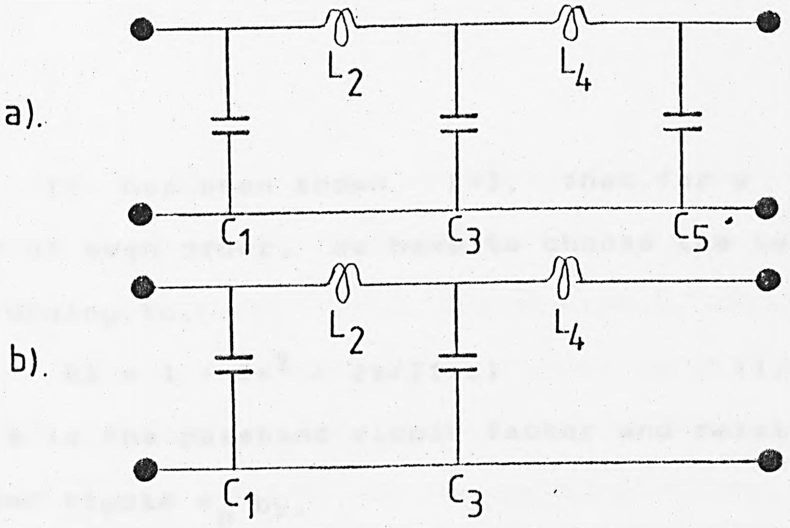


Fig. 1.11 _ LC _ Ladder of a). odd, and b). even order.

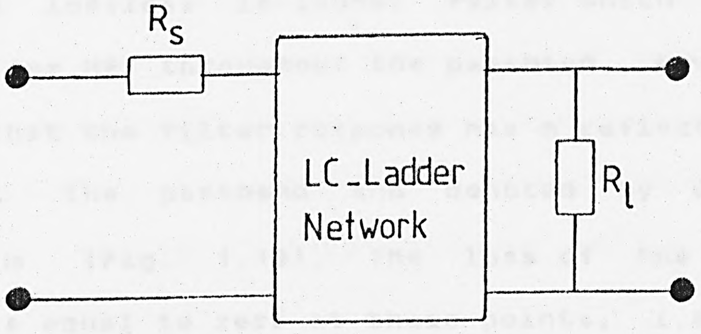


Fig. 1.12 _ Block representation of a doubly terminated network.

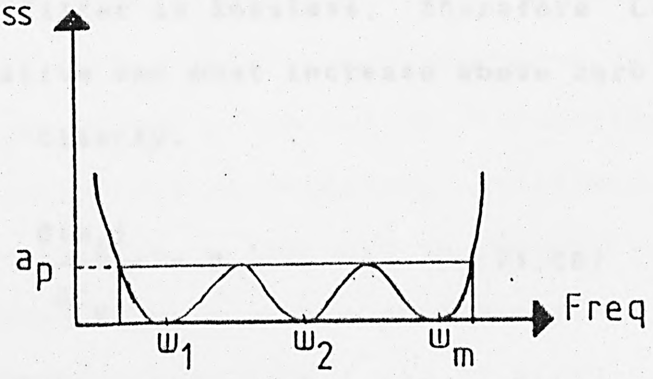


Fig. 1.13 _ Passband characteristics of a lossless filter.

Rs=Rl. It has been shown, [9], that for a lc-ladder filter of even order, we have to choose the termination Rl according to,

$$Rl = 1 + 2\varepsilon^2 + 2\varepsilon\surd(1+\varepsilon) \qquad (1.18)$$

where ε is the passband ripple factor and related to the passband ripple a_p by,

$$\varepsilon = \surd(10^{a_p/10} - 1) \qquad (1.19)$$

for maximum power transfer. Now consider a doubly terminated lossless lc-ladder filter which has been designed for MPT throughout the passband. Now let us consider that the filter response has m reflection zeros throughout the passband and denoted by ω_i where i=1,2,...,m (Fig. 1.13). The loss of the transfer function is equal to zero at these points, i.e $L(\omega_i)=0$, and elsewhere in the passband $L(\omega)<a_p$ DB. Now consider the effect on $L(\omega)$ when any element in the filter, i.e L_k or C_k, is decreased or increased from its ideal value. Since the filter is lossless, therefore $L(\omega_k)$ cannot become negative and must increase above zero as shown in Fig. 1.13. Clearly,

$$\frac{\partial(\omega_i)}{\partial L_k} = \frac{\partial(\omega_i)}{\partial C_k} = 0 \qquad (1.20)$$

for i=1,2,...,m. This argument has been stated very briefly here and the reader is referred to [10] for more details. Therefore from (1.20), it can be deduced that, as long as the number of reflection zeros in the passband is large and also the maimum ripple in the pass band is kept small, then the frequency response of a doubly terminated lc-ladder filter which has been designed for MPT is very insensitive to small variations in the values of its components. This is not particularly true in the stopband of such a filter, but usually in a filtering application the effect of variations in the stopband are far less important than variations in the passband. Apart from lc-ladder filters there are other types of analogue filters which exhibit low sensitivity characteristics to variations in the components values. These include unit-element and lattice filters.

In previous section, we illustrated how a continuous filter design can be transformed into a discrete filter design and preserve all the properties of the analogue filter using bilinear transformation. The transformation of unit-element filters, lattice and lc-ladder filters results in a digital filter which is also insensitive to variation in its multiplier coefficients. This is very important, as will be seen in the next section, when the filter is implemented using discrete components.

The design of the reference filters can be achieved in several ways. The first approach would be to use analogue filter design tables [11]. This approach is limited to some extent since tables cannot provide designs for every specification. The alternative would be to synthesize or use some form of optimization to derive the reference filters [12-15]. It is also possible to use explicit formulae to calculate the element values for a given specification [9]. The problem with the explicit formulae is that only Butterworth and Chebyshev filters can be designed and unfortunately no such formulae have yet been developed for the design of Elliptic filters.

1.2.6- Finite wordlength effects

Fig. 1.14 shows how a digital filter may be used in a practical filtering application. The output of the A/D convertor is a digital representation of the input samples $x_s(t)$. Also the coefficients are stored as m-bit binary numbers. Therefore inherent errors exist in this representation of the parameters and they give rise to three types of error sources,

1) Error due to the quantization of the filter coefficients.

2) Error due to the quantization of the input.

3) Error due to rounding or truncating the results of any arithmetic operations.

- Coefficient Quantization

When the filter is designed the coefficients are normally evaluated to a high degree of accuracy.

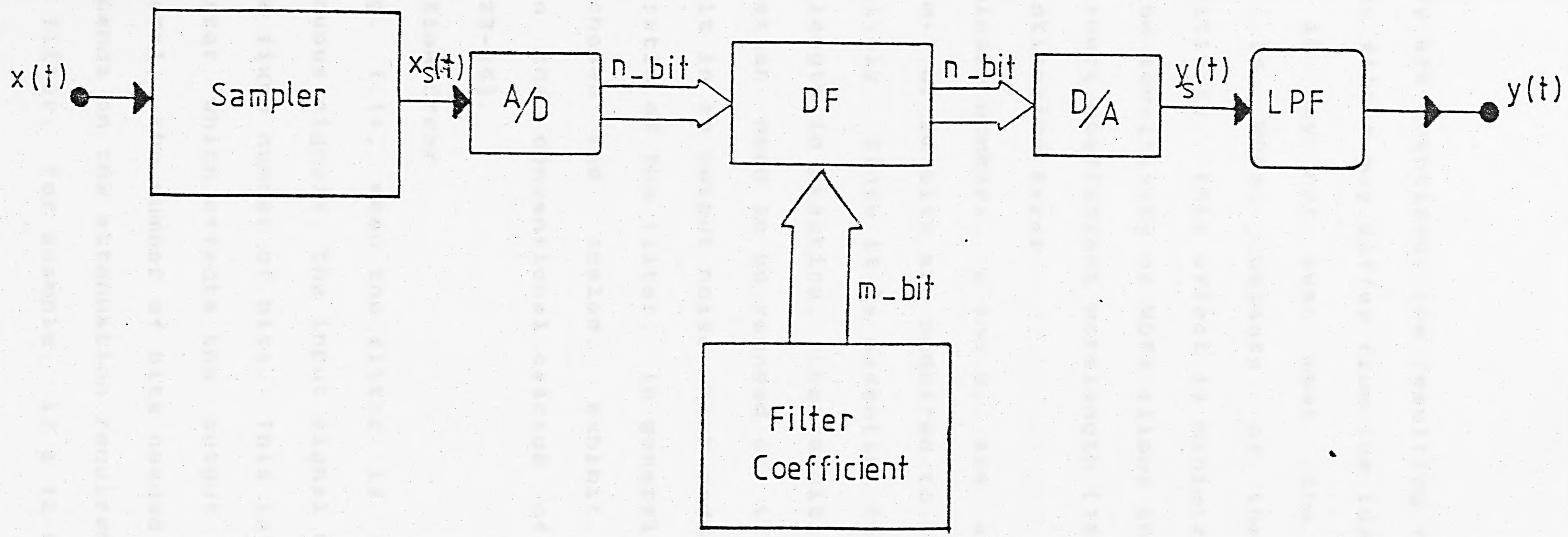


Fig. 1.14_ Practical use of a Digital filter.

However, if they are quantized, the resulting frequency response of the filter may differ from the ideal case and sometimes it may not even meet the design specifications. In WDFs, because of their low attenuation sensitivity, this effect is minimized [16-17]. Also the low sensitivity of WDFs allows the design of filters with short coefficient wordlength [18-22].

- Arithmetic Quantization Error

If two m-bit binary numbers, a and b, are added or multiplied then m+1 or 2m bits are required to store the results respectively. Since it is essential to use a fixed register length in practice, the results of any arithmetic operation need to be rounded or truncated. This will result in an output noise which reduces the signal-to-noise ratio of the filter. In general, WDFs, if carefully chosen and scaled, exhibit better performance than the conventional cascade of second order sections [23-25].

- Input Quantization Error

As shown in Fig. 1.14, when the filter is used to operate on continuous signals, the input signal needs to be quantized to a fixed number of bits. This introduces a non-linear error which effects the output of the filter. In general, the number of bits needed for the A/D convertor depends on the attenuation required in the stopband of the filter. For example, if a 12-bit A/D convertor is used the maximum attenuation, or the dynamic range, which can be obtained is equal to $20\log 2^{12}$, i.e 72 DB. Therefore for a given specification we can choose a suitable A/D convertor.

1.2.7- Summary

In the previous sections, we looked at the theory of digital filters and sampled data processing. We stated the problems associated with finite wordlength effects which are non-existence for analogue filters. We also introduced very briefly the concept of WDFs and highlighted some of their main advantages. In the following sections, the design and implementation of WDFs are considered in more detail.

1.3.0- Wave Digital Filters

1.3.1- Introduction

Since the early development of digital filters, there have been many different approaches to the design and implementation of digital filters. During the initial developments the transfer function was implemented directly, but this approach resulted in structures which were highly sensitive. The next method was to split the transfer function into lower order term and connect them in cascade or parallel forms. The resulting structures were much less sensitive but still they suffered from finite wordlength effects when implemented in hardware. An excellent contribution towards the design of a suitable structure for the implementation of digital filters was due to Fettweis in 1971 [2]. He referred to these type of digital filters as Wave Digital Filters (WDFs).

WDFs represent a class of digital filters which are based on classical analogue filter networks. Thus, several of the good properties of the reference filter are preserved after the transformation. One of the main advantages of WDFs is the direct consequence of the excellent low sensitivity of doubly terminated lossless reference filters. This reduces the coefficient accuracy required for the multipliers in the WDFs. The other finite wordlength effects, such as limit cycle, parasitic oscillation, stability, etc, can also be minimized or eliminated if the WDF is designed properly [26].

WDFs are derived from conventional analogue filters using the bilinear transform. The analogue filter is described using the voltage/current relationship and two variables Ak and Bk as follows,

$$\begin{aligned} Ak &= Vk + Rk.Ik \\ Bk &= Vk - Rk.Ik \end{aligned} \qquad (1.21)$$

where Ak and Bk are called the incident and reflected waves, due to their relationship to scattering matrix theory, and this is the main reason to refer to the resulting filters as Wave Digital Filters. There are two main approaches to the design of WDFs. In the first approach, the elements of the reference filter are treated as one-port network and are connected to other elements using adaptors [2]. In the second approach, due to Lawson [27], the elements in the reference filter are treated as a two-port network, therefore they can be connected to the other elements directly and without the use of adaptors.

In this section, we briefly consider the design procedures for both techniques. The reader is recommended to consult Ref [2 & 27] for further details. Recently an excellent review of WDFs has been published by Fettweis which covers in detail all the aspects in the first approach [28].

1.3.2- Theory and Design of WDFs (I)

- Elements and Sources Realisation

The first step in deriving a WDF from a reference filter is to find the translation of the elements in the reference filter into the digital-domain. This is achieved by applying the bilinear transformation to the wave relationships of all the elements which may exist in the reference filter. Here we only consider the realisation of a capacitor, an inductor, a resistor and a resistive voltage source.

- Inductor

The steady-state voltage/current relationship for an inductor is,

$$V = RIs \tag{1.22}$$

Using eqn. 1.21 and 1.14, we obtain,

$$B = - Az^{-1} \tag{1.23}$$

- Capacitor

The steady-state relationship of a capacitor is,

$$V = RI/s \tag{1.24}$$

and in using eqn. 1.21 and 1.14, we obtain,

$$B = Az^{-1} \tag{1.25}$$

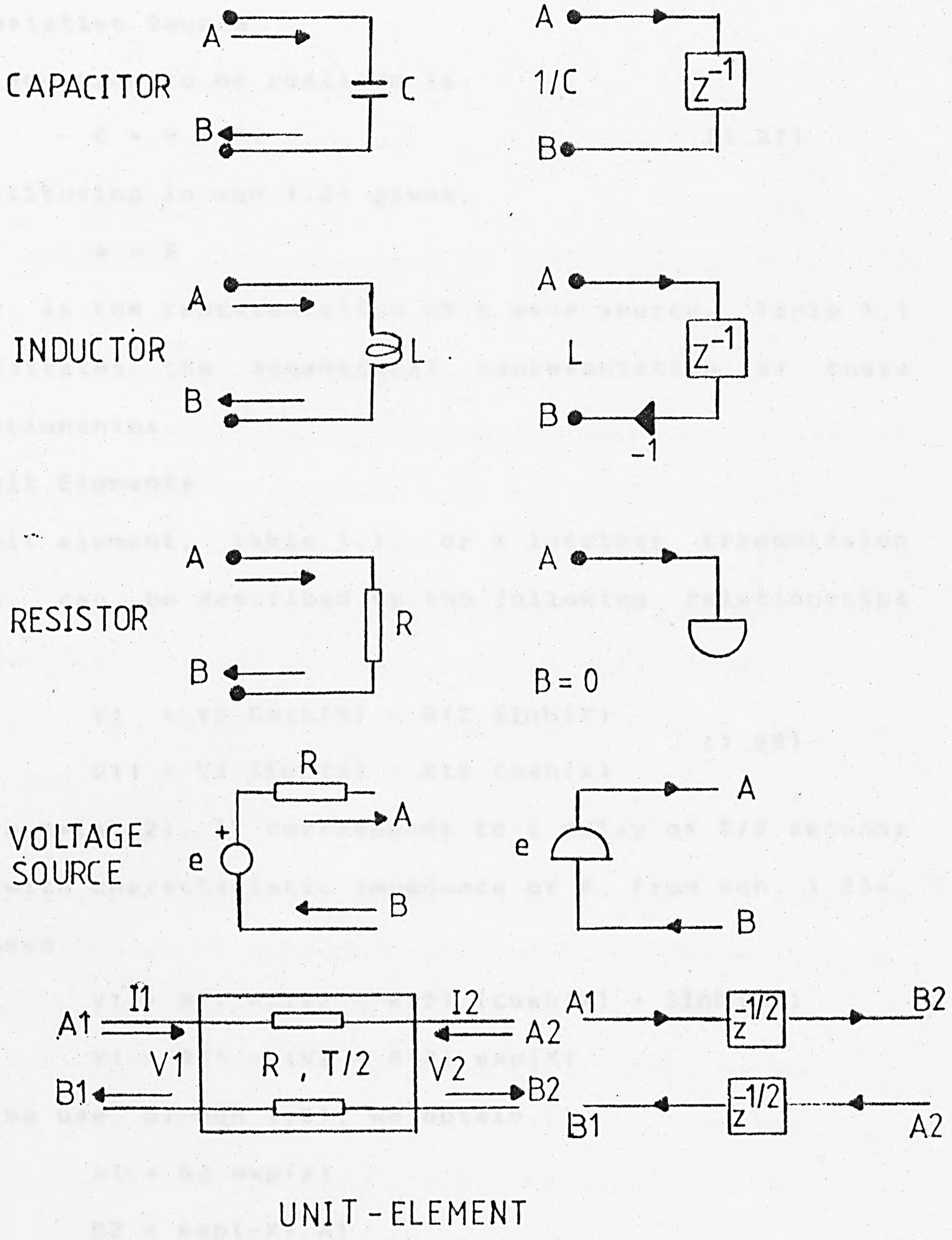
- Resistor

The equation to realise is,

$$V = RI \tag{1.26}$$

and application of eqn. 1.21 gives,

TABLE 1.1


CAPACITOR
A
B
C
A
1/C
Z^{-1}
B
INDUCTOR
A
B
L
A
L
Z^{-1}
B
-1
RESISTOR
A
B
R
A
B = 0
VOLTAGE SOURCE
R
A
+
e
B
A
e
B
I1
I2
A1
A2
V1
R , T/2
V2
B1
B2
A1
$Z^{-1/2}$
B2
B1
$Z^{-1/2}$
A2
UNIT-ELEMENT

$$B = 0$$

This represents a wave sink.

- **Resistive Source**

The equation to be realised is,

$$E = V + RI \qquad (1.27)$$

Substituting in eqn 1.21 gives,

$$A = E$$

This is the representation of a wave source. Table 1.1 illustrates the schematical representation of these relationships.

- **Unit Elements**

A unit element, table 1.1, or a lossless transmission line, can be described by the following relationships [29],

$$\begin{aligned} V1 &= V2\ \mathrm{Cosh}(X) - RI2\ \mathrm{Sinh}(X) \\ RI1 &= V2\ \mathrm{Sinh}(X) - RI2\ \mathrm{Cosh}(X) \end{aligned} \qquad (1.28)$$

where X=(sT/2). It corresponds to a delay of T/2 seconds and with characteristic impedence of R. From eqn. 1.28a, we have,

$$V1 + RI1 = (V2 - RI2)\ (\mathrm{Cosh}(X) + \mathrm{Sinh}(X))$$

or $$V1 + RI1 = (V2 - RI2)\ \exp(X)$$

By the use of eqn 1.21, we obtain,

$$A1 = B2\ \exp(X)$$

or $$\begin{aligned} B2 &= \exp(-X)\ A1 \\ &= \exp(-sT/2)\ A1 \end{aligned}$$

and therefore,

$$B2 = z^{-1/2} A1 \qquad (1.29a)$$

Similary,

$$B1 = \exp(-X)\ A2$$

$$= \exp(-sT/2)\ A2$$

and $$B1 = z^{-(1/2)} A2 \qquad (1.29b)$$

The schematic representation of 1.29 is also given in table 1.1.

- Interconnection and Adaptors

Now that the necessary elements from the reference filter have been translated into the digital-domain, we need to consider the realisation of the interconnections, so called adaptors. There are three main types of adaptors which must be considered, the 2-port, the 3-port serial and the 3-port parallel adaptors.

- 2-port Adaptor

This is the simplest form of adaptor and is used to connect two ports with different port resistances. Fig. 1.15a illustrates the interconnection of two ports with port resistances R1 and R2. From Fig. 1.15a, we have,

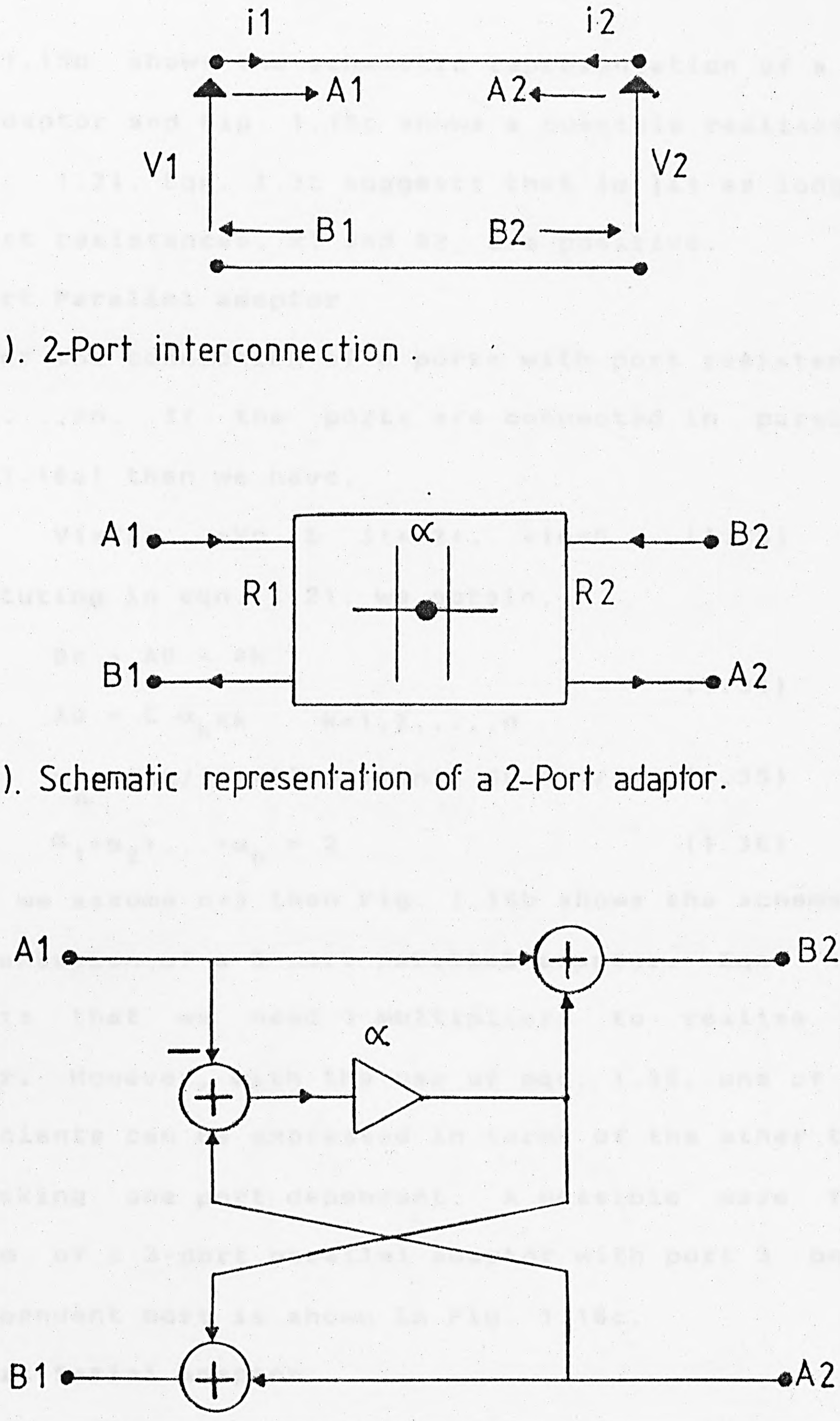
$$V1 = V2\ \&\ I1 = -\ I2 \qquad (1.30)$$

Substituting 1.30 into eqn. 1.21, we obtain,

$$\begin{aligned} B1 &= A2 + \alpha(A2 - A1) \\ B2 &= A1 + \alpha(A2 - A1) \end{aligned} \qquad (1.31)$$

where $$\alpha = (R1 - R2)/(R1 + R2) \qquad (1.32)$$

Fig.1.15



a). 2-Port interconnection.

b). Schematic representation of a 2-Port adaptor.

c). One possible realisation of a 2-Port adaptor.

Fig. 1.15b shows the schematic representation of a 2-port adaptor and Fig. 1.15c shows a possible realisation of eqn. 1.31. Eqn. 1.32 suggests that $|\alpha\ | \leqslant 1$ as long as the port resistances, R1 and R2, are positive.

- 3-port Parallel adaptor

Consider the connection of n ports with port resistances R1,R2,...,Rn. If the ports are connected in parallel (Fig. 1.16a) then we have,

$$V1=V2=...=Vn \quad \& \quad I1+I2+...+In=0 \qquad (1.33)$$

Substituting in eqn. 1.21, we obtain,

$$\begin{aligned} &Bk = A0 - Ak \\ &A0 = \Sigma\ \alpha_k Ak \qquad k=1,2,...,n \end{aligned} \qquad (1.34)$$

where $\alpha_k = 2Gk/(G1+G2+...+Gn), \ Gk = 1/Rk \qquad (1.35)$

and $\alpha_1+\alpha_2+...+\alpha_n = 2 \qquad (1.36)$

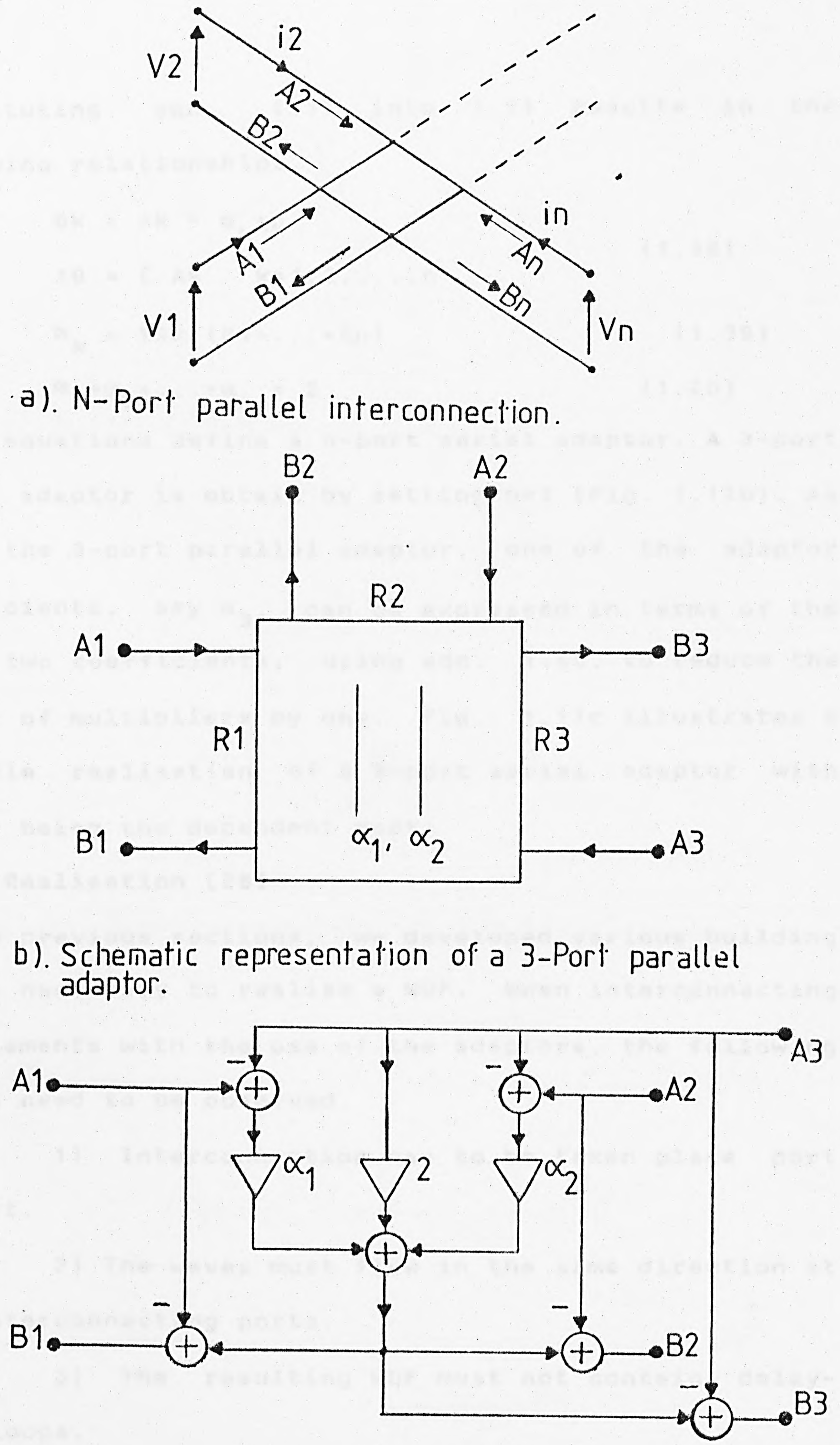
Now if we assume n=3 then Fig. 1.16b shows the schematic representation of a 3-port parallel adaptor. Eqn. 1.34 suggests that we need 3 multipliers to realise the adaptor. However, with the use of eqn. 1.36, one of the coefficients can be expressed in terms of the other two, i.e making one port dependent. A possible wave flow diagram of a 3-port parallel adaptor with port 3 being the dependent port is shown in Fig. 1.16c.

- 3-port Serial Adaptor

If the n ports in previous section are connected in series (Fig. 1.17a) then we have,

$$V1+V2+...+Vn=0 \quad \& \quad I1=I2=...=In \qquad (1.37)$$

Fig.1.16



a). N-Port parallel interconnection.

b). Schematic representation of a 3-Port parallel adaptor.

c). One possible realisation of a 3-Port parallel adaptor. (port 3 is the dependent port).

Substituting eqn. 1.37 into 1.21 results in the following relationships,

$$Bk = Ak - \alpha_k A0$$
$$A0 = \Sigma\ Ak \quad k=1,2,...,n \qquad (1.38)$$

where $\alpha_k = 2Rk/(R1+...+Rn)$ (1.39)

and $\alpha_1+\alpha_2+...+\alpha_n = 2$ (1.40)

These equations define a n-port serial adaptor. A 3-port serial adaptor is obtain by setting n=3 (Fig. 1.17b). As with the 3-port parallel adaptor, one of the adaptor coefficients, say α_3, can be expressed in terms of the other two coefficients, using eqn. 1.40, to reduce the number of multipliers by one. Fig. 1.17c illustrates a possible realisation of a 3-port serial adaptor with port 3 being the dependent port.

- WDF Realisation [28]

In the previous sections, we developed various building blocks necessary to realise a WDF. When interconnecting the elements with the use of the adaptors, the following points need to be observed,

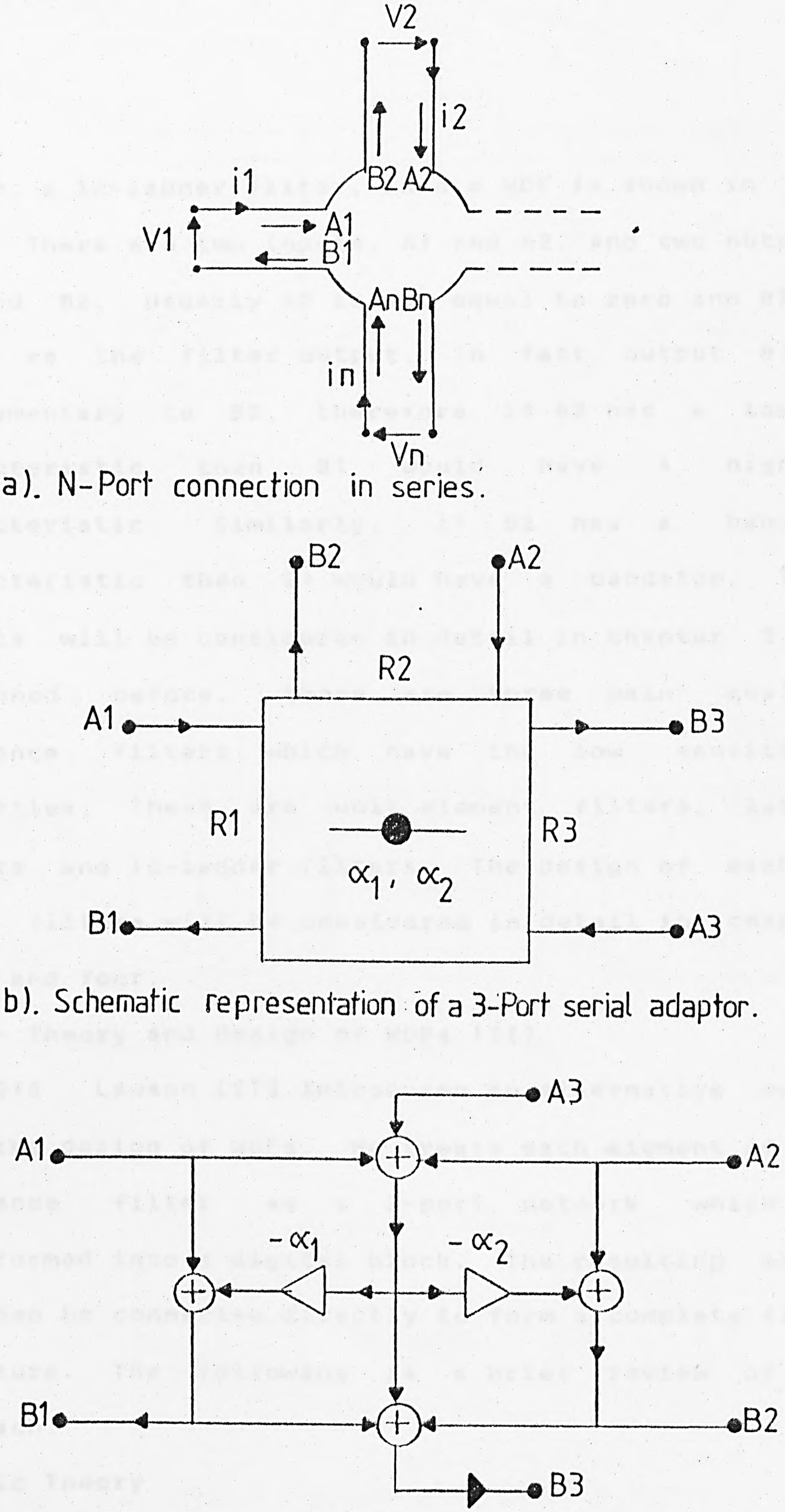
1) Interconnection has to be taken place port by port.

2) The waves must flow in the same direction at the interconnecting ports.

3) The resulting WDF must not contain delay-free loops.

An example of the transformation of a typical reference

Fig.1.17



a). N-Port connection in series.

b). Schematic representation of a 3-Port serial adaptor.

c). One possible realisation of a 3-Port serial adaptor.
(port 3 is the dependent port).

filter, a lc-ladder filter, into a WDF is shown in Fig. 1.18. There are two inputs, A1 and A2, and two outputs, B1 and B2. Usually A2 is set equal to zero and B2 is taken as the filter output. In fact, output B1 is complementary to B2, therefore if B2 has a lowpass characteristic then B1 would have a highpass characteristic. Similarly, if B2 has a bandpass characteristic then B1 would have a bandstop. These effects will be considered in detail in chapter 5. As mentioned before, there are three main analogue reference filters which have the low sensitivity properties. These are unit element filters, lattice filters and lc-ladder filters. The design of each of these filters will be considered in detail in chapters three and four.

1.3.3- Theory and Design of WDFs (II)

In 1975 Lawson [27] introduced an alternative method for the design of WDFs. He treats each element of the reference filter as a 2-port network which is transformed into a digital block. The resulting blocks can then be connected directly to form a complete filter structure. The following is a brief review of his approach.

- Basic Theory

Given a passive 2-port network (Fig. 1.19), we can describe the network using the ABCD matrix as follows,

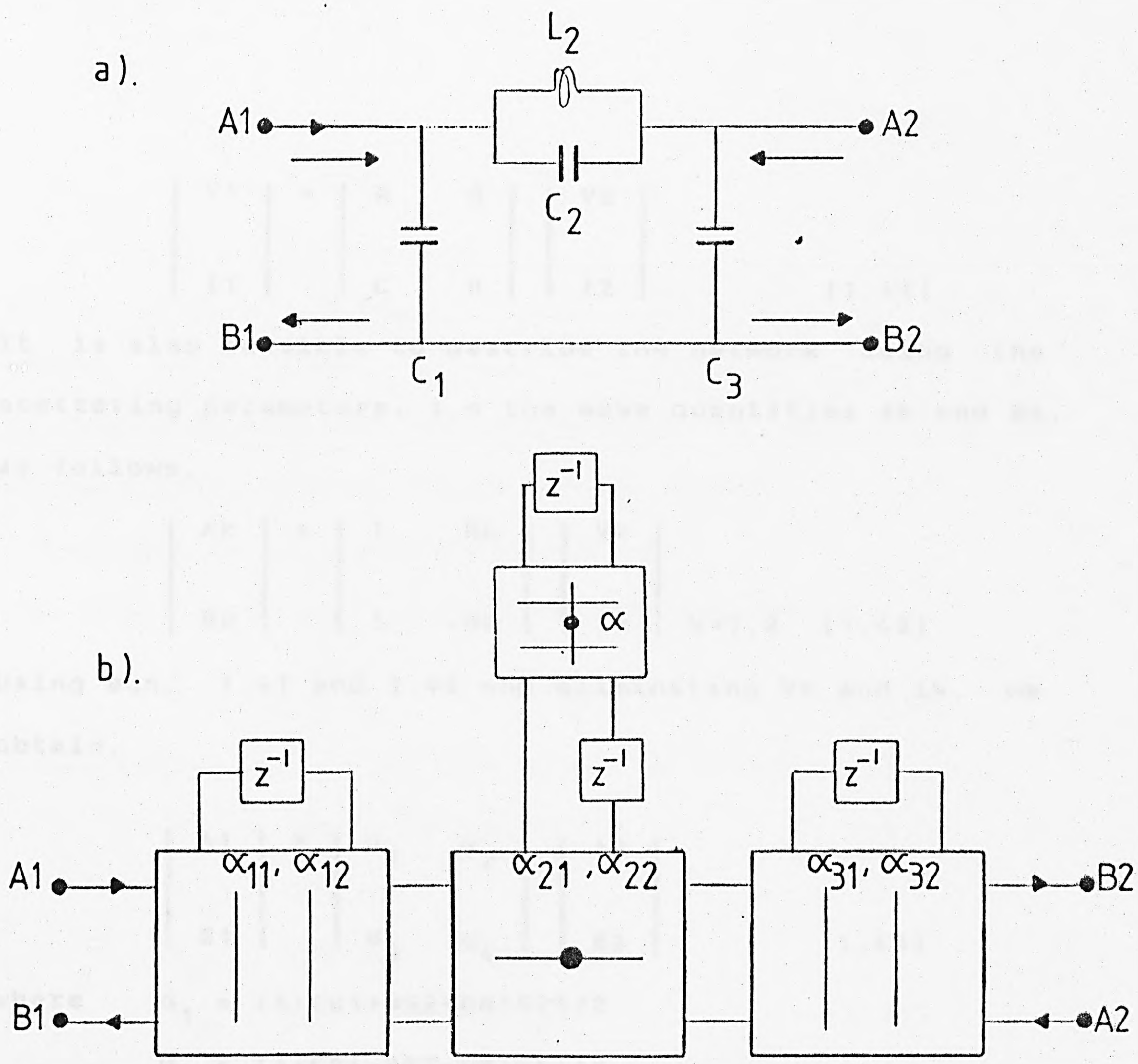

a).
L2
A1
A2
C2
B1
B2
C1
C3
b).
z^{-1}
α
z^{-1}
z^{-1}
z^{-1}
A1
α_{11}, α_{12}
α_{21}, α_{22}
α_{31}, α_{32}
B2
B1
A2

Fig.1.18 _ a) A typical LC-ladder filter.
b) Corresponding WDF derived from a).

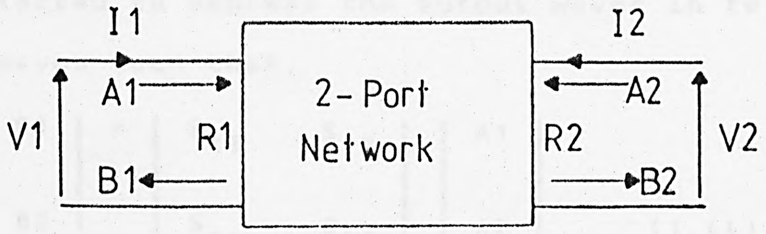


Fig.1.19 _ General representation of a 2-Port Network.

$$\begin{vmatrix} V1 \\ I1 \end{vmatrix} = \begin{vmatrix} A & B \\ C & D \end{vmatrix} \begin{vmatrix} V2 \\ I2 \end{vmatrix} \qquad (1.41)$$

It is also possible to describe the network using the scattering parameters, i.e the wave quantities Ak and Bk, as follows,

$$\begin{vmatrix} Ak \\ Bk \end{vmatrix} = \begin{vmatrix} 1 & Rk \\ 1 & -Rk \end{vmatrix} \begin{vmatrix} Vk \\ Ik \end{vmatrix} \quad k=1,2 \qquad (1.42)$$

Using eqn. 1.41 and 1.42 and eliminating Vk and Ik, we obtain,

$$\begin{vmatrix} A1 \\ B1 \end{vmatrix} = \begin{vmatrix} \alpha_1 & \alpha_2 \\ \alpha_3 & \alpha_4 \end{vmatrix} \begin{vmatrix} A2 \\ B2 \end{vmatrix} \qquad (1.43)$$

where $\alpha_1 = (A+CR1+BG2+DR1G2)/2$

$\alpha_2 = (A+CR1-BG2-DR1G2)/2$

$\alpha_3 = (A-CR1+BG2-DR1G2)/2$

$\alpha_4 = (A-CR1-BG2+DR1G2)/2$

and $G2 = 1/R2$

It is preferred to express the output waves in terms of the input waves such that,

$$\begin{vmatrix} B1 \\ B2 \end{vmatrix} = \begin{vmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{vmatrix} \begin{vmatrix} A1 \\ A2 \end{vmatrix} \qquad (1.44)$$

By transforming eqn. 1.43 in the same form as eqn. 1.44, we obtain,

$S_{11} = \alpha_1/\alpha_2$

$S_{12} = - \Delta/\alpha_2$

$S_{21} = 1/\alpha_2$

and $S_{22} = - \alpha_1/\alpha_2$

where $\Delta = - R1G2$

By applying bilinear transformation to eqn. 1.44, a corresponding digital block can be formed.

- Realisation of a series Impedence

The ABCD matrix of a series impedence Z is,

$$\begin{vmatrix} 1 & -Z \\ 0 & -1 \end{vmatrix} \quad (1.45)$$

Substituting eqn. 1.45 into 1.44, we obtain,

$S_{11} = (R2 - R1 + Z)/\alpha_2$

$S_{12} = 2R1/\alpha_2$

$S_{21} = 2R2/\alpha_2$

and $S_{22} = (R1 - R2 + Z)/\alpha_2$ (1.46)

where $\alpha_2 = R2 + R1 + Z$

It is important to note that,

$S_{11} + S_{12} = 1$

and $S_{21} + S_{22} = 1$ (1.47)

Now substituting 1.47 into 1.44, we obtain,

$B1 = S_{11}(A1 - A2) + A2$

and $B2 = S_{22}(A2 - A1) + A1$ (1.48)

This means that we only need to realise S_{11} and S_{22} to define the series impedence Z.

- Series Capacitor

For a capacitor in the series-arm, we have Z=1/Cs and by applying the bilinear transformation, we obtain,

$$Z = \frac{1 + z^{-1}}{C(1 - z^{-1})}$$

Substituting the value of Z into eqn. 1.46, we obtain,

$$S_{11} = (\alpha_1 + \alpha_3 z^{-1})/(1 + \alpha_2 z^{-1})$$

and $\quad S_{22} = (\alpha_3 - \alpha_1 z^{-1})/(1 + \alpha_2 z^{-1})$

where $\quad \alpha_1 = (R2 - R1 + 1/C)/\beta$

$$\alpha_2 = (1/C - R2 - R1)/\beta$$

$$\alpha_3 = (1/C - R2 + R1)/\beta$$

and $\quad \beta = R2 + R1 + 1/C\ ,\ \alpha_1 + \alpha_3 = 1 + \alpha_2$

To obtain a realisable digital filter, i.e a filter with no delay-free loops, one possibility is to set $\alpha_1 = 0$. Thus we have,

$$S_{11} = \alpha_3 z^{-1}/(1 - \alpha_2 z^{-1})$$

$$S_{22} = \alpha_3/(1 + \alpha_2 z^{-1})$$

also $\quad \alpha_2 = \alpha_3 - 1 = - R2/R1$

By substituting these into (1.48), we obtain the digital structure for a capacitor in the series-arm (Fig. 1.20). This technique must be applied to every possible series and shunt elements which can be found in the reference filter. Fig. 1.21 shows a typical reference filter and its corresponding WDF structure.

This approach to the design of WDFs was later generalized by Lawson [3]. In Ref [3], he studies a general 2-port transformation on the classical doubly-terminated lossless ladder network. He derives a number of conditions which must be satisfied in order to result

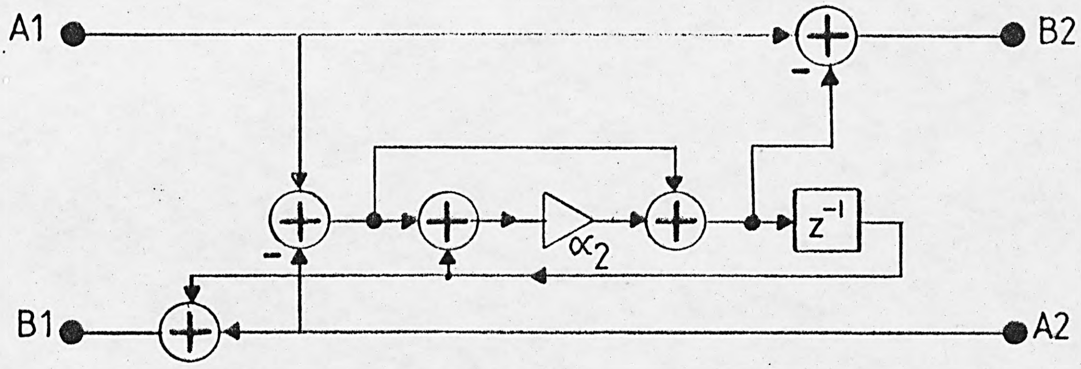
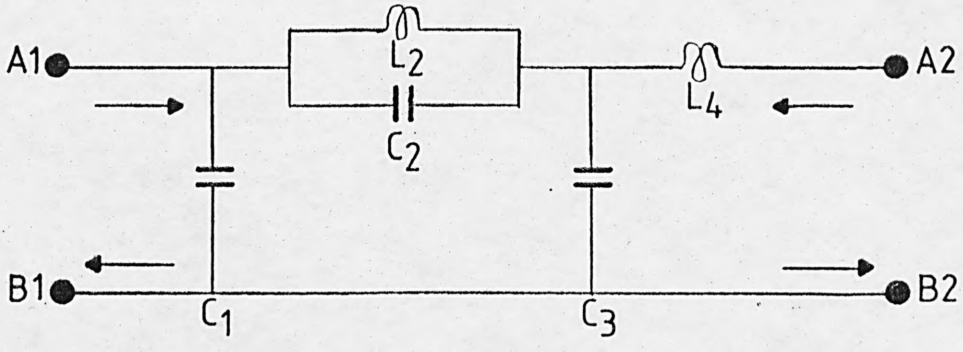
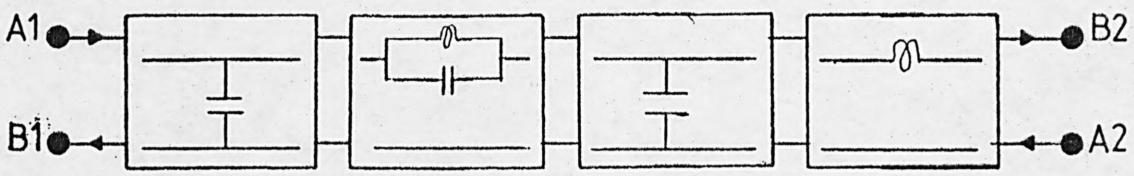


Fig. 1.20 _ Digital equivalant of a series capacitor.



(a)



(b)

Fig. 1.21 _ a) A typical analogue filter.
b) Corresponding WDF derived from (a).

a WDF which is realisable, i.e it contains no delay-free loops.

1.3.4- Comments on WDF design approaches

In the previous section we briefly looked at the two possible approaches in which a doubly-terminated analogue filter is transformed into a WDF. In this thesis, we consider the approach introduced by Fettweis for the following reason. In the Fettweis approach, there are mainly three elements which have to be implemented digitally. These are the 2-port, 3-port serial and 3-port parallel adaptors. The other elements have a very simple digital structure such as a simple delay or etc.

In the second approach, we need to consider all the possible elements, such as series and parallel capacitor, inductor, tuned circuits, etc. Therefore there are quite a number of different digital structures which must be implemented (Fig. 1.21). In the practical implementation of WDF, it is very important to have regular and modular structures. This will be better appreciated when the VLSI implementation of WDFs is considered.

1.3.5- Review of Hardware Implementation of Wave Digital Filters

Since WDFs were introduced by Fettweis, considerable attention has been given to the design of filters from low sensitive analogue filters. The low sensitivity properties of WDFs to variations in the value of the coefficients allows the use of smaller wordlength for multipliers and also reduces the quantization error due to round-off or truncation of the coefficients. The main drawback however is the hardware complexity of WDF building blocks. In spite of this, there has been a great deal of work and research on the hardware implementation of WDFs [30,79]. The first known implementation of a WDF was given in 1974 [31]. The filter was a third order and the hardware was the size of a mini-computer. The other approaches in the hardware implementation of WDFs include, a 14th order lc-ladder bandpass WDF [32], microprocessor based implementation of a 7th order lc all-pole ladder filter [33], use of distributed arithmetic [5] in implementation of WDFs [34-36], general purpose DSP based WDFs [37-42], single board WDFs based on cascaded unit element filters [38-40], etc. These approaches are all based on the use of MSI/LSI discrete components.

An approach to resolve the problem of hardware complexity of WDFs would be to consider the VLSI implementation of WDFs. In general, the hardware implementation of WDFs depends on how efficiently 2 and 3-port adaptors can be implemented. The work carried out at Edinburgh by Dr. Mavor's group concerns the design of a universal WDF adaptor which allows the realisation of either a parallel or serial 3-port adaptor. They only concentrate on one type of WDFs that is based on lc-ladder networks with inserted unit element [41-43]. Using FIRST [44], a silicon compiler, they have produced a single chip universal adaptor which may be cascaded to required filter order. A typical sampling rate for a fifth order filter would be around 50 KHZ.

Research is also being carried out at the university of Louvain, Belgium on the development of a complete CAD design tools for the VLSI design and implementation of digital filters [45-47]. A third order WDF based on lattice reference filters has been integrated on a single chip which has a sampling rate of up to 100 KHZ with a internal wordlength of 16-bits.

In all the VLSI approaches, the existing WDF structures are translated onto silicon. This however does not exploit the maximum potential of VLSI technology. To achieve this, the existing structures must be first transformed into new structures with certain properties which makes them suitable for VLSI implementation. These features are dealt with in the next section where we introduce the concept of VLSI array processing.

1.4.0- VLSI Array Processing

1.4.1- Introduction

As the scale of single chip integration increases, it is becoming increasingly apparent that potential problems of designing chips containing hundreds of thousands of transistors can only be overcome if some form of structured approach to integrated circuit design is adopted. The translation of the existing structures and printed circuit boards into VLSI circuit is inappropriate. This is mainly due to the fact that in VLSI technology the old concept that wires are cheap and components are expensive is not valid. Wires are now expensive not only in terms of the chip area occupied, but also in terms of the time delay. In general, in VLSI technology computation can be considered very cheap while communication is expensive and new algorithms and architectures should be designed accordingly.

To exploit the full potential of VLSI technology, it is important that the new VLSI algorithms result in architectures which take into account (a) the layout constraint in terms of interconnection and communication and (b) the overall cost in terms of silicon area, time delay and pin count. Therefore a suitable architecture for VLSI implementation must have the following properties,

1) **Regularity**

In VLSI, it is important to reformulate the existing algorithm so that the computations can be split up into many simple and identical tasks or sometimes referred to as cell or processing elements (PEs).

2) **Local Communication**

As mentioned before, it is preferable to avoid any long distance communications or any global communications and minimize the communication to nearest neighbouring cells.

3) **Modularity**

Modularity is the same as regularity but on a larger scale. This means that a complex structure may be constructed from a number of smaller regular structures.

4) **Pipelining**

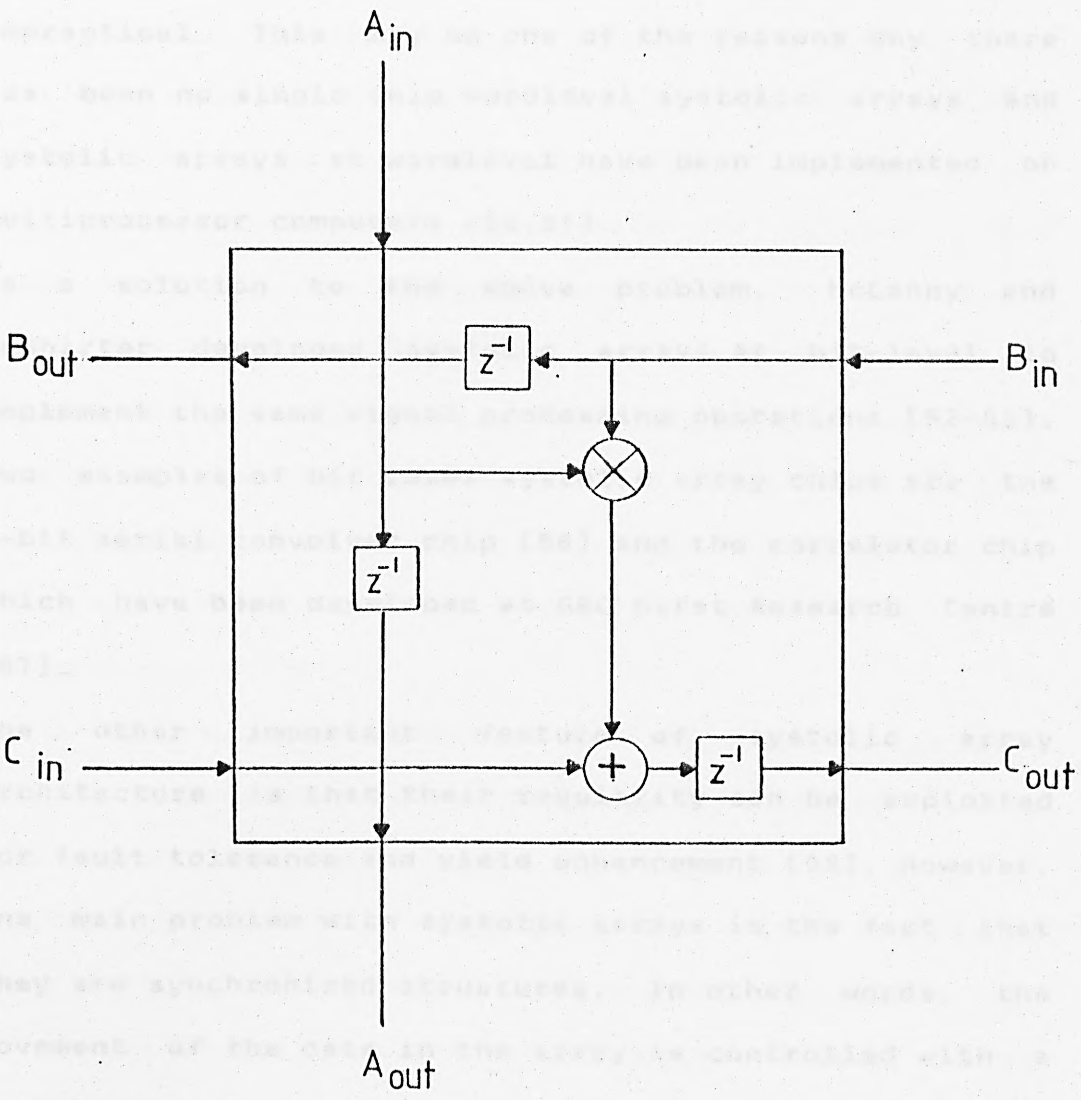
Pipelining simply means being able to process new data before the old ones have been processed completely. In other words, it means that the structure is designed in such a way that the processor elements are processing data at a maximum rate and there is no PE not active at a given time. In real time signal processing this becomes very important and pipelining at all levels should be pursued, i.e at structural and cell levels.

In this section, we describe a number of array architectures which have the potential of being used for VLSI implementation.

1.4.2- Systolic Arrays [48]

One solution for the requirements needed by VLSI technology is the concept of systolic arrays. Systolic arrays are a new class of pipelined array architectures that consist of a set of interconnected cells, which are all identical and capable of performing some simple operations. According to Kung and Leiserson [49],"A systolic system is a network of processors which rhythmically compute and pass data through the system. Physiologists use the word 'Systole' to refer to the rhythmically recurrent contraction of heart and arteries which pulses blood through the body. In a systolic computing system, the function of a processor is analogous to that of the heart. Every processor regularly pumps data in and out, each time performing some short computation, so that a regular flow of data is kept up in the network."

Kung and Leiserson have demonstrated how a number of simple PEs (Fig. 1.22), referred to as 'inner product' cells, can be locally connected to perform the pipeline computation of several important matrix and signal processing operations, such as matrix-matrix multiplication, matrix-vector multiplication, FIR filtering, convolution, etc. The systolic arrays by Kung and Leiserson use wordlevel operations in the cells. This however has the problem of I/O bandwidth when a



$$A_{out}(n) = A_{in}(n-1)$$

$$B_{out}(n) = B_{in}(n-1)$$

$$C_{out}(n) = C_{in}(n-1) + A_{in}(n-1)B_{in}(n-1)$$

Fig.1.22_Inner-Product cell.

number of wordlevel cells are implemented on a single chip and word parallel communication becomes impractical. This may be one of the reasons why there has been no single chip wordlevel systolic arrays and systolic arrays at wordlevel have been implemented on multiprocessor computers [50,51].

As a solution to the above problem, McCanny and McWhirter developed systolic arrays at bit-level to implement the same signal processing operations [52-55]. Two examples of bit level systolic array chips are the 8-bit serial convolver chip [56] and the correlator chip which have been developed at GEC Hirst Research Centre [57].

One other important feature of systolic array architecture is that their regularity can be exploited for fault tolerance and yield enhancement [58]. However, one main problem with systolic arrays is the fact that they are synchronized structures. In other words, the movement of the data in the array is controlled with a global timing reference or clock. This may become intolerable for very-large-scale arrays.

1.4.3- Wavefront Arrays [59]

A solution to the above problem is to localize the data-flow control within the basic cells in the array. In other words, convert the synchronized systolic array into an asynchronized data driven multiprocessor array, referred to as wavefront arrays. The execution of instructions in the cells of a wavefront array is controlled by the neighbouring cells, i.e a cell will be activated when all the inputs for the cell are available and the outputs of the cell are passed to the next cells when the cells are ready to receive the outputs. This would increase the hardware complexity of the basic cells but would eliminate the need for a global clock. As with the systolic arrays, wavefront arrays are highly regular and communication within the array is localized. The array can also be pipelined since the data from one cell will never intersect with the data from other neighbouring cells.

S. Y. Kung and his colleagues have developed a special-purpose wavefront-oriented language [60] called Matrix Data Flow Language (MDFL). The language is used to describe wavefront or any other data flow algorithms which exhibit the recursion and locality properties. One other useful tool for the simulation of parallel array processors is the language OCCAM [61].

1.4.4- Programmable Array Processors

Systolic and wavefront arrays are special purpose arrays which are designed to map an algorithm into a VLSI architecture. When the PEs are designed then the array would be used for the design of chips dedicated to a fixed parallel processing function. In recent years, work has been done towards the design of programmable array processors. These are two dimensional arrays of identical PEs with nearest neighbouring connection which can be programmed to do different instructions at any clock cycle. At a given clock cycle the array is exactly the same as a systolic array but in the next cycle the PEs' operation can be changed. These arrays are some times referred to as SIMD (Single Instruction, Multiple Data). The earliest machine based on SIMD is the ILLIAC IV [57] and more recent ones include CLIP, developed at the University College of London, DAP, developed at ICL, GRID, developed at the Hirst Research Centre, and last but not least is GAPP which has been developed by NCR-Microelectronics and Martin Marietta Aerospace. GAPP is the most advanced SIMD machine and is organised as a 6x12 array of 1-bit PEs each element comprises a bit-serial ALU, 128 bits of RAM, four 1-bit latches and five multiplexers. The main application of these arrays are in the field of image processing and pattern recognition. Apart from programmable array processors there are other approaches for the VLSI implementation of digital signal processing algorithms. One good example is the silicon compiler developed at the University of Edinburgh and referred to as FIRST. FIRST allows the translation of signal processing algorithms onto silicon and implementing them on single chips [44].

1.4.5- Comparison of Systolic and Wavefront Arrays

In general systolic arrays and wavefront arrays are modular, regular, locally interconnected and highly pipelined processor arrays. The main difference however is that systolic arrays are synchronized while wavefront arrays are asynchronized structures. The global control of the data movement in the systolic array would mean that the cells are much simpler than the wavefront arrays, since no additional handshaking hardware is required for the cells, but on the other hand, it will be a potential barrier in the design of very-large-scale array processors. Wavefront arrays can be extended to any desirable size as long as the technology would permit since the data movement in the array is controlled locally by the PEs.

1.5.0- Objectives in this Thesis

In the previous sections, we highlighted the good properties of WDFs and pointed out that the main drawback of WDFs is due to their hardware complexity. In this thesis, we resolve this problem (a) by considering the design of finite wordlength WDFs and (b) the VLSI implementation of WDF adaptors. Finite wordlength design of WDFs enables us to eliminate the errors due to quantization of the filter coefficients. Also due to the excellent low sensitivity properties of WDFs, we can design filters with short coefficients wordlength. This reduces the hardware complexity of the filter and in some cases the mutiplications can be replaced by arithmetic shifting since the coefficients can be expressed in powers of 2^{-m}. The VLSI implementation of WDFs are achieved by designing systolic structures to implement the WDF adaptors.

Chapter 2 is divided into two parts. In the first part, the design of WDFs using optimization techniques is described. The finite wordlength design of WDFs may be achieved by using a direct search method for the discrete optimization. In this respect, the direct search method of Hooke and Jeeves [62] is described briefly and a subroutine has been developed to implement the algorithm. In the second part of the chapter, some basic systolic arrays are developed to implement some general equations. These basic systolic arrays are used in later chapters to implement the WDF adaptors.

The 2-port adaptor can be used to design and implement WDFs based on unit element and lattice filters. In chapter 3, the finite wordlength and VLSI implementation of these two types of WDF are considered. Also in chapter 3, a single board model of the 2-port systolic adaptor is presented which has been constructed for experimental testing.

In chapter 4, the finite wordlength and systolic implementation of lc-ladder WDFs are considered. The systolic implementation of these filters is achieved by developing systolic arrays for the 3-port adaptors.

Design of frequency selective filters, i.e transformation of lowpass filters to other type of filters, is the subject of chapter 5. Finally in chapter 6, other avenues of the research are outlined for further studies in this field.

CHAPTER TWO

OPTIMIZATION AND SYSTOLIC TECHNIQUES

2.1.0- Introduction

In chapter one, we briefly reviewed the theory of digital filters and introduced the concept of Wave Digital Filters. It was shown that WDFs have many good properties but there is a major problem due to their hardware complexity. We propose to resolve this problem by considering (a) the finite wordlength design of WDFs and (b) the VLSI implementation of the WDF adaptors.

In this chapter, we present the bases on which these goals are achieved. The chapter is divided into two main parts. In the first part, the design of digital filters, in general, using optimization techniques is considered. Next the direct search method of Hooke and Jeeves [62] is briefly reviewed and a subroutine is developed to implement the search algorithm. This subroutine will be used in later chapters to design different WDFs with finite wordlength coefficients.

In the second part of the chapter, some basic systolic arrays are developed which are then used in chapter 3 and 4 to implement the WDF adaptors. At the end of this chapter, a section has been devoted to the design of a universal systolic array. This array will be then modified in chapter 4 to implement a universal systolic WDF adaptor. The universal systolic adaptor can be programmed to realise any type of WDF adaptor.

2.2.0- Use of Optimization Techniques in the Design of Digital Filters

2.2.1- Introduction

Filters play an important role in the design of communication systems and the filter design problem has been treated in several different ways for many years. In one approach, existing synthesis techniques are used to design filters which ensure a satisfactory solution. In other cases, a good design may be obtained by adjusting the parameters of the filter. This may be achieved either by building and testing a prototype or by analysing the filter structure on a computer. The foregoing methods have not been very successfull in the past and a satisfactory design could have not been guaranteed and also the analysis of complex filter structures was not possible. Therefore synthesis techniques were more commonly used in spite of the fact that some filter design cases are unsolvable using synthesis techniques, e.g modeling the effects of finite wordlength.

The advent of high-speed digital computers has provided the designer with an efficient tool which can be used to find the solutions of complex problems. In recent years, substantial effort has been devoted to developing programming techniques to solve filter design problems. Optimization is one such technique and is employed to solve these problems by successive approximation and repetitive adjustment of the filter parameters. Here, the design problem is stated as follows, 'given a filter structure with a number of adjustable parameters, find the values of these parameters such that a prescribed specification is met'. There might however be some restrictions on the values that the filter parameters may have and usually in many cases the optimization algorithm is not allowed to change the filter configuration by adding more elements or changing their interconnections.

In this section, we shall first study the characteristics of an optimization technique in general. Although some of these explanations may appear trivial it is important that they are clearly undestood. Next the concept of error and minimization criterions are introduced and some error functions are described. Depending on the nature of the system under consideration and the designer's requirement, we can divide the optimization problem into different groups. One group covers all the optimization techniques which are based on mathematical treatment of the problem. Here, the parameters are not restricted by the optimization program and may take any real or complex values as long as they meet the designer's constraints. The other group of optimization techniques is referred to as simple methods or discrete programming. Here, the parameters are selected from a set of values defined by the designer and may not take any values outside the set. The second technique is suitable for the design of WDFs with finite wordlength coefficients, since after deciding on the number of bits for the representation of the filter coefficients, then there will be a finite set of values from which the coefficients may be chosen.

2.2.2- Characteristics of an Optimization algorithm

As mentioned before, the aim of an optimization procedure is to adjust the parameters of the system until the performance of the system meets a given specification. Fig. 2.1 shows the features which every optimization procedure must possess.

First it is necessary that the performance of the system under consideration is obtained. In most cases, this is achieved by simulation. As in optimization methods this computation may have to be performed many times, this step should take as little time as possible. The box labelled 'algorithm' (Fig. 2.1) is the heart of an optimization procedure which implements the algorithm suitable for one's application. The next thing to consider is how the performance of the system may be judged. This brings us to the concept of an error criterion which is considered in the next section.

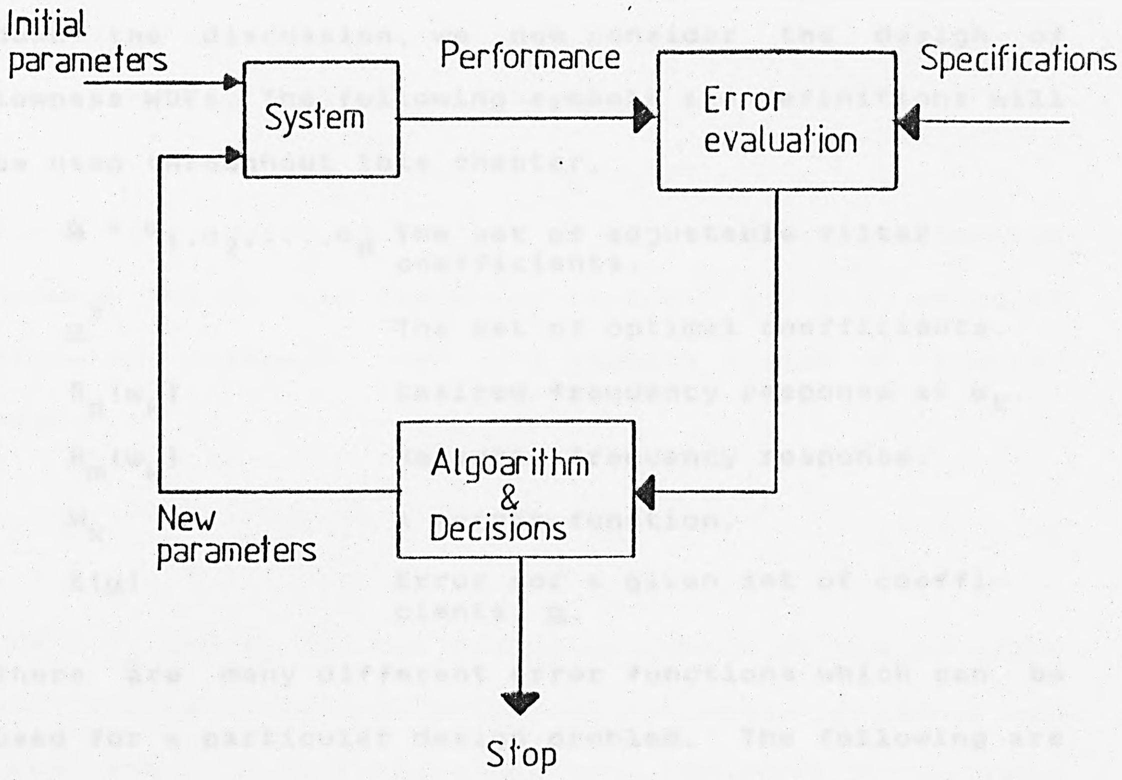


Fig. 2.1_ Features of an optimization procedure.

2.2.3- Error Criterion

In an optimization problem, an error criterion must be used with which we can associate a figure of merit. This error criterion will reflect the goodness of the design. This merit must be kept constant until the design specifications are met completely. In order to narrow down the discussion, we now consider the design of lowpass WDFs. The following symbols and definitions will be used throughout this chapter.

$\underline{\alpha} = \alpha_1, \alpha_2, \ldots, \alpha_N$	The set of adjustable filter coefficients.
$\underline{\alpha}^*$	The set of optimal coefficients.
$R_d(\omega_k)$	Desired frequency response at ω_k.
$R_m(\omega_k)$	Measured frequency response.
W_k	A weight function.
$E(\underline{\alpha})$	Error for a given set of coefficients, $\underline{\alpha}$.

There are many different error functions which can be used for a particular design problem. The following are three error functions which have been considered in this thesis,

1) $E(\underline{\alpha}) = (W_k/M)\Sigma|R_d(\omega_k) - R_m(\omega_k)|^2$ (2.1)

2) $E(\underline{\alpha}) = (W_k/M)\Sigma|R_d(\omega_k) - R_m(\omega_k)|$ (2.2)

3) $E(\underline{\alpha}) = \text{Max}\,[W_k|R_d(\omega_k) - R_m(\omega_k)|]$ (2.3)

$k = 1, 2, \ldots, m$

where m is a set of extremal frequencies at which the WDF will be analysed. In order to obtain a possible solution to the design problem, m must be larger than N, where N is the number of coefficients in the filter.

2.2.4- Statement of the Design Problem

In the previous section, it was shown that, by defining an error function, the optimization problem reduces to a search for a set of coefficients for the filter to minimize the error function. Now, let the error function, $E(\underline{\alpha})$, be as follows,

$$E(\underline{\alpha}) = \text{Max } [W_k \ |R_d(\omega_k) - R_m(\omega_k)|]$$

where the error is defined as the maximum difference between the desired frequency response and the measured frequency response. Now if a lowpass filter is required then,

$$R_d(\omega) = \begin{cases} 1 & \omega \ \varepsilon \ I_p \\ 0 & \omega \ \varepsilon \ I_s \end{cases}$$

where I_p and I_s are the frequency points in the passband and the stopband at which the frequency response is calculated. A subroutine has been developed to evaluate the error function given by eqn. 2.3. The other two error functions have also been tried but it was founded that eqn. 2.3 produces faster results. The parameters of this subroutine are as follows,

x	An array of dimension N holding the current values of the coefficients.
N_p	Number of points in the passband.
N_a	Number of points in the stopband.

N	Number of filter coefficients.
a_p	Maximum passband ripple.
a_s	Minimum stopband attenuation.
f_p	Passband edge frequency.
f_s	Stopband edge frequency.
E_{max}	Maximum error in the passband & stopband.
Flag	A boolean variable which is set to true when the specifications are met.

This error function subroutine requires a subroutine which calculates the frequency response of the filter at a given frequency point for the current values of the coefficients.

Having stated the design problem as an optimization problem, we now need to decide on what type of optimization technique to use. We can classify the optimization techniques into the following classes.

1) Gradient Methods : In which we need to obtain the first derivatives of the error function.

2) Second-order Method : In which higher order derivatives are required.

3) Simple Methods : These donot require the derivatives.

Many algorithms have been proposed for the design of FIR and IIR digital filters with finite wordlength coefficients [63-70], but there has not been enough work on the finite wordlength design of WDFs. One contribution towards this is by Wegener [22], but he only considers the design of lattice WDFs. Claesen and his colleagues at the University of Leuven, Belgium, have developed complete VLSI tools for the design and implementation of arbitrary digital filters [45,46]. They have designed many lattice WDFs with applications in the design of transmultiplexers [71,72]. Their CAD tools contain discrete optimization algorithms [73-75] which are used to optimize the filter coefficient wordlengths. They employ bit-serial architectures and use CSD codes to represent the coefficients. Therefore the optimization methods are used to minimize the number of ones required to represent the filter coefficients. Thus, one coefficient may need 8-bits to be represented while another coefficient may need 3-bits.

Our approach in this thesis is different in the following points. First, we intend to develop a set of programs as a complete package for the design, analysis and simulation of WDFs only. This includes all the well known WDFs, i.e unit element, lattice and lc-ladder WDFs. Also the designer is allowed to include the required number of bits for the coefficients and all the final coefficients will be quantized to the required number of bits. These coefficients are suitable for both word-parallel and bit-serial architectures.

The optimization method we use can be classified as a simple method. In the next section, we briefly review this optimization technique and a subroutine is developed to implement the algorithm.

2.3.0- Direct Search Methods for The Design of WDFs

2.3.1- Introduction

In a direct search method, the objective function, $E(\underline{\alpha})$, is minimized or maximized by evaluating $E(\underline{\alpha})$ at a given set of points $\alpha_1, \alpha_2, \ldots, \alpha_N$, and comparing values to find an optimal set $\underline{\alpha}^*$. These points are chosen from a set of points specified by the designer. There are many reasons for using direct search methods rather than the gradient methods,

1) If the function to be minimized or maximized is not differentiable.

2) If the derivatives of the function ares discontinuous or very difficult to evaluate.

3) If the solution set for the parameters is discrete, as is the case here.

4) Last but not least, direct search methods are very much simpler to implement than the gradient methods.

In the next section, we briefly consider the direct search method of Hooke and Jeeves. An example has also been given to illustrate the use of the search algorithm for the minimization of a function of 2-variables.

2.3.2 Hooke and Jeeves' Method [62]

In 1961, Hooke and Jeeves reported an excellent method of optimization which is now one of the most widely used direct search methods. In their original paper, they reported that the method has been used successfully to solve many curve fitting problems for which other methods had failed. This algorithm has also been used by other people for the finite wordlength design of digital filters [22,76,77]. The following is a brief description of the algorithm.

We wish to consider the problem of minimization of an error function $E(\underline{\alpha})$ of N variables. First we choose an initial base vector $(\underline{\alpha})$ and a stepsize Δ. The value of the function at $\underline{\alpha}$, i.e $E(\underline{\alpha})$, is evaluated and we procced with a sequence of exploratory and pattern moves. If an exploratory move leads to a decrease in the value of $E(\underline{\alpha})$, it is called a success, otherwise it is a failure.

- **Exploratory Moves**

The purpose of an exploratory move is to acquire information about the objective function in the neighbourhood of the current base vector. This information is obtained as follows,

1) $i=1$;

2) Evaluate $E(\alpha_i+\Delta)$. If it is a success, replace α_i by $(\alpha_i+\Delta)$. If it is a failure, evaluate $E(\alpha_i-\Delta)$. If it is a success, replace α_i by $(\alpha_i-\Delta)$, otherwise resume the initial value of α_i;

3) $i=i+1$. Repeat from (2). This is done N-times, i.e for each coefficient.

4) If there is no success, replace Δ by $(step*\Delta)$, where (step) is a constant and less than one and repeat

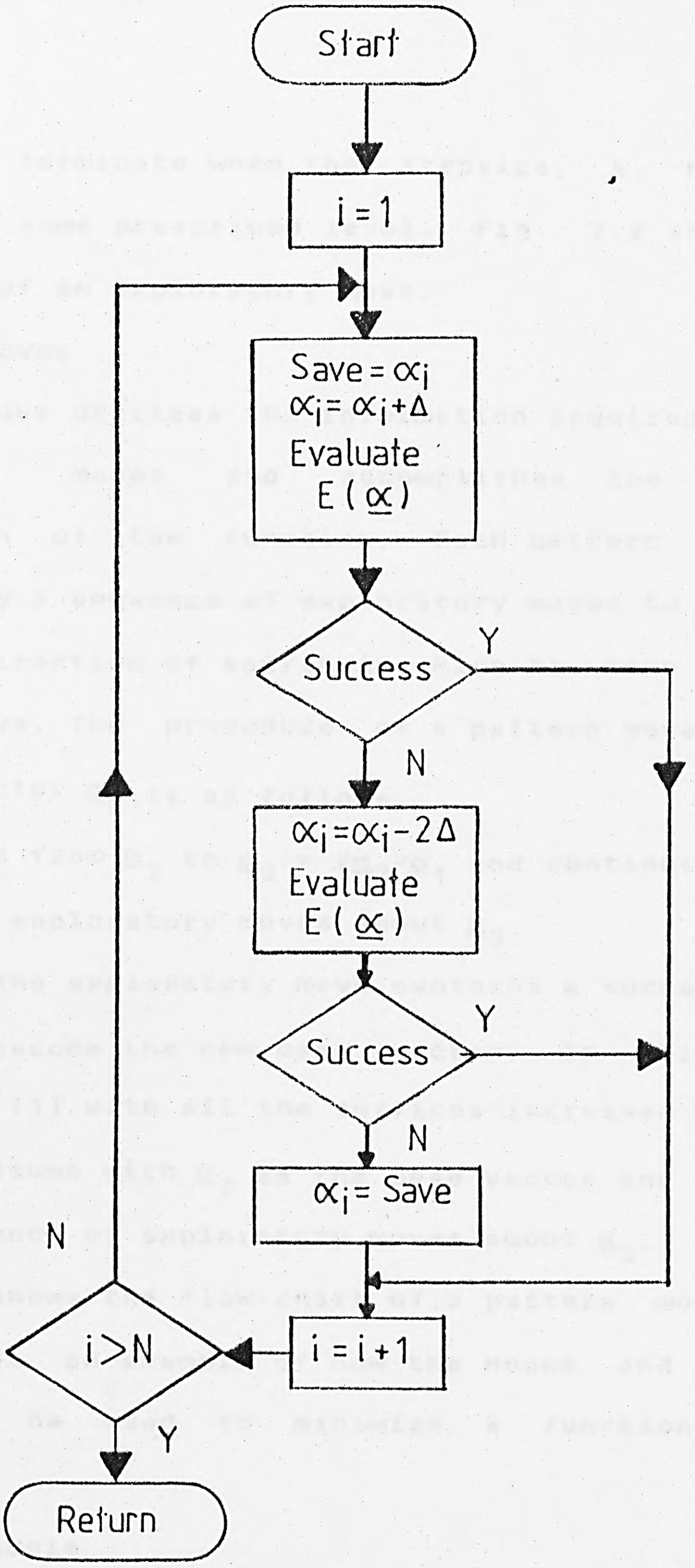

Start
i = 1
Save = α_i
$\alpha_i = \alpha_i + \Delta$
Evaluate
E ($\underline{\alpha}$)
Success
Y
N
$\alpha_i = \alpha_i - 2\Delta$
Evaluate
E ($\underline{\alpha}$)
Success
Y
N
α_i = Save
N
i = i + 1
i > N
Y
Return

Fig. 2.2 _ Flow _ chart of an Exploratory move.

from (1).

This would terminate when the stepsize, Δ, has been reduced to some prescribed level. Fig. 2.2 shows the flow-chart of an exploratory move.

- **Pattern Moves**

A pattern move utilizes the information acquired in the exploratory moves and accomplishes the actual minimization of the function. Each pattern move is followed by a sequence of exploratory moves to find an improved direction of search in which to make another pattern move. The procedure of a pattern move from a new base vector $\underline{\alpha}_2$ is as follows,

1) Move from $\underline{\alpha}_2$ to $\underline{\alpha}_3 = 2\underline{\alpha}_2 - \underline{\alpha}_1$ and continue with a sequence of exploratory moves about $\underline{\alpha}_3$.

2) If the exploratory move contains a success then $\underline{\alpha}_3$ would become the new base vector. In this case, return to (1) with all the suffices increased by one. Otherwise resume with $\underline{\alpha}_2$ as the base vector and continue with a sequence of exploratory moves about $\underline{\alpha}_2$.

Fig. 2.3 shows the flow-chart of a pattern move. The following is an example of how the Hooke and Jeeves' method may be used to minimize a function of 2-variables.

2.3.4- - **Example**

To illustrate Hooke and Jeeves' method, let us consider the minimization of the function, $f(x_1, x_2)$, given by,

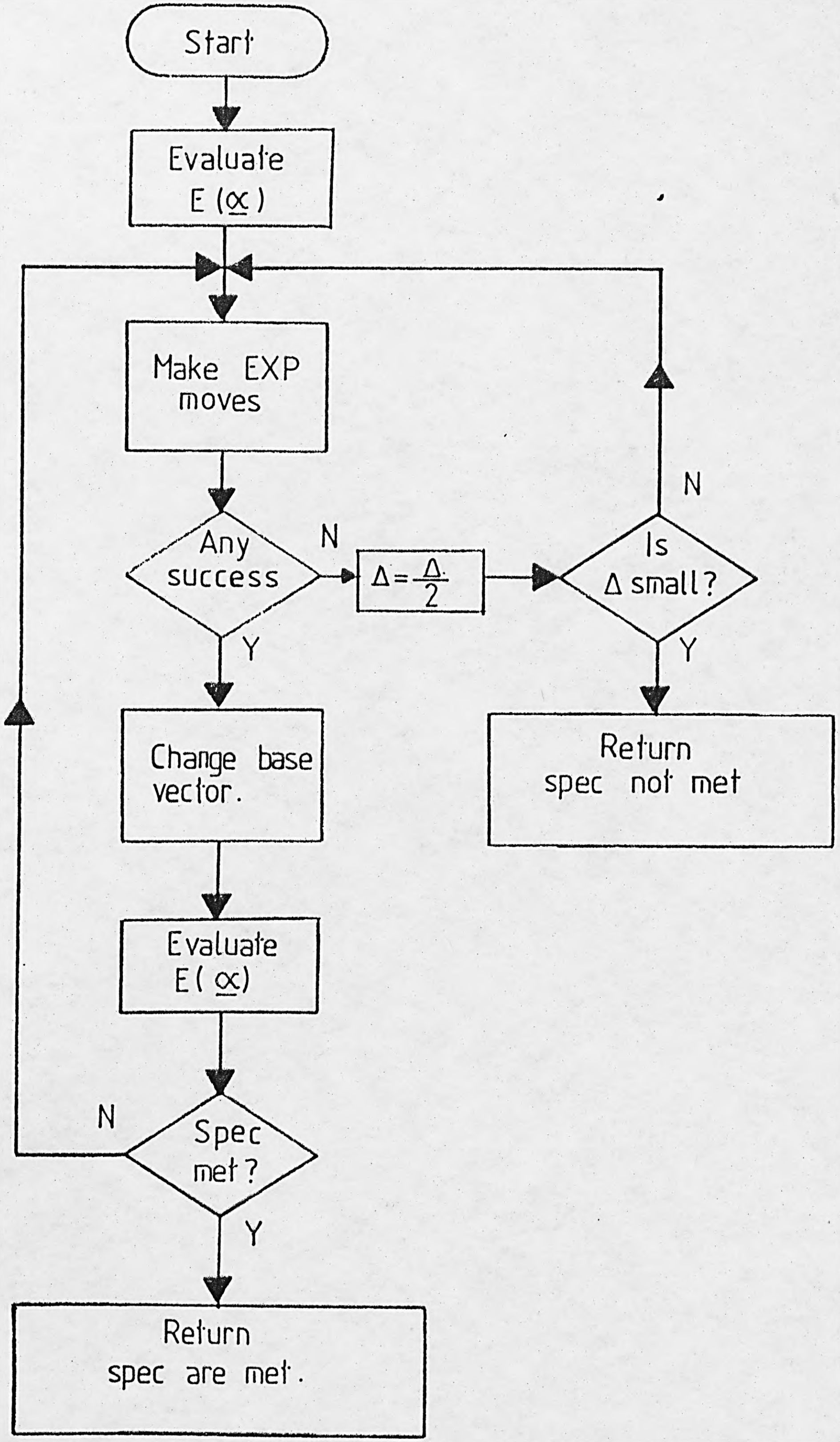


Fig. 2.3_ Flow_chart of a Pattern move.

$$f(x_1,x_2) = 4x^2 + 3x_1x_2 - 5x^2 + 3$$

Subject to

$$x_2 < 6.$$

The initial stepsize, Δ, is 1 and the initial base vector, $\underline{b}_1$, is [0,0]. Also it is required to stop when Δ<(1/4). The value of the step with which the stepsize is reduced is 1/2.

- **Solution**

Let $E(\underline{x})$ and $P(\underline{x})$ denote an exploratory and a pattern move about a base vector ($\underline{x}$) respectively. Also let S and F denote a success and a failure respectively. First we evaluate the value of the function at the initial base vector $\underline{b}_1$,

$$\underline{b}_1 = [0,0] \ ; \ f(\underline{b}_1) = 3$$

Now we make a sequence of exploratory moves about $\underline{b}_1$.

$E(\underline{b}_1)$

$f(1, 0)$	$= 7$	F
$f(-1, 0)$	$= 7$	F
$f(0, 1)$	$= -2$	S
$f(0, -1)$	$= 8$	F

$E(\underline{b}_1)$ contains a success for [0,1], therefore we make a pattern move and generate a new base vector from $\underline{b}_2$ and $\underline{b}_1$.

$$\underline{b}_2 = [0,1] \ ; \ f(\underline{b}_2) = -2$$

$P(\underline{b}_2)$

$$\underline{b}_3 = 2\underline{b}_2 - \underline{b}_1 = [0,2] \ ; \ f(\underline{b}_3) = -17$$

Now we make a sequence of exploratory moves about $\underline{b}_3$.

$E(\underline{b}_3)$

$f(1, 2) = -7$	F
$f(-1, 2) = -19$	S
$f(-1, 3) = -53$	S
$f(-1, 1) = -1$	F

Again $E(\underline{b}_3)$ contains a success for $[-1,3]$, therefore we continue with a further pattern move,

$$\underline{b}_3 = [-1,3] \;;\; f(\underline{b}_3) = -53$$

$P(\underline{b}_3)$

$$\underline{b}_4 = 2\underline{b}_3 - \underline{b}_2 = [-2,5] \;;\; f(\underline{b}_4) = -136$$

$E(\underline{b}_4)$

$f(-1, 5) = -133$	F
$f(-3, 5) = -131$	F
$f(-2, 6) = -197$	F

Since x_2 should be less than 6.

$f(-2, 4) = -85$	F

$E(\underline{b}_4)$ does not contain any success, therefore we set the stepsize, Δ, equal to $(\Delta/2)$, i.e 1/2, and carry on with a sequence of exploratory moves about $\underline{b}_4$ with new Δ,

$E(\underline{b}_4)$

$f(-1.5, 5) = -135.5$	F
$f(-2.5, 5) = -134.5$	F
$f(-2, 5.5) = -165.25$	S
$f(-2, 4.5) = -109.25$	F

Now $E(\underline{b}_4)$ contains a success for $[-2,5.5]$. We generate the new base vector by making a further pattern move,

$$\underline{b}_4 = [-2,5.5] \;;\; f(\underline{b}_4) = -165.25$$

$P(\underline{b}_4)$

$$\underline{b}_5 = 2\underline{b}_4 - \underline{b}_3 = [-3,9] \;;\; f(\underline{b}_5) = -447$$

The value of x_2 is not in the range, therefore we need to reduce the value of the stepsize, i.e $(\Delta/2)$. The new value of Δ, i.e 1/4, is also not in the prescribed range. So the algorithm stops here and the final results are,

$$\underline{x}^* = \underline{b}_4 = [-2,5.5] \; ; \; f(\underline{b}_4) = -165.25$$

Now let us suppose that it is required to design a WDF with finite wordlength coefficients. First the initial coefficients are quantized to m-bits. Then the stepsize is set equal to 2^{-q}, where q is smaller than m. With these initial parameters we start the search. If there is no success in the exploratory moves then the stepsize is multiplied by 2^{-1}. The search terminates when $\Delta=2^{-m}$. It is clear that the final coefficients will be given in m-bits or less. If at the end of the search the specifications are not met then the number of bits, m, is increased by one and we start all over again.

A subroutine has been developed which implements the Hooke and Jeeves' direct search algorithm. The algorithm has been modified in such a way that if there is no success in a series of exploratory moves then more than one coefficient is changed simultaneously. This may reduce the speed of the algorithm but results in a smaller coefficient wordlength. The parameters of the subroutine are as follows :

N	Number of filter coefficients.
Nsbit	Number of starting bits for the coefficients.

Pcoeff An array of N-dimensions containing the initial coefficients before entering the subroutine, and contains the final coefficients at the end of the algorithm.

SD The starting value of Δ.

NFE At the end of the algorithm this will contain the number of times the error function subroutine is called.

Rstep The value of (step) by which Δ is reduced.

This subroutine requires to call the subroutine which evaluates the error function for the current value of the coefficients. This optimization subroutine is used in chapters 3 and 4 to design finite word length WDFs based on different reference filters.

2.4.0- Development of some Basic Systolic Arrays

2.4.1- Introduction

In chapter one, it was pointed out that in order to gain the full potential of VLSI technology, it is necessary to translate our existing algorithms and structures into new architectures which are suitable for VLSI implementation. The main features of a VLSI structure are modularity, regularity, local communications and ability to be pipelined. In this respect, systolic arrays are good candidates for VLSI implementations. The initial systolic arrays, [49], were implemented at word level and recently bit level systolic arrays have attracted much attention, [52,53,78]. There are many reasons for considering systolic arrays at bit level [79], such as,

1) At bit level a systolic array is constructed from simple cells which consist of logic gates and a number of latches. Many cells of this type can be accommodated on a single VLSI chip and the circuits are easy to design.

2) Experience to date [55,80] suggests that the resulting structures exhibit high device packing densities.

3) Being pipelined at bit level, such circuits provide the maximum possible throughput rate.

4) The required component density of a word level

systolic cell would exceed the capabilities of existing processing technology, and the number of pins needed, when implementing many word level cells on a single VLSI chip, would be unmanageably large [57].

In this section, we develop a number of systolic arrays at bit level to implement equations of the form,

$$R1 = P + Z1(X1 - X2) \quad (2.5a)$$
$$R2 = P + Z1(X1 - X2) + W1(X3 - X4) \quad (2.5b)$$
$$R3 = P - Z1(X1 + X2 + X3) \quad (2.5c)$$

Later in chapters 3 and 4, these basic systolic arrays are used to implement the WDF adaptors. Finally in this chapter, we present a universal systolic array which can be programmed to implement all the three equations given above. It will then be used in chapter 4 to implement a universal systolic adaptor which can realise 2-port, 3-port serial and 3-port parallel adaptors. Throughout this chapter, it is assumed, without loss of generality, that all the inputs are expressed as 3-bit integers.

2.4.2- The Basic Systolic Arrays

In this section, we develop some basic systolic arrays to implement eqn 2.5, but first let us consider the bit level implementation of the basic operation of multiplication. This has also been considered in Ref [25,78] but is given here since it helps to understand how the other systolic arrays are developed.

Suppose we wish to multiply X and Y which are 3-bit integer numbers to produce Z which will be a 6-bit integer number. We use lowercase letters x,y and z to denote the individual bits of X, Y and Z respectively. Therefore,

$$\begin{aligned} X &\equiv x_2x_1x_0 \\ Y &\equiv y_2y_1y_0 \\ Z &\equiv z_5z_4z_3z_2z_1z_0 \end{aligned}$$

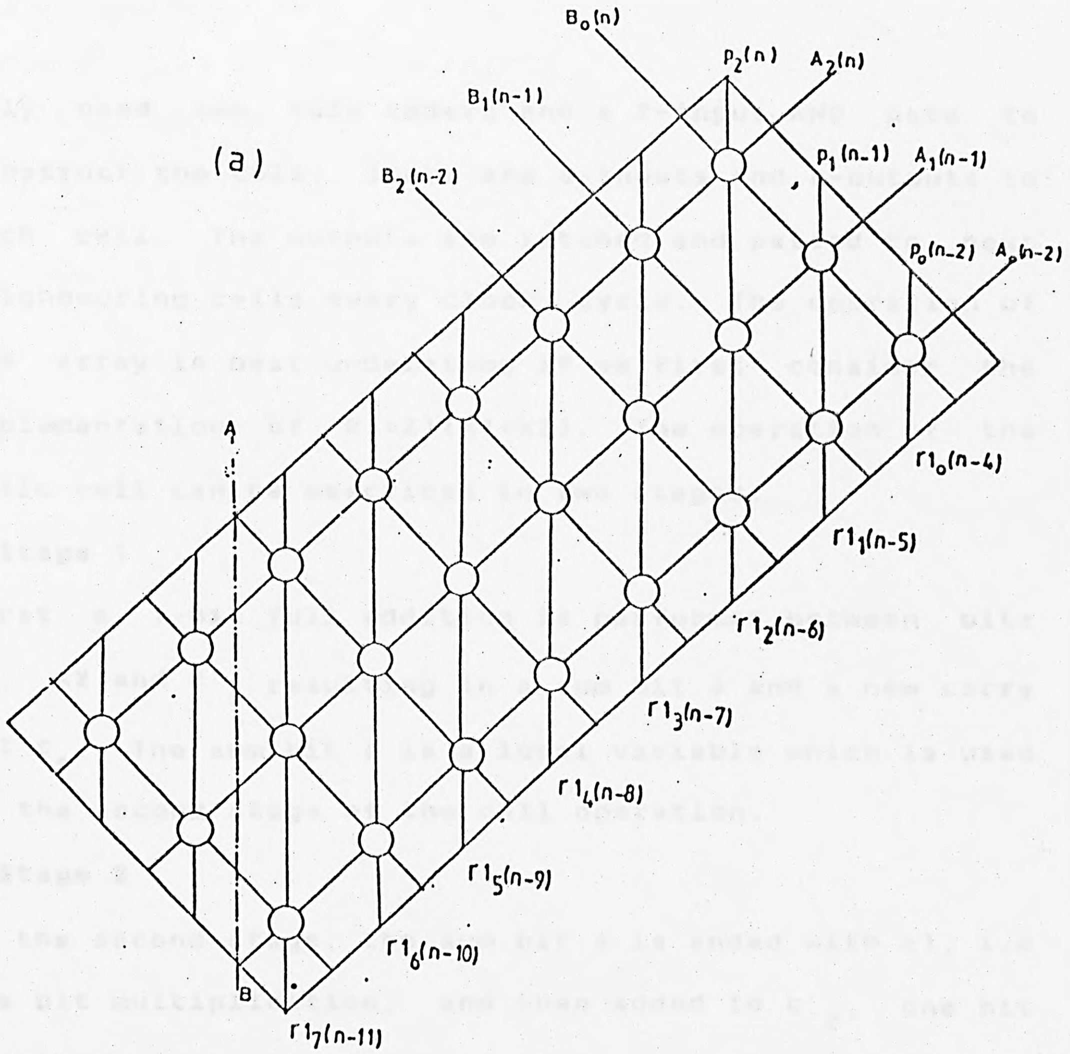
The individual bits of Z, z_j, can be expressed in terms of x_i and y_i as follows,

$$\begin{aligned} z_0 &= x_0y_0 \\ z_1 &= x_0y_1 + x_1y_0 + c_0 \\ z_2 &= x_0y_2 + x_1y_1 + x_2y_0 + c_1 \\ z_3 &= x_1y_2 + x_2y_1 + c_2 \\ z_4 &= x_2y_2 + c_3 \\ \text{and} \qquad z_5 &= c_4 \end{aligned}$$

where c_i is the carry resulting from the previous addition. The z_j bits can also be expressed as,

$$z_j = \Sigma\,(x_iy_{j-i}) + c_{j-1}$$
$$i=0,1,2 \quad \text{and} \quad j=0,1,\ldots,5$$

where $c_{-1}=0$. It can be seen that the bit level operation of multiplication can be accomplished by some form of accumulation and addition. With this in mind, Fig. 2.4a illustrates a systolic array suitable for implementing eqn. 2.5a. The array is constructed by interconnecting 24 identical cells. The interconnections are localized and there is no global communication except the system clock which synchronizes the movement of data in the array. Fig. 2.4b shows the block representation and the boolean equations of the basic cell. From Fig. 2.4b, it can be seen that the basic cell is very simple and we



$A_k = [c'_{rk}, z1_k, c'_{sk}]$

$B_k = [x1_k, x2_k]$

Fig. 2.4 – Systolic array to implement R1.

only need two full adders and a 2-input AND gate to construct the cell. There are 8-inputs and 8-outputs to each cell. The outputs are latched and passed to next neighbouring cells every clock cycle. The operation of the array is best understood if we first consider the implementation of R1=Z1(X1+X2). The operation of the basic cell can be described in two stages.

- Stage 1

First a 1-bit full addition is performed between bits x1, x2 and c'_s resulting in a sum bit s and a new carry bit c_s. The sum bit s is a local variable which is used in the second stage of the cell operation.

- Stage 2

In the second stage, the sum bit s is anded with z1, i.e one bit multiplication, and then added to c'_r, one bit carried in from the previous cell, and r', one bit of the accumulating sum of the partial result. The resulting values of r, the corresponding carry bits c_r and c_s and the inputs x1, x2 and z1 are latched and passed onto the neighbouring cells at the end of a clock cycle. One clock cycle is the time taken for one cell to complete its operation.

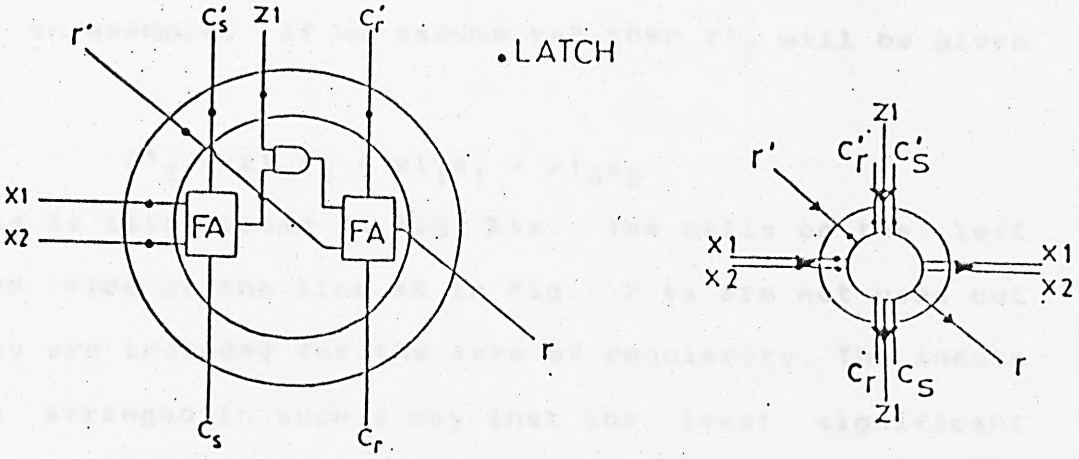
The kth bit of the result R1 can be expressed in terms of the Z1 bits and S bits, where S is given by X1+X2, as shown below,

$$s = x1 \oplus x2 \oplus c'_S$$

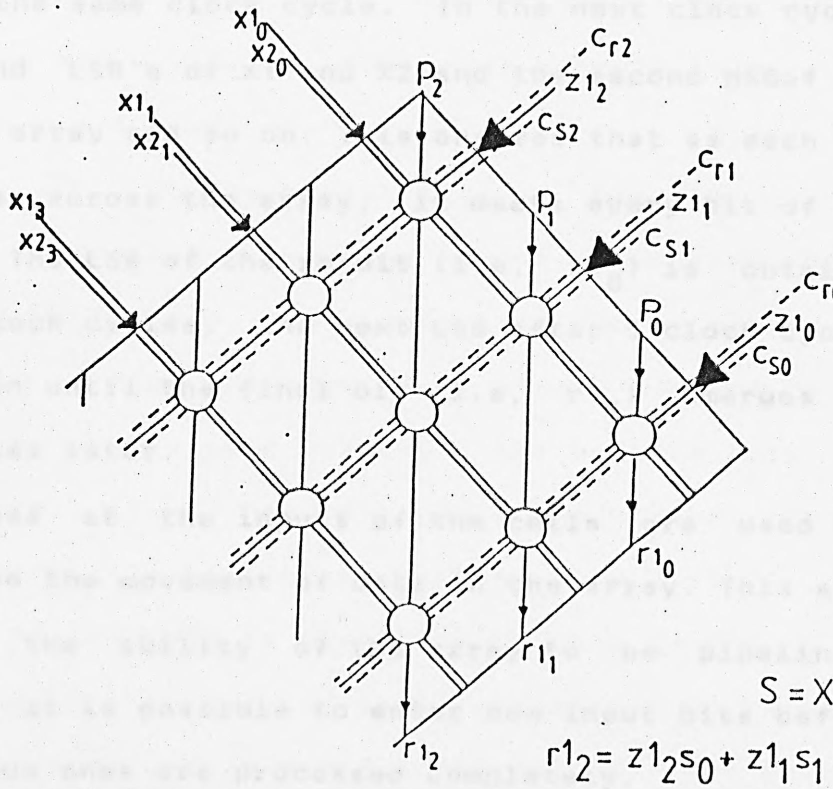
$$c_S = (x1 \,.\, x2) + (x1 \,.\, c'_S) + (x2 \,.\, c'_S)$$

$$r = r' \oplus (z1.s) \oplus c'_r$$

$$c_r = (r'.(z1.s)) + (r'.c'_r) + ((z1.s).c'_r)$$



b).



c).

$$S = X1 + X2$$

$$r1_2 = z1_2 s_0 + z1_1 s_1 + z1_0 s_2$$

Fig. 2.4 – b). Basic cell in Fig.2.4a.

c). Evaluation of $r1_2$.

$$r1_k = \Sigma\, z1_{k-i} s_i \qquad (2.7)$$

$$k=0,1,\ldots,7 \text{ and } i=0,1,2$$

As an example, if we assume k=2 then $r1_2$ will be given as,

$$r1_2 = z1_2 s_0 + z1_1 s_1 + z1_0 s_2$$

This is illustrated in Fig. 2.4c. The cells on the left hand side of the line AB in Fig. 2.4a are not used but they are included for the sake of regularity. The inputs are arranged in such a way that the least significant bits (LSB's) of X1 and X2 (i.e, $x1_0$ and $x2_0$) and the most significant bit (MSB) of Z1 (i.e, $z1_2$) enter the array on the same clock cycle. In the next clock cycle, the second LSB's of X1 and X2 and the second MSBof Z1 enter the array and so on. This ensures that as each bit of Z1 moves across the array, it meets every bit of X1 and X2. The LSB of the result (i.e, $r1_0$) is obtained after 4 clock cycles, the next LSB after 5 clock cycles and so on until the final bit (i.e, $r1_7$) emerges 11 clock cycles later.

The latches at the inputs of the cells are used to synchronize the movement of data in the array. This also increases the ability of the array to be pipelined. Therefore, it is possible to enter new input bits before the previous ones are processed completely.

When the circuit of Fig. 2.4a is used to implement Z1(X1+X2), the carry bits C'_s and C'_r and the bits of P are set equal to '0' when they enter the array. If we however initialize p_k to the kth bit of an arbitrary number P then R1 will be given as,

$$R1 = P + Z1(X1 + X2)$$

Further more, if the carry bits C'_s are set to '1' and $\overline{X2}$ (i.e, inverse of X2) is used instead of X2, then R1 can be expressed as,

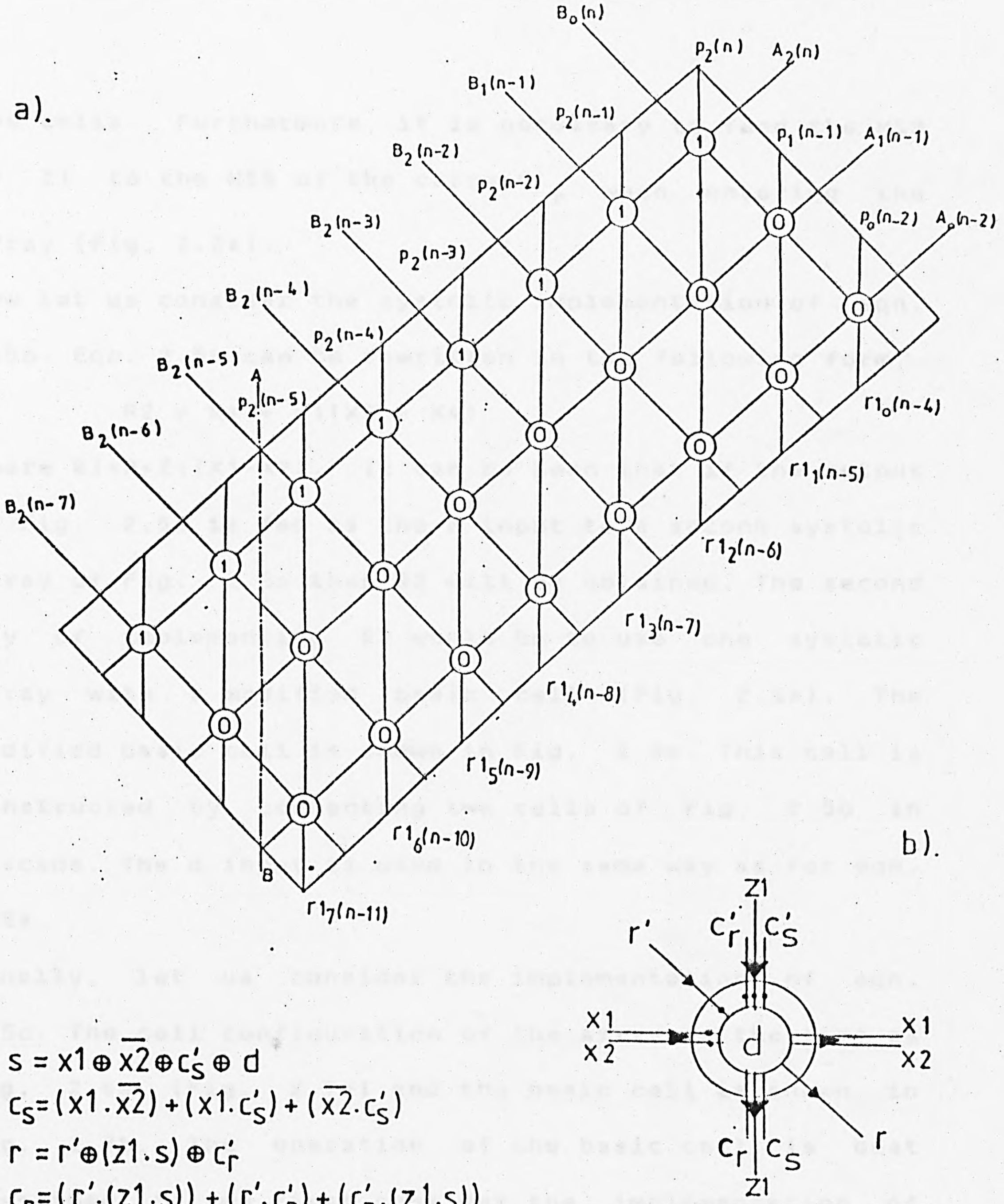
$$R1 = P + Z1(X1 + \overline{X2} + 1)$$

or $$R1 = P + Z1(X1 - X2)$$

since $(X1+\overline{X2}+1)$ is the 2'complement representation of (X1-X2). The circuit of Fig 2.4a is only suitable for unsigned numbers. In Ref [53], it has been shown that if A and B are two m-bit 2's complement numbers then,

$$A.B = a_{m-1}2^{m-1}\bar{B} + a_{m-1}2^{m-1} + \tilde{A}\ B \qquad (2.8)$$

where $(a_{m-1}2^{m-1})$ is the MSB of A (i.e, the sign bit of A) and $\tilde{A}$ represent a positive number comprising the m-1 LSB's of A. Using eqn. 2.8, it is possible to modify the basic cell of Fig. 2.4b to allow signed numbers to be used as well as unsigned numbers. The modified cell is shown in Fig. 2.5b. This cell has an additional input d which controls its mode of operation. When d is '0', the operation of the cell is identical to that of Fig. 2.4b, but when d is '1' then the local variable s in the cell will be inverted before being anded with z1. The input d is set equal to '1' on each of the cells on the left-hand boundary of the array and it is zero on the rest of



$$s = x1 \oplus \overline{x2} \oplus c'_s \oplus d$$

$$c_s = (x1 . \overline{x2}) + (x1 . c'_s) + (\overline{x2} . c'_s)$$

$$r = r' \oplus (z1 . s) \oplus c'_r$$

$$c_r = (r' . (z1 . s)) + (r' . c'_r) + (c'_r . (z1 . s))$$

Fig. 2.5 _ a). Modified array to allow signed numbers to be used.
b). Modified basic cell.

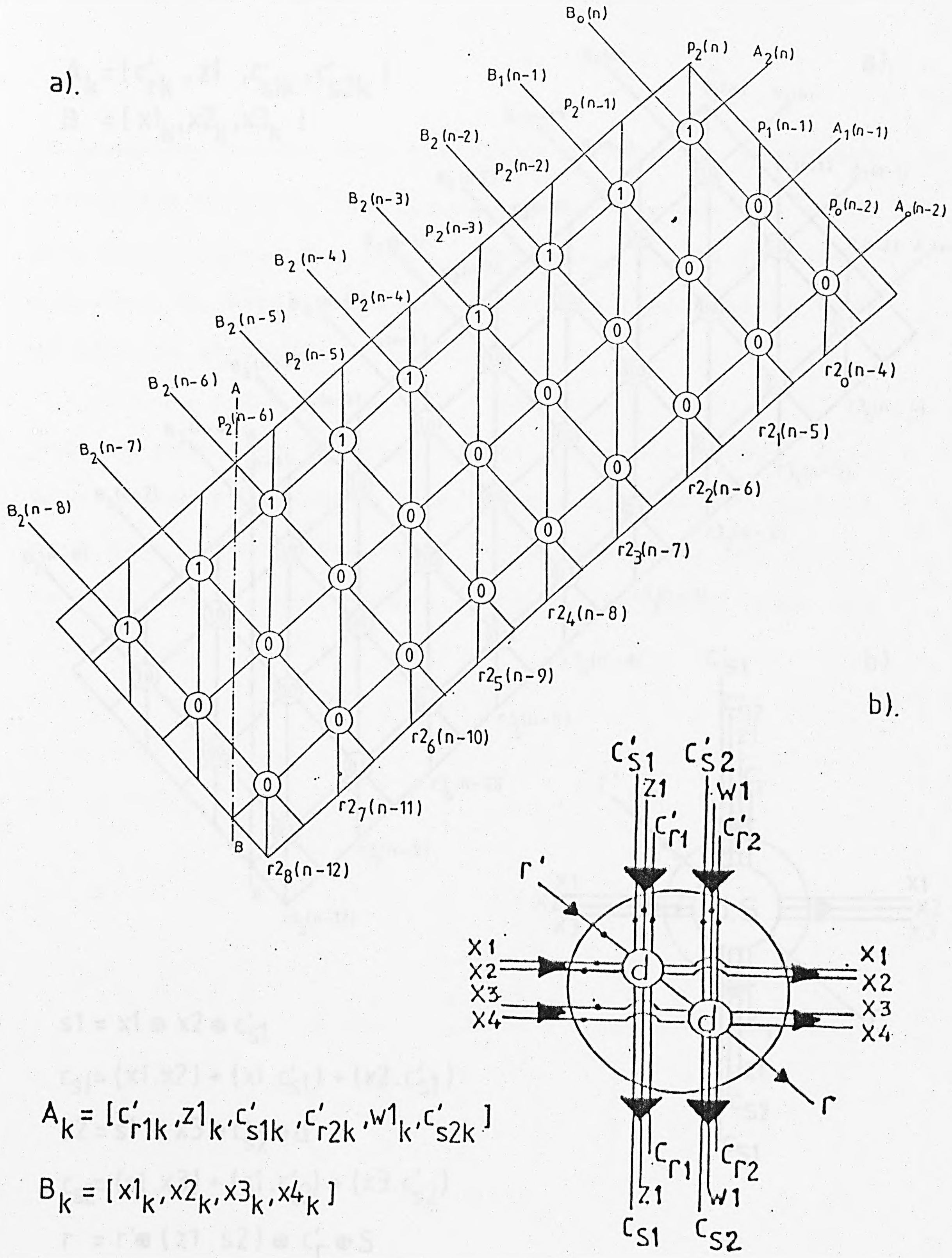
the cells. Furthermore, it is necessary to feed the MSB of Z1 to the MSB of the carry C'_r when entering the array (Fig. 2.5a).

Now let us consider the systolic implementation of eqn. 2.5b. Eqn. 2.5b can be rewritten in the following form,

$$R2 = R1 + W1(X3 - X4)$$

where R1=P+Z1(X1-X2). It can be seen that if the output of Fig. 2.5a is fed as the P input to a second systolic array of Fig. 2.5a then R2 will be obtained. The second way of implementing R2 would be to use one systolic array with a modified basic cell (Fig. 2.6a). The modified basic cell is shown in Fig. 2.6b. This cell is constructed by connecting two cells of Fig. 2.5b in cascade. The d input is used in the same way as for eqn. 2.5a.

Finally, let us consider the implementation of eqn. 2.5c. The cell configuration of the array is the same as Fig. 2.6a (Fig. 2.7a) and the basic cell is shown in Fig. 2.7b. The operation of the basic cell is best described if we first consider the implementation of R3=P+Z1(X1+X2+X3). In the first stage of the cell operation, the bits of x1,x2 and x3 are added resulting in a sum bit s2 and two carry bits c_{s1} and c_{s2}. In the second stage, the sum bit s2 is anded with z1 and added to r' and c'_r. The new values of r, c_r, c_{s1} and c_{s2} and the inputs are latched and passed to the next

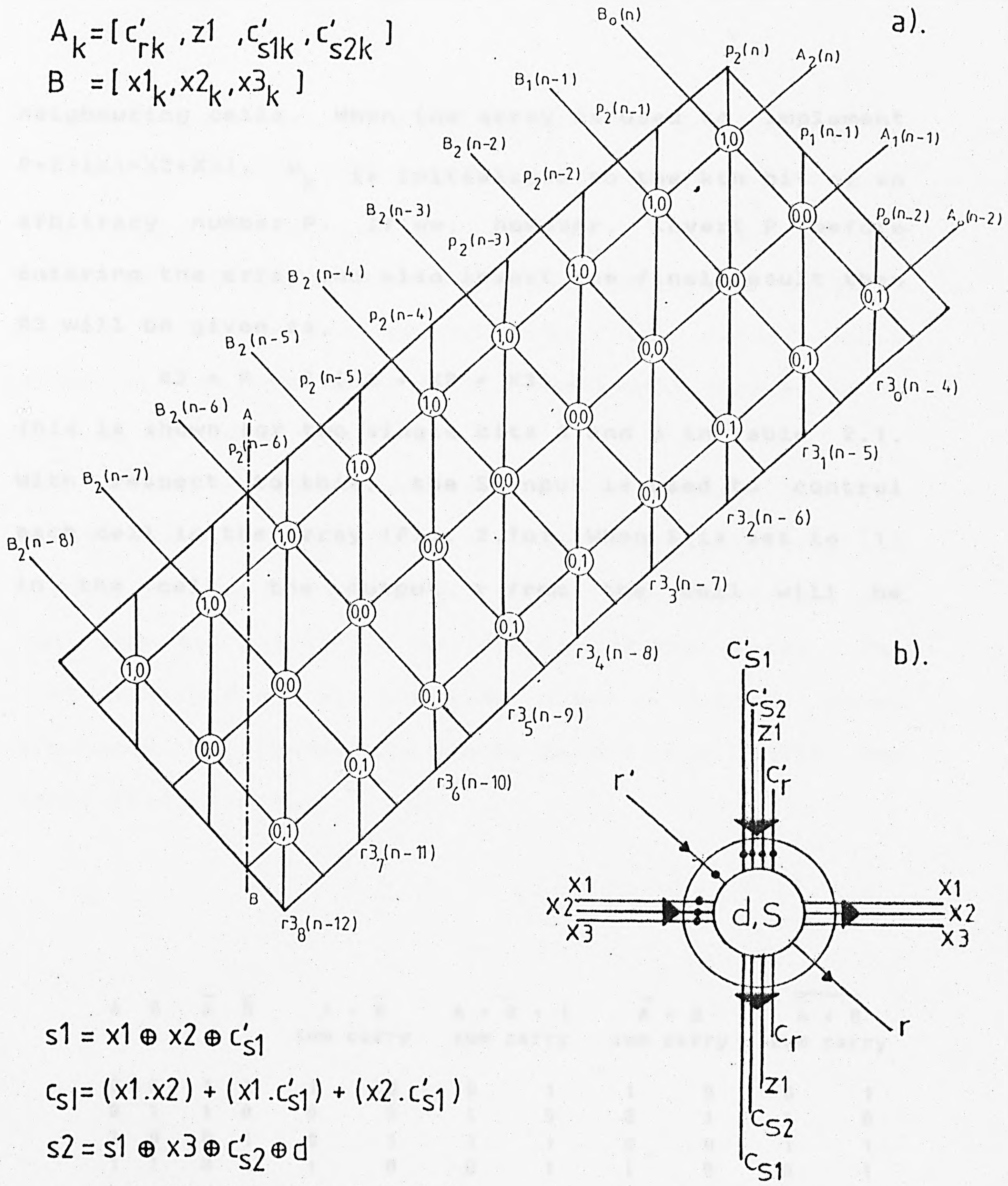


$$A_k = [c'_{r1k}, z1_k, c'_{s1k}, c'_{r2k}, w1_k, c'_{s2k}]$$

$$B_k = [x1_k, x2_k, x3_k, x4_k]$$

Fig. 2.6 _ a). Systolic array to implement R2.
b). Basic cell.

$$A_k = [\, c'_{rk} \,, z1 \,, c'_{s1k} \,, c'_{s2k} \,]$$
$$B \;= [\, x1_k , x2_k , x3_k \,]$$



$$s1 = x1 \oplus x2 \oplus c'_{s1}$$
$$c_{s1} = (x1.x2) + (x1.c'_{s1}) + (x2.c'_{s1})$$
$$s2 = s1 \oplus x3 \oplus c'_{s2} \oplus d$$
$$c_{s2} = (s1.x3) + (s1.c'_{s2}) + (x3.c'_{s2})$$
$$r \;= r' \oplus (z1 \,.\, s2) \oplus c'_r \oplus S$$
$$c_r = (r'.(z1.\, s2)) + (r'.c'_r) + ((z1.s2).c'_r)$$

Fig. 2.7_ a).Systolic array to implement R3.
b).Basic cell.

neigbouring cells. When the array is used to implement P+Z1(X1+X2+X3), p_k is initialized to the kth bit of an arbitrary number P. If we, however, invert P before entering the array and also invert the final result then R3 will be given as,

$$R3 = P - Z1(X1 + X2 + X3)$$

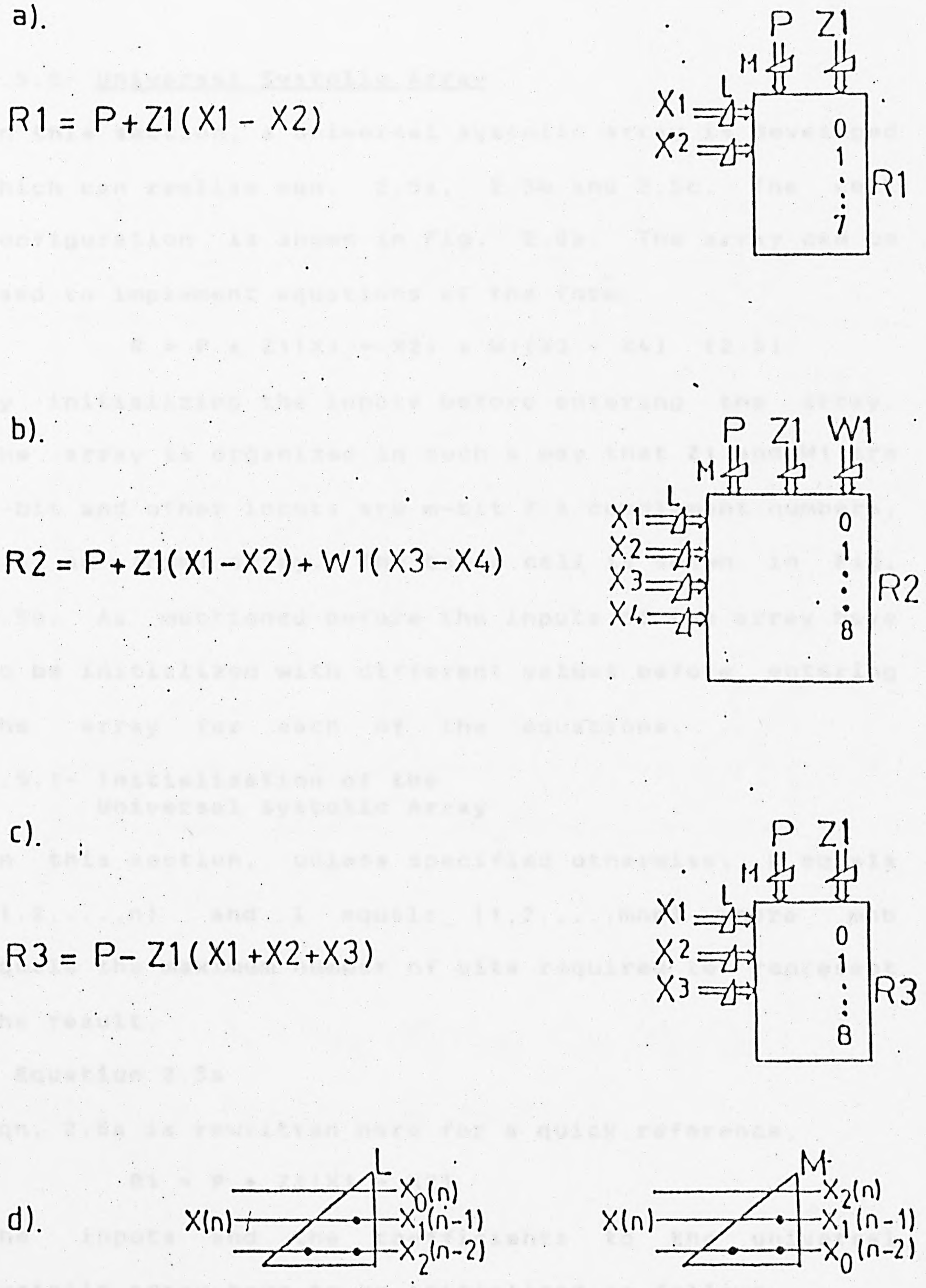
This is shown for two single bits A and B in Table 2.1. With respect to this, the S input is used to control each cell in the array (Fig. 2.7b). When S is set to '1' in the cell, the output r from the cell will be

A	B	$\overline{A}$	$\overline{B}$	$A + \overline{B}$		$A + \overline{B} + 1$		$\overline{A} + B$		$\overline{\overline{A} + B}$	
				sum	carry	sum	carry	sum	carry	sum	carry
0	0	1	1	1	0	0	1	1	0	0	1
0	1	1	0	0	0	1	0	0	1	1	0
1	0	0	1	0	1	1	1	0	0	1	1
1	1	0	0	1	0	0	1	1	0	0	1
						A - B				A - B	

Table 2.1

inverted, and if S is set to '0' then r will not be changed. Since we wish to invert the final result (i.e. $r3_k$, k=0,1,...,12) then S is set to '1' on each of the cells on the right-hand boundary of the array and it is set to '0' on the rest of the cells (Fig. 2.7a). The d input is also used to allow 2's complement numbers to be used as described above.

The schematic representation of the basic systolic arrays are shown in Fig. 2.8. These representations are simplified versions of Fig. 2.5a, 2.6a and 2.7a. and only show the inputs and the outputs of the arrays. The triangles in Fig. 2.8 represent a set of latches which are used to arrange the inputs before they enter the array (Fig. 2.8d).



a).

$$R1 = P + Z1(X1 - X2)$$

b).

$$R2 = P + Z1(X1 - X2) + W1(X3 - X4)$$

c).

$$R3 = P - Z1(X1 + X2 + X3)$$

d).

Fig. 2.8 _ a), b), c). Schematic representation of systolic arrays.
d). Triangular arrangement of latches.

2.5.0- Universal Systolic Array

In this section, a universal systolic array is developed which can realise eqn. 2.5a, 2.5b and 2.5c. The cell configuration is shown in Fig. 2.9a. The array can be used to implement equations of the form,

$$R = P \pm Z1(X1 + X2) \pm W1(X3 + X4) \quad (2.8)$$

by initializing the inputs before entering the array. The array is organized in such a way that Z1 and W1 are n-bit and other inputs are m-bit 2's complement numbers, i.e an (nxm) array. The basic cell is shown in Fig. 2.9b. As mentioned before the inputs of the array have to be initialized with different values before entering the array for each of the equations.

2.5.1- Initialisation of the Universal Systolic Array

In this section, unless specified otherwise, k equals (1,2,...,n) and l equals (1,2,...,mnb) where mnb equals the maximum number of bits required to represent the result.

- Equation 2.5a

Eqn. 2.5a is rewritten here for a quick reference,

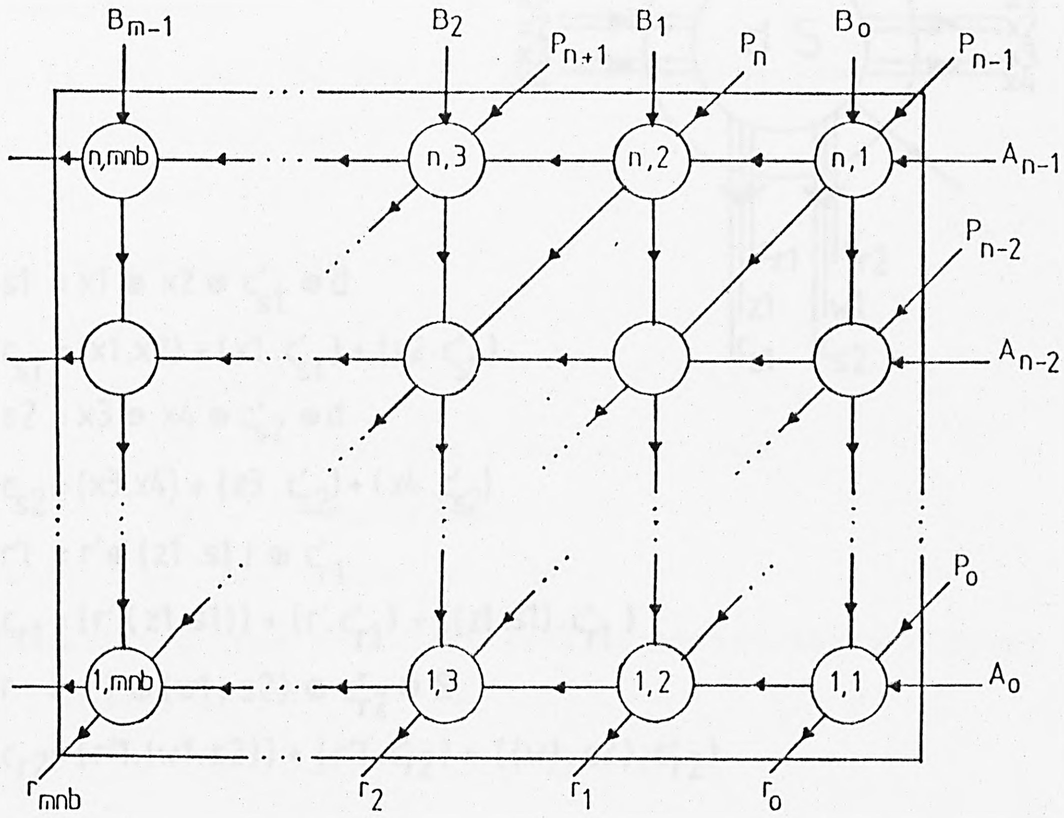
$$R1 = P + Z1(X1 - X2)$$

The inputs and the coefficients to the universal systolic array have to be initialised as follows,

- Inputs

p_l, $x1_l$ are set equal to the lth bit of arbitrary numbers P and X1 respectively.

a).

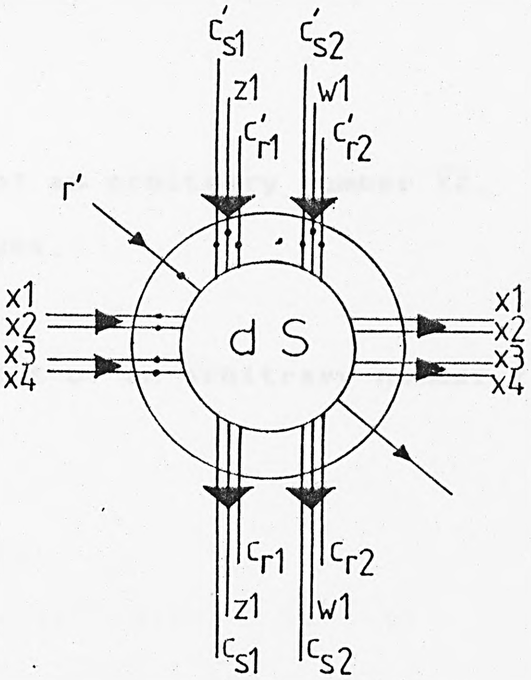


$$A_k = [c'_{r1k}, z1_k, c'_{s1k}, c'_{r2k}, w1_k, c'_{s2k}]$$

$$B_k = [x1_k, x2_k, x3_k, x4_k]$$

Fig. 2.9_ a). Universal systolic array.

b).



$$s1 = x1 \oplus x2 \oplus c'_{s1} \oplus d$$

$$c_{s1} = (x1.x2) + (x1.c'_{s1}) + (x2.c'_{s1})$$

$$s2 = x3 \oplus x4 \oplus c'_{s2} \oplus d$$

$$c_{s2} = (x3.x4) + (x3.c'_{s2}) + (x4.c'_{s2})$$

$$r'1 = r' \oplus (z1.s1) \oplus c'_{r1}$$

$$c_{r1} = (r'.(z1.s1)) + (r'.c'_{r1}) + ((z1.s1).c'_{r1})$$

$$r = r'1 \oplus (w1.s2) \oplus c'_{r2} \oplus S$$

$$c_{r2} = (r'1.(w1.s2)) + (r'1.c'_{r2}) + ((w1.s2).c'_{r2})$$

c).

$$R = P \pm z1\,(X1 \mp X2) \pm W1\,(X3 \mp X4)$$

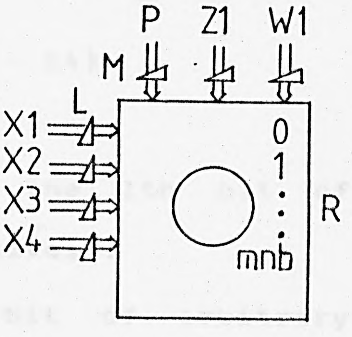


Fig. 2.9_ b). Basic cell for universal systolic array.
c). Schematic representation of universal systolic array.

$x2_l$ is set equal to lth bit of an arbitrary number $\overline{X2}$.

$x3_l$ and $x4_l$ can take any values.

- Coefficients

$z1_k$ is set equal to the kth bit of an arbitrary number Z1.

$w1_k$ is set equal to '0'.

- Carry bits

c_{s1k} is set equal to '1'.

c_{s2k} do not care.

- Control inputs

$$d(k,l) = \begin{cases} 1 & \text{for } k=n \ ; \ l=1,2,\dots,nmb \\ 0 & \text{otherwise} \end{cases}$$

S(k,l) is set equal to '0'.

Initialising the universal array with the above values will result in,

$$R = P + Z1(X1 + \overline{X2} + 1) + 0$$

or $$R = P + Z1(X1 - X2) = R1$$

- **Equation 2.5b**

$$R2 = P + Z1(X1 - X2) + W1(X3 - X4)$$

- Inputs

p_l, $x1_l$ and $x3_l$ are set equal to the lth bit of arbitrary numbers P, X1 and X3 respectively.

$x2_l$ and $x4_l$ are set equal to the lth bit of arbitrary numbers $\overline{X2}$ and $\overline{X4}$ respectively.

- Coefficients

$z1_k$ and $w1_k$ are set equal to the kth bit of arbitrary numbers Z1 and W1 respectivily.

- Carry bits

c_{s1k} and c_{s2k} are set equal to '1'.

- Control inputs

$$d(k,l) = \begin{cases} 1 & \text{for } k=n \text{ ; } l=1,2,\ldots,mnb \\ 0 & \text{otherwise} \end{cases}$$

S(k,l) is set equal to '0'.

These initial values result in,

$$R = P + Z1(X1 + \overline{X2} + 1) + W1(X3 + \overline{X4} + 1)$$

or $$R = P + Z1(X1 - X2) + W1(X3 - X4) = R2$$

-Equation 2.5c

$$R3 = P - Z1(X1 + X2 + X3)$$

- Inputs

p_l, $x1_l$, $x2_l$ and $x3_l$ are set equal to the lth bit of arbitrary numbers $\overline{P}$, X1, X2 and X3 respectively.

$x4_l$ is set equal to '0'.

- Coefficients

$z1_k$ and $w1_k$ are set equal to the kth bit of an arbitrary number Z1.

- Carry bits

c_{s1k} and c_{s2k} are set equal to '0'.

- Control inputs

$$d(k,l) = \begin{cases} 1 & \text{for } k=n \text{ ; } l=1,2,\ldots,mnb \\ 0 & \text{otherwise} \end{cases}$$

$$S(k,l) = \begin{cases} 1 & \text{for } k=1 \text{ ; } l=1,2,\ldots,mnb \\ 0 & \text{otherwise} \end{cases}$$

These initial settings will result in,

$$\overline{R} = \overline{P} + Z1(X1 + X2) + Z1(X3 + 0)$$

or $$R = P - Z1(X1 + X2 + X3) = R3$$

In all the above situations, the carry bits C'_{r1} and C'_{r2} are set as follows,

c'_{r1k} and c'_{r2k} are set equal to '0' except for $c'_{r1n}=z1_n$ and $c'_{r2n}=w1_n$.

This universal array will be used in chapter 4 to implement a universal systolic adaptor. The universal systolic adaptor can be programmed to realise the 2-port, 3-port serial and 3-port parallel adaptors.

2.6.0- Summary

In this chapter, we have seen how optimization can be used to solve filter design problems which are not solvable using direct synthesis techniques. There are many different algorithms which can be used to minimize an error function and there are many different error functions to use. In this chapter, we used the direct search method of Hooke and Jeeves and the error function was defined as the maximum error in the passband and the stopband of the filter. Two subroutines have been developed. The first one evaluates the error function for the current value of the coefficients and the second one implements the search algorithm. These subroutines are used in chapter 3 and 4 to develop a complete package for the design of finite wordlength WDFs.

We have also seen how the concept of systolic array at bit level can be used to implement a number of basic equations. The resulting arrays exhibit high regularity and modularity and the interconnections between the cells are localised. These features make the array suitable for VLSI implementation. In chapter 3 and 4, we express the WDF adaptor equations in the same form as eqns. 2.5a-c and these systolic arrays will be used to implement the adaptor equations.

Also in this chapter, we presented a universal systolic array suitable to implement eqns 2.5-c by initializing the inputs before entering the array. This array will be used to develop a universal systolic WDF adaptor. Since the basic cell of the universal array is more complex than the other arrays, the resulting universal adaptor can be very useful for experimental purposes.

CHAPTER THREE

UNIT ELEMENT AND LATTICE WDFS

3.1.0- Introduction

As mentioned in chapter one, apart from the lc-ladder filters, unit element and lattice filters also have low sensitivity properties to variations in their component values. Threrefore they can also be used to derive WDFs.

Unit element filters result in the simplest form of WDFs. They can be designed and implemented by cascading a number of 2-port adaptors. The only problem with the unit element WDFs (UEWDFs) is that only Butterworth and Chebychev filters can be designed.

There are a number of ways that a Wave Digital Lattice Filter (LTWDF) can be designed. One approach, which will be adopted in this thesis, is to design the lattice reactances using all-pass functions which can be implemented using only 2-port adaptors [81]. The resulting structures are more complex than the UEWDFs, but we can also design filters with Elliptic responses. Usually analogue lattice filters exhibit higher sensitivity in the stopband than the lc-ladder or unit element WDFs, but surprisingly more attention has been given to the design and implementation of LTWDFs in practice. This may be because of some properties of LTWDFs which make them suitable for communication systems such as the design of transmultiplexers [71,72]. In section 3.2 of this chapter, we consider the bit level systolic implementation of a 2-port adaptor using the concept developed in chapter two. A single board 2-port systolic adaptor has been constructed to prove the correctness of the design.

Next in section 3.3 and 3.4, we consider the finite wordlength design of unit element and lattice WDFs respectively. Two subroutines are developed to evaluate the responses of the UEWDFs and LTWDFs for a set of coefficients at different frequency points. These subroutines plus the subroutines developed in chapter two, i.e the error function and optimization subroutines, are used together to form a software package for the finite wordlength design of unit element and lattice WDFs. In section 3.5, we consider the hardware implementation of the UEWDFs and LTWDFs using the 2-port systolic adaptor.

In section 3.6, we briefly describe the design programs and how they can be used. The designed filters can be checked by either analysing the filters using a program called ANAWDF or simulating the filters using SIMWDF. SIMWDF also includes the systolic simulation of the WDFs.

Finally in section 3.7, we consider a number of filter design examples. The examples are carefully selected in order to cover different types of specifications such as narrow band, wide band, sharp cutoff frequencies, etc. The filters are designed both with high precisions for the coefficients and also with finite wordlength coefficients. The filters are analysed and the frequency responses of the filter for the ideal case, with quantized coefficients and with coefficients from the finite wordlength design program (FWLD) are plotted. Although WDFs have low sensitivity properties to variations in the multiplier coefficients, It will be shown that when the number of bits for the coefficients is small then the frequency response of the filter can not be guaranteed to meet the specifications. Using the FWLD programs, the response of the filter can be forced to remain within the specifications. The filters are also simulated using the ideal and the systolic 2-port adaptors.

3.2.0- 2-Port Systolic Adaptors

In chapter two, we saw how equations of the form,

$$R1 = P + Z1(X1 - X2) \qquad (3.1)$$

can be implemented using bit level systolic arrays. Here we consider the implementation of a 2-port adaptor by modifying the systolic array used to implement eqn. 3.1. Fig. 3.1a illustrates the schematic representation of a 2-port adaptor and the adaptor equations are given below,

$$\begin{aligned} B1 &= A2 + \alpha(A2 - A1) \\ B2 &= A1 + \alpha(A2 - A1) \end{aligned} \qquad (3.2)$$

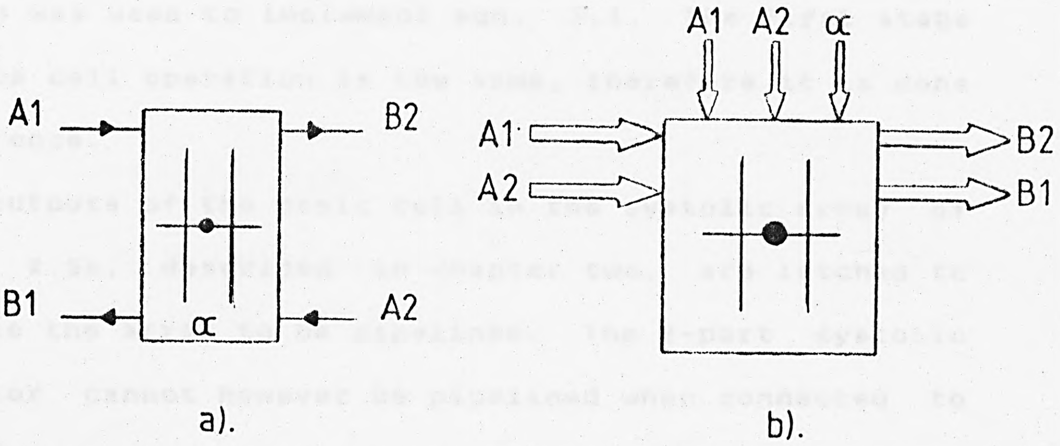


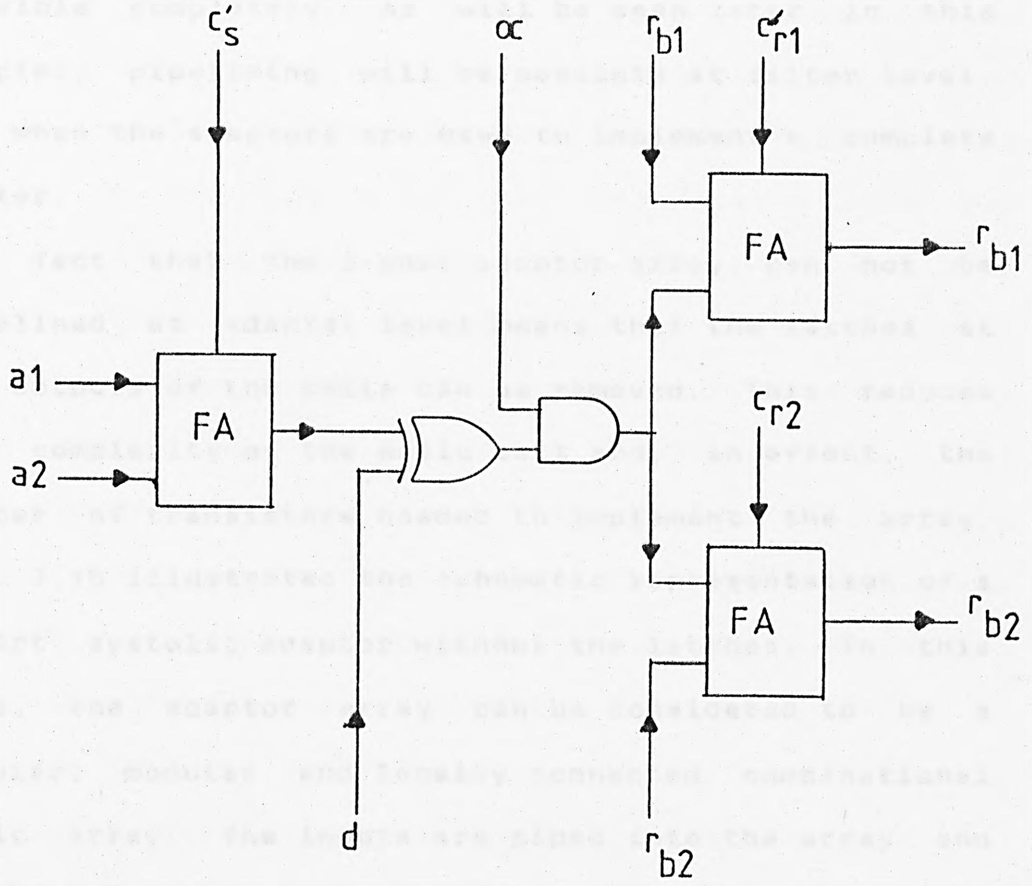
Fig. 3.1- Schematic representation of,

a) a 2-Port adaptor.

b) a 2-Port systolic adaptor.

where Ak's are the inputs, Bk's are the outputs, α is the adaptor coefficient and k=1,2. From eqn. 3.2, there are two ways of implementing the adaptor equations. One obvious way would be to use two systolic arrays of Fig. 2.5, i.e one for each equation. The term α(A2-A1) is common to both equations, therefore by using two systolic arrays this term is evaluated twice which is not necessary. Thus, an alternative method of implementing the adaptor equations would be to use one systolic array with the same cell configuration as in Fig. 2.5, and modify the basic cell. The logic diagram of the basic cell is shown in Fig. 3.2. The basic cell is constructed by overlapping two cells of Fig. 2.5b which was used to implement eqn. 3.1. The first stage of the cell operation is the same, therefore it is done only once.

The outputs of the basic cell in the systolic array of Fig. 2.5a, described in chapter two, are latched to enable the array to be pipelined. The 2-port systolic adaptor cannot however be pipelined when connected to more adaptors to form a complete filter. This is due to the fact that we need both the MSB's and the LSB's of the inputs A1 and A2 at the same clock cycle. Thus, we must wait until one adaptor completes its operation, i.e all the bits of the outputs B1 and B2 are ready, before feeding them as inputs to the next adaptor. This



FA - Full Adder.

34 Gates/Cell.

20 ns Gate Delay

Fig. 3.2 _ Logic diagram of the Basic cell .

of course does not mean that the pipelining is not possible completely. As will be seen later in this chapter, pipelining will be possible at filter level, i.e when the adaptors are used to implement a complete filter.

The fact that the 2-port adaptor array can not be pipelined at adaptor level means that the latches at the outputs of the cells can be removed. This reduces the complexity of the basic cell and, in effect, the number of transistors needed to implement the array. Fig. 3.1b illustrates the schematic representation of a 2-port systolic adaptor without the latches. In this case, the adaptor array can be considered to be a regular, modular and locally connected combinational logic array. The inputs are piped into the array and the output of the cells ripple across the array until the final bits appear on the rigth-hand boundary cells. The structure is no longer a true systolic array since there is no global clock controlling the movement of data in the array, but the concept of systolic array has been used to design the structure. For convenience we will still refer to these structures as systolic arrays.

3.2.1- A Single Board 2-Port Systolic Adaptor

In order to check the correctness of the design, it was decided to construct a single board 2-port systolic adaptor. The basic cells are implemented using EPROM's. Fig. 3.3 shows the photograph of the board. The array is organised as (4x4), i.e 4-bits for the coefficient and 4-bits for the inputs. Switches are used to select the values of the inputs and the coefficient and the outputs are displayed using LED's. The board has been tested fully by connecting it to a Cromemco micro-computer. The values of the coefficient and the inputs were set by the computer and the outputs of the array were fed back into the system. The outputs of the board were tested against the actual values and found to be working successfully. From the layout of the board, we can observe the high regularity of the array which is suitable for VLSI implementations.

The 2-port systolic adaptor has also been simulated by writing a Fortran program. Fortran is not a very efficient language for simulating parallel arrays, such as the systolic array, but it was sufficient to prove that the design was correct. The simulation of the systolic WDF was obtained by multiplexing the 2-port systolic adaptor to the required filter order. The simulation program will be described in more detail in section 3.6 of this chapter.

3.2.2- CMOS Implementation of a 2-port Systolic adaptor

In this section, we estimate how many transistors are required to implement a 2-port systolic adaptor using

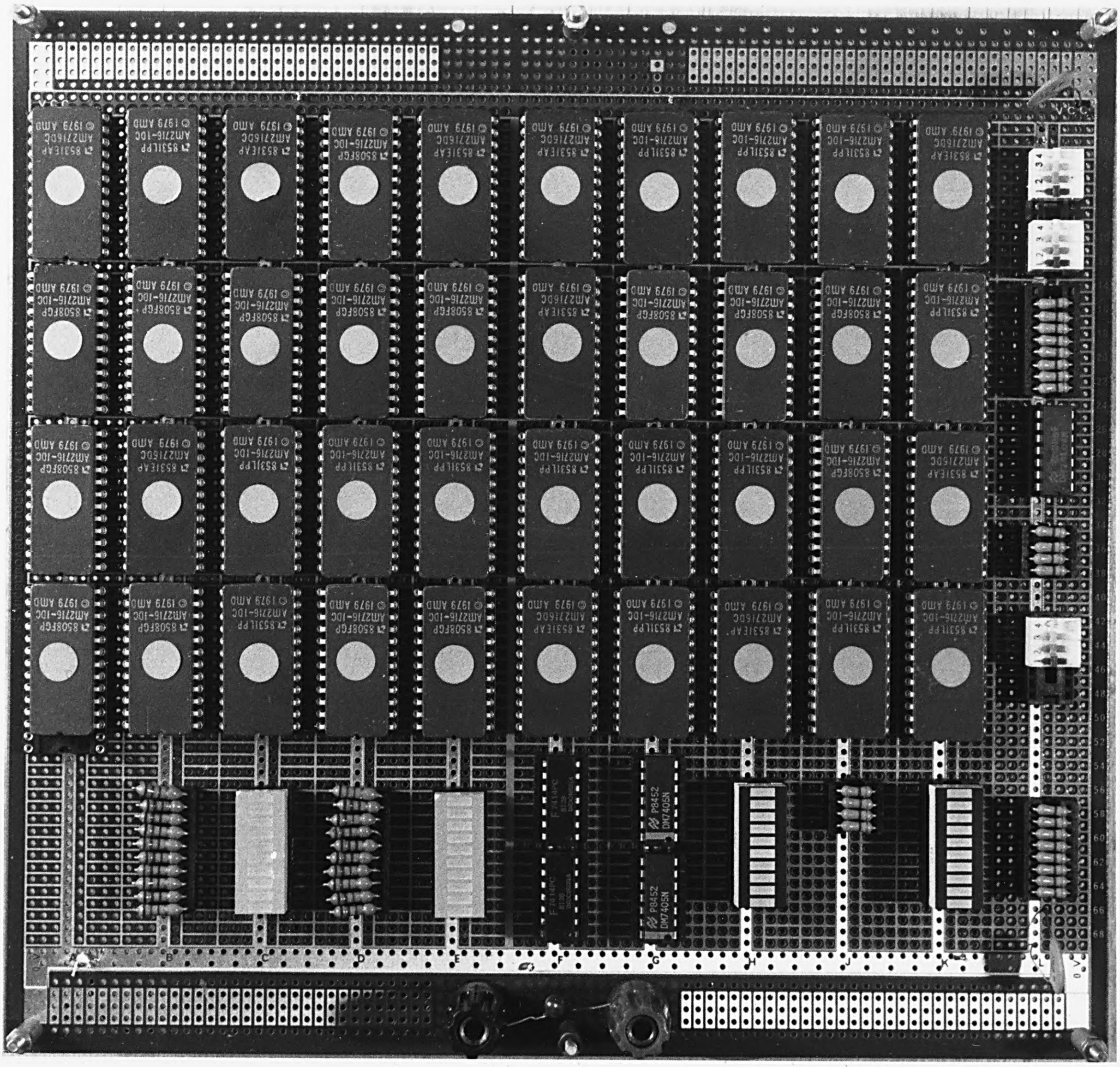


Fig. 3.3 _ Photograph of the 2_port systolic adaptor hardware .

CMOS technology. In general, if n-bits are used to represent the coefficient and m-bits are used to represent the inputs, i.e an (nxm) array, then the maximum number of bits required to represent the outputs B1 and B2 is given as,

$$mnb = (n + m + 2)$$

where mnb is the maximum number of bits required to represent the outputs of the adaptor.The number of cells in the (nxm) array is given by,

$$N_c = n(n + m + 2)$$

and the number of clock cycles required for the adaptor to complete its operation is,

$$N_{cl} = n + (n + m + 2) = 2n + m + 2$$

where one clock cycle is the time requried for one cell to complete its operation. From Fig. 3.2, we need 3-full adders, one 2-input AND gate and one 2-input EX-OR gate to implement a basic cell. In order to estimate the number of transistors required to implement the adaptor it is better if we first calculate the number of gates in terms of 2-input AND gates. A full adder can be implemented using 10 2-input AND gates and an EX-OR gate can be implemented using 3 2-input AND gates. Therefore 34 2-input AND gates are required to implement the 2-port systolic adaptor. In CMOS technology a 2-input AND gate can be implemented using 4 transistors [89]. Therefore we need 136 transistors to implement one cell of the 2-port systolic adaptor. Now we can easily calculate the number of transistors required to implement one complete adaptor.

From Fig. 3.2, the time delay of the adaptor is equal to the time delay of 5 2-input AND gates in cascade. If we consider that the time delay of an AND gate to be 4ns then the time delay of one cell will be equal to 20ns.

This information is shown in tabular form in table 3.1 for some typical values of n and m. From table 3.1, it can be seen that the number of transistors and the time delay of the adaptor are reduced exponentially when the number of bits for the coefficients, i.e n, is reduced. This emphasises the importance of WDFs where filters can be designed with short coefficient wordlengths.

No. of bits		No. of	No. of	No. of	No. of	Delay	
Coef	Signal	Cell	Gate	Tran	Clock	ns	MHZ
4	8	56	1,904	7,616	18	360	2.7
8	8	144	4,896	19,584	26	520	1.9
4	16	88	2,992	11,968	26	520	1.9
8	16	208	7,072	28,288	34	680	1.4

3.3.0- Unit Element WDFs (UEWDFs)

3.3.1- Basic Theory

The first prototype filter we use to derive the WDF is made up of a cascade of transmission lines [29] terminated at both ends by resistances (Fig. 3.4). The line network of Fig. 3.4 can also be viewed as the cascade of N unit elements connected through N 2-port adaptors. In chapter one, we derived the WDF realisation of a unit element (table 1.1). Thus, the WDF representation of the line network of Fig. 3.4 can be obtained by connecting N unit elements using 2-port adaptors. The resulting WDF is shown in Fig. 3.5a and Fig. 3.5b shows the schematic representation of the WDF. It is to be noted that the two delays of $z^{(-1/2)}$ after each adaptor may be combined to form a delay of z^{-1} as shown in Fig. 3.6 [2,82]. This combination has no effect on the magnitude response of the filter and causes a linear shift in the phase response.

The value of the adaptor coefficients, α_k, in the WDF of Fig. 3.6 can be expressed as follows,

$$\alpha_1 = (R_s - Z_1)/(R_s + Z_1) \tag{3.3a}$$

$$\alpha_k = (Z_{k+1} - Z_k)/(Z_{k+1} - Z_k) \tag{3.3b}$$

$$\alpha_N = (R_l - Z_N)/(R_l + Z_N) \tag{3.3c}$$

and $k=2,3,...,N-1$

where Z_k is the kth characteristic impedances of the transmission line filter. Thus, once the transmission

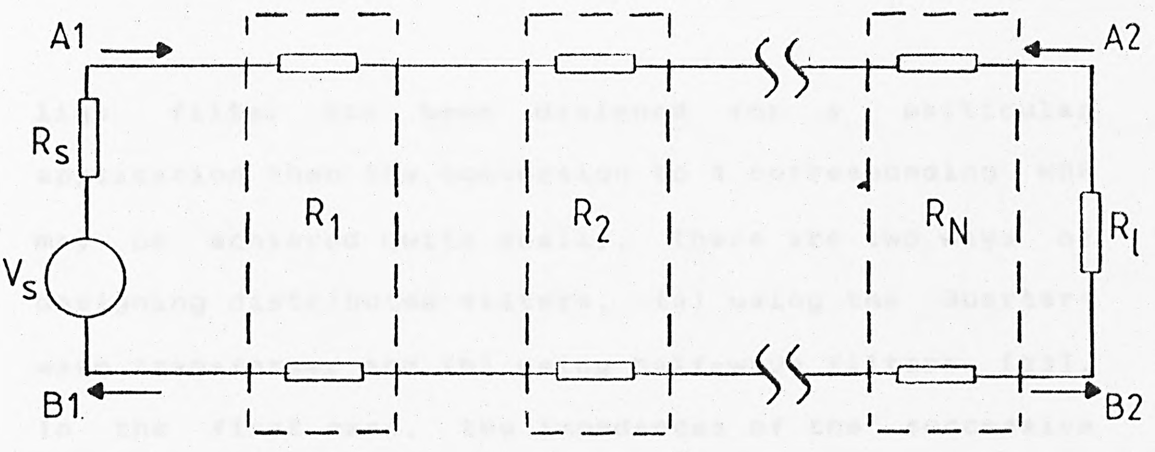


Fig.3.4 _ Cascade Transmission Line filter.

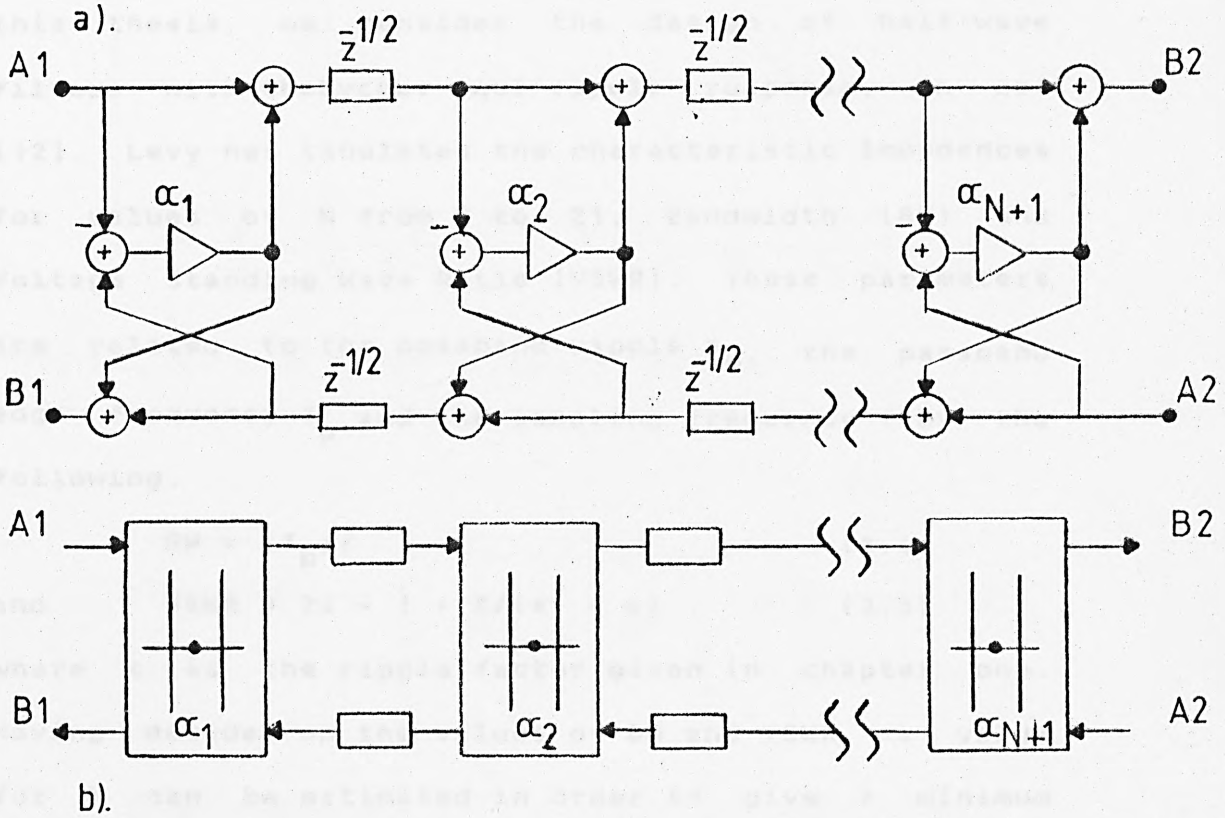


Fig.3.5 _ a). Unit element WDF.

b). Schematic representation.

line filter has been designed for a particular application then the conversion to a corresponding WDF may be achieved quite easily. There are two ways of designing distributed filters, (a) using the Quarter-wave transformer and (b) using half-wave filters [83]. In the first case, the impedances of the successive sections in the line network increase monotonically from unity to an extremely large value, while in the second case, they oscillate about unity. Therefore in this thesis, we consider the design of half-wave filters with Chebychev equi-ripple response. In ref [12], Levy has tabulated the characteristic impedances for values of N from 2 to 21, Bandwidth (BW) and Voltage Standing Wave Ratio (VSWR). These parameters are related to the passband ripple a_p, the passband edge frequency f_p and the sampling frequency f by the following,

$$BW = 4f_p/f \qquad (3.4)$$

and

$$VSWR = 2\varepsilon - 1 + 2\surd(\varepsilon^2 - \varepsilon) \qquad (3.5)$$

where ε is the ripple factor given in chapter one. Having decided on the values of BW and VSWR, a value for N can be estimated in order to give a minimum stopband attenuation of a_s as follows,

$$N > \frac{\log(\varepsilon_s) - \log(\varepsilon-1) + \log 4}{2\log(2\mathrm{Sin}\theta_s/\mathrm{Sin}\theta_p)} \qquad (3.6)$$

where $\varepsilon_s = (10^{a_s/10} - 1)$ and $\theta_s = (\pi f_s/f)$, $\theta_p = (\pi f_p/f)$ and

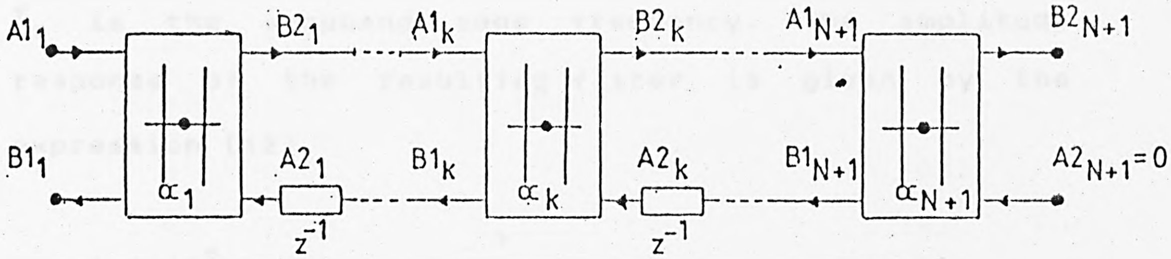


Fig.3.6 _ UEWDF with full delay.

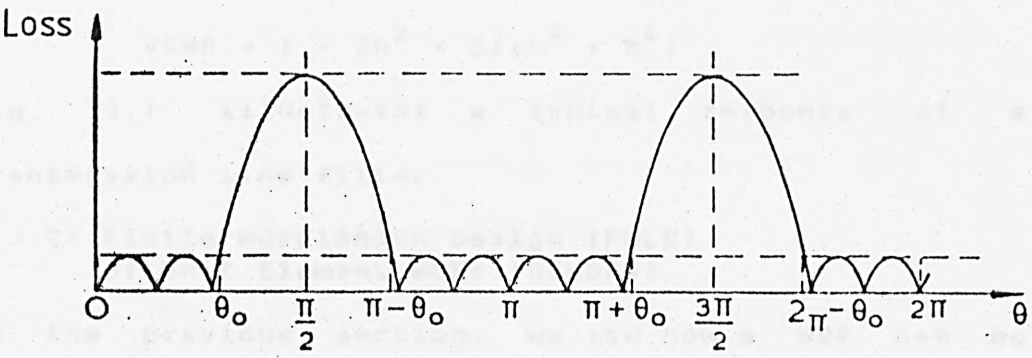


Fig.3.7_ A typical UEWDF frequency response.

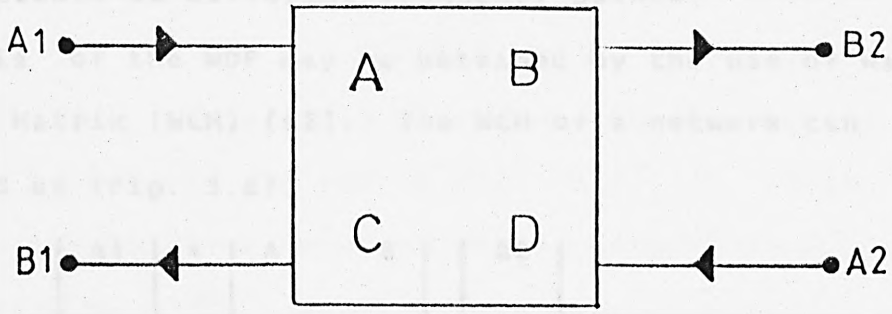


Fig.3.8_ A 2_port network.

f_s is the stopband edge frequency. The amplitude response of the resulting filter is given by the expression [12],

$$|H(s)|^2 = \frac{1}{1 + h^2 T^2_N(\text{Sin}\theta/\text{Sin}\theta_p)} \qquad (3.7)$$

where T_N denotes the Chebychev function of the first kind of degree N and h is related to VSWR by the following,

$$\text{VSWR} = 1 + 2h^2 + 2\surd(h^2 + h^4)$$

Fig. 3.7 illustrates a typical response of a transmission line filter.

3.3.2- Finite Wordlength Design (FWLD) of Unit Element WDFs (UEWDFs)

In the previous section, we saw how a WDF can be derived from a cascade of unit element filters. In this section, we develop a subroutine to evaluate the frequency response of the filter for a given set of coefficients at different frequency points.

Analysis of the WDF may be obtained by the use of Wave Chain Matrix (WCM) [82]. The WCM of a network can be defined as (Fig. 3.8),

$$\begin{vmatrix} A1 \\ \\ B1 \end{vmatrix} = \begin{vmatrix} A & B \\ \\ C & D \end{vmatrix} \begin{vmatrix} B2 \\ \\ A2 \end{vmatrix} \qquad (3.8)$$

From eqn. 3.8, we can obtain the following relationships,

$$A1 = A.B2 + B.A2 \qquad (3.9a)$$

and $\quad B1 = C.B2 + D.A2 \qquad (3.9b)$

If A2 is set equal to zero then,

$$A1 = A.B2 \qquad (3.10a)$$

and $\quad B1 = C.B2 \qquad (3.10b)$

Sustituting eqn. 3.10a into 3.10b, we obtain,

$$(B2/A1) = (1/A) \qquad (3.11a)$$

and $\quad (B1/A1) = (C/A) \qquad (3.11b)$

Eqns 3.11a and 3.11b represent the transfer functions of the network at the outputs B2 and B1 respectively. Now, let us consider the transfer function of the WDF in Fig. 3.6. The difference equations of the kth section in Fig. 3.6 are as follows,

$$B2_k = A1_k + \alpha_k(z^{-1}A2_k - A1_k) \qquad (3.12a)$$

$$B1_k = A2_k + \alpha_k(z^{-1}A2_k - A1_k) \qquad (3.12b)$$

From Appendix A1, we can write $A1_k$ and $B1_k$ in terms of $A2_k$ and $B2_k$ as follows,

$$\begin{vmatrix} A1_k \\ B1_k \end{vmatrix} = K \begin{vmatrix} 1 & -\alpha_k z^{-1} \\ \alpha_k & z^{-1} \end{vmatrix} \begin{vmatrix} B2_k \\ A2_k \end{vmatrix} \qquad (3.13)$$

where $K=(1/1-\alpha_k)$. Eqn. 3.13 represent the ABCD matrix of the kth section in the WDF of Fig. 3.6. In order to evaluate the transfer function of the network at ω_0, z^{-1} is replaced by $e^{-j\omega_0 T}$ and the ABCD matrix in eqn. 3.13 must be calculated for every section in the network. The resulting ABCD matrices are multipled together and the response of the network is evaluated using eqn.

3.11a.

A subroutine has been developed to evaluate the frequency response of the UEWDF of Nth order using the above procedure (Appendix A1). This subroutine plus the error subroutine and the optimization subroutine can now be merged together to form a complete program for the FWLD of UEWDFs. The initial coefficients are obtained from the synthesis subroutine in the program, which is based on ref [84].

A listing of the program is included in Appendix B and a brief description of how the program can be used is given in section 3.6.

3.4.0- Lattice WDFs

3.4.1- Introduction

The second prototype we use to derive the WDF is a doubly terminated lossless symmetric lattice network (Fig. 3.9). It is well known that the number of distinct elements in the two canonic lattice impedances of a symmetric lc-filter is, in general, less than that of the ladder filter [85]. There also exists a lattice equivalent for every symmetric ladder filter. This reduction in the number of elements in a lattice filter results in fewer multipliers if the analogue lattice filter is used to derive a WDF.

Analogue lattice filters are known to have low sensitivity properties in the passband, but the stopband sensitivity is high and tuning of the attenuation poles are required during the manufacturing process. This makes analogue lattice filters impractical to use. This disadvantage, however, does not exist for a Lattice Wave Digital Filter (LTWDF) since digital filters do not require tuning and are not subject to ageing. In fact, due to their simplicity and some other properties, LTWDFs are in some respect the most attractive structures available for use in communication systems [71,72,81].

The realisation of lattice filters depends on how the lattice reactances are realised. In the next section, we briefly look at the basic theory of LTWDFs and how they can be realised.

3.4.2- Basic Theory

Fig. 3.9 illustrates a classical lattice filter terminated between resistances R1 and R2. For our purpose, we assume that the termination resistances are equal, i.e R1=R2=R. Z_1 and Z_2 are the lattice impedences for port one and two respectively.

In Appendix 'C', we derive the realationships between the reflectances, transmittances, scattering parameters and the incident/reflected wave vectors. In view of these realationships, we can consider that the analogue scattering matrix, S, is equal to that of the WDF, S', since the termination resistances are chosen to be equal (eqns A.7). Thus, the two scattering matrices can replace each other without causing any confusion. Also, from ref [86], we can write,

$$S_1 = S_{11} - S_{21} \tag{3.15a}$$

and
$$S_2 = S_{11} + S_{21} \tag{3.15b}$$

where S_1 and S_2 are the reflectances for port one and two respectively and S_{11} and S_{21} are the scattering parameters. From eqn. 3.15 and the symmetrical nature of the lattice network, we can write,

$$S_{11} = S_{22} = (S_1 + S_2)/2 \tag{3.16a}$$

and
$$S_{12} = S_{21} = (S_2 - S_1)/2 \tag{3.16b}$$

Since the lattice filter is symmetric and also Z_1 and

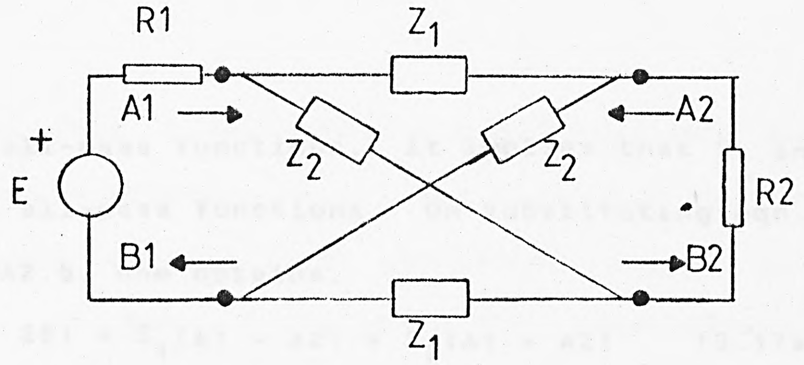


Fig.3.9_ Lattice analogue filter.

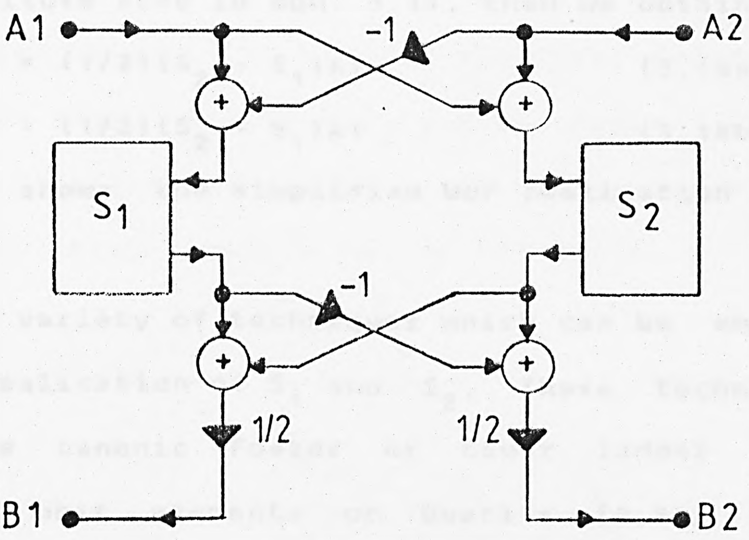


Fig.3.10_ Wave realisation of Fig.3.9.

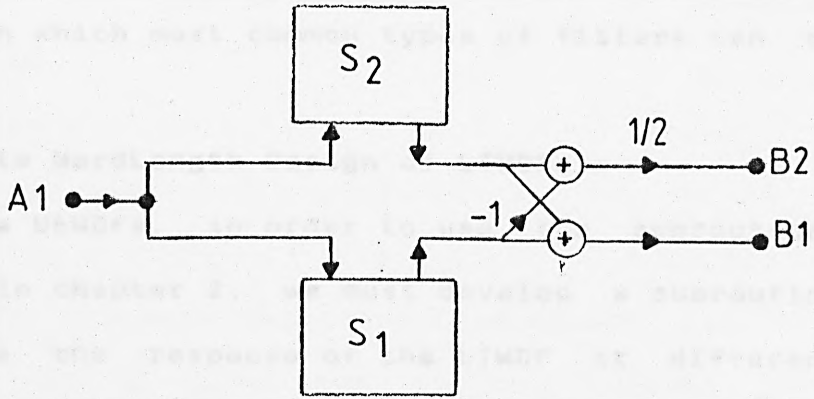


Fig.3.11_ LWDF with A2= 0.

Z_2 are all-pass functions, it implies that S_1 and S_2 are also all-pass functions. On substituting eqn. 3.16 in eqn. A2.5, one obtains,

$$2B1 = S_1(A1 - A2) + S_2(A1 + A2) \quad (3.17a)$$

and

$$2B2 = S_1(A2 - A1) + S_2(A1 + A2) \quad (3.17b)$$

Fig. 3.10 illustrates the WDF realisation of eqn. 3.17. If we substitute A2=0 in eqn. 3.17, then we obtain,

$$B1 = (1/2)(S_2 + S_1)A1 \quad (3.18a)$$

and

$$B2 = (1/2)(S_2 - S_1)A1 \quad (3.18b)$$

Fig. 3.11 shows the simplified WDF realisation of a LTWDF.

There are a variety of techniques which can be employed for WDF realisation of S_1 and S_2. These techniques include the canonic Foster or Cauer ladder [87], cascade of unit elements or Quarl's [2,88], the procedure discussed for the realisation of all-pass transfer functions [2] or using the IVR transformation [27]. Recently Gazsi has presented direct design methods with which most common types of filters can be designed.

3.4.3- Finite WordLength Design of LTWDFs

As with the UEWDFs, in order to use the subroutines developed in chapter 2, we must develop a subroutine to evaluate the response of the LTWDF at different frequency points for a set of coefficients. The LTWDFs designed using Gazsi's method require only 2-port adaptors. Fig. 3.12a shows a Nth order LTWDF realised using 2-port adaptors only. The elementry sections in Fig. 3.12a are the first and second degree all-pass sections of Fig. 3.12b and 3.12c respectively. The first degree all-pass section is a 2-port adaptor which has its A2 inputs equal to z^{-1}B2 (Fig. 3.12b) and the second one is the cascade of two 2-port adaptors as shown in Fig. 3.12c. Let us call the first degree all-pass section (Fig. 3.12b) APS_1 and the second one APS_2 and let $G_1(z)$ and $G_2(z)$ denote their transfer functions respectively. From Appendix A2, we can write,

$$G_1(z) = \frac{B1}{A1} = \frac{z^{-1} - \alpha}{1 - \alpha z^{-1}} \qquad (3.19)$$

and

$$G_2(z) = \frac{B1}{A1} = \frac{z^{-2}+\alpha_2 z^{-1}(\alpha_1-1)-\alpha_1}{1+\alpha_2 z^{-1}(\alpha_1-1)-\alpha_1 z^{-2}} \qquad (3.20)$$

Now we can redraw Fig. 3.12a using APS_1 and APS_2 blocks as shown in Fig. 3.13. The transfer fuction of the LTWDF, G(z), can now be evaluated using the following relationships,

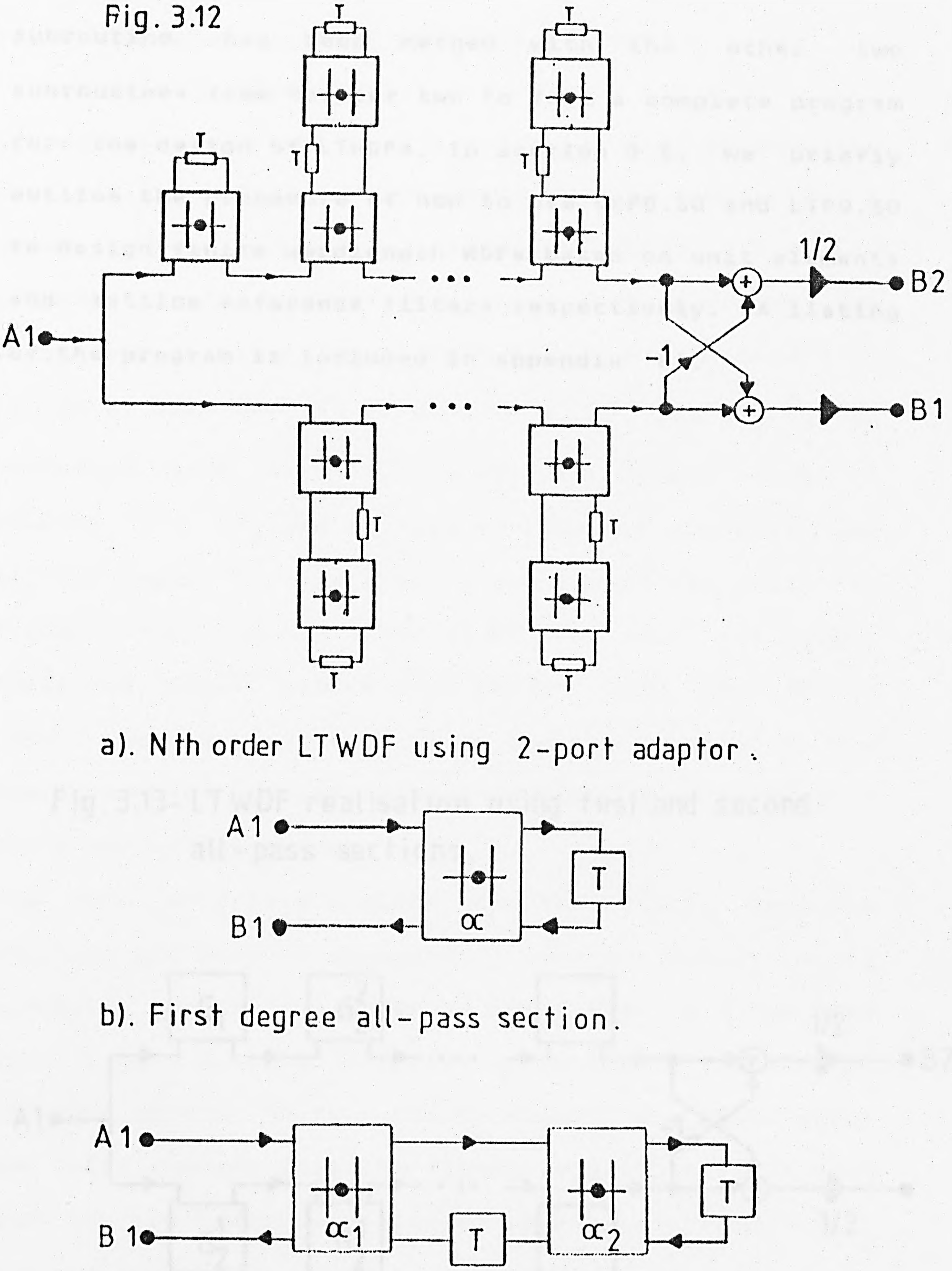
$$G(z) = (1/2)[S_1(z) + S_2(z)] \qquad (3.21)$$

where $$S_1(z) = G_1(z) \prod_k G^k{}_2(z) \quad k=1,3,\ldots \qquad (3.22)$$

and $$S_2(z) = \prod_k G^k{}_2(z) \quad k=2,4,\ldots \qquad (3.23)$$

A subroutine has been developed to evaluate $G(e^{j\omega_k T})$, $k=1,2,\ldots,$ for a given set of coefficients. This

Fig. 3.12



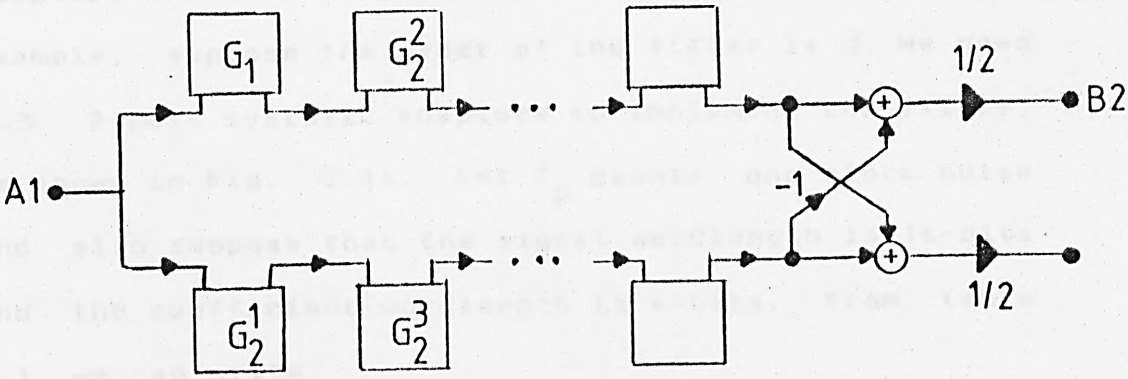
a). Nth order LTWDF using 2-port adaptor.

b). First degree all-pass section.

c). Second degree all-pass section.

subroutine has been merged with the other two subroutines from chapter two to form a complete program for the design of LTWDFs. In section 3.6, we briefly outline the procedure of how to use UEFD.SO and LTFD.SO to design finite wordlength WDFs based on unit elements and lattice reference filters respectively. A listing of the program is included in Appendix 'B'.

Fig.3.13- LTWDF realisation using first and second all-pass sections.



3.5.0- Implementation of complete filters

3.5.1- Unit Element WDFs

As mentioned in section 3.3, the two port adaptor can be cascaded to form a complete filter by connecting a delay between every two adaptors (Fig. 3.6). The introduction of the delays enables us to pipeline the structure at filter level. From Fig. 3.6 and eqns. 3.2, it can be seen that at adaptor '1', $B2_1$ and $B1_1$ can be computed since both $A1_1$ and $A2_1$ are known. Then at adaptor '2', $B2_2$ and $B1_2$ can be computed since $A1_2$ and $A2_2$ are known. At this stage a new sample can enter the filter since the new value of $A2_1$ has been evaluated. Therefore input samples can be fed into the filter every two clock pulses, where one clock pulse is the time taken for one adaptor to operate, irrespective of the order of the filter.

The same principle applies when the 2-port systolic adaptors are used to implement a complete filter. As an example, suppose the order of the filter is 3. We need 4 2-port systolic adaptors to implement the filter, as shown in Fig. 3.14. Let T_p denote one clock pulse and also suppose that the signal wordlength is 16-bits and the coefficient wordlength is 4-bits. From table 3.1, we can write,

$$T_p = 1.9 \text{ MHZ}$$

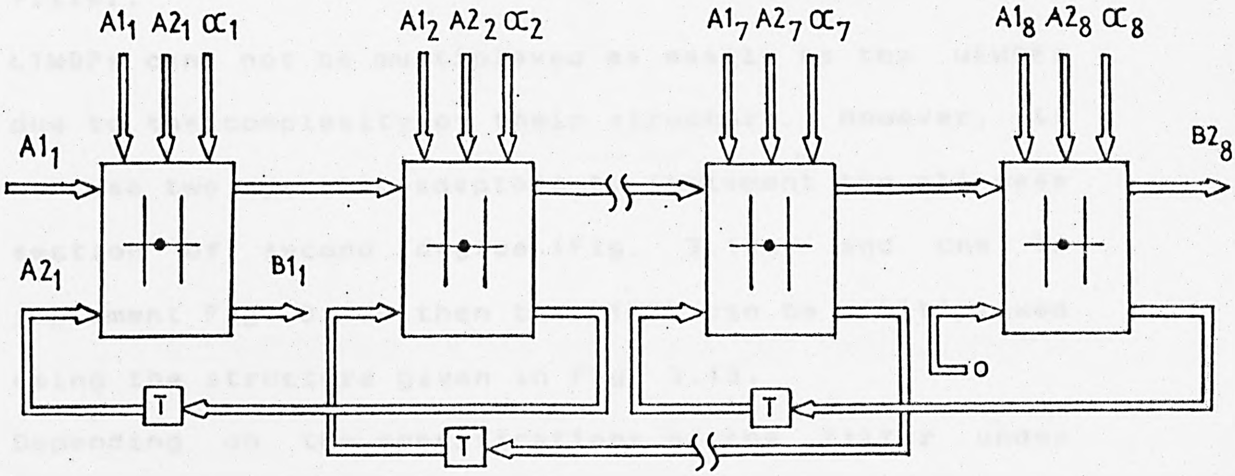
Therefore the sampling frequency of the filter can be up to about 1.0 MHZ, i.e ($T_{p/2}$). The number of transistors needed to implement the complete filter would be,

$$(4 \times 11968) + NT_d \approx 100,000$$

where NT_d denotes the number of transistors needed to implement the delays between the adaptors. Using a CMOS 1.0μ technology [89], this filter can easily be implemented on a single VLSI chip.

The other possible implementation would be to multiplex one 2-port adaptor to the required order. This will of course increase the hardware complexity, since more control is required, and also decreases the throughput of the structure. In refs [39,40], hardware has been designed to implement a unit element filter by

Fig.3.14-7th order UEWDF using 2-port systolic adaptors.



multiplexing a 2-port adaptor. Use of this systolic adaptor in this hardware would reduce the complexity of the design significantly. Fig. 3.15 shows one possible arrangement for the implementation of a multiplexed unit element WDF based on a 2-port systolic adaptor.

3.5.2- **Lattice WDFs**

LTWDFs can also be implemented using only 2-port adaptors. The hardware implementation of LTWDFs are more complex than that of the UEWDFs. When the LTWDFs are implemented in a parallel form (Fig. 3.12a), then the 2-port adaptors can be replaced with the systolic adaptors. The sampling rate of the filter would be again irrespective of the filter order, since the delays between the adaptors enable the structure to be pipelined. The sampling frequency would be $T_{p}/2$ since every two clock pulses a new sample can be fed into the filter.

LTWDFs can not be multiplexed as easily as the UEWDFs due to the complexity of their structure. However, if we use two systolic adaptors to implement the all-pass section of second degree (Fig. 3.12c) and one to implement Fig. 3.12b then the LTWDF can be multiplexed using the structure given in Fig. 3.13.

Depending on the specifications of the filter under consideration, one subclass of LTWDFs can be formed by

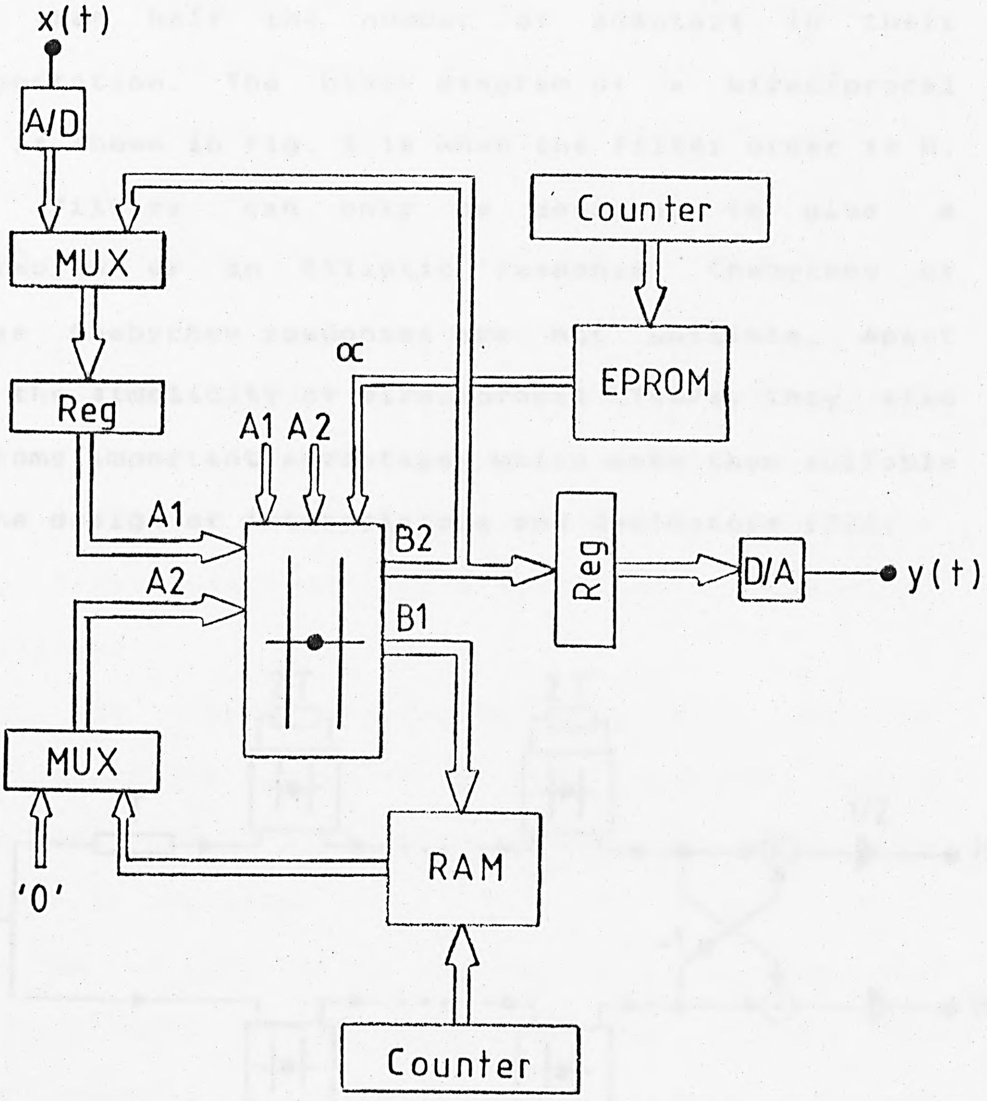


Fig.3.15 _ Multiplexed UEWDF using a 2_port systolic adaptor.

the filters with bireciprocal or self-reciprocal characteristic functions [90,91]. The resulting LTWDFs are much simpler than the usual LTWDFs and they require less than half the number of adaptors in their implementation. The block diagram of a bireciprocal LTWDF is shown in Fig. 3.16 when the filter order is N. These filters can only be designed to give a Butterworth or an Elliptic response. Chebychev or Inverse Chebychev responses are not possible. Apart from the simplicity of bireciprocal LTWDFs, they also have some important advantages which make them suitable for the design of interpolators and decimators [72].

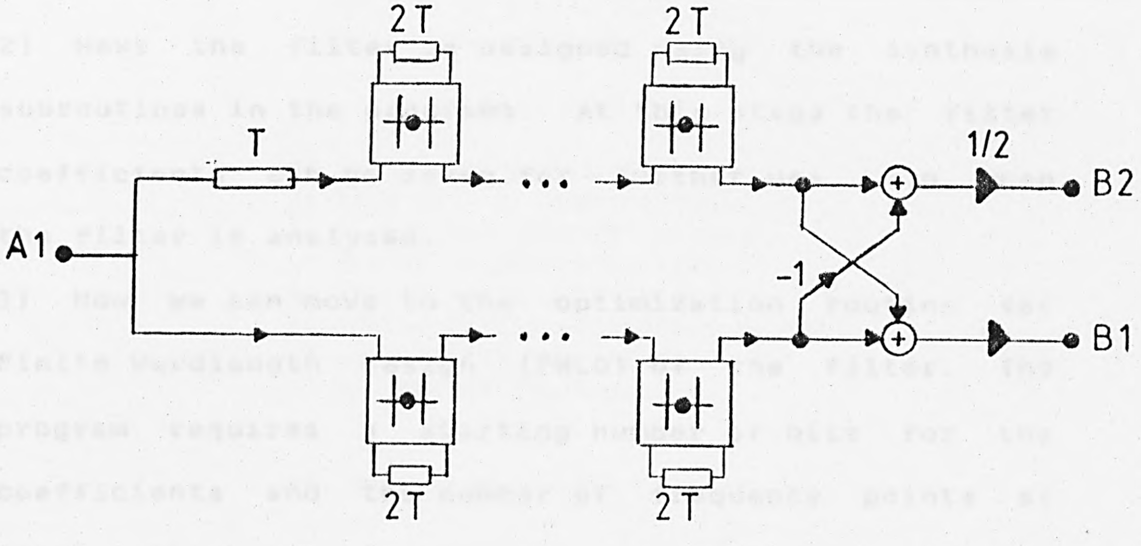


Fig. 3.16- N th order bireciprocal LT WDF.

3.6.0- Description of UEFD.SO and LTFD.SO Programs

Fig. 3.17a and 3.17b show the main menus of the UEFD.SO and LTFD.SO programs respectively. The procedures to design a finite wordlength filter in both cases are roughly the same. The design procedure can be summarized as follows,

1) First the filter specifications are entered, i.e f_p, a_p, f_s and a_s. The sampling frequency is normalised to 1 HZ. The program will estimate the minimum value of N, the filter order, to meet the specifications. In the case of LTFD.SO, this value will be given for three types of filter responses, i.e Butterworth, Chebychev and Elliptic.

2) Next the filter is designed using the synthesis subroutines in the programs. At this stage the filter coefficients can be saved for further use, e.g when the filter is analysed.

3) Now we can move to the optimization routine for Finite Wordlength Design (FWLD) of the filter. The program requires a starting number of bits for the coefficients and the number of frequency points at which the filter will be analysed. The finite wordlength subroutine generates a report of how the search algorithm is progressing by printing the values of a_p and a_s whenever the exploratory moves contain a success. If the algorithm does not manage to design the

Fig. 3.17

```
          PROGRAM TO DESIGN WDFs BASED
        ON UNIT ELEMENT REFERENCE FILTERS
                    MAIN MENU

        1) ENTER SPECIFICATIONS.
        2) DESIGN USING SYNTHESIS.
        3) DESIGN FINITE WORDLENGTH FILTERS.
        4) SAVE FILTER COEFFICIENTS.
        5) READ INITIAL COEFFICIENTS.
        6) END DESIGN PROGRAM.
```

a). Main menu of UEFD.SO program.

```
          PROGRAM TO DESIGN WDFs BASED
           ON LATTICE REFERENCE FILTERS
                    MAIN MENU
        1) ENTER SPECIFICATIONS.
        2) READ INITIAL COEFFICIENTS.
        3) DESIGN BUTTERWORHT FILTERS.
        4) DESIGN CHEBYCHEV FILTERS.
        5) DESIGN ELLIPTIC FILTERS.
        6) DESIGN FINITE WORDLENGTH FILTERS.
        7) SAVE FILTER COEFFICIENTS.
        8) END DESIGN PROGRAM.
```

b). Main menu of LTFD.SO program.

filter for the required number of bits, it increases the number of bits for the coefficients by one and starts all over again. If the specifications are met then the final coefficients, the final values of a_p and a_s and the number of times that the error function subroutine is called will be printed. At this stage the user can either save the coefficients and terminate the program or continue with a smaller number of bits.

There are also other facilities in the programs, such as the user entry of the initial coefficients for the WDF at the terminal. These values can be generated using a random number generator program. This option is useful when we wish to design the WDF directly in the discrete time domain.

When the filter has been designed, the response of the filter can be checked by using a program called ANAWDF. This program has been developed to analyse any type of WDFs based on unit element, lattice and lc-ladder filters with inserted unit elements.The program generates graphical outputs showing the different frequency responses of the filter. Fig 3.18 shows the main menu of ANAWDF. Before the program displays the main menu, the user is asked to enter the type of WDF that is going to be analysed. From Fig. 3.18, the user is able to quantize the coefficients and investigate the effects of quantization on the response of the

```
                PROGRAM TO ANALYSE WDFs
                       MAIN MENU
1) READ WDFs COEFFICIENTS.
2) QUANTIZE COEFFICIENTS.
3) ANALYSE IDEAL FILTER.
4) ANALYSE FILTER WITH QUANTIZED COEFFICIENTS.
5) ANALYSE FILTER WITH FINITE WORDLENGTH COEFFICIENTS.
6) PLOT RESPONSES ON THE SAME AXIS.
7) END ANALYSIS PROGRAM.
```

Fig. 3.18- Main menu of ANAWDF program.

```
               PROGRAM TO SIMULATE WDFs
                       MAIN MENU
1) READ COEFFICIENTS.
2) CHOOSE INPUT SIGNAL.
3) SIMULATE FILTER WITH 2-PORT ADAPTORS.
4) SIMULATE FILTER WITH 2-PORT SYSTOLIC ADAPTORS.
5) SIMULATE FILTER WITH QUANTIZED COEFFICIENTS.
6) SAVE FILTER RESPONSES.
7) END SIMULATION PROGRAM.
```

Fig. 3.19- Main menu of SIMWDF program.

filter. Also the three frequency responses, i.e the ideal case , with quantized coefficients and with FWLD program coefficients, can be plotted on the same axis. The other tool developed to check the filter designs is the simulation program called SIMWDF. Fig. 3.19 illustrates the main menu of SIMWDF. Again before this menu the user is asked to eneter the type of filter under consideration. This program also includes the simulation of systolic WDFs using 2-port and 3-port adaptors. The program allows the user to select three different input signals. These are, an impulse, a step or a sinewave. Also the user can create input signals by adding sinewaves of different frequencies. This type of inputs are useful to check whether the filter can filter out the high frequency signals, assuming that the filter under testing is a lowpass filter. Again the filter coefficients can be quantized and the filter responses can be saved to be plotted using a general purpose plot routine developed by the author.

3.7.0- Analysis and Simulation Results

In this section, we consider the performance of the FWLD programs and the systolic WDFs by means of several examples. The filter examples are designed using UEFD.SO and LTFD.SO programs. The designs are then analysied using ANAWDF program. The results from the ANAWDF program demostrate the effects of finite wordlength on the frequency response of the filters. The simulation of the systolic WDFs are obtained using SIMWDF program.

The procedure with which the filters are designed and checked is described in detail for the first two examples. For the other examples, we only present the results and detailed explanations are not given.

Table 3.2 illustrates 4 filter specifications which are going to be considered in these examples. In all the plots obtained from the ANAWDF program the curves marked

Filter No.	Pass band Edge Freq	Stop band Edge Freq	Max Ripple in Passband	Min Loss in Stopband
1	0.10	0.20	1.0	50
2	0.10	0.30	0.5	55
3	0.25	0.35	1.0	60
4	0.05	0.10	0.5	60

Sampling Frequency is Normalised to 1.0 HZ.

Table 3.2

by '+' denote the frequency responses of the ideal filters, curves marked by 'x' denote the frequency responses of the filters with quantized coefficients and curves marked by 'o' denote the frequency responses of the filters with FWLD program coefficients. When we use the term number of bits, this does not include the sign bit and only represents the number of bits required to express the magnitude of the coefficients.

One measure of the speed of the design programs is to count the number of times that the error function subroutine is called, i.e to find what the variable NFUNC is. It has been exprienced that, for many cases, this number can be kept low if the number of bits for the coefficients is reduced gradually. This means that the initial coefficients are first quantized to a number of bits which is not very small, say 10 or 8-bits. If the algorithm succeeds in designing the filter for this then the number of bits is reduced by one and the program is run again.

3.7.1- Example 3.1

In the first example, we consider the design of a UEWDF for filter No.1. Using option (1) in the main menu of the UEFD.SO program (Fig. 3.17a) the specifications of the filter are entered. A 7th order UEWDF is required to meet the specifications. The filter was then designed using the synthesis subroutine, i.e option

(2). These coefficients were saved and then fed into the optimization routine using option (3). The optimization routine managed to minimize the number of bits for the coefficients to 6. The synthesis coefficients and the FWLD program coefficients are listed below,

	Synthesis Coeff	FWLD Coeff 6-bits
1)	-0.7481353	-0.6875
2)	0.9203713	0.890625
3)	-0.9425427	-0.921875
4)	0.9457231	0.9375
5)	-0.9457231	-0.9375
6)	0.9425427	0.9375
7)	-0.9203713	-0.890625
8)	0.7481353	0.640625

Fig. 3.20 shows the frequency responses of the filter for (a) ideal filter, (b) with 6-bit coefficients and (c) with 6-bit coefficients from FWLD program. As it can be seen from Fig. 3.20, when the coefficients are quantized to 6-bits the passband ripple of the filter is more than 1.0 DB. The passband ripple of the filter with FWLD program coefficients however is less than 1.0 DB. The stopband attenuation in both cases is more than 50 DB. The number of times that the error subroutine was called is 1352, i.e NFUNC=1352.

Next we look at the simulation of the systolic UEWDF. The synthesis coefficients were fed into the SIMWDF program using option (1) in Fig. 3.19. The input signal was chosen to be an impulse. Fig. 3.21a illustrates the impulse response of the systolic filter against the

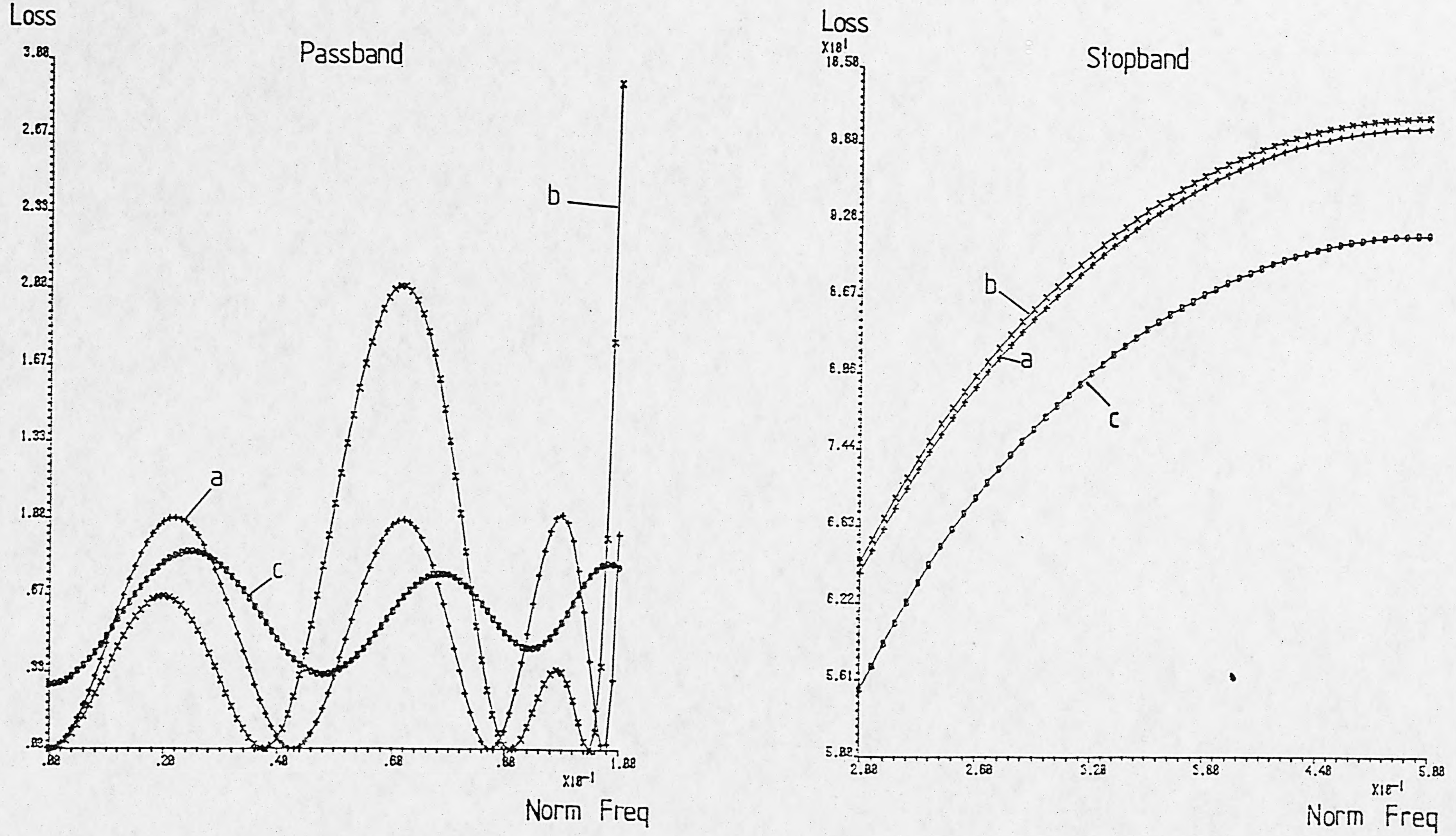


Fig. 3.20 _ Frequency responses for example 3.1.

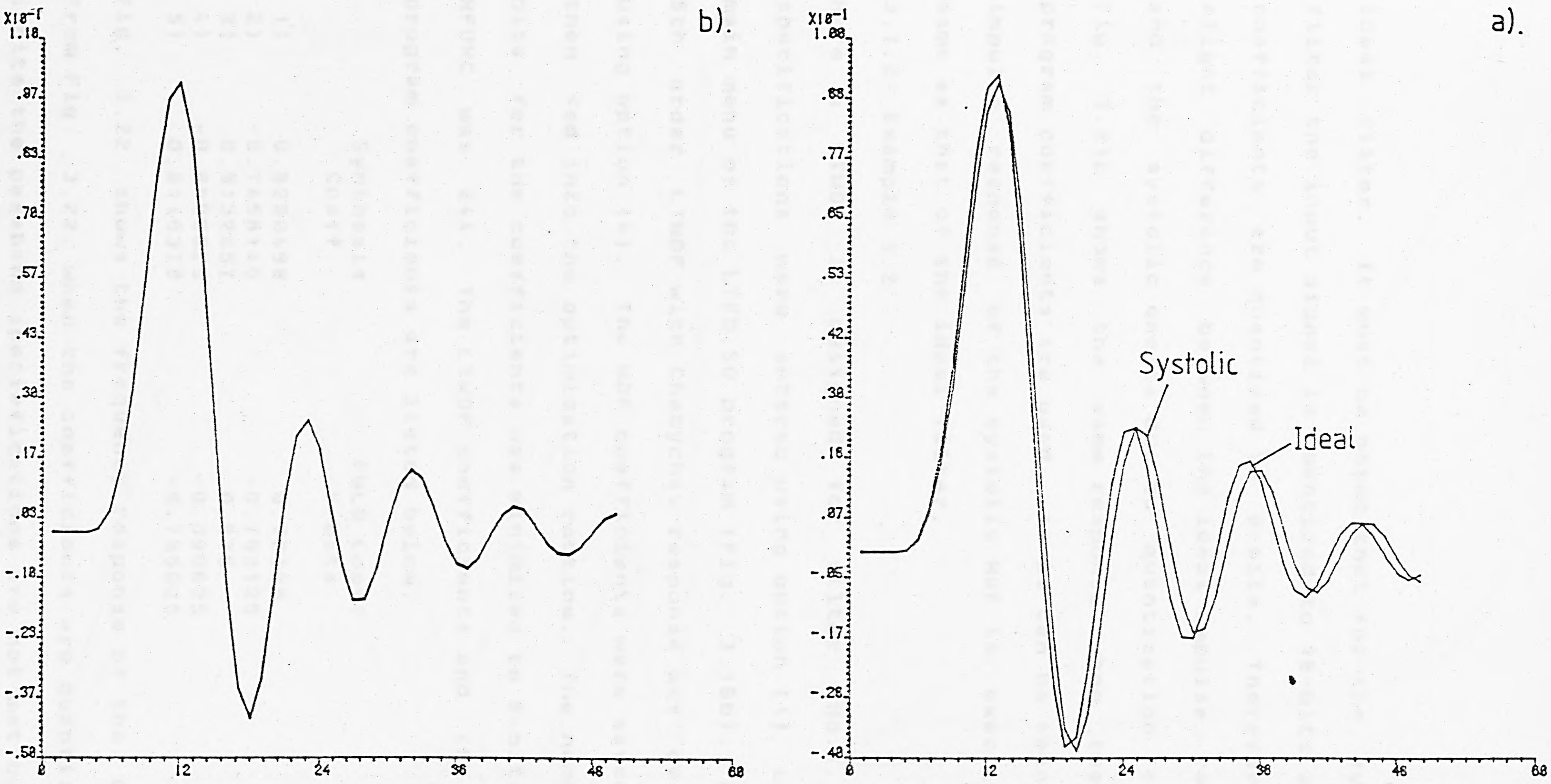


Fig. 3.21_ Impulse responses for example 3.1 .
a). With synthesis coefficients .
b). With FWLD program coefficients .

ideal filter. It must be noted that for the systolic filter the input signal is quantized to 16-bits and the coefficients are quantized to 8-bits. Therefore the slight difference between the ideal impulse response and the systolic one is due to quantization errors. Fig. 3.21b shows the same response when the FWLD program coefficients are used. As it can be seen, the impulse response of the systolic WDF is exactly the same as that of the ideal filter.

3.7.2- Example 3.2

Here a LTWDF is designed for filter No.1. The specifications were entered using option (1) in the main menu of the LTFD.SO program (Fig. 3.16b). Next a 5th order LTWDF with Chebychev response was designed using option (4). The WDF coefficients were saved and then fed into the optimization routine. The number of bits for the coefficients was minimized to 6-bits and NFUNC was 244. The LTWDF coefficients and the FWLD program coefficients are listed below,

	Synthesis Coeff	FWLD Coeff 6-bits
1)	0.8280498	0.78125
2)	-0.7458148	-0.703125
3)	0.9132857	0.875
4)	-0.8999829	-0.890625
5)	0.8110378	-0.765625

Fig. 3.22 shows the frequency response of the LTWDF. From Fig. 3.22, when the coefficients are quantized to 6-bits the passband speciefications are not met by the

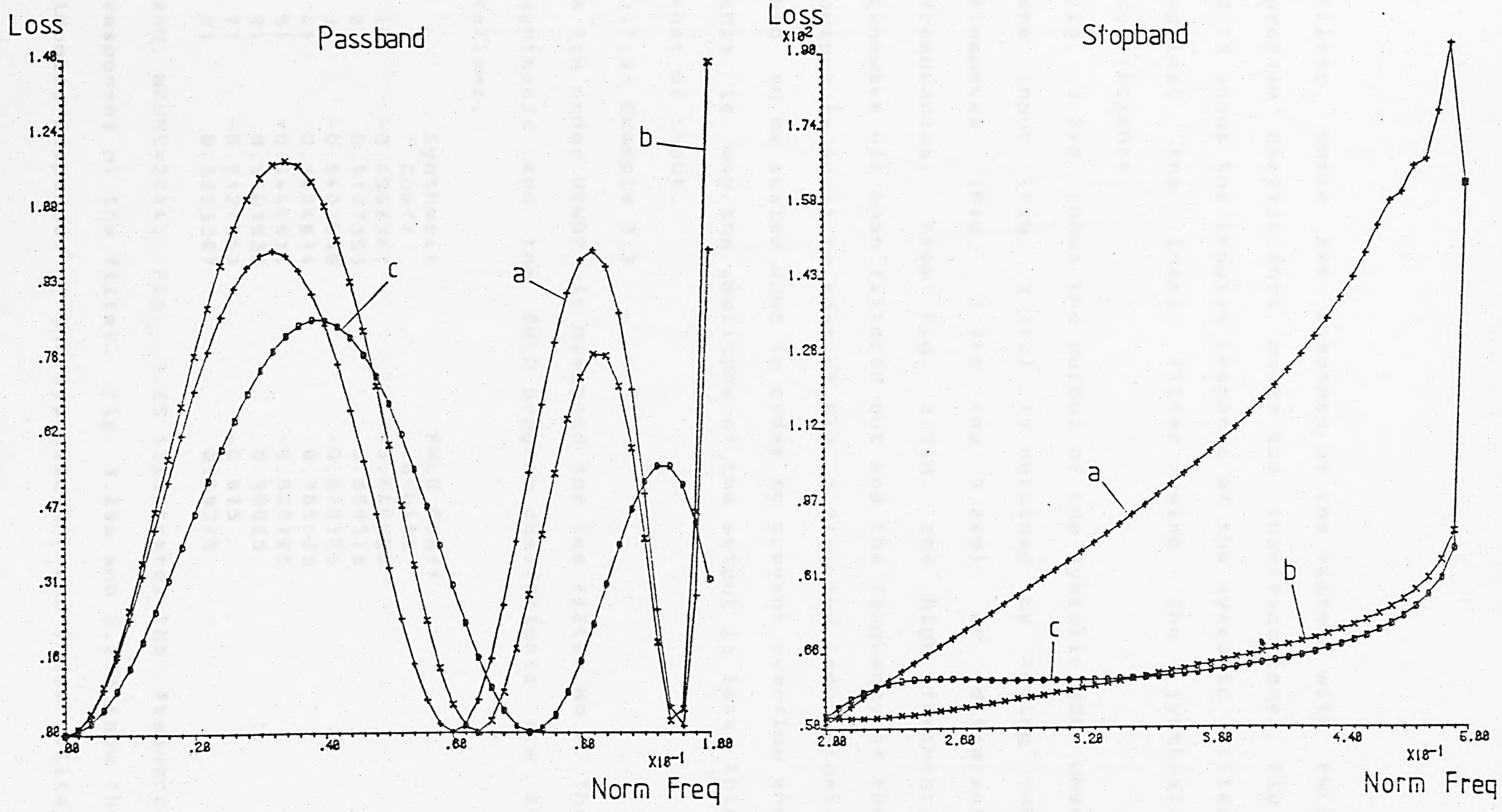


Fig. 3.22 _ Frequency responses for example 3.2 .

filter, while the response of the filter with FWLD program coefficients meets the specifications. Fig. 3.23 shows the impulse response of the systolic filter against the ideal filter using the synthesis coefficients.

Fig. 3.24d shows the output of the systolic WDF when the input (Fig. 3.24c) is obtained by adding two sinewaves (Fig. 3.24a and 3.24b) of different frequencies. From Fig. 3.24d, the high frequency sinewave has been filtered out and the frequency of the output is equal to that of Fig. 3.24a. The input signal had to be scaled down in order to prevent overflow and this is why the amplitude of the output is less than that of input.

3.7.3- Example 3.3

A 7th order UEWDF was designed for the filter No.2. The synthesis and the FWLD program coefficients are as follows,

	Synthesis Coeff	FWLD Coeff 6-bits
1)	-0.6963367	-0.640625
2)	0.9127353	0.859375
3)	-0.9403536	-0.828125
4)	0.9444614	0.765625
5)	-0.9444614	-0.828125
6)	0.9403536	0.90625
7)	-0.9127353	-0.875
8)	0.6963367	0.59375

and NFUNC=2744. Fig. 3.25 illustrates the frequency responses of the filter. Fig. 3.26a and 3.26b show the impulse responses of the systolic WDF and ideal filter

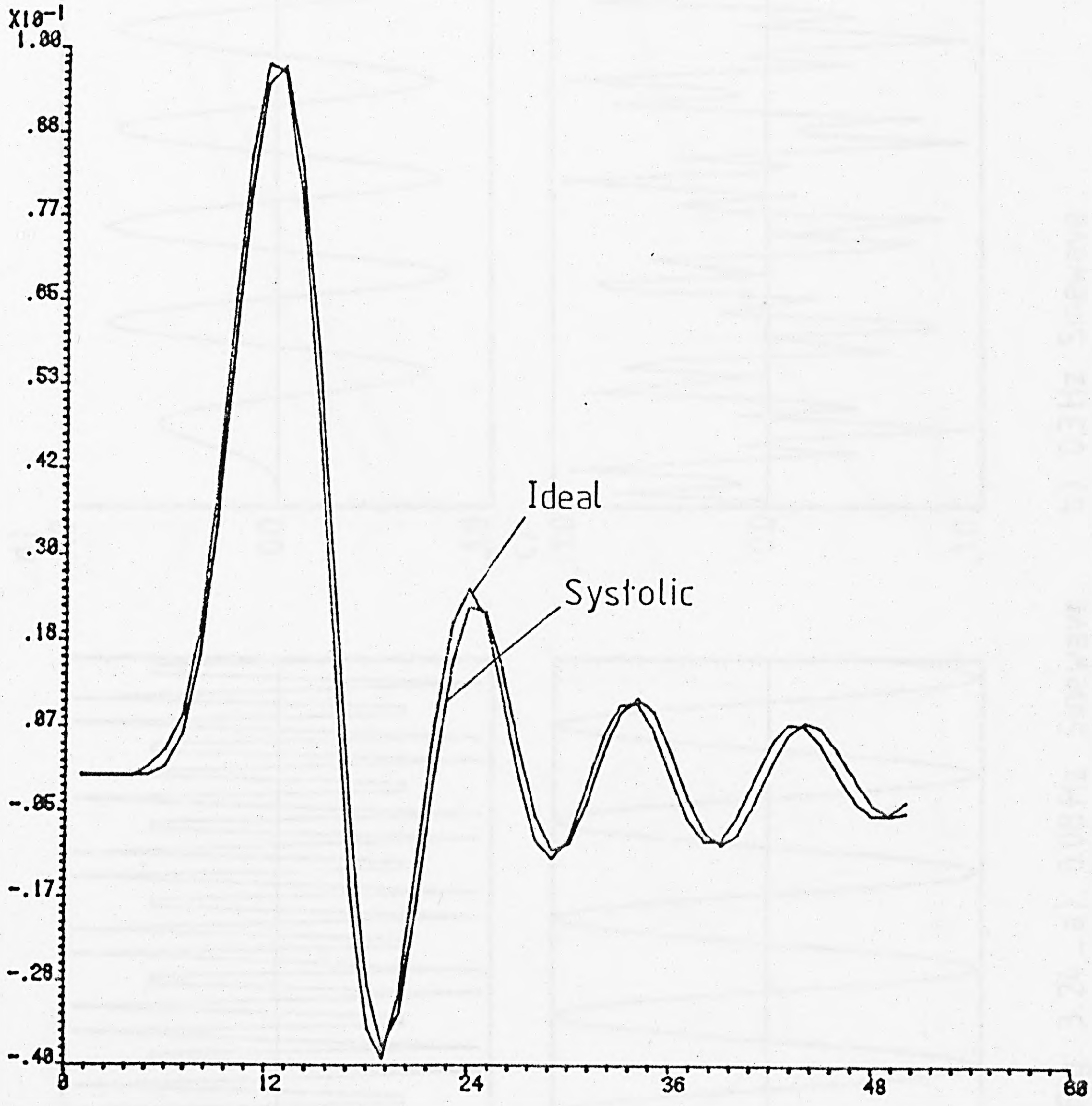


Fig. 3.23 _ Impulse responses for example 3.2 with synthesis coefficients.

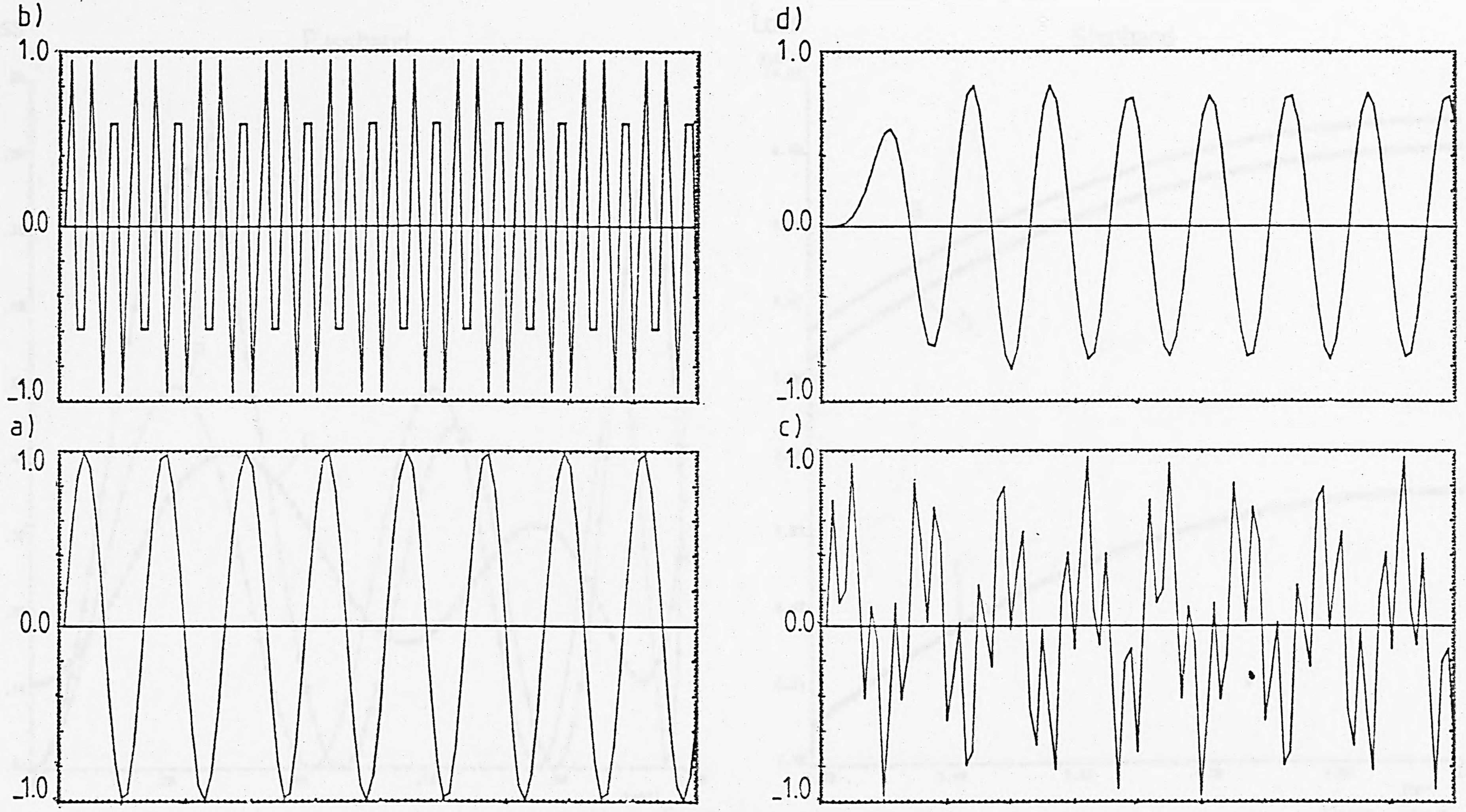


Fig. 3.24 _ a). 0.08 Hz Sinewave. b). 0.3 Hz Sinewave.
c). Input. d). Output.

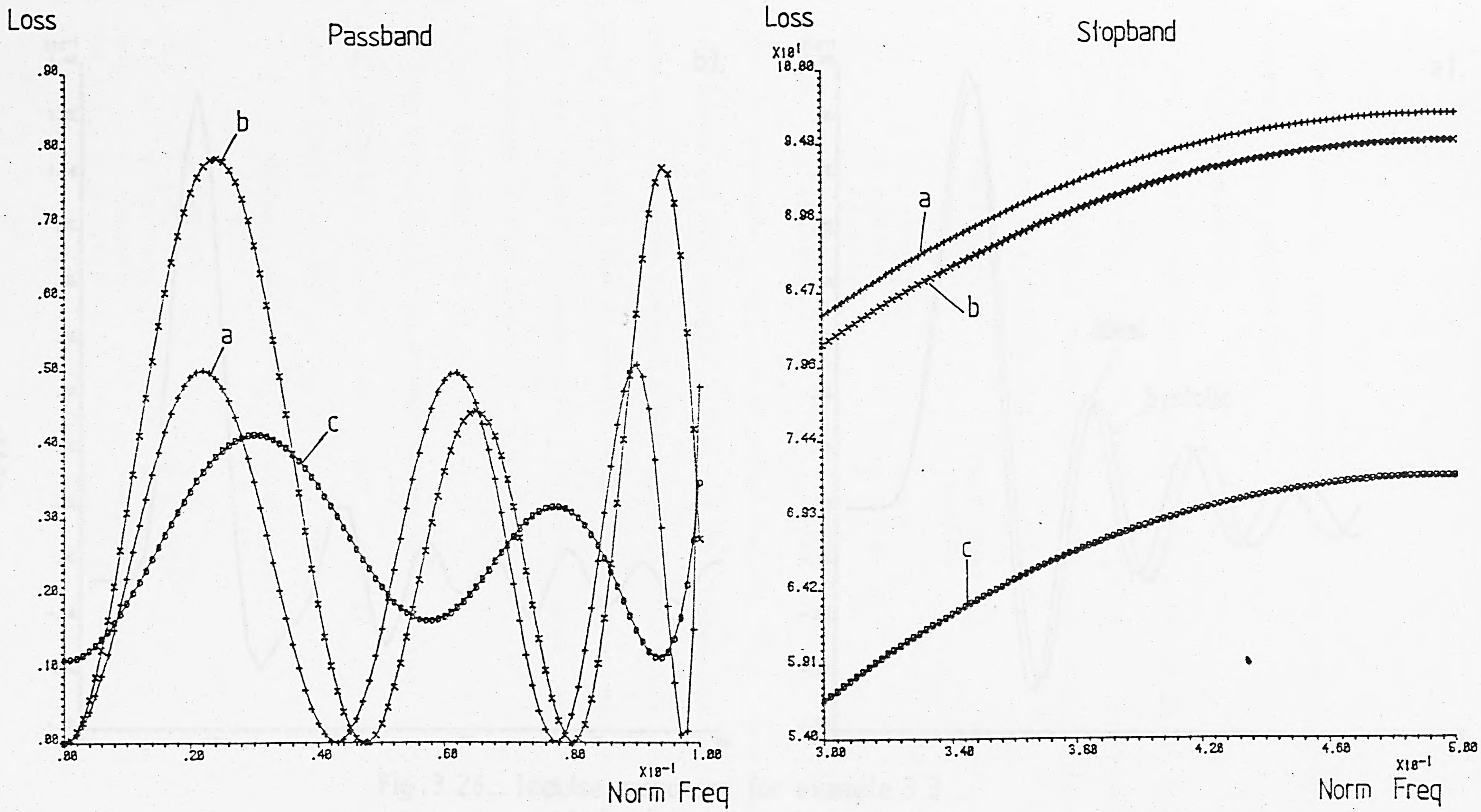


Fig. 3.25_ Frequency responses for example 3.3 .

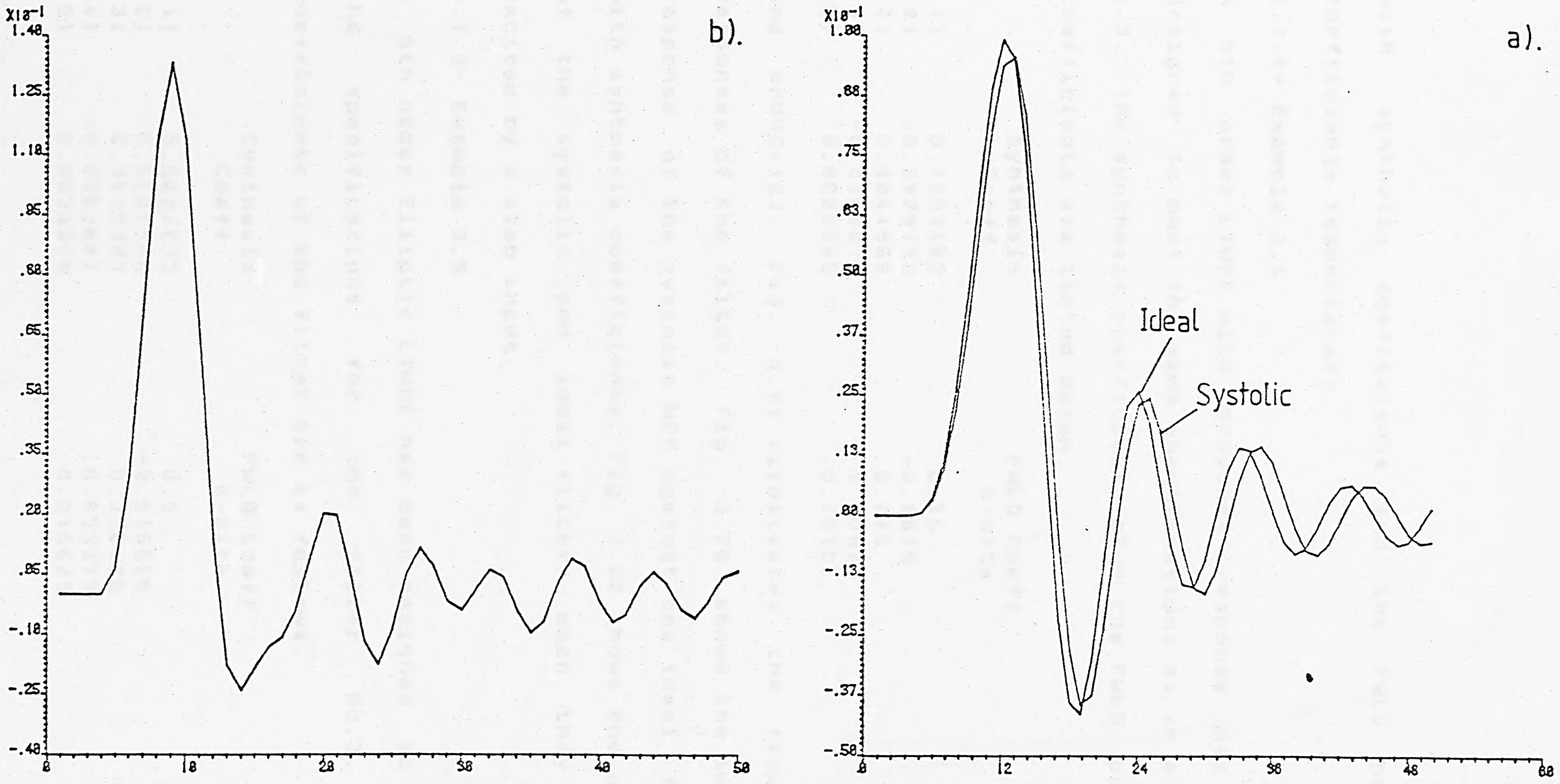


Fig.3.26_ Impulse responses for example 3.3 .
a). With synthesis coefficients.
b). With FWLD program coefficients .

with synthesis coefficients and the FWLD program coefficients respectively.

3.7.4- Example 3.4

A 5th order LTWDF with Chebychev response has been designed to meet the same specifications as in example 3.3. The synthesis coefficients plus the FWLD program coefficients are listed below,

	Synthesis Coeff	FWLD Coeff 6-bits
1)	0.7893492	0.75
2)	-0.6929720	-0.6875
3)	0.9041569	0.875
4)	-0.8769015	-0.90625
5)	0.8028565	0.78125

and NFUNC=103. Fig. 3.27 illustrates the frequency responses of the filter. Fig. 3.28 shows the impulse response of the systolic WDF against the ideal filter with synthesis coefficients. Fig. 3.29 shows the output of the systolic and ideal filters when they were excited by a step input.

3.7.5- Example 3.5

A 5th order Elliptic LTWDF has been designed to meet the specifications for the filter No.3. The coefficients of the filter are as follows,

	Synthesis Coeff	FWLD Coeff 6-bits
1)	0.5224633	0.5
2)	-0.5207588	-0.515625
3)	0.3570393	0.328125
4)	-0.8562697	-0.859375
5)	0.0039918	0.015625

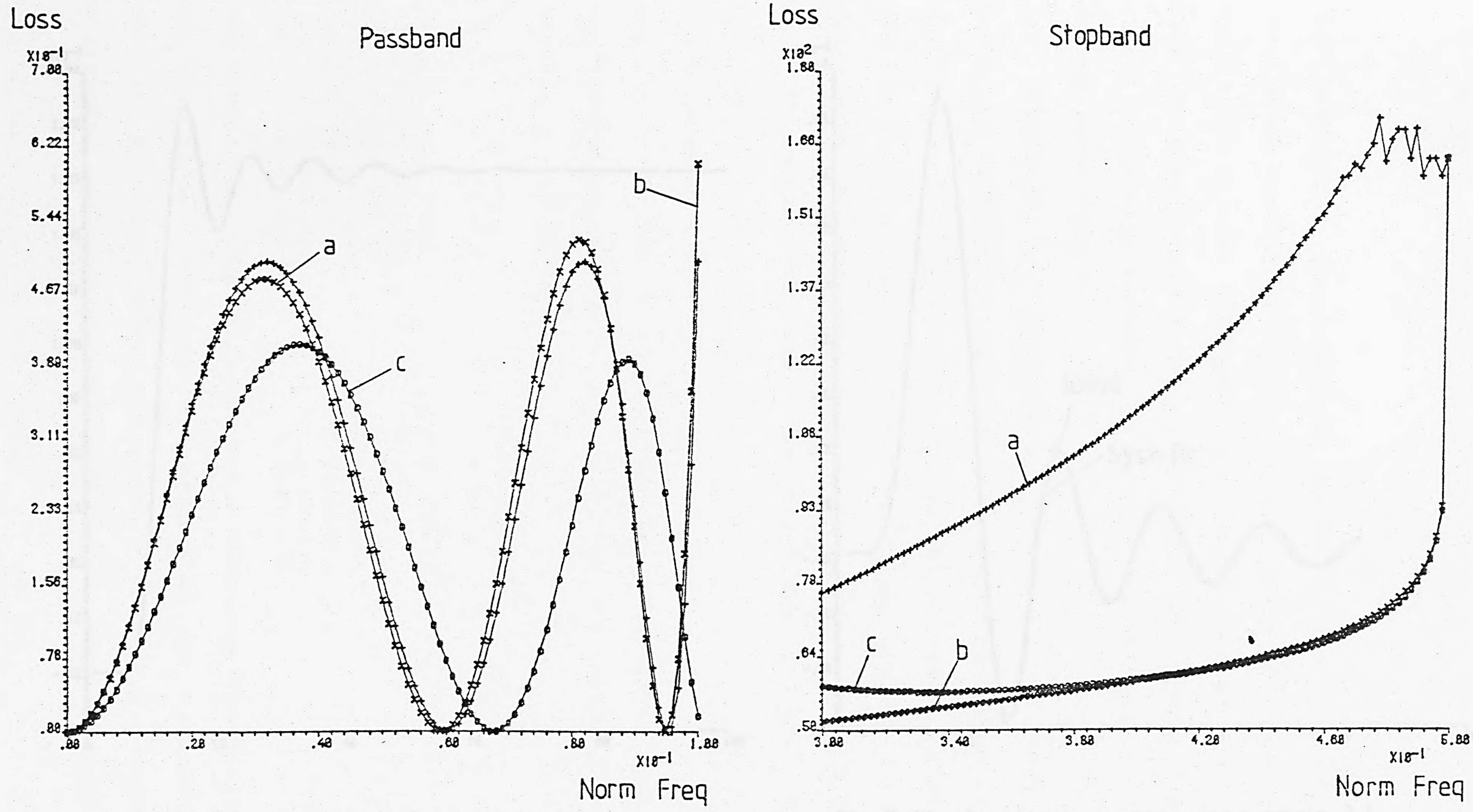


Fig. 3.27_ Frequency responses for example 3.4 .

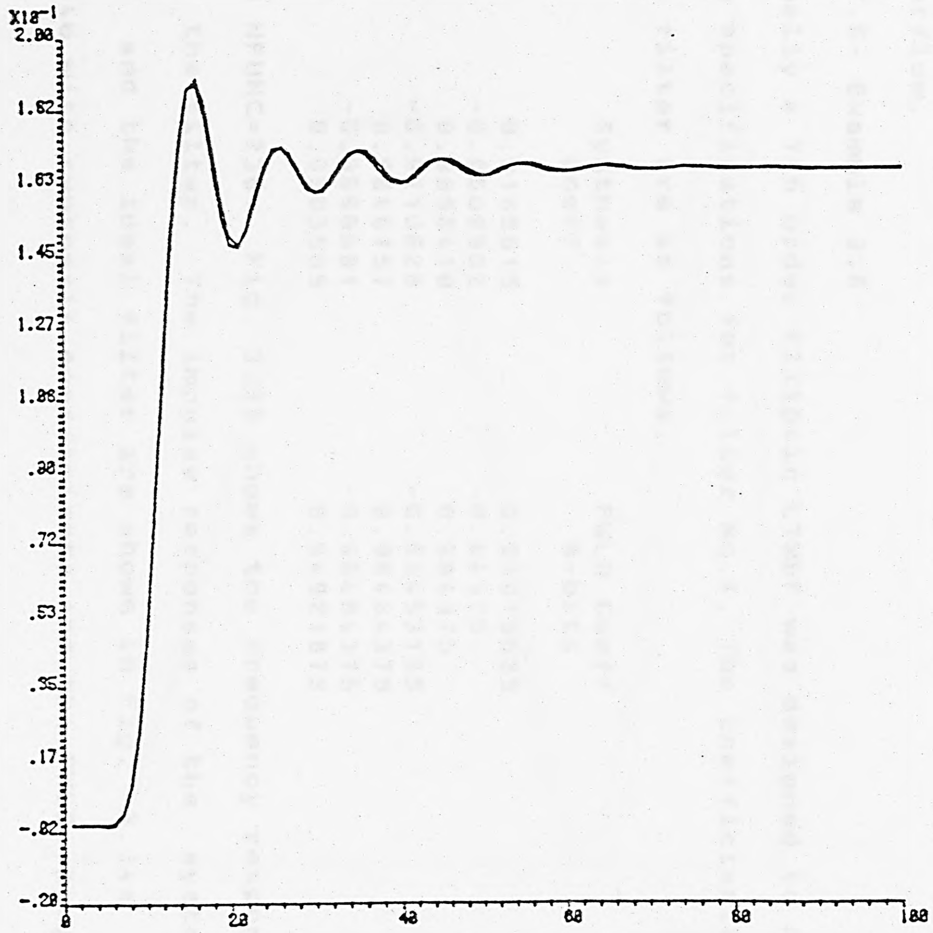


Fig.3.29_Step responses for example 3.4 .

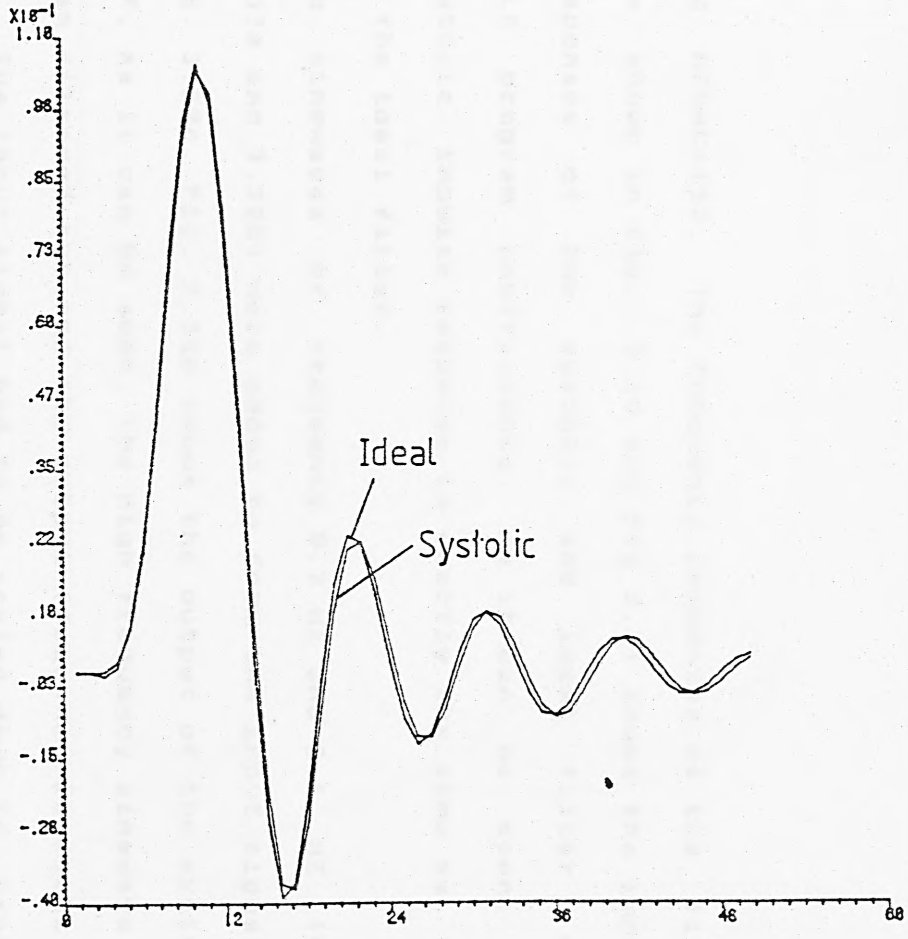


Fig.3.28_Impulse responses for example 3.4 with synthesis coefficients .

and NFUNC=152. The frequency responses of the filter are shown in Fig. 3.30 and Fig 3.31 shows the impulse responses of the systolic and ideal filter with FWLD program coefficients. As it can be seen, the systolic impulse response is exactly the same as that of the ideal filter.

Two sinewaves of frequency 0.2 HZ and 0.4 HZ (Fig. 3.32a and 3.32b) were added to form the input signal of Fig. 3.32c. Fig. 3.32d shows the output of the systolic WDF. As it can be seen, the high frequency sinewave has been filtered out and the output frequency is about 0.2 HZ. The input signal had to be scaled down to prevent overflow.

3.7.6- Example 3.6

Finally a 7th order Elliptic LTWDF was designed to meet the specifications for filter No.4. The coefficients of the filter are as follows,

	Synthesis Coeff	FWLD Coeff 8-bits
1)	0.9165015	0.91015625
2)	-0.8609902	-0.84375
3)	0.9858410	0.984375
4)	-0.9110628	-0.89453125
5)	0.9646757	0.96484375
6)	-0.9698581	-0.96484375
7)	0.9503505	0.94921875

and NFUNC=236. Fig. 3.33 shows the frequency responses of the filter. The impulse responses of the systolic WDF and the ideal filter are shown in Fig. 3.34a and 3.34b with synthesis coefficients and the FWLD program

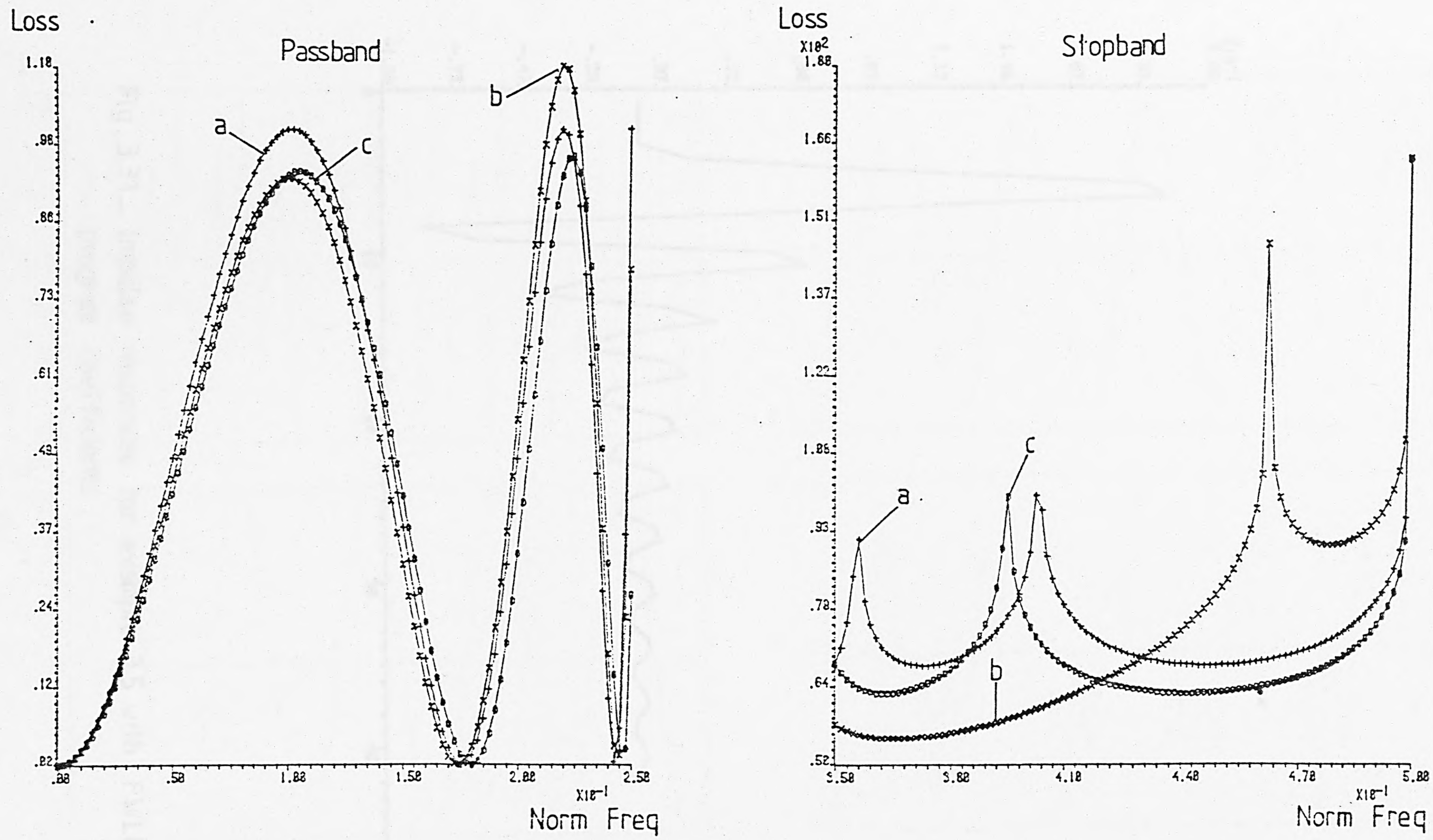


Fig. 3.30 _ Frequency responses for example 3.5 .

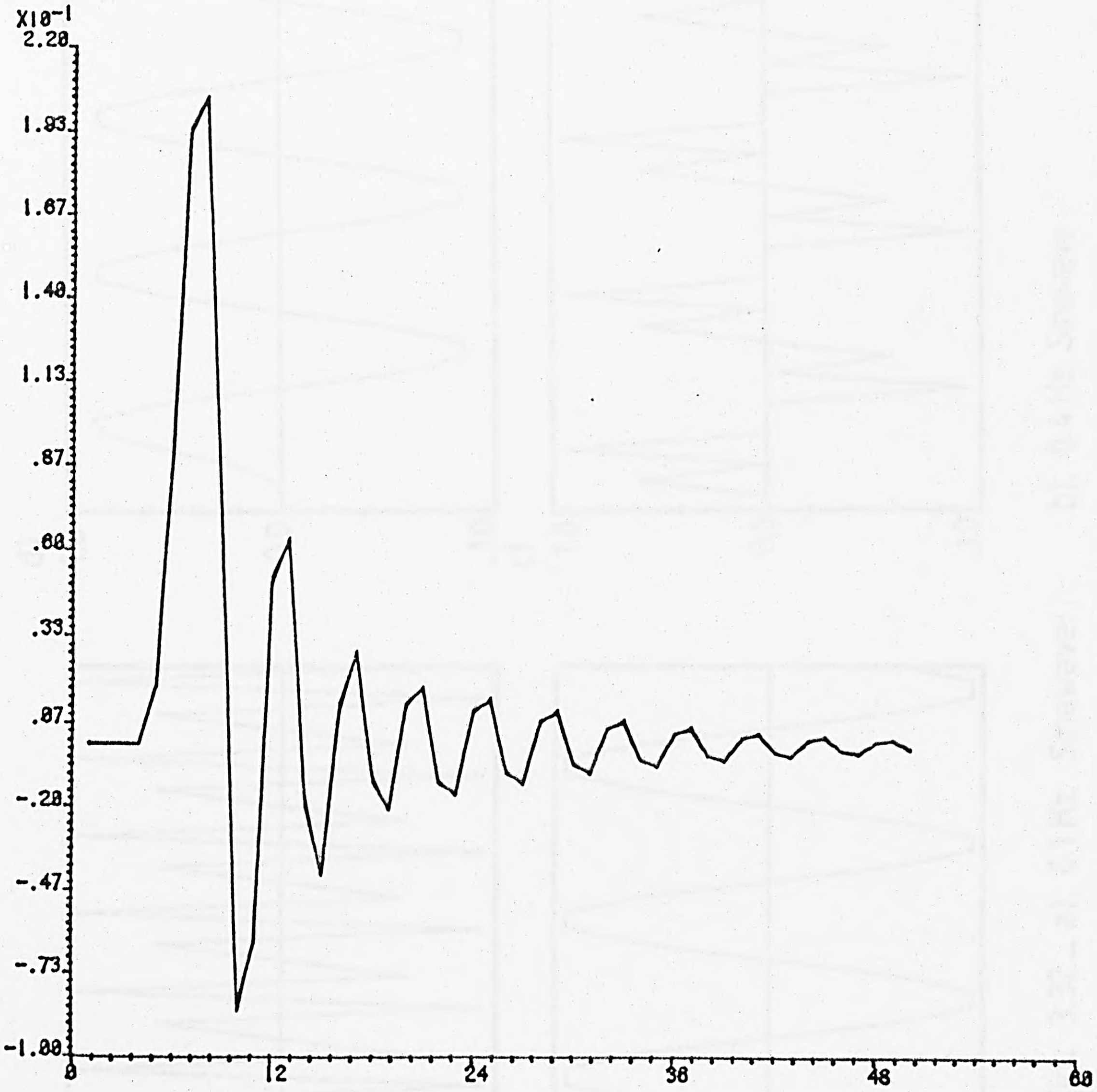


Fig. 3.31 _ Impulse responses for example 3.5 with FWLD program coefficients .

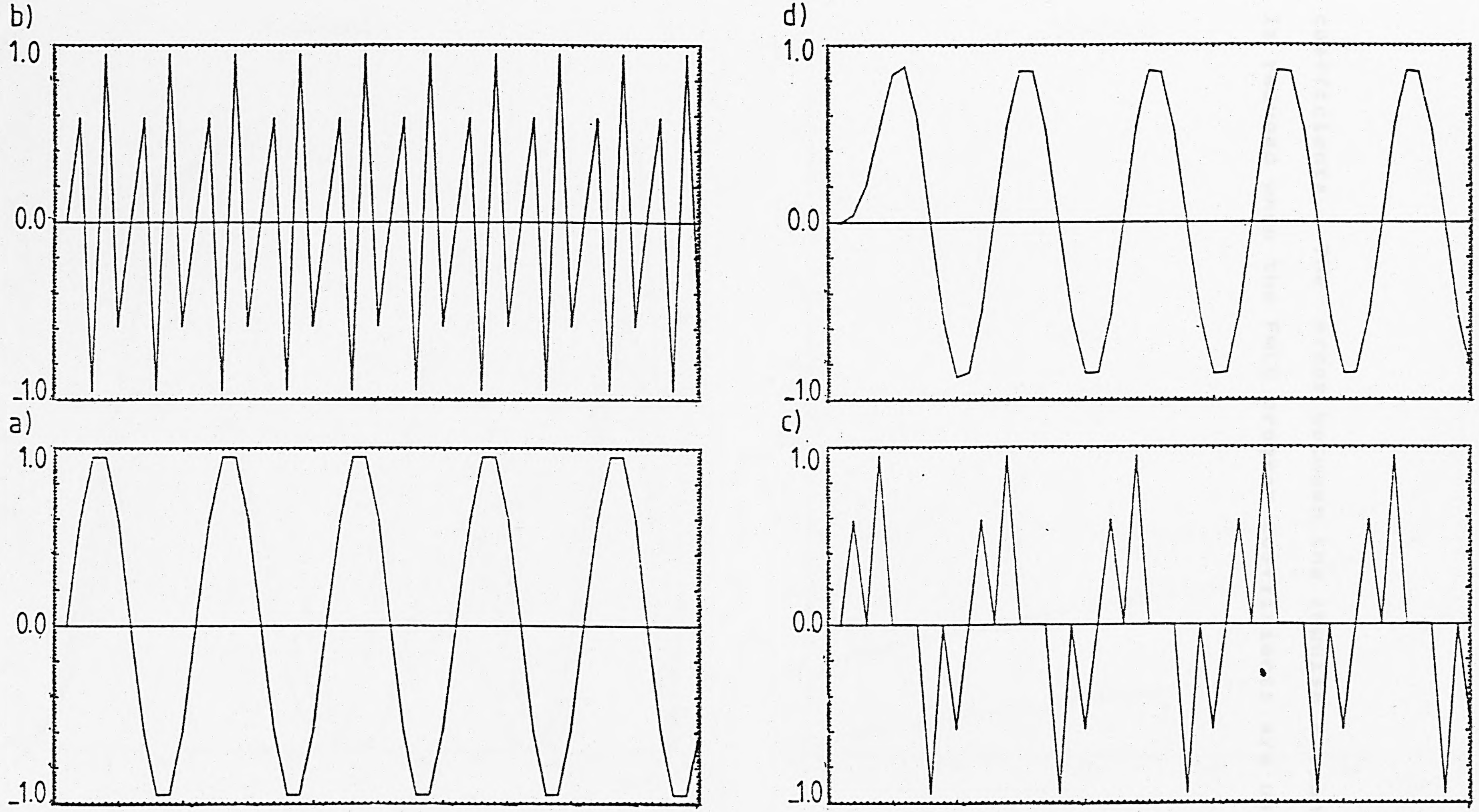


Fig. 3.32 _ a). 0.1 Hz Sinewave . b). 0.4 Hz Sinewave .
c). Input . d). Output .

coefficients. The error between the impulse responses is reduced when the FWLD program coefficients are used.

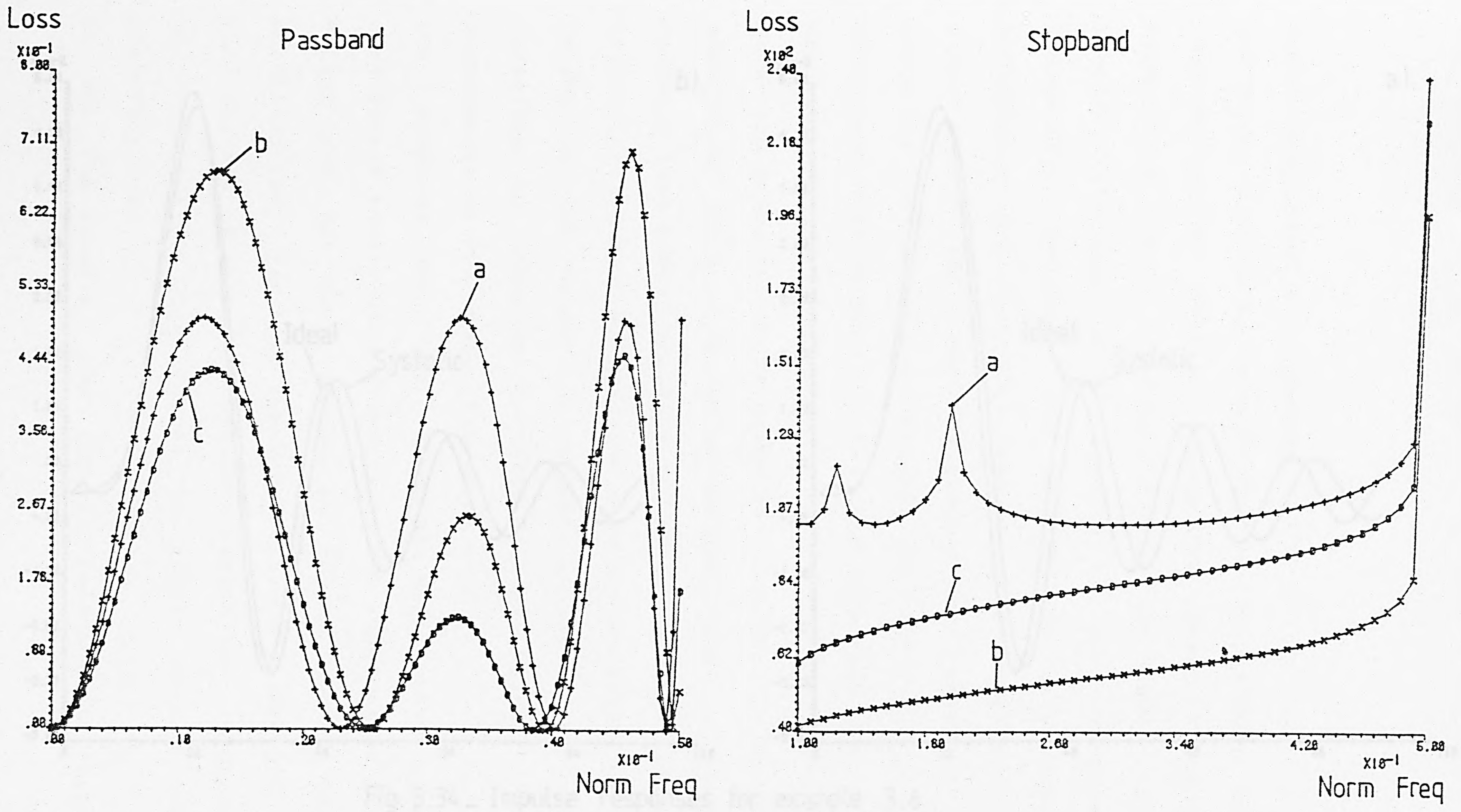


Fig. 3.33 _ Frequency responses for example 3.6 .

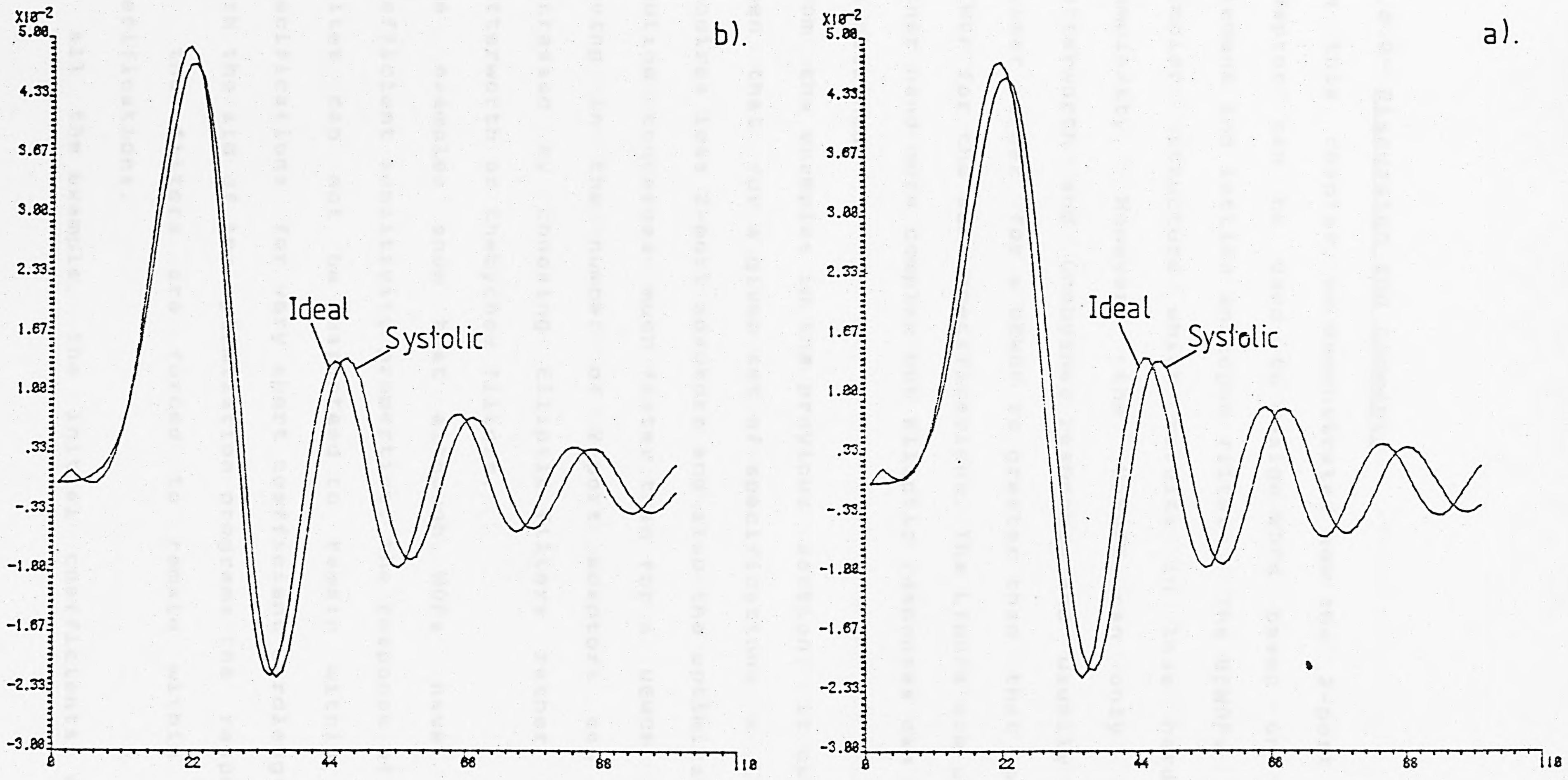


Fig. 3.34_ Impulse responses for example 3.6 .
a).With synthesis coefficients .
b).With FWLD program coefficients .

3.8.0- Discussion and Comments

In this chapter, we demonstrated how the 2-port WDF adaptor can be used to design WDFs based on unit element and lattice analogue filters. The UEWDFs have simpler structure which results in less hardware complexity. However, the UEWDFs can only have Butterworth and Chebyshev responses and usually the filter order for a UEWDF is greater than that of a LTWDF for the same specifications. The LTWDFs are on the other hand more complex but Elliptic responses can also be obtained.

From the examples in the previous section, it can be seen that for a given set of specifications a LTWDF requires less 2-port adaptors and also the optimization routine converges much faster than for a UEWDF. The saving in the number of 2-port adaptors can be increased by choosing Elliptic filters rather than Butterworth or Chebychev filters.

The examples show that although WDFs have low coefficient sensitivity properties, the response of the filter can not be guaranteed to remain within the specifications for very short coefficient wordlengths. With the aid of the optimization programs the response of the filters are forced to remain within the specifications.

In all the example, the initial coefficients were obtained using the synthesis routines within the programs. The package has the options with which the user can enter the initial coefficients and move straight to the optimization part of the programs to dsign the WDF directly in the discrete-time domain.

The 2-port systolic array in this chapter is designed by interconnecting regular and simple one-bit processor cells. The interconnections are localized to the nearest neighbouring cells. This is becoming more important as we move towards the VLSI implementation of digital signal processing hardware. The single board prototype 2-port systolic adaptor has been tested completly to verify the correctness of the design. The results from the simulation of the systolic WDFs also show good agreement with the ideal filter responses for different types of filters and excitations.

CHAPTER FOUR

LC-LADDER WDFS

4.1.0- Introduction

In the previous chapter, we considered the design and VLSI implementation of WDFs which only need 2-port adaptors in their realisations. In general, it is possible to model LC-ladder analogue filters using 2 and 3-port adaptors. In this chapter, we consider the design and systolic implementation of LC-ladder WDFs (LCWDFs). We start by developing systolic arrays for the realisation of 3-port serial and 3-port parallel adaptors using the systolic arrays developed in chapter two. Also the basic cell of the universal systolic array in chapter two is modified to implement a universal systolic WDF adaptor. This adaptor can be programmed to realise 2-port, 3-port parallel and 3-port serial adaptors. In section 4.2.4, we estimate the number of transistors required to implement the adaptors in CMOS technology.

The design of LCWDFs is considered in section 4.3.0. There are many different methods by which a WDF can be derived from a LC-ladder reference filter. These methods are briefly reviewed and in section 4.3.2, we describe in detail the use of Kuroda Transforms [92] in the design of LCWDFs with inserted unit elements. A subroutine is developed to derive a WDF from a LC-ladder reference filter. The reference filter can be designed using either filter design tables or explicit formulas. In section 4.3.3, we develop a subroutine for analysis of LCWDFs which is then merged with the two subroutines developed in chapter two to form a complete program for the design of finite wordlength LCWDFs.

In section 4.4.0, we briefly consider the hardware implementation of LCWDFs using 3-port systolic adaptors. The SIMWDF and ANAWDF programs are used to simulate and analyse the LCWDFs and in section 4.5.0 we outline how the LCFD.SO program can be used to design finite wordlength LCWDFs. Finally in section 4.6.0, we present a number of filter examples to illustrate the performance of the design program and the systolic LCWDFs.

4.2.0- <u>3-Port Systolic Adaptors</u>

In Chapter two, we saw how equations of the form,

$$R2 = P + Z1(X1 - X2) + W1(X3 - X4) \quad (4.1a)$$

and
$$R2 = P - Z1(X1 + X2 + X3) \quad (4.1b)$$

can be implemented using bit-level systolic arrays. In this section, we illustrate how these basic systolic arrays can be modified to implement 3-port adaptors.

4.2.1- 3-port Parallel Systolic Adaptors

The symbolic representation of a 3-port parallel adaptor is shown in Fig. 4.1 and the adaptor equations are given below,

$$Bk = A0 - Ak \qquad K=1,2,3 \quad (4.2a)$$

where
$$A0 = \Sigma\, \alpha_i Ai \qquad i=1,2,3 \quad (4.2b)$$

and
$$2 = \alpha_1 + \alpha_2 + \alpha_3 \quad (4.2c)$$

Using eqn. 4.2c one of the adaptor coefficients, say α_3, can be expressed in terms of the other two coefficients. Therefore,

$$\alpha_3 = 2 - \alpha_1 - \alpha_2 \quad (4.3)$$

Expanding eqn 4.2b and substituting eqn 4.3 for α_3, we obtain,

$$A0 = \alpha_1 A1 + \alpha_2 A2 + (2-\alpha_1-\alpha_2)A3$$

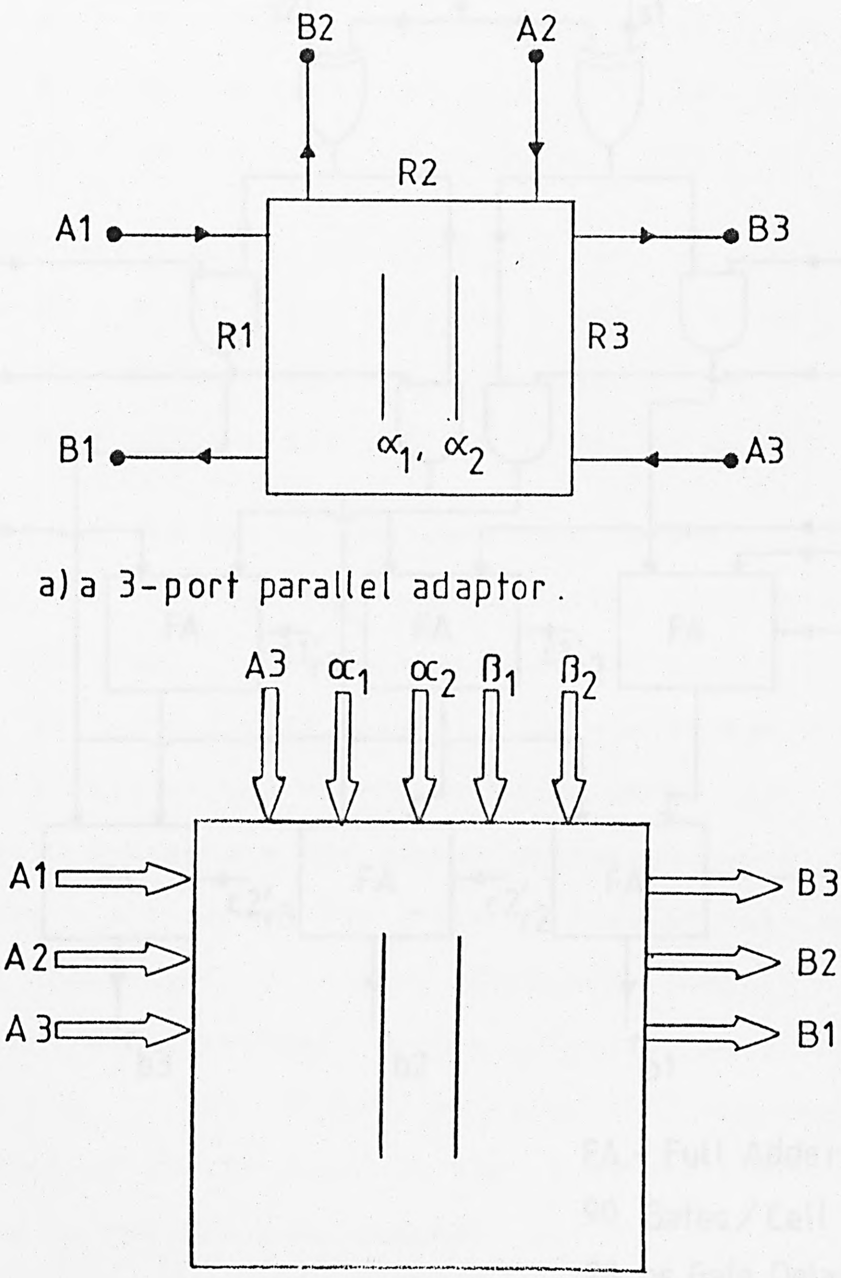
or
$$A0 = \alpha_1(A1 - A3) + \alpha_2(A2 - A3) + 2A3 \quad (4.4)$$

Now we substitute eqn. 4.4 into eqn. 4.2a and set k=1,2,3. This results in,

$$B1 = \alpha_1(A1 - A3) + \alpha_2(A2 - A3) + 2A3 - A1$$

$$B2 = \alpha_1(A1 - A3) + \alpha_2(A2 - A3) + 2A3 - A2$$

Fig.4.1-Schematic representation of,



a) a 3-port parallel adaptor.

$\beta_1 = \alpha_1 - 1$, $\beta_2 = \alpha_2 - 1$

b) a 3-port parallel systolic adaptor.

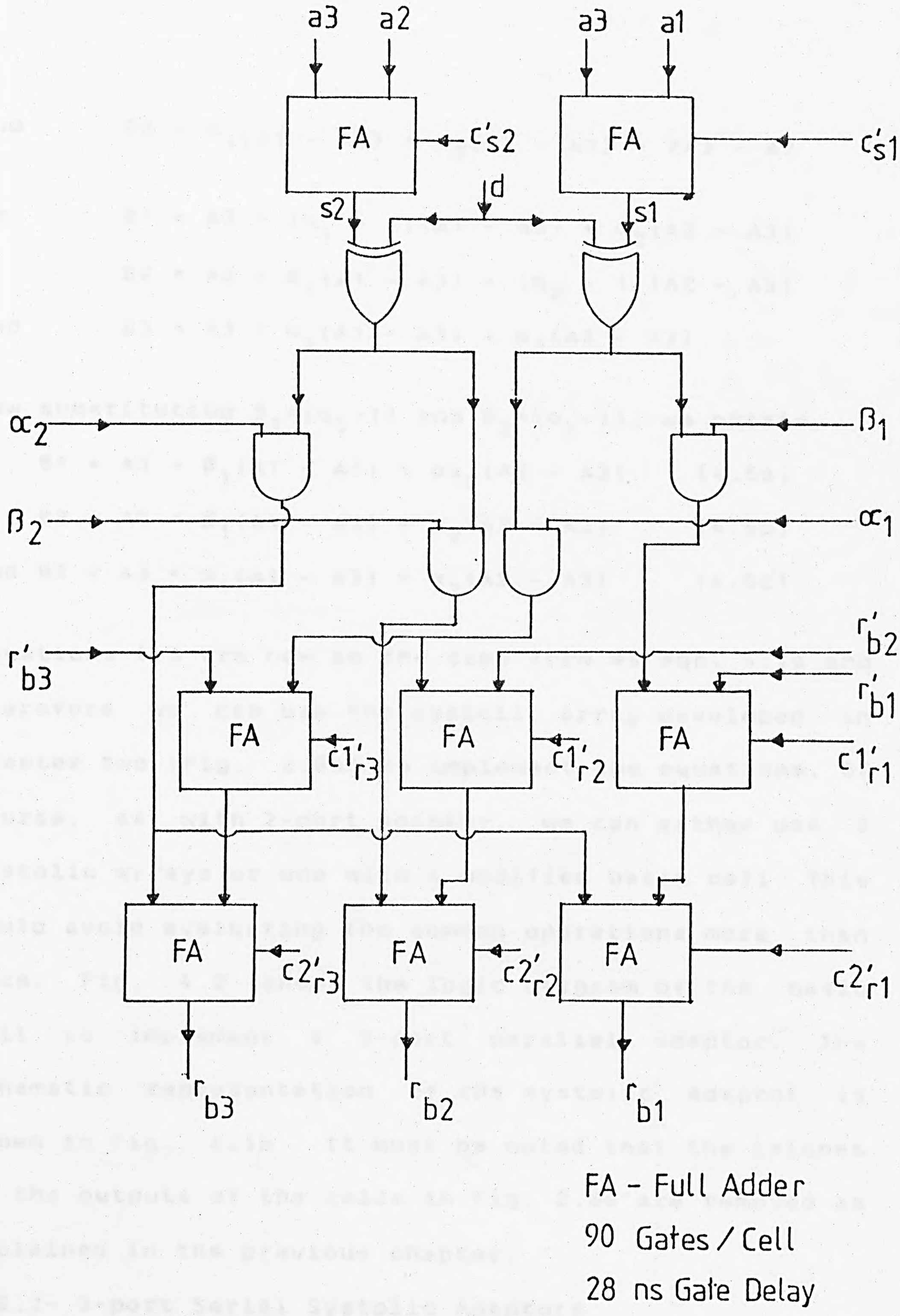


Fig. 4.2 _ Logic diagram of the basic cell in a 3_port parallel systolic adaptor.

and $$B3 = \alpha_1(A1 - A3) + \alpha_2(A2 - A3) + 2A3 - A3$$

or $$B1 = A3 + (\alpha_1 - 1)(A1 - A3) + \alpha_2(A2 - A3)$$

$$B2 = A3 + \alpha_1(A1 - A3) + (\alpha_2 - 1)(A2 - A3)$$

and $$B3 = A3 + \alpha_1(A1 - A3) + \alpha_2(A2 - A3)$$

Now substituting $\beta_1=(\alpha_1-1)$ and $\beta_2=(\alpha_2-1)$, we obtain,

$$B1 = A3 + \beta_1(A1 - A3) + \alpha a_2(A2 - A3) \qquad (4.5a)$$

$$B2 = A3 + \beta_1(A1 - A3) + \alpha_2(A2 - A3) \qquad (4.5b)$$

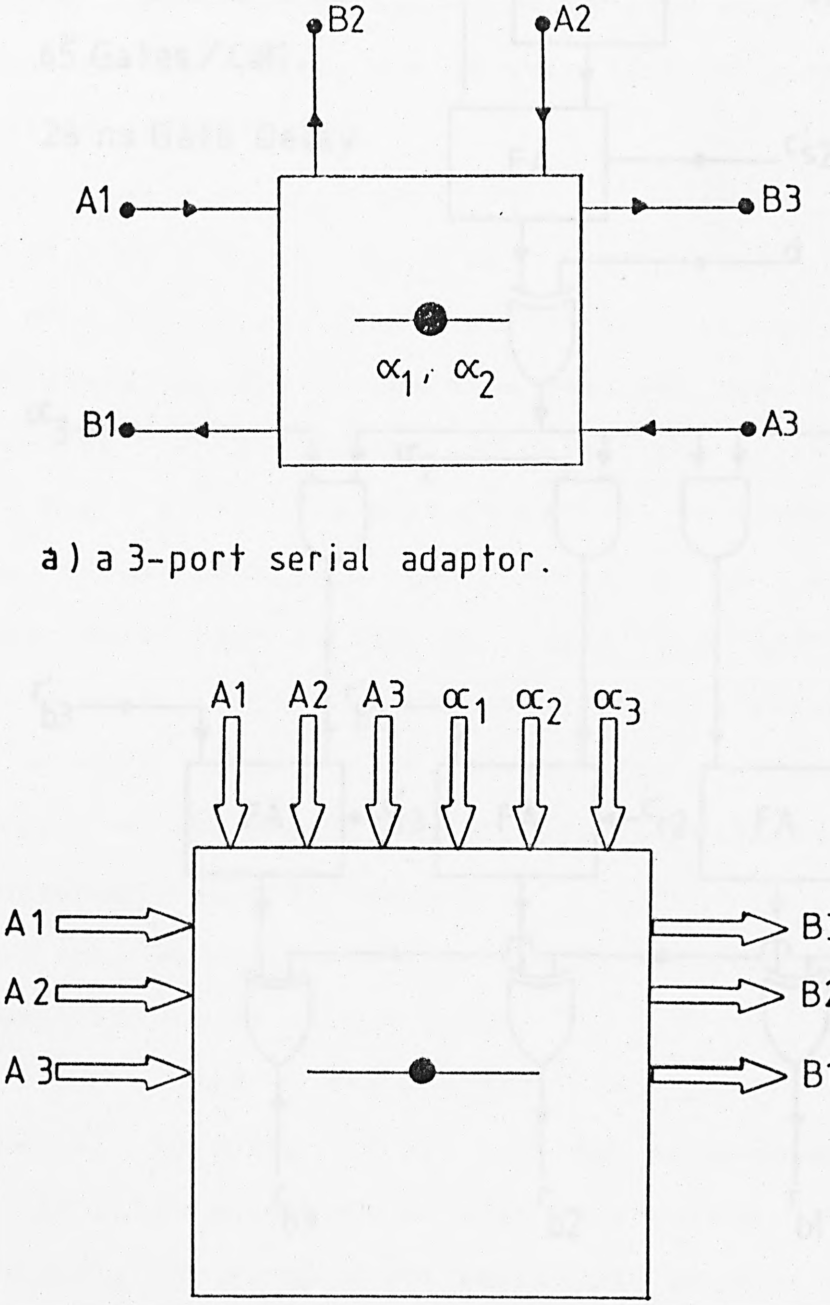
and $$B3 = A3 + \alpha_1(A1 - A3) + \alpha_2(A2 - A3) \qquad (4.5c)$$

Equations 4.5 are now in the same form as eqn. 4.1a and therefore we can use the systolic array developed in chapter two (Fig. 2.6a) to implement the equations. Of course, as with 2-port adaptor, we can either use 3 systolic arrays or one with a modified basic cell. This would avoid evaluating the common operations more than once. Fig. 4.2 shows the logic diagram of the basic cell to implement a 3-port parallel adaptor. The schematic representation of the systolic adaptor is shown in Fig. 4.1b. It must be noted that the latches at the outputs of the cells in Fig. 2.6a are removed as explained in the previous chapter.

4.2.2- 3-port Serial Systolic Adaptors

Let us now consider the bit-level systolic implementation of a 3-port adaptor. Fig. 4.3a shows the schematic representation of a 3-port serial adaptor and the adaptor equations are given below,

Fig.4.3 - Schematic representation of,



a) a 3-port serial adaptor.

b) a 3-port serial systolic adaptor.

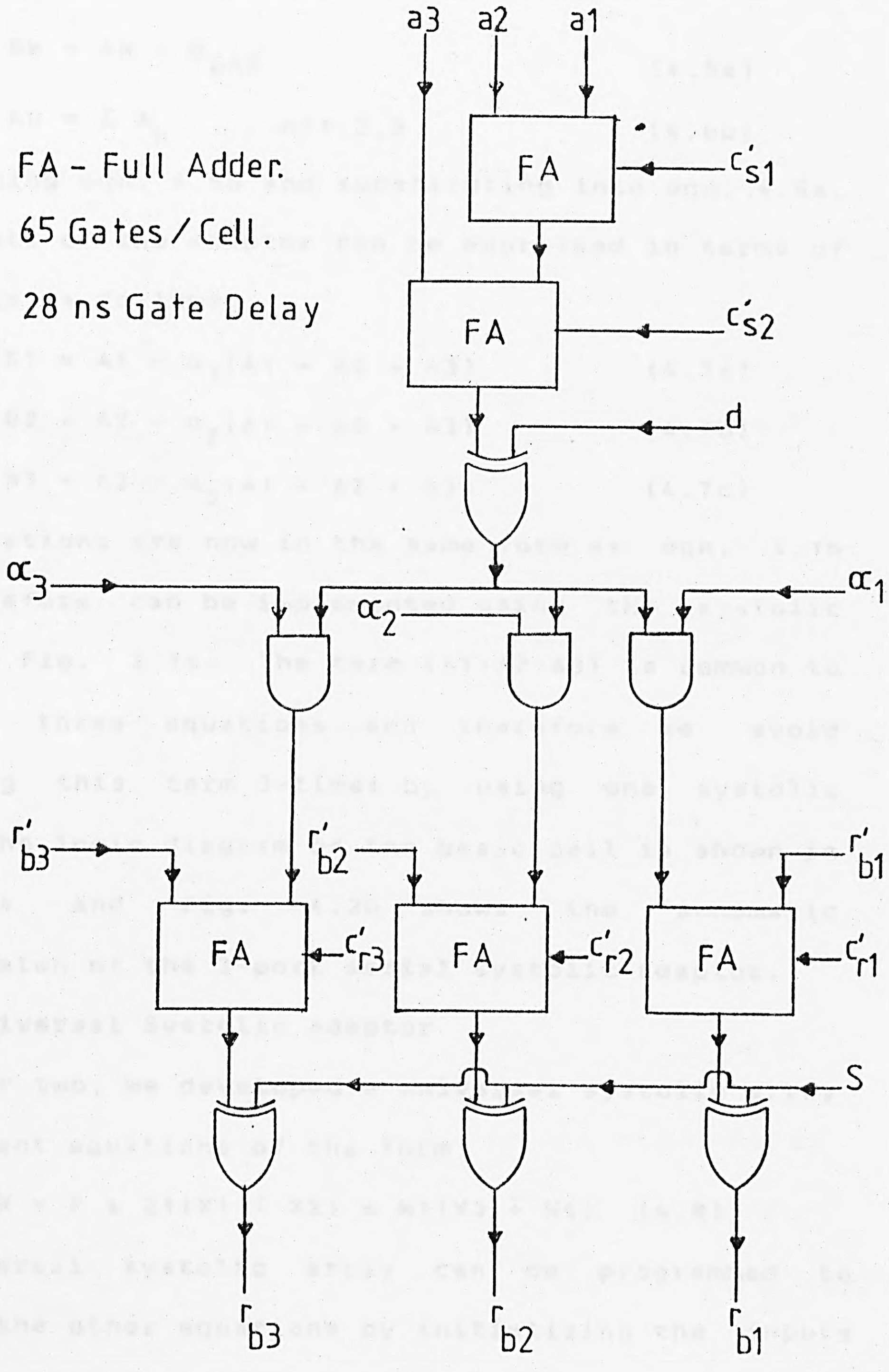


Fig. 4.4 - Logic diagram of the basic cell in a 3_port serial systolic adaptor.

$$Bk = Ak - \alpha_{kA0} \qquad (4.6a)$$

where $$A0 = \Sigma A_k \qquad k=1,2,3 \qquad (4.6b)$$

By expanding eqn. 4.6b and substituting into eqn. 4.6a, the outputs of the adaptor can be expressed in terms of the inputs as follows,

$$B1 = A1 - \alpha_1(A1 + A2 + A3) \qquad (4.7a)$$

$$B2 = A2 - \alpha_2(A1 + A2 + A3) \qquad (4.7b)$$

and $$B3 = A3 - \alpha_3(A1 + A2 + A3) \qquad (4.7c)$$

These equations are now in the same form as eqn. 4.1b and therefore can be implemented using the systolic array of Fig. 2.7a. The term (A1+A2+A3) is common to all the three equations and therefore we avoid evaluating this term 3-times by using one systolic array. The logic diagram of the basic cell is shown in Fig. 4.4 and Fig. 4.3b shows the schematic representaton of the 3-port serial systolic adaptor.

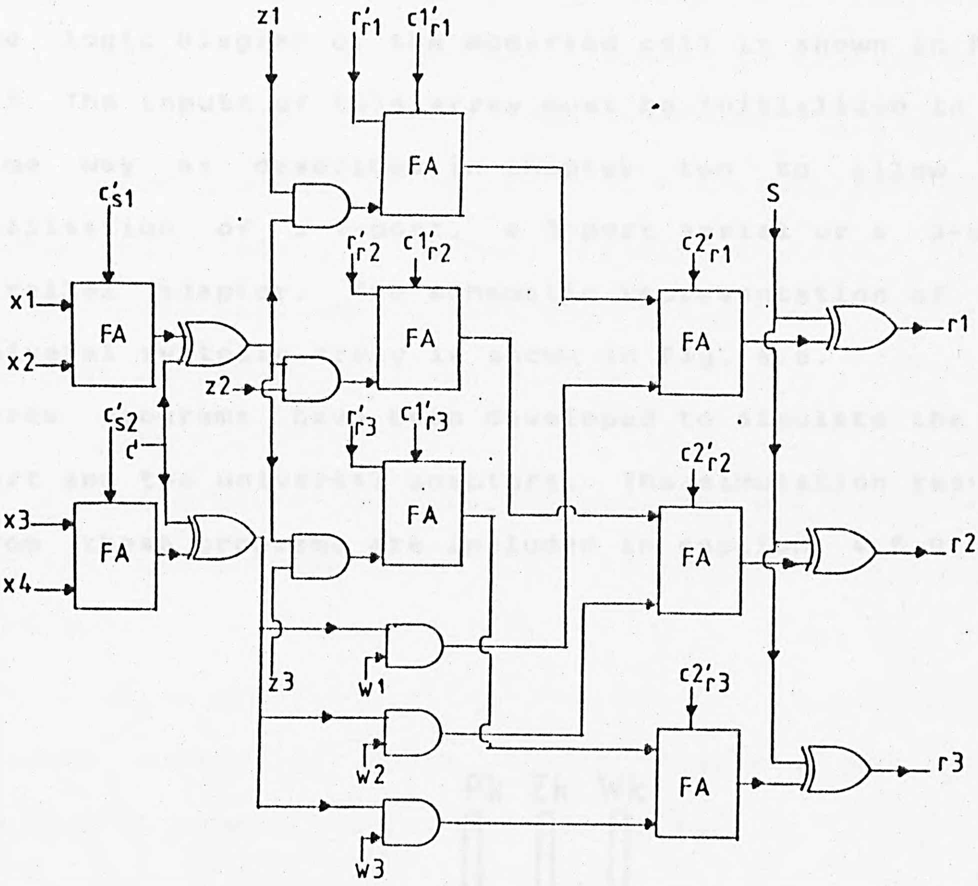
4.2.3- Universal Systolic Adaptor

In chapter two, we developed a universal systolic array to implement equations of the form,

$$R = P \pm Z1(X1 \mp X2) \pm W1(X3 \mp X4) \qquad (4.8)$$

The universal systolic array can be programmed to realise the other equations by initializing the inputs before entering the array. The basic cell of this array (Fig. 2.9a) can be modified so that the array implements a set of equations of the form,

$$R1 = P1 \pm Z1(X1 \mp X2) \pm W1(X3 \mp X4) \qquad (4.9a)$$



FA -Full Adder

101 Gates/Cell

28 ns Gate Delay

Fig.4.5-Logic diagram of the basic cell in a universal adaptor.

$$R2 = P2 \pm Z2(X1 \mp X2) \pm W2(X3 \mp X4) \quad (4.9b)$$

and

$$R3 = P3 \pm Z3(X1 \mp X2) \pm W3(X3 \mp X4) \quad (4.9c)$$

The logic diagram of the modified cell is shown in Fig. 4.5. The inputs of this array must be initialized in the same way as described in chapter two to allow the realisation of a 2-port, a 3-port serial or a 3-port parallel adaptor. The schematic representation of the univeral systolic array is shown in Fig. 4.6.

Three programs have been developed to simulate the 3-port and the universal adaptors. The simulation results from these programs are included in section 4.6.0.

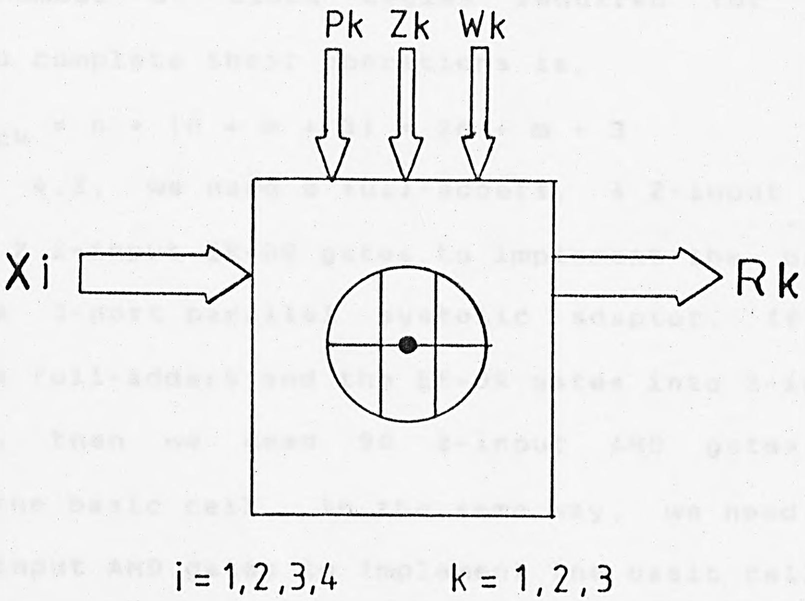


Fig. 4.6-Schematic representation of a universal adaptor.

4.2.4- CMOS Implementation of the Adaptors

Let us now estimate how many transistors are required to implement the 3-port parallel, 3-port serial and the universal adaptors. It is assumed that CMOS technology is used. In general, if we use n-bits to represent the adaptor coefficients and m-bits for the signals, then mnb-bits are required to represent the outputs of the adaptors, where mnb is given by,

$$mnb = (n + m + 3)$$

The cell configurations are the same for all the three adaptors and the number of cells in an (nxm) array is given by,

$$N_c = n(n + m + 3)$$

and the number of clock cycles required for the adaptors to complete their operations is,

$$N_{ck} = n + (n + m + 3) = 2n + m + 3$$

From Fig. 4.2, we need 8 full-adders, 4 2-input AND gates and 2 2-input EX-OR gates to implement the basic cell of a 3-port parallel systolic adaptor. If we convert the full-adders and the EX-OR gates into 2-input AND gates, then we need 90 2-input AND gates to implement the basic cell. In the same way, we need 65 and 101 2-input AND gates to implement the basic cell of a 3-port serial adaptor (Fig. 4.4) and the basic cell of the universal adaptor (Fig. 4.5) respectively. Now, we can calculate the number of transistors required to implement the adaptors.

From Fig. 4.2, 4.4 and 4.5, the time delay of the basic cells in all the cases are equivalent to the time delay of 7 gates in cascade. If we assume that the time delay of an AND gate is 4 ns then the time delay of the basic cells are 28 ns.

This information is shown in a tabular form in Table 4.1 for some typical values of n and m. As with the 2-port adaptor, the number of transistors and the time delay of the adaptors decrease exponentially when the number of bits for the coefficients is reduced slightly. In Table 4.1, column (1) represents the 3-port parallel adaptor, column (2) represents the 3-port serial adaptor and column (3) is the universal adaptor.

No, of Bits		No. of	No. of Gates			No. of Trans			No. of	Delays	
Coef	Signal	Cells	1	2	3	1	2	3	Clocks	ns	MHZ
4	8	60	5,400	3,900	6,060	21,600	15,600	24,240	19	532	1.9
8	8	152	13,680	9,880	15,332	54,720	39,520	61,408	27	756	1.3
4	16	92	8,280	5,980	9,292	33,120	23,920	37,168	27	756	1.3
8	16	216	19,440	14,040	21,816	77,760	56,160	87,264	35	980	1.0

Colum n(1): 3-portparallel systolic adaptor.
Column (2) : 3-port serial systolic adaptor.
Column (3) : The universal systolic adaptor.

Table 4.1

4.3.0- LC-Ladder WDFs (LCWDFs)

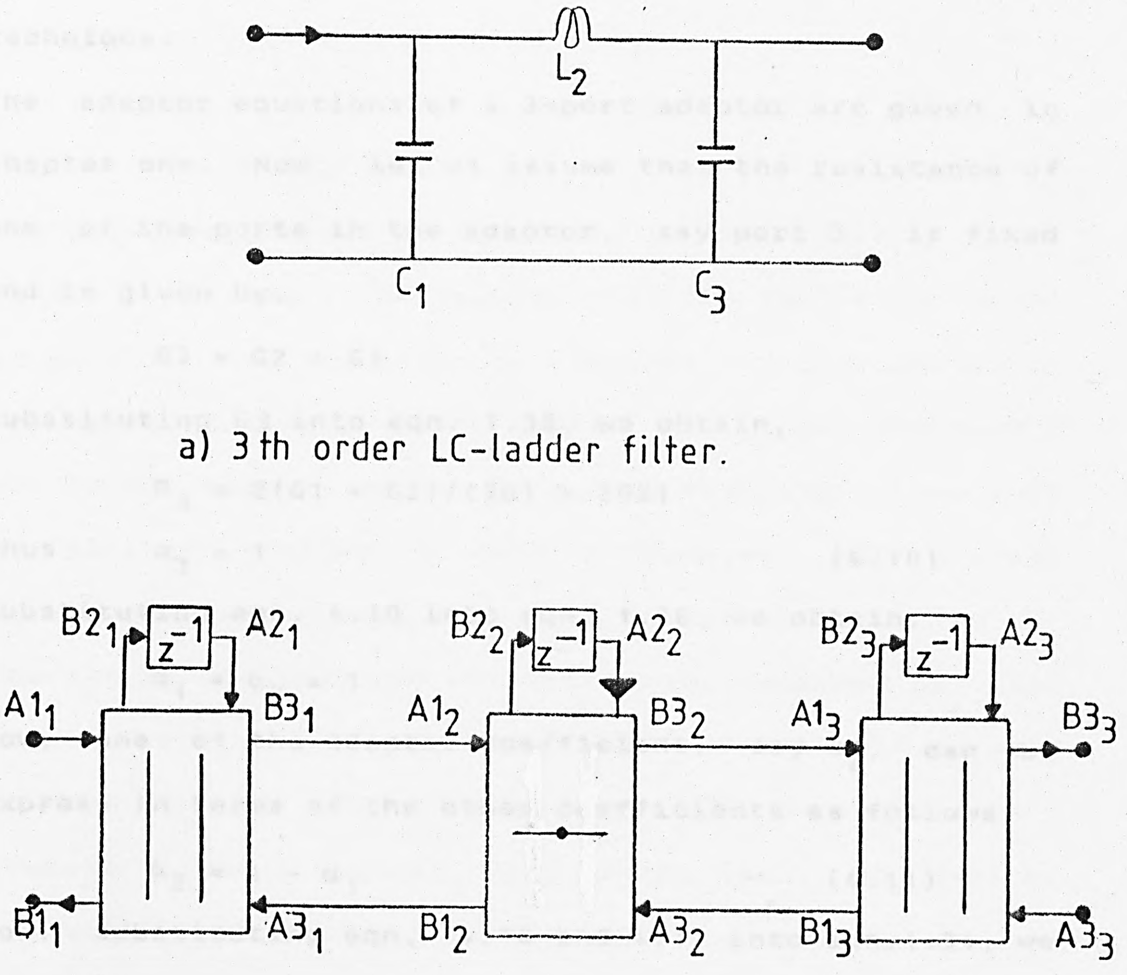
4.3.1- Introduction

Using 2-port and 3-port adaptors, it is possible to derive a WDF from a lc-ladder reference filter. There are different techniques which can be used to derive the WDF. Each technique results in a structure which is different from others in terms of number of multipliers, hardware complexity, etc. In this section, we briefly review some of these techniques.

4.3.2- Direct LC-Ladder WDFs

In principle, a WDF can be realised by direct connection of the parallel and serial adaptors which model the interconnections of the inductors and the capacitors in the reference filter. The series and parallel tuned circuits may be implemented using either 3-port adaptors or 2-port adaptors [2]. As an example, consider the 3rd order lc-ladder filter of Fig. 4.7a. The corresponding WDF is shown in Fig. 4.7b. Each 3-port adaptor requires 3 multiplications, therefore there are 9 multiplications all together. As explained in chapter one, it is possible to express one of the adaptor coefficients in terms of the other two coefficients and reduce the number of multipliers needed in one adaptor by one. This will result in 6 multipliers for the 3rd order LCWDF of Fig. 4.7b. Thus, the number of multiplications is not canonic since we need 2N multipliers for an Nth order

Fig. 4.7



a) 3 th order LC-ladder filter.

b) Corresponding WDF.

filter.

A further saving in the number of multipliers can be achieved by realising the WDF using the matched-port technique [87]. The resulting WDF corresponds to the true ladder filter in the strictly conventional sense. The following is a brief review of the matched-port technique.

The adaptor equations of a 3-port adaptor are given in chapter one. Now, let us assume that the resistance of one of the ports in the adaptor, say port 3, is fixed and is given by,

$$G3 = G2 + G1$$

Substituting G3 into eqn. 1.35, we obtain,

$$\alpha_3 = 2(G1 + G2)/(2G1 + 2G2)$$

Thus $\alpha_3 = 1$ (4.10)

Substituting eqn. 4.10 into eqn. 1.36, we obtain,

$$\alpha_1 + \alpha_2 = 1$$

Now, one of the adaptor coefficient, say α_2, can be express in terms of the other coefficients as follows,

$$\alpha_2 = 1 - \alpha_1 \qquad (4.11)$$

Now, substituting eqn. 4.10 and 4.11 into eqn.1.34, we obtain,

$$B1 = \alpha_1(A1 - A2) + A2 + A3 - A1 \qquad (4.12a)$$

$$B2 = \alpha_1(A1 - A2) + A3 \qquad (4.12b)$$

and $B3 = \alpha_1(A1 - A2) + A2$ (4.12c)

The wave realisation of eqn. 4.12 is shown in Fig. 4.8a.

As it can be seen from Fig. 4.8a, only one multiplier is required to implement the adaptor. A similar approach can be taken in order to reduce the number of multipliers in a 3-port serial adaptor to one. The final adaptor equations, when α_3 is set equal to 1, are given below,

$$B1 = A1 - \alpha_1(A1 + A2 + A3) \qquad (4.13a)$$

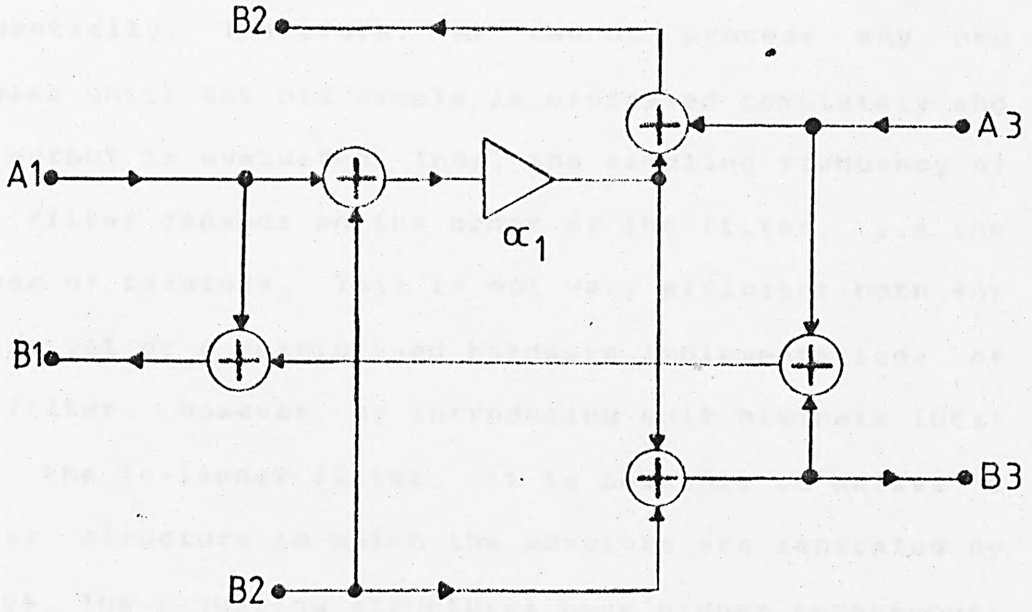
$$B2 = -(A1 + A2 - \alpha_1(A1 + A2 + A3)) \qquad (4.13b)$$

and $$B3 = -(A1 + A2) \qquad (4.13c)$$

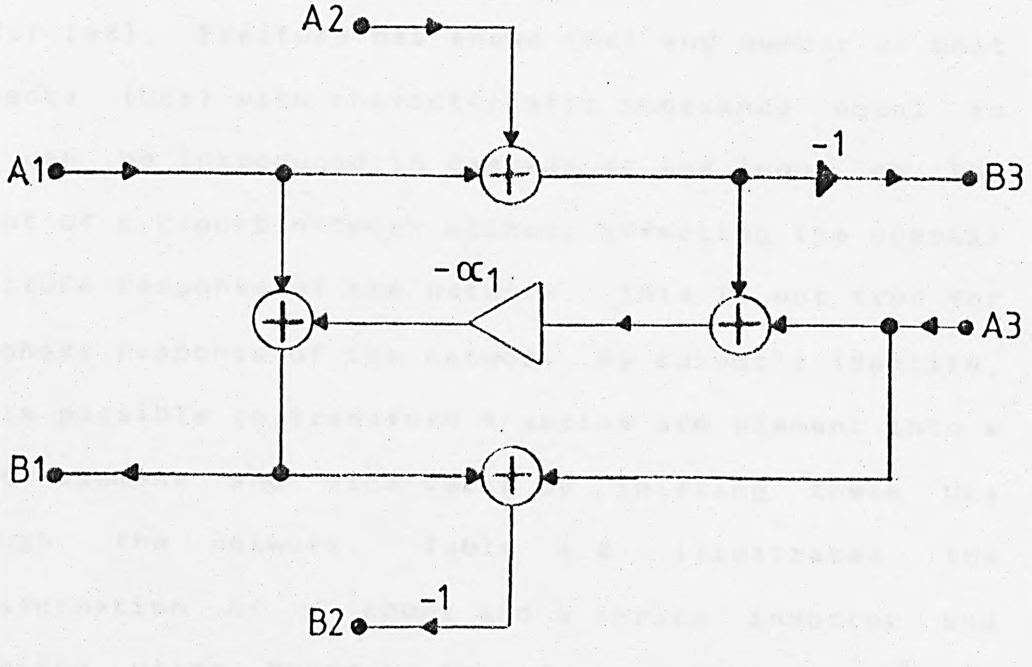
The wave realisation of eqn. 4.13 is shown in Fig. 4.8b. Eqns. 4.12c and 4.13c suggest that the reflected waves at port 3, B3, in both the adaptors are independent of the corresponding incident waves, A3. Thus, the port 3 of these adaptors may be connected directly to any port of another adaptor to form a complete WDF without introducing any closed loop into the structure.

Now let us consider the WDF structures obtained in this section from an implementation point of view. If we wish to use the WDF structure in Fig. 4.7b to implement a lowpass filter, then $A1_1$ will be the input, $B3_3$ will be the output and $A3_3$ will be set equal to zero. At adaptor '1', $B3_1$ can be calculated since all the inputs are known. Next at adaptor '2', $B3_2$ can be calculated since the new value of $A1_2$ has just been evaluated and also $A2_2$ and $A3_2$ are Known. Finally at adaptor '3', $B3_3$, the filter output, can be calculated since $A1_3$, $A2_3$ and $A3_3$

Fig.4.8



a) 3-port parallel adaptor with one multiplier.

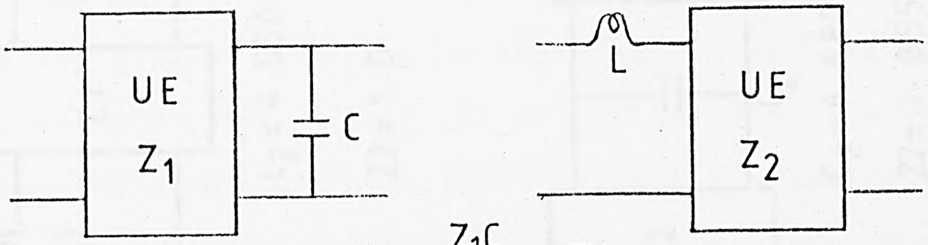


b) 3-port seril adaptor with one multiplier.

are known. These operations must take place sequentially. Therefore, we cannot process any new samples until the old sample is processed completely and the output is evaluated. Thus, the sampling frequency of the filter depends on the order of the filter, i.e the number of adaptors. This is not very efficient both for a parallel or a multiplexed hardware implementations of the filter. However, by introducing unit elemnets (UEs) into the lc-ladder filter, it is possible to derive a filter structure in which the adaptors are separated by delays. The resulting structures have higher throughputs and also can be multiplexed quite easily. This leads to the design of LCWDFs using Kuroda Transformations.

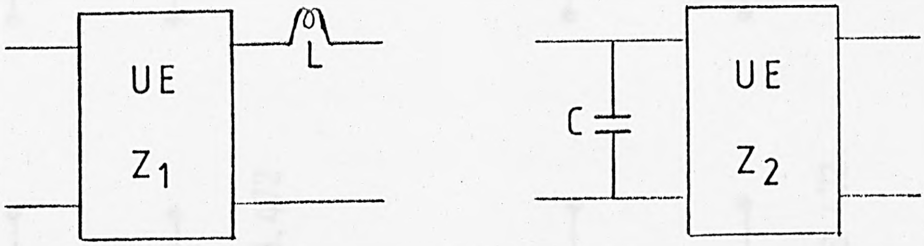
4.3.3- LCWDFs with inserted Unit Elements

In Ref [92], Fraiture has shown that any number of unit elements (UEs) with characteristic impedance equal to one can be introduced in cascade at the input or the output of a 2-port network without affecting the overall amplitude response of the network. This is not true for the phase response of the network. By Kuroda's identity, it is possible to transform a series arm element into a shunt element and vice versa by shifting these UEs through the network. Table 4.2 illustrates the transformation of a shunt and a series inductor and capacitor using Kuroda's transforms. The process of inserting and shifting the UEs into a lc-ladder filter



$$L = \frac{Z_1 C}{1 + Z_1 C} Z_1$$

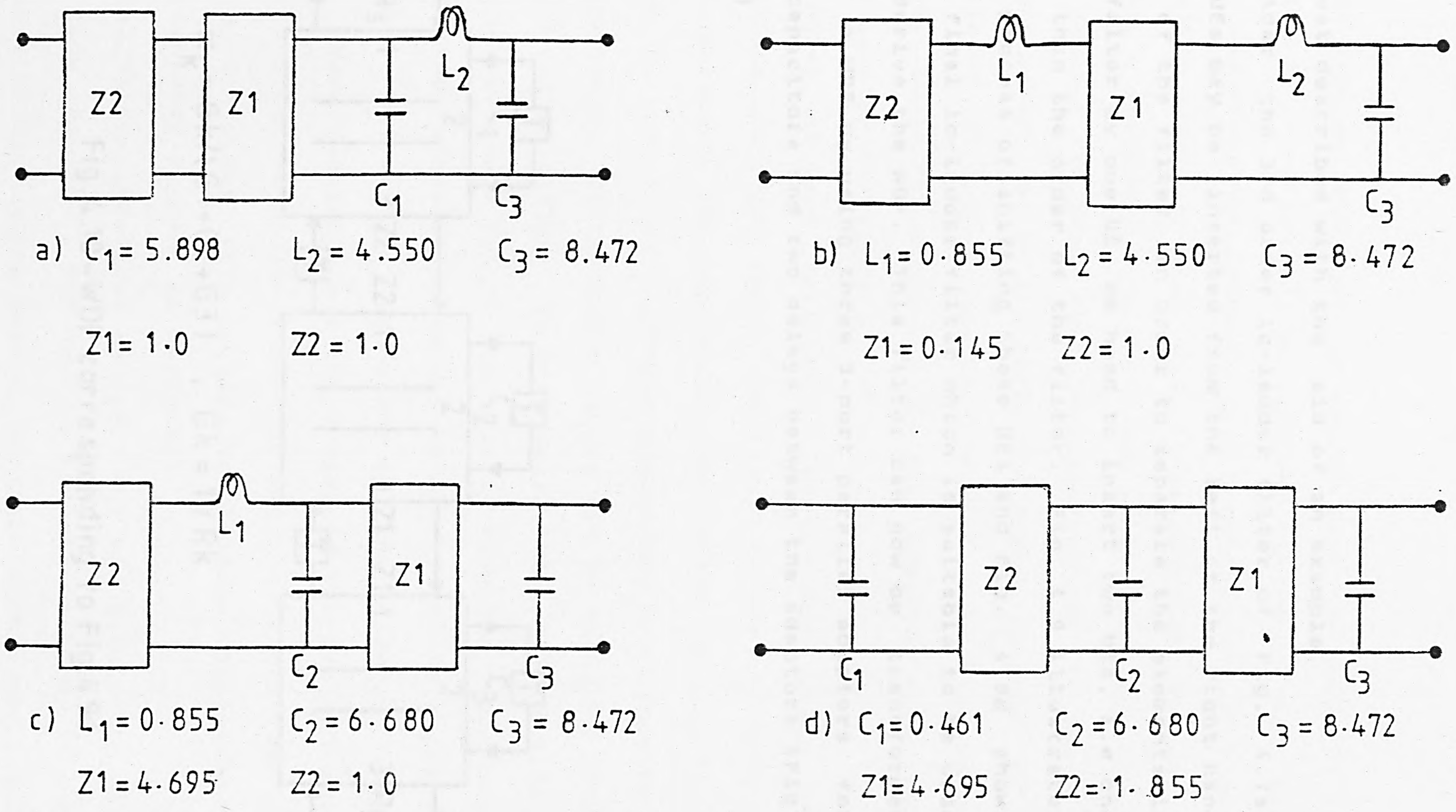
$$Z_2 = \frac{Z_1}{1 + Z_1 C}$$



$$C = \frac{1}{Z_1(Z_1 + L)}$$

$$Z_2 = Z_1 + L$$

Table 4.2- Kuroda's transforms.



a) $C_1 = 5.898$ $L_2 = 4.550$ $C_3 = 8.472$

$Z1 = 1.0$ $Z2 = 1.0$

b) $L_1 = 0.855$ $L_2 = 4.550$ $C_3 = 8.472$

$Z1 = 0.145$ $Z2 = 1.0$

c) $L_1 = 0.855$ $C_2 = 6.680$ $C_3 = 8.472$

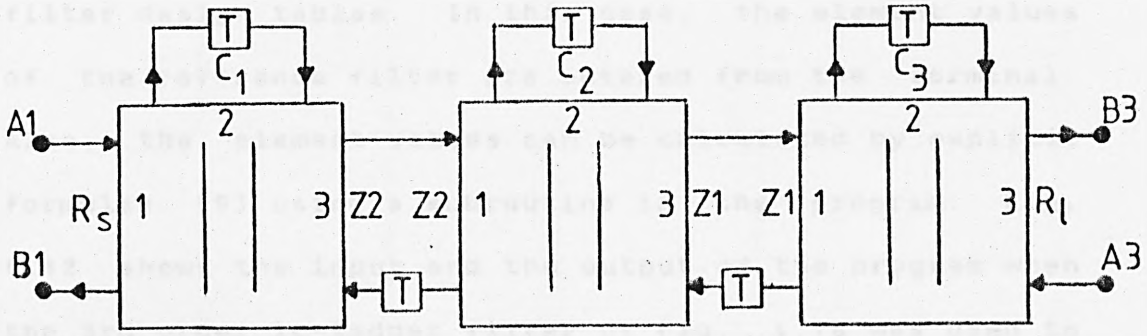
$Z1 = 4.695$ $Z2 = 1.0$

d) $C_1 = 0.461$ $C_2 = 6.680$ $C_3 = 8.472$

$Z1 = 4.695$ $Z2 = 1.855$

Fig. 4.9-Process of Kuroda's transformation.

is best described with the aid of an example. Consider the 3rd order lc-ladder filter of Fig. 4.7a. The UEs may be inserted from the left or the right hand side of the filter. In order to separate the elements in the filter by one UE, we need to insert two UEs, i.e one less than the order of the filter. Fig. 4.9 illustrates the process of shifting these UEs and Fig. 4.9d shows the final lc-ladder filter which is suitable to be used to derive the WDF. This filter can now be transformed into a WDF by using three 3-port parallel adaptors for the capacitors and two delays between the adaptors (Fig. 4.10).



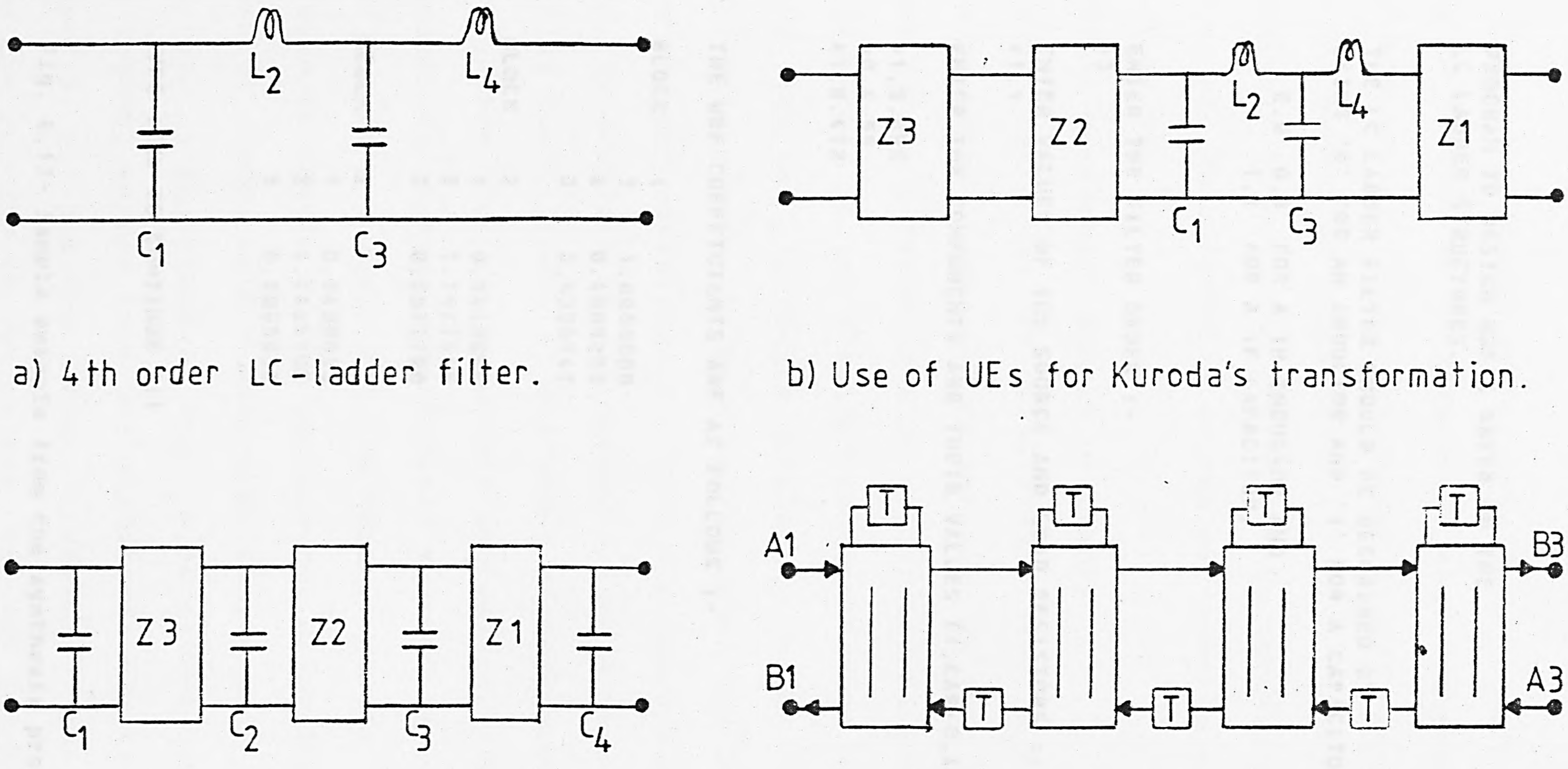
$$\alpha_k = Gk/(G1+G2+G3) \quad , \quad Gk = 1/Rk$$

Fig.4.10 - WDF corresponding to Fig.4.9d.

It is easy to show that no matter what the order of the filter is, the UEs can be inserted in such a way that the resulting WDF contains only 3-port serial or 3-port parallel adaptors. Fig. 4.11 illustrates the process of Kuroda's transformation for a 4th order lc-ladder filter and Fig. 4.11c shows the corresponding WDF. In general, if the order, N, of the reference filter is even, it is necessary to insert one UE from right hand side of the filter while the other N-2 UEs are insearted from left hand side of the filter.

A program has been developed to derive LCWDFs using the technique described above. The resulting WDFs contain 3-port parallel adaptors only. The filter is described using '0' and '1' to represent capacitors and inductors respectively. The reference filter can be designed using filter design tables. In this case, the element values of the reference filter are entered from the terminal. Also, the element values can be calculated by explicit formulas [9] using a subroutine in the program. Fig. 4.12 shows the input and the output of the program when the 3rd order lc-ladder filter of Fig. 4.7a was used to derive the LCWDF. This program only allows Butterworth and Chebychev filters to designed.

In the filter structure of Fig. 4.9 and 4.10, the UEs are made redundant and have no effect on the filter response. This means that if we start with a Nth order



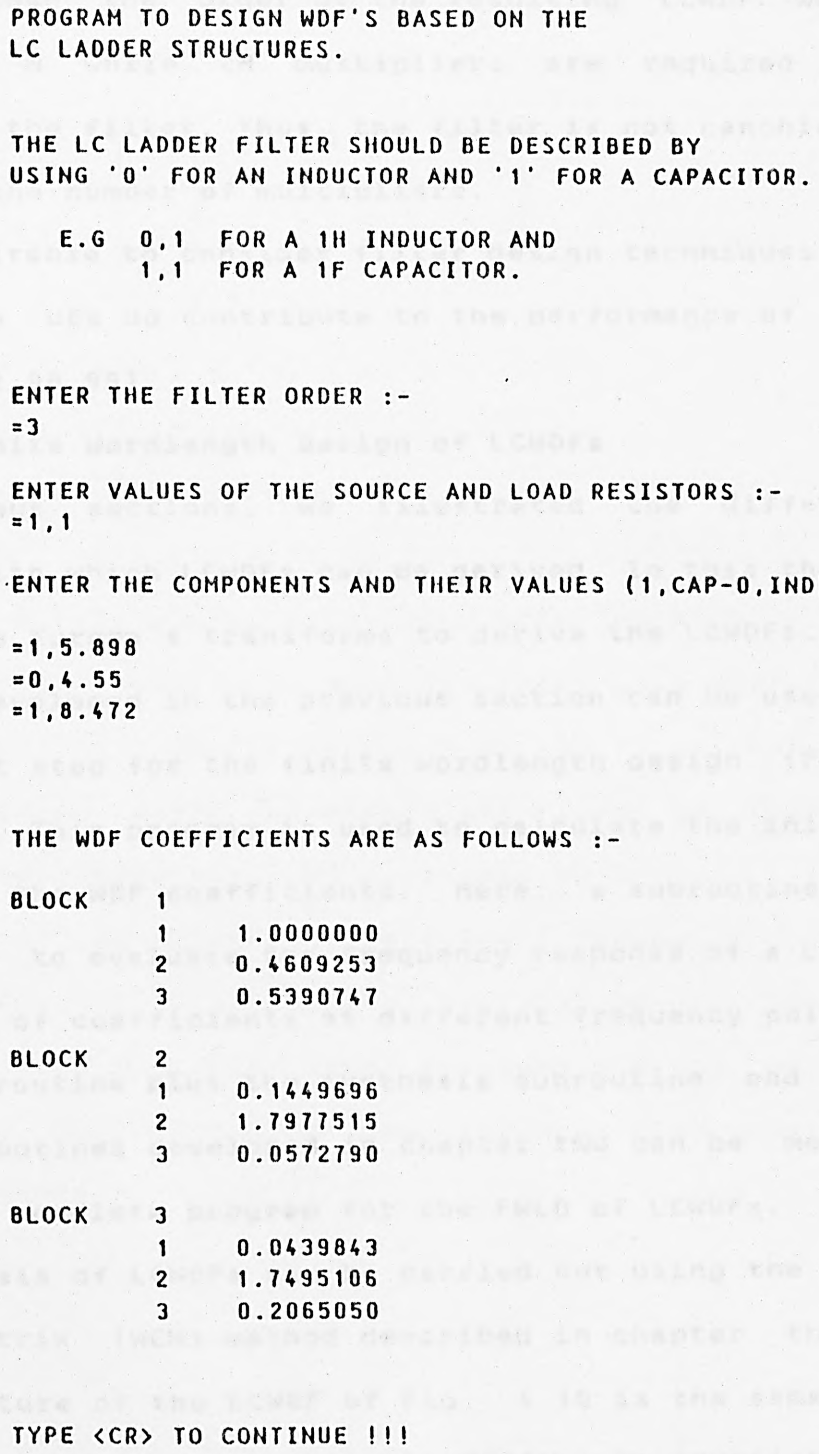
a) 4th order LC-ladder filter.

b) Use of UEs for Kuroda's transformation.

c) LC-ladder filter with inserted UEs.

d) Corresponding WDF using 3-port parallel adaptors only.

Fig.4.11



```
PROGRAM TO DESIGN WDF'S BASED ON THE
LC LADDER STRUCTURES.

THE LC LADDER FILTER SHOULD BE DESCRIBED BY
USING '0' FOR AN INDUCTOR AND '1' FOR A CAPACITOR.

   E.G  0,1  FOR A 1H INDUCTOR AND
        1,1  FOR A 1F CAPACITOR.


ENTER THE FILTER ORDER :-
=3

ENTER VALUES OF THE SOURCE AND LOAD RESISTORS :-
=1,1

ENTER THE COMPONENTS AND THEIR VALUES (1,CAP-0,IND) :-

=1,5.898
=0,4.55
=1,8.472


THE WDF COEFFICIENTS ARE AS FOLLOWS :-

BLOCK     1
          1    1.0000000
          2    0.4609253
          3    0.5390747
BLOCK     2
          1    0.1449696
          2    1.7977515
          3    0.0572790

BLOCK     3
          1    0.0439843
          2    1.7495106
          3    0.2065050



TYPE <CR> TO CONTINUE !!!
```

Fig. 4.12- Sample example from the synthesis program.

filter, then the order of the resulting LCWDF would still be N while 2N multipliers are required to implement the filter. Thus, the filter is not canonic in terms of the number of multipliers.

It is desirable to consider filter design techniques in which the UEs do contribute to the performance of the LCWDFs [42,98,99].

4.3.4- Finite Wordlength Design of LCWDFs

In previous sections, we illustrated the different methods with which LCWDFs can be derived. In this thesis we use the Kuroda's transforms to derive the LCWDFs. The program developed in the previous section can be used as the first step for the finite wordlength design (FWLD) of LCWDfs. This program is used to calculate the initial values of the WDF coefficients. Here, a subroutine is developed to evaluate the frequency response of a LCWDF for a set of coefficients at different frequency points. This subroutine plus the synthesis subroutine and the two subroutines developed in chapter two can be merged to form a complete program for the FWLD of LCWDFs.

The analysis of LCWDFs can be carried out using the wave chain matrix (WCM) method described in chapter three. The structure of the LCWDF of Fig. 4.10 is the same as that of a UEWDF except that the difference equations of the adaptors are not the same. Therefore, the same technique can be used to obtain the transfer function of the filter. From Appendix 'A3', the input/output relationship of the kth section of a LCWDF is given by,

$$\begin{vmatrix} A1_k \\ \\ B1_k \end{vmatrix} = \begin{vmatrix} A & B \\ \\ C & D \end{vmatrix} \begin{vmatrix} B3_k \\ \\ A3_k \end{vmatrix} \qquad (4.14)$$

where $K = 1/[\alpha_1(1+z^{-1})]$

$A = (1-\alpha_3) + (\alpha_1-1)z^{-1}$

$B = 1 + (\alpha_1+\alpha_3-1)z^{-1}$

$C = z^{-1} + (\alpha_1+\alpha_3-1)$

$D = (\alpha_1-1) + z^{-1}(1-\alpha_3)$

This ABCD matrix can be multiplied with the ABCD matrix of the other sections to form the total ABCD matrix of the filter. The transfer function of the filter can then be calculated using the method described in Appendix 'A3'. A subroutine has been developed to analyse a LCWDF for a set of coefficients at different frequency points. In Appendix 'C' a complete listing of the design program, LCFD.SO, has been included and in section 4.5.0, we berifly outline how it can be used for the FWLD of LCWDFs.

4.4.0- Hardware Implementation of Systolic LCWDFs

The structure of a LCWDF is very similar to that of a UEWDF. The basic blocks, i.e the adaptors, are separated by delays which enables us to pipeline the filter. As mentioned in the previous section, the LCWDF can be designed in such a way that only 3-port parallel or serial adaptors are required to implement the filter. The resulting structure will be much easier to be multiplexed. As with the UEWDFs, the 3-port parallel adaptors can be replaced by the corresponding systolic adaptors if the filter is implemented in parallel (Fig. 4.10). The sampling frequency of the filter in parallel form would be around (1.3/2) MHz for 16-bit signals and 4-bit coefficients (Table 4.2). The sampling frequency of the filter is independent of the order of the filter. The number of transistors needed to implement an Nth order filter is,

$$(N \times 33120) + NT_d$$

where NT_d is the number of transistors required to implement the delays beteween the adaptors.

It is also possible to implement the filter by multiplexing one 3-port parallel systolic adaptor to the required filter order. The filter structure of a multiplexed LCWDF is roughly the same as that of a UEWDF given in Fig. 3.15 in chapter 3. The sampling frequency of the filter now depends on the order of the filter.

4.5.0- Description of LCFD.SO program

Fig. 4.13 illustrates the main menu of LCFD.SO program. The procedure to design a finite wordlength LCWDF is very much the same as in UEFD.SO and LTFD.SO program. The following is a brief outline of how the program may be used.

1) First we must decide on how the reference filter is going to be designed. One way would be to use filter design tables to obtain the element values of the reference filter for a given set of specifications. In this case, the reference filter is described by the method shown in Fig. 4.12. Then the WDF coefficients are calculated using the KUroda's transforms. The other approach to the design of the reference filter would be to use explicit formulaes to evaluate the element values. This can be achieved by choosing option '2' in the main menu. The filter can be designed with a Butterworth or a Chebyshev response.

2) Having designed the reference filter, the next stage is to calculate the LCWDF coefficients. This is done by choosing option '3'. At this stage the coefficients can be saved in a file using option '4' for later use. The initial coefficients can also be entered directly from the terminal using option '5'.

3) Now, we can enter the optimization routine and design finite wordlength filters using option '4'. As

with UEFD.SO and LTFD.SO programs, if the algorithm is successful then the final coefficients from FWLD routine will be quantized to the required number of bits. If however the specifications are not met then the number of bits is increased by one and the algorithm is run again. The final coefficients can also be saved using option '5' from the main menu.

The frequency response of the filter designed can be checked using the ANAWDF program.

```
                PROGRAM TO DESIGN LC-LADDER WDFS


                           MAIN MENU



                1) READ INITIAL COEFFICIENTS.
                2) DESIGN LC-LADDER FILTERS.
                3) DESIGN LC-LADDER WDFS.
                4) DESIGN FINITE WORDLENGTH WDFS.
                5) SAVE COEFFICIENTS.
                6) END PROGRAM.
```

Fig. 4.14- Main menu of the LCFD.SO program.

FILTER NO.	PASS BAND EDGE FREQ	STOP BAND EDGE FREQ	MAX RIPPLE IN PASSBAND	MIN LOSS IN STOPBAND
1	0.10	0.20	1.0	50
2	0.20	0.30	0.5	60
3	0.40	0.45	0.5	60

Sampling frequency is normalised to 1.0 HZ.

Table 4.3.

4.6.0- Analysis and Simulation Results

In this section, we present a number of examples to illustrate how the LCFD.SO program can be used to design finite wordlength LCWDFs. Also results from the simulation of the 3-port and the universal adaptors are included to prove the correctness of the designs.

Table 4.3 illustrates the specifications of the filters to be designed. In all the plots obtained from the design program, the following symbols are used for different frequency responses,

(a) '+' Frequency response of the filter with synthesis coefficients.

(b) 'x' Frequency response of the filter with quantized coefficients.

(c) 'o' Frequency response of the filter with FWLD program coefficients.

From eqn. 1.35 and 1.39, the LCWDF coefficients are in the range of $0<a_k<2$. Therefore there is no need to use a sign bit when the coefficients are represented in binary form. Instead one bit has to be allocated to the real part of the coefficients and the rest of the bits to the decimal part. Here, when we use the term number of bits, it does not included the real part of the coefficients and only represents the number of bits required to express the decimal part.

4.6.1- Example 4.1

In the first example, we consider the design of LCWDF for filter No. (1), Table 4.3. In all the design, the initial coefficients are obtained using explicit formulas. Using option '2' from the main menu of the LCFD.SO program, Fig. 4.13, it was estimated that in order to meet the specifications the filter order has to be greater than 4. Thus a 5th order chebychev LC-ladder filter was designed. Next, the element values of the LC-ladder filter were used to derive a LCWDF with inserted UEs using option '3'. The resulting coefficients were then used as the initial coefficients for the optimization program. The optimization program managed to minimize the coefficients wordlength to 6-bits and the number of error function calls was 695, i.e NFUNC = 695. The synthesis and the FWLD program coefficients are as follows,

	Synthesis Coeff	FWLD Coeff 6-Bits
1)	1.0000000	0.984375
2)	0.2408364	0.250000
3)	0.6827608	0.703125
4)	0.9460297	1.078125
5)	0.1002782	0.093750
6)	1.8305648	1.843750
7)	0.0580506	0.062500
8)	1.8830614	1.875000
9)	0.0734825	0.062500
10)	1.6720402	1.546875

Fig. 4.14 shows the frequency responses of the filter with (a) synthesis coefficients, (b) quantized

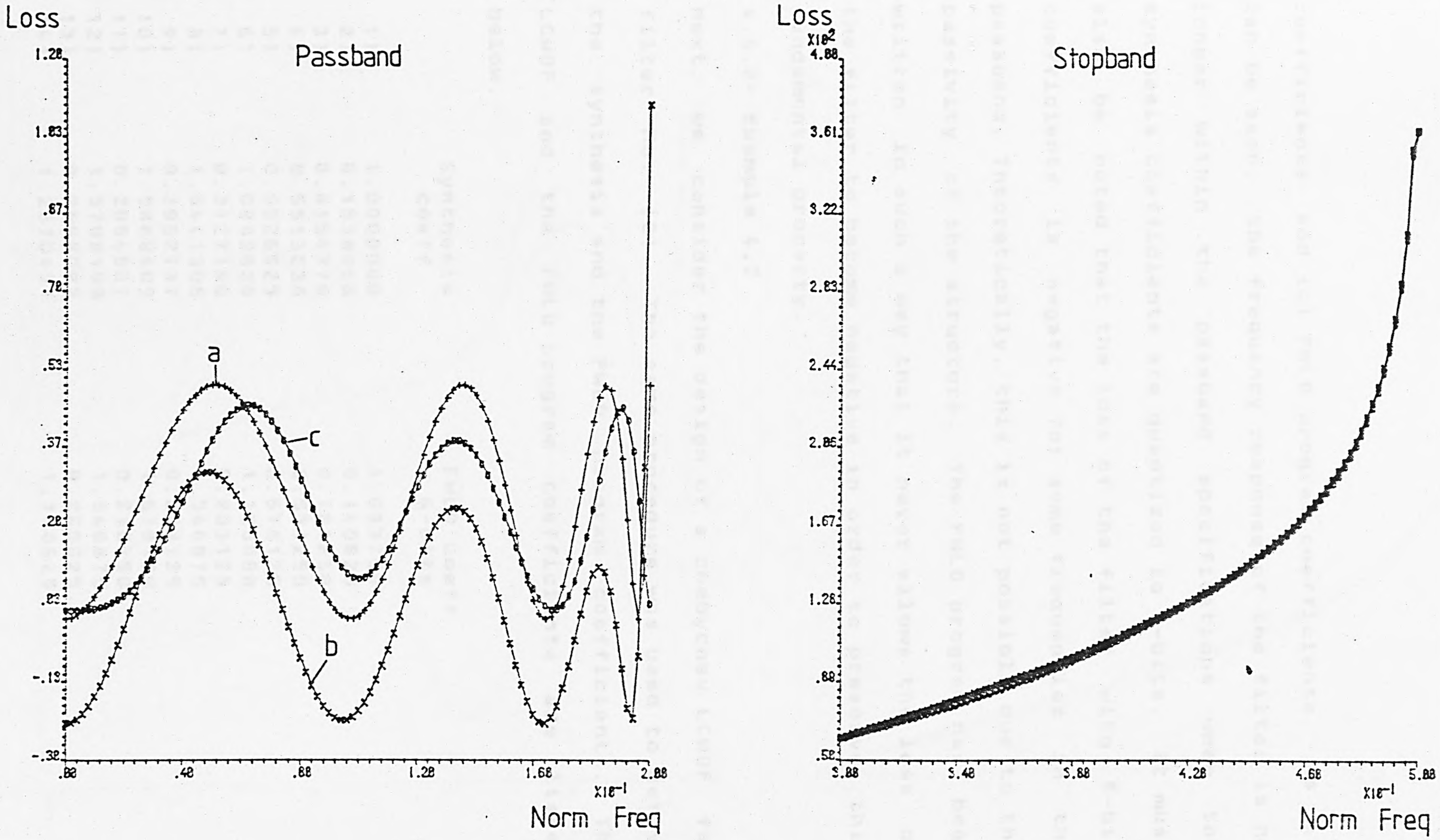


Fig. 4.14 _ Frequency responses for example 4.1 .

coefficients and (c) FWLD program coefficients. As it can be seen, the frequency response of the filter is no longer within the passband specifications when the synthesis coefficients are quantized to 6-bits. It must also be noted that the loss of the filter with 6-bit coefficients is negative for some frequencies in the passband. Theoretically, this is not possible due to the passivity of the structure. The FWLD program has been written in such a way that it never allows the loss of the filter to become negative in order to preserve this fundamental property.

4.6.2- **Example 4.2**

Next, we consider the design of a chebychev LCWDF for filter No. (2). The same procedure was used to derive the synthesis and the FWLD program coefficients. The LCWDF and the FWLD program coefficients are listed below,

	Synthesis	FWLD Coeff
	coeff	6-bits
1)	1.0000000	1.093750
2)	0.1558068	0.140625
3)	0.8154370	0.781250
4)	0.5513635	0.531250
5)	0.5526625	0.578125
6)	1.0989629	1.125000
7)	0.2127180	0.203125
8)	1.5441308	1.546875
9)	0.2052737	0.203125
10)	1.5869409	1.578125
11)	0.2084537	0.218750
12)	1.5708108	1.546875
13)	0.2598022	0.265625
14)	1.2270431	1.140625

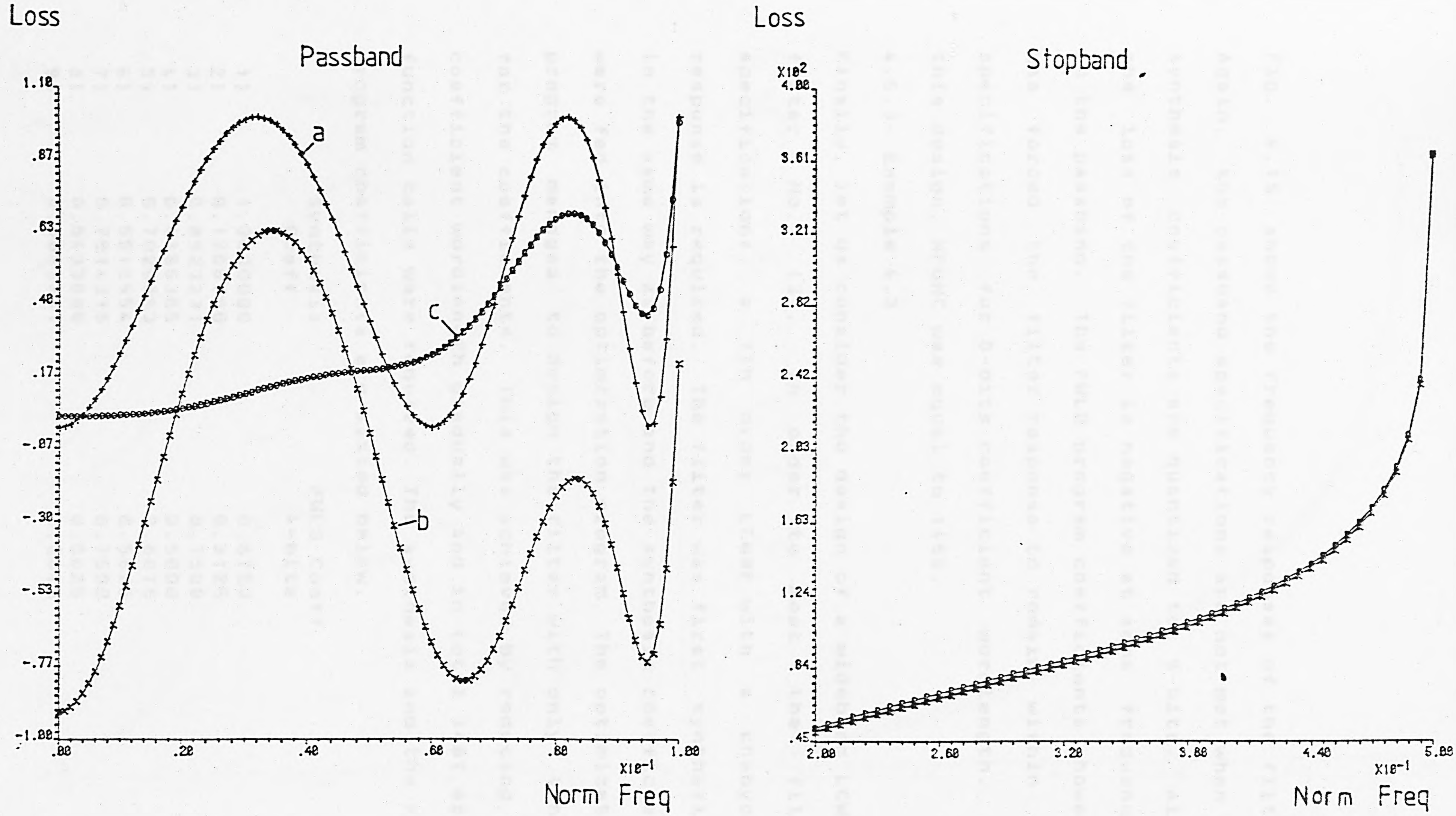


Fig. 4.15_ Frequency responses for example 4.2.

Fig. 4.15 shows the frequency responses of the filter. Again, the passband specifications are not met when the synthesis coefficients are quantized to 6-bits. Also, the loss of the filter is negative at some frequencies in the passband. The FWLD program coefficients however has forced the filter response to remain within the specifications for 6-bits coefficient wordlength. For this design, NFUNC was equal to 1456.

4.6.3- Example 4.3

Finally, let us consider the design of a wideband LCWDF, filter No. (3). In order to meet the filter specifications, a 7th order LCWDF with a chebychev response is required. The filter was first synthesised in the same way as before and the synthesis coefficients were fed into the optimization program. The optimization program managed to design the filter with only 40bits for the coefficients. This was achieved by reducing the coefficient wordlength gradually and in total 3487 error function calls were required. The synthesis and the FWLD program coefficients are listed below,

	Synthesis Coeff	FWLD Coeff 4-bits
1)	1.0000000	0.8750
2)	0.1286760	0.3125
3)	0.8523231	0.7500
4)	0.4166365	0.5000
5)	0.7020883	0.6875
6)	0.5919554	0.5625
7)	0.7514318	0.7500
8)	0.5433666	0.5625
9)	0.6099947	0.6875

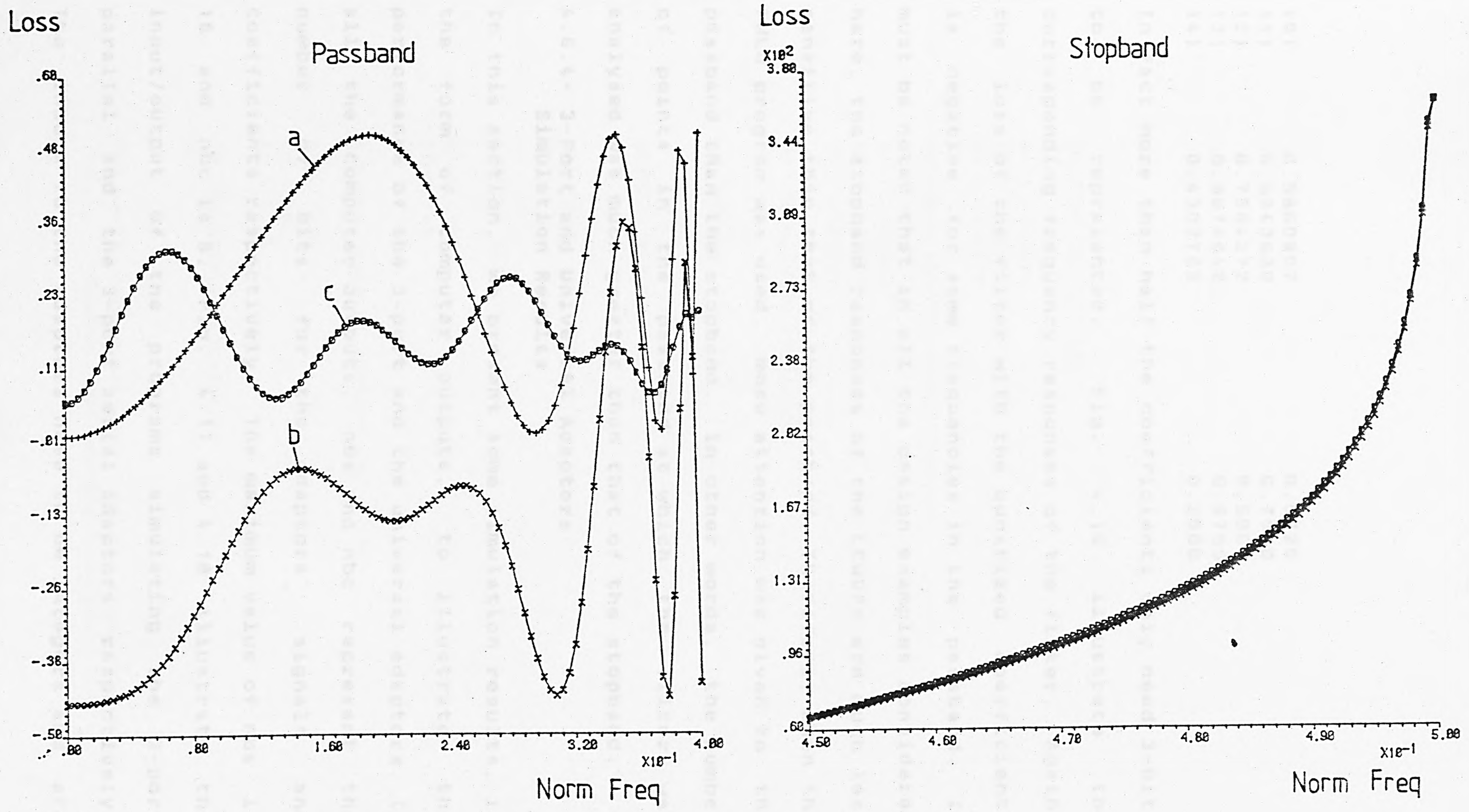


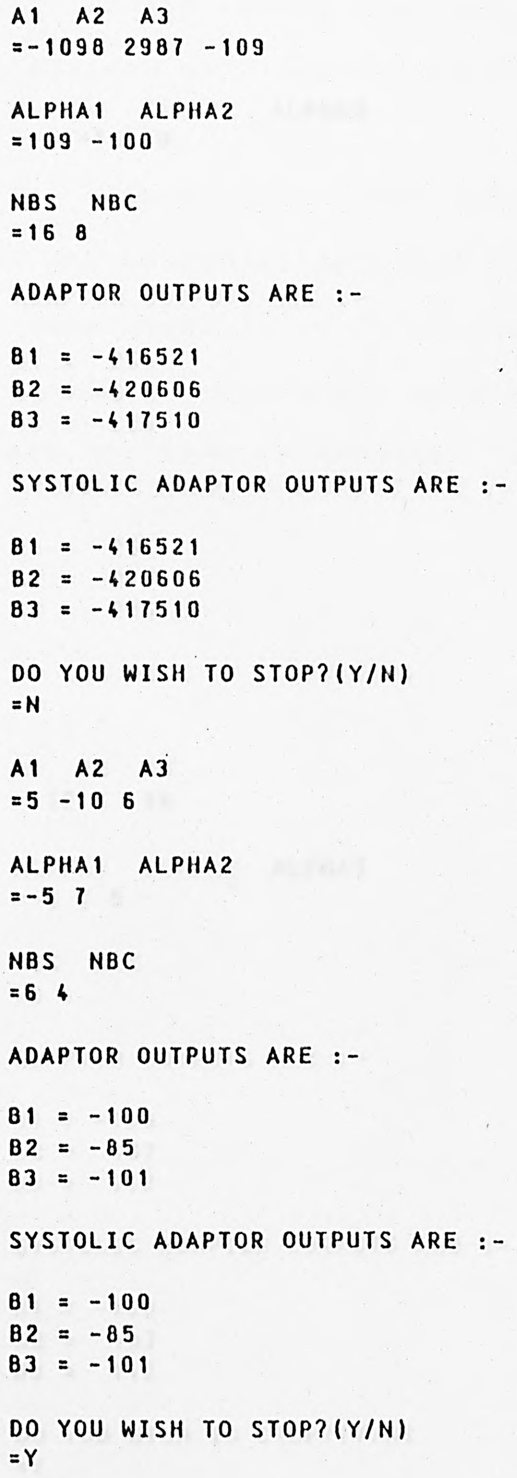
Fig. 4.16 _ Frequency résponses for example 4.3 .

10)	0.5860907	0.5625
11)	0.6643630	0.7500
12)	0.7584272	0.5000
13)	0.8074642	0.8750
14)	0.4302789	0.2500

In fact more than half the coefficients only need 3-bits to be represented. Fig. 4.16 illustrate the corresponding frequency responses of the filter. Again, the loss of the filter with the qunatized coefficients is negative for some frequencies in the passband. It must be noted that in all the design examples considered here, the stopband responses of the LCWDFs are much less sensitive than that of the passband. Therefore, when the FWLD program was used, more attention was given to the passband than the stopband. In other words, the number of points in the passband at which the filter was analysed was much greater than that of the stopband.

4.6.4- 3-Port and Universal Adaptors Simulation Results

In this section, we present some simulation results, in the form of computer outputs, to illustrate the performance of the 3-port and the universal adaptors. In all the computer outputs, nbs and nbc represent the number of bits for the adaptors' signals and coefficients respectively. The maximum value of nbs is 16 and nbc is 8. Fig. 4.17 and 4.18 illustrate the input/output of the programs simulating the 3-port parallel and the 3-port serial adaptors respectively. The inputs to the adaptors have to be integers and are



```
A1  A2  A3
=-1098 2987 -109

ALPHA1  ALPHA2
=109 -100

NBS  NBC
=16 8

ADAPTOR OUTPUTS ARE :-

B1 = -416521
B2 = -420606
B3 = -417510

SYSTOLIC ADAPTOR OUTPUTS ARE :-

B1 = -416521
B2 = -420606
B3 = -417510

DO YOU WISH TO STOP?(Y/N)
=N

A1  A2  A3
=5 -10 6

ALPHA1  ALPHA2
=-5 7

NBS  NBC
=6 4

ADAPTOR OUTPUTS ARE :-

B1 = -100
B2 = -85
B3 = -101

SYSTOLIC ADAPTOR OUTPUTS ARE :-

B1 = -100
B2 = -85
B3 = -101

DO YOU WISH TO STOP?(Y/N)
=Y
```

Fig. 4.17- Sample examples from the 3-port parallel systolic WDF adaptor program.

```
A1  A2  A3
=-19 9 -14

ALPHA1  ALPHA2  ALPHA3
=-9 -16 10

NBS  NBC
=16 8

ADAPTOR OUTPUTS ARE :-

B1 =  255
B2 = -375
B3 =  226

SYSTOLIC ADAPTOR OUTPUTS ARE :-

B1 =  255
B2 = -375
B3 =  226

DO YOU WISH TO STOP?(Y/N)
=N

A1  A2  A3
=-12 5 -18

ALPHA1  ALPHA2  ALPHA3
=-6 7 5

NBS  NBC
=6 4

ADAPTOR OUTPUTS ARE :-

B1 = -168
B2 =  187
B3 =  112

SYSTOLIC ADAPTOR OUTPUTS ARE :-

B1 = -168
B2 =  187
B3 =  112

DO YOU WISH TO STOP?(Y/N)
=Y
```

Fig. 4.18- Sample examples from the 3-port serial

converted to binary in the programs. The arrays must be first initialized with the values of the inputs, i.e the carry bits and the P bits.

Fig. 4.19 shows the input/output of the program simulating the universal systolic adaptor. A number of examples have been given to illustarte the use of the array for realising different adaptors. Listing of these programs are included in Appendix 'B'.

```
                    UNIVERSAL SYSTOLIC WDF ADAPTOR


                                 MENU


                      1) 2-PORT ADAPTOR.
                      2) 3-PORT PARALLEL ADAPTOR.
                      3) 3-PORT SERIAL ADAPTOR.
                      4) END PROGRAM.


ENTER YOUR CHOICE NUMBER :-
=2



ALPHA1  ALPHA2  A1  A2  A3
=-3 7 120 -34 28

NBS  NBC
=16 8




THE ADAPTOR OUTPUTS ARE :-

B1 = -774
B2 = -620
B3 = -682


THE SYSTOLIC ADAPTOR OUTPUTS ARE :-

B1 = -774
B2 = -620
B3 = -682

TYPE <CR> TO CONTINUE !!!
```

Fig. 4.19b- Sample result from the universal systolic WDF adaptor simulating a 3-port parallel adaptor.

```
                     UNIVERSAL SYSTOLIC WDF ADAPTOR


                                    MENU


                        1) 2-PORT ADAPTOR.
                        2) 3-PORT PARALLEL ADAPROR.
                        3) 3-PORT SERIAL ADAPTOR.
                        4) END PROGRAM.


ENTER YOUR CHOICE NUMBER :-
=3


ALPHA1  ALPHA2  ALPHA3  A1   A2  A3
=98 -109 120 1098 -1987 16540

NBS  NBC
=16 8


THE ADAPTOR OUTPUTS ARE :-

B1 = -1532700
B2 =  1703972
B3 = -1861580


THE SYSTOLIC ADAPTOR OUTPUTS ARE :-

B1 = -1532700
B2 =  1703972
B3 = -1861580


TYPE <CR> TO CONTINUE !!!
```

Fig. 4.19c- Sample result from the universal systolic WDF adaptor simulating a 3-port serial adaptor.

4.7.0- Discussion and Comments

In this chapter, we considered the design and systolic implementation of LCWDFs. These filters can be designed in many different ways. Each method results in structures which differ from the others in terms of complexity, number of multipliers, etc. A number of different techniques for the design of LCWDFs have been described briefly and the design of LCWDFs with inserted UEs have been considered in more detail. A computer program has been developed to design these filters and the element values of the LC-ladder reference filter are obtained using explicit formulas. This method of design results in a LCWDF which can be realised using 3-port parallel adaptors only. These structures can be implemented easily by multiplexing one adaptor to the required filter order.

The systolic implementation of the 3-port parallel and 3-port serial adaptor have been achieved using the systolic arrays developed in chapter two. Also, a universal systolic adaptor has been developed which can realise 2-port, 3-port serial and 3-port parallel adaptors. The results from the simulation programs of these systolic adaptors illustrate the correctness of the designs.

The filter examples have illustrated how the optimization program can be used to minimize the coefficient wordlength of the LCWDF. In some cases, the resulting FWLD program coefficients are very simple and require small number of bits for their representation. example 4.3.

Theoretically, it is not possible for the frequency response of a WDF to be negative. This is due to the fact that WDFs are derived from passive analogue filters. From the design examples, it can be seen that in all the cases the loss of the LCWDFs are negative at some frequency points in the passband. In the optimization program, this fundamental property of WDFs, i.e the passivity of the filter, is preserved by not allowing the loss to become negative.

The design of the filter No. (1) has also been considered in chapter three. From examples 3.1, 3.2 and 4.1, we need either a 7th order UEWDF, a 5th order LTWDF or a 5th order LCWDF to meet the specifications. In all the cases, the coefficient wordlength of the filters has been minimized to 6-bits. In terms of the number of multipliers, we need 8, 5 and 10 multipliers for a UEWDF, LTWDF and a LCWDF respectively. Therefore, LTWDF require less multiplications than the other two filters. The UEWDF and LCWDF can however be multiplexed much easier than the LTWDF. Also, the stopband of a LCWDF is extremely insensitive to variations in its multiplier coefficients. Therefore, for a set of specifications and a particular application, one must choose a WDF which meets his requirements.

CHAPTER FIVE

FREQUENCY TRANSFORMATIONS

5.1.0- Introduction

In the previous chapters, we have discussed different techniques for the design of lowpass WDFs. The aim of this chapter is to briefly consider the existing techniques for the design of highpass, bandpass and bandstop WDFs and present the results whhich have been obtained.

In general. there are two main approaches to the design of highpass, bandpass and bandstop digital filters. These design procedures are summarised in Fig. 5.1a and 5.1b. In the first approach, Fig. 5.1a, a lowpass prototype filter is first designed. Next, a suitable frequency transformation is used to transform the lowpass filter to a highpass, bandpass or a bandstop filter. Finally, the analogue filter is mapped into a corresponding digital filter with the use of a suitable transformation (e.g, bilinear transformation).

In the second approach, Fig. 5.1b, the analogue lowpass filter designed is first transformed into a lowpass digital filter. Next, a suitable frequency transformation is carried out in the discrete-time domain to obtain a highpass, a bandpass or a bandstop digital filter.

In chapter one, it was mentioned that any WDF has two inputs, A1 and A2, and two outputs, B1 and B2 (Fig. 5.2). The outputs of the WDF are complementary, i.e if the filter is designed to have a lowpass characteristics and if B2 is the output of the filter, then B2 will have lowpass characteristics while B1 has a highpass characteristics. Similary, if the filter is designed as a bandpass filter, then B2 will have a bandpass characteristics while B1 has a bandstop characteristics. This is the simplest and the cheapest way with which highpass or bandstop filters may be designed. However, it is known [27] that the sensitivity of the output B1 is much higher than that of B2.

In section 5.2, we consider the design of highpass WDFs. We examine the behaviour of the complementary outputs of different WDFs. Also, other approaches to the design of highpass WDFs are considered.

In section 5.3 and 5.4, we consider the design of bandpass and bandstop WDFs respectively. Finally, in section 5.5 a synthesis procedure is briefly described which enables one to design any frequency selective WDFs based on lc-ladder referenece filters with inserted unit elements [42]. Here, the UEs do contribute to the response of the filters while in the method described in chapter four, they were redundant.

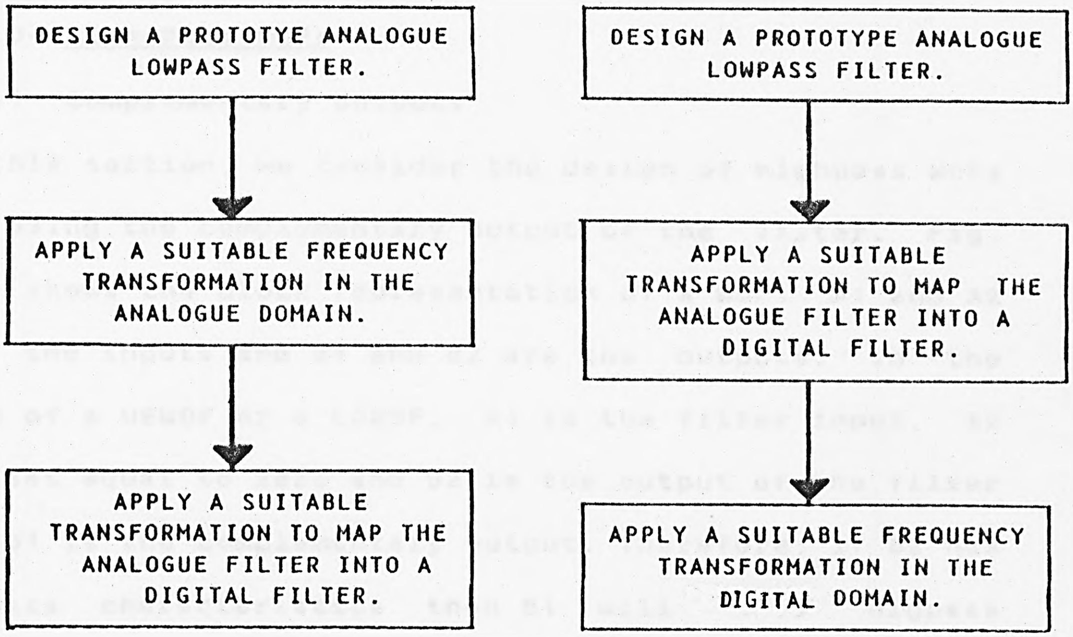

DESIGN A PROTOTYE ANALOGUE LOWPASS FILTER.
APPLY A SUITABLE FREQUENCY TRANSFORMATION IN THE ANALOGUE DOMAIN.
APPLY A SUITABLE TRANSFORMATION TO MAP THE ANALOGUE FILTER INTO A DIGITAL FILTER.
DESIGN A PROTOTYPE ANALOGUE LOWPASS FILTER.
APPLY A SUITABLE TRANSFORMATION TO MAP THE ANALOGUE FILTER INTO A DIGITAL FILTER.
APPLY A SUITABLE FREQUENCY TRANSFORMATION IN THE DIGITAL DOMAIN.

Fig. 5.1- Two procedures for the design of frequency selective digital filters.

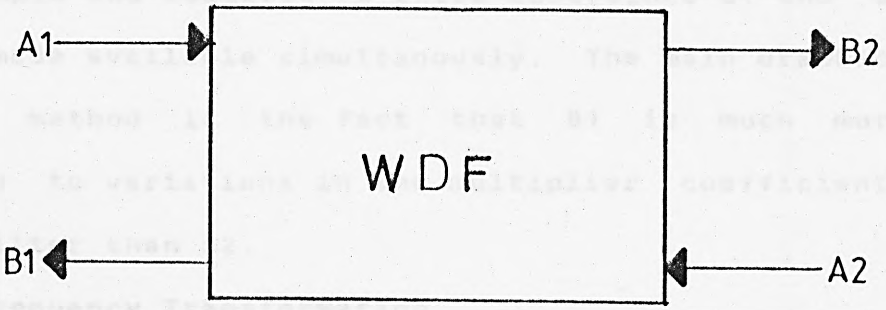


Fig. 5.2- General block representation of a WDF.

5.2.0- High-Pass WDFs

5.2.1- Complementary Outputs

In this section, we consider the design of highpass WDFs by using the complementary output of the filter. Fig. 5.2 shows the block representation of a WDF. A1 and A2 are the inputs and B1 and B2 are the outputs. In the case of a UEWDF or a LCWDF, A1 is the filter input, A2 is set equal to zero and B2 is the output of the filter and B1 is the complementary output. Therefore, if B2 has lowpass characteristics then B1 will have higpass characteristics. This is shown in Fig. 5.3a and Fig. 5.3b for a UEWDF and a LCWDF respectively. For a LTWDF, A1 is the input, A2 is set equal to zero and B1 is taken as the output while B2 is the complementary output. Fig. 5.3c shows the characteristics of B2 and B1 when the filter is designed as a lowpass filter. This approach is very simple and requires no extra cost since B1 and B2 can be made available simultanously. The main drawback of this method is the fact that B1 is much more sensitive to variations in the multiplier coefficients of the filter than B2.

5.2.2- Frequency Transformation

The simplest lowpass to highpass transformation may be obtained by replacing z^{-1} by $-z^{-1}$ [93,94] in the WDF structure. This is a special case of a more general lowpass to highpass transformation which is given as,

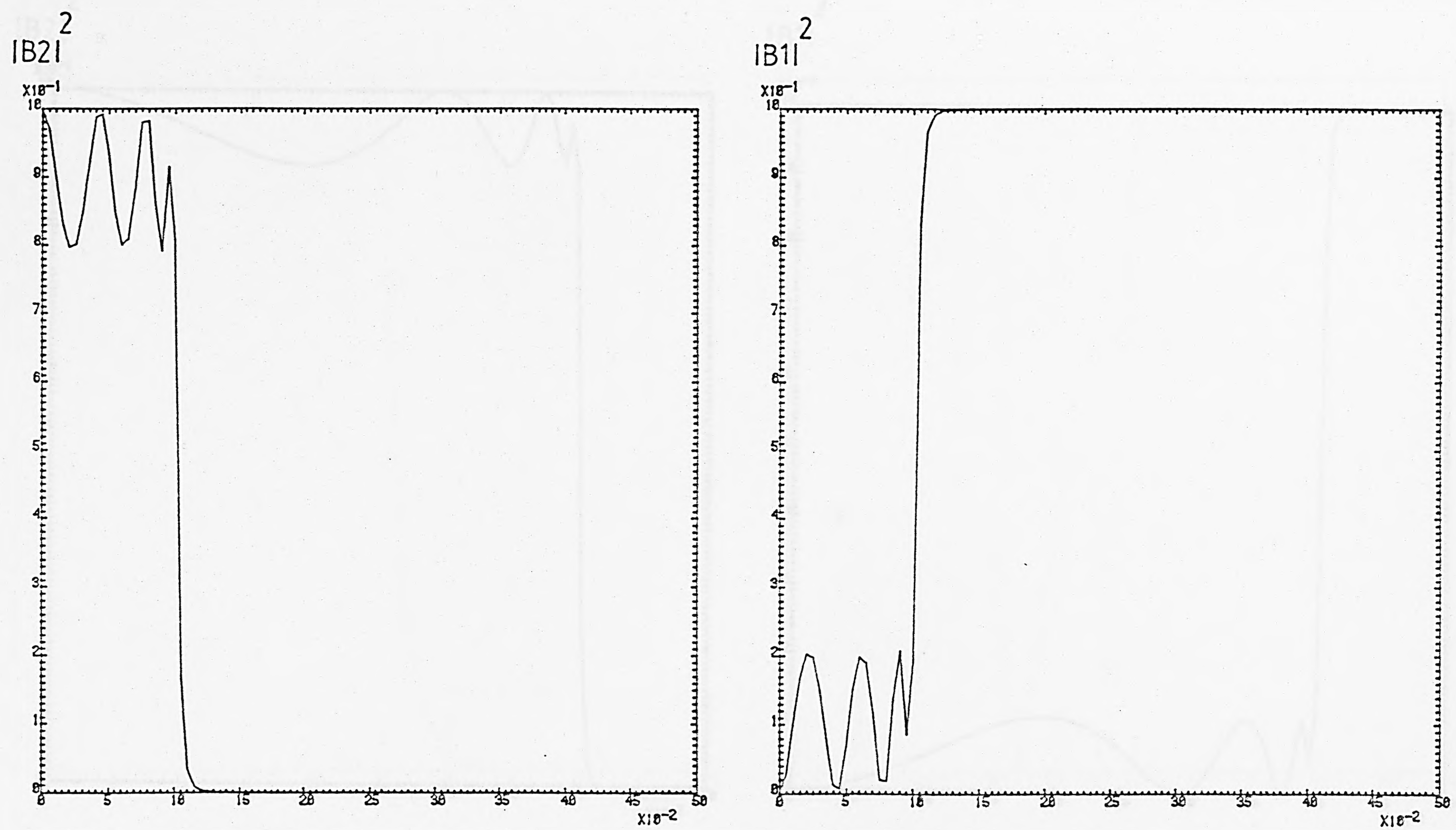


Fig. 5.3 _ a) The output and the complementary output of a UEWDF .

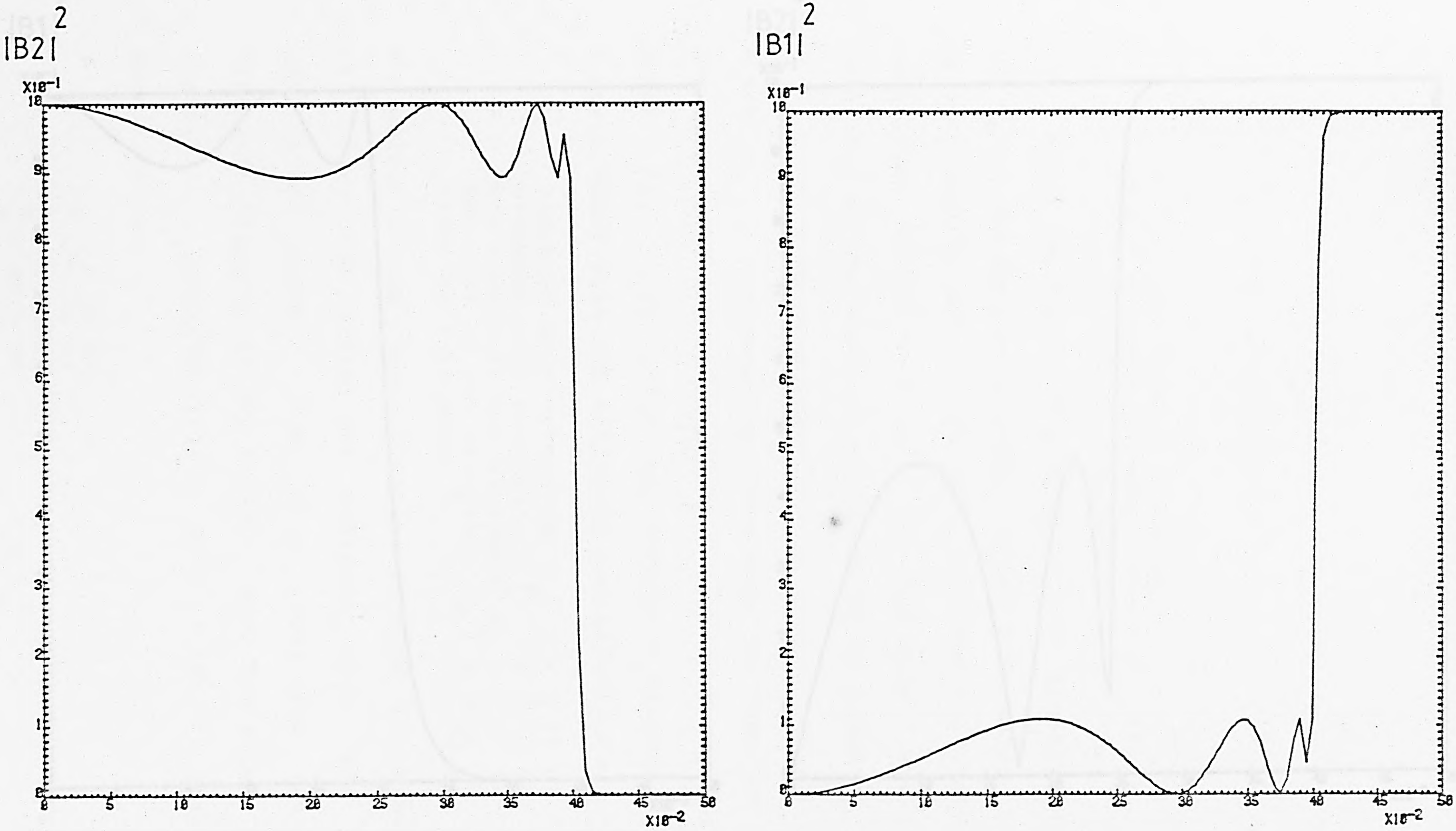


Fig. 5.3 _ b) The output and the complementary output of a LCWDF .

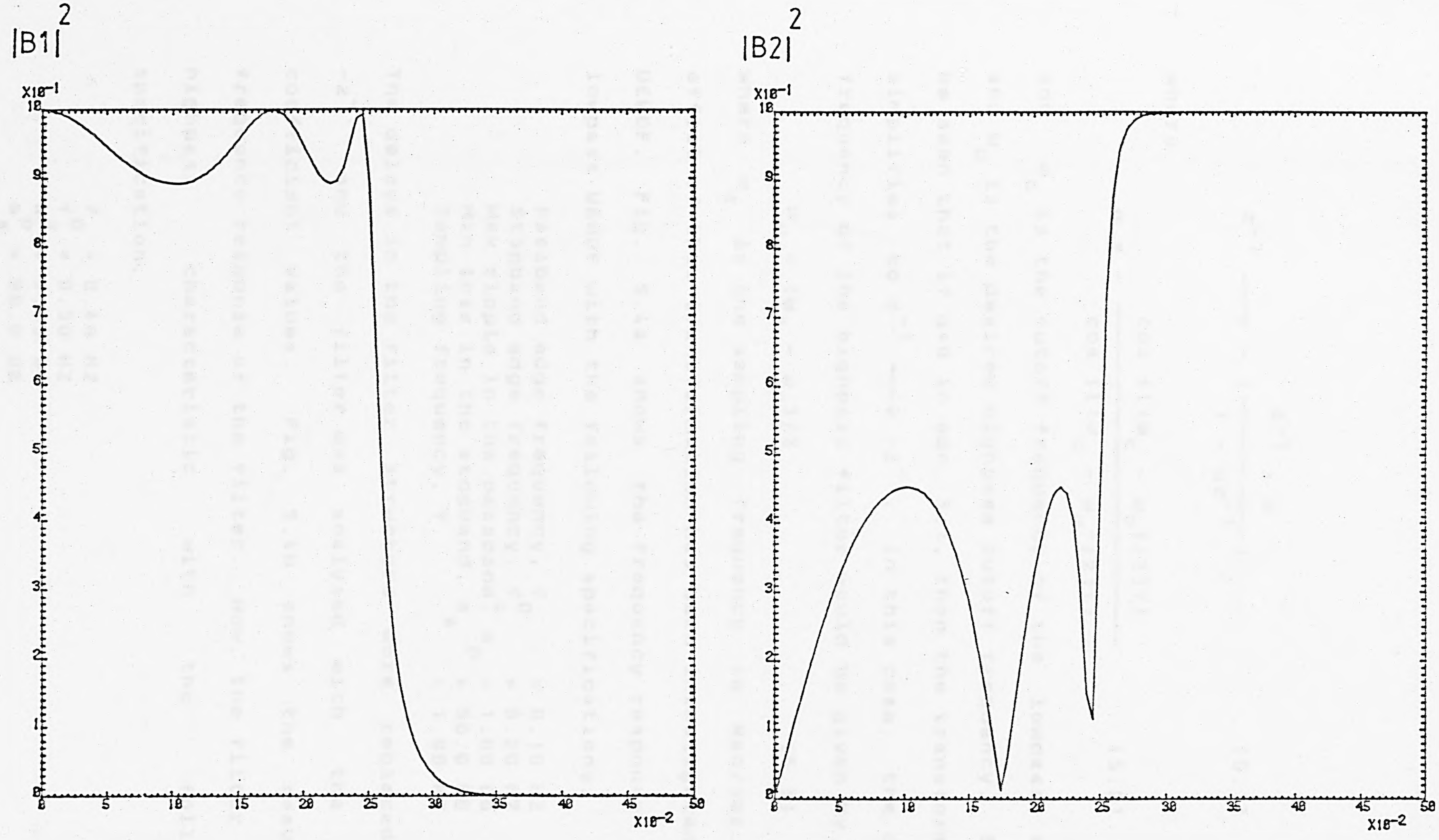


Fig. 5.3 _ c) The output and the complementary output of a LTWDF.

$$z^{-1} \longrightarrow -\left(\frac{z^{-1} - \alpha}{1 - \alpha z^{-1}}\right) \qquad (5.1)$$

where

$$\alpha = - \frac{\cos\{[(\omega_c - \omega_u)/2]T\}}{\cos\{[(\omega_c - \omega_u)/2]T\}} \qquad (5.2)$$

and ω_c is the cutoff frequency of the lowpass filter and ω_u is the desired highpass cutoff frequency. It can be seen that if α=0 in eqn. 5.1, then the transformation simplifies to $z^{-1} \longrightarrow -z^{-1}$. In this case, the cutoff frequency of the highpass filter would be given by,

$$\omega_u = (\omega_s - \omega_c)/2 \qquad (5.3)$$

where ω_s is the sampling frequency in Rad/Sec. The effect of this transformation has been investigated on a UEWDF. Fig. 5.4a shows the frequency response of a lowpass UEWDF with the following specifications,

Passband edge frequency, f_p = 0.10 HZ
Stopband edge frequency, f_s = 0.20 HZ
Max ripple in the passband, a_p = 1.00 DB
Min loss in the stopband, a_s = 50.0 DB
Sampling frequency, f = 1.00 HZ

The delays in the filter structure were replaced with $-z^{-1}$ and the filter was analysed with the same coefficient values. Fig. 5.4b shows the resulting frequency response of the filter. Now, the filter has a highpass characteristic with the following specification,

f_p = 0.40 HZ
f_s = 0.30 HZ
a_p = 1.00 DB
a_s = 50.0 DB

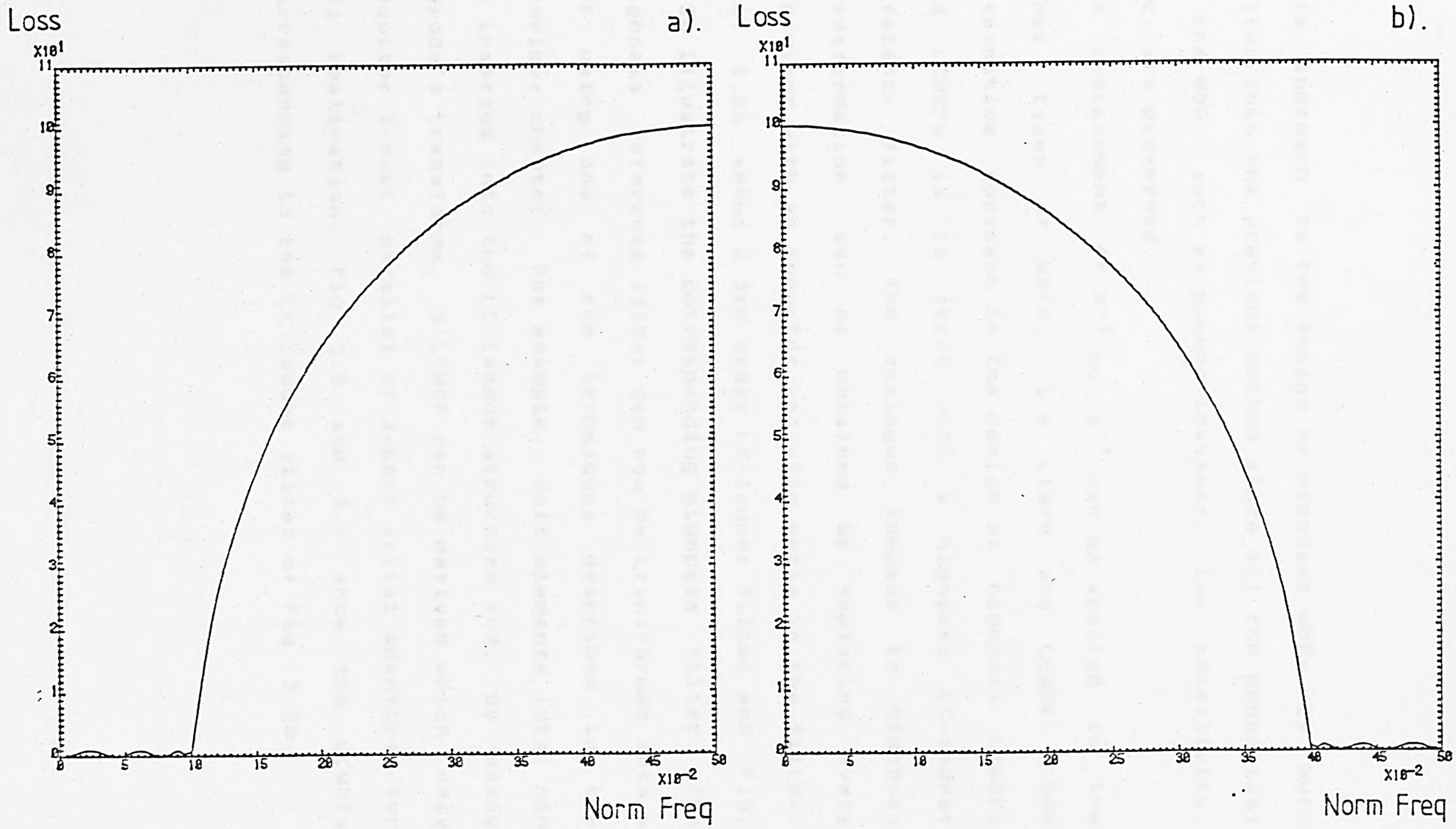
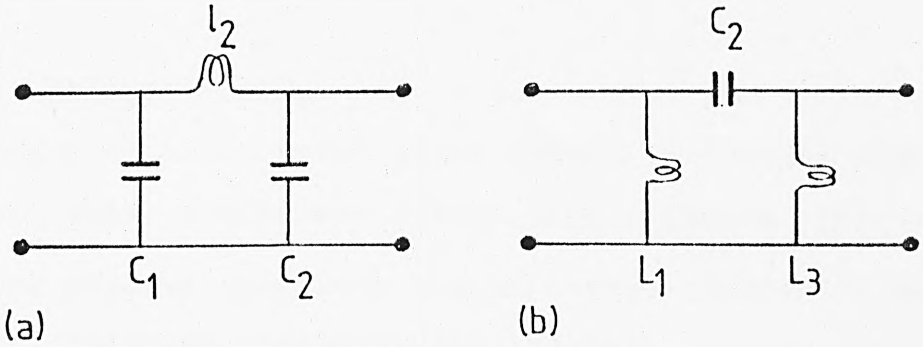


Fig. 5.4 – a) Low-Pass UEWDF . b) Corresponding High-Pass filter ($Z^{-1} \longrightarrow -Z^{-1}$).

This approach to the design of highpass WDFs is much better than the previous method since all the properties of the WDF, such as pseudo-lossless, low sensitivity, etc, are preserved.

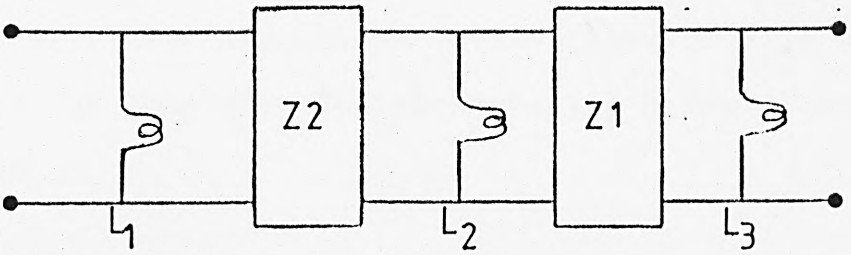
The replacement of z^{-1} by $-z^{-1}$ can be applied to the other types of WDFs, i.e LTWFD and LCWDF. One alternative approach to the design of highpass LTWDFs and LCWDFs is to start with a highpass LC-ladder reference filter. The analogue lowpass to highpass transformation can be obtained by replacing every capacitor with an inductor and vice versa in the filter. Fig. 5.5a shows a 3rd order LC-ladder filter and Fig. 5.5b illustrate the corresponding highpass filter. The highpass reference filter can now be transformed into a WDF using one of the techniques described in the previous chapter. For example, unit elements (UEs) can be inserted into the LC-ladder structure and, by using Kuroda's transforms, a LCWDF can be derived which only requires 3-port parallel or 3-port serial adaptors for its realisation. Fig 5.6 and 5.7 show the LCWDFs corresponding to the LC-ladder filter of Fig. 5.5b.



$L_k=1/W_uC$, $C_k=1/W_ul$

Fig. 5.5 _ a) 3rd order Low pass LC_ladder filter.

b) Corresponding 3rd order High_pass LC_ladder filter.



3rd order Highpass filter with UEs.

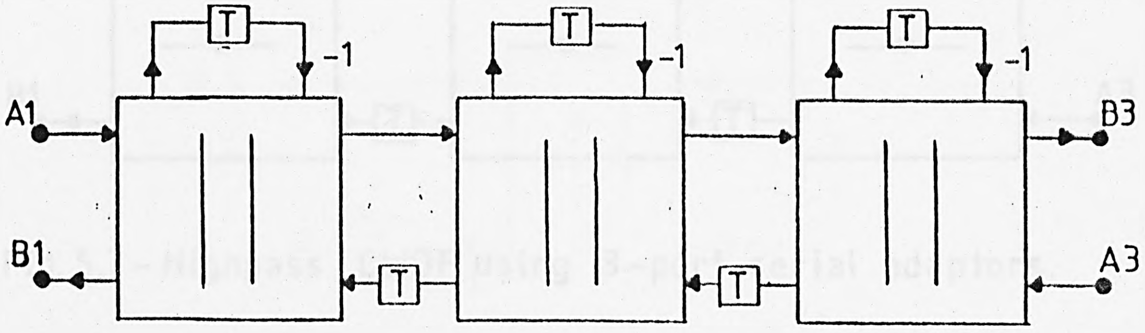
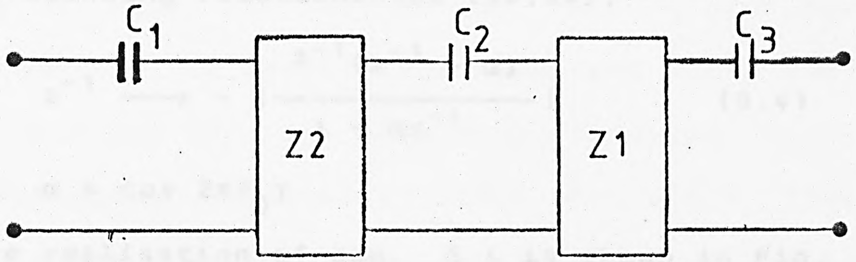


Fig. 5.6 - Highpass LCWDF using 3-port parallel adaptors.



3rd order Highpass filter with UEs.

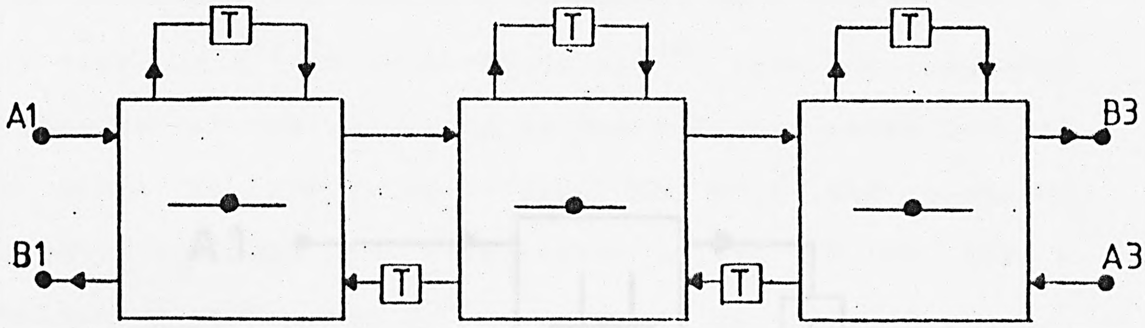


Fig. 5.7 - Highpass LCWDF using 3-port serial adaptors.

5.3.0- <u>Band-Pass WDFs</u>

Suppose that it is required to convert a lowpass digital filter into a bandpass filter with a centre frequency f_0. The digital frequency transformation needed is given by the following relationships [93,94],

$$z^{-1} \longrightarrow -\left(\frac{z^{-1}(z^{-1}-\alpha)}{1-\alpha z^{-1}}\right) \qquad (5.4)$$

where $\alpha = \cos 2\pi f_0 T$

The wave realisation of eqn. 5.4 is shown in Fig. 5.8. Therefore, in order to transform a lowpass WDF into a bandpass WDF, the delays in the filter structure must be replaced by this new block. This block is in fact equivalent to the wave realisation of a parallel tuned circuit [2].

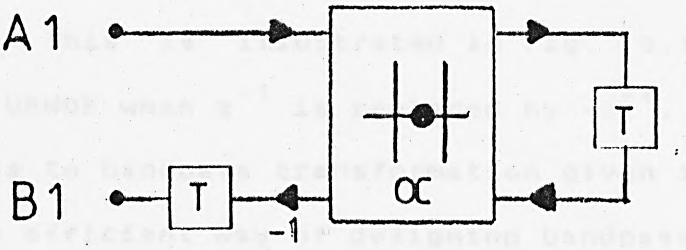


Fig.5.8 - Wave realisation of eqn. 5.4.

This transformation is done for the UEWDFs and Fig. 5.9a and 5.9b show the transformation of a lowpass UEWDF into a bandpass UEWDF with f_0=0.20 HZ. A simple form of this lowpass to bandpass transformation can be obtained by setting α equal to zero in eqn. 5.4. This form of transformation is particulary important since the loss characteristics of the resulting bandpass filter is arithmeatically symmetrical about the centre frequency of f/4, where f is the sampling frequency of the filter. In this case, the transformation simplifies to,

$$z^{-1} \longrightarrow -z^{-2} \qquad (5.5)$$

Fig. 5.10a shows the bandpass filter obtained by appling the transformation given by eqn. 5.5 to the lowpass filter of Fig. 5.4a.

The effect of replacing z^{-1} by $-z^{-2}$ in a digital filter is to reduce the sampling frequency by a factor of 2. In fact, if z^{-1} is replaced by $-z^{-2m}$, then the frequency response of the resulting bandpass filter would contain m bands in the range of zero to half the sampling frequency. This is illustrated in Fig. 5.10b for a multiband UEWDF when z^{-1} is replaced by $-z^{-4}$.

The lowpass to bandpass transformation given in eqn. 5.4 is not an efficient way of designing bandpass LTWDFs or LCWDFs. This is due to the fact that after the transformation from lowpass to bandpass the resulting structures would become very complex since every delay

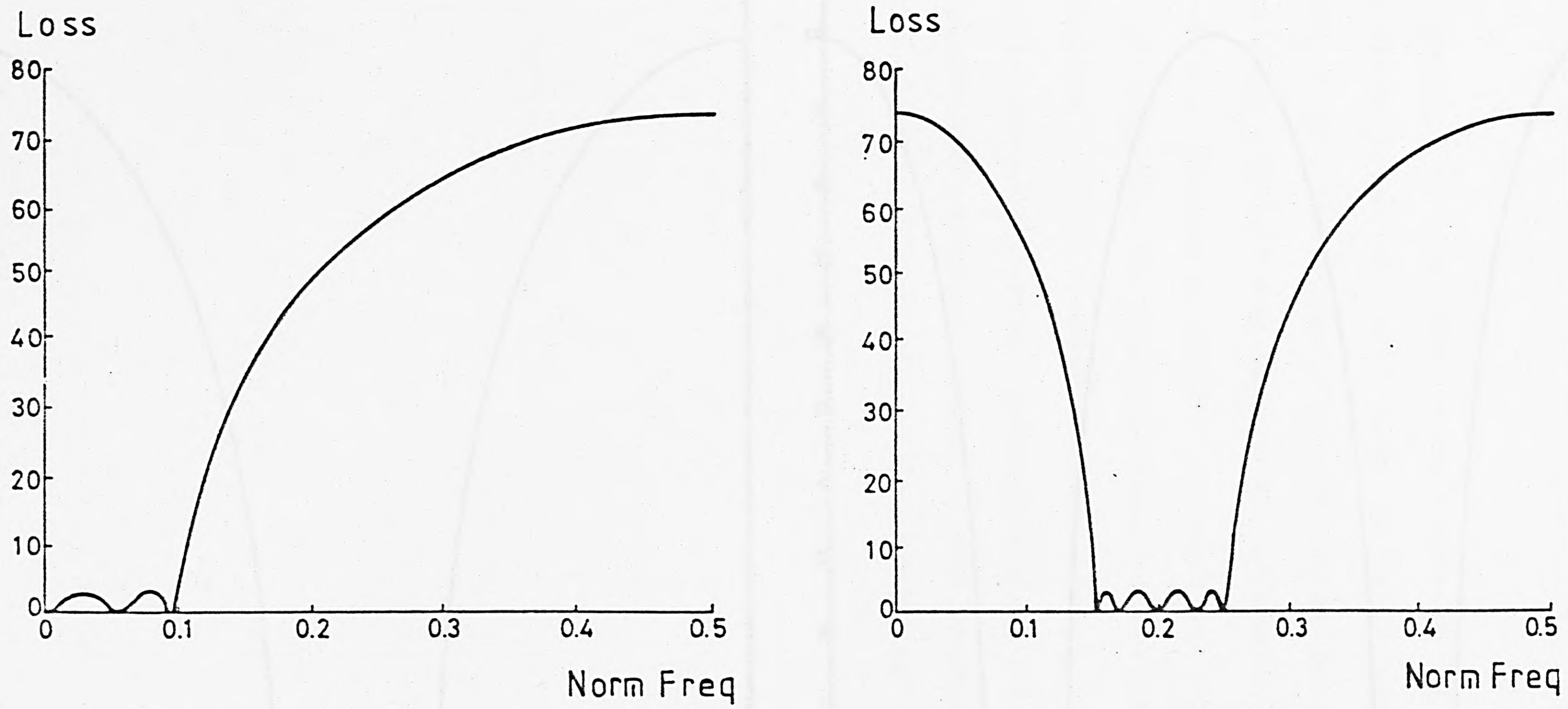


Fig. 5.9 _ a) 5th order UEWDF.

b) Corresponding 10th order Bandpass UEWDF $f_0 = 0.2$.

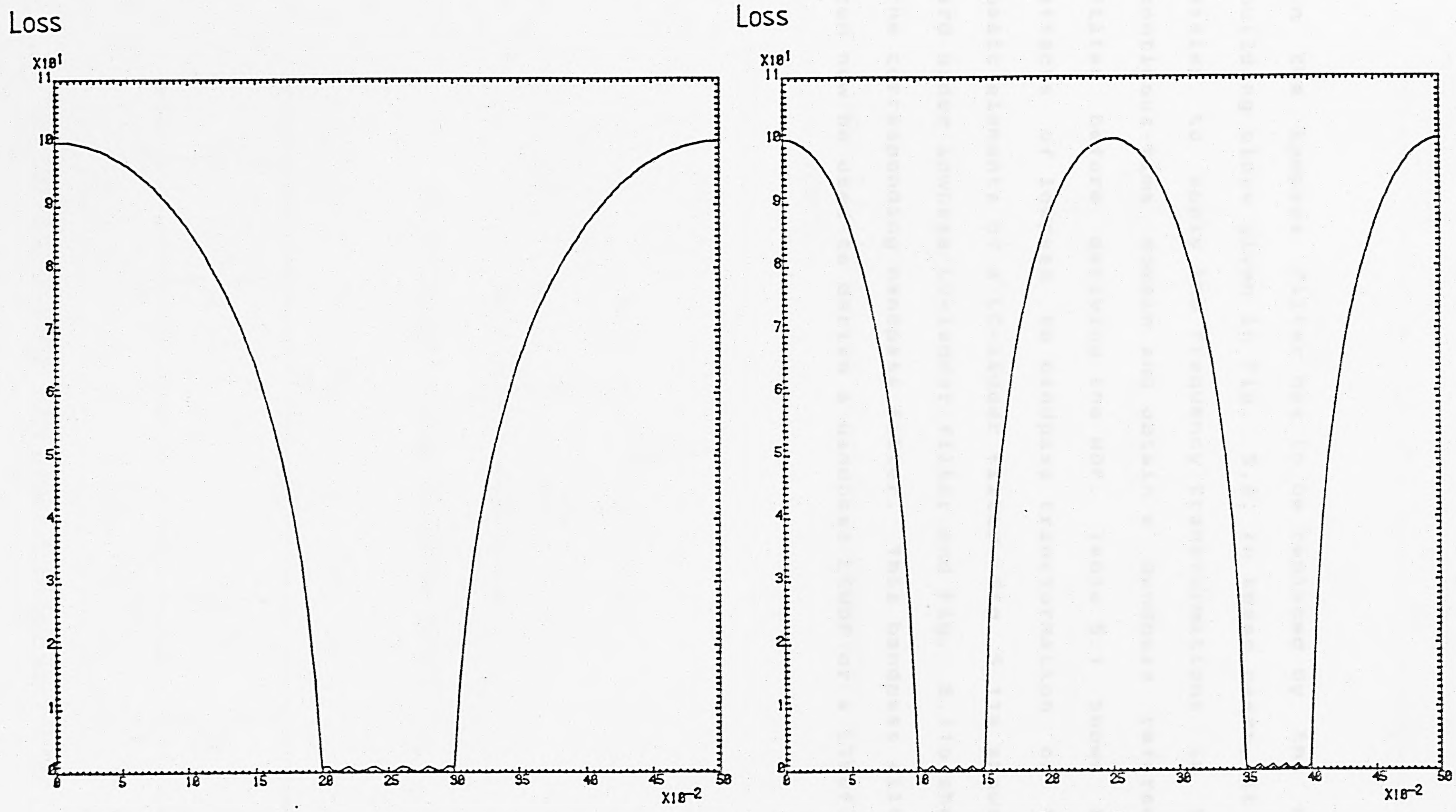
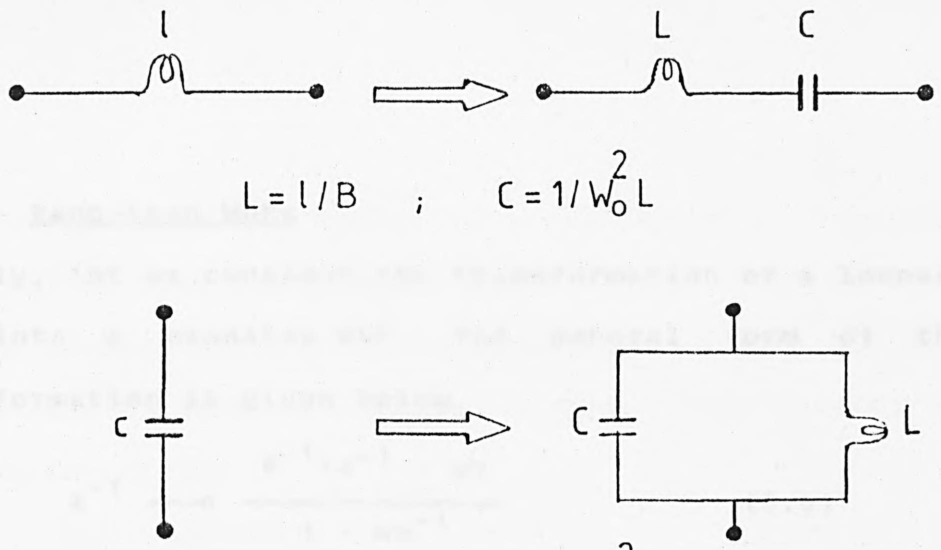


Fig. 5.10 _ a) Band_Pass UEWDF Corresponding to Fig.5.4a ($Z^{-1} \longrightarrow -Z^{-2}$).

b) Multi_band UEWDF ($Z^{-1} \longrightarrow -Z^{-4}$).

in the lowpass filter has to be replaced by the new building block given in Fig. 5.8. In these cases, it is easier to apply the frequency transformations in the continous-time domain and obtain a bandpass reference filter before deriving the WDF. Table 5.1 shows the effects of lowpass to bandpass transformation on the basic elements of a LC-ladder filter. Fig. 5.11a shows a 3rd order lowpass LC-ladder filter and Fig. 5.11b shows the corresponding bandpass filter. This bandpass filter can now be used to derive a bandpass LCWDF or a LTWDF.



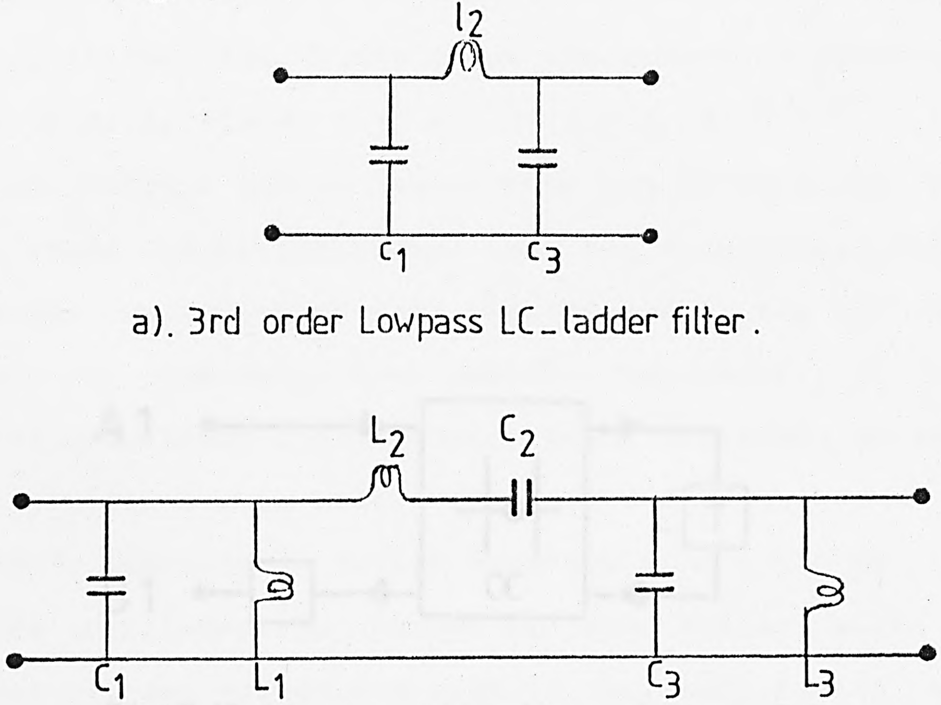
$L = l/B$; $C = 1/W_o^2 L$

$C = c/B$ $L = 1/W_o^2 C$

B is the Bandwidth of the Band_pass filter.

Table. 5.1 _ Effects of LP ⟶ BP Transformation on inductors and capacitors.

Fig. 5.11



a). 3rd order Lowpass LC_ladder filter.

b) Corresponding 6th order Bandpass filter.

5.4.0- Band-Stop WDFs

Finally, let us consider the transformation of a lowpass WDF into a bandstop WDF. The general form of the transformation is given below,

$$z^{-1} \longrightarrow \frac{z^{-1}(z^{-1} - \alpha)}{1 - \alpha z^{-1}} \tag{5.6}$$

where $\alpha = \cos 2\pi f_0 T$

and f_0 is the centre frequency of the resulting bandstop filter. The wave realization of this new block is equivalent to that of a series tuned circuit [2] and is shown in Fig. 5.12.

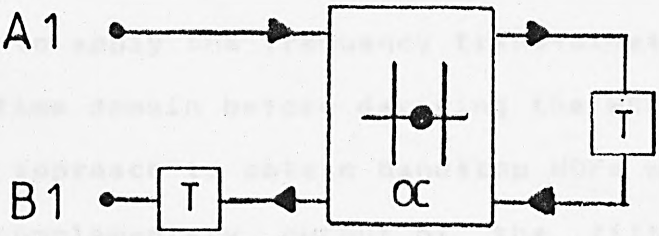


Fig. 5.12-Wave realisation of eqn. 5·6.

Fig. 5.13b show the frequency responses of a bandstop UEWDF filter when the lowpass UEWDF of Fig. 5.9a is used for the transformation. The simple form of eqn. 5.6 may be obtained by setting α equal to zero. This results in the transformation of,

$$z^{-1} \longrightarrow z^{-2} \qquad (5.7)$$

and the loss characteristics of the resulting bandstop filter would be symmetrical about the centre frequency of (f/4). Fig. 5.14a shows the frequency response of the bandpass UEWDF when the transformation of eqn. 5.7 is applied to the lowpass UEWDF of Fig. 5.4a. Reducing the power of z by a factor of 2 results in an increase in the number of bands in the frequency response of the bandstop filter. Fig. 5.14b shows the frequency response of the bandstop filter when z^{-1} is replaced by z^{-4}.

Bandstop Lattice and LC-ladder WDFs can be obtained by using these transformations, but the resulting filter structures would become very complex and it may not be possible to implement them easily. Therefore, it is preferable to apply the frequency transformations in the continous-time domain before deriving the WDFs.

One other approach to obtain bandstop WDFs would be to use the complementary output of the filter when a bandpass filter has been designed. The sensitivity of the resulting bandstop filter would not however be as good as that of the bandpass filter.

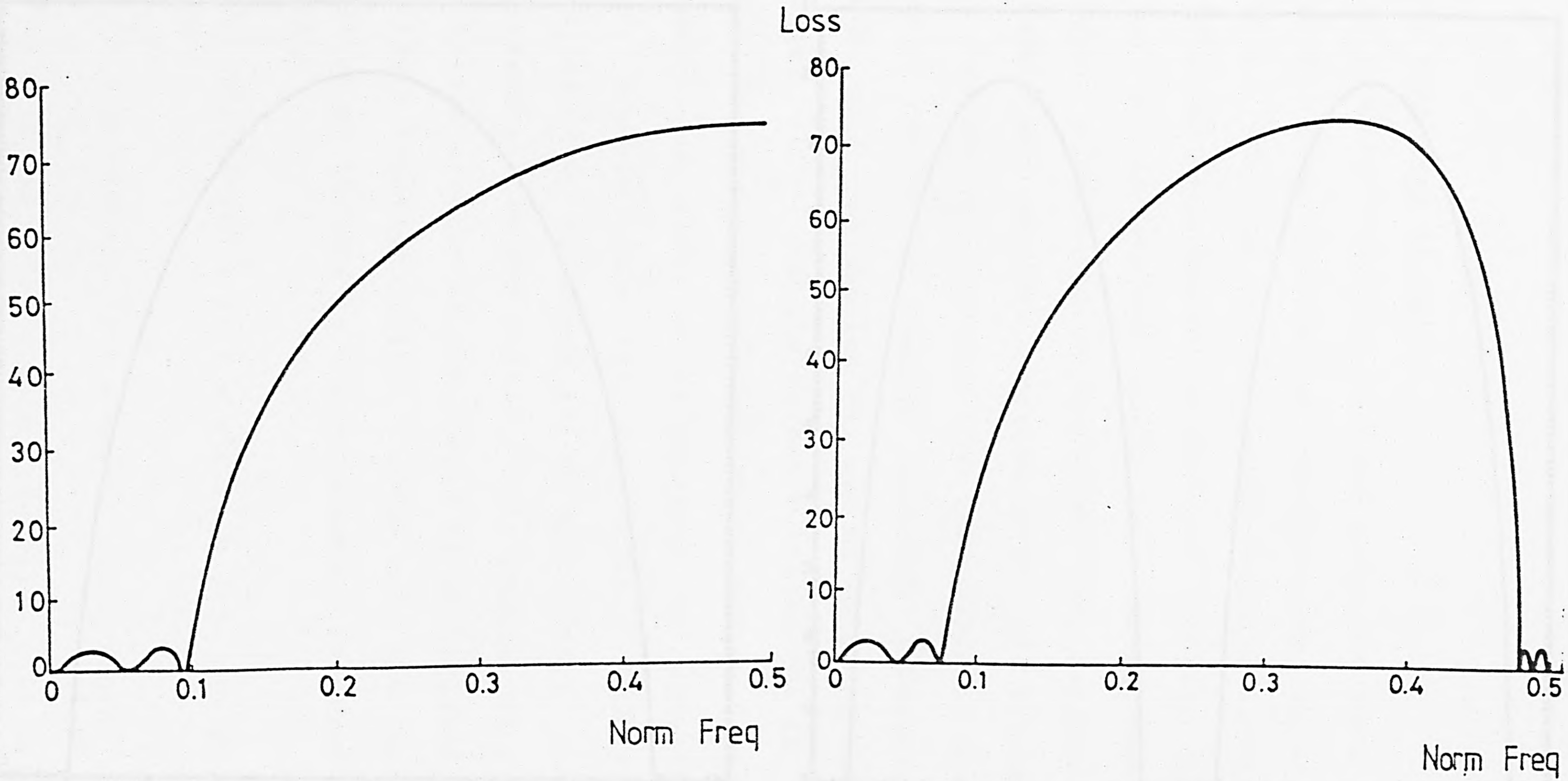


Fig. 5.13 _ a) 5th order UEWDF.

b) Corresponding 10th order Band_Stop UEWDF with $f_o = 0.35$.

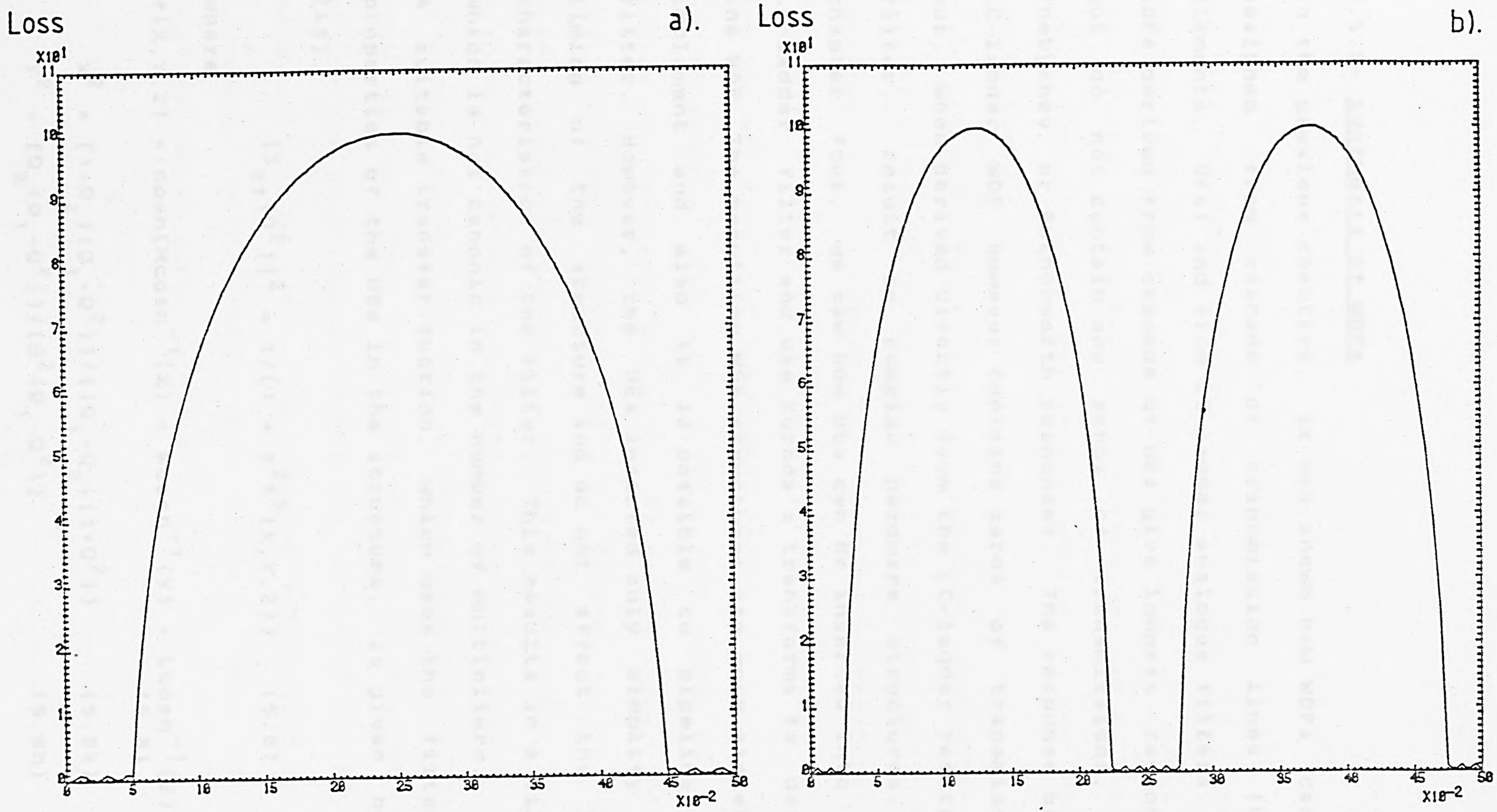


Fig . 5.14 _ a) Band_Stop UEWDF Corresponding to Fig. 5.4 a ($Z^{-1} \longrightarrow Z^{-2}$) .
b) Multi_band UEWDF ($Z^{-1} \longrightarrow Z^{-4}$) .

5.5.0- Synthesis of WDFs

In the previous chapters, it was shown how WDFs can be designed from cascade of transmission lines (Unit Elements, UEs) and from LC-ladder analogue filters. The WDFs derived from cascade of UEs give lowpass responses but do not contain any zeros of transmissions, i.e Chebyshev or Butherworth responses. The response of a LC-ladder WDF however contains zeros of transmission but, when derived directly from the LC-ladder reference filter, result in complex hardware structures. In chapter four, we saw how UEs can be inserted into the LC-ladder filter and use Kuroda's transforms to derive the WDF. The resulting WDF structures are much easier to implement and also it is possible to pipeline the filter. However, the UEs inserted only simplify the timing of the structure and do not affect the loss characteristics of the filter. This results in a filter which is not canonic in the number of multipliers.

A suitable transfer fuction, which uses the filtering properties of the UEs in the structure, is given below [42],

$$|S_{21}(\Omega^2)|^2 = 1/[1 + \varepsilon^2 f^2(X,Y,Z)] \qquad (5.8)$$

where

$$f(X,Y,Z) = \cosh[N\cosh^{-1}(X) + K\cosh^{-1}(Y) + L\cosh^{-1}(Z)] \qquad (5.8)$$

$$X^2 = [1+\Omega_2)(\Omega_1-\Omega^2)]/[(\Omega_1-\Omega_2)(1+\Omega^2)] \qquad (5.9a)$$

$$Y^2 = [\Omega_2(\Omega_1-\Omega^2)]/[\Omega^2(\Omega_1-\Omega^2)] \qquad (5.9b)$$

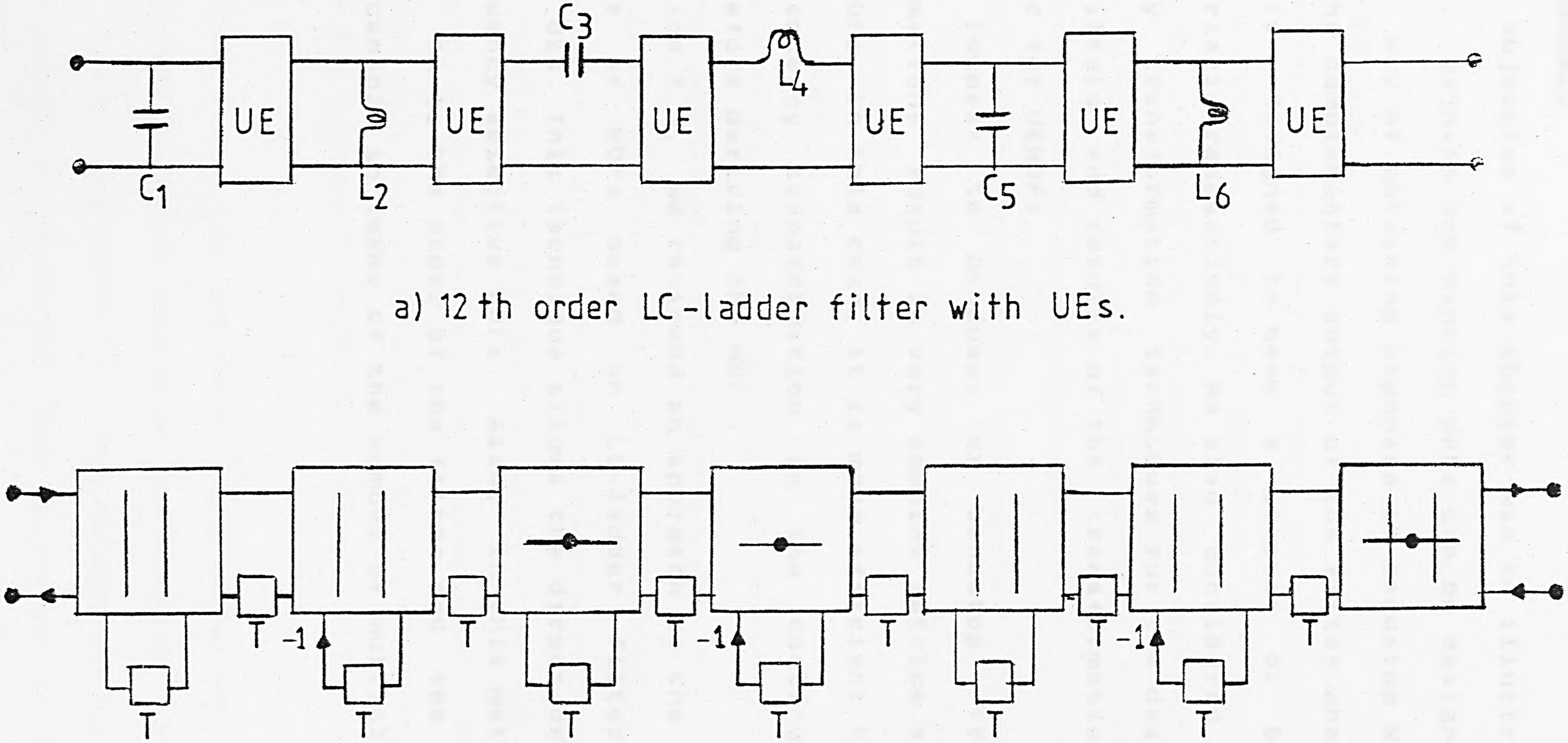
$$Z^2 = (\Omega_1 - \Omega^2)/(\Omega_1 - \Omega_2) \qquad (5.9c)$$

and $$\Omega_1 = \tan(\varphi_1) \ , \ \Omega_2 = \tan(\varphi_2) \qquad (5.10)$$

and N is the number of UEs, K is the number of zeros of transmission at DC and L is the number of zeros of transmission at the Nyquist frequency. φ_1 and φ_2 are the lower and the upper cutoff frequencies of the filter respectively.

This transfer function can be used to synthesise any frequency selective WDFs based on LC-ladder WDFs with inserted UEs. Lowpass and highpass filters can be obtained by setting K=0 and $\varphi_1=0^\circ$ for a lowpass filter and L=0 and $\varphi_2=0^\circ$ for a highpass filter.

As an example, Fig. 15a shows a 12th order LC-ladder filter with inserted UEs [42]. As can be seen, the UEs now contribute towards the order of the filter since there are 3 capacitors, 3 inductors and 6 UEs in the structure which make the filter order equal to 12. This LC-ladder filter can now be transformed into a WDF as shown in Fig. 15b. The 2-port adaptor at the right hand end of the structure is needed due to the nature of the bandpass filter [42]. The detailed explanation of the procedure with which this type of WDFs can be synthesised is given in Ref[43].



a) 12 th order LC-ladder filter with UEs.

b) Corresponding WDF.

Fig.5.15

5.6.0- Summary

The main objective of this chapter was to illustrate how highpass, banpass and banstop WDFs can be designed. The simplest way of obtaining highpass or bandstop WDFs is to use the complementary output of the filter when the filter is designed to have a lowpass or bandstop characteristic respectively. We also considered various frequency transformation techniques for the design of these filters and results of the transformation were presented for UEWDFs.

Digital lowpass to bandpass or bandstop frequency transformations result in very complex lattice and lc-ladder WDFs. In this case, it is more efficient to apply the frequency transformation in the continuos-time domain before deriving the WDF.

In section 5.5, we reviewed an approach to the direct synthesis of WDFs based on LC-ladder filters with inserted UEs. This technique allows the direct design of any frequency selective WDFs. Also, in this method the UEs add on to the order of the filter and the filter will be canonic in terms of the number of multipliers.

CHAPTER SIX

CONCLUSIONS AND COMMENTS

6.1.0- Summary

This thesis presented methods with which finite wordlength WDFs can be designed and illustrated techniques for the translation of the traditional WDF adaptor structures into bit-level systolic arrays which are suitable for VLSI implementation.

The basic theory of WDFs was considered in chapter one. Also, the concept of VLSI array processing was introduced and a number of different parallel arrays were described.

The main basis of the thesis were presented in chapter two. Two subroutines were developed. One was to find an error function for a given set of specifications and filter coefficients and one to implement the direct search method of Hooke and Jeeves. Also three basic bit-level systolic arrays were developed. Finally a universal bit-level systolic array was presented which can be programmed to implement the other three systolic arrays.

The design and the systolic implementation of unit element and lattice WDfs were the subject of chapter three. Using the systolic array developed in chapter two, a 2-port systolic WDF adaptor was designed. The 2-

port systolic adaptor was then constructed using discrete components to prove the correctness of the design. The number of transistors required to implement the adaptor using CMOS technology and the time delay of the adaptor were estimated. The subroutines developed in chapter two were used to construct complete set of programs for the synthesis and finite wordlength design of unit element and lattice WDfs. The hardware implementation of the systolic UEWDFs and LTWDFs were also considered briefly. Finally, a number of examples were presented to illustrate the performance of the design programs and also the simulation of the systolic WDFs.

The design and systolic implementation of LC-ladder WDFs were considered in chapter four. Systolic structures were developed to implement 3-port serial and parellel WDF adaptors. Also, using the universal systolic array of chapter two, a universal systolic WDF adaptor was designed which can be used to realise 2-port, 3-port parallel and 3-port serial adaptors. There are many different approaches for the design of LCWDFs. These approaches were briefly discribed. In this thesis, the design of LCWDFs was achieved by inserting unit elements into the reference LC-ladder filter and using the Kuroda's transforms. A complete program was developed for the synthesis and finite wordlength design of

LCWDFs. The performance of this program was illustrated with the aid of a number of examples. Also, the results from the simulation of the systolic adaptors were presented.

Finally, in chapter five, a number of methods were described for the design of High-Pass, Band-Pass and Band-Stop WDFs. Some of the methods have been examined and the results of the investigations were presented. Also, in chapter five, a synthesis procedure was described for the direct design of LCWDFs. This procedure allows the design of any frequency selective WDFs.

All the arrays described in this thesis are designed by interconnecting regular and simple one-bit processor cells. The interconnections are localized to the nearest neighbouring cells. This is becoming more important as we move towards VLSI implementation of digital signal processing hardware.

A complete set of programs has been developed for the design of WDFs and the simulation of systolic WDFs. The design programs enables one to design finite wordlength WDFs based on Unit-Element, Lattice and LC-ladder filters. Also, it is used to minimize the number of bits for the filter coefficients. From Table 3.1 and 4.1, it can be seen that a small reduction in the number of bits for the coefficients would exponentially reduce the complexity, and consequently the number of transistors, of the systolic WDFs.

From the examples given, it can be seen that the frequency response of the filters cannot be guaranteed to meet the specifications for very short coefficient wordlength, but with the use of the optimization program the responses can be forced to remain within the specifications. Also, the results from the simulations of the systolic WDFs show good agreement with the ideal filter responses. It must be noted that the responses of the systolic WDFs are much closer to the ideal responses when we use the FWLD program coefficients as compared with the direct synthesis coefficients. The single board 2-port systolic adaptor has been tested fully and proved that the design principle is correct.

6.2.0- Further Work

Given below are a list of suggestions for the developement of the ideas presented in this thesis.

1) It would be useful to use a work station, e.g Whitechapel MG.1, to merge the design programs and the analysis programs so that the designer can quickly check the frequency responses of the filters designed. Also, it would be possible to use the windowing facilities of the work station to plot the frequency response of the filter on the screen while changing the parameters of the filter. This would help to investigate the sensitivity of the frequency response of the filter with respect to different filter parameters.

2) It is known that multiplication is the most costly and time consuming operation in a digital filter. It is possible to replace multiplication by arithmatic shifting if the filter coefficients are expressed in terms of powers of 2^{-Q}, where Q is an integer. The finite wordlength design technique presented here has illustrated that in some cases, e.g example 4.3, it is possible to design WDFs with very short coefficient wordlength. Therefore, it would be a good exercise to investigate whether the filter coefficients of WDFs can be expressed in terms of powers of 2^{-Q}. Already work is in progress at Syracuse University [95] for the fabrication of a single chip 2-port adaptor with

programmable shifting facilities. Also, at the City University work is in progress for the design of WDFs with coefficients of powers of 2^{-Q}.

3) In some cases, when the coefficient wordlength of the filter is very small, it is not possible to meet the requirement. It would be useful to investigate the effect of increasing the filter order rather than the coefficient wordlength.

4) In all the design programs presented in this thesis, we have been approximating the magnitude response of the filters. In some applications, it necessary to approximate the phase or both phase and magnitude responses. The existing programs can be easily modified to cover phase and phase/magnitude approximations.

5) It is known [28] that WDFs are very stable and will remain stable if the coefficients are kept in a fixed range depending on the type of reference filter used. Therefore they can be used for adaptive signal processing.

6) In chapter three and four, we stated that it is not possible to pipeline the systolic arrays at adaptor level. This is due to the fact that at the input of the systolic adaptor we need both the LSBs and the MSBs of the inputs. Therefore the delays at the outputs of the cells were removed. It is possible to rearrange the basic cells in such a way that we only need the LSBs of

the inputs at the input of the adaptors [96]. This allows us to pipeline the adaptors at the adaptor level when used to implement a complete filter*.

7) One major drawback of systolic arrays is the global clock required to synchronise the movement of data in the array. This would cause problems when we consider the implementation of large number of the cells on a single VLSI chip. The concept of wavefront arrays has been introduced in chapter one to resolve this problem. However, it is known that wavefront arrays are more complex than the systolic arrays since extra hardware is required for the handshaking between the cells. It would be useful to consider the wavefront implementation of the WDF adaptors and compare them with the arrays designed in this thesis.

8) One major development in the field of systolic WDFs is to consider the VLSI implementation of the systolic adaptors. One adaptor can be fabricated on a single chip and multiplexed to implement a complete filter. The fabrication of the universal systolic adaptor would be useful for experimental purposes.

*It must be noted that if the adaptors are pipelined at the cell level then the data rate of one adaptor would be equal to the time delay of only one basic cell.

6.3.0- Practical Applications of WDFs

Digital filters have been used in many different areas of communication. In most cases, they are implemented using general purpose digital signal processors. Therefore a single chip digital filter can be used to replace these digital signal processors. One good example of a single chip digital filter is the FAD (Filter And Detect) chip developed by British Telecom (BT) [97]. The FAD chip is now used in several BT systems such as System X. A single chip systolic WDF is a potential candidate to replace the FAD chip in BT systems since the reduced coefficient sensitivity of WDFs makes them particulary attractive. Also, due to the systolic nature of the adaptors, it is possible to implement a large number of adaptors on a single VLSI chip.

Lattice WDFs have been used extensively for the design of transmultiplexers [18,19,44-47]. Other areas of interest include channel simulation, channel equalization, audio applications, etc.

One other obvious area in which WDFs may be used is to replace RLC filters in the traditional communication systems. This is due to the fact that WDFs are modelled on these filters and all the properties and the specifications of the RLC filters are preserved after the transformation.

REFERENCES

[1] L.R.Rabiner & B.Gold,"Theory and application of digital signal processing",Prentice-Hall,N.J,1975.

[2] A. Fettweis,"Digital filter structures related to classical filter networks",AEU,25,79-89,1971.

[3] S.S.Lawson,"On a generalization of the wave digital filter concept",Int.j.Circ.Theory.Appl,6,107-120,1978.

[4] A.M.Ali,"Design of low-sensitive digital filter by linear transformation",IEEE,CAS-27,435-444,1980.

[5] A.Peled & B.Liu,"A new approach to realisation of non-recursive digital filters",IEEE,AU-21,477-487,1973.

[6] C.S.Burrus,"Digital filter described by distributed arithmetic",IEEE,CAS-24,674-680,1977.

[7] L.Bruton,"Low-sensitive digital ladder filters",IEEE ,CAS-22,168-176,1975.

[8] W.K.Jenkins,"Recent advances in residue number techniques for recursive digital filters",IEEE,ASS-27,19-30, 1971.

[9] A.S.Sedra & P.O.Brackett,"Filter theory and design : Active and Passive",Matrix Publishers INC,1978.

[10] H.J.Orchard,"Inductorless filters", Electron Lett, 2, 224-, 1966.

[11] R.Saal,"The design of filters using the catalogue of normalized low-pass filters",West Germany,1963.

[12] R.Levy,"Tables of element values for the distributed lowpass filter",IEEE,MTT-13,514-536,1965.

[13] R.Saal and E.Ulbrich,"On the design of filters by synthesis",IRE,CT,284-327, Dec 1958.

[14] G.C.Temes & D.A.Calahan,"Computer-aided network optimization, The State-of-the-Art",IEEE,Proc-55,1832-1863,1967.

[15] C.A.Desoer & S.K.Mitra,"Design of lossy ladder filters by digital computer",IRE,CT-8,192-201,1961.

[16] A.Fettweis,"Pseudopassivity, sensitivity and stabi-

lity of wave digital filters",IEEE,CT-19,668-673,1972.

[17] A.Fettweis,"Round-off noise and attenuation sensitivity in digital filters with fixed-point arithmatic",IEEE,CT-20,174-175,1973.

[18] A.Fettweis,"Design of wave digital filters for communication applications",Proc. IEEE ISCAS,162-165, 1975.

[19] A.Fettweis,"Transmultiplexers with either analog conversion circuit, wave digital filters or SC filters", IEEE,COM-30,1575-1586,1982.

[20] L.Gaszi,"N-port arithmetic unit for DSP",Proc. IEEE ICASSP,707-710,1982.

[21] K.Renner & S.C.Gupta,"On the desgin of wave digital filters with a minimal number of multipliers",IEEE,CAS-21,137-145,1974.

[22] W.Wegener,"On the design of wave digital lattice filters with short wordlength and optimal dynamic range",IEEE,CAS-25,1091-1098,1978.

[23] A.Antoniou & M.G.Rezk,"A comparison of cascade and wave digital filter structures",IEEE,CAS-27,1184-1194, 1980.

[24] K.Renner & S.C.Gupta,"Reduction of round-off noise in wave digital filters",IEEE,CAS-21,305-310,1974.

[25] U.Ullrich,"Round-off noise and dynamic range of wave digital filters",Signal Process,Vol-1,45-64,1979.

[26] A.Fettweis & K.Meerkotter,"Suppression of parasitic oscillations in wave digital filters",IEEE,CAS-22,239-246,1975.

[27] S.S.Lawson,"Digital filter structures from classical analogue networks",PhD. Thesis, University of London, Oct 1975.

[28] A.Fettweis,"Wave digital filters : Theory and practice",Dept. Elect. Eng., Ruhr-Universitit, Bochum, W.Germany, August 1985.

[29] H.J.Carlin,"Distributed circuit design with transmission line elements", Proc. IEEE,Vol-59,1059-1081,1971.

[30] S.S.Lawson,"Implementation of wave digital filter

structures",IEE,PT.G,128,224-226,1981.

[31] K.Meerkotter,"Demostration given at DFG-Kolloquium on digital systems for signal processing", Erlangen, FDR ,March 1974.

[32] P.Lennarz,"Hardware realisation von wellendigital filtern",Frequenz,32,256-261,1978.

[33] E.L.Field & M.A.Soderstrand,"Performance characteristics of digital ladder networks",Proc. IEEE ISCAS, 1121-1124,1980.

[34] T.A.C.M.Claasen, W.F.G.Mecklenbrauker & J.B.H.Peek, "Some consideration on the implementation of digital systems for signal processing",Philips Res. Rep, 30, 73-84,1975.

[35] L.Wanhammar,"Implementation of wave digital filters using distributed arithmetic",Signal Processing, 2, 253-260,1980.

[36] E.Simonyi & A.Heszberger,"A modular approach to multiplexed wave digital matrix filter",Proc. IEEE ISCAS ,1125-1128,1980.

[37] M.Renfors & E.Ziguris,"Signal processing implementation of lattice wave digital filters", Proc. Int. Conf. Digital Signal Processing, Florence, 1984.

[38] L.Gaszi & G.Lucioni,"Lattice wave digital filter realisation using the TMS 320", To be published.

[39] S.S.Lawson,"Wave digital filter hardware structure",IEE,PT.G,128,307-312,1981.

[40] S.S.Lawson,"High-speed single-board flexible hardware implementation of a wave digital filter",IEE,PT.G, 131,17-23,1984.

[41] N.Petrie et al,"General purpose adaptor structure for wave digital filter realisation",Electronic Letters, 19,1038-1039,1983.

[42] M.Reekie et al,"An automated design procedure for frequency selective wave digital filters",Proc. IEEE ISCAS, 258-261,1983.

[43] N.Petrie,"The design and implementation of digital wave filter adaptors",PhD. Thesis, University of Edinbrugh,1985.

[44] R.F.Lyon,"A bit serial VLSI architectural methodology for signal processing",VLSI'81,131-140,1981.

[45] L.J.M.Claesen,"DIGEST:A digital filter evaluation and simulation tool for MOSVLSI filter implementations", IEEE,SC-19,414-424,1984.

[46] J.K.J.VanGinderdeuren,"Compact NMOS building blocks and a methodology for dedicated digital filter applications",IEEE,SC-18,300-316,1983.

[47] L.Claesen, S.Note & K.Mertens,"Flexible and efficient bit-parallel level simulation of hardware implementation of digital filters,Proc. IEEE ISACS,1986.

[48] H.T.Kung,"Why systolic architectures?", IEEE, C-15, 37-46, 1982.

[49] H.T.Kung & C.R.Leiserson,"Algorithms for VLSI processor arrays",in INTRODUCTION to VLSI,Chap 8,1980.

[50] J.G.Nash & C.Petrozolin,"VLSI implementation of a linear systolic array",Proc. IEEE ICASSP,1985.

[51] E.Arnould, H.T.Kung, O.Menzilcioglu & K.Sarocky,"A systolic array computer",Proc. IEEE ICASSP,1985.

[52] J.V.McCanny & J.G.McWhirter,"Implementation of signal processing functions using 1-bit systolic arrays",Electron Lett,18,241-243,1982.

[53] - "Completely iterative, pipelined multiplier array suitable for VLSI",IEE,PT.G,129,40,1982.

[54] J.C.McWhirter et al,"Novel multi-bit convolver/ correlation chip based on systolic array principles", SPIE,341,Real Time Signal Processing V,66-73,1982.

[55] J.C.McWhirter et al,"A CMOS implementation of a systolic multi-bit convolver chip",Proc. VLSI'83, Trondheim,N.Holland,227-235,1983.

[56] D.Wood et al,"An 8-bit serial convolver chip based on a bit level systolic array", Proc. IEEE Custom Integrated Circuit Conf.,256-261,1983.

[57] R.Dettmer,"Chip architectures for parallel processing",IEE,Electronics & Power,227-231,1985.

[58] J.V.McCanny & J.G.McWhirter,"Yiel enhancement of bit level systolic array chips using Fault Tolerant techniques",Electron Lett,19,525-,1983.

[59] S.Y.Kung,"From transversal filter to VLSI wavefront array",Proc. VLSI'83,Trondheim,N.Holland,247-261,1983.

[60] S.Y.Kung,"On supercomputing with systoli/wavefront array processors",IEEE,Proc-72,867-884,1984.

[61] Programming Manual,"OCCAM", INMOS Corporation.

[62] R.Hooke & T.A.Jeeves,"Direct search solution of numerical and statistical problems",J. ASS. Comput. Math,8,212-229,1961.

[63] E.Avenhaus,"On the design of digital filters with coefficients of limited wordlength",IEEE,AU-20,206-212, 1972.

[64] M.Suk & S.Mitra,"Computer-Aided design of digital filters with finite wordlength",IEEE,AU-20,353-356,1972.

[65] F.Brglez,"Digital filter design with short coefficient wordlength",IEEE,CAS-25,1044-1050,1978.

[66] R.E.Grochiere,"A new statistical approach to the coefficient wordlength problem of digital filters",IEEE, CAS-22,190-196,1975.

[67] C.Charalambous & M.J.Best,"Optimization of recursive digital filters with finite wordlengths",IEEE, ASSP-22,424-431,1974.

[68] N.I.Smith,"A random search method for designing finite wordlength recursive digital filters",IEEE,ASSP-27,40-46,1979.

[69] H.Kwan,"On the problem of designing IIR digital filters with short wordlengths",IEEE,ASSP-27,620-624, 1979.

[70] R.Bolton, et al.,"Computer-Aided design of recursive digital filters with coefficients having restricted minimal representation",IEEE,ASSP-29,1205-1208,1981.

[71] F.Catthoor, et al.,"Custom integration of a VLSI transmultiplexer using a Computer-Aided Design methodology based on Bit-serial architectures",Proc. IEEE ISCAS,251-254,1985.

[72] A.Fettweis,"Transmultiplexer with either analog conversion circuits, Wave Digital Filters or SC-filters-A review",IEEE,COM-30,1575-1586,1982.

[73] R.Jain, et al.,"Efficient CAD tools for coefficient optimization of arbitrary integerated digital filters", Proc. IEEE ICASSP,30.11.1-4,1984.

[74] R.Jain, et al.,"Efficient and accurate multiparameter analysis of linear digital filters using a multivariable feedback representation",IEEE,CAS-32,225-235,1985.

[75] L.Gazsi & S.N.Gulluogln,"Discrete optimization of coefficients in CSD code",Proc. IEEE Mediterranean Electrotechnical Conf.,Athens,Greece,1983.

[76] K.Steiglitz,"Designing short-word recursive digital filters",Proc. 9th Annu. Allerton Conf. on Circuit & Syst. Theory,778-788,1971.

[77] "Programs for Digital Signal Processing",IEEE Press, Edited by Digital Signal Processing Committee,1979.

[78] P.R.Cappello & K.Steiglitz,"A note on 'Free Accumulation 'in VLSI filter architectures",IEEE,CAS-32,291-296,1985.

[79] J.G.McWhirter, et al.,"Multi-bit convolution using a bit level systolic array",IEEE,CAS-32,95-99,1985.

[80] A.Corry & K.Patel,"Architecture of a CMOS correlator",Proc. IEEE ISCAS,522-525,1983.

[81] L.Gazsi,"Explicit formulas for Lattice Wave Digital Filters",IEEE,CAS-32,68-88,1985.

[82] K.Renner & S.C.Gupta,"On the design of Wave Digital Filters with low sensitivity properties",IEEE, CT-20,555-567,1973.

[83] L.Young,"Stepped impedence transformers and filter prototype",IRE,MTT-10,339-359,1962.

[84] R.Levy,"Synthesis of microwave systems with distributted parameters",PhD. Thesis, University of London, Oct 1965.

[85] A.Fettweis, et al.,"Wave Digital Lattice Filters", Int. J. Circuit Theory & Appl.,Vol-2,203-211,1974.

[86] V.Belevitch,"Classical network theory",Holden Day, San Francisco,1969.

[87] A.Sedlmeyer & A.Fettweis,"Digital filters with true ladder configuration",Int. J. Cir. Theor. Appl., Vol-1,5-10,1973.

[88] A.Fettweis,"Wave Digital Filters",Proc. Summer School on Circuit Theory,Tale,Czechoslovakia,11-1 to 11-10,1971.

[89] A.J.Greenberger,"Digital transversal filters architectures", Electron Letts,Vol-21,1985.

[90] W.Wegener,"Wave digital directional filters with reduced number of multipliers and adders",AEU,Vol-33, 239-243,1979.

[91] J.A.Nossek & H.D.Schwartz,"Wave digital lattice filters with application in communication systems",Proc. IEEE ISCAS,845-848,1983.

[92] L.Fraiture & J.Neirynck,"Theory of unit element filters",Revue HF7,325-340,1969.

[93] S.S.Lawson,"Frequency transformations for the simple 2-port WDF adaptor chain",IEE,Vol-132,PT.G,237-240,1985.

[94] A.G.Constantinides,"Spectral transformations for digital filters",IEE Proc.,Vol-117,1585-1590,1970.

[95] J.Oldfield & K.Jabbour,"Private communications", Electrical and Computer Eng. Dept,Syracuse Unv.,NY.

[96] A.Goddard,"Comparison of bit-level (CMOS) systolic and wavefront arrays",Internal Report,Centre for Information Eng.,1986.

[97] P.Challener,"FAD-felexibility in digital signal processing",Microprocessors & Microsystem,Vol-7,475-481,1981.

[98] K.A.Owenier,"Optimization of Wave Digital Filters with reduced number of multipliers",AEU,Vol-30,387-393,1976.

[99] J.P.Thiran,"On unit element structures for Wave Digital Filters",IEEE,CAS-24,20-28,1977.

BIBLIOGRAPHY

[1] P.R.Cappello et al,"A systolic vector quantization processor for real time speech coding",IEEE,ISCAS,1986.

[2] R.Chapman et al,"Design strategies for implementing systolic and wavefront arrays using OCCAM",IEEE,ISCAS,85

[3] R.Dettmer,"OCCAM and Transputers",IEE,Electronics & Power,283-287,1985.

[4] J.A.B.Fortes et al,"Systematic approaches to the design of algorithmically specified systolic arrays", IEEE,ISCAS,1985.

[5] J.M.Jover & T.Kailath,"On the analysis of systolic arrays",IEEE,ICASSP,1986.

[6] M.Hatamian & G.L.Cash,"High speed signal processing, pipelining and VLSI",IEEE,ICASSP,1986.

[7] J.V.McCanny,"Mapping system level function onto bit level systolic arrays",IEEE,ICASSP,1986.

[8] D.I.Moldovan,"On the analysis & synthesis of VLSI algorithms",IEEE,C-31,1121-1126,1982.

[9] ----,"On the design of algorithm for VLSI systolic arrays",Proc. IEEE,Vol-71,113-120,1983.

[10] J.G.Nash & C.Petrozolin,"VLSI implementation of a linear systolic array",IEEE,ICASSP,1985.

[11] R.B.Urquhart & D.Wood,"Systolic matrix and vector multiplication methods for signal processing",IEE,131, PT-F,623-631,1984.

[12] M.M.McCabe et.al,"New algorithms and architectures for VLSI",GEC.J. of Science & Tech,Vol-48,68-75,1982.

[13] H.G.Yah,"Kalman filtering and systolic processors", IEEE,ICASSP,1986.

[14] J.V.McCanny et.al,"Optimised bit level systolic array for convolution",IEEE,Vol-131,PT-F,632-637,1984.

[15] J.S.Ward et.al,"Bit level systolic array implementation of the Winograd Fourier Transform algorithm",IEE,Vol-132,PT-F,473-479,1985.

[16] T.Willey et.al,"Systolic implementations for deconvolution, DFT and FFT",IEE,132,PT-F,466-472,1985.

[17] S.S.Lawson et.al,"Design and VLSI implementation of programmable wave digital filters for frequency selective applications", Proposal to the National Science Fondation,Syracuse University,NY,1983.

[18] S.S.Lawson,"On the Hardware implementation of a multiplexed WDF structure",IEEE,ISCAS,124-127,1980.

[19] V.Venkataraman,"LSI implementation of a third order WDF 4-bit-slice",Project report,Electronic & Computer Eng. Dept.,Syracuse University,NY,1984.

[20] B.Sikstrom,"Some aspects on the LSI implementation of WDFs",IEEE,ISCAS,780-783,1982.

[21] L.Gazsi,"DSP-based implementation of a transmultiplexer using WDFs",IEEE,COM-30,1587-1597,1982.

[22] H.Gockler & H.Scheverman,"A modular approach to a digital 60-channel transmultiplexer using directional filters",IEEE,COM-30,1598-1613,1982.

[23] G.R.Walsh,"Methods of Optimization",John Wiley and Sons LTD, 1975.

[24] H.J.Orchard,"Loss sensitivities in singly and doubly terminated Filters",IEEE,CAS-26,293-297,1979.

APPENDIX 'A'

ANALYSIS OF WAVE DIGITAL FILTERS

A1.0- Unit Element WDF Transfer Function

Consider the kth section of a UEWDF as shown in Fig. A1.1. We can write the difference equations of the section as follows,

$$B2_k = A1_k + \alpha_k(z^{-1}A2_k - A1_k) \qquad (A1.1a)$$

and $$B1_k = A2_k + \alpha_k(z^{-1}A2_k - A1_k) \qquad (A1.1b)$$

or $$(1/1-\alpha_k)B2_k = A1_k + (\alpha_k/1-\alpha_k)z^{-1}A2_k \qquad (A1.2a)$$

and $$(1/\alpha_k)B1_k = -A1_k + (1+\alpha_k/\alpha_k)z^{-1}A2_k \qquad (A1.2b)$$

Substituting $A1_k$ from A1.2a into A1.2b, we obtain,

$$B1_k = (1/1-\alpha_k)z^{-1}A2_k - (\alpha_k/1-\alpha_k)B2_k$$

$$B2_k = (\alpha_k/1-\alpha_k)A2_k + (1/1-\alpha_k)B2_k$$

Using matrix notation, we have,

$$\begin{vmatrix} A1_k \\ B1_k \end{vmatrix} = K \begin{vmatrix} 1 & -\alpha_k z^{-1} \\ -\alpha_k & z^{-1} \end{vmatrix} \begin{vmatrix} B2_k \\ A2_k \end{vmatrix} \qquad (A1.3)$$

where $K=(1/1-\alpha_k)$. Eqn. A1.3 represents the ABCD matrix of the kth section in a UEWDF. Now, if N+1 sections are connected together to form an Nth order UEWDF (Fig. A1.2), then the ABCD matrix of the filter, $[ABCD_T]$, can be expressed as,

$$[ABCD_T] = [ABCD_1]\ldots[ABCD_k]\ldots[ABCD_{N+1}] \qquad (A1.4)$$

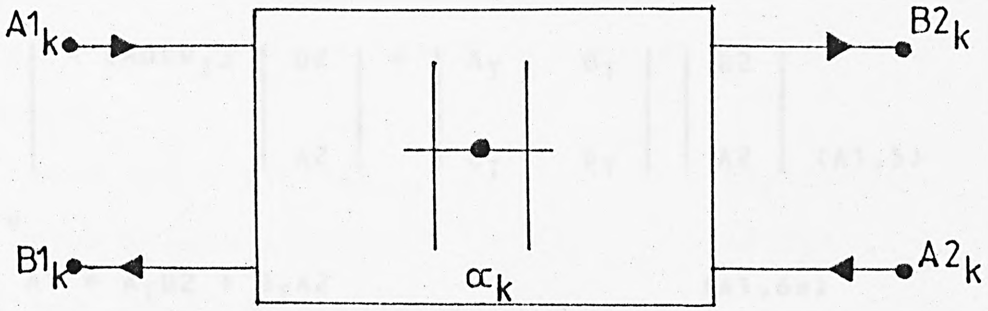


Fig. A1.1- Kth section of a UEWDF.

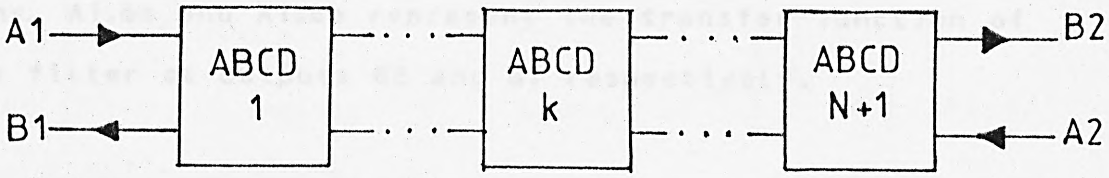


Fig. A1.2- Nth order UEWDF.

Therefore the input/outputs of the filter can be expressed by,

$$\begin{vmatrix} A1 \\ B1 \end{vmatrix} = [ABCD_T] \begin{vmatrix} B2 \\ A2 \end{vmatrix} = \begin{vmatrix} A_T & B_T \\ C_T & D_T \end{vmatrix} \begin{vmatrix} B2 \\ A2 \end{vmatrix} \qquad (A1.5)$$

Therefore

$$A1 = A_T B2 + B_T A2 \qquad (A1.6a)$$

and

$$B1 = C_T B2 + D_T A2 \qquad (A1.6b)$$

But A2=0, therefore,

$$A1 = A_T B2 \qquad (A1.7a)$$

and

$$B1 = C_T B2 \qquad (A1.7b)$$

Dividing eqn. A1.7a by A1.7b, we obtain,

$$\frac{B2}{A1} = \frac{1}{A_T} \qquad (A1.8a)$$

$$\frac{B1}{A1} = \frac{C_T}{A_T} \qquad (A1.8b)$$

Eqns. A1.8a and A1.8b represent the transfer function of the filter at outputs B2 and B1 respectively.

A2.0- Lattice WDF Transfer Function

Consider the first degree all-pass section of Fig. A2.1a. The transfer function of the section can be obtained by substituting A2=B2 in eqn. A1.3. By doing so,we obtain,

$$A1 = K(1 - \alpha z^{-1})A2 \qquad (A2.1a)$$

and

$$B1 = K(-\alpha + z^{-1})A2 \qquad (A2.1b)$$

Dividing eqn. A2.1a by A2.1b, we obtain,

$$G_1(z) = \frac{B1}{A1} = \frac{-\alpha + z^{-1}}{1 - \alpha z^{-1}} \qquad (A2.2)$$

Now consider the second degree all-pass section of Fig. A2.1b. The ABCD matrix of the section can be obatin as follows (using eqn. A1.3),

$$\begin{vmatrix} A1 \\ B1 \end{vmatrix} = K_1 \begin{vmatrix} 1 & -\alpha_1 z^{-1} \\ -\alpha_1 & z^{-1} \end{vmatrix} K_2 \begin{vmatrix} 1 & -\alpha_2 z^{-1} \\ -\alpha_2 & z^{-1} \end{vmatrix} \begin{vmatrix} B2 \\ A2 \end{vmatrix}$$

or

$$\begin{vmatrix} A1 \\ B1 \end{vmatrix} = K_1 K_2 \begin{vmatrix} 1+\alpha_1\alpha_2 z^{-1} & -\alpha_2 z^{-1} - \alpha_1 z^{-1} \\ -\alpha_1 - \alpha_2 z^{-1} & \alpha_1\alpha_2 z^{-1} + z^{-2} \end{vmatrix} \begin{vmatrix} B2 \\ A2 \end{vmatrix} \qquad (A2.3)$$

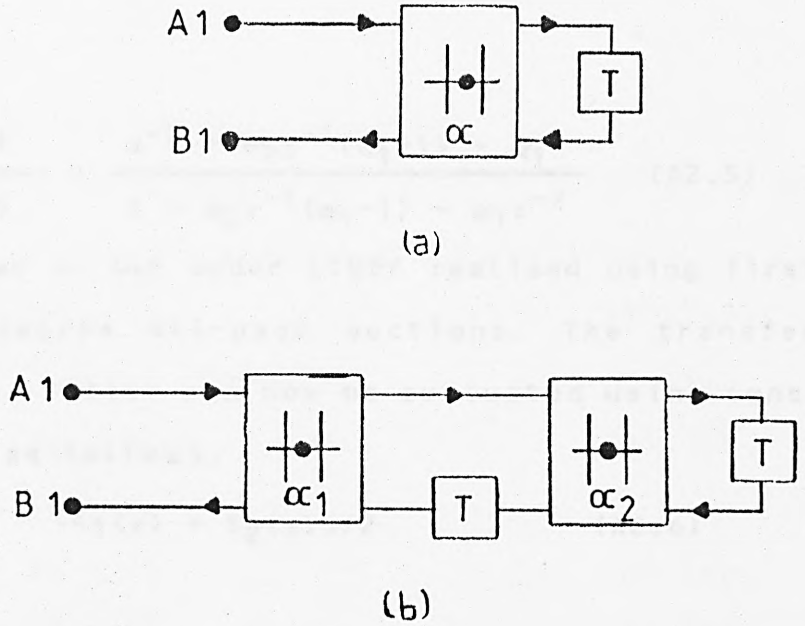
But A2=B2, therefore,

$$A1 = K_1K_2(1 + \alpha_1\alpha_2 z^{-1} - \alpha_2 z^{-1} - \alpha_1 z^{-2})B2 \qquad (A2.4a)$$

and

$$B1 = K_1K_2(-\alpha_1 - \alpha_2 z^{-1} + \alpha_1\alpha_2 z^{-1} + z^{-2})B2 \qquad (A2.4b)$$

By dividing eqn. A2.4a by A2.4b, we obtain the transfer function of a second degree all-pass section,



(a)

(b)

Fig. A2.1 - a) First degree and b) second degree all-pass sections.

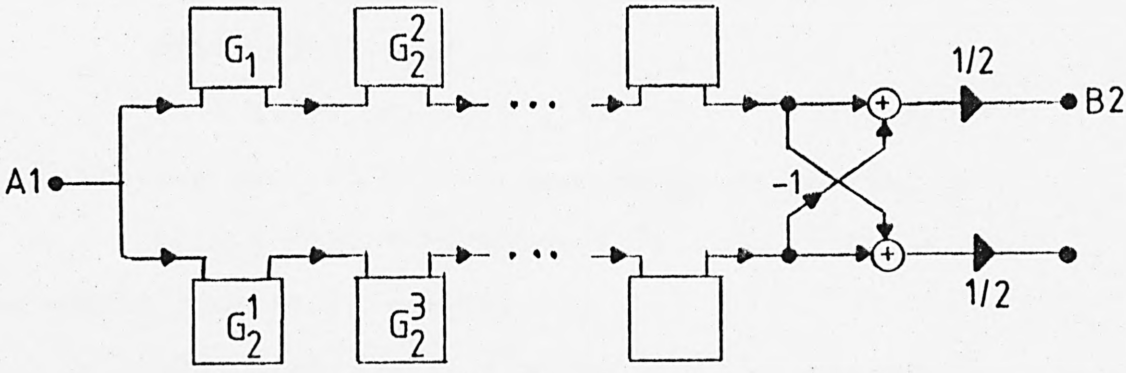


Fig. A2.2- LTWDF realisation using Fig. A.2.1 (a) and (b).

$$G_2(z) = \frac{B1}{A1} = \frac{z^{-2} + \alpha_2 z^{-1}(\alpha_1 - 1) - \alpha_1}{1 + \alpha_2 z^{-1}(\alpha_1 - 1) - \alpha_1 z^{-2}} \qquad (A2.5)$$

Fig. A2.2 shows an Nth order LTWDF realised using first and second degree all-pass sections. The transfer funtion of the filter can now be evaluated using eqns. A2.3 and A2.5 as follows,

$$G(z) = [S_1(z) + S_2(z)]/2 \qquad (A2.6)$$

where

$$S_1(z) = G_1(z) \prod G^k{}_2(z) \qquad k=1,3,\ldots$$

and $$S_2(z) = \prod G^k{}_2(z) \qquad k=2,4,\ldots$$

A3.0- LC-Ladder WDF Transfer Function

The kth section of a LCWDF is shown in Fig. A3.1. The equations of the 3-port adaptor are as follows,

$$Bk = A0 - Ak \qquad (A3.1a)$$

$$A0 = \sum \alpha_k Ak \quad K=1,2,3 \qquad (A3.1b)$$

and $$2 = \alpha_1 + \alpha_2 + \alpha_3 \qquad (A3.1c)$$

Using eqn. A3.1a, we obtain,

$$B1 = A0 - A1 \qquad (A3.2a)$$

$$B2 = A0 - A2 \qquad (A3.2b)$$

$$B3 = A0 - A3 \qquad (A3.2c)$$

Also from Fig. A3.1, we have,

$$A2 = z^{-1}B2 \qquad (A3.3)$$

Now subtracting eqn. A3.2c form A3.2b results in,

$$B2 - B3 = A3 - A2 \qquad (A3.4)$$

Substituting eqn. A3.3 into eqn. A3.4, we obtain,

$$B2(1 + z^{-1}) = A3 + B3$$

or $$B2 = (A3 + B3)/(1 + z^{-1}) \qquad (A3.5)$$

Substituting eqn. A3.5 into eqn. A3.3, we obtain,

$$A2 = z^{-1}(A3 + B3)/(1 + z^{-1}) \qquad (A3.6)$$

Expanding eqn. A3.1b results in,

$$A0 = \alpha_1 A1 + \alpha_2 A2 + \alpha_3 A3 \qquad (A3.7)$$

Now substitute eqn. A3.6 into eqn. A3.7 and then eqn. A3.7 into eqn. A3.2a and A3.2c, we obtain,

$$B1 = \alpha_1 A1+\alpha_2 z^{-1}(A3+B3)/(1+z^{-1})+\alpha_3 A3-A1 \qquad (A3.8a)$$

and $$B3 = \alpha_1 A1+\alpha_2 z^{-1}(A3+B3)/(1+z^{-1})+\alpha_3 A3-A3 \qquad (A3.8b)$$

Subtracting eqn. A3.2c from A3.2a, we obtain,

$$B1 - B3 = A3 - A1 \qquad (A3.9a)$$

or

$$A1 = A3 - B1 + B3 \qquad (A3.9b)$$

Substituting eqn. A3.9b into A3.8a, we obtain,

$$B1 = (\alpha_1 - 1)(A3 - B1 + B3) + \alpha_2 z^{-1}(A3+B3)/(1+z^{-1}) + a_3 A3 \qquad (A3.10)$$

or

$$\alpha_1 B1 = [(\alpha_1 - 1) + (\alpha_2 z^{-1})/(1+z^{-1}) + \alpha_3] A3 + [(\alpha_1 - 1) + (\alpha_2 z^{-1})/(1+z^{-1})] B3 \qquad (A3.11a)$$

Also from eqn. A3.8b, we can write,

$$\alpha_1 A1 = [(1-\alpha_3) - (\alpha_2 z^{-1})/(1+z^{-1})] A3 + [1 - (\alpha_2 z^{-1})/(1+z^{-1})] B3 \qquad (A3.11b)$$

Eqns. A3.11a and 11b can be written in matrix form as shown below,

$$\begin{vmatrix} A1 \\ B1 \end{vmatrix} = K \begin{vmatrix} A & B \\ C & D \end{vmatrix} \begin{vmatrix} A3 \\ B3 \end{vmatrix} \qquad (A3.12a)$$

where

$$K = 1/[\alpha_1(1+z^{-1})] \qquad (A3.12b)$$

$$A = (1-\alpha_3) + z^{-1}(1-\alpha_3-\alpha_2) \qquad (A3.12c)$$

$$B = 1 + z^{-1}(1-\alpha_2) \qquad (A3.12d)$$

$$C = (\alpha_1+\alpha_3-1) + z^{-1}(\alpha_1+\alpha_2+\alpha_3-1) \qquad (A3.12e)$$

and

$$D = (\alpha_1 - 1) + z^{-1}(\alpha_1+\alpha_2-1) \qquad (A3.12f)$$

Using eqn. A3.1c, one of the coefficients, say a_2, can be expressed in terms of the other two coefficients. Therefore the ABCD terms of the matrix simplif y to,

$$A = (1-\alpha_3) + (\alpha_1 - 1)z^{-1} \qquad (A3.13a)$$

$$B = 1 + (\alpha_1+\alpha_3-1)z^{-1} \qquad (A3.13b)$$

$$C = z^{-1} + (\alpha_1+\alpha_3-1) \qquad (A3.13c)$$

and $$D = (\alpha_1-1) + (1-\alpha_3)z^{-1} \qquad (A3.13d)$$

As with the UEWDFs, the transfer function of a complete filter can now be found by using the wave chain matrix method.

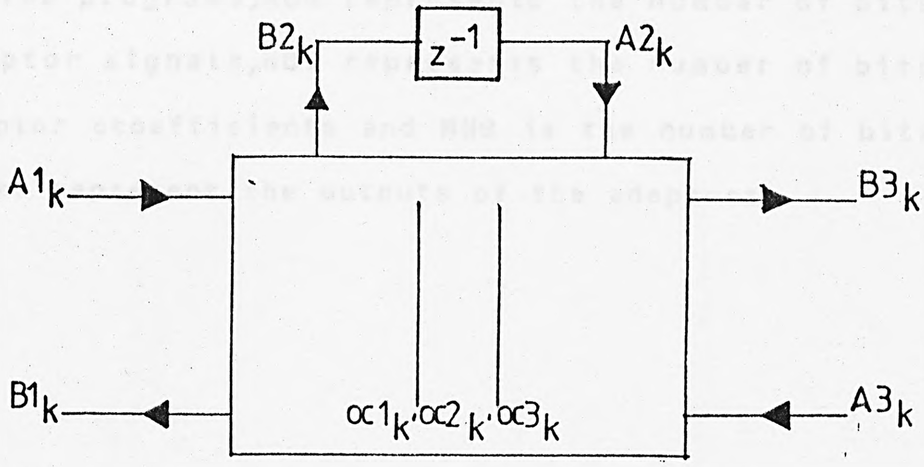


Fig. A3.1 - Kth section of a LCWDF.

APPENDIX 'B'

PROGRAM LISTINGS

B1.0- Systolic Adaptors Programs

In this Appendix, we present the programs which have been developed for the simulation of the systolic WDF adaptors. The program listings are as follows,

B1.1- 2-port systolic adaptor.

B1.2- 3-port parallel systolic adaptor.

B1.3- 3-port serial systolic adaptor.

B1.4- The universal systolic adaptor.

In all the programs,NBS represents the number of bits for adaptor signals,NBC represents the number of bits for adaptor ccoefficients and MNB is the number of bits needed to represent the outputs of the adaptors.

B1.1- 2-PORT SYSTOLIC ADAPTOR PROGRAM

```
      INTEGER IN1(27),IN2(27),ALPHA(8),NBS,NBC,MNB,OUT1,
     &X,U,OUT2,NSECT,NSPT,XK,UKP1,KCOEFF,
     &IPSET(100),OPSET(100),COEFSET(20),USET(22)
      COMMON /C/ NBS,NBC,MNB,NSECT,NSPT
 5    CALL READIP(IPSET,USET,COEFSET)
      DO 20 I=1,NSPT
      XK=IPSET(I)
      DO 10 K=1,NSECT
      UKP1=USET(K+1)
      KCOEFF=COEFSET(K)
      PRINT,'INPUTS TO BLOCK ',XK,UKP1,KCOEFF
      CALL INIT(XK,IN1,UKP1,IN2,KCOEFF,ALPHA,NBS,NBC,MNB)
      CALL BLOCK(IN1,IN2,OUT1,OUT2,ALPHA,NBS,NBC,MNB)
      PRINT,'OUTPUTS FROM BLOCK ',OUT1,OUT2
      USET(K)=OUT2
      XK=OUT1
 10   CONTINUE
      OPSET(I)=OUT1
 20   CONTINUE
      PRINT,
      DO 30 I=1,NSPT
      PRINT,'OUTPUT (XK+1) =',OPSET(I)
 30   CONTINUE
      PRINT,
      DO 40 K=1,NSECT+1
      PRINT,'USET (UK) =',USET(K)
 40   CONTINUE
      PRINT,
      PRINT,
      GOTO 5
      STOP
      END
C
C
C
      SUBROUTINE READIP(XK,UKP1,ALPHA)
      INTEGER XK(100),ALPHA(20),UKP1(22),NSECT,NBC,NBS,MNB,NSPT
      COMMON /C/ NBS,NBC,MNB,NSECT,NSPT
      PRINT,'NO. OF SAMPLES  ORDER  NBS  NBC'
      READ,NSPT,NSECT,NBS,NBC
      PRINT,
      NSECT=NSECT+1
      MNB=NBS+NBC+2
      PRINT,'ENTER INPUT VALUES (X1) :-'
      READ,(XK(I),I=1,NSPT)
      PRINT,
      PRINT,'ENTER THE COEFFICIENTS :-'
      READ,(ALPHA(I),I=1,NSECT)
      PRINT,
      PRINT,'ENTER THE INITIAL VALUES OF U INPUTS :-'
      READ,(UKP1(I),I=1,NSECT+1)
      PRINT,
      RETUPN
      END
C
C
C
      SUBROUTINE INIT(XK,XKA,UKP1,UKP1A,ALPHA,ALPHAA,NBS,NBC,MNB)
      INTEGER XKA(26),UKP1A(26),ALPHAA(8),PU(9,27),PX(9,27),
     &CPU(8,27),CPX(8,27),CS(8,27),DA(8,26),NBS,NBC,MNB,XK,UKP1,ALPHA
      COMMON PU,PX,CPU,CPX,CS,DA
      CALL DTOB(XK,XKA,MNB)
      CALL DTOB(UKP1,UKP1A,MNB)
      CALL DTOB(ALPHA,ALPHAA,NBC)
      DO 5 K1=1,27
      DO 5 K2=1,9
      PX(K2,K1)=0
      PU(K2,K1)=0
 5    CONTINUE
      DO 7 K1=1,26
      DO 7 K2=1,8
      DA(K2,K1)=0
      CPX(K2,K1)=0
      CPU(K2,K1)=0
      CS(K2,K1)=0
 7    CONTINUE
      CPX(NBC,1)=ALPHAA(NBC)
      CPU(NBC,1)=ALPHAA(NBC)
      DO 10 K=2,NBC+1
      PX(K,1)=XKA(K-1)
      PU(K,1)=UKP1A(K-1)
 10   CONTINUE
      DO 20 K=2,(MNB-NBC+1)
      PX((NBC+1),K)=XKA(NBC+K-1)
      PU((NBC+1),K)=UKP1A(NBC+K-1)
 20   CONTINUE
      DO 30 K=1,MNB
      DA(NBC,K)=1
 30   CONTINUE
      DO 40 K=1,NBC
      CS(K,1)=1
 40   CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE DTOB(A,ARRAY,NB)
      INTEGER A,ARRAY(27),NB,R
      IF (A.EQ.0) GOTO 40
      IF (A.LT.0) A=(2**NB)+A
      K=1
 10   IF (A.EQ.1) GOTO 20
      R=INT(A/2)
      ARRAY(K)=1
      IF ((R*2).EQ.A) ARRAY(K)=0
      A=R
      K=K+1
      GOTO 10
 20   ARRAY(K)=1
      DO 30 J=K+1,NB
      ARRAY(J)=0
 30   CONTINUE
      GOTO 60
 40   DO 50 K=1,NB
      ARRAY(K)=0
 50   CONTINUE
 60   RETURN
      END
C
C
C
      SUBROUTINE BLOCK(IN1,IN2,OUT1,OUT2,COEFF,NBS,NBC,MNB)
      INTEGER IN1(27),IN2(27),COEFF(8),XKP1(27),PU(9,27),PX(9,27)
     &,UK(27),CPU(8,27),CPX(8,27),CS(8,27),DA(8,27),NBS,NBC,MNB,XK,
     &UKP1,ALPHA,D,OPX,OPU,NPX,NPU,OCPX,OCPU,NCPX,NCPU,OCS,NCS,J,I,
     &K,NCELL,COUNT,OUT2,OUT1
      COMMON PU,PX,CPU,CPX,CS,DA
      COMMON /B/ OPX,NPX,OPU,NPU,OCPX,NCPX,OCPU,NCPU,OCS,NCS
      NCELL=1
      COUNT=1
      J=1
      DO 30 K=1,MNB
      DO 10 I=NCELL,1,-1
      ALPHA=COEFF(I)
      UKP1=IN2(J)
      XK=IN1(J)
      D=DA(I,J)
      OPX=PX(I+1,J)
      OPU=PU(I+1,J)
      OCPX=CPX(I,J)
      OCPU=CPU(I,J)
      OCS=CS(I,J)
      CALL CELL(ALPHA,XK,UKP1,D)
      PU(I,J+1)=NPU
      PX(I,J+1)=NPX
      CPU(I,J+1)=NCPU
      CPX(I,J+1)=NCPX
      CS(I,J+1)=NCS
      J=J+1
 10   CONTINUE
      UK(K)=PU(1,K+1)
      XKP1(K)=PX(1,K+1)
      J=1
      IF (NCELL.LT.NBC) GOTO 20
      J=J+COUNT
      COUNT=COUNT+1
      GOTO 30
 20   NCELL=NCELL+1
 30   CONTINUE
      CALL BTOD(OUT1,XKP1,MNB)
      CALL BTOD(OUT2,UK,MNB)
      RETURN
```

```
      END
C
C
C
      SUBROUTINE CELL(ALPHA,XK,UKP1,D)
      INTEGER NXK,NS,UKP1,XK,OCS,S,NCS,D,OCPX,NCPX,OCPU,NCPU,OPX,NPX
     &,ALPHA,D,NPU,OPU
      COMMON /B/ OPX,NPX,OPU,NPU,OCPX,NCPX,OCPU,NCPU,OCS,NCS
      NXK=XK
      NS=0
      IF (NXK.EQ.0) NS=1
      NXK=NS
      CALL SUM(UKP1,NXK,OCS,S,NCS)
      IF (D.EQ.0) GOTO 10
      NS=0
      IF (S.EQ.0) NS=1
      S=NS
 10   CALL PRODUCT(OPX,ALPHA,S,OCPX,NPX,NCPX)
      CALL PRODUCT(OPU,ALPHA,S,OCPU,NPU,NCPU)
      RETURN
      END
C
C
C
      SUBROUTINE SUM(A,B,CIN,S,COUT)
      INTEGER A,B,CIN,COUT,S
      S=A+B+CIN
      IF (S.LT.2) GOTO 10
      IF (S.EQ.2) GOTO 20
      S=1
      COUT=1
      GOTO 30
 10   COUT=0
      GOTO 30
 20   S=0
      COUT=1
 30   RETURN
      END
C
C
C
      SUBROUTINE PRODUCT(A,B,C,CIN,P,COUT)
      INTEGER A,B,C,CIN,COUT,P,BANDC
      BANDC=B*C
      CALL SUM(A,BANDC,CIN,P,COUT)
      RETURN
      END
C
C
C
      SUBROUTINE BTOD(A,ARRAY,NB)
      INTEGER A,ARRAY(27),NB,K
      A=0
      DO 10 K=1,NB
      A=A+ARRAY(K)*2**(K-1)
```

```
10    CONTINUE
      IF (ARRAY(NB).EQ.1) A=-((2**NB)-A)
      RETURN
      END
```

B1.2- 3-PORT PARALLEL SYSTOLIC ADAPTOR PROGRAM

```
       INTEGER IN1(28),IN2(28),IN3(28),ALPHA1B(8),NBS,NBC,MNB,OUT1
      &,X,U,OUT2,NSECT,NSPT,XK,UKP1,KCOEFF,ALPHA1,ALPHA2
      &,ALPHA2B(8),OUT3,BITA1B(8),BITA2B(8),BITA1,BITA2,A1,A2,A3
      &,IPSET(100),OPSET(100),COEFSET(20),USET(22)
      &,B1,B2,B3
       CHARACTER*1 CH
       COMMON /A/ IN1,IN2,IN3,ALPHA1B,ALPHA2B,BITA1B,BITA2B
       COMMON /C/ NBS,NBC,MNB
5      CALL READIP(A1,A2,A3,ALPHA1,ALPHA2,BITA1,BITA2)
       PRINT,
       PRINT,'ADAPTOR OUTPUTS USING DIFFERENCE EQUATIONS :-'
       PRINT,
       IA0=A1*ALPHA1+A2*ALPHA2+(2-ALPHA1-ALPHA2)*A3
       PRINT,'B1 =',IA0-A1
       PRINT,'B2 =',IA0-A2
       PRINT,'B3 =',IA0-A3
       PRINT,
       CALL INIT(A1,A2,A3,ALPHA1,ALPHA2,BITA1,BITA2)
       CALL BLOCK(OUT1,OUT2,OUT3)
       PRINT,
       PRINT,'SYSTOLIC ADAPTOR OUTPUTS :-'
       PRINT,
       PRINT,'B1 =',OUT1
       PRINT,'B2 =',OUT2
       PRINT,'B3 =',OUT3
       PRINT,
       PRINT,'DO YOU WISH TO STOP?(Y/N)'
       READ,CH
       IF(CH.EQ.'N') GOTO 5
       STOP
       END
C
C
C
       SUBROUTINE READIP(A1,A2,A3,ALPHA1,ALPHA2,BITA1,BITA2)
       INTEGER A1,A2,A3,ALPHA1,ALPHA2,BITA1,BITA2,NBC,NBS,MNB
       COMMON /C/ NBS,NBC,MNB
       PRINT,
       PRINT,'A1   A2   A3'
       READ,A1,A2,A3
       PRINT,
       PRINT,'ALPHA1 ALPHA2'
       READ,ALPHA1,ALPHA2
       PRINT,
       PRINT,'BITA1,BITA2'
       READ,BITA1,BITA2
       PRINT,'NBS   NBC'
       READ,NBS,NBC
       PRINT,
       MNB=NBS+NBC+3
       RETURN
       END
C
```

```
C
C
      SUBROUTINE INIT(A1,A2,A3,ALPHA1,ALPHA2,BITA1,BITA2)
      INTEGER A1,A2,A3,IN1(28),IN2(28),IN3(28),ALPHA1,ALPHA2,BITA1
     &,BITA2,ALPHA1B(8),ALPHA2B(8),BITA1B(8),BITA2B(8),P1(9,28)
     &,P2(9,28),P3(9,28),DA(8,27),CP1(8,28),CP2(8,28),CP3(8,28)
     &,CB1(8,28),CB2(8,28),CB3(8,28),CS1(8,28),CS2(8,28)
      COMMON /A/ IN1,IN2,IN3,ALPHA1B,ALPHA2B,BITA1B,BITA2B
      COMMON /B/ P1,P2,P3,DA,CP1,CP2,CP3,CB1,CB2,CB3,CS1,CS2
      COMMON /C/ NBS,NBC,MNB
      CALL DTOB(A1,IN1,MNB)
      CALL DTOB(A2,IN2,MNB)
      CALL DTOB(A3,IN3,MNB)
      CALL DTOB(ALPHA1,ALPHA1B,NBC)
      CALL DTOB(ALPHA2,ALPHA2B,NBC)
      CALL DTOB(BITA1,BITA1B,NBC)
      CALL DTOB(BITA2,BITA2B,NBC)
      DO 5 K1=1,MNB+1
      DO 5 K2=1,NBC+1
      P1(K2,K1)=0
      P2(K2,K1)=0
      P3(K2,K1)=0
5     CONTINUE
      DO 7 K1=1,MNB
      DO 7 K2=1,NBC
      DA(K2,K1)=0
      CP1(K2,K1)=0
      CP2(K2,K1)=0
      CP3(K2,K1)=0
      CB1(K2,K1)=0
      CB2(K2,K1)=0
      CB3(K2,K1)=0
      CS1(K2,K1)=0
      CS2(K2,K1)=0
7     CONTINUE
      CP1(NBC,1)=BITA1B(NBC)
      CP2(NBC,1)=ALPHA1B(NBC)
      CP3(NBC,1)=ALPHA1B(NBC)
      CB1(NBC,1)=ALPHA2B(NBC)
      CB2(NBC,1)=BITA2B(NBC)
      CB3(NBC,1)=ALPHA2B(NBC)
      DO 10 K=2,NBC+1
      P1(K,1)=IN3(K-1)
      P2(K,1)=IN3(K-1)
      P3(K,1)=IN3(K-1)
10    CONTINUE
      DO 20 K=2,(MNB-NBC+1)
      P1((NBC+1),K)=IN3(NBC+K-1)
      P2((NBC+1),K)=IN3(NBC+K-1)
      P3((NBC+1),K)=IN3(NBC+K-1)
20    CONTINUE
      DO 30 K=1,MNB
      DA(NBC,K)=1
30    CONTINUE
      DO 40 K=1,NBC
      CS1(K,1)=1
      CS2(K,1)=1
40    CONTINUE
      DO 50 K=1,MNB
      IN3(K)=INV(IN3(K))
50    CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE DTOB(A,ARRAY,NB)
      INTEGER A,ARRAY(27),NB,R
      IF (A.EQ.0) GOTO 40
      IF (A.LT.0) A=(2**NB)+A
      K=1
10    IF (A.EQ.1) GOTO 20
      R=INT(A/2)
      ARRAY(K)=1
      IF ((R*2).EQ.A) ARRAY(K)=0
      A=R
      K=K+1
      GOTO 10
20    ARRAY(K)=1
      DO 30 J=K+1,NB
      ARRAY(J)=0
30    CONTINUE
      GOTO 60
40    DO 50 K=1,NB
      ARRAY(K)=0
50    CONTINUE
60    RETURN
      END
C
C
C
      SUBROUTINE BLOCK(OUT1,OUT2,OUT3)
      INTEGER IN1(28),IN2(28),IN3(28),ALPHA1B(8),ALPHA2B(8)
     &,BITA1B(8),BITA2B(8),P1(9,28),P2(9,28),P3(9,28),DA(8,27)
     &,CP1(8,28),CP2(8,28),CP3(8,28),CB1(8,28),CB2(8,28),CB3(8,28)
     &,CS1(8,28),CS2(8,28),NBS,NBC,MNB,OUT1,OUT2,OUT3,OP1,OP2,OP3
     &,NP1,NP2,NP3,OCP1,OCP2,OCP3,NCP1,NCP2,NCP3,OCB1,OCB2,OCB3
     &,NCB1,NCB2,NCB3,OCS1,OCS2,NCS1,NCS2,ALPHA1,ALPHA2,BITA1,BITA2
     &,A1,A2,A3,B1(28),B2(28),B3(28),D
      COMMON /A/ IN1,IN2,IN3,ALPHA1B,ALPHA2B,BITA1B,BITA2B
      COMMON /B/ P1,P2,P3,DA,CP1,CP2,CP3,CB1,CB2,CB3,CS1,CS2
      COMMON /C/ NBS,NBC,MNB
      COMMON /D/ OP1,OP2,OP3,OCP1,OCP2,OCP3,OCB1,OCB2,OCB3,OCS1,OCS2
     &,ALPHA1,ALPHA2,BITA1,BITA2,A1,A2,A3,D
      NCELL=1
      COUNT=1
      J=1
      DO 30 K=1,MNB
      DO 10 I=NCELL,1,-1
      ALPHA1=ALPHA1B(I)
      ALPHA2=ALPHA2B(I)
      BITA1=BITA1B(I)
      BITA2=BITA2B(I)
      A1=IN1(J)
      A2=IN2(J)
      A3=IN3(J)
      D=DA(I,J)
      OP1=P1(I+1,J)
      OP2=P2(I+1,J)
      OP3=P3(I+1,J)
      OCP1=CP1(I,J)
      OCP2=CP2(I,J)
      OCP3=CP3(I,J)
      OCB1=CB1(I,J)
      OCB2=CB2(I,J)
      OCB3=CB3(I,J)
      OCS1=CS1(I,J)
      OCS2=CS2(I,J)
      CALL CELL(NCS1,NCS2,NCP1,NCP2,NCP3,NCB1,NCB2,NCB3,NP1,NP2,NP3)
      P1(I,J+1)=NP1
      P2(I,J+1)=NP2
      P3(I,J+1)=NP3
      CP1(I,J+1)=NCP1
      CP2(I,J+1)=NCP2
      CP3(I,J+1)=NCP3
      CB1(I,J+1)=NCB1
      CB2(I,J+1)=NCB2
      CB3(I,J+1)=NCB3
      CS1(I,J+1)=NCS1
      CS2(I,J+1)=NCS2
      J=J+1
   10 CONTINUE
      B1(K)=P1(1,K+1)
      B2(K)=P2(1,K+1)
      B3(K)=P3(1,K+1)
      J=1
      IF(NCELL.LT.NBC) GOTO 20
      J=J+COUNT
      COUNT=COUNT+1
      GOTO 30
   20 NCELL=NCELL+1
   30 CONTINUE
      CALL BTOD(OUT1,B1,MNB)
      CALL BTOD(OUT2,B2,MNB)
      CALL BTOD(OUT3,B3,MNB)
      RETURN
      END
C
C
C
      SUBROUTINE CELL(NCS1,NCS2,NCP1,NCP2,NCP3,NCB1,NCB2,NCB3
     &,NP1,NP2,NP3)
      INTEGER NCS1,NCS2,NCP1,NCP2,NCP3,NCB1,NCB2,NCB3,NP1,NP2,NP3
     &,OCS1,OCS2,OCP1,OCP2,OCP3,OCB1,OCB2,OCB3,OP1,OP2,OP3,ALPHA1
     &,ALPHA2,BITA1,BITA2,A1,A2,A3,NA3,D,S1,S2,D1,D2,D3
      COMMON /D/ OP1,OP2,OP3,OCP1,OCP2,OCP3,OCB1,OCB2,OCB3,OCS1,OCS2
     &,ALPHA1,ALPHA2,BITA1,BITA2,A1,A2,A3,D
      CALL SUM(A1,A3,OCS1,S1,NCS1)
      CALL SUM(A2,A3,OCS2,S2,NCS2)
      S1=XOR(S1,D)
      S2=XOR(S2,D)
      CALL PROD(OP1,BITA1,S1,OCP1,D1,NCP1)
      CALL PROD(OP2,ALPHA1,S1,OCP2,D2,NCP2)
      CALL PROD(OP3,ALPHA1,S1,OCP3,D3,NCP3)
      CALL PROD(D1,ALPHA2,S2,OCB1,NP1,NCB1)
      CALL PROD(D2,BITA2,S2,OCB2,NP2,NCB2)
      CALL PROD(D3,ALPHA2,S2,OCB3,NP3,NCB3)
      RETURN
      END
C
C
C
      FUNCTION INV(A)
      INTEGER A
      INV=1
      IF(A.EQ.1) INV=0
      RETURN
      END
C
C
C
      FUNCTION XOR(A,B)
      INTEGER A,B
      XOR=1
      IF(A.EQ.B) XOR=0
      RETURN
      END
C
C
C
      SUBROUTINE SUM(A,B,CIN,S,COUT)
      INTEGER A,B,CIN,COUT,S
      S=A+B+CIN
      IF (S.LT.2) GOTO 10
      IF (S.EQ.2) GOTO 20
      S=1
      COUT=1
      GOTO 30
   10 COUT=0
      GOTO 30
   20 S=0
      COUT=1
   30 RETURN
      END
C
C
C
      SUBROUTINE PROD(A,B,C,CIN,P,COUT)
      INTEGER A,B,C,CIN,COUT,P,BANDC
      BANDC=B*C
      CALL SUM(A,BANDC,CIN,P,COUT)
      RETURN
      END
C
C
C
      SUBROUTINE BTOD(A,ARRAY,NB)
      INTEGER A,ARRAY(27),NB,K
      A=0
      DO 10 K=1,NB
      A=A+ARRAY(K)*2**(K-1)
10    CONTINUE
      IF (ARRAY(NB).EQ.1) A=-((2**NB)-A)
      RETURN
      END
```

B1.3- 3-PORT SERIAL SYSTOLIC ADAPTOR PROGRAM

```
      INTEGER IN1(28),IN2(28),IN3(28),ALPHA1B(6),NBS,NBC,MNB,OUT1
     &,X,U,OUT2,NSECT,NSPT,XK,UKP1,KCOEFF,ALPHA1,ALPHA2
     &,ALPHA2B(8),OUT3,ALPHA3B(8),BITA2B(8),ALPHA3,BITA2,A1,A2,A3
     &,IPSET(100),OPSET(100),COEFSET(20),USET(22)
     &,B1,B2,B3
      CHARACTER*1 CH
      COMMON /A/ IN1,IN2,IN3,ALPHA1B,ALPHA2B,ALPHA3B,BITA2B
      COMMON /C/ NBS,NBC,MNB
5     CALL READIP(A1,A2,A3,ALPHA1,ALPHA2,ALPHA3,BITA2)
      PRINT,
      PRINT,'ADAPTOR OUTPUTS USING DIFFERENCE EQUATIONS :-'
      PRINT,
      IAO=A1+A2+A3
      PRINT,'B1 =',A1-ALPHA1*IAO
      PRINT,'B2 =',A2-ALPHA2*IAO
      PRINT,'B3 =',A3-ALPHA3*IAO
      PRINT,
      CALL INIT(A1,A2,A3,ALPHA1,ALPHA2,ALPHA3,BITA2)
      CALL BLOCK(OUT1,OUT2,OUT3)
      PRINT,
      PRINT,'SYSTOLIC ADAPTOR OUTPUTS :-'
      PRINT,
      PRINT,'B1 =',OUT1
      PRINT,'B2 =',OUT2
      PRINT,'B3 =',OUT3
      PRINT,
      PRINT,'DO YOU WISH TO STOP?(Y/N)'
      READ,CH
      IF(CH.EQ.'N') GOTO 5
      STOP
      END
C
C
C
      SUBROUTINE READIP(A1,A2,A3,ALPHA1,ALPHA2,ALPHA3,BITA2)
      INTEGER A1,A2,A3,ALPHA1,ALPHA2,ALPHA3,BITA2,NBC,NBS,MNB
      COMMON /C/ NBS,NBC,MNB
      PRINT,
      PRINT,'A1   A2   A3'
      READ,A1,A2,A3
      PRINT,
      PRINT,'ALPHA1 ALPHA2 ALPHA3'
      READ,ALPHA1,ALPHA2,ALPHA3
      PRINT,
      PRINT,'NBS    NBC'
      READ,NBS,NBC
      PRINT,
      MNB=NBS+NBC+3
      RETURN
      END
C
C
C
```

```
      SUBROUTINE INIT(A1,A2,A3,ALPHA1,ALPHA2,ALPHA3,BITA2)
      INTEGER A1,A2,A3,IN1(28),IN2(28),IN3(28),ALPHA1,ALPHA2,ALPHA3
     &,BITA2,ALPHA1B(8),ALPHA2B(8),ALPHA3B(8),BITA2B(8),P1(9,28)
     &,P2(9,28),P3(9,28),DA(8,27),CP1(8,28),CP2(8,28),CP3(8,28)
     &,CONTA(8,28),CB1(8,28),CB2(8,28),CB3(8,28),CS1(8,28),CS2(8,28)
      COMMON /A/ IN1,IN2,IN3,ALPHA1B,ALPHA2B,ALPHA3B,BITA2B
      COMMON /B/ P1,P2,P3,DA,CONTA,CP1,CP2,CP3,CB1,CB2,CB3,CS1,CS2
      COMMON /C/ NBS,NBC,MNB
      CALL DTOB(A1,IN1,MNB)
      CALL DTOB(A2,IN2,MNB)
      CALL DTOB(A3,IN3,MNB)
      CALL DTOB(ALPHA1,ALPHA1B,NBC)
      CALL DTOB(ALPHA2,ALPHA2B,NBC)
      CALL DTOB(ALPHA3,ALPHA3B,NBC)
      DO 5 K1=1,MNB+1
      DO 5 K2=1,NBC+1
      P1(K2,K1)=0
      P2(K2,K1)=0
      P3(K2,K1)=0
5     CONTINUE
      DO 7 K1=1,MNB
      DO 7 K2=1,NBC
      DA(K2,K1)=0
      CONTA(K2,K1)=0
      CP1(K2,K1)=0
      CP2(K2,K1)=0
      CP3(K2,K1)=0
      CB1(K2,K1)=0
      CB2(K2,K1)=0
      CB3(K2,K1)=0
      CS1(K2,K1)=0
      CS2(K2,K1)=0
7     CONTINUE
      CP1(NBC,1)=ALPHA1B(NBC)
      CP2(NBC,1)=ALPHA2B(NBC)
      CP3(NBC,1)=ALPHA3B(NBC)
      DO 10 K=2,NBC+1
      P1(K,1)=INV(IN1(K-1))
      P2(K,1)=INV(IN2(K-1))
      P3(K,1)=INV(IN3(K-1))
10    CONTINUE
      DO 20 K=2,(MNB-NBC+1)
      P1((NBC+1),K)=INV(IN1(NBC+K-1))
      P2((NBC+1),K)=INV(IN2(NBC+K-1))
      P3((NBC+1),K)=INY(IN3(NBC+K-1))
20    CONTINUE
      DO 30 K=1,MNB
      DA(NBC,K)=1
      CONTA(1,K)=1
30    CONTINUE
      RETURN
      END
C
C
C
```

```
      SUBROUTINE DTOB(A,ARRAY,NB)
      INTEGER A,ARRAY(27),NB,R
      IF (A.EQ.0) GOTO 40
      IF (A.LT.0) A=(2**NB)+A
      K=1
10    IF (A.EQ.1) GOTO 20
      R=INT(A/2)
      ARRAY(K)=1
      IF ((R*2).EQ.A) ARRAY(K)=0
      A=R
      K=K+1
      GOTO 10
20    ARRAY(K)=1
      DO 30 J=K+1,NB
      ARRAY(J)=0
30    CONTINUE
      GOTO 60
40    DO 50 K=1,NB
      ARRAY(K)=0
50    CONTINUE
60    RETURN
      END
C
C
C
      SUBROUTINE BLOCK(OUT1,OUT2,OUT3)
      INTEGER IN1(28),IN2(28),IN3(28),ALPHA1B(8),ALPHA2B(8)
     &,ALPHA3B(8),BITA2B(8),P1(9,28),P2(9,28),P3(9,28),DA(8,27)
     &,CP1(8,28),CP2(8,28),CP3(8,28),CB1(8,28),CB2(8,28),CB3(8,28)
     &,CS1(8,28),CS2(8,28),NBS,NBC,MNB,OUT1,OUT2,OUT3,OP1,OP2,OP3
     &,NP1,NP2,NP3,OCP1,OCP2,OCP3,NCP1,NCP2,NCP3,OCB1,OCB2,OCB3
     &,NCB1,NCB2,NCB3,OCS1,OCS2,NCS1,NCS2,ALPHA1,ALPHA2,ALPHA3,BITA2
     &,A1,A2,A3,B1(28),B2(28),B3(28),D,CONT,CONTA(8,28)
      COMMON /A/ IN1,IN2,IN3,ALPHA1B,ALPHA2B,ALPHA3B,BITA2B
      COMMON /B/ P1,P2,P3,DA,CONTA,CP1,CP2,CP3,CB1,CB2,CB3,CS1,CS2
      COMMON /C/ NBS,NBC,MNB
      COMMON /D/ OP1,OP2,OP3,OCP1,OCP2,OCP3,OCS1,OCS2
     &,ALPHA1,ALPHA2,ALPHA3,A1,A2,A3,D,CONT
      NCELL=1
      COUNT=1
      J=1
      DO 30 K=1,MNB
      DO 10 I=NCELL,1,-1
      ALPHA1=ALPHA1B(I)
      ALPHA2=ALPHA2B(I)
      ALPHA3=ALPHA3B(I)
      A1=IN1(J)
      A2=IN2(J)
      A3=IN3(J)
      D=DA(I,J)
      CONT=CONTA(I,J)
      OP1=P1(I+1,J)
      OP2=P2(I+1,J)
      OP3=P3(I+1,J)
      OCP1=CP1(I,J)
      OCP2=CP2(I,J)
      OCP3=CP3(I,J)
      OCS1=CS1(I,J)
      OCS2=CS2(I,J)
      CALL CELL(NCS1,NCS2,NCP1,NCP2,NCP3,NP1,NP2,NP3)
      P1(I,J+1)=NP1
      P2(I,J+1)=NP2
      P3(I,J+1)=NP3
      CP1(I,J+1)=NCP1
      CP2(I,J+1)=NCP2
      CP3(I,J+1)=NCP3
      CS1(I,J+1)=NCS1
      CS2(I,J+1)=NCS2
      J=J+1
 10   CONTINUE
      B1(K)=P1(1,K+1)
      B2(K)=P2(1,K+1)
      B3(K)=P3(1,K+1)
      J=1
      IF(NCELL.LT.NBC) GOTO 20
      J=J+COUNT
      COUNT=COUNT+1
      GOTO 30
 20   NCELL=NCELL+1
 30   CONTINUE
      CALL BTOD(OUT1,B1,MNB)
      CALL BTOD(OUT2,B2,MNB)
      CALL BTOD(OUT3,B3,MNB)
      RETURN
      END
C
C
C
      SUBROUTINE CELL(NCS1,NCS2,NCP1,NCP2,NCP3
     &,NP1,NP2,NP3)
      INTEGER NCS1,NCS2,NCP1,NCP2,NCP3,NCB1,NCB2,NCB3,NP1,NP2,NP3
     &,OCS1,OCS2,OCP1,OCP2,OCP3,OCB1,OCB2,OCB3,OP1,OP2,OP3,ALPHA1
     &,ALPHA2,ALPHA3,BITA2,A1,A2,A3,NA3,D,S1,S2,D1,D2,D3,CONT
      COMMON /D/ OP1,OP2,OP3,OCP1,OCP2,OCP3,OCS1,OCS2
     &,ALPHA1,ALPHA2,ALPHA3,A1,A2,A3,D,CONT
      CALL SUM(A1,A2,OCS1,S1,NCS1)
      CALL SUM(A3,S1,OCS2,S2,NCS2)
      S2=XOR(S2,D)
      CALL PROD(OP1,S2,ALPHA1,OCP1,NP1,NCP1)
      CALL PROD(OP2,S2,ALPHA2,OCP2,NP2,NCP2)
      CALL PROD(OP3,S2,ALPHA3,OCP3,NP3,NCP3)
      NP1=XOR(NP1,CONT)
      NP2=XOR(NP2,CONT)
      NP3=XOR(NP3,CONT)
      RETURN
      END
C
C
C
      FUNCTION INV(A)
      INTEGER A
      INV=1
      IF(A.EQ.1) INV=0
      RETURN
      END
C
C
C
      FUNCTION XOR(A,B)
      INTEGER A,B
      XOR=1
      IF(A.EQ.B) XOR=0
      RETURN
      END
C
C
C
      SUBROUTINE SUM(A,B,CIN,S,COUT)
      INTEGER A,B,CIN,COUT,S
      S=A+B+CIN
      IF (S.LT.2) GOTO 10
      IF (S.EQ.2) GOTO 20
      S=1
      COUT=1
      GOTO 30
 10   COUT=0
      GOTO 30
 20   S=0
      COUT=1
 30   RETURN
      END
C
C
C
      SUBROUTINE PROD(A,B,C,CIN,P,COUT)
      INTEGER A,B,C,CIN,COUT,P,BANDC
      BANDC=B*C
      CALL SUM(A,BANDC,CIN,P,COUT)
      RETURN
      END
C
C
C
      SUBROUTINE BTOD(A,ARRAY,NB)
      INTEGER A,ARRAY(27),NB,K
      A=0
      DO 10 K=1,NB
      A=A+ARRAY(K)*2**(K-1)
 10   CONTINUE
      IF (ARRAY(NB).EQ.1) A=-((2**NB)-A)
      RETURN
      END
```

B1.4- UNIVERSAL SYSTOLIC ADAPTOR PROGRAM

```
      INTEGER ADAPTOR
      COMMON /E/ ADAPTOR
 5    PRINT,
      PRINT,
      PRINT,"         UNIVERSAL SYSTOLIC WDF ADAPTOR"
      PRINT,
      PRINT,"                     MENU"
      PRINT,
      PRINT,"           1) 2-PORT ADAPTOR."
      PRINT,"           2) 3-PORT PARALLEL ADAPTOR."
      PRINT,"           3) 3-PORT SERIAL ADAPTOR."
      PRINT,"           4) END PROGRAM."
      PRINT,
      PRINT,"ENTER YOUR CHOICE NUMBER :-"
      READ,ADAPTOR
      PRINT,
      GOTO (10,20,30,40),ADAPTOR
      GOTO 5
10    CALL TWOPA
      GOTO 5
20    CALL THRPPA
      GOTO 5
30    CALL THRPSA
      GOTO 5
40    STOP
      END
C
C
C
      SUBROUTINE TWOPA
      INTEGER A1,A2,A3,ALPHA1,D,NBS,NBC,MNB,OUT1,OUT2,OUT3
      COMMON /C/ NBS,NBC,MNB
      PRINT,
      PRINT,"ALPHA   A1   A2"
      READ,ALPHA1,A1,A2
      PRINT,
      PRINT,"NBS    NBC"
      READ,NBS,NBC
      MNB=NBS+NBC+2
      D=ALPHA1*(A2-A1)
      PRINT,
      PRINT,"THE ADAPTOR OUTPUTS ARE :-"
      PRINT,
      PRINT,"B1 = ",A1+D
      PRINT,"B2 = ",A2+D
      PRINT,
      CALL BLOCK(A1,A2,0,ALPHA1,0,0,0,0,OUT1,OUT2,OUT3)
      PRINT,"THE SYSTOLIC ADAPTOR OUTPUTS ARE :-"
      PRINT,
      PRINT,"B1 = ",OUT1
      PRINT,"B2 = ",OUT2
      PRINT,
      PRINT,"TYPE <CR> TO CONTINUE !!!"
      READ,D
      PRINT,
      RETURN
      END
C
C
C
      SUBROUTINE THRPPA
      INTEGER A1,A2,A3,ALPHA1,ALPHA2,BITA1,BITA2,A0,MNB,NBS,NBC
     &,OUT1,OUT2,OUT3
      COMMON /C/ NBS,NBC,MNB
      PRINT,
      PRINT,"ALPHA1    ALPHA2    A1    A2    A3"
      READ,ALPHA1,ALPHA2,A1,A2,A3
      PRINT,
      PRINT,"NBS    NBC"
      READ,NBS,NBC
      PRINT,
      MNB=NBS+NBC+3
      BITA1=ALPHA1-1
      BITA2=ALPHA2-1
      A0=A1*ALPHA1+A2*ALPHA2+A3*(2-ALPHA1-ALPHA2)
      PRINT,"THE ADAPTOR OUTPUTS ARE :-"
      PRINT,
      PRINT,"B1 = ",A0-A1
      PRINT,"B2 = ",A0-A2
      PRINT,"B3 = ",A0-A3
      PRINT,
      CALL BLOCK(A1,A2,A3,ALPHA1,ALPHA2,0,BITA1,BITA2,OUT1,OUT2,OUT3)
      PRINT,
      PRINT,"THE SYSTOLIC ADAPTOR OUTPUTS ARE :-"
      PRINT,
      PRINT,"B1 = ",OUT1
      PRINT,"B2 = ",OUT2
      PRINT,"B3 = ",OUT3
      PRINT,
      PRINT,"TYPE <CR> TO CONTINUE !!!"
      READ,A0
      PRINT,
      RETURN
      END
C
C
C
      SUBROUTINE THRPSA
      INTEGER ALPHA1,ALPHA2,ALPHA3,A1,A2,A3,A0,NBS,NBC,MNB
     &,OUT1,OUT2,OUT3
      COMMON /C/ NBS,NBC,MNB
      PRINT,
      PRINT,"ALPHA1  ALPHA2  ALPHA3  A1   A2   A3"
      READ,ALPHA1,ALPHA2,ALPHA3,A1,A2,A3
      PRINT,
      PRINT,"NBS    NBC"
      READ,NBS,NBC
      PRINT,
      MNB=NBS+NBC+3
      A0=A1+A2+A3
      PRINT,"THE ADAPTOR OUTPUTS ARE :-"
      PRINT,
      PRINT,"B1 = ",A1-ALPHA1*A0
      PRINT,"B2 = ",A2-ALPHA2*A0
      PRINT,"B3 = ",A3-ALPHA3*A0
      PRINT,
      CALL BLOCK(A1,A2,A3,ALPHA1,ALPHA2,ALPHA3,0,0,OUT1,OUT2,OUT3)
      PRINT,
      PRINT,"THE SYSTOLIC ADAPTOR OUTPUTS ARE :-"
      PRINT,
      PRINT,"B1 = ",OUT1
      PRINT,"B2 = ",OUT2
      PRINT,"B3 = ",OUT3
      PRINT,
      PRINT,"TYPE <CR> TO CONTINUE !!!"
      READ,A0
      PRINT,
      RETURN
      END
C
C
C
      SUBROUTINE INIT(A1,A2,A3,ALPHA1,ALPHA2,ALPHA3,BITA1,BITA2)
      INTEGER A1,A2,A3,X1B(28),X2B(28),X3B(28),X4B(28),ALPHA1,ALPHA2
     &,BITA2,ALPHA3,Z3B(8),W3B(8),Z1B(8),Z2B(8),W1B(8),W2B(8),P1(9,28)
     &,P2(9,28),P3(9,28),D1A(8,27),D2A(8,27),CP1(8,28),CP2(8,28)
     &,CP3(8,28),CB1(8,28),CB2(8,28),CB3(8,28),CS1(8,28),CS2(8,28)
     &,ADAPTOR,Z1,Z2,Z3,W1,W2,W3,DP1,DP2,DP3,DCS1,DCS2,S
     &,DP1B(28),DP2B(28),DP3B(28),BITA1,X1,X2,X3,X4
      COMMON /A/ X1B,X2B,X3B,X4B,Z1B,Z2B,Z3B,W1B,W2B,W3B
      COMMON /B/ P1,P2,P3,D1A,D2A,CP1,CP2,CP3,CB1,CB2,CB3,CS1,CS2
      COMMON /C/ NBS,NBC,MNB
      COMMON /E/ ADAPTOR
      IF (ADAPTOR.EQ.1) GOTO 10
      IF (ADAPTOR.EQ.2) GOTO 20
      Z1=ALPHA1
      Z2=ALPHA2
      Z3=ALPHA3
      W1=Z1
      W2=Z2
      W3=Z3
      X1=A1
      X2=A2
      X3=A3
      X4=0
      DP1=(2**MNB)-1-A1
      DP2=(2**MNB)-1-A2
      DP3=(2**MNB)-1-A3
      DCS1=0
      DCS2=0
      S=1
      GOTO 30
10    Z1=ALPHA1
      Z2=ALPHA1
      Z3=0
      W1=0
      W2=0
      W3=0
      X1=A2
      X2=(2**MNB)-1-A1
      DP1=A1
      DP2=A2
      DCS1=1
      S=0
      GOTO 30
20    Z1=BITA1
      Z2=ALPHA1
      Z3=ALPHA1
      W1=ALPHA2
      W2=BITA2
      W3=ALPHA2
      X1=A1
      X2=(2**MNB)-1-A3
      X3=A2
      X4=X2
      DP1=A3
      DP2=A3
      DP3=A3
      DCS1=1
      DCS2=1
      S=0
30    CALL DTOB(X1,X1B,MNB)
      CALL DTOB(X2,X2B,MNB)
      CALL DTOB(X3,X3B,MNB)
      CALL DTOB(X4,X4B,MNB)
      CALL DTOB(Z1,Z1B,NBC)
      CALL DTOB(Z2,Z2B,NBC)
      CALL DTOB(Z3,Z3B,NBC)
      CALL DTOB(W1,W1B,NBC)
      CALL DTOB(W2,W2B,NBC)
      CALL DTOB(W3,W3B,NBC)
      CALL DTOB(DP1,DP1B,MNB)
      CALL DTOB(DP2,DP2B,MNB)
      CALL DTOB(DP3,DP3B,MNB)
      DO 40 K1=1,MNB+1
      DO 40 K2=1,NBC+1
      P1(K2,K1)=0
      P2(K2,K1)=0
      P3(K2,K1)=0
40    CONTINUE
      DO 50 K1=1,MNB
      DO 50 K2=1,NBC
      D1A(K2,K1)=0
      D2A(K2,K1)=0
      CP1(K2,K1)=0
      CP2(K2,K1)=0
      CP3(K2,K1)=0
      CB1(K2,K1)=0
      CB2(K2,K1)=0
      CB3(K2,K1)=0
      CS1(K2,K1)=0
      CS2(K2,K1)=0
   50 CONTINUE
      CP1(NBC,1)=Z1B(NBC)
      CP2(NBC,1)=Z2B(NBC)
      CP3(NBC,1)=Z3B(NBC)
      CB1(NBC,1)=W1B(NBC)
      CB2(NBC,1)=W2B(NBC)
      CB3(NBC,1)=W3B(NBC)
      DO 60 K=2,NBC+1
      P1(K,1)=DP1B(K-1)
      P2(K,1)=DP2B(K-1)
      P3(K,1)=DP3B(K-1)
   60 CONTINUE
      DO 70 K=2,(MNB-NBC+1)
      P1((NBC+1),K)=DP1B(NBC+K-1)
      P2((NBC+1),K)=DP2B(NBC+K-1)
      P3((NBC+1),K)=DP3B(NBC+K-1)
   70 CONTINUE
      DO 80 K=1,MNB
      D1A(NBC,K)=1
      D2A(1,K)=S
   80 CONTINUE
      DO 90 K=1,NBC
      CS1(K,1)=DCS1
      CS2(K,1)=DCS2
   90 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE DTOB(A,ARRAY,NB)
      INTEGER A,ARRAY(28),NB,R
      IF (A.EQ.0) GOTO 40
      IF (A.LT.0) A=(2**NB)+A
      K=1
   10 IF (A.EQ.1) GOTO 20
      R=INT(A/2)
      ARRAY(K)=1
      IF ((R*2).EQ.A) ARRAY(K)=0
      A=R
      K=K+1
      GOTO 10
   20 ARRAY(K)=1
      DO 30 J=K+1,NB
      ARRAY(J)=0
   30 CONTINUE
      GOTO 60
   40 DO 50 K=1,NB
      ARRAY(K)=0
   50 CONTINUE
   60 RETURN
```

```
      END
C
C
C
      SUBROUTINE BLOCK(A1,A2,A3,ALPHA1,ALPHA2,ALPHA3,BITA1,BITA2
     &,OUT1,OUT2,OUT3)
      INTEGER X1B(28),X2B(28),X3B(28),X4B(28),Z1B(8),Z2B(8),Z3B(8)
     &,W1B(8),W2B(8),W3B(8),P1(9,28),P2(9,28),P3(9,28),D1A(8,27)
     &,CP1(8,28),CP2(8,28),CP3(8,28),CB1(8,28),CB2(8,28),CB3(8,28)
     &,CS1(8,28),CS2(8,28),NBS,NBC,MNB,OUT1,OUT2,OUT3,OP1,OP2,OP3
     &,NP1,NP2,NP3,OCP1,OCP2,OCP3,NCP1,NCP2,NCP3,OCB1,OCB2,OCB3
     &,NCB1,NCB2,NCB3,OCS1,OCS2,NCS1,NCS2,Z1,Z2,Z3,W1,W2,W3
     &,X1,X2,X3,X4,B1(28),B2(28),B3(28),D1,D2,D2A(8,27)
     &,A1,A2,A3,ALPHA1,ALPHA2,ALPHA3,BITA1,BITA2
      COMMON /A/ X1B,X2B,X3B,X4B,Z1B,Z2B,Z3B,W1B,W2B,W3B
      COMMON /B/ P1,P2,P3,D1A,D2A,CP1,CP2,CP3,CB1,CB2,CB3,CS1,CS2
      COMMON /C/ NBS,NBC,MNB
      COMMON /D/ OP1,OP2,OP3,OCP1,OCP2,OCP3,OCB1,OCB2,OCB3,OCS1,OCS2
     &,Z1,Z2,Z3,W1,W2,W3,X1,X2,X3,X4,D1,D2
      CALL INIT(A1,A2,A3,ALPHA1,ALPHA2,ALPHA3,BITA1,BITA2)
      NCELL=1
      COUNT=1
      J=1
      DO 30 K=1,MNB
      DO 10 I=NCELL,1,-1
      Z1=Z1B(I)
      Z2=Z2B(I)
      Z3=Z3B(I)
      W1=W1B(I)
      W2=W2B(I)
      W3=W3B(I)
      X1=X1B(J)
      X2=X2B(J)
      X3=X3B(J)
      X4=X4B(J)
      D1=D1A(I,J)
      D2=D2A(I,J)
      OP1=P1(I+1,J)
      OP2=P2(I+1,J)
      OP3=P3(I+1,J)
      OCP1=CP1(I,J)
      OCP2=CP2(I,J)
      OCP3=CP3(I,J)
      OCB1=CB1(I,J)
      OCB2=CB2(I,J)
      OCB3=CB3(I,J)
      OCS1=CS1(I,J)
      OCS2=CS2(I,J)
      CALL CELL(NCS1,NCS2,NCP1,NCP2,NCP3,NCB1,NCB2,NCB3,NP1,NP2,NP3)
      P1(I,J+1)=NP1
      P2(I,J+1)=NP2
      P3(I,J+1)=NP3
      CP1(I,J+1)=NCP1
      CP2(I,J+1)=NCP2
      CP3(I,J+1)=NCP3
      CB1(I,J+1)=NCB1
      CB2(I,J+1)=NCB2
      CB3(I,J+1)=NCB3
      CS1(I,J+1)=NCS1
      CS2(I,J+1)=NCS2
      J=J+1
 10   CONTINUE
      B1(K)=P1(1,K+1)
      B2(K)=P2(1,K+1)
      B3(K)=P3(1,K+1)
      J=1
      IF(NCELL.LT.NBC) GOTO 20
      J=J+COUNT
      COUNT=COUNT+1
      GOTO 30
 20   NCELL=NCELL+1
 30   CONTINUE
      CALL BTOD(OUT1,B1,MNB)
      CALL BTOD(OUT2,B2,MNB)
      CALL BTOD(OUT3,B3,MNB)
      RETURN
      END
C
C
C
      SUBROUTINE CELL(NCS1,NCS2,NCP1,NCP2,NCP3,NCB1,NCB2,NCB3
     &,NP1,NP2,NP3)
      INTEGER NCS1,NCS2,NCP1,NCP2,NCP3,NCB1,NCB2,NCB3,NP1,NP2,NP3
     &,OCS1,OCS2,OCP1,OCP2,OCP3,OCB1,OCB2,OCB3,OP1,OP2,OP3,Z1
     &,Z2,Z3,W1,W2,W3,X1,X2,X3,X4,D,S1,S2,D1,D2,DP1,DP2,DP3
      COMMON /D/ OP1,OP2,OP3,OCP1,OCP2,OCP3,OCB1,OCB2,OCB3,OCS1,OCS2
     &,Z1,Z2,Z3,W1,W2,W3,X1,X2,X3,X4,D1,D2
      CALL SUM(X1,X2,OCS1,S1,NCS1)
      CALL SUM(X3,X4,OCS2,S2,NCS2)
      S1=XOR(S1,D1)
      S2=XOR(S2,D1)
      CALL PROD(OP1,Z1,S1,OCP1,DP1,NCP1)
      CALL PROD(OP2,Z2,S1,OCP2,DP2,NCP2)
      CALL PROD(OP3,Z3,S1,OCP3,DP3,NCP3)
      CALL PROD(DP1,W1,S2,OCB1,NP1,NCB1)
      CALL PROD(DP2,W2,S2,OCB2,NP2,NCB2)
      CALL PROD(DP3,W3,S2,OCB3,NP3,NCB3)
      NP1=XOR(NP1,D2)
      NP2=XOR(NP2,D2)
      NP3=XOR(NP3,D2)
      RETURN
      END
C
C
C
      FUNCTION INV(A)
      INTEGER A
      INV=1
      IF(A.EQ.1) INV=0
      RETURN
```

```
      END
C
C
C
      FUNCTION XOR(A,B)
      INTEGER A,B
      XOR=1
      IF(A.EQ.B) XOR=0
      RETURN
      END
C
C
C
      SUBROUTINE SUM(A,B,CIN,S,COUT)
      INTEGER A,B,CIN,COUT,S
      S=A+B+CIN
      IF (S.LT.2) GOTO 10
      IF (S.EQ.2) GOTO 20
      S=1
      COUT=1
      GOTO 30
 10   COUT=0
      GOTO 30
 20   S=0
      COUT=1
 30   RETURN
      END
C
C
C
      SUBROUTINE PROD(A,B,C,CIN,P,COUT)
      INTEGER A,B,C,CIN,COUT,P,BANDC
      BANDC=B*C
      CALL SUM(A,BANDC,CIN,P,COUT)
      RETURN
      END
C
C
C
      SUBROUTINE BTOD(A,ARRAY,NB)
      INTEGER A,ARRAY(27),NB,K
      A=0
      DO 10 K=1,NB
      A=A+ARRAY(K)*2**(K-1)
 10   CONTINUE
      IF (ARRAY(NB).EQ.1) A=-((2**NB)-A)
      RETURN
      END
```

B2.0- WDF Design Programs

This Appendix presents the programs used for the design of the three WDFs considered in this thesis. Each program includes the synthesis routines plus the optimization routines for the finite wordlength design of the filters. The programs need a number of files to be created before running which will be used to store the WDF coefficients. These files are defined using the 'OPEN' command in Fortran at the begining of the program. The program listings are as follows,

B2.1- UEFD.SO, Unit Element Filter Design.Synthesis Optimization.

B2.2- LTFD.SO.

B2.3- LCFD.SO.

It was intended to merge all these three programs together to form a complete software tool for the design of WDFs but due to lack of computer facilities this was not achieved.

B2.1- UEFD.SO PROGRAM

```
C--------------------------------------------------------
C
C                  WDF DESIGN PROGRAM
C     BASED ON SYNTHESIS AND OPTIMIZATION OF THE
C     UNIT ELEMENT FILTERS.(FORTRAN 77)
C
C
C AUTHOR : A. R. MIRZAI           DATE : MARCH 1985
C
C             THE CENTRE FOR INFORMATION ENG
C                 THE CITY UNIVERSITY
C             NORTHAMPTON SQ. LONDON EC1V0HB
C
C
C---------------------------------------------------------
      REAL F1,RP,F2,ML,ALPHA(21)
      INTEGER N,CHOICE,I,IPCC,FPCC
      CHARACTER*1 CH
C
C THE FINITE AND INFINIT PRECESION COEFFICENTS CHANNEL
C
      OPEN(UNIT=1,FILE='UECOEF.I',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='READ/WRITE')
      OPEN(UNIT=2,FILE='UECOEF.F',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='READ/WRITE')
      OPEN(UNIT=3,FILE='UECOEF.T',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='READ/WRITE')
C
C MAIN PROGRAM
C
 5    PRINT,
      PRINT,
      PRINT,
      PRINT,'          PROGRAM TO DESIGN WDFS BASED ON'
      PRINT,'           CASCADED UNIT ELEMENT FILTERS'
      PRINT,
      PRINT,
      PRINT,'                       MAIN MENU'
      PRINT,
      PRINT,
      PRINT,
      PRINT,
      PRINT,'   1- ENTER SPECIFICATIONS.'
      PRINT,'   2- DESIGN USING SYSNTHESIS (INFINIT PRECISION).'
      PRINT,'   3- DESIGN USING OPTIMIZATION (FINITE PRECISION).'
      PRINT,'   4- SAVE COEFFICIENTS.'
      PRINT,'   5- READ INITIAL COEFFICIENTS.'
      PRINT,'   6- END PROGRAM.'
      PRINT,
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ*,CHOICE
      GOTO (10,20,30,40,50,60),CHOICE
 10   PRINT,
      PRINT,
      PRINT,'PASS BAND  PASS BAND  STOP BAND  MIN STOPBAND'
      PRINT,'EDGE FREQ    RIPPLE   EDGE FREQ  ATTNUATION'
      PRINT,'   (HZ)       (DB)       (HZ)         (DB)'
      READ*,F1,RP,F2,ML
      PRINT,
      CALL FDEG(F1,RP,F2,ML,N)
      GOTO 5
 20   CALL SYSWDF(F1,RP,F2,ML,ALPHA,N)
      GOTO 5
 30   CALL OPTWDF(F1,RP,F2,ML,ALPHA,N)
      GOTO 5
 40   CALL SAVE(ALPHA,N)
      GOTO 5
 50   CALL RDCOEF(ALPHA,N)
      GOTO 5
 60   STOP
      END
C
C SUBROUTINE FDEG :- GIVEN A SET OF SPECIFICATIONS
C THIS SUBROUTINE CALCULATES THE MINIMUM DEGREE
C REQUIRED TO MEET THE SPECIFICATIONS.
C
      SUBROUTINE FDEG(PEF,RIPPLE,SEF,MINLOSS,DEG)
      REAL PR,FP,FS,RN,T5,T6,H2,E,B1,V1,X,ML,PI
     &PEF,SEF,RIPPLE,MINLOSS
      INTEGER N,DEG
      FP=PEF
      FS=SEF
      PR=RIPPLE
      ML=MINLOSS
      PI=3.14159263
      T5=PI*FP
      T6=PI*FS
      H2=10**(PR/10)-1
      E=1+H2
      B1=4*FP
      V=1+2*H2+2*SQRT(H2*(1+H2))
      X=SIN(T6)/SIN(T5)
      RN=LOG10(10**(ML/10)-1)-LOG10(H2)+LOG10(4)
      RN=RN/(2*LOG10(2*X))
      N=INT(RN+0.999999)
      PRINT,'VSWR   = ',V
      PRINT,'BW     = ',B1
      PRINT,'DEGREE = ',N
      PRINT,
      PRINT,'ENTER THE REQUIRED DEGREE :-'
      READ*,DEG
      PRINT,
      RETURN
      END
C-------------------------------------------------------
C
C  TRANSMISSION LINE FILTER SYNTHESIS.
C
```

```
C  AUTHOR : RALPH LEVY    DATE : 1965
C         [IEEE,MTT-13,PP 514-536]
C
C  MODIFIED TO RUN ON HONEYWELL BASIC
C  BY S.S.LAWSON          DATE : APRIL 1983
C
C  MODIFIED TO RUN ON HONEYWELL FORTRAN 77
C  BY A.R.MIRZAI          DATE : JUNE 1984
C
C---------------------------------------------------
      SUBROUTINE SYSWDF(PEF,RIPPLE,SEF,MINLOSS,ALPHA,DEG)
      REAL PI,F1,F2,F3,T,T5,T6,H,H2,H3,H4,H5
      REAL P2,P3,P4,P5,P6,P7,P8,P9,P1
      REAL E1,A1,B1,R2,V1,X2,RN,S,UV,ZV,YV
      REAL W,W1,V,ZV,YV,AV,MINLOSS,PEF,SEF,RIPPLE
      REAL A(78),B(11),C(11),D(21),E(23),F(23)
      REAL G(22),U(23),Y(78),ALPHA(21),Z(78)
      INTEGER I,J,K,L,N,DEG,M,O,P,Q,R,S1,T1,O1,IN,OUT
      PI=3.14159263
      F1=PEF
      F2=SEF
      R2=RIPPLE
      A1=MINLOSS
      N=DEG
      T5=PI*F1
      T6=PI*F2
      H2=10**(R2/10)-1
      E1=1+H2
      B1=4*F1
      V1=1+2*H2+2*SQRT(H2*(1+H2))
      X2=SIN(T6)/SIN(T5)
      F3=1.0
      M=INT(N/2+0.1)
      UV=FLOAT(N)/FLOAT(2)-FLOAT(M)
      P2=V1
      P4=SQRT(P2)
      H=(P2-1)/(2*P4)
      P3=1/H
      H2=SQRT(1+P3**2)
      H5=LOG(P3+H2)
      H3=EXP(2*H5/N)
      H4=1/H3
      H3=(H3+H4)/2
      H4=H3-H4
      P6=SIN(B1*PI/4)
      P1=1/(P6**2)
      P4=2*P1
      P3=P4-1
      P7=SQRT(P3**2-1)
      P2=LOG(P3+P7)
      P2=N*P2/2
      P2=EXP(P2)
      P5=1/P2
      P2=(P2+P5)/2
      P2=LOG(H*H*P2*P2+1)
```

```
      P5=LOG(10)
      W1=10*P2/P5
      P8=H3*P3+H4*P7
      P9=H3*P3-H4*P7
      DO 20 R=1,M
      C=COS((2*R-1)*PI/N)
      P5=P8-C
      P7=P9-C
      P5=SQRT(P5*P7)
      A(R+1)=P5/(H3+C)
      P7=H3+C
      P5=H3*C*P4+P4
      P5=P5/P7**2
      B(R+1)=SQRT(2-2*P5+2*A(R+1))
      P5=(C+1)/2
      P5=P1/P5
      C(R+1)=P5-1
20    CONTINUE
      IF (UV.LT.0.2) GOTO 30
      P5=(H3+P3)/(H3-1)
       A(M+2)=SQRT(P5)
      AV=N*H/P6
      GOTO 40
30    AV=1/H2
40    F(2)=0.0
      F(1)=1.0
      DO 80 R=1,M
      DO 70 I=1,R
      G(I+1)=C(R+1)*F(I)
      IF (I.EQ.R) GOTO 50
      G(I+1)=F(I+1)+G(I+1)
      GOTO 70
50    DO 60 J=1,R
      F(J+1)=G(J+1)
60    CONTINUE
70    CONTINUE
80    CONTINUE
      DO 90 I=1,M
      J=2*I
      D(J+1)=AV*F(I+1)
90    CONTINUE
      D(1)=AV
      G(2)=0.0
      DO 160 R=1,M
      L=2*R
      DO 150 I=1,L
      IF (I.EQ.1) GOTO 100
      G(I+1)=A(R+1)*F(I-1)
      IF (I.EQ.L) GOTO 130
100   G(I+1)=B(R+1)*F(I)+G(I+1)
      O=L-I
      IF (O.EQ.1) GOTO 150
      IF (I.EQ.1) GOTO 150
      G(I+1)=G(I+1)+F(I+1)
      GOTO 150
```

```
 130  DO 140 J=1,L
      F(J+1)=G(J+1)
 140  CONTINUE
 150  CONTINUE
 160  CONTINUE
      IF (UV.LT.0.2) GOTO 190
      L=2*M
      DO 180 I=0,L
      F(I+1)=A(M+2)*F(I+1)
      IF (I.EQ.L) GOTO 170
      F(I+1)=F(I+1)+F(I+2)
 170  G(I+1)=F(I+1)
      G(I+1)=F(I+1)
 180  CONTINUE
 190  DO 200 I=0,L,2
      F(I+1)=F(I+1)-D(I+1)
      G(I+1)=F(I+1)+2*D(I+1)
 200   CONTINUE
      DO 210 I=0,L
      O=L-I
      E(I+1)=F(O+1)
      U(I+1)=G(O+1)
 210   CONTINUE
      E(L+2)=2*UV
      U(L+2)=E(L+2)
      K=M+1
      P=INT(M+E(L+2)+0.1)
      O1=P
      DO 310 J=N,K,-1
      F(1)=E(1)
      F(2)=E(2)
      G(1)=U(1)
      G(2)=U(2)
      I=INT(J/2+0.1)
      V=FLOAT(J)/FLOAT(2)-FLOAT(I)
      Q=N-J+1
      DO 220 R=1,I
      L=2*R
      F(L+2)=E(L+2)+F(L)
      F(L+1)=E(L+1)+F(L-1)
      G(L+2)=U(L+2)+G(L)
      G(L+1)=U(L+1)+G(L-1)
 220  CONTINUE
      IF (UV.LT.0.2) GOTO 230
      ZV=F(J+1)/F(J)
      YV=G(J)/G(J+1)
      GOTO 240
 230  ZV=F(J)/F(J+1)
      YV=G(J+1)/G(J)
 240  Z(Q+1)=(ZV+YV)/2
      Y(Q+1)=ABS(ZV-YV)
      IF (Z(Q+1).LT.1.0) GOTO 260
      IF (Z(Q+1).GT.10.0) GOTO 250
      IF (Y(Q+1).GT.0.03) GOTO 320
      GOTO 270
```

```
 250  A(Q+1)=0.003*Z(Q+1)
      IF (Y(Q+1).GT.A(Q+1)) GOTO 320
      GOTO 270
 260  IF (Y(Q+1).LT.0.001) GOTO 270
      IF (Z(Q+1).LT.0.1) GOTO 320
      IF (Y(Q+1).GT.0.01) GOTO 320
 270  W=UV-V
      W=ABS(W)
      S1=Q
      ZV=Z(Q+1)
      IF (W.LT.0.2) GOTO 280
      ZV=1/ZV
 280  DO 290 R=1,I
      L=2*R
      F(L+2)=E(L+1)-E(L+2)/ZV
      F(L+1)=E(L)-E(L+1)*ZV
      G(L+2)=U(L+1)-ZV*U(L+2)
      G(L+1)=U(L)-U(L+1)/ZV
 290  CONTINUE
      L=2*I
      F(L+3)=E(L+2)
      E(L+3)=0.0
      E(L+2)=0.0
      G(L+3)=U(L+2)
      U(L+3)=0.0
      U(L+2)=0.0
      DO 300 R=L,0,-1
      E(R+1)=F(R+3)+E(R+3)
      U(R+1)=G(R+3)+U(R+3)
 300  CONTINUE
 310  CONTINUE
C
C OUTPUT SECTION
C
 320  PRINT,
      PRINT,'DEGREE, N  =    ',N
      PRINT,'VOLTAGE STANDING WAVE RATIO, VSWR  =  ',V1
      PRINT,'BANDWIDTH, BW  =  ',B1
      PRINT,'MAXIMUN ATTENUATION, L(DB)  =  ',W1
      PRINT,
      PRINT,' THE STEP IMPEDENCES ARE AS FOLLOWS :- '
      PRINT,'  '
      DO 330 I=1,P
      PRINT,I,Z(I+1)
 330  CONTINUE
      PRINT,'  '
      Z(1)=1.0
      Z(N+2)=1.0
      M=INT(N/2)
      IF (N.EQ.(2*M)) GOTO 350
      DO 340 I=1,M
      Z(N+2-I)=Z(I+1)
 340  CONTINUE
      GOTO 370
 350  DO 360 I=1,M
```

```
      Z(N+2-I)=Z(I+1)
 360  CONTINUE
 370  PRINT,'THE COEFFICIENTS OF THE WDF STRUCTURE'
      PRINT,'ARE AS FOLLOWS :-'
      DO 380 I=1,N+1
      ALPHA(I)=(Z(I)-Z(I+1))/(Z(I)+Z(I+1))
      WRITE(6,390) I,ALPHA(I)
 380  CONTINUE
      PRINT,
      PRINT,
      PRINT,'TYPE <CR> TO CONTINUE !!'
      READ*,I
      PRINT,
 390  FORMAT(' ALPHA(',I4,') = ',F15.13)
      RETURN
      END
C
C
C
      SUBROUTINE SAVE(ALPHA,N)
      REAL ALPHA(21)
      INTEGER I,J,N
      PRINT,
      PRINT,
      PRINT,'THE CURRENT VALUES OF THE COEFFICIENTS CAN BE SAVED IN,'
      PRINT,
      PRINT,'    1) UECOEF.I (INFINITE PRECISION),OR'
      PRINT,'    2) UECOEF.F ( FINITE PRECISION), OR'
      PRINT,'    3) UECOEF.T (A TEMPORARY FILE).'
      PRINT,
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ,J
      PRINT,
      REWIND(J)
      DO 10 I=1,N+1
      WRITE(J,*) ALPHA(I)
 10   CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE RDCOEF(ALPHA,N)
      REAL ALPHA(21)
      INTEGER N,J,I
      PRINT,
      PRINT,'ENTER THE FILTER ORDER :-'
      READ,N
      PRINT,
      PRINT,
      PRINT,'    1) READ COEFFICIENTS FROM UECOEF.I, OR'
      PRINT,'    2) READ COEFFICIENTS FROM UECOEF.F, OR'
      PRINT,'    3) READ COEFFICIENTS FROM UECOEF.T, OR'
      PRINT,'    4) READ COEFFICIENTS FROM TERMINAL.'
      PRINT,
```

```
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ,J
      PRINT,
      IF(J.EQ.4) GOTO 20
      REWIND(J)
      READ(J,*) (ALPHA(I),I=1,N+1)
      GOTO 30
 20   PRINT,
      PRINT,'ENTER THE COEFFICINTS :-'
      PRINT,
      READ,(ALPHA(I),I=1,N+1)
 30   RETURN
      END
C--------------------------------------------------------
C
C       FINITE-WORD LENGTH DESIGN OF WDFS
C    USING HOOKE AND JEEVES' DIRECT SEARCH METHOD.
C         [ JACM, VOL-8, PP 212-229, 1961 ]
C
C
C    AUTHOR : A. R. MIRZAI           DATE : NOV  1984
C
C--------------------------------------------------------
      SUBROUTINE OPTWDF(PEF,RIPPLE,SEF,MINLOSS,ALPHA,DEG)
      REAL ALPHA(20),MIND,SD,MFACT,DP,PR,ML,
     &DFP,DFS,FP,FS,E1,PI,MINLOSS,PEF,SEF,RIPPLE
      INTEGER NBIT,NS,NSBIT,MNFE,NFE,N,I,DEG
     &,NP,NCCOEF
      CHARACTER*1 CH
      LOGICAL EOP
      COMMON/A/ FS,DFP,DFS,NP,NS,DP,PR,ML
      COMMON/B/ NFE
      COMMON/D/ NCCOEF
      ML=MINLOSS
      DP=RIPPLE/2
      PR=RIPPLE
      FP=PEF
      FS=SEF
      N=DEG+1
      PRINT,
      PRINT,
      PRINT,'              FINITE-WORD LENGTH DESIGN PROGRAM'
      PRINT,
      PRINT,
      PRINT,'DO YOU WISH TO ENTER THE SPECIFICATIONS?(Y/N)'
      READ,CH
      IF(CH.EQ.'N') GOTO 5
      PRINT,
      PRINT,'PASS BAND  MAX RIPPLE  STOP BAND  MIN LOSS'
      PRINT,'EDGE FREQ  IN PASSBAND EDGE FREQ  IN STOPBAND'
      READ,FP,PR,FS,ML
 5    PRINT,
      PRINT,'               NO. OF POINTS '
      PRINT,' IN THE PASS BAND     IN THE STOP BAND'
      READ(5,*) NP,NS
      PRINT,
      DFP=FP/(NP-1)
      DFS=(0.5-FS)/(NS-1)
      PRINT,'ENTER COEFFICIENT WORD-LENGTH :-'
      READ(5,*) NSBIT
      PRINT,
      MFACT=0.5
      SD=(0.5)**NSBIT
      CALL QUANT(ALPHA,N,SD)
      CALL FEFUNC(N,ALPHA,E1,EP,ES,EOP)
      PRINT,
      WRITE(6,100) NSBIT
      DO 10 I=1,N
      WRITE(6,110) I,ALPHA(I)
 10   CONTINUE
      PRINT,
      PRINT,'MAX RIPPLE IN THE PASSBAND = ',EP
      PRINT,'MIN LOSS IN THE STOPBAND   = ',ES
      PRINT,
      PRINT,
      PRINT,'********************** DESIGN STAGE *****************'
      PRINT,
      PRINT,
      NFE=0
      CALL PAT(N,SD,NSBIT,MFACT,ALPHA)
      PRINT,
      PRINT,'TYPE <CR> TO CONTINUE !!'
      READ*,I
      PRINT,
 100  FORMAT(' WDF COEFFICIENTS QUANTIZED TO',I4,' BITS ARE :-')
 110  FORMAT(' ALPHA(',I4,') = ',F20.15)
      RETURN
      END
C
C SUBROUTINE QUANT :- THE INITIAL COEFFICIENTS ARE
C QUANTIZED TO NSBIT BITS, WHERE NSBIT IS THE STARTING
C BIT NUMBER.
C
      SUBROUTINE QUANT(ALPHA,N,MIND)
      REAL ALPHA(N),A,MIND
      INTEGER N,I
      DO 10 I=1,N
      A=ALPHA(I)
      ALPHA(I)=SIGN(MIND*FLOAT(INT(ABS(A)/MIND+0.5)),A)
 10   CONTINUE
      RETURN
      END
C
C SUBROUTINE PAT :- THIS SUBROUTINE MAKES A PATTERN
C MOVE TO FIND AN IMPROVED SET OF COEFFICIENTS. IT
C ALSO CHECKS FOR THE END OF ALGORITHM, I.E WHEN THE
C SPECIFICATIONS ARE MET.
C
      SUBROUTINE PAT(N,SD,NSBIT,MFACT,PCOEFF)
      REAL PCOEFF(N),SAVE(20),ECOEFF(20),PFV,EP,ES,EFV,DLT,
```

```
     &MFACT,SD,MIND
      INTEGER NFE
      LOGICAL EOP
      COMMON/B/ NFE
      CALL FEFUNC(N,PCOEFF,PFV,EP,ES,EOP)
      DLT=SD
      PRINT,'   NO. OF     NO. OF  MAX RIPPLE    MIN LOSS'
      PRINT,'FEFUNC CALLS   BITS   IN PASS BAND  IN STOP BAND'
 10   IF(EOP) GOTO 50
      CALL SETEQ(ECOEFF,PCOEFF,N)
      EFV=PFV
      CALL EXPLOR(ECOEFF,EFV,N,DLT,EOP)
      I=0
 20   IF(EFV.GE.PFV) GOTO 40
      I=I+1
      CALL FEFUNC(N,ECOEFF,DFV,EP,ES,EOP)
      WRITE(6,100) NFE,NSBIT,EP,ES
      REWIND(3)
      DO 25 J=1,N
      WRITE(3,*) ECOEFF(J)
 25   CONTINUE
      IF(EOP) GOTO 50
      CALL SETEQ(SAVE,PCOEFF,N)
      CALL SETEQ(PCOEFF,ECOEFF,N)
      PFV=EFV
      DO 30 J=1,N
      ECOEFF(J)=2*ECOEFF(J)-SAVE(J)
 30   CONTINUE
      CALL FEFUNC(N,ECOEFF,EFV,EP,ES,EOP)
      CALL EXPLOR(ECOEFF,EFV,N,DLT,EOP)
      CALL FEFUNC(N,ECOEFF,DFV,EP,ES,EOP)
      GOTO 20
 40   IF (I.GT.0) GOTO 10
      DLT=MFACT*DLT
      NSBIT=NSBIT+1
      GOTO 10
 50   PRINT,
      IF(NFE.EQ.1) GOTO 60
      CALL SETEQ(PCOEFF,ECOEFF,N)
 60   PRINT,'FINAL VALUES :-'
      PRINT,
      PRINT,'NO. OF FEFUNC CALLS =',NFE
      PRINT,'MAX RIPPLE IN PASS BAND =',EP
      PRINT,'MIN LOSS IN STOP BAND   =',ES
      PRINT,
      WRITE(6,110) NSBIT
      PRINT,
      DO 70 I=1,N
      WRITE(6,120) I,PCOEFF(I)
 70   CONTINUE
 100  FORMAT(3X,I6,7X,I2,4X,F11.9,4X,F11.9)
 110  FORMAT(' THE FINAL WDF COEFFICIENTS IN,',I4,' BITS ARE :-')
 120  FORMAT(' ALPHA(',I4,') = ',F20.15)
      RETURN
      END
```

```
C
C SUBROUTINE EXPLOR :- THIS SUBROUTINE MAKES AN
C EXPLOTARY MOVE. IT USES THE TWOPT TECHNIQUE TO
C SPEED UP THE ALGORITHM.
C
      SUBROUTINE EXPLOR(ECOEFF,EFV,N,DLT,EOP)
      REAL ECOEFF(N),EFV,DLT,SAVE,FNEW,SAVEI,SAVEJ
      INTEGER NCCOEF
      LOGICAL EOP
      COMMON/D/ NCCOEF
      I=NCCOEF+1
 10   IF(I.GT.N) RETURN
      SAVE=ECOEFF(I)
      DAMYI=ECOEFF(I)+DLT
      IF(DAMYI.GE.1 .OR. DAMYI.LE.-1) GOTO 100
      ECOEFF(I)=ECOEFF(I)+DLT
      CALL FEFUNC(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 20
      EFV=FNEW
      GOTO 100
 20   DAMYI=ECOEFF(I)-2*DLT
      IF(DAMYI.GE.1 .OR. DAMYI.LE.-1) GOTO 100
      ECOEFF(I)=ECOEFF(I)-2*DLT
      CALL FEFUNC(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 30
      EFV=FNEW
      GOTO 100
 30   ECOEFF(I)=SAVE
      J=I+1
 40   IF(J.GT.N) GOTO 100
      SAVEI=ECOEFF(I)
      SAVEJ=ECOEFF(J)
      DAMYI=ECOEFF(I)+DLT
      DAMYJ=ECOEFF(J)+DLT
      IF(DAMYI.GE.1 .OR. DAMYJ.GE.1 .OR. DAMYI.LE.-1 .OR.
     & DAMYJ.LE.-1) GOTO 90
      ECOEFF(I)=ECOEFF(I)+DLT
      ECOEFF(J)=ECOEFF(J)+DLT
      CALL FEFUNC(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 50
      EFV=FNEW
      GOTO 90
 50   DAMYI=ECOEFF(I)-2*DLT
      IF(DAMYI.GE.1 .OR. DAMYI.LE.-1) GOTO 90
      ECOEFF(I)=ECOEFF(I)-2*DLT
      CALL FEFUNC(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 60
      EFV=FNEW
      GOTO 90
 60   DAMYJ=ECOEFF(J)-2*DLT
      IF(DAMYJ.GE.1 .OR. DAMYJ.LE.-1) GOTO 90
      ECOEFF(J)=ECOEFF(J)-2*DLT
      CALL FEFUNC(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 70
      EFV=FNEW
      GOTO 90
 70   DAMYI=ECOEFF(I)+2*DLT
      IF(DAMYI.GE.1 .OR. DAMYI.LE.-1) GOTO 90
      ECOEFF(I)=ECOEFF(I)+2*DLT
      CALL FEFUNC(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 80
      EFV=FNEW
      GOTO 90
 80   ECOEFF(I)=SAVEI
      ECOEFF(J)=SAVEJ
 90   J=J+1
      GOTO 40
 100  I=I+1
      GOTO 10
      END
      SUBROUTINE SETEQ(A,B,N)
      REAL A(N),B(N)
      INTEGER N,I
      DO 10 I=1,N
        A(I)=B(I)
 10   CONTINUE
      RETURN
      END
C
C SUBROUTINE FEFUNC :- THIS SUBROUTINE CALCULATES
C THE OBJECTIVE FEFUNCION FOR THE CURRENT SET OF
C COEFFICIENTS. IT ALSO MAKE EOP EQUAL TO .TRUE.
C WHEN THE SPECIFICATIONS ARE MET.
C
      SUBROUTINE FEFUNC(N,XC,FC,EP,ES,EOP)
      REAL XC(N),FC,FREQ,MR,FS,DFP,DFS,MEP,MES,ML
      INTEGER N,I,NP,NS
      LOGICAL EOP
      COMMON/A/ FS,DFP,DFS,NP,NS,DP,PR,ML
      COMMON/B/ NFE
      MEP=0
      MES=0
      ES=10E5
      EOP=.FALSE.
      FREQ=0
      DO 30 I=1,NP
      CALL FRESP(XC,N,FREQ,MR)
      D=ABS(DP-MR)
      IF(D.LE.MEP) GOTO 20
      MEP=D
      EP=MR
 20   FREQ=FREQ+DFP
 30   CONTINUE
      FREQ=FS
      DO 70 I=1,NS
      CALL FRESP(XC,N,FREQ,MR)
      IF(MR.LT.ES) ES=MR
      D=ML-MR
 50   IF(D.LE.MES) GOTO 60
      MES=D
```

```
60    FREQ=FREQ+DFS
70    CONTINUE
      IF(MEP.LE.DP .AND. ES.GE.ML) EOP=.TRUE.
      IF(MES.GT.MEP) GOTO 80
      FC=MEP
      GOTO 90
80    FC=MES
90    NFE=NFE+1
      RETURN
      END
C
C SUBROUTINE FRESP :- GIVEN A SET OF COEFFICIENTS
C AND A FREQUENCY THEN THIS SUBROUTINE EVALUATES
C THE LOSS OF THE FILTER.
C
      SUBROUTINE FRESP(XC,N,FREQ,MR)
      REAL XC(N),FREQ,MR,W,SC,IM,RL,PI
      INTEGER K,N,MAG
      COMPLEX ABCD(2,2),DABCD(2,2),ZPM1
      COMMON/C/ MAG
      PI=3.14159263
      W=(2*PI*FREQ)
      ZPM1=CMPLX(COS(W),-SIN(W))
      CALL MATIDN(ABCD)
      DO 30 K=1,N
      IF(XC(K).EQ.1) PRINT,K,XC(K)
      SC=1/(1-XC(K))
      IF (K.EQ.N) GOTO 10
      DABCD(1,1)=CMPLX(SC,0)
      DABCD(1,2)=-XC(K)*ZPM1*SC
      DABCD(2,1)=CMPLX(-XC(K)*SC,0)
      DABCD(2,2)=ZPM1*SC
      GOTO 20
10    DABCD(1,1)=CMPLX(SC,0)
      DABCD(1,2)=CMPLX(-XC(K)*SC,0)
      DABCD(2,1)=DABCD(1,2)
      DABCD(2,2)=DABCD(1,1)
20    CALL MATMUL(ABCD,DABCD)
30    CONTINUE
      RL=REAL(ABCD(1,1))
      IM=AIMAG(ABCD(1,1))
      MR=1/((RL**2)+(IM**2))
      IF(MAG.EQ.1) GOTO 40
      MR=-20*ALOG10(SQRT(MR))
40    RETURN
      END
C
C SUBROUTINE MATIDN :- THIS SUBROUTINE MAKE A COMPLEX
C MATRIX EQUAL TO THE IDENTITY MATRIX.
C
      SUBROUTINE MATIDN(ABCD)
      COMPLEX ABCD(2,2)
      ABCD(1,1)=(1,0)
      ABCD(2,2)=(1,0)
      ABCD(1,2)=(0,0)
      ABCD(2,1)=(0,0)
      RETURN
      END
C
C SUBROUTINE MATMUL :- THIS SUBROUTINE MULTIPLIES TWO
C COMPLEX MATRICES AND LEAVE THE RESULTS IN THE FIRST
C MATRIX.
C
      SUBROUTINE MATMUL(ABCD,DABCD)
      COMPLEX ABCD(2,2),DABCD(2,2),C(2,2)
      C(1,1)=ABCD(1,1)*DABCD(1,1)+ABCD(1,2)*DABCD(2,1)
      C(1,2)=ABCD(1,1)*DABCD(1,2)+ABCD(1,2)*DABCD(2,2)
      C(2,1)=ABCD(2,1)*DABCD(1,1)+ABCD(2,2)*DABCD(2,1)
      C(2,2)=ABCD(2,1)*DABCD(1,2)+ABCD(2,2)*DABCD(2,2)
      ABCD(1,1)=C(1,1)
      ABCD(1,2)=C(1,2)
      ABCD(2,1)=C(2,1)
      ABCD(2,2)=C(2,2)
      RETURN
      END
```

B2.2- LTFD.SO PROGRAM

```
C------------------------------------------------------
C
C              WDF DEGIN PROGRAM
C     BASED ON SYNTHESIS AND OPTIMIZATION OF THE
C     LATTICE FILTERS.(FORTRAN 77)
C
C
C AUTHOR : A. R. MIRZAI            DATE : DEC 1985
C
C           THE CENTRE FOR INFORMATION ENG
C                THE CITY UNIVERSITY
C           NORTHAMPTON SQ. LONDON EC1VOHB
C
C
C-----------------------------------------------------
      REAL ALPHA(20)
      INTEGER CHOIC,NBIT
      CHARACTER*1 CH
      COMMON /A/ ES,EP,PHS,PHP
      COMMON /B/ FP,FS,AP,AS,F
      OPEN(UNIT=1,FILE='LTCOEF.I',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='READ/WRITE')
      OPEN(UNIT=2,FILE='LTCOEF.F',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='READ/WRITE')
      OPEN(UNIT=3,FILE='LTCOEF.T',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='READ/WRITE')
 5    PRINT,
      PRINT,
      PRINT,'             PROGRAM TO DESIGN LATTICE WDF'
      PRINT,
      PRINT,'                       MAIN MENU'
      PRINT,
      PRINT,'      1) ENTER SPECIFICATIONS.'
      PRINT,'      2) READ INITIAL COEFFICIENTS.'
      PRINT,'      3) DESIGN BUTTERWORTH FILTER.'
      PRINT,'      4) DESIGN CHEBYCHEV FILTERS.'
      PRINT,'      5) DESIGN ELLIPTIC FILTERS.'
      PRINT,'      6) DESIGN FINIT-WORD LENGTH FILTER.'
      PRINT,'      7) SAVE COEFFICIENTS.'
      PRINT,'      8) END PROGRAM.'
      PRINT,
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ,CHOIC
      PRINT,
      GOTO(10,20,30,40,50,60,70,80),CHOIC
      GOTO 5
 10   CALL RDSPEC(N)
      GOTO 5
 20   CALL RDCOEF(N,ALPHA)
      GOTO 5
 30   CALL DBUTT(N,ALPHA)
      GOTO 5
 40   CALL DCHEB(N,ALPHA)
      GOTO 5
 50   CALL DELLIP(N,ALPHA)
      GOTO 5
 60   CALL DFWL(ALPHA,N)
      GOTO 5
 70   CALL SAVE(ALPHA,N)
      GOTO 5
 80   STOP
      END
C
C
C
      SUBROUTINE RDSPEC(N)
      COMMON /A/ ES,EP,PHS,PHP
      COMMON /B/ FP,FS,AP,AS,F
      PI=3.14159263
      PRINT,
      PRINT,'PASS BAND   STOP BAND   MAX RIPPLE   MIN LOSS   '
      PRINT,'FREQ (HZ)   FREQ (HZ)      (DB)        (DB)     '
      PRINT,
      READ,FP,FS,AP,AS
      F=1.0
      PRINT,
      ES=SQRT(10**(AS/10)-1)
      EP=SQRT(10**(AP/10)-1)
      PHS=TAN(PI*FS/F)
      PHP=TAN(PI*FP/F)
      RK=SQRT(PHS/PHP)
      MINN=INT(ALOG(ES/EP)/ALOG(RK**2))
      PRINT,
      PRINT,'THE FILTER ORDER MUST BE GREATER THAN '
      PRINT,
      WRITE(6,100) MINN
      RK=(RK**2)+SQRT(RK**4-1)
      MINN=INT(ALOG(2*ES/EP)/ALOG(RK))
      WRITE(6,110) MINN
      DO 10 I=1,3
      RK=(RK**2)+SQRT(RK**4-1)
 10   CONTINUE
      MINN=INT(8*ALOG(4*ES/EP)/ALOG(2*RK))
      WRITE(6,120) MINN
      PRINT,
      PRINT,'ENTER THE REQUIRED FILTER ORDER :-'
      READ,N
      PRINT,
 100  FORMAT('          1) ',I4,' FOR A BUTTERWORTH TYPE, OR')
 110  FORMAT('          2) ',I4,' FOR A CHEBYCHEV TYPE, OR')
 120  FORMAT('          3) ',I4,' FOR A ELLIPTIC TYPE FILTER')
      RETURN
      END
C
C
C
      SUBROUTINE RDCOEF(N,ALPHA)
      REAL ALPHA(N)
      INTEGER N,I
      CHARACTER*8 FILENAME
5     PRINT,
      PRINT,'           1) READ COEFFICIENTS FROM TERMINAL.'
      PRINT,'           2) READ COEFFICIENTS FROM LTCOEF.I.'
      PRINT,'           3) READ COEFFICIENTS FROM LTCOEF.F.'
      PRINT,'           4) READ COEFFICIENTS FROM LTCOEF.T.'
      PRINT,
      PRINT,'ENTER YOUR CHIOCE NO. :-'
      READ,I
      GOTO (10,20,30,40),I
      GOTO 5
10    PRINT,
      PRINT,'ENTER INITIAL COEFFICIENTS :-'
      READ,(ALPHA(I),I=1,N)
      RETURN
20    REWIND(1)
      READ(1,*) (ALPHA(I),I=1,N)
      ENDFILE(UNIT=1)
      RETURN
30    REWIND(2)
      READ(2,*) (ALPHA(I),I=1,N)
      ENDFILE(UNIT=2)
      RETURN
40    REWIND(3)
      READ(3,*) (ALPHA(I),I=1,N)
      ENDFILE(UNIT=3)
      RETURN
      END
C
C
C
      SUBROUTINE DBUTT(N,ALPHA)
      REAL ALPHA(20)
      COMMON /A/ ES,EP,PHS,PHP
      PI=3.14159263
      RKS=((ES**(2.0/N))-PHS**2.0)/((ES**(2.0/N))+PHS**2.0)
      RKP=((EP**(2.0/N))-PHP**2.0)/((EP**(2.0/N))+PHP**2.0)
      PRINT,
      PRINT,'KP  =',RKP
      PRINT,'KS  =',RKS
      PRINT,
      IF(RKP.LT.0 .OR. RKS.GT.0) GOTO 10
      PRINT,'NOTE !!!!'
      PRINT,'YOU CAN CHOOSE TO DESIGN BIRECIPROCAL FILTER BY'
      PRINT,'SETTING THE VALUE OF THE PARAMETER Y EQUAL TO ZERO.'
      PRINT,
10    PRINT,'ENTER THE VALUE OF THE Y PARAMETER (KS<=Y<=KP) :-'
      READ,Y
      PRINT,
      DUMY=SQRT(1-Y**2)
      PRINT,'THE WDF COEFFICIENTS ARE AS FOLLOWS :-'
      PRINT,
      ALPHA(1)=(1+Y-DUMY)/(1+Y+DUMY)
      PRINT,'ALPHA(',0,') =',ALPHA(1)
      J=1
      DO 20 I=1,(N-1)/2
      DUMY=SQRT(1-Y**2)*COS(PI*I/N)
      ALPHA(I+J)=(DUMY-1)/(DUMY+1)
      ALPHA(I+J+1)=Y
      PRINT,'ALPHA(',I+J-1,') =',ALPHA(I+J)
      PRINT,'ALPHA(',I+J,') =',ALPHA(I+J+1)
      J=J+1
20    CONTINUE
      PRINT,
      PRINT,
      PRINT,'TYPE <CR> TO CONTINUE !!!'
      READ,I
      PRINT,
      RETURN
      END
C
C
C
      SUBROUTINE DCHEB(N,ALPHA)
      REAL ALPHA(20)
      COMMON /A/ ES,EP,PHS,PHP
      PI=3.14159263
      RK=SQRT(PHS/PHP)
      RK=(RK**2)+SQRT(RK**4-1)
      EPMIN=(2*ES)/(RK**N)
      AP=10*ALOG10(1+EP**2)
      APMIN=10*ALOG10(1+EPMIN**2)
      PRINT,
      PRINT,'MIN AP =',APMIN,' DB'
      PRINT,'MAX AP =',AP,' DB'
      PRINT,
      PRINT,'ENTER REQUIRED AP (DB) :-'
      READ,AP
      PRINT,
      EP=SQRT(10**(AP/10)-1)
      W=(1.0/EP+SQRT(1.0/(EP**2)+1.0))**(1.0/N)
      R=(W-1/W)*PHP
      PRINT,
      PRINT,'THE WDF COEFFICIENTS ARE AS FOLLOWS :-'
      PRINT,
      ALPHA(1)=(2-R)/(2+R)
      PRINT,'ALPHA(',0,') =',ALPHA(1)
      J=1
      DO 10 I=1,(N-1)/2
      AI=R*COS(PI*I/N)
      BI=(W**2+(1/(W**2))-2*COS(2*PI*I/N))*(PHP**2)/4
      ALPHA(I+J)=(AI-BI-1)/(AI+BI+1)
      ALPHA(I+J+1)=(1-BI)/(1+BI)
      PRINT,'ALPHA(',I+J-1,') =',ALPHA(I+J)
      PRINT,'ALPHA(',I+J,') =',ALPHA(I+J+1)
      J=J+1
10    CONTINUE
      PRINT,
      PRINT,'TYPE <CR> TO CONTINUE !!!'
      READ,I
      PRINT,
      RETURN
      END
C
C
C
      SUBROUTINE DELLIP(N,ALPHA)
      REAL Q(0:10),M(0:10),G(0:10),C(0:10,10),Y(10),BA(10),AA(10)
     &,W(0:10),ALPHA(20)
      CHARACTER*1 CH
      COMMON /A/ ES,EP,PHS,PHP
      COMMON /B/ FP,FS,AP,AS,F
      PI=3.14159263
      R=SQRT(ES/EP)
      R=R**2+SQRT((R**4)-1)
      R=R**2+SQRT((R**4)-1)
      X=((2*R)**(4.0/N))/2
      DO 20 I=1,4
      X=SQRT((X+(1/X))/2)
 20   CONTINUE
      FSMIN=(F*ATAN(PHP*(X**2)))/PI
      PRINT,
      PRINT,'MIN FS =',FSMIN
      PRINT,'MAX FS =',FS
      PRINT,
      PRINT;'ENTER REQUIRED FS :-'
      READ,FS
      PRINT,
      PHS=TAN(PI*FS/F)
      Q(0)=SQRT(PHS/PHP)
      DO 30 I=0,3
      Q(I+1)=(Q(I)**2)+SQRT((Q(I)**4)-1)
 30   CONTINUE
      M(3)=((SQRT(2*Q(4)))**N)/2
      DO 40 I=3,1,-1
      M(I-1)=SQRT((M(I)+(1/M(I)))/2)
 40   CONTINUE
      EPMIN=ES/(M(0)**2)
      APMIN=10*ALOG10(1+EPMIN**2)
      AP=10*ALOG10(1+EP**2)
      PRINT,
      PRINT,'MIN AP =',APMIN,' DB'
      PRINT,'MAX AP =',AP,' DB'
      PRINT,
      PRINT,'ENTER REQUIRED AP (DB) :-'
      READ,AP
      PRINT,
      EP=SQRT(10**(AP/10)-1)
      ES=EP*(M(0)**2)
      G(1)=(1/EP)+SQRT((1/EP**2)+1)
      DO 50 I=1,2
      G(I+1)=(M(I)*G(I))+SQRT(((M(I)*G(I))**2)+1)
 50   CONTINUE
      DUMY=M(3)/G(3)
      W(5)=(DUMY+SQRT((DUMY**2)+1))**(1.0/N)
      DO 60 I=5,1,-1
      W(I-1)=(W(I)-(1/W(I)))/(2*Q(I-1))
 60   CONTINUE
      DUMY=W(0)*Q(0)*PHP
      PRINT,
      PRINT,'THE WDF COEFFICIENTS ARE AS FOLLOWS :-'
      PRINT,
      ALPHA(1)=(1+DUMY)/(1-DUMY)
      PRINT,'ALPHA(',0,') =',ALPHA(1)
      DO 80 I=1,(N-1)/2
      C(4,I)=Q(4)/SIN(I*PI/N)
      DO 70 J=4,1,-1
      C(J-1,I)=(1/(2*Q(J-1)))*(C(J,I)+(1/C(J,I)))
 70   CONTINUE
      Y(I)=1/C(0,I)
      BA(I)=((W(0)**2)+(Y(I)**2))*((Q(0)*PHP)**2)/(1+(W(0)*Y(I))**2)
      AA(I)=-2*W(0)*Q(0)*PHP*SQRT(1-(((Q(0)**2)+(1/(Q(0)**2))-(Y(I)
     &**2))*Y(I)**2))/(1+((W(0)*Y(I))**2))
 80   CONTINUE
 90   J=1
      DO 100 I=1,(N-1)/2
      ALPHA(I+J)=(AA(I)-BA(I)-1)/(AA(I)+BA(I)+1)
      ALPHA(I+J+1)=(1-BA(I))/(1+BA(I))
      PRINT,'ALPHA(',I+J-1,') =',ALPHA(I+J)
      PRINT,'ALPHA(',I+J,') =',ALPHA(I+J+1)
      J=J+1
 100  CONTINUE
      PRINT,
      PRINT,'THE CRITICAL FREQUECNCIES ARE AS FOLLOWS :-'
      PRINT,
      PRINT,'                        TRANSMISSION        ZERO PASSBAND'
      PRINT,'                            ZERO                LOSS    '
      PRINT,
      DUMY=Q(0)*PHP
      DO 110 I=1,(N-1)/2
      FTZ=F*ATAN(DUMY/Y(I))/PI
      FZP=F*ATAN(DUMY*Y(I))/PI
      PRINT,I,FTZ,FZP
 110  CONTINUE
      PRINT,'TYPE <CR> TO CONTINUE !!!'
      READ,LL
      RETURN
      END
C
C
C
      SUBROUTINE FRESP(ALPHA,N,FREQ,MR)
      REAL ALPHA(20),FREQ,MR
      INTEGER IC1,IC2
      COMPLEX ZPM1,ZPM2,G1Z,G2Z,GZ
      COMMON /B/ FP,FS,AP,AS,F
      PI=3.14159263
      G1Z=(1,0)
      G2Z=(1,0)
      W=(2*PI*FREQ)/F
      ZPM1=CMPLX(COS(W),-SIN(W))
      ZPM2=ZPM1**2
      G1Z=(ZPM1-ALPHA(1))/(1-ALPHA(1)*ZPM1)
      G2Z=(1,0)
      IC1=1
      IC2=2
14    CALL TRANSF(ALPHA(IC2),ALPHA(IC2+1),G2Z,ZPM1,ZPM2)
      IC1=IC1+2
      IF(IC1.EQ.N) GOTO 18
      CALL TRANSF(ALPHA(IC2+2),ALPHA(IC2+3),G1Z,ZPM1,ZPM2)
      IC1=IC1+2
      IC2=IC2+4
      IF(IC1.NE.N) GOTO 14
18    GZ=(G1Z+G2Z)/2
      MR=-20*ALOG10(CABS(GZ))
      RETURN
      END
C
C
C
      SUBROUTINE TRANSF(A1,A2,G,Z1,Z2)
      REAL A1,A2
      COMPLEX G,Z1,Z2,G1
      G1=Z2+A2*(A1-1)*Z1-A1
      G1=G1/(1+A2*(A1-1)*Z1-A1*Z2)
      G=G*G1
      RETURN
      END
C
C
C
      SUBROUTINE QUANT(ALPHA,N,MIND)
      REAL ALPHA(N),A,MIND
      INTEGER N,I
      DO 10 I=1,N
      A=ALPHA(I)
      ALPHA(I)=SIGN(MIND*FLOAT(INT(ABS(A)/MIND+0.5)),A)
10    CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE DFWL(ALPHA,N)
      REAL ALPHA(N),MIND,SD,DUMMY,MFACT,EST,F
     &DFP,DFS,FP,FS,E1,ML
      INTEGER NBIT,NSBIT,MNFE,NFE,N,TWOPT,NP,NS
      LOGICAL EOP
      COMMON /B/ FP,FS,AP,AS,F
      COMMON /C/ DFP,DFS,NP,NS,DP,RP,ML
      COMMON /D/ NFE
      PRINT,
      PRINT,
      PRINT,'      FINIT-WORD LENGTH DESIGN PROGRAM'
      PRINT,
      PRINT,
      PRINT,
      PRINT,
      PRINT,'MAX RIPPLE IN THE PASSBAND, AP =',AP
      PRINT,'MIN LOSS IN THE STOPBAND, AS   =',AS
      PRINT,
      PRINT,'ENTER REQUIRED AP AND AS :-'
      READ,AP,AS
      PRINT,
      ML=AS
      DP=AP/2
      RP=AP
      PRINT,
      NP=INT(FP*100)+1
      NS=INT((0.5-FS)*100)+1
      PRINT,NP,NS
      DFP=FP/(NP-1)
      DFS=((F/2)-FS)/(NS-1)
      PRINT,'ENTER THE COEFFICIENT WORD-LENGTH :-'
      READ,NSBIT
      PRINT,
      MFACT=0.5
      SD=(0.5)**NSBIT
      CALL QUANT(ALPHA,N,SD)
      CALL FEFUNC(N,ALPHA,E1,EP,ES,EOP)
      PRINT,'INITIAL VALUES :-'
      PRINT,
      WRITE(6,100) NSBIT
      PRINT,
      DO 10 I=1,N
      WRITE(6,110) I,ALPHA(I)
10    CONTINUE
      PRINT,
      PRINT,'MAX RIPPLE IN THE PASSBAND =',EP
      PRINT,'MIN LOSS IN THE STOPBAND   =',ES
      PRINT,
      PRINT,
      PRINT,'******************* DESIGN STAGE *********************'
      PRINT,
      PRINT,
      NFE=0
      CALL PAT(N,SD,NSBIT,MFACT,ALPHA)
      PRINT,
      PRINT,'TYPE <CR> TO CONTINUE !!!'
      READ,I
      PRINT,
100   FORMAT(' WDF COEFFICIENTS QUANTIZED TO',I4,' BITS ARE :-')
110   FORMAT(' ALPHA(',I4,') = ',F20.15)
      RETURN
      END
C
C
C
      SUBROUTINE PAT(N,SD,NSBIT,MFACT,PCOEFF)
      REAL PCOEFF(N),SAVE(20),ECOEFF(20),PFV,EP,ES,EFV,DLT,
     &MFACT,SD,MIND
      INTEGER NFE
      LOGICAL EOP
      COMMON /D/ NFE
      CALL FEFUNC(N,PCOEFF,PFV,EP,ES,EOP)
      DLT=SD
      PRINT,'  NO. OF     NO. OF   MAX RIPPLE      MIN LOSS'
      PRINT,'FEFUN CALL    BITS    IN PASS BAND   IN STOP BAND'
 10   IF(EOP) GOTO 50
      CALL SETEQ(ECOEFF,PCOEFF,N)
      EFV=PFV
      CALL EXPLOR(ECOEFF,EFV,N,DLT,EOP)
      I=0
 20   IF(EFV.GE.PFV) GOTO 40
      I=I+1
      CALL FEFUNC(N,ECOEFF,DFV,EP,ES,EOP)
      WRITE(6,100) NFE,NSBIT,EP,ES
      REWIND(3)
      DO 25 J=1,N
      WRITE(3,*) ECOEFF(J)
 25   CONTINUE
      IF(EOP) GOTO 50
      CALL SETEQ(SAVE,PCOEFF,N)
      CALL SETEQ(PCOEFF,ECOEFF,N)
      PFV=EFV
      DO 30 J=1,N
      ECOEFF(J)=2*ECOEFF(J)-SAVE(J)
 30   CONTINUE
      CALL FEFUNC(N,ECOEFF,EFV,EP,ES,EOP)
      CALL EXPLOR(ECOEFF,EFV,N,DLT,EOP)
      CALL FEFUNC(N,ECOEFF,DFV,EP,ES,EOP)
      GOTO 20
 40   IF (I.GT.0) GOTO 10
      DLT=MFACT*DLT
      NSBIT=NSBIT+1
      GOTO 10
 50   PRINT,
      IF(NFE.EQ.1) GOTO 60
      CALL SETEQ(PCOEFF,ECOEFF,N)
 60   PRINT,'FINAL VALUES :-'
      PRINT,
      PRINT,'NO. OF FEFUN CALLS      =',NFE
      PRINT,'MAX RIPPLE IN PASS BAND =',EP
      PRINT,'MIN LOSS IN STOP BAND   =',ES
      PRINT,
      WRITE(6,110) NSBIT
      PRINT,
      DO 70 I=1,N
      WRITE(6,120) I,PCOEFF(I)
 70   CONTINUE
 100  FORMAT(3X,I6,7X,I2,4X,F11.9,4X,F11.9)
 110  FORMAT(' THE FINAL WDF COEFFICIENTS IN',I4,' BITS ARE :-')
 120  FORMAT(' ALPHA(',I4,') = ',F20.15)
      RETURN
```

```
      END
C
C
C
      SUBROUTINE EXPLOR(ECOEFF,EFV,N,DLT,EOP)
      REAL ECOEFF(N),EFV,DLT,SAVE,FNEW,SAVEI,SAVEJ
      INTEGER LIST(20),TWOPT
      LOGICAL EOP
      ICOUNT=1
 10   IF(ICOUNT.GT.N) RETURN
      I=ICOUNT
      SAVE=ECOEFF(I)
      ECOEFF(I)=ECOEFF(I)+DLT
      CALL FEFUNC(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 20
      EFV=FNEW
      GOTO 100
 20   ECOEFF(I)=ECOEFF(I)-2*DLT
      CALL FEFUNC(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 30
      EFV=FNEW
      GOTO 100
 30   ECOEFF(I)=SAVE
      IF(TWOPT.EQ.1) GOTO 100
      JCOUNT=ICOUNT+1
 40   IF(JCOUNT.GT.N) GOTO 100
      J=JCOUNT
      SAVEI=ECOEFF(I)
      SAVEJ=ECOEFF(J)
      ECOEFF(I)=ECOEFF(I)+DLT
      ECOEFF(J)=ECOEFF(J)+DLT
      CALL FEFUNC(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 50
      EFV=FNEW
      GOTO 90
 50   ECOEFF(I)=ECOEFF(I)-2*DLT
      CALL FEFUNC(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 60
      EFV=FNEW
      GOTO 90
 60   ECOEFF(J)=ECOEFF(J)-2*DLT
      CALL FEFUNC(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 70
      EFV=FNEW
      GOTO 90
 70   ECOEFF(I)=ECOEFF(I)+2*DLT
      CALL FEFUNC(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 80
      EFV=FNEW
      GOTO 90
 80   ECOEFF(I)=SAVEI
      ECOEFF(J)=SAVEJ
 90   JCOUNT=JCOUNT+1
      GOTO 40
 100  ICOUNT=ICOUNT+1
```

```
      GOTO 10
      END
C
C
C
      SUBROUTINE SETEQ(A,B,N)
      REAL A(N),B(N)
      INTEGER N,I
      DO 10 I=1,N
      A(I)=B(I)
 10   CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE FEFUNC(N,XC,FC,EP,ES,EOP)
      REAL XC(N),FC,FREQ,MR,ML,FS,DFP,DFS,MEP,MES
      INTEGER N,I,NP,NS
      LOGICAL EOP
      COMMON /B/ FP,FS,AP,AS,F
      COMMON /C/ DFP,DFS,NP,NS,DP,RP,ML
      COMMON /D/ NFE
      MEP=0
      MES=0
      EP=0
      ES=10E5
      EOP=.FALSE.
      FREQ=0
      DO 30 I=1,NP
      CALL FRESP(XC,N,FREQ,MR)
      D=ABS(DP-MR)
      IF(D.LE.EP) GOTO 20
      MEP=D
      EP=MR
 20   FREQ=FREQ+DFP
 30   CONTINUE
      FREQ=FS
      DO 70 I=1,NS
      CALL FRESP(XC,N,FREQ,MR)
      IF(MR.LT.ES) ES=MR
      D=ML-MR
 50   IF(D.LE.MES) GOTO 60
      MES=D
 60   FREQ=FREQ+DFS
 70   CONTINUE
      IF(MEP.LE.DP .AND. ES.GE.ML) EOP=.TRUE.
      FC=MEP
      IF(MES.GT.MEP) FC=MES
 80   NFE=NFE+1
      RETURN
      END
C
C
C
```

```
      SUBROUTINE SAVE(ALPHA,N)
      REAL ALPHA(N)
      INTEGER N,I
      PRINT,
 5    PRINT,
      PRINT,'         1) INFINIT PRECISION COEFFICIENTS, OR'
      PRINT,'         2) FINITE PRECISION COEFFICIENTS.'
      PRINT,
      PRINT,'ENTER YOUR CHOICE NO. :-'
      PRINT,
      READ,I
      GOTO (10,20),I
      PRINT,
      GOTO 5
 10   REWIND(1)
      WRITE(1,*) (ALPHA(I),I=1,N)
      ENDFILE(UNIT=1)
      RETURN
 20   REWIND(2)
      WRITE(2,*) (ALPHA(I),I=1,N)
      ENDFILE(UNIT=2)
      RETURN
      END
```

B2.3- LCFD.SO PROGRAM

```
C-----------------------------------------------------------
C
C                WDF DESIGN PROGRAM
C     BASED ON SYNTHESIS AND OPTIMIZATION OF THE
C     LC-LADDER FILTERS.(FORTRAN 77)
C
C
C AUTHOR : A. R. MIRZAI          DATE : MARCH 1986
C
C             THE CENTRE FOR INFORMATION ENG
C                  THE CITY UNIVERSITY
C             NORTHAMPTON SQ. LONDON EC1VOHB
C
C
C-----------------------------------------------------------
      REAL ALPHA(60),COMPVAL(20),RL,RS,FP,FS,AP,AS
      INTEGER NSECT,I,CIR(20)
      COMMON /A1/ COMPVAL,CIR,RS,RL
      COMMON /B1/ FP,AP,FS,AS
      OPEN(UNIT=1,FILE='LCCOEF.I',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='READ/WRITE')
      OPEN(UNIT=2,FILE='LCCOEF.F',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='READ/WRITE')
      OPEN(UNIT=3,FILE='LCCOEF.T',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='READ/WRITE')
 5    PRINT,
      PRINT,'                PROGRAM TO DESIGN AND SIMULATE'
      PRINT,'                       LC-LADDER WDFS'
      PRINT,
      PRINT,'                           MAIN MENU'
      PRINT,
      PRINT,'              1) READ INITIAL COEFFICIENTS.'
      PRINT,'              2) DESIGN LC-LADDER FILTERS.'
      PRINT,'              3) DESIGN LC-LADDER WDFS.'
      PRINT,'              4) DESIGN FINIT-WORD LENGTH WDFS.'
      PRINT,'              5) SAVE COEFFICIENTS.'
      PRINT,'              6) END  PROGRAM.'
      PRINT,
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ,I
      GOTO (10,20,30,40,50,60),I
      I=0
      GOTO 5
 10   CALL RDCOEF(ALPHA,NSECT)
      GOTO 5
 20   CALL DLCLAD(NSECT)
      GOTO 5
 30   CALL DLCWDF(ALPHA,NSECT)
      GOTO 5
 40   CALL DFWLF(ALPHA,NSECT)
      GOTO 5
 50   CALL SAVE(ALPHA,NSECT)
      GOTO 5
```

```
 60   STOP
      END
C
C
C
      SUBROUTINE RDCOEF(ALPHA,N)
      REAL ALPHA(60),FP,FS,AP,AS
      INTEGER I,J,N
      PRINT,
      PRINT,'ENTER THE FILTER ORDER :-'
      READ,N
      PRINT,
      PRINT,
      PRINT,'YOU CAN READ THE INITIAL COEFFICIENTS FROM,'
      PRINT,
      PRINT,'        1) LCCOEF.I (INFINITE PRECISION),OR'
      PRINT,'        2) LCCOEF.F (FINITE  PRECISION) ,OR'
      PRINT,'        3) LCCOEF.T (A TEMPORARY FILE ) ,OR'
      PRINT,'        4) ENTER COEFFICIENTS FROM THE TERMINAL.'
      PRINT,
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ,J
      PRINT,
      IF(J.EQ.4) GOTO 20
      REWIND(J)
      DO 10 I=1,N*2
      READ(J,*) ALPHA(I)
 10   CONTINUE
      GOTO 30
 20   PRINT,
      PRINT,'ENTER THE INITIAL COEFFICIENTS :-'
      PRINT,
      READ,(ALPHA(I),I=1,N*2)
      PRINT,
 30   PRINT,
      RETURN
      END
C
C
C
      SUBROUTINE DLCLAD(N)
      REAL A(20),RS,RL
      INTEGER N,CIR(10)
      COMMON /A1/ A,CIR,RS,RL
      COMMON /B1/ FP,AP,FS,AS
      PI=3.14159263
      PRINT,
      PRINT,
      PRINT,'PROGRAM TO DESIGN BUTTERWORTH AND CHEBYSHEV LC-LADDER'
      PRINT,'FILTERS USING EXPLICIT FOURMULIES.'
      PRINT,'[                                   ]'
      PRINT,
      PRINT,
      PRINT,
      PRINT,
      PRINT,'PASS BAND  MAX RIPPLE  STOP BAND  MIN LOSS'
      PRINT,'EDGE FREQ  IN PASSBAND EDGE FREQ  IN STOPBAND'
      READ,FP,AP,FS,AS
      FPD=FP
      FSD=FS
      PRINT,
      FP=(TAN(PI*FP)/(2*PI))
      FS=(TAN(PI*FS)/(2*PI))
      PRINT,
      EP=SQRT((10**(AP/10))-1)
      ES=SQRT((10**(AS/10))-1)
      OMGS=FS/FP
      NMIN=INT((ALOG(ES/EP))/ALOG(OMGS))
      PRINT,
      PRINT,'THE FILTER ORDER MUST BE GREATER THAN :-'
      PRINT,
      WRITE(6,100) NMIN
      G=SQRT((ES**2)/(EP**2))
      D1=ALOG(G+SQRT((G**2)-1))
      D2=ALOG(OMGS+SQRT((OMGS**2)-1))
      NMIN=INT(D1/D2)
      WRITE(6,110) NMIN
      PRINT,
      PRINT,'ENTER FILTER TYPE AND ORDER :-'
      READ,I,N
      PRINT,
      IF(I.EQ.1) GOTO 10
      CALL DCHEB(A,N,EP,RS,RL,FP)
      GOTO 20
 10   CALL DBUTT(A,N)
 20   PRINT,
      CIR(1)=1
      DO 30 K=2,N
      CIR(K)=1-CIR(K-1)
 30   CONTINUE
      PRINT,
      FP=FPD
      FS=FSD
 100  FORMAT(' 1) FOR BUTTERWORTH ,',I4)
 110  FORMAT(' 2) FOR CHEBYSHEV   ,',I4)
      RETURN
      END
C
C
C
      SUBROUTINE DBUTT(A,N)
      REAL A(20)
      INTEGER N
      PI=3.14159263
      DO 10 K=1,N
      D1=(2*K-1)*180/(2*N)
      A(K)=2*SIN(D1*PI/180)
      PRINT,'COMP( ',K,') = ',A(K)
 10   CONTINUE
      PRINT,
```

```
      RETURN
      END
C
C
C
      SUBROUTINE DCHEB(A,N,EP,RS,RL,FP)
      REAL A(20),PI,EP,H,E
      INTEGER N
      PI=3.14159263
      H=((1.0/EP)+SQRT(1+1/(EP**2)))**(1.0/N)
      E=(H-(1/H))
      A(1)=(4*SIN(PI/(2*N)))/E
      M=INT(N/2)
      M1=M
      IF(M*2.EQ.N) M1=M1-1
      DO 10 K=1,M1
      D1=16*SIN((4*K-3)*PI/(2*N))*SIN((4*K-1)*PI/(2*N))
      D2=(E**2)+4*((SIN((2*K-1)*PI/N))**2)
      A(2*K)=D1/(D2*A(2*K-1))
      D1=16*SIN((4*K-1)*PI/(2*N))*SIN((4*K+1)*PI/(2*N))
      D2=(E**2)+4*((SIN(2*K*PI/N))**2)
      A(2*K+1)=D1/(D2*A(2*K))
 10   CONTINUE
      IF(2*M.NE.N) GOTO 20
      D1=16*SIN((4*M-3)*PI/(2*N))*SIN((4*M-1)*PI/(2*N))
      D2=(E**2)+4*((SIN((2*M-1)*PI/N))**2)
      A(N)=D1/(D2*A(N-1))
      RL=(E*A(N))/(4*SIN(PI/(2*N)))
      GOTO 30
 20   RL=(4*SIN(PI/(2*N)))/(E*A(N))
 30   PRINT,
      PRINT,'THE NORMALIZED VALUES OF THE FILTER ELEMENTS ARE :-'
      PRINT,
      DO 40 K=1,N
      PRINT,K,A(K)
 40   CONTINUE
      PRINT,
      PRINT,'LOAD RESISTOR =',RL
      PRINT,
      PRINT,'ENTER VALUE OF THE SOURCE RESISTOR :-'
      READ,RS
      DO 50 K=1,N
      A(K)=A(K)/(2*PI*FP)
 50   CONTINUE
      K=0
 60   A(K+1)=A(K+1)/RS
      A(K+2)=A(K+2)*RS
      K=K+2
      IF(K.EQ.N) GOTO 80
      IF(K.EQ.(N-1)) GOTO 70
      GOTO 60
 70   A(N)=A(N)/RS
 80   RL=RL*RS
      PRINT,
      PRINT,'THE DENORMALIZED VALUES ARE :-'
      PRINT,
      DO 90 K=1,N
      PRINT,K,A(K)
90    CONTINUE
      PRINT,
      PRINT,'THE LOAD RESISTOR =',RL
      PRINT,
      PRINT,'TYPE <CR> TO CONTINUE !!!'
      READ,L
      RETURN
      END
C
C
C
      SUBROUTINE DLCWDF(ALPHA,N)
      REAL COMPVAL(20),UNIT(20),RECT(50),ALPHA(60)
     &,R1,R2,R3,G1,G2,G3,UNITELEM
      INTEGER CIR(10),I,J,COUNT1,COUNT2,N
      LOGICAL FLAG
      CHARACTER*1 CH
      COMMON /A1/ COMPVAL,CIR,RS,RL
      PRINT,
      PRINT,'DO YOU WISH TO ENTER THE COMPONENT VALUES?(Y/N)'
      READ,CH
      PRINT,
      IF(CH.EQ.'N') GOTO 22
      PRINT,
      PRINT,
      PRINT,
      PRINT,
      PRINT,"PROGRAM TO DESIGN WDF'S BASED ON THE "
      PRINT,'LC LADDER STRUCTURES.'
      PRINT,
      PRINT,
      PRINT,'THE LC LADDER FILTER SHOULD BE DESCRIBED BY'
      PRINT,'USING "0" FOR AN INDUCTOR AND "1" FOR A CAPACITOR.'
      PRINT,
      PRINT,' E.G 0,1 FOR A 1H INDUCTOR AND'
      PRINT,'    1,1 FOR A 1F CAPACITOR.'
      PRINT,
10    PRINT,
      PRINT,
      PRINT,'ENTER THE FILTER ORDER :-'
      READ,N
      PRINT,
      FLAG=.FALSE.
      PRINT,'ENTER VALUES OF THE SOURCE AND LOAD RESISTORS :-'
      READ,RS,RL
      PRINT,
      PRINT,'ENTER THE COMPONENTS AND THEIR VALUES (1,CAP-0,IND) :-'
      PRINT,
      DO 20 I=1,N
      READ,CIR(I),COMPVAL(I)
20    CONTINUE
22    PRINT,
      IF(CIR(N).EQ.1) GOTO 25
      FLAG=.TRUE.
      UNIT(N-1)=1
      CALL IND(COMPVAL(N),UNIT(N-1),CIR(N))
      N=N-1
25    NCOMP=N
      DO 50 I=N-1,1,-1
      UNITELEM=1
      DO 40 J=1,NCOMP-1
      IF(CIR(J).EQ.0) GOTO 30
      CALL CAP(COMPVAL(J),UNITELEM,CIR(J))
      GOTO 40
30    CALL IND(COMPVAL(J),UNITELEM,CIR(J))
40    CONTINUE
      NCOMP=NCOMP-1
      UNIT(I)=UNITELEM
50    CONTINUE
      PRINT,
      IF(FLAG) N=N+1
      RECT(1)=RS
      RECT(2)=COMPVAL(1)
      COUNT1=0
      DO 60 I=1,N-1
      COUNT2=COUNT1+I
      RECT(COUNT2+2)=UNIT(I)
      RECT(COUNT2+3)=COMPVAL(I+1)
      COUNT1=COUNT1+1
60    CONTINUE
      RECT(2*N+1)=RL
      COUNT1=0
      COUNT2=0
      R1=RECT(1)
      DO 70 I=1,N
      R2=RECT(I+1+COUNT1)
      R3=RECT(I+2+COUNT1)
      G1=1/R1
      G2=R2
      G3=1/R3
      G=2/(G1+G2+G3)
      ALPHA(I+COUNT2)=G1*G
      ALPHA(I+1+COUNT2)=G2*G
      ALPHA(I+2+COUNT2)=G3*G
      R1=RECT(I+2+COUNT1)
      COUNT1=COUNT1+1
      COUNT2=COUNT2+2
70    CONTINUE
      COUNT1=0
      PRINT,
      PRINT,
      PRINT,'THE WDF COEFFICIENTS ARE AS FOLLOWS :-'
      PRINT,
      DO 60 I=1,N
      PRINT,'BLOCK',I
      COUNT2=COUNT1+I
      PRINT,COUNT2,ALPHA(COUNT2)
      PRINT,COUNT2+1,ALPHA(COUNT2+1)
      PRINT,COUNT2+2,ALPHA(COUNT2+2)
      PRINT,
      COUNT1=COUNT1+2
 80   CONTINUE
      PRINT,
      PRINT,'TYPE <CR> TO CONTINUE !!!'
      READ,I
      PRINT,
      COUNT1=0
      COUNT2=0
      DO 90 I=1,N
      ALPHA(I+COUNT1)=ALPHA(I+COUNT2)
      ALPHA(I+COUNT1+1)=ALPHA(I+COUNT2+1)
      COUNT1=COUNT1+1
      COUNT2=COUNT2+2
 90   CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE CAP(COMPVAL,UNITELEM,ELEMENT)
      REAL COMPVAL,UNITELEM,DUMY
      INTEGER ELEMENT
      DUMY=1+UNITELEM*COMPVAL
      COMPVAL=(DUMY-1)*UNITELEM/DUMY
      UNITELEM=UNITELEM/DUMY
      ELEMENT=0
      RETURN
      END
C
C
C
      SUBROUTINE IND(COMPVAL,UNITELEM,ELEMENT)
      REAL COMPVAL,UNITELEM,DUMY
      INTEGER ELEMENT
      DUMY=1+(COMPVAL/UNITELEM)
      COMPVAL=(DUMY-1)/(DUMY*UNITELEM)
      UNITELEM=UNITELEM*DUMY
      ELEMENT=1
      RETURN
      END
C
C
C
      SUBROUTINE DFWLF(ALPHA,NSECT)
      REAL ALPHA(60),MIND,SD,MFACT,DP,RP,
     &DFP,DFS,FP,FS,E1,ML
      INTEGER NSBIT,NFE,N,NP,NS,NSECT
      LOGICAL EOP
      CHARACTER*1 CH
      COMMON /A/ DFP,DFS,NP,NS,DP
      COMMON/B/ NFE
      COMMON /B1/ FP,RP,FS,ML
```

```
      PRINT,
      PRINT,
      N=NSECT*2
      PRINT,
      PRINT,'          FINITE-WORD LENGTH DESIGN PROGRAM'
      PRINT,
      PRINT,
      PRINT,'DO YOU WISH TO ENTER THE SPECIFICATIONS?(Y/N)'
      READ,CH
      PRINT,
      IF(CH.EQ.'N') GOTO 10
      PRINT,
      PRINT,'PASS BAND  MIN RIPPLE   STOP BAND  MAX LOSS'
      PRINT,'EDGE FREQ  IN PASSBAND EDGE FREQ  IN STOPBAND'
      READ,FP,RP,FS,ML
 10   PRINT,
      PRINT,'      NO. OF POINTS'
      PRINT,'PASS BAND    STOP BAND'
      READ,NP,NS
 15   PRINT,
      PRINT,'ENTER THE STARTING BIT :-'
      READ,NSBIT
      PRINT,
      MFACT=0.5
      SD=(0.5)**NSBIT
      DFP=FP/(NP-1)
      DFS=(0.5-FS)/(NS-1)
      DP=RP/2
      CALL QUANT(ALPHA,N,NSBIT)
      CALL FUNCT(N,ALPHA,E1,EP,ES,EOP)
      PRINT,
      WRITE(6,100) NSBIT
      PRINT,
      DO 20 I=1,N
      WRITE(6,110) I,ALPHA(I)
 20   CONTINUE
      PRINT,
      PRINT,'MAX RIPPLE     = ',EP
      PRINT,'MIN LOSS       = ',ES
      PRINT,
      PRINT,'****************** DESIGN STAGE *******************'
      PRINT,
      NFE=0
      CALL PAT(N,SD,NSBIT,MFACT,ALPHA)
      PRINT,
      PRINT,
      PRINT,'TYPE <CR> TO CONTINUE !!!'
      READ,LL
      PRINT,
 100  FORMAT(' WDF COEFFICIENTS QUANTIZED TO',I4,' BITS ARE :-')
 110  FORMAT(' ALPHA(',I4,') =',F20.15)
      RETURN
      END
C
C
```

```
C
      SUBROUTINE QUANT(ALPHA,N,NBIT)
      REAL ALPHA(N),A
      INTEGER N,NBIT,I
      QL=(2**NBIT)/2
      DO 10 I=1,N
      A=ALPHA(I)
      ALPHA(I)=INT((A*QL)+0.5)/QL
 10   CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE PAT(N,SD,NSBIT,MFACT,PCOEFF)
      REAL PCOEFF(N),SAVE(20),ECOEFF(20),PFV,EP,ES,EFV,DLT,
     &MFACT,SD,MIND
      INTEGER NFE
      LOGICAL EOP
      COMMON /B/ NFE
      CALL FUNCT(N,PCOEFF,PFV,EP,ES,EOP)
      DLT=SD
      PRINT,'     NO. OF        NO. OF    MAX RIPPLE     MIN LOSS'
      PRINT,'FEFUNC CALLS       BITS     IN PASSBAND    IN STOPBAND'
 10   IF(EOP) GOTO 50
      CALL SETEQ(ECOEFF,PCOEFF,N)
      EFV=PFV
      CALL EXPLOR(ECOEFF,EFV,N,DLT,EOP)
      I=0
 20   IF (EFV.GE.PFV) GOTO 40
      I=I+1
      CALL FUNCT(N,ECOEFF,DFV,EP,ES,EOP)
      WRITE(6,100) NFE,NSBIT,EP,ES
      REWIND(3)
      DO 25 J=1,N
      WRITE(3,*) ECOEFF(J)
 25   CONTINUE
      IF(EOP) GOTO 50
      CALL SETEQ(SAVE,PCOEFF,N)
      CALL SETEQ(PCOEFF,ECOEFF,N)
      PFV=EFV
      DO 30 J=1,N
      ECOEFF(J)=2*ECOEFF(J)-SAVE(J)
 30   CONTINUE
      CALL FUNCT(N,ECOEFF,EFV,EP,ES,EOP)
      CALL EXPLOR(ECOEFF,EFV,N,DLT,EOP)
      CALL FUNCT(N,ECOEFF,DFV,EP,ES,EOP)
      GOTO 20
 40   IF (I.GT.0) GOTO 10
      DLT=MFACT*DLT
      NSBIT=NSBIT+1
      GOTO 10
 50   PRINT,
      IF(NFE.EQ.1) GOTO 60
      CALL SETEQ(PCOEFF,ECOEFF,N)
```

```
 60   PRINT,'FINAL VALUES :-'
      PRINT,
      PRINT,'NO. OF FEFUNC CALLS    =',NFE
      PRINT,'MAX RIPPLE IN PASS BAND =',EP
      PRINT,'MIN LOSS IN STOP BAND   =',ES
      PRINT,
      WRITE(6,110) NSBIT
      PRINT,
      DO 70 I=1,N
      WRITE(6,120) I,PCOEFF(I)
 70   CONTINUE
 100  FORMAT(3X,I6,9X,I2,4X,F11.9,4X,F11.9)
 110  FORMAT(' THE FINAL WDF COEFFICIENTS IN',I4,' BITS ARE :-')
 120  FORMAT(' ALPHA(',I4,') =',F20.15)
      RETURN
      END
C
C
C
      SUBROUTINE EXPLOR(ECOEFF,EFV,N,DLT,EOP)
      REAL ECOEFF(N),EFV,DLT,SAVE,FNEW,SAVEI,SAVEJ
      LOGICAL EOP
      I=1
 10   IF(I.GT.N) RETURN
      SAVE=ECOEFF(I)
      DI=ECOEFF(I)+DLT
      IF(DI.GE.2 .OR. DI.LE.0) GOTO 100
      ECOEFF(I)=DI
      CALL FUNCT(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 20
      EFV=FNEW
      GOTO 100
 20   DI=ECOEFF(I)-2*DLT
      IF(DI.GE.2 .OR. DI.LE.0) GOTO 100
      ECOEFF(I)=DI
      CALL FUNCT(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 30
      EFV=FNEW
      GOTO 100
 30   ECOEFF(I)=SAVE
      J=I+1
 40   IF(J.GT.N) GOTO 100
      SAVEI=ECOEFF(I)
      SAVEJ=ECOEFF(J)
      DI=ECOEFF(I)+DLT
      DJ=ECOEFF(J)+DLT
      IF(DI.GE.2 .OR. DJ.GE.2 .OR. DI.LE.0 .OR.
     &DJ.LE.0) GOTO 90
      ECOEFF(I)=DI
      ECOEFF(J)=DJ
      CALL FUNCT(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 50
      EFV=FNEW
      GOTO 90
 50   DI=ECOEFF(I)-2*DLT
```

```
      IF(DI.GE.2 .OR. DI.LE.0) GOTO 90
      ECOEFF(I)=DI
      CALL FUNCT(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 60
      EFV=FNEW
      GOTO 90
60    DJ=ECOEFF(J)-2*DLT
      IF(DJ.GE.2 .OR. DJ.LE.0) GOTO 90
      ECOEFF(J)=DJ
      CALL FUNCT(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 70
      EFV=FNEW
      GOTO 90
70    DI=ECOEFF(I)+2*DLT
      IF(DI.GE.2 .OR. DI.LE.0) GOTO 90
      ECOEFF(I)=DI
      CALL FUNCT(N,ECOEFF,FNEW,EP,ES,EOP)
      IF(FNEW.GE.EFV) GOTO 80
      EFV=FNEW
      GOTO 90
80    ECOEFF(I)=SAVEI
      ECOEFF(J)=SAVEJ
90    J=J+1
      GOTO 40
100   I=I+1
      GOTO 10
      END
C
C
C
      SUBROUTINE SETEQ(A,B,N)
      REAL A(N),B(N)
      INTEGER N,I
      DO 10 I=1,N
      A(I)=B(I)
10    CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE FUNCT(N,XC,FC,EP,ES,EOP)
      REAL XC(N),FC,FREQ,MR,ML,FS,DFP,DFS,MEP,MES
      INTEGER N,I,NP,NS
      LOGICAL EOP
      COMMON /A/ DFP,DFS,NP,NS,DP
      COMMON /B/ NFE
      COMMON /B1/ FP,RP,FS,ML
      MEP=0
      MES=0
      ES=10E5
      EOP=.FALSE.
      FREQ=0
      DO 30 I=1,NP
      CALL FRESP(XC,N,FREQ,MR)
      D=ABS(DP-MR)
      IF(D.LE.MEP) GOTO 20
      MEP=D
      EP=MR
20    FREQ=FREQ+DFP
30    CONTINUE
      FREQ=FS
      DO 70 I=1,NS
      CALL FRESP(XC,N,FREQ,MR)
      IF(MR.LT.ES) ES=MR
      D=ML-MR
50    IF(D.LE.MES) GOTO 60
      MES=D
60    FREQ=FREQ+DFS
70    CONTINUE
      IF(MEP.LE.DP .AND. ES.GT.ML) EOP=.TRUE.
      IF(MES.GT.MEP) GOTO 80
      FC=MEP
      GOTO 90
80    FC=MES
90    NFE=NFE+1
      RETURN
      END
C
C
C
      SUBROUTINE FRESP(XC,N,FREQ,MR)
      REAL XC(N),FREQ,MR,W,IM,RL,PI
      INTEGER K,N,COUNT1,COUNT2
      COMPLEX ABCD(2,2),DABCD(2,2),ZPM1,A1,B1,C1,D1,SC
      PI=3.14159263
      W=(2*PI*FREQ)
      ZPM1=CMPLX(COS(W),-SIN(W))
      CALL MATIDN(ABCD)
      COUNT1=0
      DO 30 K=1,N/2
      COUNT2=COUNT1+K
      SC=1/(XC(COUNT2)*(1+ZPM1))
      A1=(1+(1-XC(COUNT2+1))*ZPM1)*SC
      B1=(XC(COUNT2)+XC(COUNT2+1)-1-(1-XC(COUNT2))*ZPM1)*SC
      C1=(XC(COUNT2)-1+(XC(COUNT2)+XC(COUNT2+1)-1)*ZPM1)*SC
      D1=(1-XC(COUNT2+1)+ZPM1)*SC
      DABCD(1,1)=A1
      DABCD(2,1)=C1
      IF(K.EQ.N/2) GOTO 10
      DABCD(1,2)=B1*ZPM1
      DABCD(2,2)=D1*ZPM1
      GOTO 20
10    DABCD(1,2)=B1
      DABCD(2,2)=D1
20    CALL MATMUL(ABCD,DABCD)
      COUNT1=COUNT1+1
30    CONTINUE
      RL=REAL(ABCD(1,1))
      IM=AIMAG(ABCD(1,1))
      MR=1/SQRT((RL**2)+(IM**2))
      MR=-20*ALOG10(MR)
      RETURN
      END
C
C
C
      SUBROUTINE MATIDN(ABCD)
      COMPLEX ABCD(2,2)
      ABCD(1,1)=(1,0)
      ABCD(2,2)=(1,0)
      ABCD(1,2)=(0,0)
      ABCD(2,1)=(0,0)
      RETURN
      END
C
C
C
      SUBROUTINE MATMUL(ABCD,DABCD)
      COMPLEX ABCD(2,2),DABCD(2,2),C(2,2)
      C(1,1)=ABCD(1,1)*DABCD(1,1)+ABCD(1,2)*DABCD(2,1)
      C(1,2)=ABCD(1,1)*DABCD(1,2)+ABCD(1,2)*DABCD(2,2)
      C(2,1)=ABCD(2,1)*DABCD(1,1)+ABCD(2,2)*DABCD(2,1)
      C(2,2)=ABCD(2,1)*DABCD(1,2)+ABCD(2,2)*DABCD(2,2)
      ABCD(1,1)=C(1,1)
      ABCD(1,2)=C(1,2)
      ABCD(2,1)=C(2,1)
      ABCD(2,2)=C(2,2)
      RETURN
      END
C
C
C
      SUBROUTINE SAVE(ALPHA,N)
      REAL ALPHA(60)
      INTEGER N,I,J
      PRINT,
      PRINT,'THE CURRENT COEFFICIENTS CAN BE STORE IN.'
      PRINT,
      PRINT,'      1) LCCOEF.I (INFINITE PRECISION),OR'
      PRINT,'      2) LCCOEF.F (FINITE PRECISION),  OR'
      PRINT,'      3) LCCOEF.T (TEMPORARY FILE).'
      PRINT,
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ,J
      PRINT,
      REWIND(J)
      DO 10 I=1,N*2
      WRITE(J,*) ALPHA(I)
   10 CONTINUE
      RETURN
      END
```

B3.0- Analysis Program

This program has been developed as a tool for the analyses of the WDFs designed using the design prorams in the previous sections. It covers the analysis of the three different WDFs. The program is menu deriven and very easy to use. The WDF coefficients can either be entered at the terminal or they can be read from the files used and created by the design programs. The outputs are presented graphically and it is possible to plot the responses of the filter for 3 different cases, i.e the ideal filter (i.e using the synthesis coefficients), with quantized coefficients and with FWLD coefficients (i.e coefficients from the Finite WordLength Design program). The plots can be obtained for each individual response or the responses can be plotted on the same axis.

```
C----------------------------------------------------------------
C
C PROGRAM TO ANALYSE WAVE DIGITAL FILTERS (WDFS)
C BASED ON UNIT ELEMENT, LATTICE AND LC-LADDER
C REFERENCE FILTERS.(FORTRAN 66)
C
C AUTHOR : A. R. MIRZAI          DATE: APRIL 1986
C
C       THE CENTRE FOR INFORMATION ENG
C            THE CITY UNIVERSITY
C       NORTHAMPTON SQ, LONDON EC1VOHB
C
C----------------------------------------------------------------
      INTEGER I
C
C     MAIN PRORAM
C
 5    PRINT,
      PRINT,
      PRINT,'               PROGRAM TO ANALYSE WDFS'
      PRINT,
      PRINT,'                       MAIN MENU'
      PRINT,
      PRINT,'            1) UNIT ELEMENT WDFS.'
      PRINT,'            2) LATTICE WDF.'
      PRINT,'            3) LC-LADDER WDFS.'
      PRINT,'            4) END ANALYSIS PROGRAM.'
      PRINT,
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ,I
      GOTO(10,20,30,40),I
      I=0
      GOTO 5
 10   CALL ANAUE
      GOTO 5
 20   CALL ANALT
      GOTO 5
 30   CALL ANALC
      GOTO 5
 40   STOP
      END
C----------------------------------------------------------------
C
C PROGRAM TO ANALYSE  WAVE DIGITAL FILTERS BASED
C ON THE CASCADE OF UNIT ELEMENT FILTERS.
C
C DATE : JULY 1984
C
C----------------------------------------------------------------
      SUBROUTINE ANAUE
      REAL ALPHA(20),QALPHA(20),FALPHA(20),FREQ(300)
      INTEGER N,CHOICE
 5    PRINT,
      PRINT,
      PRINT,'  PROGRAM TO ANALYS WDF BASED ON THE UNIT'
```

```
      PRINT,'       ELEMENT CASCADED REFRENCE FILTERS'
      PRINT,
      PRINT,'                     MENU'
      PRINT,
      PRINT,
      PRINT,' 1) READ IMPEDENCES.'
      PRINT,' 2) READ COEFFICIENTS.'
      PRINT,' 3) QUANTIZE THE COEFFICIENTS.'
      PRINT,' 4) ANALYSE IDEAL FILTER.'
      PRINT,' 5) ANALYSE FILTER WITH QUANTIZE COEFF.'
      PRINT,' 6) ANALYSE FILTER WITH FINITE)WORD LENGTH COEFF.'
      PRINT,' 7) PLOT RESPONSES ON THE SAME AXIS.'
      PRINT,' 8) RETURN TO MAIN MENU.'
      PRINT,
      PRINT,
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ,CHOICE
      PRINT,
      GOTO (10,20,30,40,50,60,70,80),CHOICE
      CHOICE=0
      GOTO 5
 10   CALL RIFWDF(ALPHA,N)
      GOTO 5
 20   CALL RCOEF1(ALPHA,FALPHA,N)
      GOTO 5
 30   CALL QUANT(ALPHA,QALPHA,N,1)
      GOTO 5
 40   CALL ANAS1(ALPHA,N,1)
      GOTO 5
 50   CALL ANAS1(QALPHA,N,1)
      GOTO 5
 60   CALL ANAS1(FALPHA,N,1)
      GOTO 5
 70   CALL PROSA1(ALPHA,QALPHA,FALPHA,N)
      GOTO 5
 80   RETURN
      END
C
C SUBROUTINE MATMUL : MULTIPLIES TWO COMPLEX MATRIX.
C
      SUBROUTINE MATMUL(A,B)
      COMPLEX A(2,2),B(2,2),C(2,2)
      C(1,1)=A(1,1)*B(1,1)+A(1,2)*B(2,1)
      C(1,2)=A(1,1)*B(1,2)+A(1,2)*B(2,2)
      C(2,1)=A(2,1)*B(1,1)+A(2,2)*B(2,1)
      C(2,2)=A(2,1)*B(1,2)+A(2,2)*B(2,2)
      A(1,1)=C(1,1)
      A(1,2)=C(1,2)
      A(2,1)=C(2,1)
      A(2,2)=C(2,2)
      RETURN
      END
C
C SUBROUTINE MATIDN : MAKES A COMPLEX MATRIX EQUAL TO
C IDENTITY MATRIX.
```

```
C
      SUBROUTINE MATIDN(A)
      COMPLEX A(2,2)
      INTEGER I,J
      A(1,1)=(1,0)
      A(2,2)=(1,0)
      A(1,2)=(0,0)
      A(2,1)=(0,0)
      RETURN
      END
C
C SUBROUTINE RIFWDF : THIS SUBROUTINE READS THE VALUES OF THE
C STEP IMPEDENCES AND CALCULATES THE COEFFICIENTS OF THE
C WDF STRUCTURE.
C
      SUBROUTINE RIFWDF(A,DEG)
      REAL IMP(51),A(50),S
      INTEGER I,M,DEG,J,NBIT
      CHARACTER*1 CH
      PRINT,
      PRINT,
      PRINT,
      PRINT,
      PRINT,
      PRINT,"THIS PROGRAM ASSUMES THAT THE STEP IMPEDENCES"
      PRINT,"FOR THE REFERENCE FILTER ARE OBTAINED USING"
      PRINT,"THE TRASMISSION LINE FILTER PROGRAM."
      PRINT,
      PRINT,
      PRINT,"ENTER FILTER DEGREE:-"
      READ,DEG
      M=INT(DEG/2)
      PRINT,
      PRINT,"ENTER THE STEP IMPEDENCES. NOTE, Z(1) IS PUT"
      PRINT,"EQUAL TO 1 BY THE PROGRAM."
      PRINT,
      IMP(1)=1.0
      IMP(DEG+2)=1.0
      PRINT,
      PRINT,
      DO 5 I=1,M
      WRITE(6,65) I+1
      READ,IMP(I+1)
      PRINT,
 5    CONTINUE
      IF (DEG.EQ.(2*M)) GOTO 20
      WRITE(6,65) M+2
      READ,IMP(M+2)
      DO 10 I=1,M
      IMP(DEG+2-I)=IMP(I+1)
 10   CONTINUE
      M=(DEG+3)/2
      GOTO 40
 20   PRINT,"ENTER THE PASS-BAND RIPPLE(DB) :-"
      READ,S
      PRINT,
      S=2*S-1+S*SQRT(S**2-S)
      DO 30 I=1,M
      IMP(DEG+2-I)=S/IMP(I+1)
 30   CONTINUE
      M=(DEG+2)/2
 40   PRINT,
      PRINT,'THE IMPEDENCES ARE AS FOLLOWS :-'
      PRINT,
      DO 45 I=1,M
      J=DEG+3-I
      WRITE(6,70) I,IMP(I),J
 45   CONTINUE
      PRINT,
      PRINT,'ARE THESE CORRECT?(Y/N)'
      READ,CH
      PRINT,
      IF(CH.EQ.'Y') GOTO 50
      PRINT,
      PRINT,'WHICH IMPEDENCE IS NOT CORRECT?(ONE AT A TIME)'
      READ,I
      PRINT,
      PRINT,'ENTER THE CORRECT VALUE :-'
      READ,IMP(I)
      IMP(DEG+3-I)=IMP(I)
      PRINT,
      GOTO 40
 50   PRINT,
      PRINT,
      PRINT,"THE COEFFICIENTS OF THE WDF STRUCTURE"
      PRINT,"ARE AS FOLLOWS :-"
      PRINT,
      DO 60 I=1,DEG+1
      A(I)=(IMP(I)-IMP(I+1))/(IMP(I)+IMP(I+1))
      WRITE(6,80) I,A(I)
 60   CONTINUE
      PRINT,
      PRINT,
      PRINT,'TYPE "1" TO CONTINUE!!'
      READ,I
 65   FORMAT(' Z(',I4,')  ')
 70   FORMAT(' Z(',I4,')= ',F20.15,5X,' =Z(',I4,') ')
 80   FORMAT(' ALPHA(',I4,')= ',F20.15)
 90   RETURN
      END
C
C SUBROUTINE ANAS1 : THIS SUBROUTINE FINDS THE FREQUENCY
C RESPONSE OF A WDF BASED ON UNIT ELEMENT FILTERS.
C
      SUBROUTINE ANAS1(ALPHA,N,TYPE)
      REAL LF,UF,SF,W,DF,SC,AS,PI,C(300),D(300),
     &B(300),ALPHB(300),RL,IM,ALPHA(20)
      INTEGER N,NP,I,K,CHOICE,TYPE
      CHARACTER*1 CH
      COMPLEX DABCD(2,2),ABCD(2,2),ZPMN
      COMMON /ANA1/ B,C,NP,LF,UF,SF
      PI=3.14159263
      IF (TYPE.EQ.0) GOTO 20
      PRINT,
      PRINT,'**************** ANALYSIS STAGE ***************'
      PRINT,
 10   PRINT,
      PRINT,
      PRINT,
      PRINT,'LOWER FREQ  UPPER FREQ NO. OF  SAMPLING '
      PRINT,'   (HZ)        (HZ)    POINTS  FREQ(HZ) '
      READ,LF,UF,NP,SF
 20   DF=(UF-LF)/(NP-1)
      B(1)=LF
      DO 180 I=1,NP
      W=(2*PI*B(I))/SF
      ZPMN=CMPLX(COS(W),-SIN(W))
      CALL MATIDN(ABCD)
      DO 160 K=1,N+1
      SC=1/(1-ALPHA(K))
      IF (K.EQ.N+1) GOTO 140
      DABCD(1,1)=CMPLX(SC,0)
      DABCD(1,2)=-ALPHA(K)*ZPMN*SC
      DABCD(2,1)=CMPLX(-ALPHA(K)*SC,0)
      DABCD(2,2)=ZPMN*SC
      GOTO 150
 140  DABCD(1,1)=CMPLX(SC,0)
      DABCD(1,2)=CMPLX(-ALPHA(N-1)*SC,0)
      DABCD(2,1)=CMPLX(-ALPHA(N+1)*SC,0)
      DABCD(2,2)=CMPLX(SC,0)
 150  CALL MATMUL(ABCD,DABCD)
 160  CONTINUE
      RL=REAL(ABCD(1,1))
      IM=AIMAG(ABCD(1,1))
      C(I)=1/((RL**2)+(IM**2))
      C(I)=-20*ALOG10(SQRT(C(I)))
 170  D(I)=-ATAN2(IM,RL)*180/PI
      B(I+1)=B(I)+DF
 180  CONTINUE
      IF (TYPE.EQ.0) GOTO 220
      PRINT,
      PRINT,'DO YOU WISH TO HAVE LISTING OF THE RESULTS?(Y/N)'
      READ,CH
      IF(CH.EQ.'N') GOTO 200
      PRINT,
      PRINT,'          FREQ(HZ)       LOSS(DB)'
      PRINT,
      DO 190 I=1,NP
      PRINT,I,B(I),C(I)
 190  CONTINUE
      PRINT,
 200  PRINT,
      PRINT,'DO YOU WISH TO HAVE A PLOT?(Y/N)'
      READ,CH
      PRINT,
      IF(CH.EQ.'N') GOTO 210
      CALL PLOT(B,C,NP)
 210  PRINT,'DO YOU WISH TO PLOT AGAIN?(Y/N)'
      READ,CH
      PRINT,
      IF(CH.EQ.'Y') GOTO 10
 220  RETURN
      END
C
C SUBROUTINE PLOT : THIS SUBROUTINE PLOTS CURVES USING THE GINO
C ROUTINES AND IMLAC/TEK AS THE OUTPUT DEVICE.
C
      SUBROUTINE PLOT(A,B,NP)
      REAL A(300),B(300)
      INTEGER NP,SYM
      CHARACTER*1 CH
      PRINT,"ARE YOU USING IMLACE OR TEK?(I/T)"
      READ,CH
      CALL PICCLE
      IF(CH.EQ.'I' .OR. CH.EQ.'I') GOTO 10
      CALL T4010
      GOTO 20
 10   CALL APDS4
 20   CALL PICCLE
      CALL GRAF(A,B,NP)
      CALL DEVEND
      READ,CH
      RETURN
      END
C
C
C
      SUBROUTINE DRAAXI
      REAL LF,UF,LL,UL
      CHARACTER*1 CHAR
      PRINT,
      PRINT,
      PRINT,'LOWER   UPPER   LOWER   UPPER'
      PRINT,'FREQ    FREQ    LOSS    LOSS'
      READ,LF,UF,LL,UL
      PRINT,
      PRINT,'ARE YOU USING TEK OR IML(I/T)?'
      READ,CHAR
      PRINT,
      IF(CHAR.EQ.'T') GOTO 10
      CALL APDS4
      GOTO 20
 10   CALL T4010
 20   CALL PICCLE
      CALL AXISCA(3,5,LF,UF,1)
      CALL AXISCA(3,NYI,LL,UL,2)
      CALL AXIDRA(2,1,1)
      CALL AXIDRA(-2,-1,2)
      RETURN
      END
C
C
C
      SUBROUTINE PROSA1(ALPHA,QALPHA,FALPHA,N)
      REAL ALPHA(20),QALPHA(20),FALPHA(20),FREQ(300),DARRAY(300)
     &,LF,UF,SF,ARRAY1(300),ARRAY2(300),ARRAY3(300)
      INTEGER NPT
      COMMON /ANA1/ FREQ,DARRAY,NPT,LF,UF,SF
      PRINT,'LOWER FREQ   UPPER FREQ   NO. OF   SAMPLING'
      PRINT,'   (HZ)         (HZ)      POINTS   FREQ(HZ)'
      READ,LF,UF,NPT,SF
      PRINT,
      CALL ANAS1(ALPHA,N,0)
      DO  10 I=1,NPT
      ARRAY1(I)=DARRAY(I)
 10   CONTINUE
      CALL ANAS1(QALPHA,N,0)
      DO 20 I=1,NPT
      ARRAY2(I)=DARRAY(I)
 20   CONTINUE
      CALL ANAS1(FALPHA,N,0)
      DO 30 I=1,NPT
      ARRAY3(I)=DARRAY(I)
 30   CONTINUE
      CALL PLTAXS(FREQ,ARRAY1,ARRAY2,ARRAY3,NPT)
      RETURN
      END
C
C
C
      SUBROUTINE PLTAXS(FREQ,ARRAY1,ARRAY2,ARRAY3,NPT)
      REAL FREQ(300),ARRAY1(300),ARRAY2(300),ARRAY3(300)
      INTEGER NPT
      CALL DRAAXI
      CALL GRAPOL(FREQ,ARRAY1,NPT)
      CALL GRASYM(FREQ,ARRAY1,NPT,3,0)
      CALL GRAPOL(FREQ,ARRAY2,NPT)
      CALL GRASYM(FREQ,ARRAY2,NPT,4,0)
      CALL GRAPOL(FREQ,ARRAY3,NPT)
      CALL GRASYM(FREQ,ARRAY3,NPT,7,0)
      CALL DEVEND
      READ,I
      RETURN
      END
C
C
C SUBROUTINE RCOEF1 : THIS SUBROUTINE READS THE WDF COEFFICIENTS
C FOR THE UNIT ELEMENT FILTERS.
C
      SUBROUTINE RCOEF1(ALPHA,FALPHA,N)
      REAL ALPHA(20),FALPHA(20)
      INTEGER N
      PRINT,
      PRINT,'ENTER THE FILTER ORDER :-'
```

```
      READ,N
 5    PRINT,
      PRINT,'YOU CAN READ THE COEFFICIENTS FROM,'
      PRINT,
      PRINT,'     1) UECOEF.I (INFINITE-PRECSION),OR'
      PRINT,'     2) UECOEF.F (FINITE-PRECSION) , OR'
      PRINT,'     3) READ COEFFICIENTS FROM THE TERMINAL.'
      PRINT,'     4) RETURN TO PREVIOUS MENU.'
      PRINT,
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ,I
      GOTO(10,20,30,40),I
      I=0
      GOTO 5
 10   REWIND(1)
      READ(1,100) (ALPHA(I),I=1,N+1)
      GOTO 5
 20   REWIND(2)
      READ(2,100) (FALPHA(I),I=1,N+1)
      GOTO 5
 30   READ,(ALPHA(I),I=1,N+1)
      GOTO 5
 40   RETURN
 100  FORMAT(V)
      END
C-------------------------------------------------------------------
C
C PROGRAM TO ANALYSE WAVE DIGITAL FILTERS BASED
C ON LATTICE REFERENCE FILTERS.
C
C DATE : OCT 1985
C
C-------------------------------------------------------------------
      SUBROUTINE ANALT
      REAL ALPHA(20),QALPHA(20),FALPHA(20)
      INTEGER N,CHOIC
      PRINT,
      PRINT,
 5    PRINT,'          PROGRAM TO ANALYSE WDFS BASE'
      PRINT,'          ON LATTICE REFERENCE FILTERS'
      PRINT,
      PRINT,'                        MENU'
      PRINT,
      PRINT,'          1) READ COEFFICIENTS.'
      PRINT,'          2) QUANTIZE THE COEFFICIENTS.'
      PRINT,'          3) ANALYSE IDEAL FILTER.'
      PRINT,'          4) ANALYSE FILTER WITH QUANTIZED  COEFF.'
      PRINT,'          5) ANALYSE FILTER WITH FINITE-WORD LENGTH COEFF.'
      PRINT,'          6) PLOT RESPONSES ON THE SAME AXIS.'
      PRINT,'          7) RETURN TO MAIN MENU.'
      PRINT,
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ,CHOIC
      PRINT,
      GOTO (10,20,30,40,50,60,70),CHOIC
      GOTO 5
 10   CALL RCOEF2(ALPHA,FALPHA,N)
      GOTO 5
 20   CALL QUANT(ALPHA,QALPHA,N,0)
      GOTO 5
 30   CALL ANAS2(ALPHA,N,1)
      GOTO 5
 40   CALL ANAS2(QALPHA,N,1)
      GOTO 5
 50   CALL ANAS2(FALPHA,N,1)
      GOTO 5
 60   CALL PROSA2(ALPHA,QALPHA,FALPHA,N)
      GOTO 5
 70   RETURN
      END
C
C SUBROUTINE RCOEF2 : THIS SUBROUTINE READS THE WDF COEFFICIENTS
C BASED ON LATTICE FILTERS.
C
      SUBROUTINE RCOEF2(ALPHA,FALPHA,N)
      REAL ALPHA(20),FALPHA(20)
      INTEGER N,I,LTCOEFF,LTCOEFI
      PRINT,
      PRINT,'ENTER FILTER ORDER :-'
      READ,N
 5    PRINT,
      PRINT,'YOU CAN READ THE COEFFICIENTS FROM,'
      PRINT,
      PRINT,'   1) LTCOEF.I (INFINITE PRECISION),OR'
      PRINT,'   2) LTCOEF.F ( FINITE PRECISION), OR'
      PRINT,'   3) ENETR THE COEFFICIENTS  FROM THE  TERMINAL.'
      PRINT,'   4) RETURN TO PREVIOUS MENU.'
      PRINT,
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ,I
      GOTO (10,20,30,40),I
      I=0
      GOTO 5
 10   REWIND(3)
      READ(3,100) (ALPHA(I),I=1,N)
      GOTO 5
 20   REWIND(4)
      READ(4,100) (FALPHA(I),I=1,N)
      GOTO  5
 30   PRINT,
      PRINT,'ENTER THE COEFFICIENTS NOW :-'
      READ,(ALPHA(I),I=1,N)
      GOTO 5
 40   RETURN
 100  FORMAT(V)
      END
C
C SUBROUTINE QUANT : THIS SUBROUTINE QUANTIZES A SET OF COEFFICIENTS
C TO REQUIRED NUMBER OF BITS.
C
```

```
      SUBROUTINE QUANT(ALPHA,QALPHA,N,TYPE)
      REAL ALPHA(50),QALPHA(50),SD,A
      INTEGER N,I,NBIT,TYPE
      PRINT,
      PRINT,'ENTER REQUIRED NUMBER OF BITS :-'
      READ,NBIT
      PRINT,
      SD=0.5**NBIT
      PRINT,'THE WDF FILTER COEFFICIENTS ARE :-'
      PRINT,
      PRINT,'      ACTUAL          QUANTIZED'
      PRINT,
      DO 10 I=1,N+TYPE
      A=ALPHA(I)
      QALPHA(I)=SIGN(SD*FLOAT(INT(ABS(A)/SD+0.5)),A)
      PRINT,I,ALPHA(I),QALPHA(I)
 10   CONTINUE
      PRINT,
      PRINT,
      PRINT,'TYPE <CR> TO CONTINUE !!!'
      READ,LLL
      RETURN
      END
C
C SUBROUTINE ANAS2 : THIS SUBROUTINE FINDS THE FREQUENCY
C RESPONSE OF A WDF BASED ON LC-LADDER FILTERS.
C
      SUBROUTINE ANAS2(ALPHA,N,TYPE)
      REAL ALPHA(20),RESP(300),FREQ(300)
     &,LF,UF,SF
      INTEGER IC1,IC2,MAG,N,I,NFP,CHOIC,TYPE
      CHARACTER*1 CH
      COMPLEX ZPM1,ZPM2,G1Z,G2Z,GZ
      COMMON /ANA2/ FREQ,RESP,NFP,LF,UF,SF
      IF (TYPE.EQ.0) GOTO 20
      PRINT,
      PRINT,'**************** ANALYSIS STAGE ***************'
      PRINT,
 10   PRINT,
      PRINT,
      PRINT,'LOWER FREQ   UPPER FREQ   NO. OF   SAMPLING'
      PRINT,'    (HZ)         (HZ)      POINTS   FREQ(HZ)'
      READ,LF,UF,NFP,SF
      PRINT,
 20   PI=3.14159263
      DF=(UF-LF)/(NFP-1)
      FREQ(1)=LF
      G1Z=(1,0)
      G2Z=(1,0)
      DO 100 I=1,NFP
      W=(2*PI*FREQ(I))/SF
      ZPM1=CMPLX(COS(W),-SIN(W))
      ZPM2=ZPM1**2
      G1Z=(ZPM1-ALPHA(1))/(1-ALPHA(1)*ZPM1)
      G2Z=(1,0)
      IC1=1
      IC2=2
 80   CALL TRANSF(ALPHA(IC2),ALPHA(IC2+1),G2Z,ZPM1,ZPM2)
      IC1=IC1+2
      IF(IC1.EQ.N) GOTO 90
      CALL TRANSF(ALPHA(IC2+2),ALPHA(IC2+3),G1Z,ZPM1,ZPM2)
      IC1=IC1+2
      IC2=IC2+4
      IF(IC1.NE.N) GOTO 80
 90   GZ=(G1Z+G2Z)/2
      RESP(I)=-20*ALOG10(CABS(GZ))
      IF(MAG.EQ.1) RESP(I)=CABS(GZ)
      FREQ(I+1)=FREQ(I)+DF
 100  CONTINUE
      IF (TYPE.EQ.0) GOTO 140
      PRINT,
      PRINT,'DO YOU WISH TO HAVE LISTING OF THE RESULT?(Y/N)'
      READ,CH
      PRINT,
      IF(CH.EQ.'N') GOTO 120
      PRINT,
      PRINT,'          FREQ         LOSS'
      PRINT,
      DO 110 I=1,NFP
      PRINT,I,FREQ(I),RESP(I)
 110  CONTINUE
      PRINT,
 120  PRINT,'DO YOU WISH TO HAVE A PLOT?(Y/N)'
      READ,CH
      PRINT,
      IF(CH.EQ.'N') GOTO 130
      CALL PLOT(FREQ,RESP,NFP)
 130  PRINT,'DO YOU WISH TO PLOT AGAIN?(Y/N)'
      READ,CH
      PRINT,
      IF(CH.EQ.'Y') GOTO 10
 140  RETURN
      END
C
C SUBROUTINE TRANSF : THIS SUBROUTINE FINDS THE TRANSFER
C FUNCTION OF TWO 2-PORTS IN CASCADE.
C
      SUBROUTINE TRANSF(A1,A2,G,Z1,Z2)
      REAL A1,A2
      COMPLEX G,Z1,Z2,G1
      G1=Z2+A2*(A1-1)*Z1-A1
      G1=G1/(1+A2*(A1-1)*Z1-A1*Z2)
      G=G*G1
      RETURN
      END
C
C
C
      SUBROUTINE PROSA2(ALPHA,QALPHA,FALPHA,N)
      REAL ALPHA(20),QALPHA(20),FALPHA(20),FREQ(300),DARRAY(300)
     &,LF,UF,SF,ARRAY1(300),ARRAY2(300),ARRAY3(300)
      INTEGER NPT
      COMMON /ANA2/ FREQ,DARRAY,NPT,LF,UF,SF
      PRINT,'LOWER FREQ   UPPER FREQ   NO. OF   SAMPLING'
      PRINT,'   (HZ)         (HZ)      POINTS   FREQ(HZ)'
      READ,LF,UF,NPT,SF
      PRINT,
      CALL ANAS2(ALPHA,N,0)
      DO 10 I=1,NPT
      ARRAY1(I)=DARRAY(I)
 10   CONTINUE
      CALL ANAS2(QALPHA,N,0)
      DO 20 I=1,NPT
      ARRAY2(I)=DARRAY(I)
 20   CONTINUE
      CALL ANAS2(FALPHA,N,0)
      DO 30 I=1,NPT
      ARRAY3(I)=DARRAY(I)
 30   CONTINUE
      CALL PLTAXS(FREQ,ARRAY1,ARRAY2,ARRAY3,NPT)
      RETURN
      END
C-------------------------------------------------------------
C
C PROGRAM TO ANALYSE WAVE DIGITAL FILTERS BASED
C ON THE LC-LADDER REFERENCE FILTERS.
C
C DATE : AUG  1985
C
C-------------------------------------------------------------
      SUBROUTINE ANALC
      REAL ALPHA(40),QALPHA(40),FALPHA(40) .
      INTEGER NBLOCK,I
      PRINT,
 5    PRINT,
      PRINT,
      PRINT,'         PROGRAM TO ANALYSE WDFS BASED ON THE'
      PRINT,'            LC-LADDER REFERENCE FILTERS'
      PRINT,
      PRINT,'                       MENU'
      PRINT,
      PRINT,'   1) READ WDF COEFFICIENTS.'
      PRINT,'   2) QUANTIZE THE FILTER COEFFICIENTS.'
      PRINT,'   3) ANALYSE IDEAL FILTER.'
      PRINT,'   4) ANALYSE FILTER WITH QUANTIZE COEFF.'
      PRINT,'   5) ANALYSE FILTER WITH FINITE-WORD LENGTH COEFF'
      PRINT,'   6) PLOT RESPONSES ON THE SAME AXIS.'
      PRINT,'   7) RETURN TO MAIN MENU.'
      PRINT,
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ,I
      PRINT,
      GOTO(10,20,30,40,50,60,70),I
      I=0
      GOTO 5
 10   CALL RCOEF3(ALPHA,FALPHA,NBLOCK)
      GOTO 5
 20   N=NBLOCK*2
      CALL QUANT(ALPHA,QALPHA,N,0)
      GOTO 5
 30   CALL ANAS3(ALPHA,NBLOCK,1)
      GOTO 5
 40   CALL ANAS3(QALPHA,NBLOCK,1)
      GOTO 5
 50   CALL ANAS3(FALPHA,NBLOCK,1)
      GOTO 5
 60   CALL PROSA3(ALPHA,QALPHA,FALPHA,NBLOCK)
      GOTO 5
 70   RETURN
      END
C
C SUBROUTINE RCOEF3 : THIS SUBROUTINE READ THE WDF COEFFICIENTS
C BASED ON LC-LADDER FILTERS.
C
      SUBROUTINE RCOEF3(ALPHA,FALPHA,N)
      REAL ALPHA(40),FALPHA(40)
      INTEGER I,N
      PRINT,
      PRINT,'ENTER FILTER ORDER :-'
      READ,N
 5    PRINT,
      PRINT,
      PRINT,'YOU CAN READ THE COEFFICIENTS FROM,'
      PRINT,
      PRINT,'    1) LCCOEF.I (INFINITE PRECISION),OR'
      PRINT,'    2) LCCOEF.F (FINITE PRECISION), OR'
      PRINT,'    3) RETURN TO MAIN MENU.'
      PRINT,
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ,I
      PRINT,
      GOTO(10,20,30),I
      I=0
      GOTO 5
 10   REWIND(7)
      READ(7,100) (ALPHA(I),I=1,2*N)
      GOTO 5
 20   REWIND(8)
      READ(8,100) (FALPHA(I),I=1,2*N)
      GOTO 5
 30   RETURN
 100  FORMAT(V)
      RETURN
      END
C
C SUBROUTINE ANAS3 : THIS SUBROUTINE FINDS THE FREQUENCY
C RESPONSE OF A WDF BASED ON LC-LADDER FILTERS.
C
      SUBROUTINE ANAS3(A,NBLOCK,TYPE)
      REAL LF,UF,SF,W,DF,AS,PI,C(300),D(300),
```

```
     &A(40),B(300),RL,IM
      INTEGER NBLOCK,NP,I,K,TYPE
      CHARACTER*1 CH
      COMPLEX DABCD(2,2),ABCD(2,2),ZPM1,A1,B1,C1,D1,SC
      COMMON /ANA3/ B,C,NP,LF,UF,SF
      IF (TYPE.EQ.0) GOTO 7
      PRINT,
      PRINT,'**************** ANALYSIS STAGE ***************'
      PRINT,
5     PRINT,
      PRINT,
      PRINT,
      PRINT,"LOWER FREQ   UPPER FREQ   NO. OF   SAMPLING"
      PRINT,"    (HZ)        (HZ)       POINTS   FREQ(HZ)"
      READ,LF,UF,NP,SF
7     DF=(UF-LF)/(NP-1)
      B(1)=LF
      PI=3.14159263
      DO 40 I=1,NP
      W=(2*PI*B(I))/SF
      ZPM1=CMPLX(COS(W),-SIN(W))
      CALL MATIDN(ABCD)
      COUNT1=0
      DO 30 K=1,NBLOCK
      COUNT2=COUNT1+K
      SC=1/(A(COUNT2)*(1+ZPM1))
      A1=(1+(1-A(COUNT2+1))*ZPM1)*SC
      B1=(A(COUNT2)+A(COUNT2+1)-1-(1-A(COUNT2))*ZPM1)*SC
      C1=(A(COUNT2)-1+(A(COUNT2)+A(COUNT2+1)-1)*ZPM1)*SC
      D1=(1-A(COUNT2+1)+ZPM1)*SC
      DABCD(1,1)=A1
      DABCD(2,1)=C1
      IF (K.EQ.NBLOCK) GOTO 10
      DABCD(1,2)=B1*ZPM1
      DABCD(2,2)=D1*ZPM1
      GOTO 20
10    DABCD(1,2)=B1
      DABCD(2,2)=D1
20    CALL MATMULT(ABCD,DABCD)
      COUNT1=COUNT1+1
30    CONTINUE
      RL=REAL(ABCD(1,1))
      IM=AIMAG(ABCD(1,1))
      C(I)=1/((RL**2)+(IM**2))
      C(I)=-20*ALOG10(SQRT(C(I)))
      B(I+1)=B(I)+DF
40    CONTINUE
      IF (TYPE.EQ.0) GOTO 80
      PRINT,
      PRINT,'DO YOU WISH TO HAVE A LIST OF THE RESULTS?(Y/N)'
      READ,CH
      PRINT,
      IF(CH.EQ.'N' .OR. CH.EQ.'N') GOTO 60
      PRINT,'                     FREQ         LOSS'
      DO 50 I=1,NP
      PRINT,I,B(I),C(I)
 50   CONTINUE
 60   PRINT,
      PRINT,'DO YOU WISH TO HAVE A PLOT OF THE RESULTS?(Y/N)'
      READ,CH
      IF(CH.EQ.'N' .OR. CH.EQ.'N') GOTO 70
      CALL PLOT(B,C,NP)
 70   PRINT,
      PRINT,"DO YOU WISH TO PLOT AGAIN (Y/N)?"
      READ,CH
      IF (CH.EQ.'Y' .OR. CH.EQ.'Y') GOTO 5
 80   RETURN
      END
C
C
C
      SUBROUTINE PROSA3(ALPHA,QALPHA,FALPHA,N)
      REAL ALPHA(40),QALPHA(40),FALPHA(40),FREQ(300),DARRAY(300)
     &,LF,UF,SF,ARRAY1(300),ARRAY2(300),ARRAY3(300)
      INTEGER NPT
      COMMON /ANA3/ FREQ,DARRAY,NPT,LF,UF,SF
      PRINT,'LOWER FREQ   UPPER FREQ   NO. OF   SAMPLING'
      PRINT,'    (HZ)          (HZ)     POINTS   FREQ(HZ)'
      READ,LF,UF,NPT,SF
      PRINT,
      CALL ANAS3(ALPHA,N,0)
      DO 10 I=1,NPT
      ARRAY1(I)=DARRAY(I)
 10   CONTINUE
      CALL ANAS3(QALPHA,N,0)
      DO 20 I=1,NPT
      ARRAY2(I)=DARRAY(I)
 20   CONTINUE
      CALL ANAS3(FALPHA,N,0)
      DO 30 I=1,NPT
      ARRAY3(I)=DARRAY(I)
 30   CONTINUE
      CALL PLTAXS(FREQ,ARRAY1,ARRAY2,ARRAY3,NPT)
      RETURN
      END
```

B4.0- Simulation Programs

This program was used to simulate the WDFs designed. As with the ANAWDF program, this is also menu d riven. The program includes the simulation of WDFs using the traditional adaptors and the systolic adaptors. It is also possible to see the effects of quantization on the simulation results by quantizing the coefficients and the signals to the required number of bits. The inputs to the filters can be chosen to be an impulse, a step, a sinewave or a combination of sinewaves. The simulation results, i.e the inputs and the outputs of the filters, can be stored in a file which can be plotted using a general purpose plot program developed by the author.

```
C---------------------------------------------------
C
C PROGRAM TO SIMULATE WAVE DIGITAL FILTERS (WDFS)
C BASED ON UNIT ELEMENT, LATTICE AND LC-LADDER
C REFERENCE FILTERS.(FORTRAN 77)
C
C AUTHOR : A. R. MIRZAI         DATE : APRIL 1986
C
C            THE CENTRE FOR INFORMATION ENG
C                 THE CITY UNIVERSITY
C            NORTHAMPTON SQ, LONDON EC1VOHB
C
C---------------------------------------------------
      INTEGER I
      OPEN(UNIT=1,FILE='UECOEF.I',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='READ')
      OPEN(UNIT=2,FILE='UECOEF.F',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='READ')
      OPEN(UNIT=3,FILE='LTCOEF.I',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='READ')
      OPEN(UNIT=4,FILE='LTCOEF.F',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='READ')
      OPEN(UNIT=5,FILE='LCCOEF.I',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='READ')
      OPEN(UNIT=6,FILE='LCCOEF.F',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='READ')
      OPEN(UNIT=7,FILE='INSIN',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='WRITE')
      OPEN(UNIT=8,FILE='INIMP',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='WRITE')
      OPEN(UNIT=9,FILE='INSTP',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='WRITE')
      OPEN(UNIT=10,FILE='OUTUE',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='WRITE')
      OPEN(UNIT=11,FILE='OUTLT',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='WRITE')
      OPEN(UNIT=12,FILE='OUTLC',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='WRITE')
      OPEN(UNIT=13,FILE='OUTEM',STATUS='OLD',FORM='UNFORMATTED'
     &,ACTION='WRITE')
5     PRINT,
      PRINT,
      PRINT,'              PROGRAM TO SIMULATE WDFS'
      PRINT,
      PRINT,'                      MAIN MENU'
      PRINT,
      PRINT,'        1) SIMULATE UNIT ELEMENT WDFS.'
      PRINT,'        2) SIMULATE LATTICE WDFS.'
      PRINT,'        3) SIMULATE LC-LADDER WDFS.'
      PRINT,'        4) END SIMULATION PROGRAM.'
      PRINT,
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ,I
      PRINT,
      GOTO(10,20,30,40),I
      I=0
      GOTO 5
10    CALL SIMUE
      GOTO 5
20    CALL SIMLT
      GOTO 5
30    PRINT,
      GOTO 5
40    STOP
      END
C
C
C
      SUBROUTINE SIMUE
      REAL  RIPSET(300),QIPSET(300),RCOEFSET(20),QCOEFSET(20)
     &,RIPSET1(300)
     &,ROP(300),QOP(300),SOP(300)
      INTEGER IPSET(300),COEFSET(20),MNB,NBS,NBC,NSECT,NSPT,CHOICE
5     PRINT,
      PRINT,
      PRINT,
      PRINT,'              PROGRAM TO SIMULATE  '
      PRINT,'               UNIT ELEMNET WDFS'
      PRINT,
      PRINT,'        1) READ COEFFICIENTS.'
      PRINT,'        2) CHOOSE INPUT SIGNAL.'
      PRINT,'        3) SIMULATE USING 2-PORT ADAPTOR.'
      PRINT,'        4) SIMULATE USING 2-PORT SYSTOLIC ADAPTOR.'
      PRINT,'        5) SIMULATE USING QUANTIZED SIGNALS & COEFF.'
      PRINT,'        6) SAVE FILTER RESPONCES.'
      PRINT,'        7) RETURN TO MAIN MENU.'
      PRINT,
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ,CHOICE
      PRINT,
      GOTO(10,20,30,40,50,60,70),CHOICE
      CHOICE=0
      GOTO 5
10    CALL RDCOEF(COEFSET,RCOEFSET,QCOEFSET,NSECT,NBS,NBC,MNB,0)
      GOTO 5
20    CALL RDIPSI(IPSET,RIPSET,QIPSET,NSPT,NBS)
      GOTO 5
30    DO 35 I=1,NSPT
      RIPSET1(I)=RIPSET(I)
35    CONTINUE
      CALL SIMDUE(RIPSET1,RCOEFSET,ROP,NSECT,NBS,NBC,MNB,NSPT)
      GOTO 5
40    DO 45 I=1,NSPT
      RIPSET1(I)=RIPSET(I)
45    CONTINUE
      CALL SIMSUE(IPSET,RIPSET1,COEFSET,SOP,NSECT,NBS,NBC,MNB,NSPT)
      GOTO 5
50    CALL SIMDUE(QIPSET,QCOEFSET,QOP,NSECT,NBS,NBC,MNB,NSPT)
      GOTO 5
60    CALL SAVEFR(ROP,QOP,SOP,NSPT,0)
```

```
      GOTO 5
   70 RETURN
      END
C
C
C
      SUBROUTINE SIMLT
      REAL RIPSET(300),QIPSET(300),RCOEFSET(20),QCOEFSET(20)
     &,RIPSET1(300),ROP(300),QOP(300),SOP(300)
      INTEGER IPSET(300),COEFSET(20),MNB,NBS,NBC,NSECT,NSPT,CHOICE
    5 PRINT,
      PRINT,
      PRINT,
      PRINT,'                  PROGRAM TO SIMULATE'
      PRINT,'                     LATTICE WDFS'
      PRINT,
      PRINT,'        1) READ COEFFICIENTS.'
      PRINT,'        2) CHOOSE INPUT SIGNAL.'
      PRINT,'        3) SIMULATE USING 2-PORT ADAPTOR.'
      PRINT,'        4) SIMULATE USING 2-PORT SYSTOLIC ADAPTOR.'
      PRINT,'        5) SIMULATE USING QUANTIZED SIGNAL & COEFF.'
      PRINT,'        6) SAVE FILTER RESPONCES.'
      PRINT,'        7) RETURN TO MAIN MENU.'
      PRINT,
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ,CHOICE
      PRINT,
      GOTO(10,20,30,40,50,60,70),CHOICE
      CHOICE=0
      GOTO 5
   10 CALL RDCOEF(COEFSET,RCOEFSET,QCOEFSET,NSECT,NBS,NBC,MNB,2)
      GOTO 5
   20 CALL RDIPSI(IPSET,RIPSET,QIPSET,NSPT,NBS)
      GOTO 5
   30 DO 35 I=1,NSPT
      RIPSET1(I)=RIPSET(I)
   35 CONTINUE
      CALL SIMDLT(RIPSET1,RCOEFSET,ROP,NSECT,NBS,NBC,MNB,NSPT)
      GOTO 5
   40 DO 45 I=1,NSPT
      RIPSET1(I)=RIPSET(I)
   45 CONTINUE
      CALL SIMSLT(IPSET,RIPSET1,COEFSET,SOP,NSECT,NBS,NBC,MNB,NSPT)
      GOTO 5
   50 CALL SIMDLT(QIPSET,QCOEFSET,QOP,NSECT,NBS,NBC,MNB,NSPT)
      GOTO 5
   60 CALL SAVEFR(ROP,QOP,SOP,NSPT,1)
      GOTO 5
   70 RETURN
      END
C
C
C
      SUBROUTINE RDCOEF(COEFSET,RCOEFSET,QCOEFSET,NSECT,NBS,NBC,MNB,IFT)
      INTEGER COEFSET(20),NSECT,NBS,NBC,MNB,I
```

```
      REAL RCOEFSET(20),X(20),QCOEFSET(20),QL
      PRINT,
      PRINT,'NBS  NBC  ORDER  '
      READ, NBS,NBC,NSECT
      PRINT,
      MNB=NBS+NBC+2
      IF(IFT.EQ.0) NSECT=NSECT+1
      IF(IFT.EQ.4) MNB=MNB+1
      QL=.5**NBC
    5 PRINT,
      PRINT,
      PRINT,'YOU CAN CHOICETHE COEFFICIENTS FROM,'
      PRINT,
      PRINT,'  1) THE INFINITE PRECISION, OR'
      PRINT,'  2) THE FINITE PRECISION FILE.'
      PRINT,
      PRINT,'ENTER YOUR CHOICE NUMBER :-'
      READ,I
      PRINT,
      REWIND(I+IFT)
      READ(I+IFT,*) (X(J),J=1,NSECT)
      DO 10 I=1,NSECT
      QCOEFSET(I)=SIGN(QL*FLOAT(INT(ABS(X(I))/QL+0.5)),X(I))
      RCOEFSET(I)=X(I)
      COEFSET(I)=SIGN(1,X(I))*INT(ABS(X(I))*2**(NBC-1))
      PRINT,X(I),COEFSET(I),QCOEFSET(I)
   10 CONTINUE
      PRINT,
      RETURN
      END
C
C
C
      SUBROUTINE RDIPSI(IPSET,RIPSET,QIPSET,NSPT,NBS)
      INTEGER IPSET(300),NSPT,NBS,I,CHOIC
      REAL RIPSET(300),QIPSET(300)
    5 PRINT,
      PRINT,'THE INPUT CAN BE :-'
      PRINT,
      PRINT,'  1) SINEWAVES, OR'
      PRINT,'  2) AN IMPULSE, OR'
      PRINT,'  3) A STEP.'
      PRINT,
      PRINT,'THE INPUT CHOICE IS :-'
      READ, CHOIC
      PRINT,
      GOTO (10,20,30),CHOIC
      GOTO 5
   10 CALL SINWAV(IPSET,RIPSET,QIPSET,NSPT,NBS)
      RETURN
   20 CALL IMPULS(IPSET,RIPSET,QIPSET,NSPT,NBS)
      RETURN
   30 CALL STEP(IPSET,RIPSET,QIPSET,NSPT,NBS)
      RETURN
      END
```

```
C
C
C
      SUBROUTINE SINWAV(IPSET,RIPSET,QIPSET,NSPT,NBS)
      INTEGER IPSET(300),NSPT,I,NBS
      REAL RIPSET(300),X,FREQ,QIPSET(300),QL
      PI=3.14159263
      PRINT,
      PRINT,'NUMBER OF SAMPLE POINTS IS :-'
      READ, NSPT
      PRINT,
      PRINT,'FREQUENCY OF  THE FIRST SINEWAVE IS :-'
      READ, FREQ1
      PRINT,
      PRINT,'FREQUENCY OF THE SECOND SINEWAVE IS :-'
      READ,FREQ2
      PRINT,
      QL=.5**NBS
      REWIND(7)
      DO 10 I=1,NSPT
      X1=SIN(2*PI*FREQ1*(I-1))
      X2=SIN(2*PI*FREQ2*(I-1))
      X=(X1+X2)/2
      IF(ABS(X).EQ.1) X=SIGN(1,X)*(0.999999999)
      WRITE(7,*) X1,X2,X
      QIPSET(I)=SIGN(QL*FLOAT(INT(ABS(X)/QL+0.5)),A)
      RIPSET(I)=X
      IPSET(I)=SIGN(1,X)*INT(ABS(X)*2**(NBS-1))
 10   CONTINUE
      PRINT,
      RETURN
      END
C
C
C
      SUBROUTINE IMPULS(IPSET,RIPSET,QIPSET,NSPT,NBS)
      INTEGER IPSET(300)
      REAL RIPSET(300),QIPSET(300),QL
      PRINT,
      PRINT,'ENTER THE NUMBER OF SAMPLE POINTS REQUIRED :-'
      READ, NSPT
      PRINT,
      PRINT,'ENTER THE POSITION OF THE IMPULSE :-'
      READ, IPOS
      PRINT,
      QL=0.5**NBS
      REWIND(8)
      DO 10 I=1,NSPT
      RIPSET(I)=0
      QIPSET(I)=0
      IPSET(I)=0
 10   CONTINUE
      QIPSET(IPOS)=QL*FLOAT(INT(1/QL+0.5))
      RIPSET(IPOS)=0.99999999
      IPSET(IPOS)=INT(0.9999999*2**(NBS-1))
```

```
      DO 20 I=1,NSPT
      WRITE(8,*) RIPSET(I)
 20   CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE STEP(IPSET,RIPSET,QIPSET,NSPT,NBS)
      INTEGER IPSET(300),NSPT,NBS,I
      REAL RIPSET(300),QIPSET(300),QL
      PRINT,
      PRINT,'THE NUMBER OF INPUT SAMPLE IS :-'
      READ, NSPT
      PRINT,NSPT
      PRINT,
      PRINT,'THE POSITION OF THE STEP IS :-'
      READ, IPOS
      PRINT,IPOS
      PRINT,
      QL=0.5**NBS
      REWIND(9)
      DO 10 I=1,IPOS-1
      RIPSET(I)=0
      IPSET(I)=0
      WRITE(9,*) RIPSET(I)
 10   CONTINUE
      DO 20 I=IPOS,NSPT
      QIPSET(I)=QL*FLOAT(INT(1/QL+0.5))
      RIPSET(I)=(0.999999999)
      IPSET(I)=INT(RIPSET(I)*2**(NBS-1))
      WRITE(9,*) RIPSET(I)
 20   CONTINUE
      PRINT,
      RETURN
      END
C
C
C
      SUBROUTINE SIMSUE(IPSET,RIPSET,COEFSET,OPSET
     &,NSECT,NBS,NBC,MNB,NSPT)
      REAL OPSET(300),GAIN,RIPSET(300)
      INTEGER NBS,NBC,MNB,OUT1,OUT2,NSECT,XK,UKP1,KCOEFF,
     &IPSET(300),COEFSET(20),USET(22)
      LOGICAL OVERF
      ISCAL=1
 10   DO 20 I=1,NSECT+1
      USET(I)=0
 20   CONTINUE
      OVERF=.FALSE.
      DF=2**(NBS-1)
      REWIND(4)
      DO 50 I=1,NSPT
      XK=IPSET(I)
      DO 40 K=1,NSECT
      UKP1=USET(K+1)
      KCOEFF=COEFSET(K)
      CALL SYSADP(XK,UKP1,OUT1,OUT2,KCOEFF,NBS,NBC,MNB,OVERF)
      IF(OVERF) GOTO 30
      USET(K)=OUT2
      XK=OUT1
      GOTO 40
 30   ISCAL=ISCAL+1
      CALL SCALIP(IPSET,RIPSET,NSPT,ISCAL,NBS)
      GOTO 10
 40   CONTINUE
      OPSET(I)=OUT1/DF
 50   CONTINUE
      IF(ISCAL.EQ.1) GOTO 60
      PRINT,'OVERFLOW HAS BEEN DETECTED DURING THE CALCULATION.'
      PRINT,'THE INPUT NEEDED TO BE SCALED DOWN BY FACTOR OF,',ISCAL
      PRINT,
      PRINT,'TYPR <CR> TO CONTINE !!!'
      READ,I
      PRINT,
 60   RETURN
      END
C
C
C
      SUBROUTINE SIMDUE(IPSET,COEFSET,OPSET,NSECT,NBS,NBC,MNB,NSPT)
      REAL IPSET(300),COEFSET(20),OPSET(300),GAIN,XK,UKP1,UK,
     &USET(22),KCOEF
      INTEGER NSECT,NBS,NBC,MNB,ISCAL,ISTART
      LOGICAL OVERF
 5    ISCAL=1
 10   DO 20 I=1,NSECT+1
      USET(I)=0
 20   CONTINUE
      OVERF=.FALSE.
      DO 50 I=1,NSPT
      XK=IPSET(I)
      DO 40 K=1,NSECT
      UKP1=USET(K+1)
      KCOEF=COEFSET(K)
      CALL TWOPRT(XK,UKP1,KCOEF,UK,XKP1,OVERF)
      IF(OVERF) GOTO 30
      USET(K)=UK
      XK=XKP1
      GOTO 40
 30   ISCAL=ISCAL+1
      CALL SCLIP2(IPSET,NSPT,ISCAL)
      GOTO 10
 40   CONTINUE
      OPSET(I)=XK
 50   CONTINUE
      IF(ISCAL.EQ.1) GOTO 60
      PRINT,'OVERFLOW HAS BEEN DETECTED DURING THE CALCULATION.'
      PRINT,'THE INUPT NEEDED TO BE SCALED DOWN BY FACTOR OF,',ISCAL
      PRINT,
      PRINT,'TYPE <CR> TO CONTINUE !!!'
      READ,I
      PRINT,
 60   RETURN
      END
C
C
C
      SUBROUTINE TWOPRT(XK,UKP1,KCOEF,UK,XKP1,OVERF)
      REAL XK,UK,XKP1,UKP1,KCOEF,D
      LOGICAL OVERF
      D=KCOEF*(UKP1-XK)
      UK=UKP1+D
      XKP1=XK+D
      IF(UK.LT.1 .AND. XKP1.LT.1) GOTO 10
      PRINT,
      OVERF=.TRUE.
 10   RETURN
      END
C
C
C
      SUBROUTINE SCLIP2(IPSET,NSPT,ISCAL)
      REAL IPSET(300)
      INTEGER NSPT,I,ISCAL
      DO 10 I=1,NSPT
      IPSET(I)=IPSET(I)/ISCAL
 10   CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE SCALIP(IPSET,RIPSET,NSPT,ISCAL,NBS)
      INTEGER IPSET(300),NBS,ISCAL,NSPT,I
      REAL RIPSET(300),QIPSET(300),QL
      QL=.5**NBS
      DO 10 I=1,NSPT
      RIPSET(I)=RIPSET(I)/ISCAL
      X=RIPSET(I)
      IPSET(I)=SIGN(1,X)*INT(ABS(X)*2**(NBS-1))
 10   CONTINUE
      PRINT,
      RETURN
      END
C
C
C
      SUBROUTINE SIMSLT(IPSET,RIPSET,COEFSET,OPSET
     &,NSECT,NBS,NBC,MNB,NSPT)
      REAL OPSET(300),RIPSET(300),DF
      INTEGER NBS,NBC,MNB,OUT1,OUT2,NSECT,NSPT,IN1,IN2,X,USET(20)
     &,IPSET(300),COEFSET(20),IC1,IC2
      LOGICAL OVERF
      ISCAL=1
```

```
10    DO 20 I=1,NSECT
      USET(I)=0
20    CONTINUE
      OVERF=.FALSE.
      DF=2**(NBS-1)
      REWIND(7)
      DO 70 I=1,NSPT
      IN1=IPSET(I)
      IN2=IPSET(I)
      CALL SYSADP(IN1,USET(1),X,OUT1,COEFSET(1),NBS,NBC,MNB,OVERF)
      IF(OVERF) GOTO 50
      IN1=OUT1
      USET(1)=X
      IC1=1
      IC2=2
30    CALL BLOCK1(COEFSET(IC2),COEFSET(IC2+1),IN2,OUT2,USET(IC2)
     &,USET(IC2+1),NBS,NBC,MNB,OVERF)
      IF(OVERF) GOTO 50
      IN2=OUT2
      IC1=IC1+2
      IF(IC1.EQ.NSECT) GOTO 40
      CALL BLOCK1(COEFSET(IC2+2),COEFSET(IC2+3),IN1,OUT1,USET(IC2+2)
     &,USET(IC2+3),NBS,NBC,MNB,OVERF)
      IF(OVERF) GOTO 50
      IN1=OUT1
      IC1=IC1+2
      IC2=IC2+4
      IF(IC1.NE.NSECT) GOTO 30
40    OPSET(I)=(OUT1+OUT2)/(2*DF)
      GOTO 70
50    ISCAL=ISCAL+1
      CALL SCALIP(IPSET,RIPSET,NSPT,ISCAL,NBS)
      GOTO 10
70    CONTINUE
      IF(ISCAL.EQ.1) GOTO 80
      PRINT,'OVERFLOW HAS BEEN DETECTED DURING THE CALCULATIONS.'
      PRINT,'THE INPUT NEEDED TO BE SCALED DOWN BY FACTOR OF,',ISCAL
      PRINT,
      PRINT,'TYPE <CR> TO CONTINUE !!!'
      READ,I
      PRINT,
80    RETURN
      END
C
C
C
      SUBROUTINE BLOCK1(A1,A2,IN,OUT,USET1,USET2,NBS,NBC,MNB,OVERF)
      INTEGER NBS,NBC,MNB,A1,A2,IN,OUT,USET1,USET2,X1,X2,X3
      LOGICAL OVERF
      CALL SYSADP(IN,USET1,X1,OUT,A1,NBS,NBC,MNB,OVERF)
      CALL SYSADP(X1,USET2,X3,X2,A2,NBS,NBC,MNB,OVERF)
      USET1=X2
      USET2=X3
      RETURN
      END
```

```
C
C
C
      SUBROUTINE SIMDLT(IPSET,COEFSET,OPSET
     &,NSECT,NBS,NBC,MNB,NSPT)
      REAL IPSET(300),OPSET(300),OUT1,OUT2,IN1,IN2,USET(20)
     &,COEFSET(20),X
      INTEGER NSECT,I,NPT,IC1,IC2
      CHARACTER*1 CH
      LOGICAL OVERF
      COMMON OVERF
      ISCAS=1
10    DO 20 I=1,NSECT
      USET(I)=0
20    CONTINUE
      OVERF=.FALSE.
      DO 70 I=1,NSPT
      IN1=IPSET(I)
      IN2=IPSET(I)
      CALL TWOPRT(IN1,USET(1),COEFSET(1),OUT1,X,OVERF)
      IF(OVERF) GOTO 50
      IN1=OUT1
      USET(1)=X
      IC1=1
      IC2=2
30    CALL BLOCK2(COEFSET(IC2),COEFSET(IC2+1),IN2,OUT2,USET(IC2),
     &USET(IC2+1))
      IF(OVERF) GOTO 50
      IN2=OUT2
      IC1=IC1+2
      IF(IC1.EQ.NSECT) GOTO 40
      CALL BLOCK2(COEFSET(IC2+2),COEFSET(IC2+3),IN1,OUT1,USET(IC2+2)
     &,USET(IC2+3))
      IF(OVERF) GOTO 50
      IN1=OUT1
      IC1=IC1+2
      IC2=IC2+4
      IF(IC1.NE.NSECT) GOTO 30
40    OPSET(I)=(OUT1+OUT2)/2
      GOTO 70
50    ISCAL=ISCAL+1
      CALL SCLIP2(IPSET,NSPT,ISCAL)
      GOTO 10
70    CONTINUE
      IF(ISCAL.EQ.1) GOTO 80
      PRINT,'OVERFLOW HAS BEEN DETECTED DURING THE CALCULATIONS.'
      PRINT,'THE INPUT NEEDED TO BE SCALED DOWN BY FACTOR OF,',ISCAL
      PRINT,
      PRINT,'TYPE <CR> TO CONTINUE !!!'
      READ,I
      PRINT,
80    RETURN
      END
C
C
```

```
C
      SUBROUTINE BLOCK2(A1,A2,IN,OUT,USET1,USET2)
      REAL A1,A2,IN,OUT,USET1,USET2,X1,X2,X3
      LOGICAL OVERF
      COMMON OVERF
      CALL TWOPRT(IN,USET1,A1,OUT,X1,OVERF)
      CALL TWOPRT(X1,USET2,A2,X2,X3,OVERF)
      USET1=X2
      USET2=X3
      RETURN
      END
C
C
C
      SUBROUTINE SAVEFR(ROP,QOP,SOP,NSPT,IFT)
      REAL ROP(300),QOP(300),SOP(300)
      PRINT,
      PRINT,'     REAL     QUANT    SYSTOLIC'
      PRINT,
      REWIND(10+IFT)
      DO 10 I=1,NSPT
      PRINT,I,ROP(I),QOP(I),SOP(I)
      WRITE(10+IFT,*) ROP(I),QOP(I),SOP(I)
 10   CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE INIT(XK,XKA,UKP1,UKP1A,ALPHA,ALPHAA,NBS,NBC,MNB)
      INTEGER XKA(26),UKP1A(26),ALPHAA(8),PU(9,27),PX(9,27),
     &CPU(8,27),CPX(8,27),CS(8,27),DA(8,26),NBS,NBC,MNB,XK,UKP1,ALPHA
     &,DX,DY,DXA(26),DYA(26)
      COMMON PU,PX,CPU,CPX,CS,DA
      DX=XK*2**(NBC-1)
      DY=UKP1*2**(NBC-1)
      CALL DTOB(XK,XKA,MNB)
      CALL DTOB(UKP1,UKP1A,MNB)
      CALL DTOB(ALPHA,ALPHAA,NBC)
      CALL DTOB(DX,DXA,MNB)
      CALL DTOB(DY,DYA,MNB)
      DO 5 K1=1,27
      DO 5 K2=1,9
      PX(K2,K1)=0
      PU(K2,K1)=0
 5    CONTINUE
      DO 7 K1=1,26
      DO 7 K2=1,8
      DA(K2,K1)=0
      CPX(K2,K1)=0
      CPU(K2,K1)=0
      CS(K2,K1)=0
 7    CONTINUE
      CPX(NBC,1)=ALPHAA(NBC)
      CPU(NBC,1)=ALPHAA(NBC)
```

```
      DO 10 K=2,NBC+1
      PX(K,1)=DXA(K-1)
      PU(K,1)=DYA(K-1)
 10   CONTINUE
      DO 20 K=2,(MNB-NBC+1)
      PX((NBC+1),K)=DXA(NBC+K-1)
      PU((NBC+1),K)=DYA(NBC+K-1)
 20   CONTINUE
      DO 30 K=1,MNB
      DA(NBC,K)=1
 30   CONTINUE
      DO 40 K=1,NBC
      CS(K,1)=1
 40   CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE DTOB(A,ARRAY,NB)
      INTEGER A,ARRAY(27),NB,R
      IF (A.EQ.0) GOTO 40
      IF (A.LT.0) A=(2**NB)+A
      K=1
 10   IF (A.EQ.1) GOTO 20
      R=INT(A/2)
      ARRAY(K)=1
      IF ((R*2).EQ.A) ARRAY(K)=0
      A=R
      K=K+1
      GOTO 10
 20   ARRAY(K)=1
      DO 30 J=K+1,NB
      ARRAY(J)=0
 30   CONTINUE
      GOTO 60
 40   DO 50 K=1,NB
      ARRAY(K)=0
 50   CONTINUE
 60   RETURN
      END
C
C
C
      SUBROUTINE SYSADP(IP1,IP2,OUT1,OUT2,ACOEF,NBS,NBC,MNB,OVERF)
      INTEGER IN1(27),IN2(27),COEFF(8),XKP1(27),PU(9,27),PX(9,27)
     &,UK(27),CPU(8,27),CPX(8,27),CS(8,27),DA(8,27),NBS,NBC,MNB,XK,
     &UKP1,ALPHA,D,OPX,OPU,NPX,NPU,OCPX,OCPU,NCPX,NCPU,OCS,NCS,J,I,
     &K,NCELL,COUNT,OUT2,OUT1
     &,IX,UKP1,ACOEF
      LOGICAL OVERF
      COMMON PU,PX,CPU,CPX,CS,DA
      COMMON /B/ OPX,NPX,OPU,NPU,OCPX,NCPX,OCPU,NCPU,OCS,NCS
      IDX=IP1
      IDUKP1=IP2
      ICOEFF=ACOEF
      CALL INIT(IDX,IN1,IDUKP1,IN2,ICOEFF,COEFF,NBS,NBC,MNB)
      NCELL=1
      COUNT=1
      J=1
      DO 30 K=1,MNB
      DO 10 I=NCELL,1,-1
      ALPHA=COEFF(I)
      UKP1=IN2(J)
      XK=IN1(J)
      D=DA(I,J)
      OPX=PX(I+1,J)
      OPU=PU(I+1,J)
      OCPX=CPX(I,J)
      OCPU=CPU(I,J)
      OCS=CS(I,J)
      CALL CELL(ALPHA,XK,UKP1,D)
      PU(I,J+1)=NPU
      PX(I,J+1)=NPX
      CPU(I,J+1)=NCPU
      CPX(I,J+1)=NCPX
      CS(I,J+1)=NCS
      J=J+1
10    CONTINUE
      UK(K)=PU(1,K+1)
      XKP1(K)=PX(1,K+1)
      J=1
      IF (NCELL.LT.NBC) GOTO 20
      J=J+COUNT
      COUNT=COUNT+1
      GOTO 30
20    NCELL=NCELL+1
30    CONTINUE
      CALL BTOD(X,XKP1,MNB,NBC)
      CALL BTOD(Y,UK,MNB,NBC)
      IF((ABS(Y).LT.1) .AND. (ABS(X).LT.1)) GOTO 40
      OVERF=.TRUE.
      IF(ABS(X).GE.1) X=SIGN(1,X)*(0.9999999999)
40    OUT1=SIGN(1,X)*INT(ABS(X)*2**(NBS-1))
      OUT2=SIGN(1,Y)*INT(ABS(Y)*2**(NBS-1))
      RETURN
      END
C
C
C
      SUBROUTINE CELL(ALPHA,XK,UKP1,D)
      INTEGER NXK,NS,UKP1,XK,OCS,S,NCS,D,OCPX,NCPX,OCPU,NCPU,OPX,NPX
     &,ALPHA,D,NPU,OPU
      COMMON /B/ OPX,NPX,OPU,NPU,OCPX,NCPX,OCPU,NCPU,OCS,NCS
      NXK=XK
      NS=0
      IF (NXK.EQ.0) NS=1
      NXK=NS
      CALL ADD(UKP1,NXK,OCS,S,NCS)
      IF (D.EQ.0) GOTO 10
      NS=0
      IF (S.EQ.0) NS=1
      S=NS
10    CALL MULTIP(OPX,ALPHA,S,OCPX,NPX,NCPX)
      CALL MULTIP(OPU,ALPHA,S,OCPU,NPU,NCPU)
      RETURN
      END
C
C
C
      SUBROUTINE ADD(A,B,CIN,S,COUT)
      INTEGER A,B,CIN,COUT,S
      S=A+B+CIN
      IF (S.LT.2) GOTO 10
      IF (S.EQ.2) GOTO 20
      S=1
      COUT=1
      GOTO 30
10    COUT=0
      GOTO 30
20    S=0
      COUT=1
30    RETURN
      END
C
C
C
      SUBROUTINE MULTIP(A,B,C,CIN,P,COUT)
      INTEGER A,B,C,CIN,COUT,P,BANDC
      BANDC=B*C
      CALL ADD(A,BANDC,CIN,P,COUT)
      RETURN
      END
C
C
C
      SUBROUTINE BTOD(A,ARRAY,MNB,NBC)
      INTEGER ARRAY(27),NB,K
      REAL A,P
      A=0
      I=4
      DO 10 K=MNB,MNB-3,-1
      A=A+ARRAY(K)*2**I
      I=I-1
10    CONTINUE
      I=1
      DO 20 K=MNB-4,1,-1
      P=2**I
      A=A+ARRAY(K)/P
      I=I+1
20    CONTINUE
      IF(ARRAY(MNB).EQ.1) A=-(2**5-1-A)
      RETURN
      END
```

APPENDIX 'C'

SCATTERING MATRIX

In this Appendix, we present some relationships between the reflectances, transmittances, scattering matrix parameters and the wave quantities of a 2-port network. Fig. C.1a shows a 2-port network terminated between resistances R1 and R2 and the voltage sources E1 and E2. Let the corresponding wave network of N be N' (Fig. c1b). Also, let $\underline{S}$ and $\underline{S}'$ denote the scattering matrices corresponding to N and N' respectively. The entries of $\underline{S}$,

$$\underline{S} = \begin{vmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{vmatrix}$$

are known to be given by [86],

$$S_{11} = (Z1 - R1)/(Z1 + R1)$$

$$S_{21} = 2-(R1/R2)\ V2/E1 \Big|_{E2=0}$$

$$S_{22} = (Z2 - R2)/(Z2 + R2)$$

$$S_{12} = 2-(R1/R2)\ V1/E2 \Big|_{E1=0}$$

where Z1 and Z2 are the input impedences at port one and two respectively. Also, according to eqn 1.21, we have,

$$A1 = V1 + R1.I1 \qquad A2 = V2 + R2.I2$$

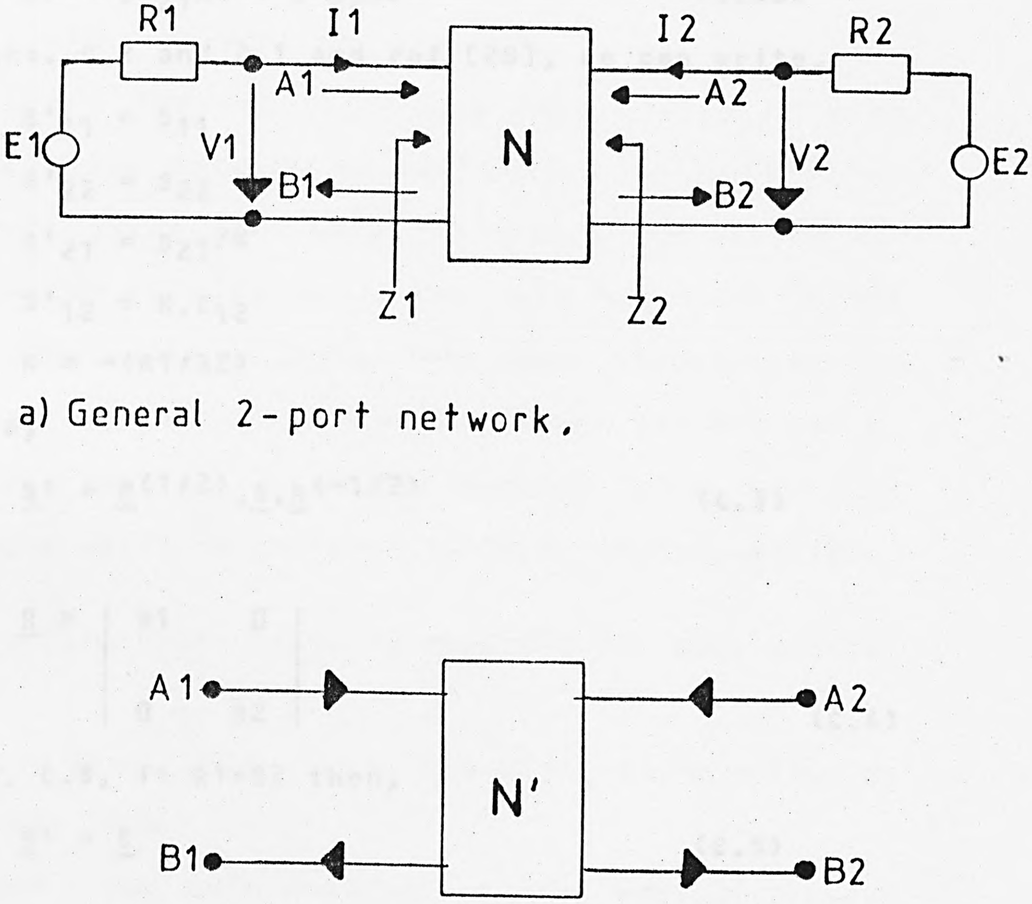
and

$$B1 = V1 - R1.I1 \qquad B2 = V2 - R2.I2$$

Therefore, from Fig C.1 and Ref [28], we obtain,

$$A1 = E1 \qquad A2 = E2$$

Fig.C.1



a) General 2-port network.

b) Corresponding WDF.

and $$B1 \Big|_{E1=0} = 2V1 \qquad B2 \Big|_{E2=0} = 2V2$$

B1 and B2 can be expressed in terms of A1, A2 and the entries of $\underline{S}'$ as shown below,

$$B1 = S'_{11}A1 + S'_{12}A2 \qquad (C.2a)$$

and $$B2 = S'_{21}A1 + S'_{22}A2 \qquad (C.2b)$$

Using eqns. C.2 and C.1 and ref [28], we can write,

$$S'_{11} = S_{11}$$

$$S'_{22} = S_{22}$$

$$S'_{21} = S_{21}/K$$

$$S'_{12} = K.S_{12}$$

and $$K = -(R1/R2)$$

Therefore,

$$\underline{S}' = \underline{R}^{(1/2)}.\underline{S}.\underline{R}^{(-1/2)} \qquad (C.3)$$

where

$$\underline{R} = \begin{vmatrix} R1 & 0 \\ 0 & R2 \end{vmatrix} \qquad (C.4)$$

From eqn. C.3, if R1=R2 then,

$$\underline{S}' = \underline{S} \qquad (C.5)$$

APPENDIX 'D'

LIST OF AUTHOR'S PUBLICATIONS

1) **Title** : "Systolic Wave Digital Filter Structures suitable for VLSI implementation."

Source : Proc. of IEE Saraga Colloquium on Electronic Filters, London, pp 9/1-9/7, May 1985.

Abstract : The excellent low sensitivity of Wave Digital Filters (WDF's) to variations in multiplier coefficients makes them very attractive for speech and communication applications. The main drawback is the hardware complexity of WDF's. This paper illustrates how a WDF based on the cascaded unit element filters can be transformed into a regular and modular 1-bit systolic architecture which is suitable for VLSI implementation.

2) **Title** : "Bit-Level Systolic Adaptors for Wave Digital Filters."

Source : Proc. IEEE ISCAS, San Jose, California, pp 853-856, May 1986.

Abstract : The main drawback of Wave Digital Filters (WDF's) is the hardware complexity. In general, the hardware implementation of WDFs depends on how efficiently two and three port adaptors are implemented. This hardware complexity can therefore be resolved by considering the VLSI implementation of WDF adaptors. This paper illustrates how two and three port adaptors can be transformed into regular and modular 1-bit systolic architectures which are suitable for VLSI implementation.

3) **Title** : "Finite Wordlength Design of Wave Digital Filters."

Source : Electronics Letters, Vol-22, No. 16, pp 851-853, July 1986.

Abstract : The letter illustrates some examples from a software package for the desgin of finite wordlength wave digital filters (WDFs). Given a set of initial coefficients, the program generates a new set of coefficients which are quantised to the required number of bits. Some filter design examples have been considered to illustrate this for three type of WDFs, i.e unit element, lattice and LC-ladder WDFs.

4) **Title** : "Software Tools for the Design and an approach to the VLSI implementation of Wave Digital Filters."

Source : Submitted for publication in the Special Issue of the IEEE Proc. on "Hardware and Software for Digital Signal Proccessing."

Abstract : In spite of the excellent low sensitivity of Wave Digital Filters (WDFs) to variations in the multiplier coefficients, the practical applications of WDFs are limited due to their hardware complexity. In this paper, we resolve this problem by considering the bit-level systolic implementation of WDF adaptors. Also a complete software package is described which enables one to design finite wordlength WDFs based on three well known reference filters, i.e unit element, lattice and lc-ladder filters.