1995

Digital network analysis automation.

Jacky W. Y. Luk
University of Windsor

NOTICE

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

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

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

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

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

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

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

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

Digital Network Analysis Automation

by

Jacky W. Y. Luk

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

Windsor, Ontario, Canada
August, 1995

ISBN 0-612-10944-5

Name Jacky Lui

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

(R) Electronics & Electrical
SUBJECT TERM

0594
SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

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

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

LANGUAGE, LITERATURE AND LINGUISTICS
- Language
  - General 0679
  - Ancient 0289
  - Linguistics 0290
  - Modern 0291
- Literature
  - General 0401
  - Classical 0294
  - Comparative 0295
  - Medieval 0297
  - Modern 0298
  - African 0316
  - American 0591
  - Asian 0305
  - Canadian (English) 0352
  - Canadian (French) 0355
  - English 0593
  - Germanic 0311
  - Latin American 0312
  - Middle Eastern 0315
  - Romance 0313
  - Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY
- Philosophy 0422
- Religion
  - General 0318
  - Biblical Studies 0321
  - Clergy 0319
  - History of 0320
  - Philosophy of 0322
- Theology 0469

SOCIAL SCIENCES
- American Studies 0323
- Anthropology
  - Archaeology 0324
  - Cultural 0326
  - Physical 0327
- Business Administration
  - General 0310
  - Accounting 0272
  - Banking 0770
  - Management 0454
  - Marketing 0338
- Canadian Studies 0385
- Economics
  - General 0501
  - Agricultural 0503
  - Commerce-Business 0505
  - Finance 0508
  - History 0509
  - Labor 0510
  - Theory 0511
- Folklore 0358
- Geography 0366
- Gerontology 0351
- History
  - General 0578
  - Ancient 0579
  - Medieval 0581
  - Modern 0582
  - Black 0328
  - African 0331
  - Asia, Australia and Oceania 0332
  - Canadian 0334
  - European 0335
  - Latin American 0336
  - Middle Eastern 0333
  - United States 0337
- History of Science 0585
- Law 0398
- Political Science
  - General 0615
  - International Law and Relations 0616
  - Public Administration 0617
- Recreation 0814
- Social Work 0452
- Sociology
  - General 0626
  - Criminology and Penology 0627
  - Demography 0938
  - Ethnic and Racial Studies 0631
  - Individual and Family Studies 0628
  - Industrial and Labor Relations 0629
  - Public and Social Welfare 0630
  - Social Structure and Development 0700
  - Theory and Methods 0344
- Transportation 0709
- Urban and Regional Planning 0999
- Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES
- Agriculture
  - General 0473
  - Agronomy 0285
  - Animal Culture and Nutrition 0475
  - Animal Pathology 0476
  - Food Science and Technology 0359
  - Forestry and Wildlife 0478
  - Plant Culture 0479
  - Plant Pathology 0480
  - Plant Physiology 0817
  - Range Management 0777
  - Wood Technology 0746
- Biology
  - General 0306
  - Anatomy 0287
  - Biostatistics 0308
  - Botany 0309
  - Cell 0379
  - Ecology 0329
  - Entomology 0353
  - Genetics 0369
  - Limnology 0793
  - Microbiology 0410
  - Molecular 0307
  - Neuroscience 0317
  - Oceanography 0416
  - Physiology 0433
  - Radiation 0821
  - Veterinary Science 0778
  - Zoology 0472
- Biophysics
  - General 0786
  - Medical 0760

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

HEALTH AND ENVIRONMENTAL SCIENCES
- Environmental Sciences 0768
- Health Sciences
  - General 0566
  - Audiology 0300
  - Chemotherapy 0992
  - Dentistry 0567
  - Education 0350
  - Hospital Management 0769
  - Human Development 0758
  - Immunology 0982
  - Medicine and Surgery 0564
  - Mental Health 0347
  - Nursing 0569
  - Nutrition 0570
  - Obstetrics and Gynecology 0380
  - Occupational Health and Therapy 0354
  - Ophthalmology 0381
  - Pathology 0571
  - Pharmacology 0419
  - Pharmacy 0572
  - Physical Therapy 0382
  - Public Health 0573
  - Radiology 0574
  - Recreation 0575
  - Speech Pathology 0460
  - Toxicology 0383
- Home Economics 0386

PHYSICAL SCIENCES

Pure Sciences
- Chemistry
  - General 0485
  - Agricultural 0749
  - Analytical 0486
  - Biochemistry 0487
  - Inorganic 0488
  - Nuclear 0738
  - Organic 0490
  - Pharmaceutical 0491
  - Physical 0494
  - Polymer 0495
  - Radiation 0754
- Mathematics 0405
- Physics
  - General 0605
  - Acoustics 0986
  - Astronomy and Astrophysics 0606
  - Atmospheric Science 0608
  - Atomic 0748
  - Electronics and Electricity 0607
  - Elementary Particles and High Energy 0798
  - Fluid and Plasma 0759
  - Molecular 0609
  - Nuclear 0610
  - Optics 0752
  - Radiation 0756
  - Solid State 0611
- Statistics 0463

Applied Sciences
- Applied Mechanics 0346
- Computer Science 0984
- Engineering
  - General 0537
  - Aerospace 0538
  - Agricultural 0539
  - Automotive 0540
  - Biomedical 0541
  - Chemical 0542
  - Civil 0543
  - Electronics and Electrical 0544
  - Heat and Thermodynamics 0348
  - Hydraulic 0545
  - Industrial 0546
  - Marine 0547
  - Materials Science 0794
  - Mechanical 0548
  - Metallurgy 0743
  - Mining 0551
  - Nuclear 0552
  - Packaging 0549
  - Petroleum 0765
  - Sanitary and Municipal 0554
  - System Science 0790
- Geotechnology 0428
- Operations Research 0796
- Plastics Technology 0795
- Textile Technology 0994

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

Dedicated with love

to my parents,

Abstract

This thesis presents the development of a software package for the purpose of analyzing digital networks. The software package consists of two graphic user interfaces, one for the signal flow graph schematic drawing, and one for the analysis of digital network. Both are written in the WATFOR-77 programming language and by using the VGAWAT graphics library. The first graphic user interface software is to draw the signal flow graph of a 1-D, 2-D, or 3-D digital network using adders, delays, and multipliers. It can extract the graphical representation of a digital network with error checking capability into a netlist for analysis purpose. The netlist is formatted in the simplified matrix representation format. The second graphic user interface software uses the netlist to compute the frequency response, group delay and slope of magnitude response, first-order coefficient sensitivity, noise, and impulse response of a 1-D, 2-D, or 3-D digital network. Results are plotted in 2-D graphs for 1-D digital networks, and in 3-D graphs for 2-D and 3-D digital networks. Examples of 1-D, 2-D, and 3-D digital networks are given in this thesis.

Acknowledgements

I would like to acknowledge Dr. H. K. Kwan, my thesis supervisor, for suggesting to me this project and for his numerous suggestions throughout this project to add and improve the various features of the two graphic user interface software packages or programs presented in this thesis. I would also like to acknowledge Dr. Kwan for introducing me the graphic user interface software, the node numbering, minimization and renumbering methods presented in this thesis; for providing me the signal flow graphs of the digital network examples presented in this thesis and a copy of his DINCAP User Manual; for his suggestion on the cell editor work for systolic digital networks; and for his help in the use of his 1-D, 2-D, and 3-D DINCAP analysis computer programs, and in the software development of the 2-D and 3-D plots. I am also grateful to my department reader, Prof. P. H. Alexander, and my outside department reader, Dr. M. Wang for their comments. Last but not least, I would like to express my deepest gratitude to my parents, my twin sister, and Lily Tjhia for their support and encouragement.

Table of Contents

Chapter 4 Signal Flow Graph Schematic Drawing Software

Chapter 5 Digital Networks Analysis Software

Chapter 6 Summary and Future Works

List of Figures

List of Tables

Chapter 1

Introduction

1.1 CAD Software of Digital Network Design

Because of the increasing complexity of digital networks, CAD software has become popular in today's digital network design. The approach of designing a digital network can be concluded to obtain its transfer function with appropriate specification. Software has been developed to design the transfer function. ***SIG*** is a general-purpose customize signal processing, analysis, and display program [1]. Its main purpose is to perform manipulations on time- and frequency-domain signals; however it accommodates other representations for data such as transfer function polynomials. ***filtorX*** is a programming language for filter design which is both customizable and extensible [2]. It is an interactive design aid which is based on mathematical objects of filter design: complex numbers and rational functions. It is even capable of designing more complicated digital filters, such as multi-rate switched capacitor filter design with aggressive sampling-rates [3] and passive ladder structure digital filters [4]. Other CAD software packages are written for the design of special-structure digital filters, such as ***FILSYN*** [5], ***PANDDA*** [6], ***FDT*** [7] and ***DIGICAP*** [8].

1.2 Motivation of this Thesis

Once a transfer function is designed, digital filter structure realization naturally follows. During the past decade, considerable effort has been focussed on the design of digital filter structures for hardware implementation. A more efficient approach was proposed that forms the framework for the representation and analysis of digital filters, whereby

the number of nodes and topology of a structure have little influence on the computational complexity [9][38]. An analysis package, ***DINCAP*** [9][38], was also developed. It is capable of various time and frequency domain analyses of 1-D, 2-D, and 3-D digital filters by using the method in [9][38]. However, the netlist which describes the digital filter and is used for the analysis package has to be extracted manually. It is desirable to develop a software package which can extract the netlist automatically from the graphical representation of digital filter structure (Signal Flow Graph).

1.3 Thesis Organization

Chapter 2 gives an account of the representation of a digital network with a signal flow graph. Detailed derivations of the matrix representation and simplified matrix representation are also introduced. Five different types of analyses and several typical types of structures are mentioned.

Chapter 3 discusses the programming language and graphics library used for the development of the software. Some basic graphics capabilities of VGAWAT are mentioned. They include : initialization, mouse handling, creating frames, panels and filling colors, drawing lines and text, 3-D drawing, XOR mode and keyboard input and output. Extended graphics capabilities of the graphics library and how they can be used in practice are also described.

Chapter 4 starts with the development of the general software mechanisms. Then, features of the signal flow graph schematic drawing software are discussed. Finally, the method of obtaining the netlist from the signal flow graph is mentioned. There are two kinds of netlists to be used for the analysis of digital network. First of all, a common netlist which stores the numerical values includes: total number of delay nodes in 1-D, 2-D and 3-D, total number of signal nodes, signal node number that network output is drawn, total number of non-zero elements in the transmittance matrix, and transmittance matrix. Secondly, a unique netlist stores the information that depends on the type of analysis. The format of the common netlist and the unique netlist can be found in

Appendix A.

Chapter 5 discusses the development and features of digital network analysis software. The software is able to conduct five different types of analyses such as frequency response, group delay and slope of magnitude response, noise, first order coefficient sensitivity and impulse response, using the netlists that are created by the software as mentioned in chapter 4. The output data will be plotted in 2-D graphs for 1-D digital networks or 3-D graphs for 2-D and 3-D digital networks.

Finally, Chapter 6 summarizes the work done and discusses the possible improvements of both software packages.

Chapter 2

Multidimensional Digital Networks

2.1 Introduction

Due to significant advances in digital computer technology and integrated-circuit fabrication, multidimensional digital signal processing has been developed rapidly, and the application of digital filters is also widely applied in our real-world. In one dimension, the most usual application is in the time domain, in which they may be applied to the modification of speech or music signals, either to enhance the intelligibility or to reduce the noise. To be more general, it can be applied in the field of data or speech communication. It can also be used in seismology, biomedical engineering, acoustics, sonar and many others. In two dimensions, the manipulation of data is commonly referred to as image processing. Interest in digital image processing methods stems from two principal application areas: improvement of pictorial information for human interpretation, and processing of scene data for autonomous machine perception [10]. Fields of application can be in any area where 2-D data is encountered, such as automatic character recognition, industrial robots for product assembly and inspection, military reconnaissance, automatic processing of fingerprints, screening of x-rays, and machine processing of aerial and satellite imagery for weather prediction and crop assessment. With the knowledge of one dimensional and two dimensional digital signal processing, digital networks can even be extended to three dimensions.

Implementation of digital networks in special-purpose hardware has become more and more common. Because of this increasing practicality, there has been considerable effort directed towards the development and investigation of digital filter structures. Digital filter structures can be conveniently represented in terms of linear signal flow graphs or equivalent matrix representation. Through this matrix representation, many of the theorems and signal flow graphs can be specifically adapted for digital networks.

The signal flow graph representation can utilize Tellegen's theorem to show that the matrix of transfer functions for a network and its transpose are interreciprocal [12][13][14][15][16]. It also leads to a number of useful relations for sensitivity analysis of digital filter structures [12][13][15][16].

The matrix representation of a digital network can be used to analyze several properties of the network. Frequency response and phase response are major filter characteristics we use to impart frequency dependent attenuation and phase shift onto the input signal, respectively. Impulse response analysis is the response of the system to a unit sample excitation in the time-domain. Slope of magnitude response and group delay response are used to look at the slope of the frequency response versus frequency and the negative slope of phase response versus frequency, respectively. These are the common analyses when we design a digital network.

When one implements a digital filter transfer function using a digital machine, it invariably involves quantization of signals and coefficients in the system. As a result, the overall input-output behaviour is not ideal [11]. The error due to arithmetic roundoff can be used in noise analysis to determine the noise power density spectrum. Another error due to coefficient quantization can be used in first order coefficient sensitivity analysis to compute the lower and upper passband, and stopband statistical word lengths of a digital network. In fact, the quantization of a coefficient can affect the overall filter characteristics.

Each different structure has different noise and sensitivity behaviour even with the same input-output relationships. Therefore, the choice of a filter structure is an important

consideration in designing a digital filter. In this chapter, we will briefly discuss some common structures that are used to design digital networks. They are, direct-form II, ladder, cascade, parallel and wave-digital structures. With the present technology, digital network structures can use the systolic architectures for implementation because of their regularity and ease of reconfiguration.

2.2 Signal Flow Graph Representation of Digital Networks

From a mathematical point of view, a circuit or system can be described by a set of simultaneous equations. The algebraic solutions can be obtained either by the matrix method or by the method of successive substitutions [17]. A signal flow graph is a diagram showing a collection of nodes and directed branches [18]. Dependent or independent variables are represented by nodes. Nodes can be separated into signal nodes, delay nodes, source nodes and sink nodes. When relationships exist between nodes, then branches are used to represent such relationships. A branch can be defined as the directed line joining two nodes with the direction which is indicated by an arrow. For the case of a digital network, the value specified by a multiplier which is called transmittance, and the node and branch signals are discrete-time signals or their z-transforms. The relationships of nodes, branches and transmittance are shown in Fig 2.1, where a branch joins node a and b with a transmittance T_{ab}.

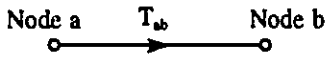


Fig. 2.1 : Relationships of Nodes, Branch and Transmittance

2.2.1 Basic Elements

According to the basic operations required for implementation of a digital filter, the basic elements required to represent a set of difference equations pictorially are an adder, a delay, and a constant multiplier. Commonly used symbols are shown in Fig. 2.2. Physically, Fig 2.2a represents a means of adding two sequences together, Fig 2.2b

represents a means for multiplying a sequence by a constant, and Fig 2.2c represents a means for storing the previous value of a sequence.

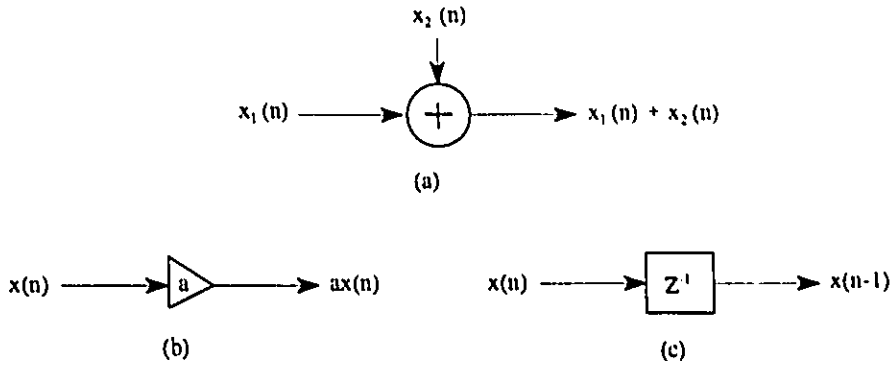


Fig. 2.2 : Block-Diagram Symbols for a Digital Network: (a) Addition of Two Sequences (b) Multiplication of a Sequence by a Constant (c) Unit Delay

In multidimensional digital networks, delay symbols of 1-D, 2-D and 3-D are distinguished as shown in Fig. 2.3. The discrete time signals in Fig 2.3 only show the changing of one discrete time axis, the remaining discrete time axis/axes are unchanged and not shown in the figure. Fig. 2.3a-2.3c represent a unit delay in discrete time n_1, n_2 and n_3, respectively. The operations of addition and multiplication will be the same in 1-D, 2-D and 3-D digital networks. The representation used for a single delay arises from the fact that the z-transform of x(n-1) is simply z^{-1} times the z-transform of x(n) in 1-D digital network. In 2-D digital network, the z-transform of a single delay, $x(n_1-1, ..)$ and $x(.. ,n_2-1)$, are z_1^{-1} and z_2^{-1} times the z-transform of $x(n_1, ..)$ and $x(.. ,n_2)$, respectively. In 3-D digital network, the z-transform of $x(n_1-1, .., ..)$, $x(.. ,n_2-1, ..)$ and $x(.. , ..,n_3-1)$ are z_1^{-1}, z_2^{-1} and z_3^{-1} times the z-transform of $x(n_1, .., ..)$, $x(.. ,n_2, ..)$ and $x(.., ..,n_3)$, respectively. The operations of addition and multiplication in z-domain will be the same as in discrete time domain.

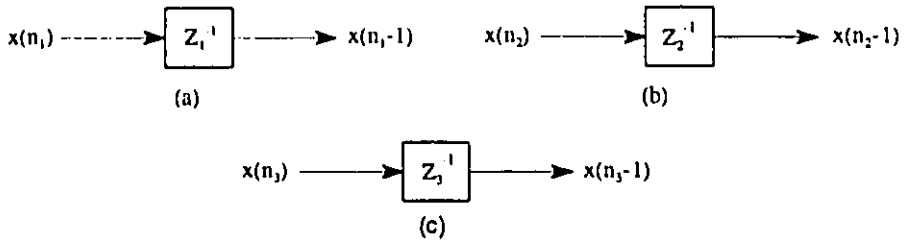


Fig. 2.3 : Block-Diagram of Delay Elements; (a) 1-D, (b) 2-D, (c) 3-D

Using the symbols as shown in Fig 2.2 and Fig. 2.3, we can construct any structures of any multidimensional linear digital networks. In other words, we can utilize these elements to represent any difference equations in digital filters. The amount of hardware required to realize the filter and the complexity of a digital filter algorithm will be depicted by these elements.

2.2.2 Definition of Nodes [9][38]

As mentioned in section 2.2, nodes can be separated as source nodes, sink nodes, delay nodes and signal nodes. Source nodes represent locations for injection of external inputs or sources into the graph. A source node has no entering branches as shown in Fig. 2.4a. Sink nodes represent nodes that extract outputs from the graph to the outside environment. A sink node has no outgoing branches as shown in Fig. 2.4b. Delay nodes represent nodes immediately after delay elements. Finally, the remaining nodes of the graph are defined as signal nodes. They can have multiple entering branches and multiple outgoing branches. It is convenient to represent a signal node immediately after a source node as an input that is injected into the digital network, and a signal node immediately before sink nodes as an output that is drawn from a digital network. The remaining nodes of the graph will then be delay nodes and signal nodes.

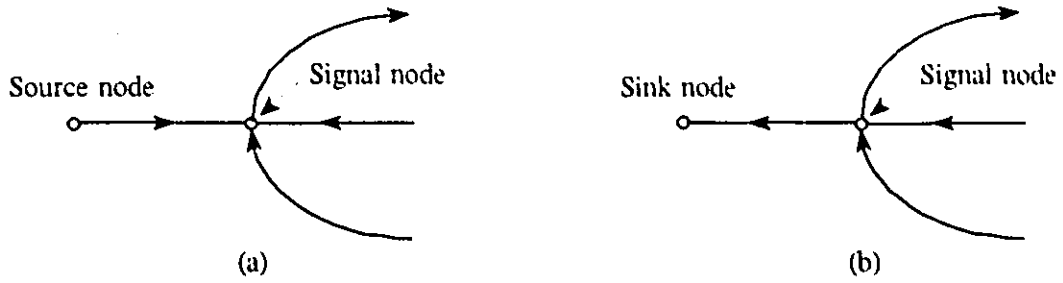


Fig. 2.4 : (a) Source Node/Input Node; (b) Sink Node/Output Node

There are some restrictions on the definition of a delay node. Only one incoming coefficient-delay branch is allowed, and one or more outgoing coefficient and/or coefficient-delay branch(es) are allowed as shown in Fig. 2.5a-b. In other words, it does not have any incoming source branch, and/or any incoming coefficient branch, and/or any outgoing output branch.

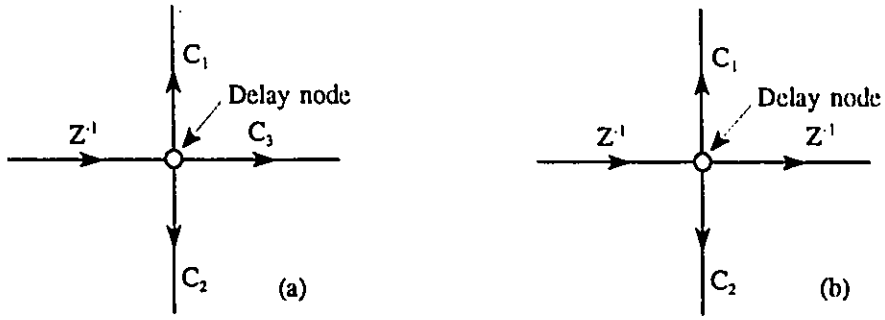


Fig. 2.5a-b : Definition of a Delay Node of a Digital Network (C_1, C_2, C_3 are real coefficients) [9][38]

Since an external input (source node) must be injected into and a network output (sink node) must be drawn from a signal node of the graph, Fig. 2.6a shows that there is no incoming coefficient-delay branch so that we can define the node that the external input is injected into as a signal node. In the case that a node has both an incoming coefficient-delay branch and an external input injected into, an artificially created signal node has to be defined. When an external input is injected into this artificially created signal node,

a delay node can also be defined as shown in Fig 2.6b. This restricts external inputs to be injected into the signal node. We can also define an output node using the same definition as for an input node. Fig. 2.7a shows that there is no incoming coefficient-delay branch so that the node can be defined as a signal node from which the output is extracted. Fig. 2.7b shows an artificially created signal node from which an output is drawn, and a delay node can also be defined. This also restricts that an output can only be extracted from a signal node.

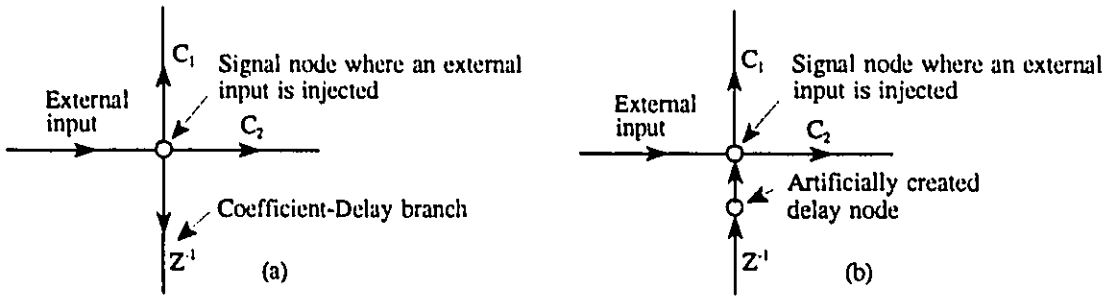


Fig. 2.6 : (a) Definition of an Input Node without an Incoming Coefficient-Delay Branch (b) Definition of an Input Node and Delay Node with an Incoming Coefficient-Delay Branch [9][38]

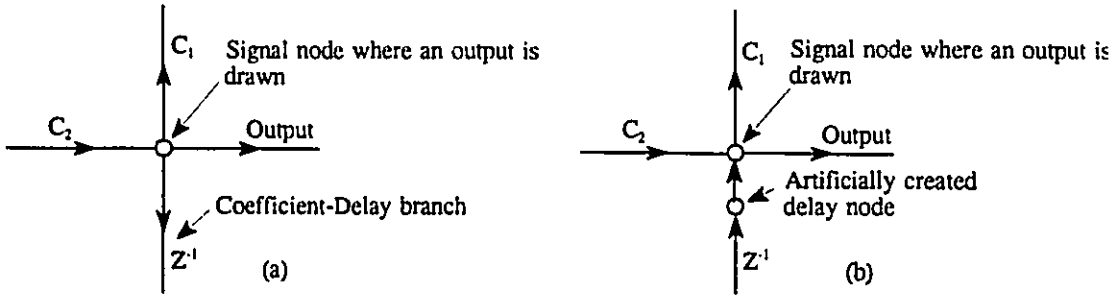


Fig. 2.7 : (a) Definition of an Output Node without an Incoming Coefficient-Delay Branch (b) Definition of an Output Node and Delay Node with an Incoming Coefficient-Delay Branch [9][38]

2.3 Matrix Representation and Simplified Matrix Representation

After a signal flow graph of a digital network has been constructed, sets of state-space equations have to be defined in order to solve each delay and signal node of the signal flow graph. Most of the digital signal processing books have mentioned the conversion of a topological view (signal flow graph) into mathematical view (state-space equations) by using the matrix representation of the digital filter [11][19][20][21][22][23].

The matrix representation has a disadvantage that the coefficient branch matrix and the coefficient-delay branch matrix are usually sparse, it requires a large amount of memory to store the matrix and it also leads to requiring more computer execution time. These problems mean the computer-aided design software for analysis of digital networks becomes less effective. However, H. K. Kwan proposed that the coefficient branch matrix and the coefficient-delay branch matrix can be combined to form one matrix [9][38]. The degree of sparsity can be reduced prior to any computation by numbering only the delay nodes and those signal nodes where signal values are required. This method is called simplified matrix representation and it can be extended to 2-D and 3-D digital network analysis.

2.3.1 Matrix Representation of Digital Networks

For the representation of a digital network, signal branches correspond to a constant gain. The input node of a signal branch has to be a signal node and those signal branches are called coefficient branches. Delay branches correspond to a constant gain in cascade with a unit delay. The input node of a delay branch has to be a delay node, and those branches are called coefficient-delay branches. The output of the coefficient branch from node j to node k will be denoted as $w_{cjk}(n)$ and the output of the delay branch from node j to node k will be denoted as $w_{djk}(n)$. The node signal value at node j will be denoted as $y_j(n)$. The constant gain of the coefficient branch which is connected from node j to node k will be denoted as f_{cjk} . The constant gain of the coefficient-delay branch which is connected from node j to node k will be denoted as f_{djk}. The relationship of branch output to branch input

will be

$$w_{cjk}(n) = f_{cjk}\, y_j(n) \tag{2.1a}$$

and

$$w_{djk}(n) = f_{djk}\, y_j(n-1) \tag{2.1b}$$

Fig. 2.8 shows a signal flow graph of a second order IIR digital filter with the notations of branches and nodes. The external input is injected into node 1 and the output is drawn from node 5. Node 3 and node 4 are numbered as delay nodes and the remaining nodes in the signal flow graph are signal nodes. The coefficient-delay branches f_{d23} and f_{d34} are equal to unity so that coefficient branches f_{c23} and f_{c34} have to be equal to zero. For example, the coefficient branch output $w_{c45}(n)$ equals to $f_{c45}y_4(n)$ and the coefficient-delay branch output $w_{d23}(n)$ equals to $f_{d23}y_2(n-1)$.

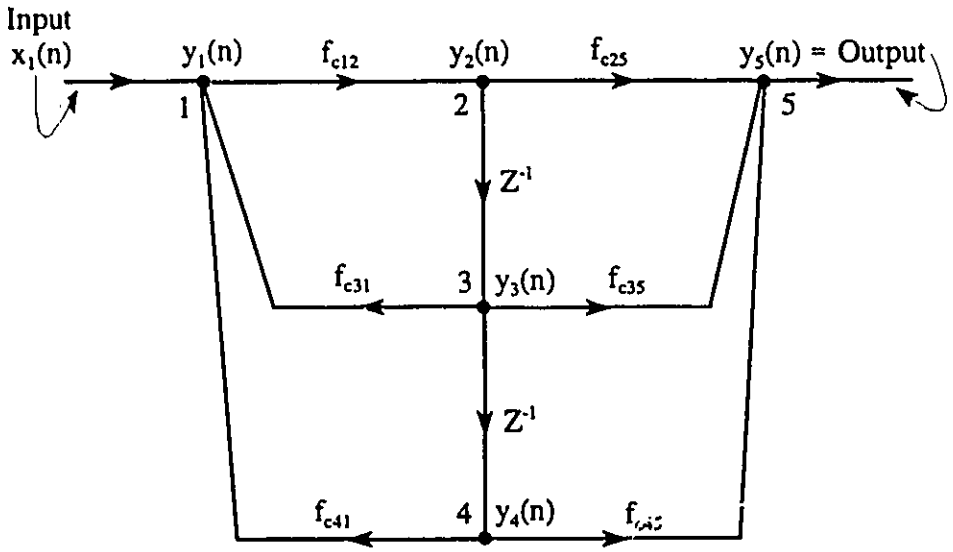


Fig. 2.8 : Signal Flow Graph Notation

Alternatively, the node values and branch outputs can be expressed in terms of their z-transforms, written as

$$W_{cjk}(z) = F_{cjk}(z)\, Y_j(z) \tag{2.2a}$$

and

$$W_{djk}(z) = F_{djk}(z)\, Y_j(z) \tag{2.2b}$$

where

$$F_{cjk}(z) = f_{cjk} \quad \text{and} \quad F_{djk}(z) = z^{-1} f_{djk}$$

The terms $F_{cjk}(z)$ and $F_{djk}(z)$ correspond to the branch transmittances of the coefficient and the coefficient-delay branches, respectively.

By definition, a node value at each node in a digital network is given by the sum of the outputs of all branches entering the node. Notationally, it is convenient to assume that there is a branch in each direction between every pair of network nodes and that each source node is connected to each network node, although clearly some of the branch outputs may then be zero. Therefore, the node signal value at each node in a network is the sum of the branch signals entering the node. Thus

$$Y_k(z) = X_k(z) + \sum_{j=1}^{N} [W_{cjk}(z) + W_{djk}(z)] \ , \quad k = 1, 2, \ldots, N \tag{2.3}$$

where N is the total number of nodes in the network. Alternatively, substituting from (2.2a) and (2.2b) for $W_{cjk}(z)$ and $W_{djk}(z)$ respectively, equation (2.3) can be written as

$$Y_k(z) = X_k(z) + \sum_{j=1}^{N} [f_{cjk} + f_{djk} z^{-1}] Y_j(z) \ , \quad k = 1, 2, \ldots, N \tag{2.4}$$

Equation (2.4) only shows the node output at node k. In order to obtain the total description of the network, N corresponding equations are required. Outputs of the network can be obtained through the signal nodes where outputs are drawn.

Then, N equations in (2.4) can be expressed more compactly in matrix form as

$$Y(z) = X(z) + f_c^t Y(z) + f_d^t Y(z) z^{-1} \tag{2.5}$$

where $\underline{Y}(z)$ is the column vector of the N nodes signal values, $\underline{X}(z)$ is the column vector of the N branch signal values of the source branches, $\underline{f}_c^t$ is a $N \times N$ matrix of the coefficient branches and $\underline{f}_d^t$ is a $N \times N$ matrix of the coefficient-delay branches. The superscript t in (2.5) denotes the matrix transpose, which is used for consistency between the subscript conventions of signal flow graph and matrices. In order to solve for node outputs $Y_k(z), k=1,...,N$, equation (2.5) has to be written as

$$(I - \underline{f}_c^t - \underline{f}_d^t z^{-1})\,\underline{Y}(z) = \underline{X}(z) \tag{2.6}$$

where I is a $N \times N$ identity matrix. According to equation (2.6), the Gauss-Elimination method can be used for solving node outputs.

The inverse z-transforms of (2.5) can be written as

$$\underline{y}(n) = \underline{x}(n) + \underline{f}_c^t \underline{y}(n) + \underline{f}_d^t \underline{y}(n-1) \tag{2.7}$$

and it expresses each node value as a linear combination of inputs and the present and previous values of node values. It is possible to have a situation in computing one node value where other node values have not yet been computed. Therefore, the computation of nodes has to be in sequence. In other words, the node numbers of a network have to be numbered so that the upper triangle and the diagonal of $\underline{f}_c^t$ are zero. If a network cannot be written in a form such that $\underline{f}_c^t$ is zero on and above the main diagonal, the network is non-computable.

2.3.2 Simplified Matrix Representation of Digital Networks [9][38]

In equation (2.6), $\underline{f}_c^t$ and $\underline{f}_d^t$ are usually sparse and they have the dimensions that equal to the total number of nodes in the network. It is necessary to simplify the matrices in order to obtain faster computational time. This can be done by taking advantage of the

sparsity of both matrices. In the matrix representation method, when the kth node is a delay node, the kth row of f_c^t will be set to zero. When the jth node is a signal node, the jth row of f_d^t will be set to zero. Therefore, f_c^t and f_d^t can be combined to form one matrix with $P \times P$ dimensions as shown

$$\begin{pmatrix} z^{-1}S & z^{-1}T \\ U & V \end{pmatrix} \tag{2.8}$$

where P is the total number of nodes in the network and S, T, U, V are real coefficient matrices. The node outputs $Y(z)$ are separated into delay node outputs $Y_d(z)$, and signal node outputs $Y_c(z)$. The input node $\_(z)$ are only allowed to be injected into the network through signal nodes. Therefore, it is written as $X_c(z)$, where the subscribe c represents signal node notation. Equation (2.7) will now be written as

$$\begin{pmatrix} I-z^{-1}S & -z^{-1}T \\ -U & I'-V \end{pmatrix} \begin{pmatrix} Y_d(z) \\ Y_c(z) \end{pmatrix} = \begin{pmatrix} 0 \\ X_c(z) \end{pmatrix} \tag{2.9}$$

Let N be the number of delay nodes and M be the number of signal nodes. Then, I is an identity matrix with $N \times N$ elements, I' is an identity matrix with $M \times M$ elements, S is a matrix with $N \times N$ elements, T is a matrix with $N \times M$ elements, U is a matrix with $M \times N$ elements, V is a matrix with $M \times M$ elements, $Y_d(z)$ is a column vector with $N \times 1$ elements , $Y_c(z)$ is a column vector with $M \times 1$ elements, 0 is a column vector with $N \times 1$ zero values and $X_c(z)$ is a column vector $M \times 1$ elements.

As mentioned in section 2.3.2, a computable network has to have all delay nodes numbered first; then the remaining signal nodes are numbered sequentially according to their orders of computation. Therefore, the delay node outputs $Y_d(z)$ are written before the signal node outputs $Y_c(z)$ in (2.9). Based on (2.9), the inverse z-transform can be written as

$$\underline{y}_d(n) = \underline{S}\,\underline{y}_d(n-1) + \underline{T}\,\underline{y}_c(n-1) \tag{2.10}$$

and

$$\underline{y}_c(n) = \underline{U}\,\underline{y}_d(n) + \underline{V}\,\underline{y}_c(n) + \underline{x}_c(n) \tag{2.11}$$

(2.10) and (2.11) are called Time Domain Form I (TDF I) of Simplified Matrix Representation (SMR). In order to improve the computational time, the signal node outputs $\underline{y}_c(n-1)$ in (2.10), have to be substituted by the delay node outputs $\underline{y}_d(n-1)$, and input signals $\underline{x}_c(n-1)$. Since, (2.11) can be written as

$$\underline{y}_c(n-1) = (\underline{I}'-\underline{V})^{-1}\,\underline{U}\,\underline{y}_d(n-1) + (\underline{I}'-\underline{V})^{-1}\,\underline{x}_c(n-1) \tag{2.12}$$

and substitute (2.12) into (2.10). Then,

$$\underline{y}_d(n) = [\underline{S} + \underline{T}(\underline{I}'-\underline{V})^{-1}\underline{U}]\,\underline{y}_d(n-1) + \underline{T}(\underline{I}'-\underline{V})^{-1}\underline{x}_c(n-1) \tag{2.13}$$

and (2.11) can be written in terms of $\underline{y}_d(n)$ and $\underline{x}_c(n)$ as

$$\underline{y}_c(n) = (\underline{I}'-\underline{V})^{-1}\,\underline{U}\,\underline{y}_d(n) + (\underline{I}'-\underline{V})^{-1}\,\underline{x}_c(n) \tag{2.14}$$

(2.13) and (2.14) are called Time Domain Form II (TDF II) of SMR. This gives a non-unique representation of a digital network with reduced computational complexity. In Frequency Domain Form (FDF), (2.9) can be rewritten as

$$(z\underline{I}-\underline{E})\underline{Y}_d(z) = \underline{F}\underline{X}_c(z) \tag{2.15}$$

$$\underline{Y}_c(z) = \underline{G}\underline{Y}_d(z) + \underline{H}\underline{X}_c(z) \tag{2.16}$$

where,

$$E = S + T(L' - V)^{-1}U \text{ and } E = T(L' - V)^{-1}$$

$$G = (L' - V)^{-1}U \text{ and } H = (L' - V)^{-1}$$

The transpose of a digital network can be obtained by taking the transpose of (2.9) and can be written as

$$\begin{pmatrix} I - z^{-1}S' & -U' \\ -z^{-1}T' & L' - V' \end{pmatrix} \begin{pmatrix} Y_d(z) \\ Y_c(z) \end{pmatrix} = \begin{pmatrix} 0 \\ X_c(z) \end{pmatrix} \tag{2.17}$$

The Time Domain Form II (TDF II) of (2.17) can be written as

$$y_d(n) = E' y_d(n-1) + G' x_c(n-1) \tag{2.18}$$

$$y_c(n) = E' y_d(n-1) + H' x_c(n) \tag{2.19}$$

and the Frequency Domain Form (FDF) of (2.17) can be written as

$$(zI - E)' Y_d(z) = zG' X_c(z) \tag{2.20}$$

$$Y_c(z) = z^{-1}E' Y_d(z) + H' X_c(z) \tag{2.21}$$

where,

$$E = S + T(L' - V)^{-1}U \text{ and } E = T(L' - V)^{-1}$$

$$G = (L' - V)^{-1}U \text{ and } H = (L' - V)^{-1}$$

2.4 General Network Analysis [9][38]

By using (2.13) and (2.14) of TDF II of the original digital network, (2.15) and (2.16) of FDF of the original digital network, (2.18) and (2.19) of TDF II of the transposed digital network, (2.20) and (2.21) of the FDF of the transposed digital network, we can apply any

of the five different analyses discussed below.

2.4.1 Frequency Response Analysis

Using (2.15) and (2.16), we can compute the frequency response and the phase response. From (2.15), the real part $\underline{Y}_{Rd}(e^{j\omega})$, and imaginary part $\underline{Y}_{Id}(e^{j\omega})$ of $\underline{Y}_{d}(e^{j\omega})$ can be expressed as

$$(I-2\cos\omega E+\underline{E}^2)\underline{Y}_{Rd}(e^{j\omega}) = (\cos\omega I-E)f_{ic} \tag{2.22}$$

$$(I-2\cos\omega E+\underline{E}^2)\underline{Y}_{Id}(e^{j\omega}) = -\sin\omega f_{ic} \tag{2.23}$$

where f_{ic} represents the column matrix at column i of matrix E and input signal is injected into signal node i. From (2.16), the real part $\underline{Y}_{Rc}(e^{j\omega})$, and imaginary part $\underline{Y}_{Ic}(e^{j\omega})$ of the signal node outputs $\underline{Y}_{c}(e^{j\omega})$ can be expressed as

$$Y_{Rcj}(e^{j\omega}) = g_{jr}\underline{Y}_{Rd}(e^{j\omega}) + h_{ji} \tag{2.24}$$

$$Y_{Icj}(e^{j\omega}) = g_{jr}\underline{Y}_{Id}(e^{j\omega}) \tag{2.25}$$

where g_{jr} represents the row matrix at row j of matrix G. (2.24) and (2.25) calculate the signal value at signal node j, $Y_{cj}(e^{j\omega})$, due to an input at signal i, $X_{ci}(e^{j\omega})$. Therefore, the computational time can be reduced by only calculating the signal values of every delay node in the network using (2.22) and (2.23), and extracting the network output value from the signal node j using (2.24) and (2.25), instead of computing every signal value of both delay nodes and signal nodes in the network. For the transposed digital network, at a particular frequency, the total response at a signal node j due to inputs from k arbitrary signal nodes of an original digital network can be obtained by summing the responses at signal node k due to an input at the signal node j of the transposed digital network. Then, (2.20) can be written as

$$(I-2\cos\omega E+\underline{E}^2)^t\underline{Y}_{Rd}(e^{j\omega}) = (I-\cos\omega E)^t g^t_{jr} \tag{2.26}$$

$$(I - 2\cos\omega E + E^2)^t Y_{Id}(e^{j\omega}) = (-\sin\omega E)^t g^t_{jr} \tag{2.27}$$

and the real part of (2.21) can be written as

$$Y_{Rci}(e^{j\omega}) = f^t_{iC}[\cos\omega Y_{Rd}(e^{j\omega}) + \sin\omega Y_{Id}(e^{j\omega})] + h_{ji} \tag{2.28}$$

and the imaginary part of (2.21) can be written as

$$Y_{Ici}(e^{j\omega}) = f^t_{iC}[\cos\omega Y_{Id}(e^{j\omega}) - \sin\omega Y_{Rd}(e^{j\omega})] \tag{2.29}$$

The magnitude response of both original digital network and the transposed digital network can be computed using

$$|Y_{cj}(e^{j\omega})| = \sqrt{Y_{Rcj}(e^{j\omega})^2 + Y_{Icj}(e^{j\omega})^2} \tag{2.30}$$

and the phase response can be computed using

$$\Theta(\omega) = \tan^{-1}\frac{Y_{Icj}(e^{j\omega})}{Y_{Rcj}(e^{j\omega})} \tag{2.31}$$

where the subscript cj represents the network output which is extracted from signal node j.

2.4.2 Group Delay and Slope of Magnitude Response Analysis

The evaluation of the group delay of a system function in a network and the slope of the magnitude of the frequency response are sometimes desired in general computer-aided analysis systems. With the phase response of the system derived in (2.31), then the group delay τ is given by

$$\tau = -\frac{\partial\Theta(\omega)}{\partial\omega} \tag{2.32}$$

The evaluation of the group delay can be derived in terms of appropriate system functions

in the network as in [11]

$$\tau = \sum_{n=1}^{N} \sum_{m=1}^{N} Re\left[\frac{f_{dnm} T_{an}(z) T_{mb}(z) z^{-1}}{T_{ab}(z)}\right]\Big|_{z=e^{j\omega}} \tag{2.33}$$

and the slope of magnitude of the frequency response, $\frac{\partial |H(e^{j\omega})|}{\partial \omega}$, is given by

$$\frac{\partial |H(e^{j\omega})|}{\partial \omega} = \frac{1}{|T_{ab}(e^{j\omega})|} \sum_{n=1}^{N} \sum_{m=1}^{N} Im\left[\frac{f_{dnm} T_{an}(z) T_{mb}(z) z^{-1}}{T_{ab}(z)}\right]\Big|_{z=e^{j\omega}} \tag{2.34}$$

where the double sums correspond to the operation of summing over all possible coefficient-delay branches in the network and N is the total number of delay nodes in the network. f_{dnm} is a nonzero coefficient value of a coefficient-delay branch from node n to node m. $T_{ab}(z)$ represents the overall network transfer function from node a (input) to node b (output). $T_{an}(z)$ is a transfer function which is directed from the input, a, of the network to an internal point, n, of the network and $T_{mb}(z)$ is a transfer function which is directed from an internal point, m, in the network to the output, b. $T_{an}(z)$ can be obtained from the analysis of the original network (2.22) and (2.23), and $T_{mb}(z)$ can be obtained from the analysis of the transpose network (2.26) and (2.27). $T_{ab}(z)$ can be obtained from either of the two analyses.

2.4.3 First Order Coefficients Sensitivity Analysis

Since in practice, multiplier coefficients can only be represented with a limited number of bits, this quantization effect produces a deviation of $|H(e^{j\omega})|$ from its ideal value and this is termed "sensitivity problem" [19]. It is obvious that using a large number of bits in the multiplier coefficients can obtain a response that is satisfactorily close to the infinite-precision response. However, using a minimum number of bits in the multiplication process in the hardware implementation is the major concern of designing digital filters. We cannot avoid the deviation of the frequency response due to the quantization of the multiplier coefficients. The only solution is to design a low-sensitivity structure where the variation of $|H(e^{j\omega})|$ with respect to multiplier values is small. The

first order coefficients sensitivity analysis is to compute the variation of $|H(e^{j\omega})|$, with respect to multiplier values of the designed structure (signal flow graph).

The expression for evaluating the sensitivities of the system function, with respect to the branch coefficients, has been proposed by various authors, including Seviora and Sablatash [12], Fettweis [13], and Lee [15], [16]. We use the derivation of the sensitivity proposed by [15] and expressed as

$$\frac{\partial|H(z)|}{\partial f_{cnm}} = Re[\frac{|H(z)|}{H(z)} T_{an}(z) T_{mb}(z)] \tag{2.35}$$

and

$$\frac{\partial|H(z)|}{f_{dnm}} = Re[\frac{|H(z)|}{H(z)} T_{an}(z) T_{mb}(z) z^{-1}] \tag{2.36}$$

where f_{cnm} and f_{dnm} are the real coefficients of the coefficient branches and coefficient-delay branches, respectively. H(z) is the system transfer function $T_{ab}(z)$. $T_{an}(z)$ and $T_{mb}(z)$ are described as in section 2.4.2.

2.4.4 Noise Analysis

There are two kinds of quantization effects when implementing digital filters. The first type is the quantization of multiplier coefficients which had been discussed in section 2.4.3. The second type of quantization is due to signal rounding. Since there is also a limited number of bits to represent the internal signal in the digital filter, when performing addition, overflow may occur in the feedback loop of the IIR digital filter. Therefore, quantization of the internal signal is necessary and the quantization error $e(n)$ is introduced and defined as $e(n)=Q[x]-x$. The quantization error, due to the arithmetic roundoff, will be injected into the network as shown in fig. 2.9.

In fig. 2.9, the transfer function due to $e_1(n)$ with $x(n)=e_2(n)=0$ can be written as

$$G_1(z) = \frac{Y_5(z)}{E_1(z)} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 - b_1 z^{-1} - b_2 z^{-2}}$$

and the transfer function due to $e_2(n)$ with $x(n)=e_1(n)=0$ can be written as

$$G_2(z) = \frac{Y_5(z)}{E_2(z)} = 1$$

There are k_j error sources input to the jth summation node and that each noise source is white, of uniform power spectrum density $\frac{Q^2}{12}$ where $Q=2^{-b}$= quantization step size, and b is the number of bits of the signal. The output power density spectrum may then be obtained using linear system theory [24] and expressed as

$$N_y(e^{j\omega}) = \frac{Q^2}{12}\sum_j k_j |G_j(e^{j\omega})|^2 \qquad (2.37)$$

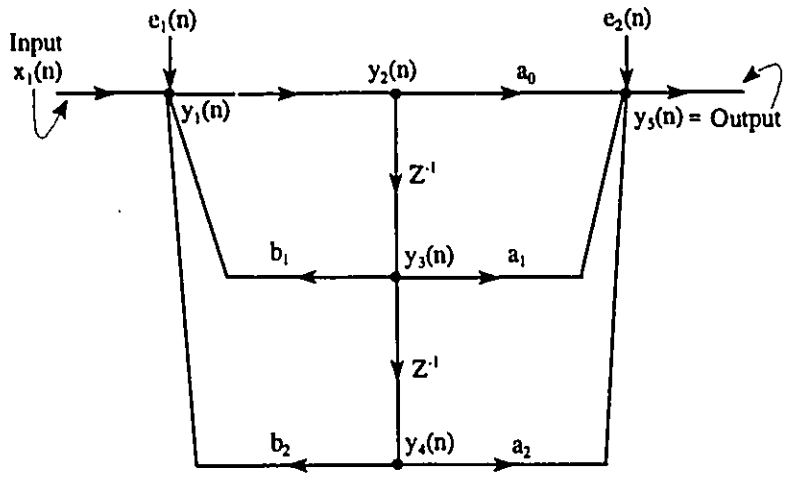


Fig. 2.9 : Direct Form Realization of Second-Order IIR Filter with Error Functions

2.4.5 Time Domain Analysis

The most common analysis in the time domain is to evaluate the impulse response of the system. This can be done by setting the input $x_{ci}(n)$ as an impulse and injected into the network at node i as

$$x_{ck}(n) = 1 \text{ , at } k=i \text{ and } n=0$$

$$= 0 \text{ , at other values of k and n}$$

Assume the network output is drawn from signal node j. By using (2.11) and (2.12), the output at node j, $y_{cj}(n)$, can be calculated by solving the N scalar equations corresponding to these matrix equations consecutively at time increment n. The initial conditions of the delay nodes and signal nodes output are chosen as $y_d(n)=0$ and $y_c(n)=0$ for $n < 0$.

As this set of calculations corresponds directly to that which must be performed in the actual implementation of the structure, simulation of effects such as limit cycles and overflow can also be incorporated into this procedure by duplicating on the computer the actual arithmetic operations which would be performed in the hardware implementation of the network [11].

2.5 Structures of Digital Network

The choice of the digital filter structures directly affects the word-length requirements for the signals and coefficients in the filter and determines its characteristics such as roundoff noise, limit cycle behaviour, and dynamic range [19]. Therefore, it is an important consideration in the design of the digital filter. The most simple structure can be directly obtained from the filter transfer function which is expressed as a ratio of two polynomials and is written as

$$H(z) = \frac{\sum_{k=1}^{M} a_k z^{-k}}{1 - \sum_{k=1}^{N} b_k z^{-k}} \tag{2.38}$$

, this is called direct-form II as shown in fig. 2.8. It has already been simplified to minimize the number of delay elements used. Another structure can also use the filter transfer function as a sum of quadratic sections and is expressed as

$$H(z) = \sum_{j=1}^{k} H_j(z) \tag{2.39}$$

where

$$H_i(z) = \frac{a_{0i} + a_{1i} z^{-1}}{1 + b_{1i} z^{-1} + b_{2i} z^{-2}} \tag{2.40}$$

$H_i(z)$ can be obtained by a partial-fraction expression to get the various sections. Fig. 2.10 shows the parallel realization.

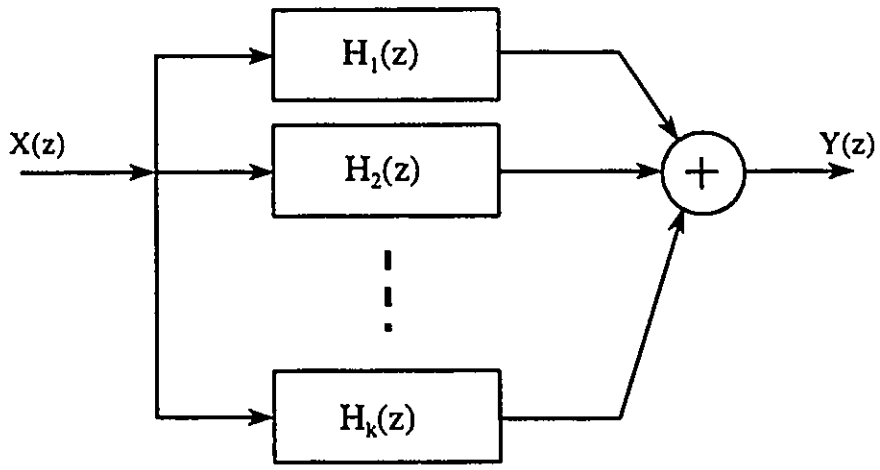


Fig. 2.10 : Parallel Realization

The third structure of using the filter transfer function as a product of ratios of second-order polynomials is called cascade structure and is written as

$$H(z) = \prod_{i=1}^{K} H_i(z) \tag{2.41}$$

where

$$H_i(z) = \frac{a_{0i} + a_{1i} z^{-1} + a_{2i} z^{-2}}{1 + b_{1i} z^{-1} + b_{2i} z^{-2}} \tag{2.42}$$

A typical cascade realization is shown in fig. 2.11.

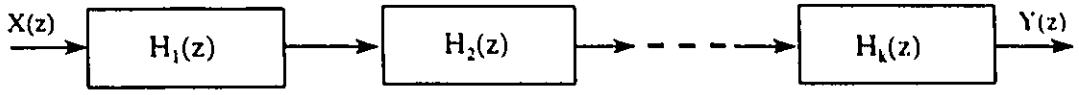


Fig. 2.11 : Cascade Realization

The previous three structures are synthesized by representing the system function in terms of some factorization or expansion. Unlike the previous category, Gray and Markel [25] introduced a lattice or ladder structure so that the mechanisms behind the synthesis procedures for these structures are closely related to the conventional two-port network procedures used in analog circuit theory [21]. The overall appearance is shown in Fig. 2.12, where each building block is a digital two-pair characterized by a single parameter k_m, making a total of N parameters altogether [19]. The N building blocks in Fig. 2.12 have the same structure as shown in Fig. 2.13. In order to realize an arbitrary transfer function, a pole-zero system, the structure in Fig. 2.12 is extended to a tapped cascaded lattice structures [25] as shown in Fig. 2.14. In Fig. 2.14, $C_m, m = 0, 2, \ldots, N$, are computed depending on the desired transfer function numerator.

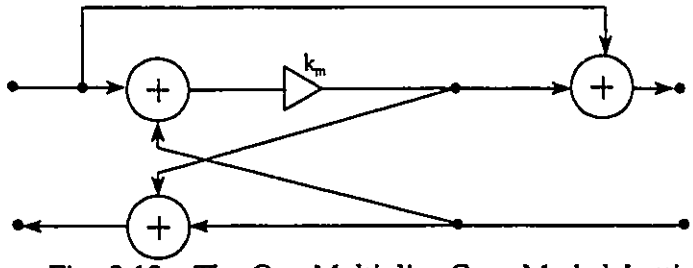


Fig. 2.12 : The One-Multiplier Gray-Markel Lattice

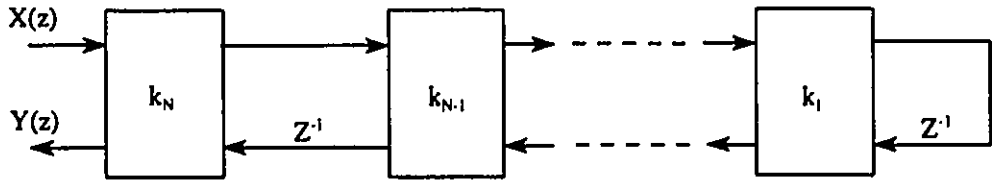


Fig. 2.13 : The Cascaded Gray and Markel Lattice Structures

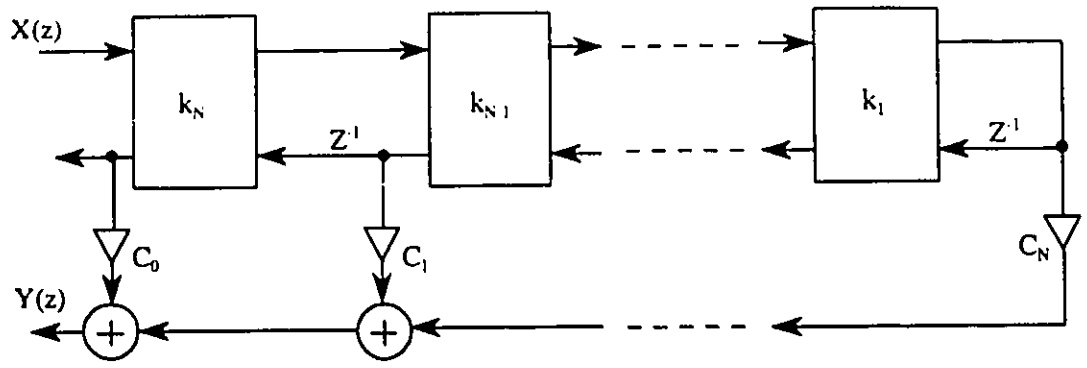


Fig. 2.14 : The Tapped Cascaded Lattice Structure

Another class of structures, which is based on digital simulation of continuous-time LC filters, was introduced by Fettweis [26]. This approach leads to the design of low-sensitivity digital filters called wave digital filters. There exists a class of continuous-time LC filters, called doubly terminated LC structures [27] as shown in Fig. 2.15. The explanation for this is based on the concepts of maximum available power and perfect impedance matching [28]. Then, the digital structures are obtained by converting these analog designs to corresponding digital designs. When a digital filter structure is built to simulate such a LC network, it inherits the low passband sensitivity property [19]. In addition, due to the inherent passivity of the LC prototype, the digital filter is also passive in a certain sense [29], and this can be exploited to suppress limit cycle oscillations. Since there are many ways to design LC analog structures, there are many ways to synthesize the wave digital filters.

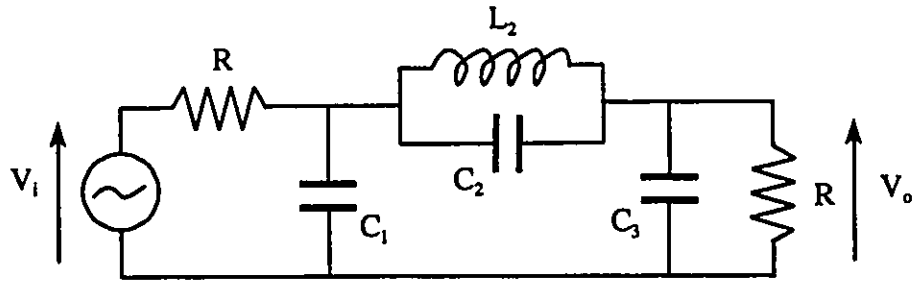


Fig. 2.15 : A Doubly Terminated LC Filter

2.6 Systolic Arrays

Another approach of designing digital filters is to use systolic architecture which was introduced by H. T. Kung [30]. A systolic system is characterized by its regularity with modular structures and is easy to reconfigure. If a structure can truly be decomposed into a few types of simple substructures or building blocks, which are used repetitively with simple interfaces, great savings in VLSI design can be achieved. Except for the low cost of using systolic system for implementing a special-purpose hardware, the ultimate performance that a computation rate balances the available I/O bandwidth with host is also an advantage.

A systolic system consists of a set of interconnected cells, each capable of performing some simple operation. Because simple, regular communication and control structures have substantial advantages over complicated ones in design and implementation, cells in a systolic system are typically interconnected to form a systolic array or a systolic tree [30]. Signal and image processing is one of the fields that can make use of the systolic design [31] [32] [33].

2.7 Summary

By using the simplified matrix representation (SMR) introduced by H. K. Kwan [9][38], the computational time and memory storage can be reduced. The derivation of SMR in section 2.3 is only restricted to 1-D digital networks. It can be extended to 2-D and 3-D digital networks by numbering all the 1-D delay nodes first, then the 2-D delay nodes, and finally 3-D delay nodes. The signal nodes are numbered sequentially according to their orders of computation. The digital network is computable only if the diagonal and the upper triangle of the coefficient matrix $\mathcal{L}_C^t$ in section 2.3.1 of the signal nodes are all zero. In SMR, the diagonal and the upper triangle of $\mathcal{L}$ in (2.8) are all zero for a computable network. Orders of delay nodes in the same dimension need not be in sequence because they store the previous signal values. However, for computer-aided analysis, the multidimensional digital network has to number 1-D delay nodes first, then

2-D delay nodes and finally 3-D delay nodes.

The frequency response, group delay and slope of magnitude response, first order coefficient sensitivity, noise, and impulse response can be computed by using the equations as mentioned in section 2.4. A brief description of digital filter structures was also introduced. Direct form II, parallel form and cascade form were derived from the system transfer function as a ratio of polynomials, an addition of ratios of second-order polynomials and a product of ratios of second-order polynomials, respectively. Lattice or ladder structures and wave digital structures can design a low-sensitivity filter. Finally, systolic structures create low costs and more effective special-purpose hardware for digital filters.

Chapter 3

Programming Language

3.1 Introduction

The programming language is a major concern in developing a Computer-Aided Design (CAD) software. The system environment leads to the selection of a suitable programming language. Graphics capability supported by the programming language is another constraint to develop a Graphical User Interface (GUI) CAD software. Due to limited resources that the system environment is restricted to the IBM-PC platform, the operating system used under the IBM-PC platform can be DOS, Windows, Windows NT or the UNIX environment. We have chosen DOS as the environment for developing the CAD software because of its simplicity.

Graphics capability is now the only criterion on which to choose the programming language. The VGAWAT graphics library was developed by Dr. Wilson of the University of Windsor using the WATFOR-77 programming language. WATFOR-77 is an implementation of American National Standard programming language FORTRAN, ANSI X3.9-1978, commonly referred to as FORTRAN 77 [34]. Besides the common routines that normal graphics library can perform, it also has the ability of mouse handling and 3-D drawing. FORTRAN 77 is selected because previous source codes are programmed using FORTRAN 77; for example, the five different analyses of digital filters as mentioned in chapter 2 were implemented using FORTRAN 77.

3.2 WATFOR-77 Programming Language

WATFOR-77 is based upon some well known FORTRAN language compilers, namely the University of Waterloo's WATFOR, WATFIV-S compilers and the WATFOR-11 compiler [34]. The compiler can be accessed by any students in the University of Windsor. Copying of the compiler is encouraged and is restricted to students and staff of University of Windsor only. The instructions and conditions of copying the compiler are listed in the CAD/CAM center. In the WATFOR-77 compiler package, there are two different editors integrated into its corresponding compiler. The first editor can compile a program in the system where a math co-processor is not installed and the second editor can compile a program in the system where a math co-processor is installed. The advantage of the integration of the editor and compiler together is to allow the execution of the program without leaving the editor. However, it cannot create the executable file and occupies a certain amount of memory. Therefore, we have compiled the program after editing its source code by using 'Watcomp.exe' compiler which is also included in the package. For massive use of graphics routines, the computer with a math co-processor is preferred because it allows faster drawing and repainting of graphics. The 'Watcomp.exe' compiler allows compilation of the program where the system is installed with a math co-processor.

WATFOR-77 has the same programming statements as in FORTRAN 77. The statements supported by WATFOR-77 can be found in language reference [34]. Noted that the syntax models are enclosed by either a singly-edged box which denotes a standard FORTRAN 77 statement or a doubly-edged box which denotes a WATFOR-77 extension to the language. As mentioned in section 3.1, the analysis programs were implemented using FORTRAN 77. We can pre-compile the analysis programs and use the function sub-program EXDOS(CMD) to execute the modules. The function EXDOS(CMD) uses the integer-valued FORK and DOSENV functions located in WATFOR.LIB. It can issue the DOS commands from within an executing FORTRAN application. CMD can be any DOS commands with single quotations or any character variables that store any DOS

commands. For example,

```
INTEGER STATUS, EXDOS
CHARACTER*30 COMMAND
STATUS=EXDOS('DIR *.* > FILE.TXT')
COMMAND='COPY *.FOR B:'
STATUS=EXDOS(COMMAND)
STOP
END
```

3.3 VGAWAT Graphics Library

The VGAWAT graphics library was developed by Dr. Wilson of the Department of Mechanical Engineering of the University of Windsor. The library was written especially for the WATFOR-77 programming language and it is still used by the first year engineering students for the introductory course of the computer graphics programming. It is implemented for PC's with the VGA resolution of 640×480 pixels with 16 colors. There is a total of 58 commands in the library. The capability of the library includes: mouse handling, drawing lines, filling colors, displaying PCX picture files, scrolling the screen, 3-D drawing, capture and re-displaying the graphics windows and many others. Details of each graphics routine can be found in the VGAWAT graphics manual [35].

3.3.1 Initialization

There are two different modes in which VGAWAT can be used, text mode and graphics mode. Text mode can be initialized by using the calling sequence:

```
CALL INIT_TEXT
```

The subroutine initializes the screen to 25 rows by 80 columns, and it clears the whole screen into black color. Most of the commands in the library are designed to be used in graphics mode. Graphics mode can be initialized by using the calling sequence:

```
CALL INIT_GRAPHICS
```

The subroutine initializes the screen to 640 by 480 pixel in 16 colors. The origin (0,0) is

located at the lower left corner, and the upper right corner is at (639,479) as shown in fig. 3.1. In graphics mode, text can be displayed on the screen in 30 rows by 80 columns using the subroutines, GOTO_XY(ICOL,IROW) and WRITE_TEXT (CHR). However, displaying text in graphics mode is more convenient using the subroutines, MOVE_TO(IX,IY) and WRITE_GRAPHICS(CHR). The height and width of each character in pixels are shown in fig. 3.2. The arguments IX and IY, in subroutine MOVE_TO(IX,IY), represent integer variables for horizontal coordinate and vertical coordinate, respectively.

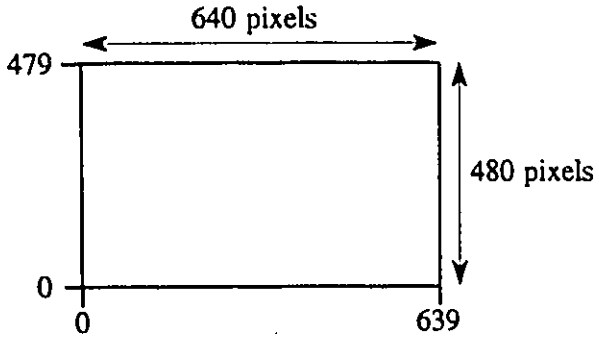


Fig. 3.1 : Screen Coordinates

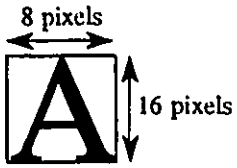


Fig. 3.2 : Dimensions of a Character

The initialization has to be done immediately after the declaration of all the necessary variables in order to utilize the commands in the graphics library.

3.3.2 Mouse Handling

It is assumed that the appropriate mouse driver has been included in the CONFIG.SYS or AUTOEXEC.BAT files when the computer is started. The initialization of the mouse can be used by the calling sequence:

```
CALL INIT_MOUSE
```

The coordinate specifying the mouse location uses the screen coordinate which is (0,0) at the bottom left corner and (639,479) at the top right corner as shown in Fig. 3.1. The mouse icon will be visible by calling the subroutine:

```
CALL MOUSE_ON
```

and invisible by calling the subroutine:

CALL MOUSE_OFF

The current mouse cursor location and the button status can be detected by calling the subroutine:

CALL GET_MOUSE_B(IX,IY,IBUT)

where IX represents the horizontal coordinate and IY represents the vertical coordinate. Since we are only concentrating on graphics mode, IX and IY represent the integer number of pixels. IBUT represents which mouse button to be pressed and Table 3.1 explains the mouse button status with the specified integer number IBUT.

IBUT	Mouse Button Status
0	no button is pressed
1	left button is pressed
2	right button is pressed
3	left and right button are pressed

Table 3.1 : Explanation of IBUT

3.3.3 Creating Frames, Panels and Filling Colors

The frames and panels of the CAD software can be created by using the subroutine:

CALL SET_WINDOW(IX1,IY1,IX2,IY2)

A rectangular graphics window will be displayed on the screen defined by two diagonal corners, (IX1,IY1) and (IX2,IY2), as shown in Fig. 3.3. The coordinates that defined the window are using screen coordinate as shown in Fig. 3.1. Unpredictable results will occur if the boundary of the window is specified beyond the screen coordinate. After the SET_WINDOW subroutine has been called, any subsequent calls of other graphics commands which are related to the coordinate system will be using the window coordinate rather than screen coordinate. The origin will be changed to the lower left

corner of the window as shown in Fig. 3.4.

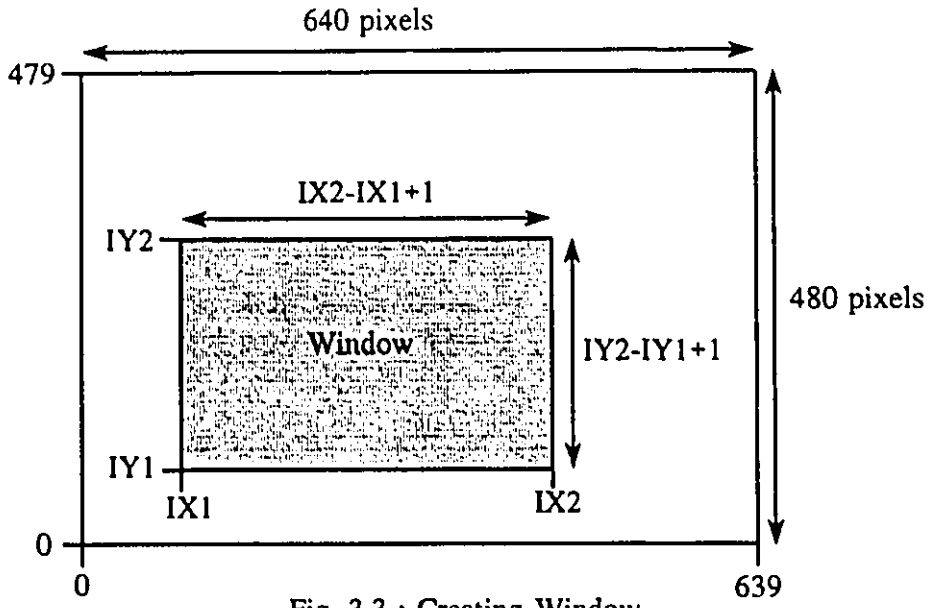


Fig. 3.3 : Creating Window

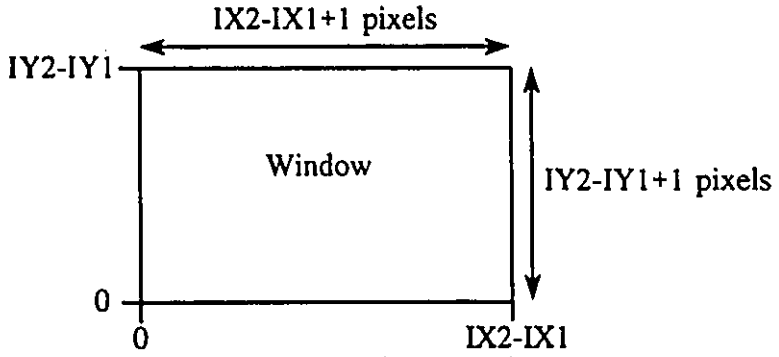


Fig. 3.4 : Window Coordinates

The foreground and background colors of the window can be specified by calling the subroutine:

CALL SET_COLORS(IFG,IBG)

where IFG is the window foreground color and IBG is the window background color. The foreground color relates to the color of text and graphics lines drawn on the window, and the background color relates to the color of the window. Subsequent calls of any routines

relating to colors will change to the colors specified in the subroutine, SET_COLORS(IFG,IBG). Table 3.2 shows the integer number and its colors' representation.

Integer	0	1	2	3	4	5
Color	Black	Blue	Green	Cyan	Red	Magenta
Integer	6	7	8	9	10	11
Color	Brown	White	Grey	Bright Blue	Bright Green	Bright Cyan
Integer	12	13	14	15		
Color	Bright Red	Bright Magenta	Bright Yellow	Bright White		

Table 3.2 : Colors and their Integer Numbers

The window can be colored as shown in the following source codes:

```
.
.
CALL SET_COLORS(14,1)
CALL SET_WINDOW(100,100,300,300)
CALL CLEAR_WINDOW
CALL BORDER_WINDOW
.
.
```

From the source codes, the foreground color is set to yellow and the background color is set to blue. The two diagonal corners of the window are (100,100) and (300,300) in pixels. CLEAR_WINDOW set the window's foreground and background colors into the most current colors, called by SET_COLORS. The window's boundary can be wrapped by the foreground color of the most recent call to SET_COLORS by calling BORDER_WINDOW.

3.3.4 Drawing Lines and Text

After a window is created, 2-D lines can be drawn on the window by using:

CALL MOVE_TO(IX,IY)

and

CALL DRAW_TO(IX,IY)

Text can be drawn on the window by using the subroutine:

CALL WRITE_GRAPHICS(CHR) or CALL WRITE_GRAPHICS('Write text here')

The coordinates for drawing lines and text will depend on the most recent call of SET_WINDOW. In other words, it will be using window coordinates. CHR in WRITE_GRAPHICS represents the character variable. If the text is stored in the character variable, there is no need to use the single quote marks. Otherwise, a single quote is necessary which is similar to the PRINT statement in WATFOR-77. WRITE_GRAPHICS only accepts character strings or character variables. The following source codes show how to draw a line and a string and Fig. 3.5 shows its related screen and window coordinates:

```
      .
      .
C     Create window and set the foreground and background colors of the window.
      CALL SET_COLORS(14,1)
      CALL SET_WINDOW(100,100,400,300)
      CALL CLEAR_WINDOW

C     Drawing lines from (10,10) to (100,100) using window coordinates.
      CALL MOVE_TO(10,10)
      CALL DRAW_TO(100,100)
C
C     Write string starting at (10,200) using window coordinates.
      CALL MOVE_TO(10,150)
      CALL WRITE_GRAPHICS('This is a test')
      .
      .
```

If the coordinates defining a line are beyond the window boundary, the portions of the line beyond the window boundary will not be drawn. However, the whole text will be drawn even some of the portion is beyond the window boundary. The starting point of writing a string is located at the lower left corner of the first character of the string.

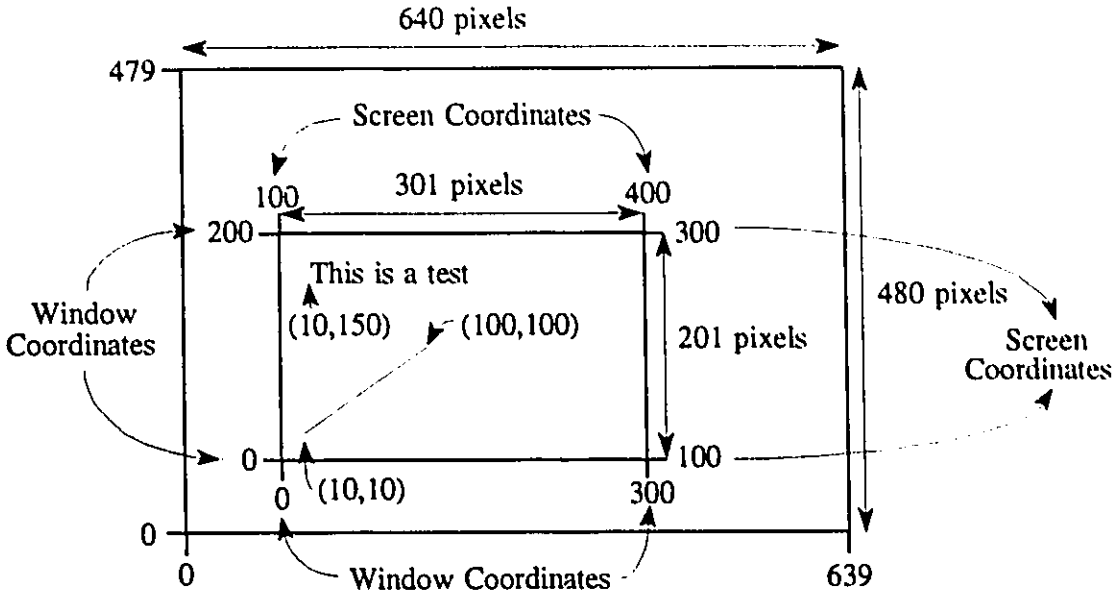


Fig. 3.5 : Drawing Line and String on a Window with Related Screen and Window Coordinates

3.3.5 3-D Drawing

3-D drawing is the most important feature for which we have selected the VGAWAT graphics library to develop a CAD software. Besides the 3-D lines drawing, the library can also perform rotation, translation and scaling by changing the transformation matrix. The matrix is used internally by the library and reset to an identity matrix by using the subroutine:

CALL SET_IDENTITY

In the CAD software, 3-D drawing is only used for plotting the 3-D graph. The projection of 3-D lines should be set to parallel projection. Therefore, the projection angle in subroutine:

CALL SET'_PROJECTION(ANGLE)

should be set to zero for parallel projection.

The origin has to be set in order to perform scaling and rotation. In the rotation operation,

the origin is the point around which the graphics object is rotated. In the scaling operation, it is the point which will remain in the center of the subsequent expansion and contraction. The subroutine is:

CALL SET_ORIGIN(OX,OY,OZ)

The scaling operation can be performed by calling the subroutine:

CALL SET_ZOOM(IXC,IYC,SCALE,IFLAG)

The subroutine can be used to set the scale of the image, specified by SCALE and to move the center of the image to the point of the window, specified by IXC and IYC. If IFLAG equals zero, the zooming is not be used. If IFLAG is another value, the zooming in SET_ZOOM will take effect.

The rotation operation can be performed by the following subroutines:

CALL SET_ROTATE_X(ANGLE)
CALL SET_ROTATE_Y(ANGLE)
CALL SET_ROTATE_Z(ANGLE)

If the real variables OX, OY and OZ in subroutine SET_ORIGIN are set to zero, the operation will be rotated about its own axis as specified in the last character of the subroutines. The real variable, ANGLE, specifies the angle of rotation.

After setting the type of projection, point of origin, rotation angles about x, y and z axis, scaling and zooming, the 3-D lines can be drawn by calling the subroutine:

CALL MOVE_TO_3D(X1,Y1,Z1)

and

CALL DRAW_TO_3D(X2,Y2,Z2)

where the arguments in the subroutines correspond to a real variable or constant Cartesian coordinate values that specify a point in three dimensional space. The coordinates are first

transformed appropriately using the current model origin and the current model matrix, then, use parallel projection to project all the points on the line in three dimensional space onto two dimensional display screen by calling SET_PROJECTION(0.).

3.3.6 XOR Mode

In graphics mode, the lines and text can be drawn with different mode by calling the subroutine:

CALL SET_MODE(IMODE)

There are five different modes that the text and lines can be drawn on the screen. It is specified by the integer variable IMODE. Table 3.3 describes the mode of specific integer variable IMODE.

IMODE	Description
0	Replacement mode
1	Complement mode
2	XOR mode
3	OR mode
4	AND mode

Table 3.3 : Mode Descriptions

In the CAD software, only replacement mode and XOR mode are used. Therefore, other modes will not be discussed but the full descriptions can be found in the VGAWAT graphics manual [35]. When IMODE equals zero, the color of the screen will be replaced by the most recent call of SET_COLORS. This is the most common mode we use in the development of the CAD software. When IMODE equals to two, the color of the pixel on the screen will be replaced by performing the boolean XOR operation on the color specified by most recent call of SET_COLORS. With the color currently at each affected pixel, the colors mapping of using XOR mode is shown in Table 3.5. The integers shown in Table 3.5 represent the colors as listed in Table 3.2. The mouse icon has to be turned

off by calling MOUSE_OFF before XOR mode is being used. In the CAD software, the XOR mode can be used for highlighting the command, drawing busy button and moving the graphics lines without changing the background graphics contents. Detail descriptions will be discussed in chapter 4.

Present Pixel Color	Background and Foreground Color Specified in Most Recent Call to SET_COLOR(IFG,IBG)															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	3	2	5	4	7	6	9	8	11	10	13	12	15	14
2	2	3	0	1	6	7	4	5	10	11	8	9	14	15	12	13
3	3	2	1	0	7	6	5	4	11	10	9	8	15	14	13	12
4	4	5	6	7	0	1	2	3	12	13	14	15	8	9	10	11
5	5	4	7	6	1	0	3	2	13	12	15	14	9	8	11	10
6	6	7	4	5	2	3	0	1	14	15	12	13	10	11	8	9
7	7	6	5	4	3	2	1	0	15	14	13	12	11	1	9	8
8	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
9	9	8	11	10	13	12	15	14	1	0	3	2	5	4	7	6
10	10	11	8	9	14	15	12	13	2	3	0	1	6	7	4	5
11	11	10	9	8	15	14	13	12	3	2	1	0	7	6	5	4
12	12	13	14	15	8	9	10	11	4	5	6	7	0	1	2	3
13	13	12	15	14	9	8	11	10	5	4	7	6	1	0	3	2
14	14	15	12	13	10	11	8	9	6	7	4	5	2	3	0	1
15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 3.5 : XOR Mode Mapping Colors

3.3.7 Keyboard Input and Output

WATFOR-77 uses READ statement to receive input from the user. In VGAWAT graphics library, it uses the READ_GRAPHICS(CHR) to receive the input from the GUI programs. The argument CHR has to be any character variables. Once the command is executed, the screen will display the characters you have typed in the color by the most recent call of SET_COLORS. The string will be stored into CHR when the <Enter> key is pressed. If the input is a numerical value, it is necessary to convert the characters into numerical value. This can be done by using the READ statement in WATFOR-77 as shown:

READ(STRING, '(f14.10)', IOSTAT=IOS) VAL

where STRING is a character variable, VAL is a real variable and IOSTAT is an integer that store the I/O status of the conversion. The command will convert the character variable, STRING, into real variable, VAL, with the format 'f14.10'. If the conversion cannot be done due to the format error, the IOS will not be equal zero. The conversion from numerical value to character string can also be done by using the WRITE statement in WATFOR-77 as shown:

WRITE(STRING, '(f14.10)') VAL

The mouse is a major device to give commands in the GUI software. However, the keystrokes may be useful to order commands for frequent users. The graphics library allows recognizing of a key or combination of keys pressed by calling the subroutines:

CALL GET_KEY(IKEY) or CALL GET_KEY_I(IKEY)

In subroutine GET_KEY, the computer waits for a key or combination of keys to be pressed. In other words, the computer will pause until a valid key is pressed to continue the execution. In subroutine GET_KEY_I, it will not interrupt the execution of the program, the computer will only determine whether or not a valid key is pressed in order to perform suitable command by using IF statement in WATFOR-77. This action is similar to detect mouse button status as mentioned in section 3.3.2. If a normal ASCII key is pressed, the argument IKEY will return a normal ASCII code. Otherwise, IKEY will return a unique code which is determined by the graphics library. Therefore, it is

necessary to write a simple program using subroutine GET_KEY_I to obtain integer numbers for those which are not normal ASCII keys.

3.4 Extended Graphics Routines

It is necessary to use the routines in the VGAWAT graphics library to develop more complicated graphics contents and commands; such as drawing the graphical representation of digital elements on the screen, changing the shape of the mouse icon and moving it along the screen without affecting the background graphics contents, snapping digital elements onto grids, changing the position and the size of a signal flow graph.

3.4.1 Graphical Representation of Digital Elements

An adder can be drawn using the following subroutine, where *ixx* and *iyy* represent the centre of an adder, *irad* represents the angle of rotation, and *iadder* represents the type of adder. There are four different angle of rotations; such as 0, 90, 180 and 270 degrees. There are three different orientations of a two inputs and one output adder when the input lines and output line are restricted to draw vertically or horizontally. The circle of an adder is drawn by calculating five points in one quadrant using the circle equation and duplicate those five points into the remaining three quadrants, then join all the points with straight lines. The beginning points and end points of input lines and output line are first rotated using the subroutines: hori_out, hori_in, vert_up and vert_down, then input lines and output line is drawn by joining beginning point and end point together.

```
      subroutine draw_circle(ixx,iyy,irad,iadder)
      common /zoom/ izoom, ispacex,ispacey
      integer x1,y1,x2,y2
      real rad1,rad2,r,pi
      pi=3.14159
c      DRAWING CIRCLE
      do i=0,80,20
       r=(10*izoom)/20
       rad1=i/180.*pi
       x1=r*cos(rad1)
       y1=r*sin(rad1)
```

```
      rad2=(i+20)/180.*pi
      x2=r*cos(rad2)
      y2=r*sin(rad2)
      call move_to(ixx+x1,iyy+y1)
      call draw_to(ixx+x2,iyy+y2)
      call move_to(ixx-x1,iyy+y1)
      call draw_to(ixx-x2,iyy+y2)
      call move_to(ixx-x1,iyy-y1)
      call draw_to(ixx-x2,iyy-y2)
      call move_to(ixx+x1,iyy-y1)
      call draw_to(ixx+x2,iyy-y2)
     end do
c      ROTATE INPUT LINES AND OUTPUT LINE
     if(irad.ge.0.and.iadder.eq.1)then
       call hori_out(irad,isx1,isy1,isx2,isy2,isx5,isy5,isx6,
     *   isy6,isx9,isy9)
       call hori_in(irad,isx3,isy3,isx4,isy4,isx7,isy7,isx8,isy8,
     * isx10,isy10)
       call vert_up(irad,isx11,isy11,isx12,isy12,isx13,isy13,
     * isx14,isy14,isx15,isy15)
     else if(irad.ge.0.and.iadder.eq.2)then
       call hori_out(irad,isx1,isy1,isx2,isy2,isx5,isy5,isx6,
     *   isy6,isx9,isy9)
       call vert_down(irad,isx3,isy3,isx4,isy4,isx7,isy7,isx8,isy8,
     * isx10,isy10)
       call vert_up(irad,isx11,isy11,isx12,isy12,isx13,isy13,
     * isx14,isy14,isx15,isy15)
     else if(irad.ge.0.and.iadder.eq.3)then
       call hori_out(irad,isx1,isy1,isx2,isy2,isx5,isy5,isx6,
     *   isy6,isx9,isy9)
       call hori_in(irad,isx3,isy3,isx4,isy4,isx7,isy7,isx8,isy8,
     * isx10,isy10)
       call vert_down(irad,isx11,isy11,isx12,isy12,isx13,isy13,
     * isx14,isy14,isx15,isy15)
     end if
c      DRAW INPUT LINES AND OUTPUT LINE
     iz6=(6*izoom)/20
     call move_to(ixx+isx1,iyy+isy1)
     call draw_to(ixx+isx2,iyy+isy2)
     call move_to(ixx+isx3,iyy+isy3)
     call draw_to(ixx+isx4,iyy+isy4)
     call move_to(ixx+isx5,iyy+isy5)
     call draw_to(ixx+isx6,iyy+isy6)
     call move_to(ixx+isx5,iyy+isy5)
```

```
call draw_to(ixx+isx9,iyy+isy9)
call move_to(ixx+isx7,iyy+isy7)
call draw_to(ixx+isx8,iyy+isy8)
call move_to(ixx+isx7,iyy+isy7)
call draw_to(ixx+isx10,iyy+isy10)
call move_to(ixx+isx11,iyy+isy11)
call draw_to(ixx+isx12,iyy+isy12)
call move_to(ixx+isx13,iyy+isy13)
call draw_to(ixx+isx14,iyy+isy14)
call move_to(ixx+isx13,iyy+isy13)
call draw_to(ixx+isx15,iyy+isy15)
call move_to(ixx,iyy-iz6)
call draw_to(ixx,iyy+iz6)
call move_to(ixx+iz6,iyy)
call draw_to(ixx-iz6,iyy)
return
end
```

A delay can be drawn using the following subroutine, where *ixx* and *iyy* represent the centre of a delay, *irad* represents the angle of rotation, *idelay* represents the type of delay elements. There are three kinds of delay elements representing three different dimensions in digital networks. When *idelay* equals to one, delay symbol of first dimension is drawn. When *idelay* equals to two, delay symbol of second dimension is drawn. When *idelay* equals to three, delay symbol of third dimension is drawn. The subroutine *ro_pt* is used to rotate the beginning points and end points of the input line and output line, where the last two arguments in *ro_pt* return the point after rotation.

```
subroutine draw_delay(ixx,iyy,irad,idelay)
 common /zoom/ izoom, ispacex,ispacey
 iz1=(10*izoom)/20
 call move_to(ixx+iz1,iyy+iz1)
 call draw_to(ixx+iz1,iyy-iz1)
 call draw_to(ixx-iz1,iyy-iz1)
 call draw_to(ixx-iz1,iyy+iz1)
 call draw_to(ixx+iz1,iyy+iz1)
 if(irad.ge.0)then
   call ro_pt(11,0,irad,isx1,isy1)
   call ro_pt(20,0,irad,isx2,isy2)
   call ro_pt(19,0,irad,isx5,isy5)
   call ro_pt(13,3,irad,isx6,isy6)
   call ro_pt(13,-3,irad,isx9,isy9)
```

```
        call ro_pt(-11,0,irad,isx3,isy3)
        call ro_pt(-20,0,irad,isx4,isy4)
        call ro_pt(-12,0,irad,isx7,isy7)
        call ro_pt(-18,3,irad,isx8,isy8)
        call ro_pt(-18,-3,irad,isx10,isy10)
      end if
      call move_to(ixx+isx1,iyy+isy1)
      call draw_to(ixx+isx2,iyy+isy2)
      call move_to(ixx+isx3,iyy+isy3)
      call draw_to(ixx+isx4,iyy+isy4)
      call move_to(ixx+isx5,iyy+isy5)
      call draw_to(ixx+isx6,iyy+isy6)
      call move_to(ixx+isx5,iyy+isy5)
      call draw_to(ixx+isx9,iyy+isy9)
      call move_to(ixx+isx7,iyy+isy7)
      call draw_to(ixx+isx8,iyy+isy8)
      call move_to(ixx+isx7,iyy+isy7)
      call draw_to(ixx+isx10,iyy+isy10)
c     DRAW THE NON-LINEAR FUNCTION
      iz2=(2*izoom)/20
      iz3=(3*izoom)/20
      iz4=(4*izoom)/20
      iz6=(6*izoom)/20
      iz7=(7*izoom)/20
      iz8=(8*izoom)/20
      call move_to(ixx-iz7,iyy+iz4)
      call draw_to(ixx+iz3,iyy+iz4)
      call draw_to(ixx-iz7,iyy-iz7)
      if(idelay.eq.1)then
        call draw_to(ixx+iz3,iyy-iz7)
        call move_to(ixx+iz6,iyy-iz4)
        call draw_to(ixx+iz6,iyy-iz8)
      else if(idelay.eq.2)then
        call draw_to(ixx+iz2,iyy-iz7)
        call move_to(ixx+iz4,iyy-iz4)
        call draw_to(ixx+iz8,iyy-iz4)
        call draw_to(ixx+iz8,iyy-iz6)
        call draw_to(ixx+iz4,iyy-iz6)
        call draw_to(ixx+iz4,iyy-iz8)
        call draw_to(ixx+iz8,iyy-iz8)
      else if(idelay.eq.3)then
        call draw_to(ixx+iz2,iyy-iz7)
        call move_to(ixx+iz4,iyy-iz4)
        call draw_to(ixx+iz8,iyy-iz4)
```

```
  call draw_to(ixx+iz8,iyy-iz6)
  call draw_to(ixx+iz4,iyy-iz6)
  call move_to(ixx+iz8,iyy-iz6)
  call draw_to(ixx+iz8,iyy-iz8)
  call draw_to(ixx+iz4,iyy-iz8)
end if
call move_to(ixx+iz4,iyy+iz6)
call draw_to(ixx+iz6,iyy+iz6)
call move_to(ixx+iz8,iyy+iz8)
call draw_to(ixx+iz8,iyy+iz4)
return
end
```

A multiplier can be drawn using the following subroutine, where *ixx* and *iyy* represent the centre of a multiplier, and *irad* represents the angle of rotation.

```
subroutine draw_mult(ixx,iyy,irad)
common /zoom/ izoom, ispacex,ispacey
if(irad.ge.0)then
call ro_pt(5,0,irad,isx20,isy20)
call ro_pt(-5,5,irad,isx21,isy21)
call ro_pt(-5,-5,irad,isx22,isy22)
call ro_pt(20,0,irad,isx23,isy23)
call ro_pt(-5,0,irad,isx24,isy24)
call ro_pt(-20,0,irad,isx25,isy25)
end if
call move_to(ixx+isx20,iyy+isy20)
call draw_to(ixx+isx21,iyy+isy21)
call draw_to(ixx+isx22,iyy+isy22)
call draw_to(ixx+isx20,iyy+isy20)
call move_to(ixx+isx20,iyy+isy20)
call draw_to(ixx+isx23,iyy+isy23)
call move_to(ixx+isx24,iyy+isy24)
call draw_to(ixx+isx25,iyy+isy25)
return
end
```

An input pin can be drawn using the following subroutine, where *ix* and *iy* represent the centre of an input pin.

```
subroutine draw_in(ix,iy)
common /zoom/ izoom,ispacex,ispacey
```

```
      iz5=5*izoom/20
      iz6=6*izoom/20
      iz9=9*izoom/20
      iz18=18*izoom/20
      iz24=24*izoom/20
      iz27=27*izoom/20
      iz40=40*izoom/20
      call move_to(ix,iy)
      call draw_to(ix-iz6,iy+iz9)
      call draw_to(ix-iz40,iy+iz9)
      call draw_to(ix-iz40,iy-iz9)
      call draw_to(ix-iz6,iy-iz9)
      call draw_to(ix,iy)
c     DRAWING A WORD 'IN'
      call move_to(ix-iz27,iy+iz5)
      call draw_to(ix-iz27,iy-iz5)
      call move_to(ix-iz24,iy-iz5)
      call draw_to(ix-iz24,iy+iz5)
      call draw_to(ix-iz18,iy-iz5)
      call draw_to(ix-iz18,iy+iz5)
      return
      end
```

An output pin can be drawn using the following subroutine, where *ix* and *iy* represent the centre of an output pin.

```
c     Draw output pin
c     ***************
      subroutine draw_out(ix,iy)
      common /zoom/ izoom,ispacex,ispacey
      iz5=5*izoom/20
      iz6=6*izoom/20
      iz9=9*izoom/20
      iz12=12*izoom/20
      iz17=17*izoom/20
      iz20=20*izoom/20
      iz25=25*izoom/20
      iz28=28*izoom/20
      iz31=31*izoom/20
      iz34=34*izoom/20
      iz40=40*izoom/20
      call move_to(ix,iy)
      call draw_to(ix+iz6,iy+iz9)
```

```
      call draw_to(ix+iz40,iy+iz9)
      call draw_to(ix+iz40,iy-iz9)
      call draw_to(ix+iz6,iy-iz9)
      call draw_to(ix,iy)
c       DRAWING A WORD 'OUT'
      call move_to(ix+iz12,iy+iz5)
      call draw_to(ix+iz12,iy-iz5)
      call draw_to(ix+iz17,iy-iz5)
      call draw_to(ix+iz17,iy+iz5)
      call draw_to(ix+iz12,iy+iz5)
      call move_to(ix+iz20,iy+iz5)
      call draw_to(ix+iz20,iy-iz5)
      call draw_to(ix+iz25,iy-iz5)
      call draw_to(ix+iz25,iy+iz5)
      call move_to(ix+iz28,iy+iz5)
      call draw_to(ix+iz34,iy+iz5)
      call move_to(ix+iz31,iy+iz5)
      call draw_to(ix+iz31,iy-iz5)
      return
      end
```

A point can be rotated using the following subroutine, where *irx* and *iry* represent the original coordinate of a point, *irad* represents the angle of rotation, and *isx* and *isy* represent the coordinate of a point after rotation.

```
      subroutine ro_pt(irx,iry,irad,isx,isy)
        common /zoom/ izoom, ispacex,ispacey
        rad=irad/180.*3.141592654
        tc=cos(rad)
        ts=sin(rad)
        if(tc.gt.0.95.and.tc.le.1.0)then
          tc=1.0
        end if
        if(tc.ge.-0.05.and.tc.lt.0.05)then
          tc=0.0
        end if
        if(ts.gt.0.95.and.ts.le.1.0)then
          ts=1.0
        end if
        if(ts.ge.-0.05.and.ts.lt.0.05)then
          ts=0.0
        end if
```

```
  isx=((irx*tc-iry*ts)*izoom)/20
  isy=((irx*ts+iry*tc)*izoom)/20
return
end
```

3.4.2 Changing the Mouse Icon

The default mouse icon is an arrow with white color. It can be changed to any shapes when the default mouse icon is turned off by calling MOUSE_OFF. The following source codes show how to change the mouse icon into digital element and move it without affecting the background graphics contents. From the source codes, *ixx* and *iyy* represent the present mouse location, *ixold* and *iyold* represent the previous mouse location, *ibg* and *ifg* represent the background and foreground colors respectively, *and ii* represents a flag. When *ii* equals 1000, element was placed on the screen in that location in the previous moment (*ixx=ixold* and *iyy=iyold*). The subroutine *align* is used to snap the element onto the grid and it will be discussed in the next section.

From the source codes, the mouse is first turned off and the drawing mode is set to XOR mode. If an adder was not placed on the screen in the previous moment, draw an adder at the previous location. The previous mouse icon will be drawn with the original color that is covered by the icon. Then, a mouse icon will be drawn to a present position defined by *ix* and *iy*. The present position *ix* and *iy* will be stored to the previous position *ixold* and *iyold,* and wait for the further movement of the mouse.

```
Loop
  .
  .
call get_mouse_b(ixx,iyy,ibut)
  .
    .
call align(ixx,iyy)
if(ixold.ne.ixx.or.iyold.ne.iyy)then
  if(ibg.eq.15)then
    call set_colors(ifg+1,0)
  else
   call set_colors(15-ibg,0)
```

```
  end if
   call set_window(84,37,619,426)
   call mouse_off
   call set_mode(2)
   if(ii.ne.1000)then
    call draw_circle(ixold-84,426-iyold-37,0,1)
   end if
   ii=0
   call draw_circle(ixx-84,426-iyy-37,0,1)
   call set_mode(0)
 end if
ixold=ixx
iyold=iyy
  .
  .
End Loop
```

3.4.3 Coordinates System of Placing Elements

Since, the total width and height of a signal flow graph may be larger than the maximum width and height of the screen, a new coordinate system for the record of the position of elements and lines has to be created. Fig. 3.6 shows the new coordinates system to draw a signal flow graph. The origin is set to the upper left corner and the maximum coordinates is set to the lower right corner, IXMAX and IYMAX.

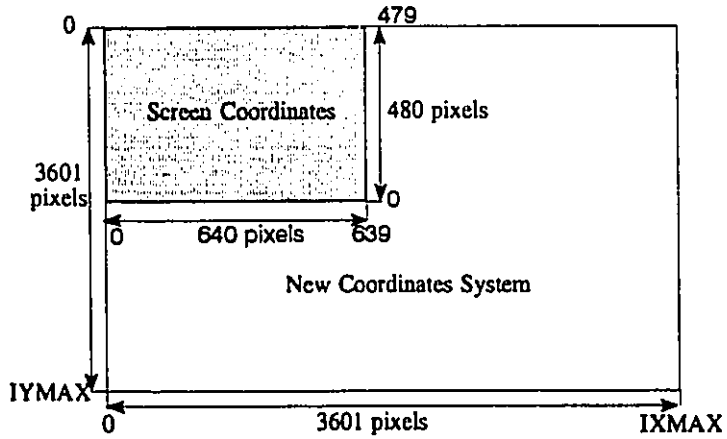


Fig. 3.6 : New Coordinates System

In order to draw a signal flow graph on the screen to be more neatly, the position of placing digital elements is designed to snap into the grids. The width and height of the grids are both set to 20 pixels. The movement of the mouse can be snapped into the grids by using the following source codes:

```
subroutine align(ixx,iyy)
common /zoom/ izoom,ispacex,ispacey
if(mod(ixx-84,izoom).ne.0)then
  if(mod(ixx-84,izoom).le.izoom/2)then
    ixx=ixx-mod(ixx-84,izoom)
  else
    ixx=ixx-mod(ixx-84,izoom)+izoom
  end if
end if
iyy=426-iyy
if(mod(iyy,izoom).ne.0)then
  if(mod(iyy,izoom).le.izoom/2)then
    iyy=iyy-mod(iyy,izoom)
  else
    iyy=iyy-mod(iyy,izoom)+izoom
  end if
end if
return
end
```

where ixx and iyy represent the current position of the mouse, and *izoom* represents both vertical and horizontal distances between the grids. The variable *izoom* is used to specify the zoom range of the graph. If *izoom* is less than the default grid distance (20 pixels), the graph size will be decreased. If *izoom* is larger than the default grid distance, the graph size will be increased. Besides, *izoom* also control the size of each elements and they are drawn on the screen as mentioned in section 3.4.1.

The position of an element is stored in the arrays using the following source codes:

```
inx=(ix-84)*20/izoom+ipanx*20+84
iny=iy*20/izoom+ipany*20
```

where *ix* and *iy* represent the centre of an element (mouse location) after it has been

snapped into the grid, *ipanx* and *ipany* represent the movement of the graph in the horizontal and vertical directions under the new coordinates system, respectively. According to the new coordinates system, the maximum *ipanx* and *ipany* value equal to *IXMAX/20* and *IYMAX/20*, respectively. The position of each element is recorded by using the default grid distance (20 pixels). Elements can be placed on the screen from the arrays using the following source codes:

```
ixx=((nodex(i)-ipanx*20-84)*izoom)/20
iyy=426-((nodey(i)-ipany*20)*izoom)/20-37
```

where *nodex(i)* and *nodey(i)* represent the arrays that store the horizontal and vertical distance according to the new coordinates system, *ixx* and *iyy* represent the horizontal and vertical distance (pixels) according to the screen coordinates.

3.5 Summary

The CAD software for design and analysis of multidimensional digital networks will be developed on and for the IBM-PC computer. DOS has been chosen to be used as the operating system because of its simplicity. The programming language is WATFOR-77 which is an implementation of FORTRAN-77. FORTRAN is selected because the original source codes for implementing the five different analyses in 1-D, 2-D and 3-D digital networks are using FORTRAN. With an external function subprogram, WATFOR-77 allows pre-compiling the programs and issuing DOS command, such as execute 'EXE' file from within an executing FORTRAN application. This method helps save memory by preventing every analysis program to be included into the source codes of the CAD software. Details will be discussed in chapter 5.

VGAWAT graphics library is selected to the development of the CAD software. It was written especially for WATFOR-77. The capability of VGAWAT graphics library is enough to develop Graphical User Interface (GUI) software. Section 3.3 only discussed the basic features of the graphics library. It is important to utilize those features to develop a more effective and more 'user friendly' software. We have written a signal flow

graph schematic drawing software and a multidimensional digital networks analysis software using WATFOR-77 and VGAWAT graphics library and will be discussed in chapter 4 and chapter 5.

Finally, the WATFOR-77 compiler and the VGAWAT graphics library can be worked on any IBM-PC computer with the following specifications:

1. Computer Type : IBM-PC AT or compatible.

2. System : 80386 with math co-processor or higher.

3. Video Adapter : Video Graphics Array or higher.

4. Keyboard Type : IBM Enhanced (101- or 102- key) keyboard.

5. Mouse Type : Microsoft compatible mouse.

Chapter 4

Signal Flow Graph Schematic Drawing Software

4.1 Introduction

The software is developed using the WATFOR-77 programming language and the VGAWAT graphics library. It allows drawing of 1-D, 2-D and 3-D digital networks in the form of signal flow graphs. Signal flow graph is constructed by using three basic elements, such as adder, delay and multiplier. It also has the ability to extract signal flow graphs into a netlist. The netlist includes the necessary information to represent the signal flow graphs in numerical values, such as node numbers and branch transmittance, called simplified matrix representation. It also includes the total number of delay nodes in 1-D, 2-D and 3-D digital network, the total number of signal nodes, the total number of non-zero elements in the matrix, signal node number where input is injected and signal node number where output is extracted.

The design of the software is able to construct a signal flow graph in a simple and convenient way. The netlist should be extracted automatically and the order of the nodes should be renumbered in order to obtain a computable network.

4.2 General Software Mechanisms

There are two kinds of devices to give commands to the software; they are mouse and

keyboard. The mouse location and the mouse button status can be detected by calling the subroutine GET_MOUSE_B(IX,IY,IBUT), which was mentioned in chapter 3. Therefore, the commands of the software can be placed on the screen and use the mouse to click on the commands in order to perform operations. Some of the commands may not be frequently used. It is convenient to create a main menu into several categories and use pull-down menus to include the related commands. Another mechanism is using buttons to represent commands which are frequently used. All the commands in the pull-down menus or buttons can be performed by pressing the mouse button when the mouse location is within the commands in the pull-down menus or the commands in the buttons.

The flow chart in Fig. 4.1 shows how the operation of the commands can be performed by using the mouse. The program uses the infinite loop in WATFOR-77, LOOP and END LOOP, to get the current mouse location and mouse button status; then it uses IF and ELSE IF statements to search for the operations. If the mouse location is within the button boundary and the appropriate mouse button is pressed, the operation specified by the button will be performed. If the mouse location is within the category name of the main menu and the appropriate mouse button is pressed, a pull-down menu will appear on the screen. It will search for all the commands until the loop starts again. The integer variable IPBUT stored the mouse button status before it returns to get a new mouse button status. If IPBUT is not equal to IBUT, the operation will be executed, otherwise, it will return and start the loop again. This can prevent the same operation performed repeatedly when the mouse button is clicked once only. When the mouse button is pressed and released once, the infinite main loop may be running for several times and leads to the GET_MOUSE_B running for several times. This might happened especially for a simple operation.

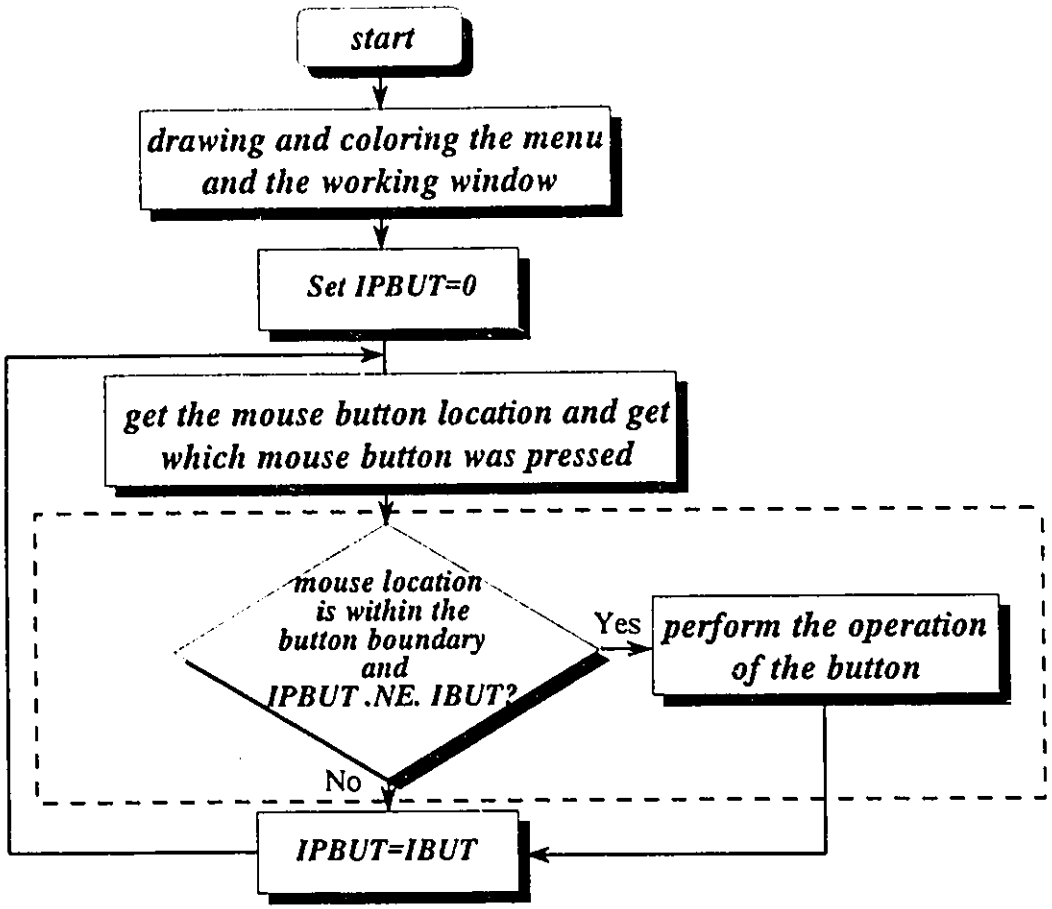


Fig. 4.1 : Flow Chart of the Main Control Loop

4.2.1 Pull-down Menu

The advantage of the pull-down menu is to save more space of the screen in order to maximize the working window. The working window is a window in which the signal flow graph is drawn. The pull-down menu uses the subroutines: SET_WINDOW, SET_COLORS and WRITE_GRAPHICS in VGAWAT graphics library to specify location, coloring menu window and text, and writing names of commands by using MOVE_TO to specify the location of the names, respectively. When the pull-down menu appears on the screen, the program will search for the mouse location and the mouse button status to determine whether it is within the boundary of the specific command. Then, the command name will be highlighted and the area hidden by the pull-down menu

will be repainted. Finally, the operation of the command will be executed.

The subroutine XOR mode in VGAWAT can be used to highlight the command name as shown in the following source codes:

```
.
.
CALL MOUSE_OFF
CALL SET_MODE(2)
CALL SET_COLORS(0,15)
CALL SET_WINDOW(IX1,IY1,IX2,IY2)
CALL CLEAR_WINDOW
CALL SET_MODE(0)
CALL MOUSE_ON
.
.
```

Mouse has to be turned off when using XOR mode. It is assumed that the present background is black (0) and the present foreground color is white (15). The most recent call of SET_COLORS specified the background color of white (15) and the foreground color of black (0). The source codes only show filling of the window with background color because the foreground color in SET_COLORS is not used. Using Table 3.5, the present background color, black (0), performs the boolean XOR with the background color, white (15), in SET_COLOR that give the result color of white (15). Same as the present foreground color, obviously, it is the color of the command name, the boolean XOR of present foreground color, white (15), with the background color in SET_COLOR, white (15), that give the color of black (0). Fig. 4.2a-b shows an example of the pull-down menu before and after one of the commands was selected.

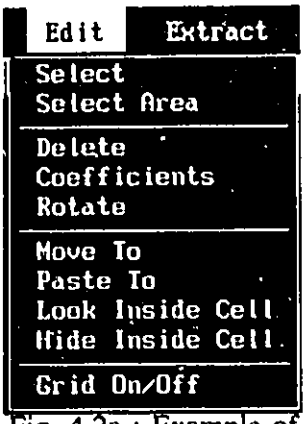


Fig. 4.2a : Example of Pull-Down Menu

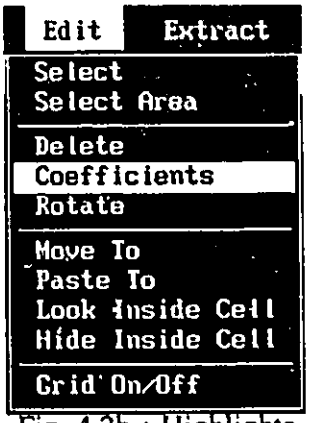


Fig. 4.2b : Highlights of Command

The graphics contents covered by the pull-down menu can be stored by using the subroutine:

CALL SAVE_BLOCK(NAME)

where the argument is any character variable or character string, which is the filename that store the graphics contents. The area to be stored will be specified by SET_WINDOW immediately before the subroutine SAVE_BLOCK. The hidden graphics contents, covered by the pull-down menu, can be redisplayed by calling the subroutine:

CALL DISPLAY_BLOCK(NAME)

The image can be displayed in the current graphics window with the lower left corner of the image at the location defined by the most recent call to the MOVE_TO subroutine. The simple method is to define the window with the same coordinates as defined in SAVE_BLOCK. Then, move the mouse cursor to the origin of the window by calling the subroutine MOVE_TO(0,0). The block of the previous graphics contents will exactly cover the area of the pull-down menu. This may create a faster repaint, especially for large area and complicate graphics contents.

When other pull-down menu or button is selected, or the right mouse button is pressed, the graphics contents covered by the current pull-down menu should be redisplayed.

4.2.2 Button Shadow, Pressed Mechanism and Highlight Mechanism

Another technique to issue commands may be using buttons. Buttons are desirable for those commands that are frequently used. The command specified on the button can be executed by searching the mouse location whether it is within the button boundary and the button pressed status. Normally, the command will be executed when the left mouse button is pressed.

The button can be created by drawing shadow around the boundary. The background color of the button will be using grey color, and the shadow can be drawn by using dark grey and white lines to wrap around the boundary of the button. Fig. 4.3a shows the lines and the appropriate coordinates in pixels to form the shadow of the button. All the buttons will be created by using the method as shown in Fig. 4.3a. The button will be displayed on the screen as shown in Fig. 4.3b.

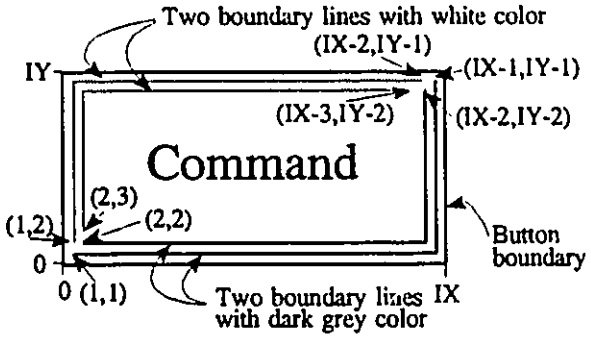


Fig. 4.3a : Creation of Button Shadow

Fig. 4.3b : Real Display of Button

The button pressed mechanism includes the animation of the button when the button is pressed. This animation will be used for the command which is executed once when the button is pressed. The button shadow will first be redrawn by changing the color of the white shadow into dark grey shadow and the dark grey shadow will be changed to the button color, grey. Then, the command name on the button will shift right by two pixels and shift down by one pixel. The press-down scene of the button will be displayed on the screen for a while by using a dummy DO loop. Finally, the original, unpressed, button will be redrawn. This will complete the whole animation when the button is pressed. Fig.

4.4 explains the animation of the button pressed mechanism.

Fig. 4.4 : Button Pressed Animation

The delay loop will be using the dummy DO loop as shown:

```
DO I=1,5000
  Y=COS(1.5*I)
END DO
```

Since there is no function subprogram for the delay in both WATFOR 77 and VGAWAT, a delay function has to be developed. The disadvantage of using a customized delay function is the delay time depending on the type of computer. It is hard to exactly calculate the delay time in the above source codes. However, the exact delay time is not really important for the button animation.

The final mechanism related to the button is the highlight of button. This is useful when the button is selected and the command is still active or executing. This kind of button may be called busy button. Same as highlight of pull-down menu, the busy button can be created by using XOR mode. If the most recent call of SET_COLORS has the background color of white (15), all the pixels on the button will be performing the boolean XOR with the white color. The foreground color in SET_COLOR is also not used. Fig 4.5 shows the busy button where the button color is changed to dark grey, and the shadow is changed from white to black and from dark grey to grey, and the foreground color of name or graphical representation is changed to white.

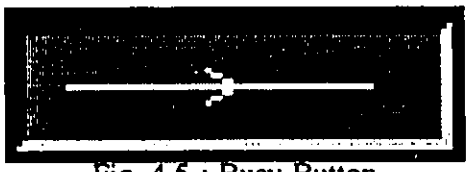


Fig. 4.5 : Busy Button

4.2.3 Digital Elements Icon Bar

Three basic elements of constructing a signal flow graph include: adder, delay and multiplier. An adder has a graphical representation as shown in Fig. 2.2a with two inputs and one output. If the connecting lines between elements are restricted to horizontal lines and vertical lines, there will be three different orientation of two inputs in order to connect the connecting lines in any horizontal and vertical directions. Since the software is capable of constructing 1-D, 2-D and 3-D digital networks, the graphical representation of 1-D, 2-D and 3-D will be using Fig. 2.3a-c, respectively. The external input will be represented by an input pin and the network output will be represented by an output pin.

Since the commands for drawing of the digital elements are the frequently used commands, buttons will be the suitable devices to issue drawing commands. Besides, the graphical representation of the elements can be drawn on the button, it is also necessary to group the buttons of the elements together for easy access. The grouping of the buttons of digital elements is called the digital elements icon bar and is shown in Fig. 4.6.

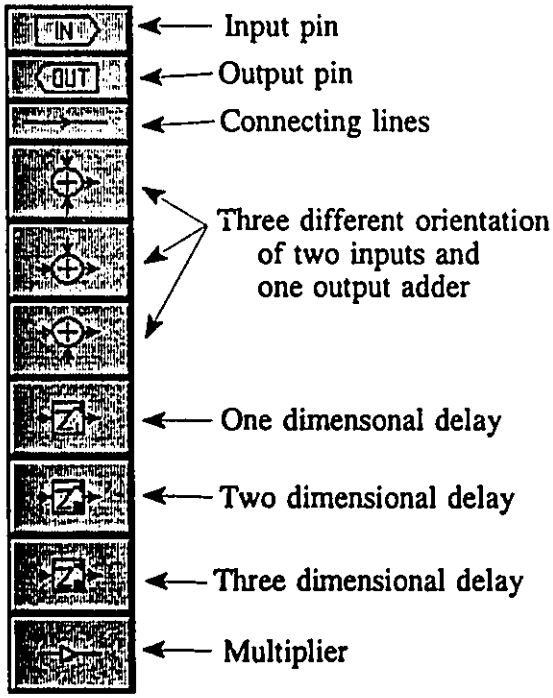


Fig. 4.6 : Digital Elements Icon Bar

The best location of the icon bar is to locate either at the left or right side of the screen vertically in order to maximize the working window. However, left side of the screen is preferred because right side of the screen is reserved for the vertical scrollbar.

4.2.4 Vertical and Horizontal Scrollbar Mechanism

Most of the design of digital networks is constructed with a large amount of digital elements so that the space of the working window may not enough to hold the whole structure of the digital network. Assume the signal flow graph is drawn on a paper with a size much larger than the working window. The working window is a view window placed on the top of the paper so that only a portion of the paper is shown. It is necessary to create vertical and horizontal scrollbars to scroll to anywhere of the paper in order to manipulate digital elements on the paper. Fig. 4.7 shows the view window, vertical scrollbar and horizontal scrollbar of the software.

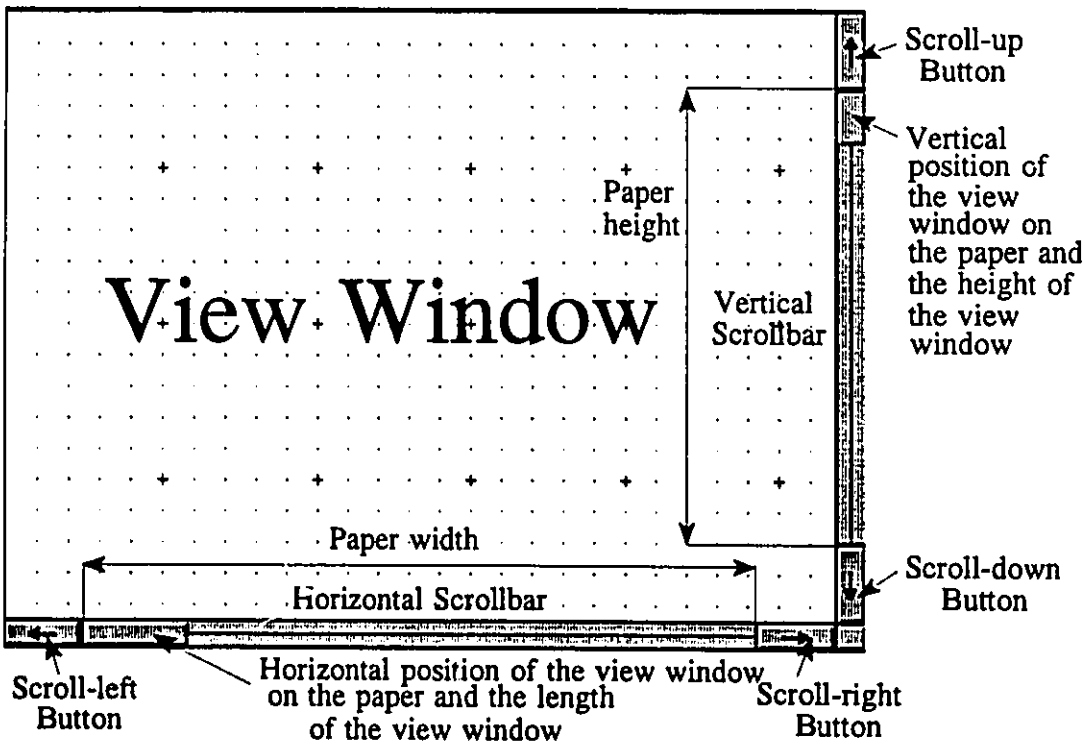


Fig. 4.7 : View Window, Horizontal Scrollbar and Vertical Scrollbar

The vertical scrollbar has two buttons which are used to scroll-up or scroll-down of the

paper. When the scroll-up button is pressed, the paper will move downward in order to show the top portion of the paper. When the scroll-down button is pressed, the paper will move upward. The vertical position bar shows the position of the view window on the paper and the height of the view window. It can be dragged to any vertical position of the paper by holding the left mouse button and moving the mouse to the desired position along the track. The horizontal scroll bar has the same functions of the vertical scrollbar but in horizontal direction. Scroll-left will move the paper to the right and scroll-right will move the paper in opposite direction. The horizontal position bar can also be dragged along the track.

The buttons of the scrollbars have different mechanism with other buttons in the software. They allow the software to continue the execution when the left mouse button is holding. The dragging of the position bars can be done by checking the mouse button status. If the mouse location is within buttons and the left mouse button is pressed and hold, both current mouse button status IBUT and previous mouse button status IPBUT are equal to one. The software will wait until mouse button is released, IBUT equals to zero, and IPBUT equals to one. Then, the paper will move to the position of the paper where the position bars located along the tracks.

The area of the paper is extended to 3601 by 3601 pixels, and the maximum width or height of elements are 60 pixels. The scrolling will be too slow if the paper is moving pixel by pixel. Therefore, the paper will move 140 pixels in either vertical or horizontal direction when the scroll button is pressed once. The vertical coordinate of the paper will be different from the vertical screen coordinate in which it starts from the top, 0, to the bottom, 3600. The edge of the paper will be displayed when the view window is scrolled to the edge. Elements cannot be drawn beyond the boundary of the paper.

4.3 Design of a Signal Flow Graph

The software mechanisms, mentioned in section 4.2, can now be integrated to create a schematic drawing of signal flow graph software. The software will appear on the screen

as shown in Fig. 4.8. The main menu is located at the top of the software where every commands can be found in the pull-down under the category names. The category name ***File*** includes the commands of handling files of graph, files of cell and termination of the software. ***Edit*** includes the commands of the modification of SFG and the command of disable or enable the grid. ***Extract*** includes the commands of creating the netlist. ***Window*** includes the commands of changing view window appearance. Finally, ***Help*** displays the detail description of all the commands in the software.

The ***Graph/Cell Display Bar*** displays a graph or cell is being drawn. The ***Filename Display Bar*** displays the full path name of the graph/cell only when the graph/cell has been saved previously, otherwise, 'no_name' will be displayed on the bar. It also has two quick access buttons which are used to redraw the view window and enable/disable the grids. The remaining mechanism of the software have been discussed in the previous section.

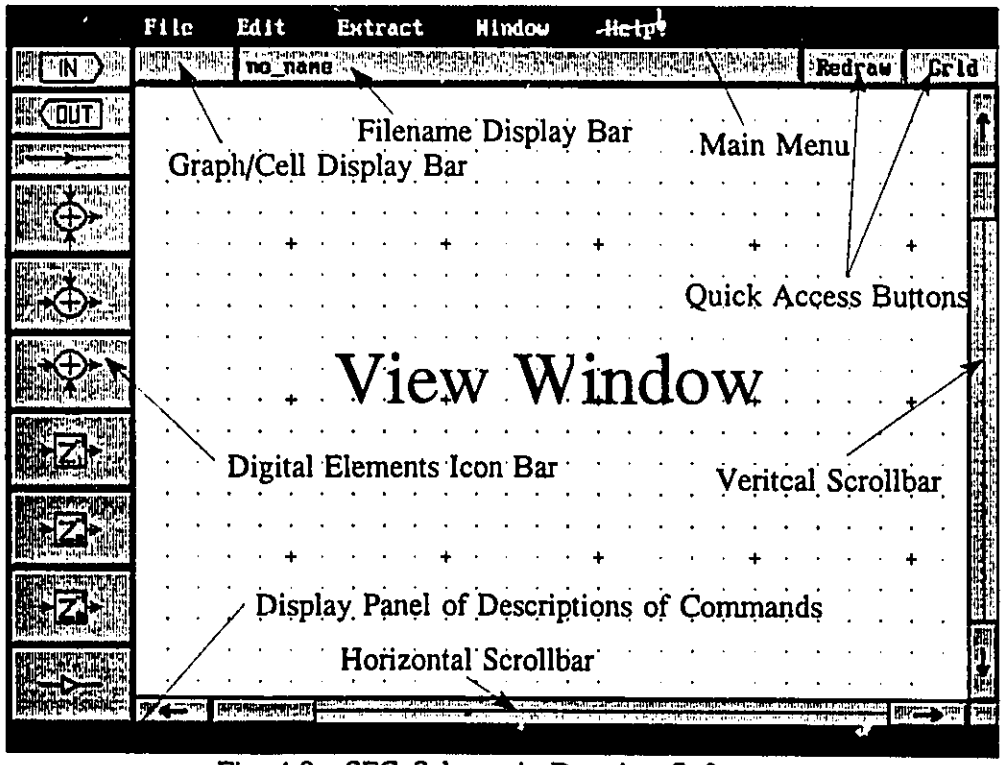


Fig. 4.8 : SFG Schematic Drawing Software

4.3.1 Placing Elements

Elements can be placed on the view window when the desired element is selected from the digital elements icon bar. The button of the selected element will be highlighted. When the mouse moves to the view window, the mouse icon will be changed to the graphical representation of the selected element and it will follow the mouse position. The movement of the element will snap to the grid with the spacing of 20 pixels. If the desired position is fixed, the element will be placed on the screen by pressing the left mouse button once. The coordinate of the position will be stored to the integer arrays called NODE_X and NODE_Y. The type of the elements will be stored to the character array called NODE. Fig. 4.9 shows the button of the selected element and the mouse location when placing elements. Every element on the icon bar will have the same procedures of placing on the view window except the multiplier and the connecting line.

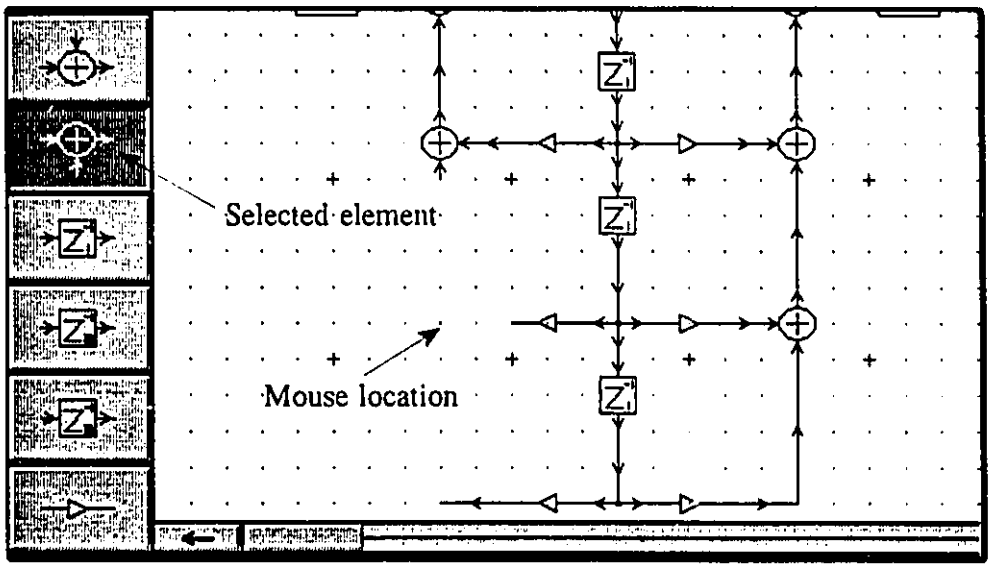


Fig. 4.9 : Placing of Element

After the multiplier is placed on the view window, a window will be displayed on the lower left corner of the screen and a line from the window will point to that specific multiplier. The multiplier value can be input through the dialogue box and it only allows inputting numerical value. If the multiplier is first appeared on the screen, the current multiplier value will be set to zero and it will shown on the top of the window. Fig 4.10

shows the window of coefficient input and the display of current coefficient.

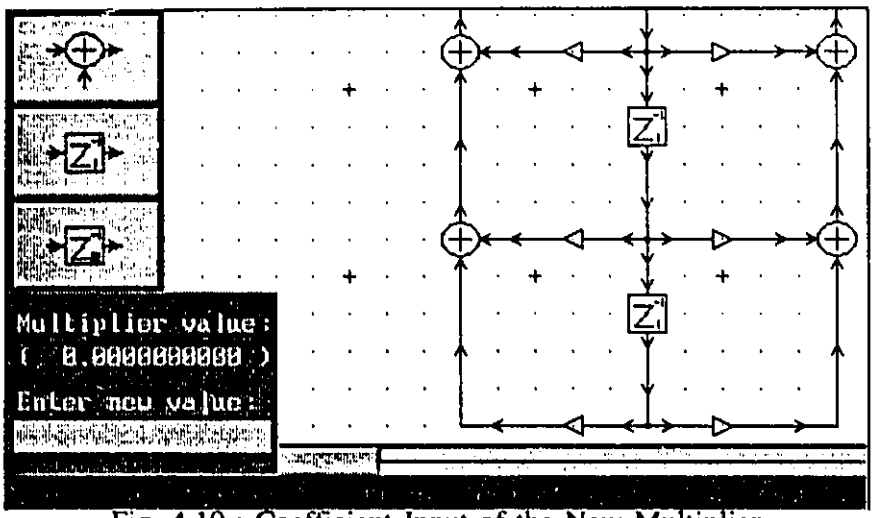


Fig. 4.10 : Coefficient Input of the New Multiplier

The starting point of the connecting line can be located when the connecting line button is selected and the mouse button is clicked once within the view window. A line will be drawn from the starting point to the end point where the mouse pointer is located. In other words, a 'rubber band' line will follow the mouse location with a defined starting point. When the mouse moves to the desired position and the left mouse button is clicked once, the end point of the line will be fixed. The connecting line will then be drawn from the starting point to the end point. Another line can be drawn continuously with the previous end point as the starting point and the current end point is fixed when the left mouse button is clicked once. Therefore, continuous lines can be drawn by locating the end points. It is obvious that discontinuous lines can be drawn by locating the starting point again. The integer arrays that store the coordinates of starting points are LFSTX and LFSTY, and the coordinates of end points are LENDX and LENDY.

Signal flow graph will transmit the signal into a node and transmit the same signal from that node to other nodes through branches. The node that acts as a relay may called extension node. The graphical representation of an extension node in the software is represented by a filled square. The software can automatically create an extension node

but is restricted only to the lines that are drawn horizontally or vertically. If the starting point of a new line is placed along any of the existing lines, an extension node will be created and the existing line will split into two lines. Another technique is checking the starting point of the new line to see whether it has the same coordinate with any of the existing lines but did not connect to any existing elements. Fig. 4.11 shows the above two techniques that create an extension node.

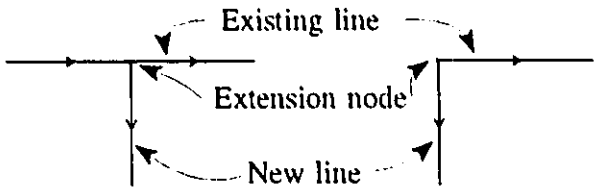


Fig. 4.11 : Creation of Extension Node

All the buttons in the digital element icon bar can be deselected by clicking the right mouse button once. Since the position of the element icons or "rubber band" line will keep changing when the mouse is moving, XOR mode has to be used so that the graphics contents that are already drawn will not be affected. The software allows drawing up to 500 elements and 1500 lines.

4.3.2 Creating and Placing Cells

Cells of digital networks can be constructed with the same method as in constructing ordinary digital networks except that input pin and output pin cannot be placed inside the cell. They are useful when some parts of the digital network are repeatedly used throughout the design, especially a digital network using systolic architecture.

After a cell has been constructed on the working window, the information about the cell will be saved by performing the command, which is in the pull-down menu under the main menu ***File*** called ***Save as cell.*** The command will first search for any lines that did not connect to or connect from any elements. The unconnected starting points or end points of the lines will then be extended out of the boundary of the cell in which they act as cell inputs or outputs. If no elements are connected to or connected from any lines,

lines will be connected automatically to the inputs or from the outputs of the elements and extended out of the boundary of the cell in which they also become inputs or outputs of the cell. The boundary of the cell will be equal to the maximum distance minus the minimum distance of the elements and add one pixel in both horizontal and vertical direction. Fig. 4.12 shows how to extend the lines out of the boundary of the cell.

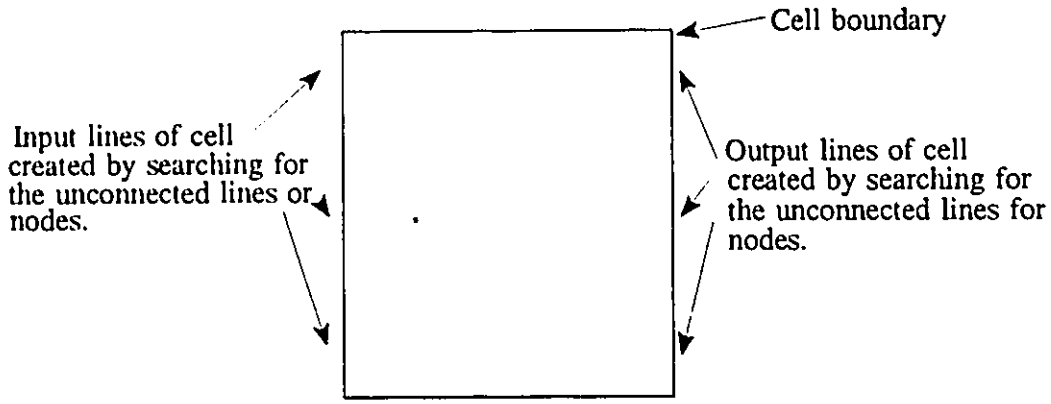


Fig. 4.12 : Creating Cell

After the information of the cell is saved to a specific file, the working window will be cleared and all the values stored in the arrays will be deleted. The structure of the cell can be modified by editing the cell using the command ***Edit cell*** which is in the pull-down menu under ***File***. The ***Graph/Cell Display Bar*** will display a name **Cell** which indicates a cell is displayed on the working window. The filename of the cell will be displayed in the ***Filename Display Bar***. Note that the input and output lines of the cell, which are created previously, will not be removed. Therefore, when the cell is saved again, the boundary of the cell will become larger and larger. It is necessary to delete the unnecessary lines before the cell is saved.

The cell can now be used for constructing the signal flow graph by retrieving the filename of the cell, which is saved previously, using the command ***Get cell*** which can be found in the pull-down menu under ***File***. Once the appropriate filename is selected, the mouse icon will be changed to the shape of the cell as shown in Fig. 4.13. The cell can be placed anywhere within the view window by moving the mouse. The exact position of the boundary, the input lines and the output lines of the cell can be seen when

the mouse is moving. When the desired location is fixed, pressing the left mouse button will place the cell to that location. It allows placing of more than one cell, with the same filename, when the command ***Get Cell*** was executed once.

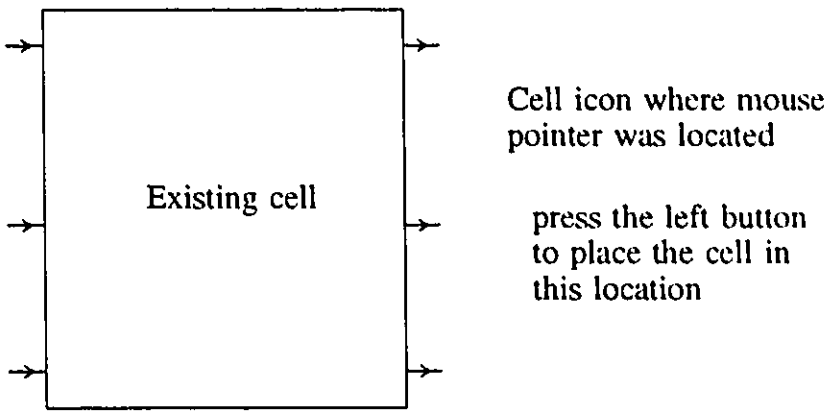


Fig. 4.13 : Place Cells on the View Window

The structure inside the cell can be made visible or invisible by using the commands, ***Look Inside Cell*** or ***Hide Inside Cell***, in the pull-down menu under ***Edit***. However, cells have to be selected first by using the commands ,***Select*** or ***Select Area***, which will be discussed in the next section. The software allows drawing up to 100 cells.

4.3.3 Select and Select Area Commands

In order to modify the existing structure such as delete, rotate, editing multiplier values, moving part of the structure, copying part of the structure, made visible or invisible of the structure of the cells, those elements, lines or cells have to be selected first. To select an element, line or cell individually, the command ***Select***, which can be found in the pull-down under ***Edit***, can be used. To select a group of elements, lines or cells, the command ***Select Area***, which can also be found in the pull-down menu under ***Edit***, can be used. The software allows selecting up to 100 elements, 200 lines and 50 cells together. When the selected elements exceed 100 elements, an error message will be displayed on the screen

rather than the program being terminated. All the selected elements, lines and cells will be deselected and the coordinates will also be deleted from the appropriate arrays.

When the command ***Select*** is selected, move the mouse to the view window and choose the desired element, line or cell. The selected element, line or cell will change to a green color. It allows selecting of more than one element, line or cell but only restricted to select within the viewing window. Since ***Select*** will search for the coordinates of the element, line or cell from the appropriate arrays which are previously stored, there is no problem to select element and cell individually because only one coordinate will be recorded for both element and cell. However, lines will have two coordinates which store starting point and end point. ***Select*** will first search for the starting point throughout the arrays LFSTX and LFSTY. If there is no matching, it will search for the end point throughout the arrays LENDX and LENDY. However, there may be more than one line which has the same starting point or end point. Therefore, if the index number of the array of the line is larger than the line with the same starting point or end point, then that line will not be selected. All the elements, lines or cells can be deselected individually by clicking the right mouse button once when the command ***Select*** is still active.

The command ***Select Area*** allows selecting of more than one element, line or cell. It is useful when part of the existing structure needs to be moved to other location or duplicate part of the existing structure. It will use the subroutine, defined in VGAWAT,

CALL SET_WINDOW_MOUSE

The subroutine allows a rectangular window to be defined using the mouse. When the subroutine is called and the left mouse button is pressed within the viewing window, the upper left corner of the rectangular window will be fixed at the location where the left mouse button is pressed. Further movement of the mouse results in a 'rubber band' rectangular box that allow the mouse to move over the entire screen. When the 'rubber band' rectangular box is covered with part of the structure that needs to be selected and the left mouse button is clicked again, the opposing diagonal coordinates of the

rectangular box will be stored in the integer variable by using the subroutine:

CALL GET_WINDOW(IX1,IY1,IX2,IY2)

where IX1 and IY1 are the bottom left corner of the box, and IX2 and IY2 are the top right corner. Then the command will search for the elements, lines or cells that are within the above rectangular box. All the selected items will also change to green color. If some of those elements, lines or cells need to be deselected, it can be done individually by using the command ***Select*** as mentioned in the previous paragraph.

4.3.4 Delete, Rotate and Coefficient Commands

The commands, ***Delete***, ***Rotate*** and ***Coefficient***, can be found in the pull-down menu under ***Edit***. When executing these commands, the target elements, lines or cells have to be selected first.

The command ***Delete*** is used to delete all the selected elements, lines and cells. It includes removing the graphical representation of above items from the view window, and removing all the necessary values from the arrays which store the information of the elements, lines and cells.

Rotation of elements is required so that they can connect from or connect to any direction. Normally, lines of the signal flow graph are drawing in the horizontal or vertical direction. Therefore, the rotation angle can be restricted to only four directions, 0°,90°,180° and 270°. When the command ***Rotate*** is selected, a sub-menu as shown in Fig. 4.14 will be used to select the rotation angle. Elements will be rotated in the counter-clockwise direction. An integer array IAN will be used to store the rotation angle of the elements. The function of rotation is to draw the signal flow graph more neatly, and not to affect the result of the netlist when extracting. All the elements can be rotated except input pin and output pin. This is because the signal in the signal flow graph normally flows from left to right. Therefore, it is only for the convenience when we look at the signal flow graph.

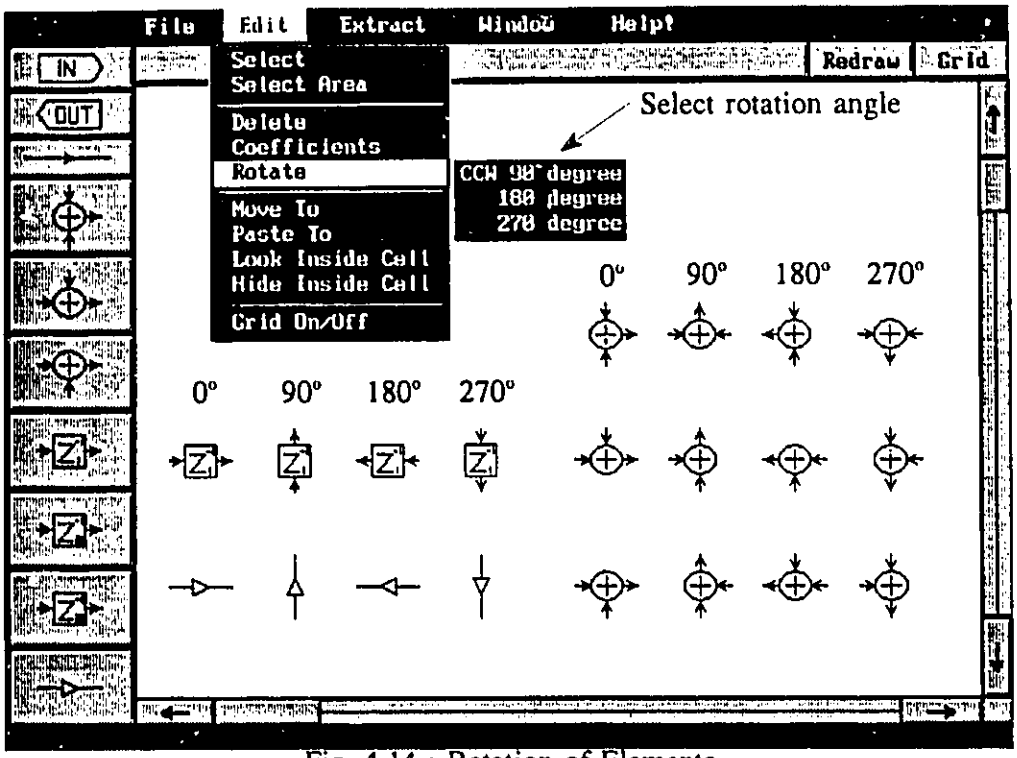


Fig. 4.14 : Rotation of Elements

Multiplier values can affect the whole characteristics of a digital filter. It is convenient to change multiplier values any time during drawing of signal flow graph. It allows us to use the same structure but with different multiplier values. Therefore, we only need to load the existing structure and change multiplier values rather than drawing the whole structure again. Besides, the multiplier values are using double precision numbers, hence it is very easy to mistype the values. When the command ***Coefficient*** is executed, a window as shown in Fig. 4.10 will be displayed at the bottom left corner of the screen. Since the multiplier value is normally defined when a new multiplier is drawn, the current coefficient will be displayed at the top of the window instead of a zero value. This can be checked to determine whether the value needs to be changed or not. If the current coefficient is correct, press <Enter> to go to the next multiplier or finish the operation and wait for other commands. Otherwise, a new value can be typed through the dialogue box at the bottom of the window. The value will be stored when <Enter> is pressed and the command will also search for the next multiplier or finish the operation.

4.3.5 Move to and Paste to Commands

The commands ***Move To*** and ***Paste To*** can be found in the pull-down menu under ***Edit***. Since it is very difficult to adjust the spacing between elements when the graph was first drawn, the command ***Move To*** allows us to move part of the graph to any location within the view window so that the graph can be drawn more neatly. Besides, it is common that some parts of the signal flow graph are duplicate, and may be with different multiplier values. Then, the command ***Paste To*** can be used to copy part of the graph and paste it to any location within the view window. The multiplier values can be changed by using the command ***Coefficient*** as mentioned in section 4.3.4.

When part of the graph has been selected, and the command ***Move To*** or ***Paste To*** is chosen, the selected structure will follow the mouse location as shown in Fig. 4.15. When the desired location is fixed, simply click the left mouse button and the selected structure will be placed on that location. The appropriate arrays that store the coordinates will be changed to a new location. The only difference between these two commands is the one with the original structure being removed where the other will be retained.

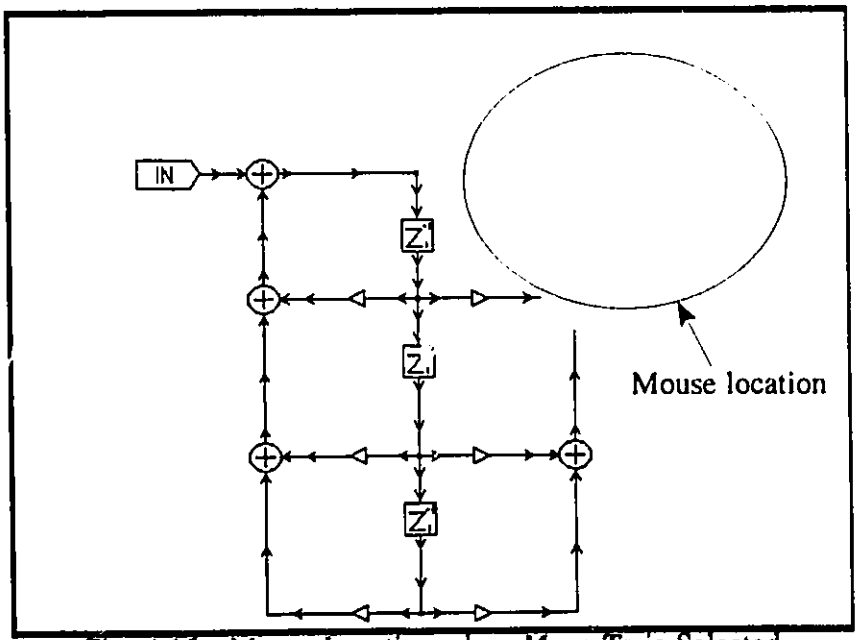


Fig. 4.15 : Mouse Location when ***Move To*** is Selected

4.3.6 Zoom In, Zoom Out and Fill Window Commands

The commands ***Zoom In***, ***Zoom Out*** and ***Fill Window*** can be found in the pull-down menu under ***Window***. Most of the graphs have a size which is even larger than the view window, therefore it is convenient to have zoom function to look at the graph in any size within the view window. The scale of the graph will depend on the spacing between the grids and the default spacing which is 20 pixels. The minimum spacing is 3 pixels and the maximum spacing is 60 pixels. When performing zooming, the upper left corner of the paper, which was mentioned in 4.2.4, will be the reference point. When the command ***Zoom Out*** is selected, the spacing will be decreased by 10 pixels when the current spacing is between 20 and 60 pixels, or when the current spacing is between 4 and 20 pixels, the spacing will be decreased by 10 pixels, otherwise, the spacing will be decreased by 1 pixel. In other words, more parts of the graph will be seen within the view window. When the command ***Zoom In*** is selected, the scaling method will be the same as the command ***Zoom Out*** except that the command will perform increasing the spacing instead of decreasing. Therefore, we can concentrate on one part of the graph and the connection of elements, lines or cells will be seen more clearly. The software allows for the executing of any commands no matter the size of the graph, and all the elements, lines and cells also snap to the grids.

The command ***Fill Window*** is used to show the whole graph on the view window. The command will first search for the width and height of the graph, then choose the closest scale that can show the whole graph on the view window. Sometimes the graph may not start drawing at the upper left corner of the paper. The command also allows moving the view window to the paper where the graph starts drawing instead of using the upper left corner as the origin. It can avoid the unused space of the paper to be shown on the view window in order to maximize the size of the graph.

4.3.7 Redraw, Grid and Clear Window Commands

The command ***Redraw*** and ***Grid*** can be found in the ***quick access buttons*** which are

shown in Fig. 4.8. ***Redraw*** command is used to refresh the view window. Every element line and cell that have already been stored in the appropriate arrays and within the view window will be drawn again. It also clears all the arrays using the command ***Select*** or ***Select Area***. ***Grid*** command is used to disable or enable the grids. It can also be found in the pull-down menu under ***Edit***. When the current status of the grid is enabled, click the button or the command in the pull-down menu will disable the grids. When the current status of the grid is disabled, click the button or command will enable the grid. The function of the grids is to place the elements, lines or cells in the exact location in order to draw a neatly graph. The elements, lines or cells are drawn to snap into the grids.

The command ***Clear Window*** can be found in the pull-down menu under ***Window***. When the command is selected, a window as shown in Fig. 4.16 will be displayed on the screen. It allows us to clarify whether the command should or should not be executed. This is because it will remove all the values in the arrays that store the information of the graph and everything on the view window. If the graph is not saved previously, all the information of the graph will be lost and cannot be recovered. When the command is confirmed, simply click the ***Yes*** button in the window or press 'y' on the keyboard. Click the ***No*** button or press 'n' will cancel the command.

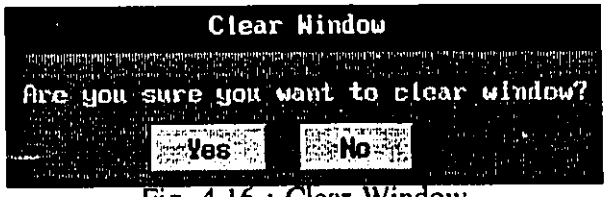


Fig. 4.16 : Clear Window

4.3.8 Function Keys Summary

It is more convenient to use the keyboard to perform the commands. Section 3.3.7 referred to the method of how the software accepts keystrokes. The combination of keystrokes will be used to execute the command in order to avoid pressing only one keystroke accidentally. The combination of keystrokes representing the commands can be found in Table 4.1.

Keystrokes	Commands	Keystrokes	Commands
Alt+s	Select	Alt+g	Grid
Alt+a	Select Area	Alt+f	Fill Window
Alt+d	Delete	Alt+i	Zoom In
Alt+c	Coefficient	Alt+o	Zoom Out
Alt+m	Move To	Alt+c	Clear Window
Alt+p	Paste To	Cltr+x	Quit
Alt+l	Look Inside Cell	F1	Help
Alt+h	Hide Inside Cell		

Table 4.1 : Keystrokes of Commands

4.4 Special Features

There are some features of the software that should be included in order to construct the graph in a more convenient way.

4.4.1 File Manager

Since graph or cell can be saved to or retrieved from a file, it is helpful to create a file manager to get the job done. When one of the commands of ***Load Graph***, ***Save Graph***, ***Edit Cell***, ***Get Cell***, or ***Save as Cell*** is selected, a file manager will display on the screen to allow a user to choose the right file. Fig. 4.17 shows a file manager of the software It includes dialogue box, current directory path, file display window, vertical scrollbar, drive display window, ***OK*** button and ***Cancel*** button.

When the file manager was first called, the dialogue box is empty, the current directory path shows the directory where the software is executed and the file display window will display the first 18 files and sub-directories according to the current directory path. Using the mouse to click on the dialogue box once, a cursor will appear. The full path and the

name of the file of the graph or cell can be entered, then the graph or cell will be retrieved or saved by pressing <Enter>. If the file is located in the current directory, simply enter the filename instead of the full path. Wild cards can also be entered through in order to filter the files that displayed on the file display window.

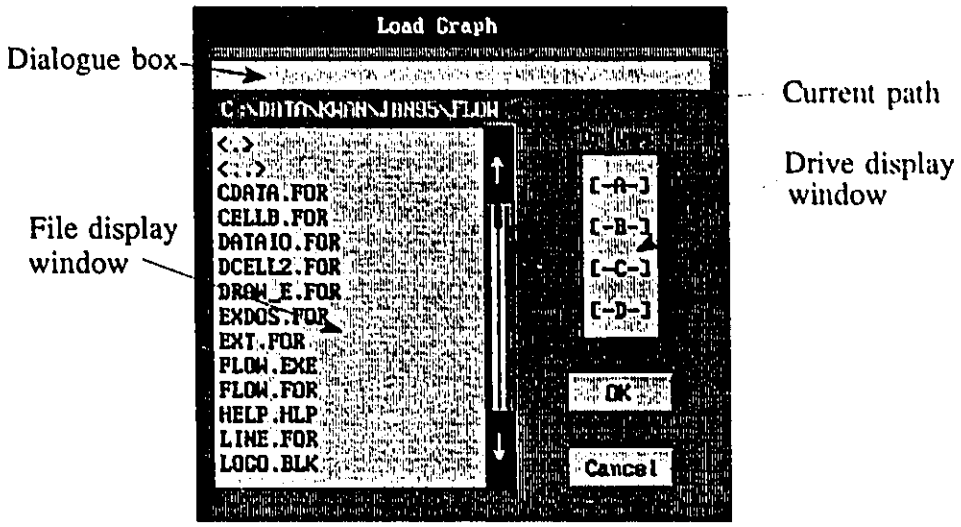


Fig. 4.17 : File Manager

Another choice of retrieving and saving a file is to select the file in the file display window. All the files and sub-directories in the current directories can be displayed by scrolling up or down using the vertical scrollbar. The order of the scrolling list will display all the sub-directories first and the files will be displayed in the ascending alphabetical order. At the top of the scrolling list, <.> represents the root directory and <..> represents the directory before the current directory. By clicking <.> once, the file display window will display all the files and sub-directories in the root directory. By clicking <..> once, it will go up one directory and display all the files and sub-directories. By clicking the sub-directory once, it will go down to one directory. The change of the directory can be monitored by checking the current directory path. When the right filename is shown on the file display window and the filename is clicked on once, the dialogue box will display the filename so that we can double check whether the right file is selected. Then, click on the ***OK*** button or press <Enter> to load from or save to the

file. There is a limit for the character arrays to store up to 500 files in one directory.

The drive display window is used to select the drive and it can access up to four drives which are specified in A, B, C and D. Click on one of the four letters, the file display window will then handle the files in that drive and the selected letter will be highlighted. The file manager can be cancelled by clicking the ***Cancel*** button or press <Esc>.

4.4.2 Changing Background and Foreground Colors

The command of ***Set Colors*** can be found in the pull-down menu under ***Window***. It can change the foreground and background colors in the view window. Since VGAWAT can display up to 16 colors, the combination of the foreground and background will be within those 16 colors. The default background color is blue (1) and the foreground color is yellow (14). Fig. 4.18 shows a window when the command is executed. The current background and foreground colors will be highlighted by wrapping around the color rectangular box in white. The color can be changed by clicking the left mouse button within the desired color box and the box will be highlighted. When both foreground and background colors have been fixed, click ***OK*** or Press <Enter> to change the current colors, or click ***Cancel*** or press <Esc> to retain current colors.

Since grids, selected elements, lines or cells, inside contents of the cells, and element icons are also using those 16 colors to display, when the selected foreground or background colors are contradicted with those colors, the program will change those colors so that everything will display properly.

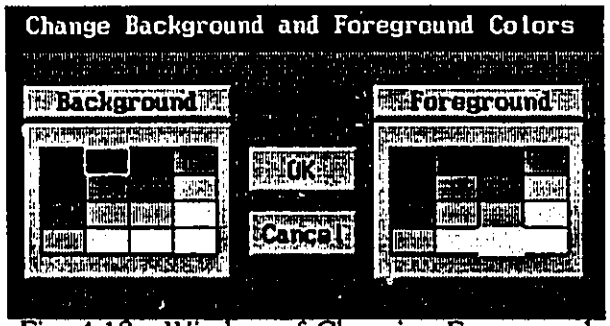


Fig. 4.18 : Window of Changing Foreground and Background Colors

4.4.3 On-Line Help

A help menu will be displayed on the screen when the ***Help!*** on the main menu is clicked. Fig. 4.19 shows the help menu in six categories. Each category has several index names which are normally the name of the commands. Click on the index name and the explanation of that index name will appear on the screen. There are two buttons at the top left corner of the help menu, ***Close*** and ***Index***. Click on the ***Index*** button will return to the help menu with index names displayed when current display of the help menu shows the explanation of that specific index name. Click on the ***Close*** button or press <Esc> will exit the help menu.

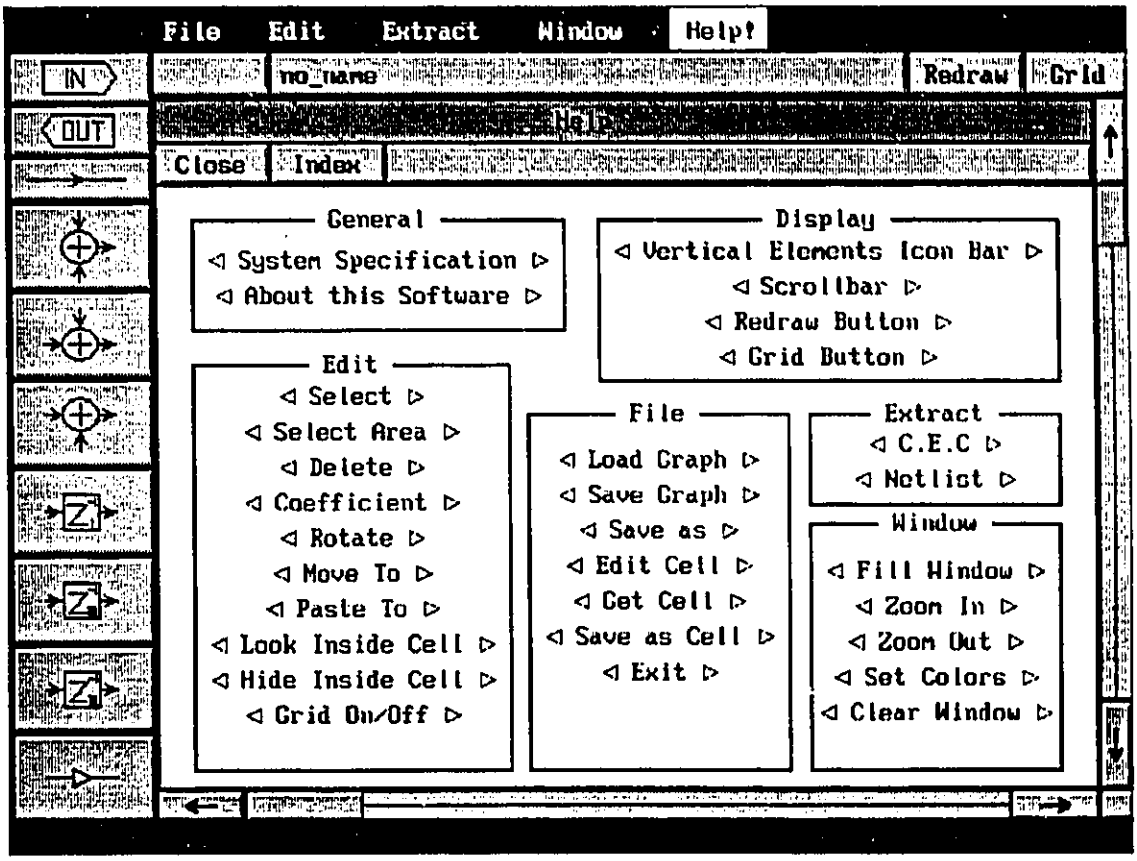


Fig. 4.19 : Help Menu

4.5 Extract

The graphical representation of digital network can now be extracted to create a netlist. The netlist includes total number of delay nodes in 1-D, 2-D and 3-D, total number of signal nodes, total number of delay nodes and signal nodes, total number of non-zero elements in the transmittance matrix, signal node number representing network output, signal node number representing external input and transmittance matrix that represent the simplified matrix representation. The netlist format, variables and its definitions of 1-D, 2-D and 3-D digital networks can be found in Appendix A.

The process to create a netlist will first search all the connections between elements of signal flow graph and record the connections by using array index numbers of the elements. Then, some of the nodes will be removed in order to minimize the total number of nodes representing a digital network without changing its characteristics. Finally, the nodes will be renumbered using the method that was proposed by Reyer, Heinen and Ninederjohn [36] in order to get a transmittance matrix which represent a computable network. The netlist will then be displayed on the screen and save to a file for the analysis of its characteristics.

4.5.1 Node Numbering

There are two different sets of arrays to store the elements and lines. The connection between two elements will exist when lines are connected between them. In the beginning, the index number will be set to one and integer arrays LFROM and LTO will all be set to zero. The index number of lines will be fixed during the search for the coordinates of the elements. If the starting point of a line is connected from the output of an element, then the index number of that element will be stored to an integer array LFROM with the same index number as in lines. If the end point of a line is connected to the input of an element, then the index number of that element will be stored to an integer LTO with the same index number as in lines. If the element is an extension node, the direction of signal flow will depend on the direction of the line. If the starting point

of the line is connected from the extension node, the signal is flowing out of the node and the index number of the extension node will be stored to LFROM. If the end point is connected to the extension node, the signal is flowing into the node and the index number of the extension node will be stored to LTO. The index number of lines will then be increased by one until all the connections have been numbered into arrays LFROM and LTO with the index numbered of the elements.

From the graph, it may need more than one line to connect from one element to another. However, the starting point of one line may have the same end point as the other line. After all the starting points and the end points of the lines, which are directly connected from or connected to the elements, have been numbered into arrays LFROM and LTO, remaining unconnected nodes will still be set to zero in the arrays. The program will first search for zero in the array LTO and let I be the index number when a zero is found, set LTO(I) equal to LFROM(I). Then, it will compare the end point LENDX(I) and LENDY(I) with the starting point LFSTX and LFSTY of all the lines. Assume J be the index number when a match is found and LFROM(J) equals zero, LFROM(J) will be set to equal LTO(I). If LTO(J) is a non-zero number, set LTO(J) equals to LFROM(J) and compare the end point LENDX(J) and LENDY(J) with the starting point of all the lines again until LTO(M) has a non-zero number. This will complete one set of lines that is connected from one element to another. The program will then search for another set of lines with the same procedure as above until all the zeros have been replaced with node numbers.

The description of the graph will now be represented by the arrays LFROM and LTO. The number of connections can now be reduced by removing LFROM and LTO that have the same node numbers. If LFROM or LTO still having zero values, that means some of the lines in the graph are not connected from or connected to any elements. The software has the ability to find the unconnected nodes or the lines that are not connected to or connected from any nodes. When the command *C.E.C* is clicked, a connection error check will be processed. A window as shown in Fig 4.20 will be displayed at the bottom of the screen. If connection error has been found, the dialogue box will display the explanation

of the error and the problem line or element will be changed to green color.

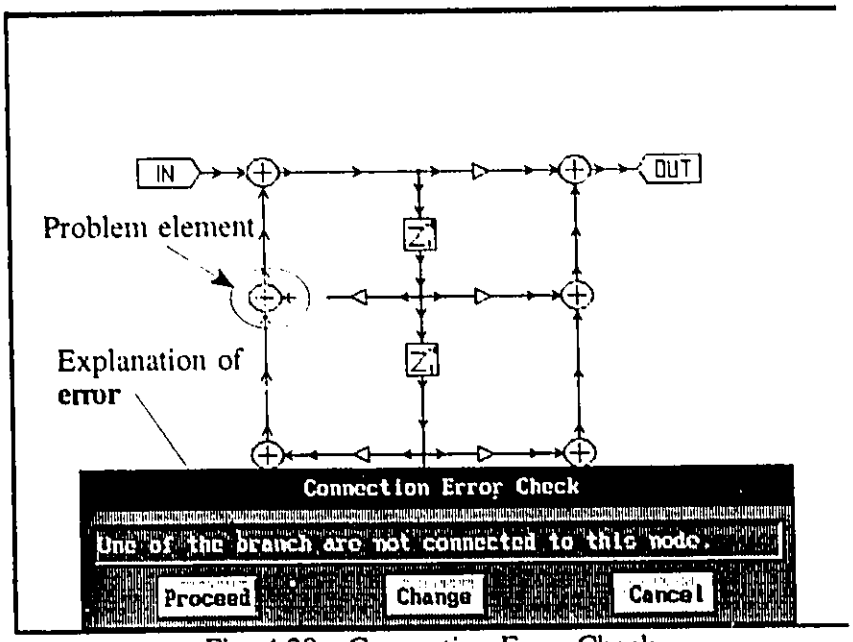


Fig. 4.20 : Connection Error Check

4.5.2 Node Minimization

The column and row of the transmittance matrix will depend on the node numbers that are stored in the arrays LTO and LFROM where LTO represents the column number and LFROM represents the row number. The transmittance will be stored in the array COE. If node LFROM(I) connects to node LTO(I) has a multiplier value, then COE(I) will be equal to that multiplier value. If node LFROM(J) connects to node LTO(J) has a transmittance of unity, then COE(J) will be equal to one.

Since node numbers are numbered with index numbers of the array which stores the information of the elements, multiplier will first be assigned as a node. The node of the multiplier can be removed by searching the node input to the multiplier and the node output from the multiplier. For example, LTO(I) and LFROM(J) store the node numbers of a multiplier, then the node of the multiplier can be removed by deleting column I. Then, set LFROM(J)=LFROM(I), and COE(J) equals to the multiplier value.

One of the node numbers can be removed if the connections are as follows:

1. If an extension node is connected to another extension node and the value between two nodes is unity.
2. If an extension node is connected after a delay node and is not connected before a multiplier.
3. If an adder is connected before an extension node and the value between two nodes is unity.

After the nodes have been minimized, the signal node number, that an external input is injected, will be stored to integer variable IP, and the signal node number, that a network output is drawn, will be stored to the integer variable LP. Since the nodes will no longer be in sequence after the minimization has been performed, the node numbers will be numbered again in order to get the node numbers in sequence.

4.5.3 Node Renumbering

Although node numbers are in sequence, a computable network should reorder the node numbers so that nth node only depends on n-1 nodes. In other words, the diagonal and the upper triangle of $\underline{V}$ in (2.8) have to be zero values.

All delay nodes in 1-D will be numbered first, then delay nodes in 2-D, and finally, delay nodes in 3-D. The signal node will be renumbered by using the algorithm in [36]. The procedure will be summarized as follows:

1. Set I equals to one.
2. Search for a row which is all zeros.
3. If Kth row is the row which is all zeros, set node K equals to I.
4. Cross out row K and column K.
5. I=I+1
6. Repeat procedure 2 to 5 until all the signal nodes have been renumbered.

If no all-zero row exists, the network is non-computable and Fig. 4.20 will display an error message.

4.5.4 Displaying Netlist

The result of the netlist will be displayed on the screen as shown in Fig. 4.21. Using the scrollbar at the right or pressing the up and down arrow keys on the keyboard, the netlist can be scrolled up and down. There are two different netlists representing a digital network, common netlist and unique netlist. The common netlist will include all the information that are shown on the screen except the node number of the input. The unique netlist will depend on the five types of analyses, hence there are five different formats of the unique netlist. The format of the common netlist and the unique netlist can be found in Appendix A.

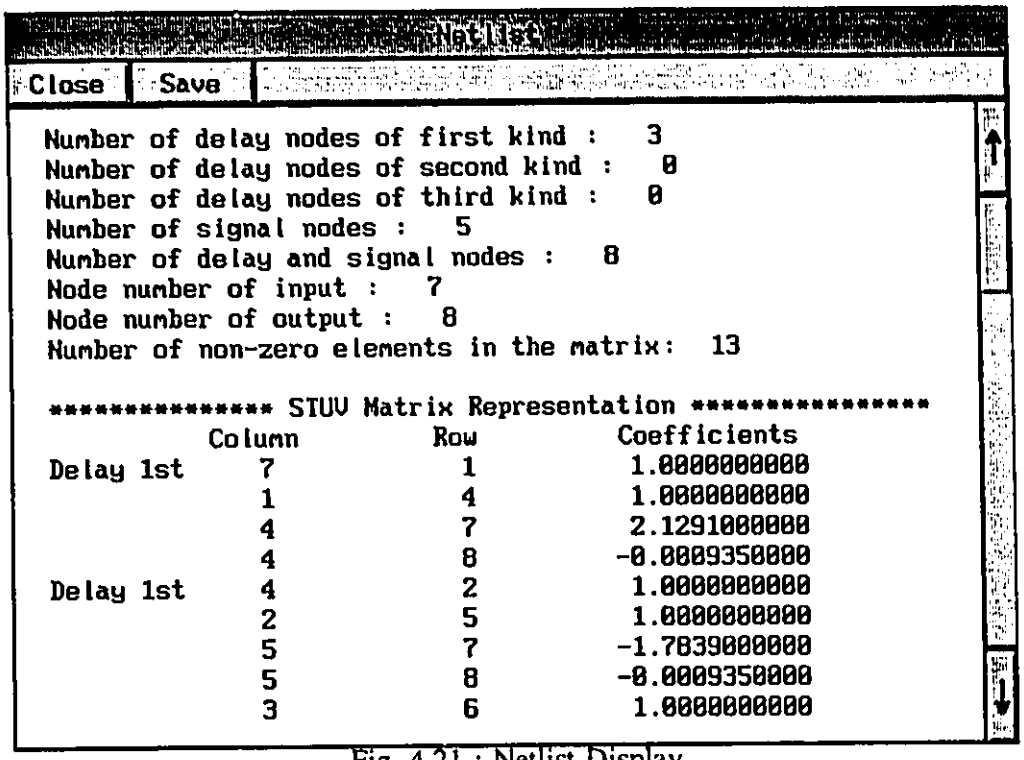


Fig. 4.21 : Netlist Display

The netlist can be saved by clicking the ***Save*** button. A file manager will be displayed, then simply enter the filename or click on the existing filename and click ***OK*** or press <Enter>. The common netlist will be saved to that specific file. An analysis choice window as shown in Fig. 4.22 will be displayed after the common netlist has been saved. It allows us to choose one or more type of analysis. The required values for each of the

analyses can be input from the window as shown in Fig. 4.23. The filename of the unique netlist will be the same as the common netlist except with different extension. The following shows the file extension of five different digital network analyses:

1. Frequency Response Analysis- .FRQ
2. Group Delay and Slope of Magnitude Response Analysis - .DSM
3. First Order Coefficient Sensitivity - .FCS
4. Noise Analysis - .NOI
5. Impulse Response Analysis - .IMP

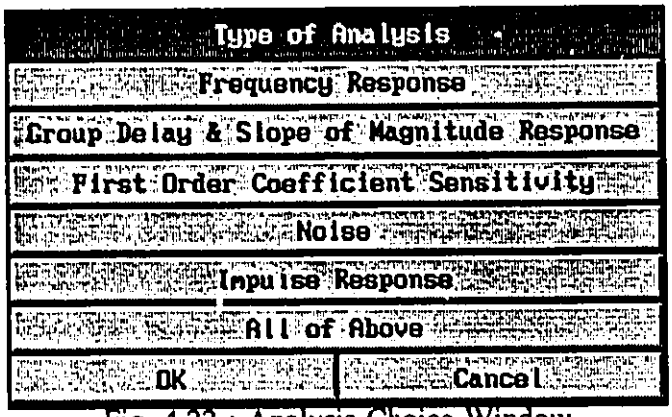


Fig. 4.22 : Analysis Choice Window

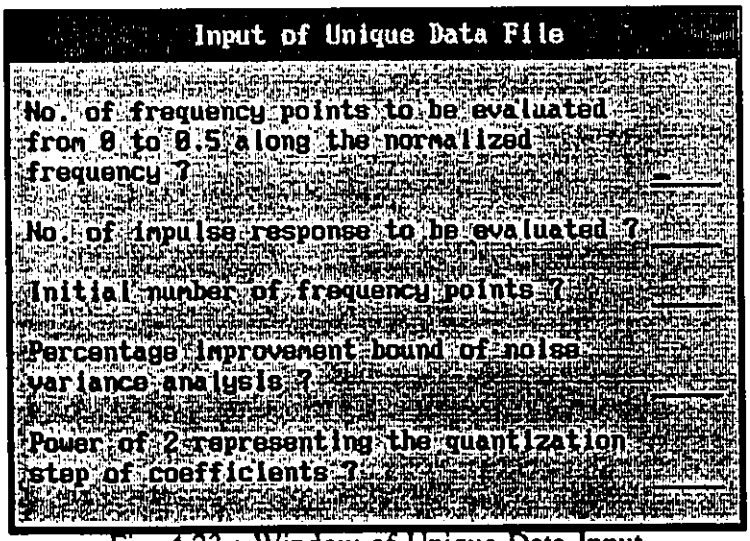


Fig. 4.23 : Window of Unique Data Input

4.6 Summary

Signal flow graph schematic drawing software has been developed completely. It is able to design 1-D, 2-D or 3-D digital network in a graphical user interface environment. The features of the software include file handling of graph and cell, cell structure creation, editing multiplier values, modification of graph and cell structure, changing view window appearance and on-line help. The graphical representation of the SFG can be extracted into netlist and can be saved to a file for further analysis. There are two different netlists, common netlist and unique netlist. The node numbers of the transmittance matrix in the common netlist will be renumbered in order to obtain a computable network. The unique netlist will store the information that depend on the type of analysis. They will have the same filename as in common netlist but with different file extension on different analysis.

The format of the transmittance matrix in the common netlist is especially for the use of simplified matrix representation [9][38]. The numerical information of describing the digital network can be found on those common netlist and unique netlist. However, they do not mean anything without the use of the analysis software to obtain its characteristics in both frequency domain and time domain. In other words, this software only accomplishes half of the task in order to design a digital network.

Chapter 5

Digital Network Analysis Software

5.1 Introduction

The analysis of digital network will employ the netlists created by the SFG schematic drawing software which was discussed in the previous chapter. The reason for creating an individual analysis software is the size limit of the arrays in WATFOR. Both schematic software and analysis software require huge arrays. Schematic software needs those arrays to store the information of the graph where the analysis software needs those arrays to compute outputs. As mentioned in chapter 2, there are five different types of analyses for the use of analyzing characteristic of digital network. It will use the simplified matrix representation method as discussed in 2.3.2 to represent a digital network. The types and the required mathematical equations of different analysis had been discussed in 2.4.

In order to analyze 1-D, 2-D and 3-D digital networks, it requires a total of 15 individual programs to compute output of 5 different analyses in those 3 different dimensions. Those 15 individual programs will be pre-compiled and run its 'EXE' file within the software. The input parameters of each analysis program module will be input by the user through this software and pass them into selected module. The input parameters include dimension of digital network, filename of netlist (extracted from SFG schematic drawing software) and type of analysis. The output data of selected analysis in the selected dimension will

be stored in a temporary file, then the software will retrieve the file and plot the data on the screen.

5.2 Module Interaction

Basically the software integrates analysis modules together and plots the computed output data on the screen. Fig. 5.1 shows the modular structure related to the main module and the analysis modules. In Fig. 5.1, the circles represent the modules that compute the output of each analysis. The lines with arrows in both ends represent the input parameters that pass through the main program into the selected module and the computed output data pass back into the main program. The function of the main program is to collect input parameters for analysis and plot the output of the analysis of 1-D digital network in 2-D graph mode, and 2-D and 3-D digital networks into 3-D graph mode.

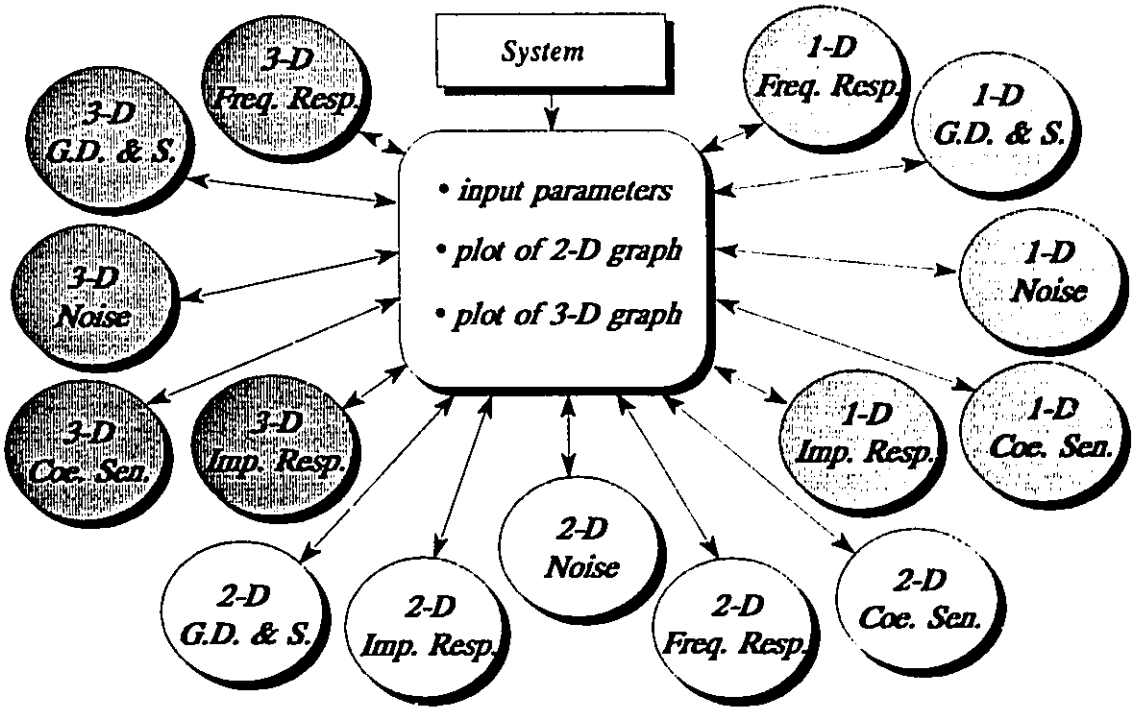


Fig. 5.1 : Modular Structure of the Digital Network Analysis Software

The main program is written in WATFOR-77 with the use of VGAWAT graphics library

to develop a graphical user interface software. All the analysis modules are written in FORTRAN-77 with slightly modifications that are suitable to use in WATFOR-77. The modifications include changing declaration of COMPLEX*16 into COMPLEX*8 and input and output data format. The software appear on the screen as shown in Fig. 5.2.

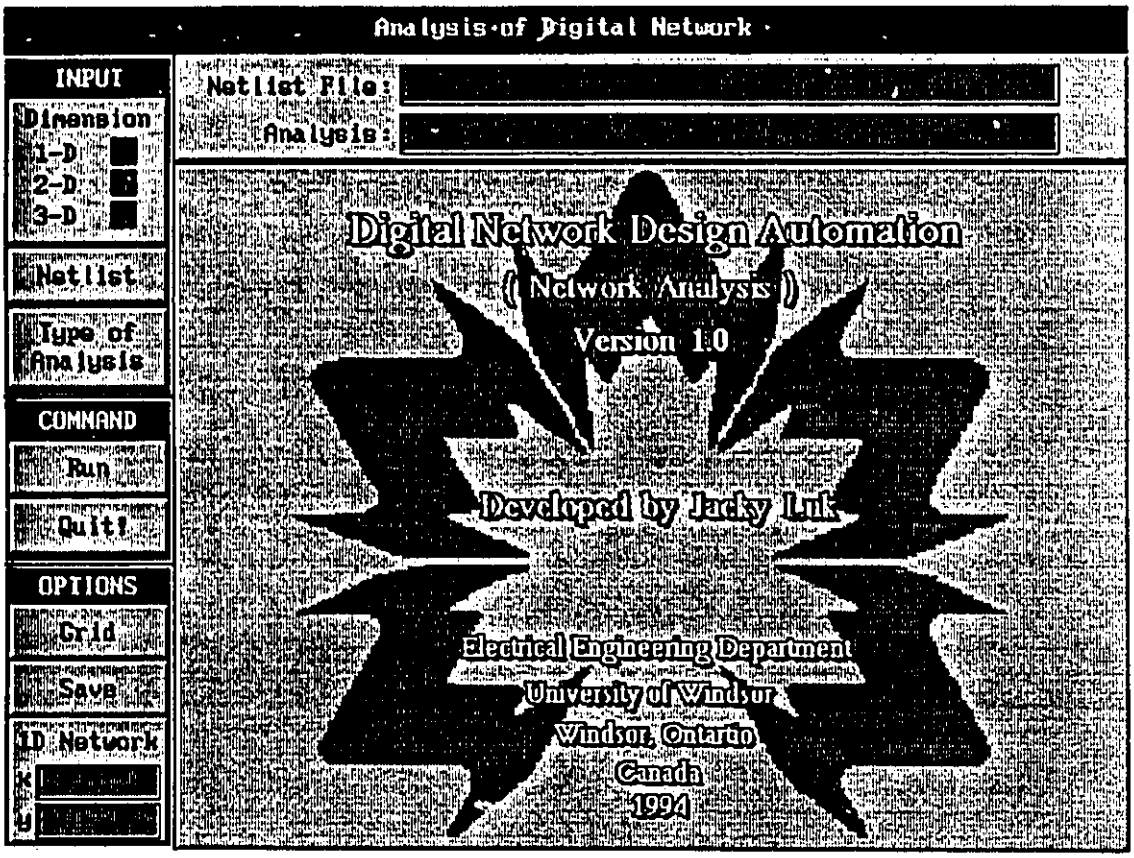


Fig. 5.2 : Digital Network Analysis Software

5.2.1 Input parameters

The input parameters include dimension of network, common netlist filename and type of analysis. In Fig. 5.2, the top panel with the name ***INPUT*** shows input parameters to be passed to the selected analysis module. Dimension of digital network can be clicked within the box beside 1-D, 2-D or 3-D. When 1-D is selected, the display window where the logo is displayed will change to display two 2-D graph windows as shown in Fig. 5.3. When 2-D is selected, the display window will display the 3-D graph window with

buttons at the right and the angles of rotation and elevation at the bottom as shown in Fig. 5.4. When 3-D is selected, the display window will display the same features as in 2-D but with the third frequency axis as shown in Fig. 5.5.

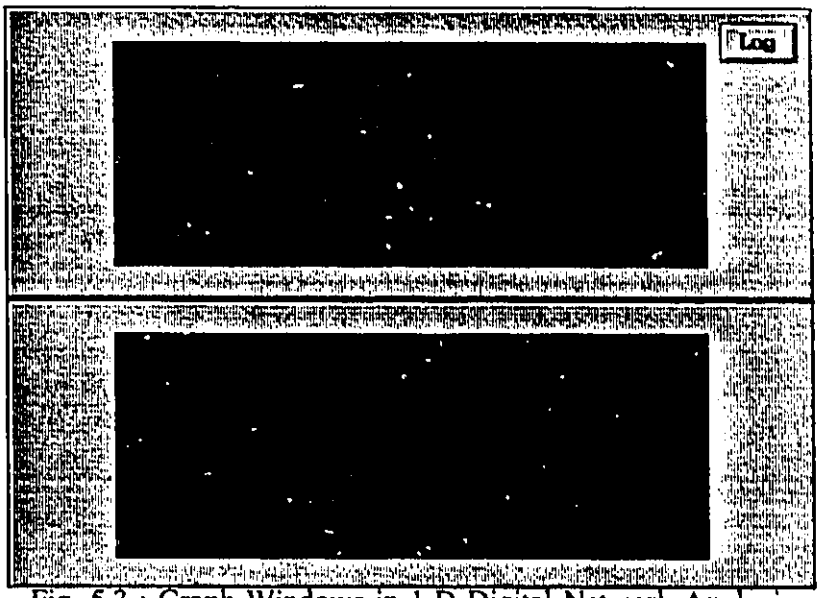


Fig. 5.3 : Graph Windows in 1-D Digital Network Analysis

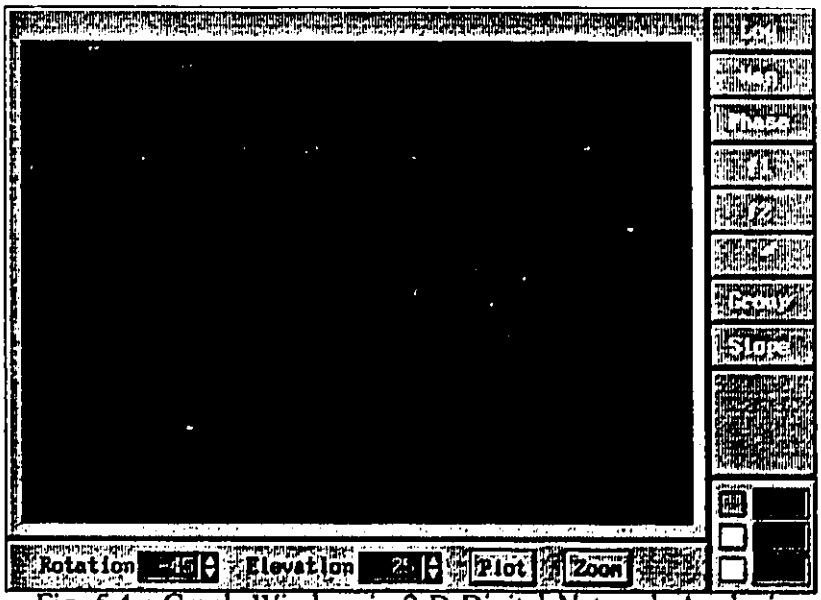


Fig. 5.4 : Graph Window in 2-D Digital Network Analysis

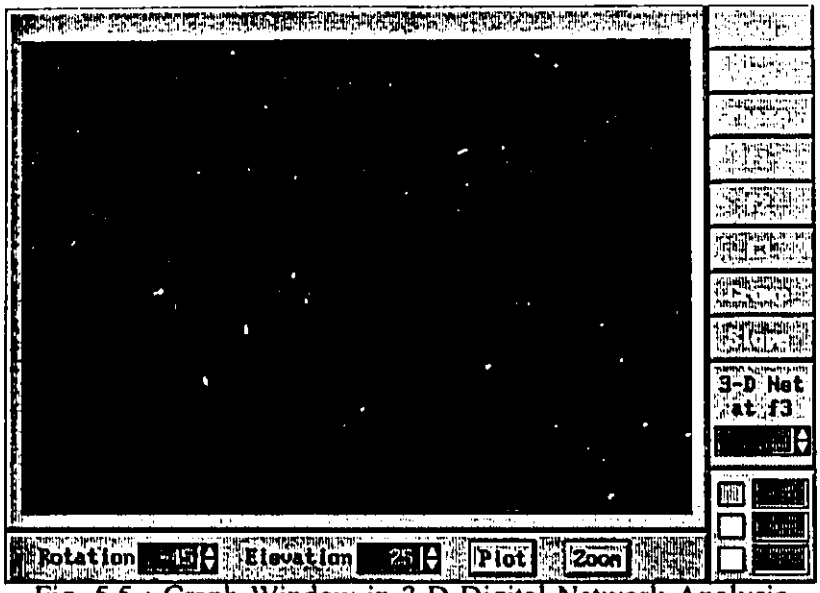


Fig. 5.5 : Graph Window in 3-D Digital Network Analysis

When ***Netlist*** button is pressed, a file manager which is the same as in Fig. 4.16 will display on the screen. The common netlist filename which is created in SFG Schematic Drawing Software can be retrieved and the full path of the filename will be displayed on the top dialogue box called ***Netlist***. When ***Type of Analysis*** button is pressed, the analysis choice window as shown in Fig. 4.22 will display on the screen. However, this software only allows us to choose one type of analysis at a time. As mentioned in section 4.5.4, the filename of the unique netlist has the same filename of common netlist but with different file extension. Therefore, the file extension will be specified when specific analysis is selected. Selected analysis will be displayed on the second dialogue box called ***Analysis***. All input parameters will be saved to a temporary file called ***TEMP1*** and the format is as follows:

common netlist filename
unique netlist filename
output filename

The file ***TEMP1*** has to be saved to a directory where analysis modules are located. The output filename is also a temporary file called ***DATA.DAT*** which stores the computed output data. The main program will plot the 2-D graph or 3-D graph by retrieving the

data from ***DATA.DAT***. Since the filename of the unique netlist depends on the filename of the common netlist, the software will not permit us to input the unique netlist before inputting common netlist. It is also able to check the file status and file format.

5.2.2 Execution of the analysis modules

When the button ***Run*** is clicked, the input parameters will pass to the specific analysis module. The module will use the information from the common and unique netlist to compute the output. The function sub-program that can run the dos commands within the software had been discussed in section 3.2. All analysis modules are executed as shown in the following source codes:

```
.
.
DIRNAME(1:3)='CD '
DIRNAME(4:)='CURRENT PATH'
STATUS=EXDOS(DIRNAME)
STATUS=EXDOS('MODULE1')
.
.
```

The program will first change to the directory where analysis modules and ***TEMP1*** are located. Then, the analysis execution file ***MODULE1.EXE*** will be executed. Table 5.1 summarize the filename of each analysis module in each dimension.

	1-D	2-D	3-D
Freq. Resp.	FR1DPMR.EXE	FR2DPMR.EXE	FR3DPMR.EXE
G.D. & S.	G1DPMR.EXE	G2DPMR.EXE	G3DPMR.EXE
Noise	N1DPMR.EXE	N2DPMR.EXE	incomplete
Coe. Sen.	S1DPMR.EXE	S2DPMR.EXE	S3DPMR.EXE
Impulse Resp.	IR1D1.EXE	IR2D1.EXE	IR3D1.EXE

Table 5.1 : Filenames of the Analysis Modules

The required node numbers format of the unique netlist in 3-D noise analysis as shown

in appendix A are different from the unique netlist input format of the original source codes. The unique netlist input format of the original source codes requires the node numbers immediately before a non-zero and non-unity coefficients. In theory, those nodes are not required to compute noise response. However, the source codes show that they are used for the calculation of the output. Therefore, the analysis of noise in 3-D cannot be completed.

After the calculation is completed, the output data will be stored to the file ***DATA.DAT*** as specified in the file ***TEMP1***. The main program will retrieve the data from the file ***DATA.DAT*** by using the same format that the data are stored into the file ***DATA.DAT***, from the selected analysis module. Since only one analysis module can be executed at one time, this method can avoid including all the source codes of analysis programs into the main program in order to save memory to develop more features of the software.

5.3 Plotting of 2-D Graph

All types of analyses in one dimensional digital network will be using the 2-D graph as shown in Fig. 5.3 to plot the output. There are two graph windows that are used to plot two different output in one analysis. Table 5.2 summarize the output to be plotted in the top and bottom graph windows in five different analyses.

	Top Graph Window	Bottom Graph Window
Frequency Response	Frequency Response	Phase Response
G.D. & S. of Mag. Resp.	Group Delay	Slope of Mag. Resp.
Noise	Noise	Not Used
Coefficient Sensitivity	Coefficient Sensitivity	Not Used
Impulse Response	Impulse Response	Not Used

Table 5.2 : Output to be Plotted in the Top and Bottom Graph Windows

The background and foreground colors of 2-D graph windows are black and green, respectively. The width and height of both graph windows are 401 and 151 pixels, respectively. It is capable to plot up to 1001 data points in both graph windows. The

horizontal axis is the normalized frequency f in frequency domain and the sample number n in time domain. The vertical axis is the magnitude of the corresponding response.

5.3.1 Plotting of Data

It will first search for maximum and minimum data values throughout the output file ***DATA1.DAT***, then the total number of data points from the unique netlist. In the vertical axis, the maximum value is set to the top of the graph window, 150 pixels, and the minimum value is set to the bottom of the graph window, 0 pixel. Remaining data values along the vertical axis are converted into pixels using equation, $150 \frac{(ymax\ \ ydata)*150}{ymax\ \ ymin}$, where ydata is the data values to be converted, and ymax and ymin are the maximum and minimum values, respectively. In the horizontal axis, the total number of data is set at the right of the graph window, 400 pixels, and 1 is set to left of the graph window, 0 pixel. Remaining values along the horizontal axis are converted into pixels using equation, $400-\frac{(n-i)*400}{n-1}$, where i is the index number of the array that store the data, and n is the total number of data. The data can be plotted on the graph windows by using the following source codes:

```
call set_colors(10,0)
call set_window(169,216,569,366)
call clear_window
do i=1,n-1
 itemp1=150-int(((ymax1-data1(i))*150)/(ymax1-ymin1))
 itemp2=150-int(((ymax1-data1(i+1))*150)/(ymax1-ymin1))
 call move_to(400-((n-i)*400)/(n-1), itemp1)
 call draw_to(400-((n-(i+1))*400)/(n-1), itemp2)
end do
```

The subroutine SET_WINDOW has set the width and height of the graph window with 151 and 401, respectively. The real variables, ymax1 and ymin1, are the maximum and the minimum values that have been checked throughout the whole set of data points. The real array DATA1 is used to store one set of data values from the file ***DATA.DAT***. If there is another set of data values in ***DATA.DAT***, it will be stored to another real array DATA2. All the graphs are simply drawn by connecting all the data points together with

straight lines except impulse response analysis. For the impulse response analysis, a vertical line is drawn from the data point to the vertical zero value in each sample number. The integer variable, itemp2, in the above source codes will be modified to calculate the zero value , then itemp2=150-int((ymax1*150)/(ymax1-ymin1). Fig. 5.5 shows the graphs in frequency response analysis and Fig. 5.6 shows the graph in impulse response analysis. If the total number of points is larger than 400 in one set of data, the graph will be drawn with more than one data point in one pixel along the horizontal axis.

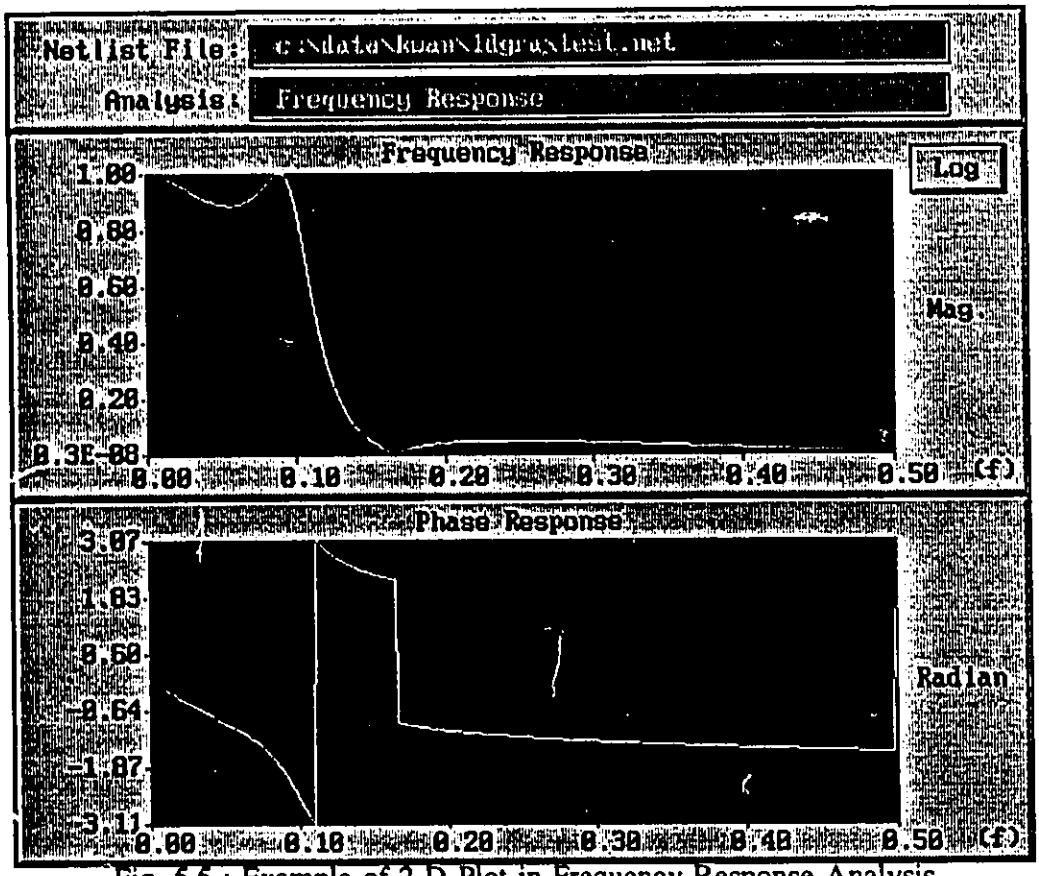


Fig. 5.5 : Example of 2-D Plot in Frequency Response Analysis

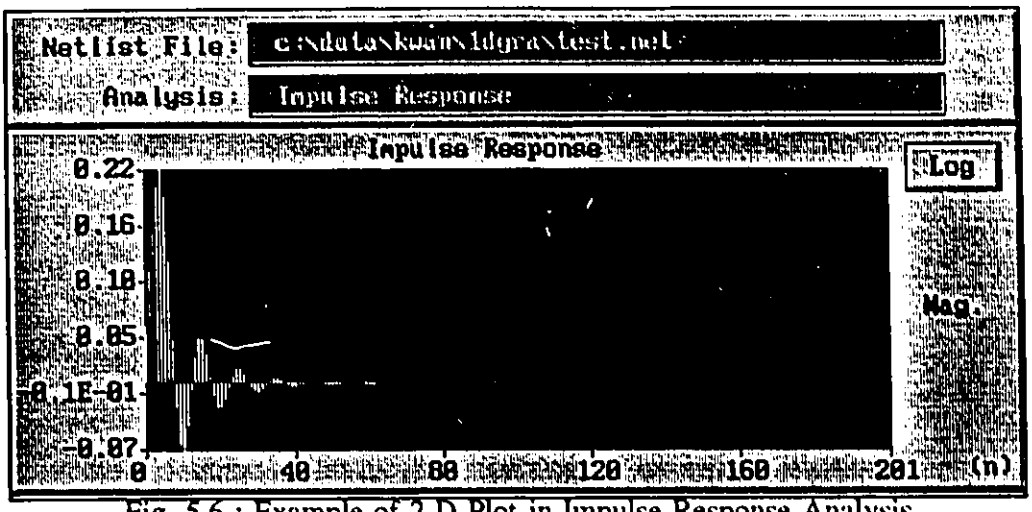


Fig. 5.6 : Example of 2-D Plot in Impulse Response Analysis

5.3.2 Displaying Coordinates in Both Graph Windows

When the mouse moves to either of the graph windows, the mouse icon will change from an arrow to a cross. The screen coordinate of the mouse within the graph window in pixel can be detected by calling the subroutine GET_MOUSE_B. Then, it converts the screen coordinate into appropriate window coordinate in pixel. By using the maximum and minimum data values in both horizontal and vertical axes, the approximate data value, where the mouse is located, can be calculated and displayed on the dialogue box located at the lower left corner of the software as shown in Fig. 5.2. The dialogue box beside 'x' represents the horizontal axis, which is normalized frequency or sample number. The dialogue box beside 'y' represents the vertical axis, which is the magnitude. The following source codes show the procedures from drawing the mouse icon to displaying the data values of the top graph window. Note that the movement of the mouse icon is drawing in XOR mode so that graphic contents of the graph window will not be affected. The initial values of ixp and iyp will be set to zero.

```
Loop
  .
call get_mouse_b(ix,iy,ibut)
 if(dim.eq.'1d'.and.ymax1.ne.99999.and.ymin1.ne.99999)then
  if(ix.ne.ixp.or.iy.ne.iyp)then
                    .
```

```
            call mouse_off
            call set_mode(2)
            call set_colors(15,0)
            call set_window(169,216,569,366)
c  Drawing the previous location of mouse icon with background color of the
c  graph window.
            call move_to(ix-169-4,iy-216)
            call draw_to(ix-169+4,iy-216)
            call move_to(ix-169,iy-4-216)
            call draw_to(ix-169,iy+4-216)
c Drawing the present location of mouse icon in white.
            call move_to(ixp-169-4,iyp-216)
            call draw_to(ixp-169+4,iyp-216)
            call move_to(ixp-169,iyp-4-216)
            call draw_to(ixp-169,iyp+4-216)
c
          call set_mode(0)
          if(ix.le.569.and.ix.ge.169.and.iy.ge.216.and.iy.le.366)then
c Convert the screen coordinate, ix and iy, into screen coordinate, ixx and iyy.
            ixx=ix-169
            iyy=iy-216
c Convert iyy into data value (magnitude).
            tdatay=ymax1-((150-iyy)*(ymax1-ymin1))/150.
            call trun_data(tdatay,sdata)
            call set_colors(10,8)
            call set_window(18,25,88,6)
            call move_to(4,17)
            call write_graphics(sdata)
c Convert ixx into normalized frequency or sample number.
            tdatax=xmax1-((400-ixx)*(xmax1-xmin1))/400.
            write(sdata,'(f8.5)') tdatax
            call set_window(18,47,88,28)
            call move_to(4,17)
            call write_graphics(sdata)
           else
            call mouse_on
           end if
          end if
         end if
c Store the present mouse location into ixp and iyp.
         ixp=ix
         iyp=iy
           .
         End Loop
```

5.3.3 Zooming and Grid Commands

The grids of both graph windows can be enabled or disabled by clicking the button ***Grid*** under the panel ***Option***. If the grids are shown on the graph windows, clicking the button will disable the grids. If the grids are not shown on the graph windows, clicking the button will enable the grids.

The software also has the function of zooming so that the pass-band, transition- band and stop-band can be zoomed to display clearly. Zooming uses the subroutine, SET_WINDOW_MOUSE in VGAWAT, to define the area that needs to be enlarged. The subroutine has already been discussed in section 4.3.3. The horizontal coordinates of the starting and ending points of the rectangular graphics window will first convert from the screen coordinates to appropriate window coordinates. Then, the window coordinates will convert to horizontal data values, normalized frequency or sample number. The closest index numbers of array ***DATA1*** or ***DATA2*** that represent those normalized frequency or sample number will be searched. The graph will be plotted according to the range of the index numbers in ***DATA1 or DATA2***. Fig. 5.7 shows the zooming feature of the software. Zooming can be performed in either of the two 2-D graph windows. The whole set of data values can be plotted again by using the mouse to wrap around the whole graph window.

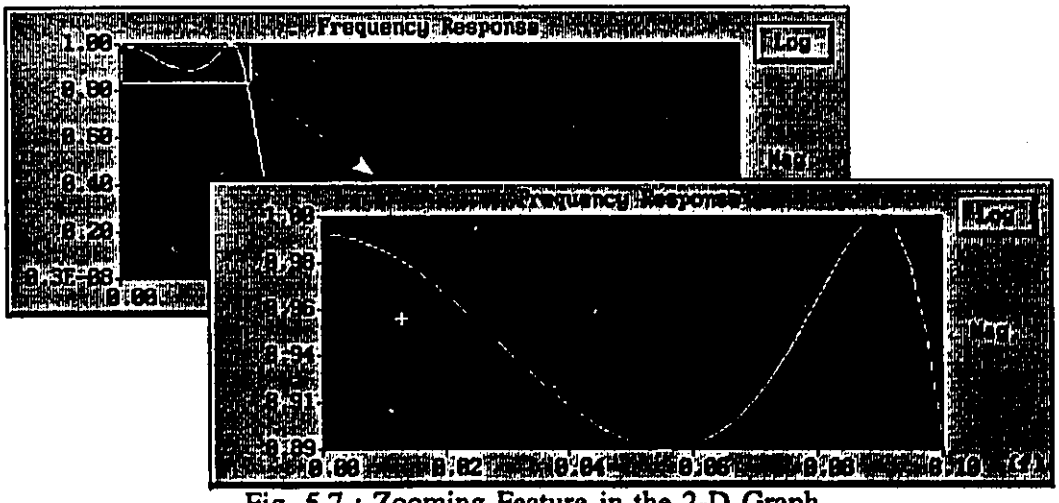


Fig. 5.7 : Zooming Feature in the 2-D Graph

5.3.4 Plot of dB in Frequency Response

It is necessary to plot the magnitude of the frequency response in the logarithmic graph using equation $dB=-20Log_{10}(Magnitude)$. When the button ***Log*** which is located at the top right corner of the top graph window is clicked, the graph window will plot the dB of the current frequency response and the button will be highlighted. The logarithmic graph can also perform those features as discussed in section 5.4.3 and 5.4.4. Fig. 5.8 shows an example of logarithmic graph. Remaining analyses will not enable the ***Log*** button.

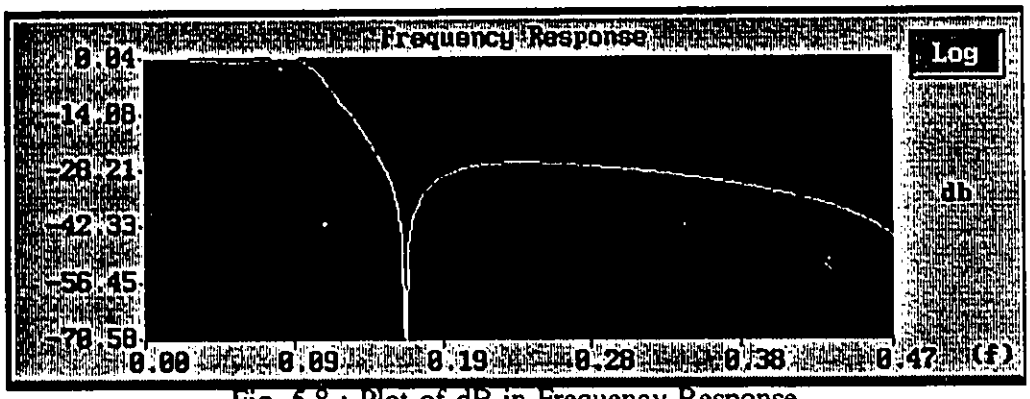


Fig. 5.8 : Plot of dB in Frequency Response

5.3.5 Changing of Multiplier Value in the Analysis of Coefficient Sensitivity

Normally, there is more than one multiplier in digital network. It is necessary to create a feature so that coefficient sensitivity of each multiplier in digital network can be plotted on the graph window. Fig. 5.9 shows the top graph window that coefficient sensitivity of multiplier is plotted. There is a dialogue box which shows the multiplier value of the current graph of coefficient sensitivity. By clicking the up or down arrow beside the dialogue box, the rest of the coefficient sensitivity of multipliers can be plotted and the related multiplier value will display on the dialogue box. Orders of multiplier values to be plotted depend on the orders of multiplier values that are listed in the common netlist.

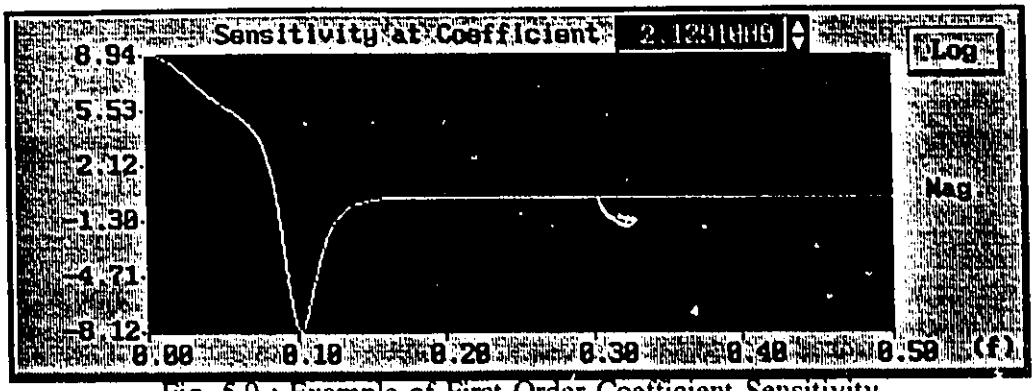


Fig. 5.9 : Example of First Order Coefficient Sensitivity

5.4 Plotting of 3-D Graph

All five types of analyses in 2-D and 3-D digital network will be using 3-D graph to plot the output. For plotting the output computed from 2-D digital network, two of the axes in 3-D graph represent two of the frequency axes in 2-D digital network and the third axis represents the magnitude of the response. Since there is one more frequency axis in 3-D digital network, the third frequency axis has to be fixed in order to plot the analysis output on 3-D graph window. Dimensions of the 3-D graph are 201×201×141 pixels along the f1, f2 and magnitude axis, respectively. It is capable of plotting up to 101 by 101 data values on the screen.

5.4.1 Plot of Data from 2-D Digital Network

The analysis output is stored in two-dimensional array INPUT. The maximum and minimum values will be searched from the array INPUT. Total number of data along two frequency axes can be found in the unique netlist file. The initial position of those three axes in the 3-D graph projecting onto the 2-D window are: f1 along the horizontal axis, f2 perpendicular to the screen and the magnitude along the vertical axis. The origin is located at the middle of the 2-D graph window. This cannot view all three axes on the 2-D graph window. It is necessary to rotate all three axes in order to get the isometric view. The position of the graph can be set to an isometric view by calling the subroutine, SET_POSITION(25.,-45.) with 25 represents the degree of elevation and -45 represents

the degree of rotation, with the following source codes:

```
      subroutine set_position(i,j)
c
c     assigned new angle of rotation and elevation.
c       elevation = i
c       rotation  = j
c
c   Rotate the x-y plane.
      call set_identity
      call set_origin(0.,0.,0.)
      call set_projection(0.0)
      call set_zoom(306,273,1.0,1)
      call set_scale(1.2,1.2,1.2)
      call set_rotate_z(1.*(j+45.)+180.)
      call set_rotate_x(-45.)
      call set_rotate_y(35.)
      call set_rotate_z(30.)
c
c   The angle of elevation.
      call set_rotate_x(1.*(i-37.))
c
      return
      end
```

Subroutines related to 3-D drawing have been discussed in section 3.3.5. In 3-D graph, the pixel can be in any real numbers and the origin is in the middle of the 3-D graph so that the graph is plotted in both positive and negative side along each axis. In f1 and f2 axes, both maximum pixel numbers are 100 and both minimum pixel numbers are -100. In the magnitude axis, the maximum pixel number is 70 and the minimum is -70. Data values of the output can be converted into pixel numbers using equation, $70-\frac{140*(zmax-input)}{zmax-zmin}$, where *input* is the data to be converted, and *zmax* and *zmin* are the maximum and minimum data values, respectively. The normalized frequencies, f1 and f2, in 2-D or 3-D digital network are plotted from -0.5 to 0.5. Index numbers of both dimensions in the array INPUT can be converted into pixel numbers using equation, $100-\frac{200*(nx2-i)}{nx2-nx1}$, where i is the index number in either of the dimension, and nx2 and nx1 are 0.5 and -0.5 in both dimensions. 3-D graph is plotted on 2-D screen using the following source codes:

```
      do i=nx1,nx2
       do j=ny1,ny2-1
        call move_to_3d(100.-(200.*(nx2-i))/(nx2-nx1),
     *       100.-(200.*(ny2-j))/(ny2-ny1),
     *       70.-(140.*(zmax-input(j,i)))/(zmax-zmin))
        call draw_to_3d(100.-(200.*(nx2-i))/(nx2-nx1),
     *                  100.-(200.*(ny2-(j+1)))/(ny2-ny1),
     *                 70.-(140.*(zmax-input(j+1,i)))/(zmax-zmin))
       end do
      end do
      do j=ny1,ny2
       do i=nx1,nx2-1
        call move_to_3d(100.-(200.*(nx2-i))/(nx2-nx1),
     *       100.-(200.*(ny2-j))/(ny2-ny1),
     *       70.-(140.*(zmax-input(j,i)))/(zmax-zmin))
        call draw_to_3d(100.-(200.*(nx2-(i+1)))/(nx2-nx1),
     *                  100.-(200.*(ny2-j))/(ny2-ny1),
     *                 70.-(140.*(zmax-input(j,i+1)))/(zmax-zmin))
       end do
      end do
```

The 3-D graph is plotted by the following procedures:

1. set row=1, and column=1.
2. Join the data values along the row in the array INPUT.
3. Increase the column number by one.
4. Repeat procedures 2 and 3 until the column number reaches the maximum number of column in the array.
5. set row=1, and column=1.
6. Join the data values along the column in the array, INPUT.
7. Increase the row number by one.
8. Repeat procedures 6 and 7 until the row number reached the maximum number of row in the array.

Fig. 5.10 shows an example of plotting a 3-D graph on 2-D graph window.

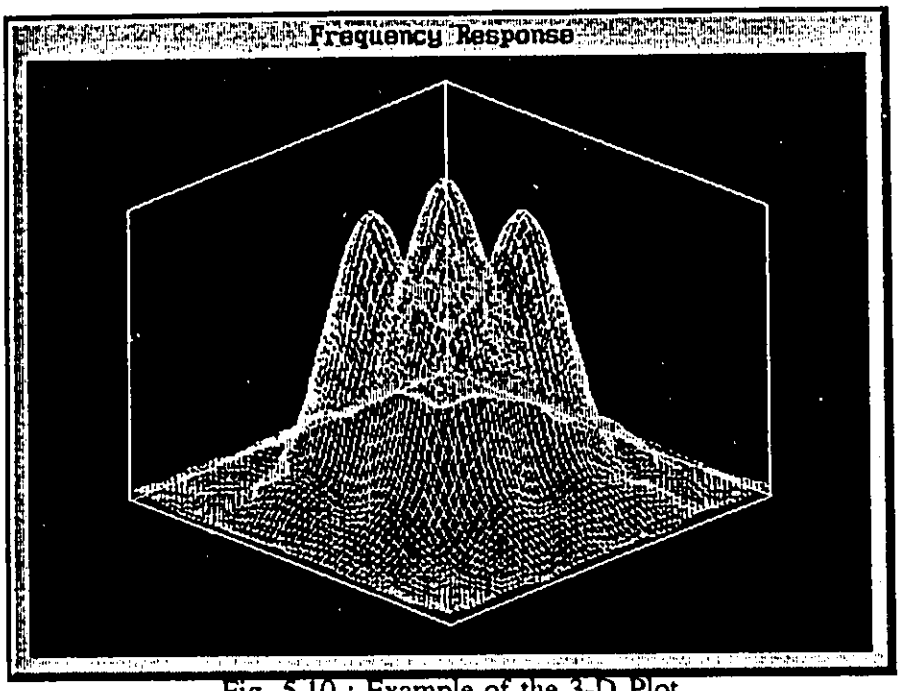


Fig. 5.10 : Example of the 3-D Plot

5.4.2 Plot of Data from 3-D Digital Network

The procedure of plotting an analysis output from a 3-D digital network is the same as mentioned in section 5.5.1. The only difference is the format of storing data values into the two-dimensional array INPUT. Since there is one more frequency axis f3 in 3-D digital network, *n* two-dimensional output data are stored in the file ***DATA.DAT***, where *n* is the total number of points in the third frequency axis that starts from -0.495 to 0.495. When one of the analysis is selected, the first graph displayed on the screen is the two-dimensional output data at f3=-0.495. Source codes of how to retrieve output data from the file ***DATA.DAT*** into the array INPUT are shown as follows:

```
      if(sim.eq.'dsm')then
       open(4,file='data.dat',err=650,status='old',
*          access='direct',form='formatted',recl=60)
       goto 660
      else if(sim.eq.'fcs')then
       open(4,file='data.dat',err=650,status='old',
*          access='direct',form='formatted',recl=10)
```

```
            goto 660
           else
            open(4,file='data.dat',err=650,status='old',
     *           access='direct',form='formatted',recl=40)
            goto 660
           end if
           norec=1
           do i=1,nf1
             do j=1,nf2
               if(sim.eq.'dsm')then
                 read(4,1010,rec=norec) input(i,j)
                 norec=norec+1
               else if(sim.eq.'fcs')then
                 read(4,1010,rec=norec) input(i,j)
                 norec=norec+jk
               else
                 read(4,1000,rec=norec) input(i,j)
                 norec=norec+1
               end if
             end do
           end do
           close(4)
1000         format(2x,e14.7)
1010         format(e10.4)
```

From the source codes, 'dsm' represents the analysis of the group delay and slope of magnitude response, 'fcs' represents the analysis of the first order coefficient sensitivity, jk represents the total number of multiplier in the network, and nf1 and nf2 represent the total number of points along f1 and f2 axis, respectively.

The third frequency axis f3 can be moved to another frequency between -0.495 to 0.495 by changing the starting record of the record specifier, norec, in the READ statement as follows:

```
norec=nrec*ntot1*ntot2+1
```

where ntot1 and ntot2 are the total number of points in f1 and f2, respectively. f3 is set to -0.495 when nrec is set to 0, and f3 is set to 0.495 when nrec is set to n, where n is the total number of points along f3. The third frequency axis f3 can be changed by

clicking the up or down arrow as shown in Fig. 5.11, and the frequency value will be displayed on the dialogue box.

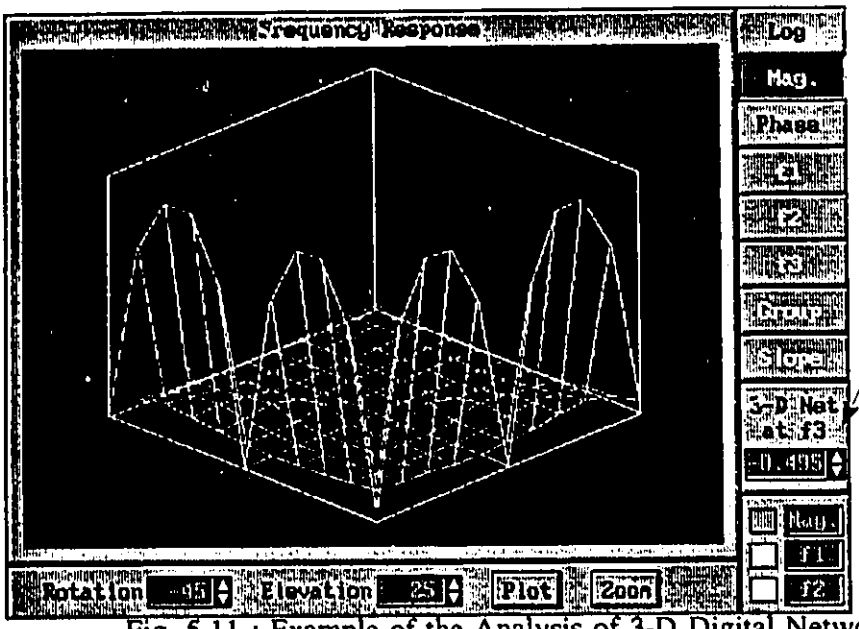


Fig. 5.11 : Example of the Analysis of 3-D Digital Network

5.4.3 Adjusting rotation and elevation angles

The subroutine SET_POSITION(IELE,IRO), as mentioned in section 5.5.1, can be used for adjusting the elevation and rotation angle of the existing 3-D graph. 'IELE' represents the elevation angle which rotates about the x-y plane, and 'IRO' represents the rotation angle which rotates about the magnitude axis. Fig. 5.12 shows the directions of rotation in positive or negative angles with its rotation axis. The default angle of rotation and elevation are -45 and 25, respectively. They can be increased or decreased by clicking the up or down arrow beside the dialogue boxes located at the bottom in Fig. 5.12. The angle will increase or decrease by an interval of one. It also allows us to input both angles using keyboard. When the mouse is clicked within one of the dialogue boxes, the numerical value of angle of rotation or elevation can be input using the keyboard. The updated angle of rotation or elevation will be displayed on the dialogue boxes. However, it will only take effect when ***Plot*** button is pressed.

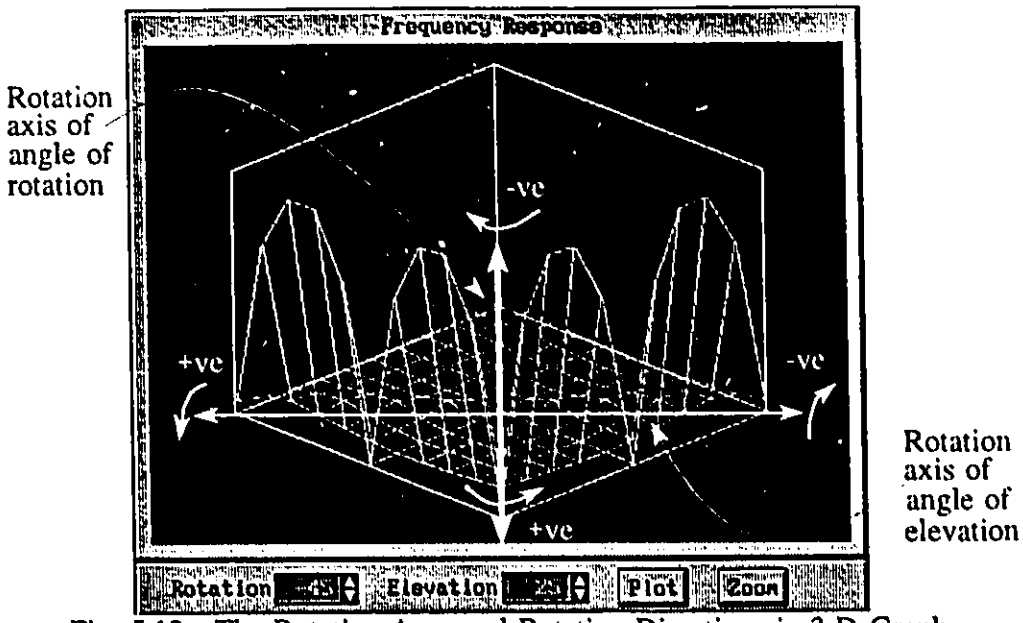


Fig. 5.12 : The Rotation Axes and Rotation Directions in 3-D Graph

5.4.4 Zooming and Grid Commands

Any 3-D graph can be zoomed to the ranges by setting beginning and ending frequency values in f1 and f2 frequency axes or sample values in n1 and n2 sample number axes. When the ***Zoom*** button is clicked, the window as shown in Fig. 5.13 is used to enter ranges. Then, the values of the frequency range or sample number range will be converted to the closest range of index numbers in the array INPUT. Updated 3-D graph will then be drawn on the screen. In Fig. 5.13, both ranges in f1 and f2 are set from 0 to 0.5.

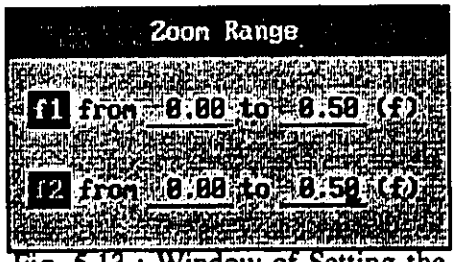


Fig. 5.13 : Window of Setting the Range in 3-D Graph

Fig. 5.14 shows the frequency response with both f1 and f2 are set to the range of 0 to

0.5. The original graph of Fig. 5.14 was shown in Fig. 5.10 with the range of -0.5 to 0.5 in both frequency axes.

Grids and scales in 3-D graph can be made disable or enable when the ***Grid*** button, which is also used for 2-D graph, is clicked. Grids and scales are shown in Fig. 5.14. The scale on each axis will be displayed in different colors. Each colors represents different axis and their legends are displayed in the panel which is located at the bottom right corner of the software.

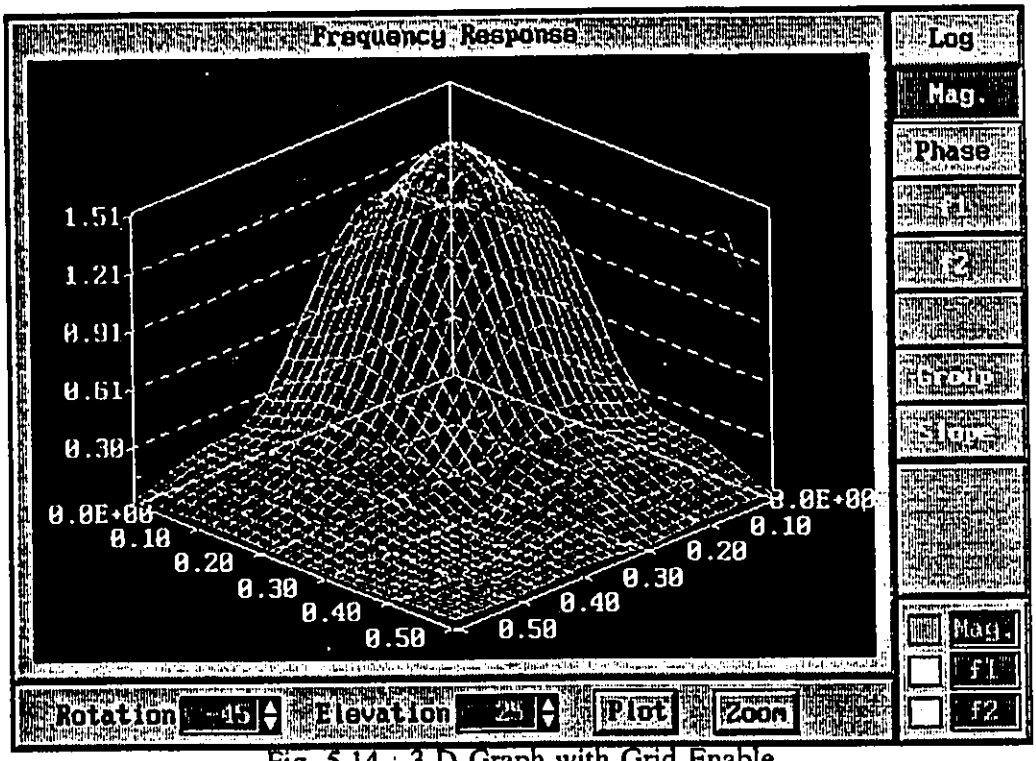


Fig. 5.14 : 3-D Graph with Grid Enable

5.4.5 Displaying different 3-D graph in the same analysis

In the analysis of frequency response, and group delay and slope of magnitude response, there are more than one graph to be used for analysis. Since only one graph is allowed to plot on the screen at one time, the buttons bar, which is located at the right of the software, will be used to select another graph in the same analysis when 2-D or 3-D digital network is selected.

When the analysis of frequency response is selected in both 2-D or 3-D digital network, the buttons, ***Log***, ***Mag.***, ***Phase***, will be enabled with the command names on the button changed to black color. The first graph to be displayed is the magnitude in the frequency response and the button ***Mag.*** will be highlighted. The dB of magnitude or phase response can be displayed by pressing the button, ***Log*** or ***Phase***, respectively.

In 2-D digital network, the group delay and slope of magnitude response are plotted with respect to the frequency axis of f1 or f2. Therefore, the buttons, ***f1***, ***f2***, ***Group*** and ***Slope***, will be enabled. The first graph to be displayed is the group delay with respect to f1, and the buttons, ***f1*** and ***Group***, will be highlighted. The rest of the graphs in this analysis can be displayed by pressing the buttons as shown in Table 5.3. Only one of the buttons in ***f1*** and ***f2*** or ***Group*** and ***Slope*** can be selected at one time. The same techniques can be used in 3-D digital network except that one more frequency axis, f3, will be added.

Button	Button	Type of Graph
f1	Group	Group Delay with respect to f1
f2	Group	Group Delay with respect to f2
f1	Slope	Slope of Magnitude Response with respect to f1
f2	Slope	Slope of Magnitude Response with respect to f2

Table 5.3 : Buttons Pressed and its Representation in the Analysis of Group Delay and Slope of Magnitude Response of 2-D Digital Network

Remaining analyses will not enable the buttons in the buttons bar. However, plot of different multiplier values of the analysis of coefficient sensitivity in both 2-D and 3-D digital networks employ the same method as mentioned in section 5.4.5.

5.5 Summary

The software is able to analyze five different analyses in 1-D, 2-D and 3-D digital networks. The types of analyses include: frequency response, group delay and slope of magnitude response, noise, first order coefficient sensitivity and impulse response. Source codes of each individual analysis are pre-compiled with a total of 14 modules to be

employed by the software except for the noise analysis in 3-D digital network.

Input parameters, such as dimension of digital network, common netlist filename and unique netlist filename, are collected and saved to a file ***TEMP1***. The software then run the selected module that read the input parameters from the file ***TEMP1***. The computed output data will store in the file ***DATA.DAT***. The software will then plot the graph by retrieving data values from the file ***DATA.DAT***. 2-D graph will be used to plot output in 1-D digital network, and 3-D graph will be used to plot the output in 2-D and 3-D digital network.

In 2-D graph, zooming can be performed using mouse to select the area to be zoomed. The grids can be disabled or enabled by pressing ***Grid*** button. The dB of the magnitude response can be plotted by pressing ***Log*** button. The coefficient sensitivity in different multiplier value can be plotted by pressing the up or down arrow located at the top of the top graph window.

In 3-D graph, the difference between 2-D and 3-D digital networks is the 3-D digital network has one more frequency axis, f3. The frequency f3 can be increased or decreased by pressing the up or down arrow located at the middle right of the software. Zooming can be performed by setting the range in the frequency axes, f1 and f2, in both 2-D and 3-D digital networks. The grids and scales can be disabled or enabled by pressing ***Grid*** button. The 3-D graph can be rotated by adjusting the angle of rotation and angle of elevation located at the bottom of the software. The plot of other available graphs in the same analysis can be performed by choosing appropriate buttons in the buttons bar located at the right of the software except for the analysis of coefficient sensitivity. The coefficient sensitivity in different multiplier value can be plotted by pressing the up or down arrow located at the top of the graph window.

Chapter 6

Summary and Future Directions

6.1 Summary and Conclusion

In this thesis, a graphic package for the purpose of analyzing 1-D, 2-D and 3-D digital networks has been developed. The package includes the schematic drawing of signal flow graph and analysis of its characteristics.

Both softwares are written under WATFOR-77 programming language with the use of VGAWAT graphics library in order to develop a graphical user interface environment. VGAWAT contains 58 graphics routines that are only capable to perform simple graphics functions, such as mouse handling, 3-D drawing, coloring, creating window, drawing lines and so on. Software mechanism, such as button shadow, button pressed, button highlight, pull-down menu and scrollbar, have to be developed by using those simple graphics functions.

Signal flow graph schematic drawing software was first used to design the structure of 1-D, 2-D or 3-D digital network. Elements used to construct a digital network include: adder, delay and multiplier. External input that was injected into the network is represented by an input pin and the network output is represented by an output pin. Lines are used to connect elements together. When an output of one element is connected to more than one element's input, an extension node is automatically created. It can have

one input branch and multiple output branches. The software is capable of constructing a digital network with 500 elements, 1500 lines and 100 cells. The features of the software include: file handling of graph and cell, cell structure creation, editing multiplier values, modification of graph or cell structure, changing view window appearance and on-line help. Signal flow graph can be saved to or retrieved from a file using file manager. Any structure that is drawn on the working window can be saved as a cell. It can be retrieved and placed on the screen with the internal structure hidden and only the boundary, input and output lines shown. The internal structure can also be made visible or invisible. Multiplier values can be altered when the specific multiplier element is selected. It also shows the current multiplier value of the selected multiplier. Modification of the graph or cell structure includes: rotate, delete, move and paste commands. Note that all the necessary elements, lines or cells have to be selected first. Structure on the view window can be zoomed in or zoomed out. View window can also be moved up, down, left or right. Foreground and background colors of the view window can be changed to any color within those 16 colors in VGAWAT. User menu of SFG schematic drawing software can be found in Appendix B.

The SFG, that currently displayed on the screen, can be extracted into netlist and can be saved to a file for further analysis. The procedures of extracting the netlist will perform node numbering, node minimization and node renumbering in order to create a computable transmittance matrix with minimum number of nodes. The format of the transmittance matrix is especially for the use of simplified matrix representation method. There are two different netlists, common netlist and unique netlist, that stored the numerical information of the digital network. Common netlist includes: total number of delay nodes in 1-D, 2-D and 3-D, total number of signal nodes, signal node number that network output is extracted, total number of non-zero elements in the transmittance matrix, and transmittance matrix. The unique netlist stores the information which depends on the type of analysis. The format of common netlist and unique netlist can be found in Appendix A.

Digital network analysis software utilizes the common netlist and unique netlist, that are

created by the SFG schematic drawing software, to compute the output of five different types of analyses in 1-D, 2-D and 3-D digital networks. Types of analyses include: frequency response analysis, group delay and slope of magnitude response analysis, noise analysis, first order coefficient sensitivity analysis and impulse response analysis. Source codes of each individual analysis program are pre-compiled into individual modules and to be employed by the software.

Input parameters, such as dimension of the digital network, common netlist filename and unique netlist filename, are collected and saved to a file ***TEMP1***. The software then run the selected module which input parameters is read from the file ***TEMP1***. The computed output data will be stored in the file ***DATA.DAT***. The software will then plot the graph by retrieving the data values from the file ***DATA.DAT***. 2-D graph will be used to plot the output in 1-D digital network with the maximum data points of 1001 and 3-D graph will be used to plot the output in 2-D and 3-D digital networks with the maximum data points of 101 by 101.

In 2-D graph, zooming can be performed using the mouse to select the area to be zoomed. The grids can be disabled or enabled by pressing the ***Grid*** button. The dB of the magnitude response can be plotted by pressing the ***Log*** button. The coefficient sensitivity in different multiplier value can be plotted by pressing the up or down arrow which is located at the top of the top graph window.

In 3-D graph, the difference between 2-D and 3-D digital networks is the 3-D digital network has one more frequency axis, f3. Frequency axis, f3, can be increased or decreased by pressing the up or down arrow which is located at the middle right of the software. Zooming can be performed by setting the ranges in the frequency axes, f1 and f2, in both 2-D and 3-D digital networks. Grids and scales can be disabled or enabled by pressing the ***Grid*** button. 3-D graph can be rotated by adjusting the angle of rotation and angle of elevation. The plot of other available graphs in the same analysis can be performed by choosing the appropriate buttons in the buttons bar located at the right of the software except for the analysis of coefficient sensitivity. The coefficient sensitivity

in different multiplier value can be plotted by pressing the up or down arrow which is located at the top of the graph window. User menu of digital network analysis software can be found in Appendix C.

Examples of designing 1-D, 2-D and 3-D digital networks can be found in Appendix D, E and F, respectively. Number of statement lines in schematic drawing software and digital network analysis software are 9226 and 3521, respectively. Source codes of both software are so large that they will not be printed in this thesis.

6.2 Future Work

Results show that WATFOR-77 programming language and VGAWAT graphics library were adequate to develop graphical user interface software. However, WATFOR-77 executes the program using the base memory in PC only. This leads to the limitation of defining huge arrays. In other words, number of nodes defining in the signal flow graph are limited. This is significant when executing the analysis modules. They require storing the whole transmittance matrix into a two dimensional array in order to perform calculation. As mentioned in chapter 2, transmittance matrix is usually sparse. It is necessary to develop an algorithm to compute the analysis output without storing the entire transmittance matrix into array. Another approach may compile the analysis programs with a 32-bits compiler so that extended memory can be used. However, the compatibility of a 16-bits and 32-bits compiler needs to be investigated.

Another problem is to get a hard copy of signal flow graphs and output graphs. WATFOR-77 and VGAWAT do not have a function to print graphics. Two possible techniques may be used to print graphics. First of all, graphics contents may be grabbed from the screen into a picture format file and printed by any word-processor which accept that picture format. In this method, the software itself cannot directly print the graphics contents. Secondly, graphics contents may be programmed as a printer readable format by using POSTSCRIPT page description language. The POSTSCRIPT language is a programming language designed to convey a description of virtually any desired page to

a printer [37]. However, it requires a printer which accepts POSTSCRIPT programs.

References

[1] Darrel L. Lager, Stephen G. Azevedo, "SIG - A General-Purpose Signal Processing Program", *Proceedings of IEEE*, Vol. 75, No. 9, pp. 1322-1332, Sept. 1987.

[2] Chris Ouslis, Martin Snelgrove, Adel S. Sedra, "A Filter Designer's Filter Design Aid: filtorX", *Proceedings of IEEE International Symposium On Circuits and Systems*, Singapore, Vol. 1, pp. 376-379, June 1991.

[3] Chris Ouslis, Martin Snegrove, Adel S. Sedra, "Multi-Rate Switched Capacitor Filter Design with Aggressive Sampling-Rates: filtorX in Action", *Proceedings of IEEE International Symposium On Circuits and Systems*, San Diego, Vol. 3, pp. 1183-1186, May 1992.

[4] B. M. Silveria, "Computer-Aided Design of Passive Ladder Filters", Master's Thesis, University of Toronto, Department of Electrical Engineering, Toronto, Canada, 1992.

[5] G. Szentermai, "FILSYN - A General-Purpose Filter Synthesis Program", *Proceedings of IEEE*, Vol. 65, pp. 1443-1458, Oct. 1977.

[6] R. K. Henderson, L. Ping, J. I. Sewell, "A Design Program for Digital and Analogue Filters: PANDDA", *Proceedings of ECCTD*, pp. 289-293, 1989.

[7] Sattam Dasgupta, Mahesh M. Mehendale, V. R. Sudershan, Rajeev Jain, Nagaraj Subramanyam, James Hochschild, "FDT - A Design Tool for Switched Capacitor Filters", *Proceedings of IEEE* IC-CAD, pp. 446-449, 1989.

[8] L. E. Turner, D. A. Graham, P. B. Denyer, "The Analysis and Implementation of Digital Filters Using a Special Purpose CAD Tool", *IEEE Transactions on Eduation*, 32(3), pp. 287-297, August 1989.

[9] H. K. Kwan, "Analysis of Digital Filter Networks", Ph.D. Thesis, University of London, 1981.

[10] William K. Pratt, *Digital Image Processing*, Wiley, New York, 1978.

[11] Ronald E. Crochiere, Alan V. Oppenheim, "Analysis of Linear Digital Networks", *Proceedings of IEEE*, Vol. 63, No. 4, pp. 581-595, April 1975.

[12] R. E. Seviora, M. Sablatash, "A Tellegen's Theorem for Digital Filters", *IEEE Transactions on Circuit Theory*, Vol. CT-18, pp. 201-203, Jan. 1971.

[13] A. Fettweis, "A General Theorem for Signal-flow Networks with Applications", *IEEE International Symposium on Electrical Network Theory*, pp. 3-4, 1971.

[14] M. L. Blostein, "On the Application of Certain Network Concepts to Arbitrary Systems", *Proceedings of IEEE International Symposium on Circuit Theory*, pp. 213-217, 1972.

[15] A. Y. Lee, "Topological Properties of the Summing and Branching Matrices of Signal-flow Graphs and the Application to Sensitivity Analysis", *Proceedings of 6th Asilomar Conf. on Circuits and Systems*, Nov. 1972.

[16] "Computer-Aided Equation Formulation and Sensitivity Analysis for Discrete Systems", *Proceedings of 16th Midwest Symposium on Circuit Theory*, April 1973.

[17] Yutze Chow, Etienne Cassignol, *Linear Signal-flow Graphs and Applications*, John Wiley & Sons, Inc., New York, 1962.

[18] S. J. Mason, "Feedback Theory - Some Properties of Signal Flow Graphs", *Proceedings of IRE*, Vol. 44, pp. 1144-1156, Sept. 1953.

[19] Douglas F. Elliott, *Handbook of Digital Signal Processing - Engineering Applications*, Academic Press, Inc., San Diego, California, 1987.

[20] Dan E. Dudgeon, Russell M. Mersereau, *Multidimensional Digital Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1984.

[21] V. Cappellini, A. G. Constantinides, P. Emiliani, *Digital Filters and Their Applications*, 1978

[22] Robert King, Majid Ahmadi, Raouf Gorgui-Naguib, Alan Kwabwe, Mahmood Azimi-Sadjadi, *Digital Filtering in One and Two Dimensions*, Plenum Press, New York, 1989.

[23] John G. Proakis, Dimitris G. Manolakis, *Digital Signal Processing - Principles, Algorithms, and Applications*, Macmillan Publishing Co., New York, 1992.

[24] Lawrence R. Rabiner, Bernard Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[25] A. H. Gray. Jr., J. D. Markel, "Digital Lattice and Ladder Filter Synthesis", *IEEE Transactions Audio Electroacoust.*, AU-21, pp. 491-500, Dec. 1973.

[26] A.Fettweis, "Digital Filter Structures Related to Classical Filter Networks", *Arch Elek*, Ubertrangung, 25, pp. 79-81, Feb. 1971.

[27] G. C. Temes, J. W. La Patra, *Circuit Synthesis and Design*, McGraw-Hill, New York, 1977.

[28] H. J. Orchard, "Inductorless Filters", *Electron. Letters*, pp. 224-225, Sept 1966.

[29] A. Fettweis, "Pseudopassivity, Sensitivity, and Stability of Wave Digital Filter", *IEEE Transactions on Circuit Theory*, Vol. 19, pp. 668-673, March 1975.

[30] H. T. Kung, "Why Systolic Architectures?", *Computer Magazine*, pp. 37-46, Jan. 1982.

[31] H. T. Kung, "Let's Design Algorithms for VLSI Systems", *Proc. Conf. Very Large Scale Integration: Architecture, Design, Fabrication*, California Institute of Technology, pp. 65-90, Jan. 1979.

[32] H. T. Kung, "Special-Purpose Devices for Signal and Image Processing: An Opportunity in VLSI", *Proc. SPIE*, Vol. 241, Real-Time Signal Processing III, Society of Photo-Optical Instrumentation Engineers, pp. 76-84, July 1980.

[33] P. R. Cappello, K. Steiglitz, "Digital Signal Processing Applications of Systolic Algorithms", in VLSI Systems and Computations, H. T. Kung, R. f. Sproull, and G. L. Steele, Jr. (eds.), Carnegie-Mellon University, Computer Science Press, pp. 265-272, Oct. 1981.

[34] G. Coschi, J. B. Schueler, *WATFOR-77 Language Reference - 4th Edition*, WATCOM Publications Limited, Waterloo, Canada, 1989.

[35] Norman W. Wilson, *VGAWAT Graphics Library Manual*, University of Windsor, Windsor, Canada, 1993.

[36] Steven E. Reyer, James A. Heinen, Russell J. Niederjohn, "A Simple Method for Determining the Computable Ordering of a Digital Network", *IEEE Trans. on Acoutics, Speech and Signal Processing*, Vol. ASSP-29, No.6, Dec. 1981.

[37] Charles Geschke, *Postscript Language Tutorial and Cookbook*, Adobe Systems, Inc., 1985.

[38] H. K. Kwan, *DINCAP User Guide*, 1981.

Appendix A

[1]Symbols and Formats of Common Netlist and Unique Netlist

A1.1 Symbols on 1-D Digital Networks

N Number of delay nodes.

M Number of signal nodes.

L Number of delay and signal Nodes.

IP Node number of external input.

LP Node number of network output.

NNZ Number of nonzero elements in entire STUV matrix.

NTP Number of points to be evaluated in Impulse Response Analysis.

NF/NFP Number of frequency points to be evaluated from 0 to 0.5 along the normalized frequency.

IN(I) Row number of a nonzero element of entire STUV matrix.

JN(I) Column number of a nonzero element of entire STUV matrix.

VAL(I) Numerical value of the IN(I) row and JN(I) column element of entire STUV matrix.

JK Number of nonzero and non-unity coefficients.

JY(I) Node number immediately before a nonzero and non-unity coefficients. If the

[1]Refer to references [9][38]

node number happens to be a delay node number, the node number of an equivalent signal node has to be used.

KY(I) Node number immediately after a nonzero and non-unity coefficient.

ACC Percentage improvement bound of noise variance analysis.

INFP Initial number of frequency points in Noise Analysis.

NB Power of 2 (i.e. 2^{-NB}) representing the quantization step of coefficients.

A1.2 Common Netlist of 1-D Digital Network

```
N     M      L      LP     NNZ
IN(1)        JN(1)        VAL(1)
 .            .            .
 .            .            .
 .            .            .
IN(NNZ)      JN(NNZ)      VAL(NNZ)
```

A1.3 Unique Nelist of 1-D Digital Network

Frequency Response

```
IP    NF
```

Group Delay and Slope of Magnitude Response

```
IP    NF
```

First Order Coefficient Sensitivity

```
IP    NF     JK
JY(1)        KY(1)
 .            .
 .            .
JY(JK)       KY(JK)
```

Noise Analysis

JK INFP ACC NB

KY(1)

.

.

KY(JK)

Impulse Response

IP NTP

A2.1 Symbols on 2-D Digital Network

N1	Number of delay nodes of first kind.
N2	Number of delay nodes of second kind.
M	Number of signal nodes.
L	Number of delay and signal nodes.
IP	Node of external input.
LP	Node number of network output.
NNZ	Number of nonzero elements in entire STUV matrix.
NTP1	Number of points to be evaluated along the first discrete time axis in Impulse Response Analysis.
NTP2	Number of points to be evaluated along the second discrete time axis in Impulse Response Analysis.
NF/NFP	Number of frequency points to be evaluated from -0.5 to 0.5 along the two normalized frequency axes.
IN(I)	Row number of a non-zero element of entire STUV matrix.
JN(I)	Column number of a nonzero element of entire STUV matrix.
VAL(I)	Numerical value of the IN(I) row and JN(I) column element of entire STUV matrix.
JK	Number of nonzero and non-unity coefficients.

JY(I) Node number immediately before a nonzero and non-unity coefficients. If the node number happens to be a delay node number, the node number of an equivalent signal node has to be used.

KY(I) Node number immediately after a nonzero and non-unity coefficients.

ACC Percentage improvement bound of noise variance analysis.

INFP Initial number of frequency points in Noise Analysis.

NB Power of 2 (i.e. 2^{-NB}) representing the quantization step of coefficients.

A2.2 Common Netlist of 2-D Digital Network

```
N1   N2     M      L      LP     NNZ
IN(1)       JN(1)         VAL(1)
 .           .             .
 .           .             .
 .           .             .
IN(NNZ)     JN(NNZ)       VAL(NNZ)
```

A2.3 Unique Nelist of 2-D Digital Network

Frequency Response

```
IP   NFP
```

Group Delay and Slope of Magnitude Response

```
IP   NFP
```

First Order Coefficient Sensitivity

```
IP   NF     JK
JY(1)       KY(1)
 .           .
 .           .
JY(JK)      KY(JK)
```

Noise Analysis

JK INFP ACC NB

KY(1)

.

.

KY(JK)

Impulse Response

IP NTP1 NTP2

A3.1 Symbols on 3-D Digital Network

N1	Number of delay nodes of first kind.
N2	Number of delay nodes of second kind.
N3	Number of delay nodes of third kind.
M	Number of signal nodes.
L	Number of delay and signal nodes.
IP	Node of external input.
LP	Node number of network output.
NNZ	Number of nonzero elements in entire STUV matrix.
NTP1	Number of points to be evaluated along the first discrete time axis in Impulse Response Analysis.
NTP2	Number of points to be evaluated along the second discrete time axis in Impulse Response Analysis.
NTP3	Number of points to be evaluated along the third discrete time axis in Impulse Response Analysis.
NF/NFP	Number of frequency points to be evaluated from -0.5 to 0.5 along the three normalized frequency axes.
IN(I)	Row number of a non-zero element of entire STUV matrix.
JN(I)	Column number of a nonzero element of entire STUV matrix.

VAL(I) Numerical value of the IN(I) row and JN(I) column element of entire STUV matrix.

JK Number of nonzero and non-unity coefficients.

JY(I) Node number immediately before a nonzero and non-unity coefficients. If the node number happens to be a delay node number, the node number of an equivalent signal node has to be used.

KY(I) Node number immediately after a nonzero and non-unity coefficients.

ACC Percentage improvement bound of noise variance analysis.

INFP Initial number of frequency points in Noise Analysis.

NB Power of 2 (i.e. 2^{-NB}) representing the quantization step of coefficients.

A3.2 Common Netlist of 3-D Digital Network

N1 N2 N3 M L LP NNZ

IN(1) JN(1) VAL(1)

. . .

. . .

. . .

IN(NNZ) JN(NNZ) VAL(NNZ)

A3.3 Unique Nelist of 3-D Digital Network

Frequency Response

IP NFP

Group Delay and Slope of Magnitude Response

IP NFP

First Order Coefficient Sensitivity

IP NF JK

JY(1) KY(1)

. .

. .

JY(JK) KY(JK)

Noise Analysis

JK INFP ACC NB

KY(1)

.

.

KY(JK)

Impulse Response

IP NTP1 NTP2 NTP3

Appendix B

User Manual of Signal Flow Graph

Schematic Drawing Software

DIGITAL NETWORK ANALYSIS AUTOMATION
(Signal Flow Graph)
Version 1.0

User Manual

System Specification
****

i. Computer Type : IBM AT or compatible.

ii. System : 80386 with math co-processor or higher.

iii. Video Adapter : Video Graphics Array or higher.

iV. Keyboard Type : IBM Enchaned (101- or 102-key) keyboard.

V. Mouse Type : Microsoft compatible mouse.

Note: Mouse must be installed in order to use the software properly. A harddrive called c: must be installed. The software will use the base memory (640K RAM).

Compile the program
****

The DNAA has total of 13 programs which are written in WATFOR and

its VGAWAT graphics library.

1. FLOW.FOR
2. EXT.FOR
3. DRAW_E.FOR
4. DCELL2.FOR
5. MULVAL.FOR
6. MENU2.FOR
7. SUBMEN2.FOR
8. DATAIO.FOR
9. CDATA.FOR
10. LINE.FOR
11. SELECT2.FOR
12. CELLB.FOR
13. EXDOS.FOR

A WATFOR compiler called 'WATCOMP.EXE'.

Add a command line into the 'autoexec.bat' file:

set library=c:\watfor\watfor;c:\watfor\vgawat2;

There is one on-line help file called 'HELP.HLP' and one logo file called 'LOGO.BLK'. In order to compile the programs, you have to include all the files in the same directory. At the prompt, type 'WATCOMP/EXE FLOW.FOR'. An execution file called 'FLOW.EXE' and a list file called 'FLOW.LST' will be produced. You can type 'FLOW' at the prompt in order to run the software. Note that the file 'HELP.HLP' and 'LOGO.BLK' should exist in the same directory of 'FLOW.EXE'.

After you run the software, some files will be created automatically when you click the menu or using file manager. Those files will be used internally by the software and will be stored into your c: drive root directory. The following is the files that will be created.

These files will be used for file manager.
1. JUNK.TXT
2. FILE.TXT
3. DIR.TXT
4. END.TXT

These files will be used for storing the block that was

hidden by the menus or windows.

1. ~TEMP.BLK
2. ~TMP2.BLK
3. ~TMP3.BLK
4. ~TMP4.BLK
5. ~TMP5.BLK
6. ~TMP6.BLK
7. ~TMP7.BLK

A function key (F1) can use to access the on-line help.

About This Software
****

To construct a 1-D, 2-D or 3-D digital network by using adder, delay and multiplier which is simply called signal flow graph. The software can extract the graphical representation of a digital network into netlist which can be used for design and simulation.

Vertical Elements Icon Bar
****

The Icon Bar is locate at the left side of the screen.

Using the left mouse button to click on the element in the Icon Bar that you want to drop on the working window, then a symbol of that element will follow your mouse location. Move the mouse to the working window and click the left button once. An element will then drop on the location that you want.

For the connecting lines, first click on the left mouse button that will locate the beginning point. Then move the mouse to the end point location and click on the left button again. The continues lines can be drawn by click on the left mouse button to locate the end point.

When finished drawing, you can click on the right button to deselect an element or line.

ScrollBar
****

There is one horizontal scrollbar and vertical scrollbar. Use the left button to click on the arrows, the graph will move left, right, up and down according to which arrow you you are pressing. You can also drag the button between the arrows of the vertical scrollbar or horizontal scrollbar to any location between the tracks, so that it can make faster movement of the view window.

Redraw Button
****

The button is located at the right side of the screen and right under the main menu. Click the left button once at the redraw button. This will redraw the whole working window.

Grid Button
****

The button is locate at the right side of the screen and right beside the redraw button. Click the left button once to enable the grid line if the current grid line is disable. Click the left button once to disable the grid line if the current grid line is enable.

Select
*****

Function Key : Alt+S

When performing delete, editing coefficient, rotate an element, look inside a cell, hide the inside element of a cell, move a group of elements and lines and paste a group of elements and lines, you have to use this function to select specific elements or lines first.

Using the left mouse button to click on the specific elements, lines or cells. You can select more than one element, line or cell. The selected elements, lines or cells will change to green color. You can only using select within the veiw window.

Using the right mouse button to deselect a selected element ,line or cell.

Select Area
****

Function Key : Alt+A

Same as select, but you can select a group of elements, lines or cells. Click on the left button to to locate one corner. Further movement of the mouse results in a "rubber band" rectangular box moving over the graphics diaplay area. Click on the left button again, and the elements, lines or cells that within the rectangle will be selected and change to green color.

You cannot deselect a selected element, line or cell by using "Select Area" function. But you can use the "Select" function to deselect an element, line or cell.

Delete
******

Function Key : Alt+D

Delete the selected elements, lines or cells.

Coefficient
****

Function Key : Alt+C

Editing the coefficients that you have selected. A small window will be shown on the lower left corner. If you have assigned a value, the value will show on the top of the small window. When you don't want to change the value, press <enter> and it will go to another selected multiplier. If you want to change it, you can type the new value at the dialogue box which is locate at the bottom of the small window and press <enter>. It also restrict you to type the numeric key only.

You can also change the coefficients of a cell by doing the same thing as above.

Rotate
******

A sub-menu will show beside the command "Rotate". It allows you to rotate the selected elements at a counter-clockwise direction of 90, 180, and 270 degrees. Click on the left button to select the specific angle.

Move To
****

Function Key : Alt+M

Move a group of the selected elements, lines or cells to a specific location. You can only move within the view window. Click on the left button at the command "Move To", and the whole group of the selected elements, lines or cells will follow your mouse location. Click on the left button again when the desired position is fixed.

The original position of the group of elements, lines and cells will be removed.

Use the right mouse button to cancel the command.

Paste To
****

Function Key : Alt+P

Paste a group of selected elements, lines or cells to a specific location. You can only paste within the view window. Click on the left button at the command "Move To", and the whole group of the selected elements, lines or cells will follow your mouse location. Click on the left button again when the desired position is fixed.

The original position of the group of elements, lines and cells will be retained.

Use the right mouse button to cancel the function.

Look Inside Cell
****

Function Key : Alt+L

Look what inside a selected cell. Click on the left button at the command "Look Inside Cell".

Hide Inside Cell
****

Function Key : Alt+H

Hide the inside elements and lines of a cell when the inside elements and lines are appeared on the screen. Click on the left button at the command "Hide Inside Cell".

Grid On/Off
****

Function Key : Alt+G

Click the left button once to enable the grid line if the current grid line is disable. Click the left button once to disable the grid line if the current grid line is enable.

Load Graph
****

A file manager will show on the screen. It can access up to D: drive. Directory will have a symbol "< >", click the left button at the specific directory. Then it will change to that directory. If you find the right file, click on that file using the left button and click on "O.K.". You can also type the filename at the dialogue box and press <enter>. You are also allow to use wildcards. Click on the dialog box locate at the top and type any wildcards to filter the filename and press <enter>. Press "Cancel" or press <Esc> to quit the file manager.

Load a graph on to the screen and the existing graph will be lost.

Save Graph
****

Save a graph to the file that you have already assigned.

Save as...
****

A file manager will show on the screen. It can access up to D: drive. Directory will have a symbol "< >", click the left button at the specific directory. Then it will change to that directory. If you find the right file, click on that file using the left button and click on "O.K.". You can also type the filename at the dialogue box and press <enter>. You are also allow to use wildcards. Click on the dialog box locate at the top and type any wildcards to filter the filename and press <enter>. Press "Cancel" or press <Esc> to quit the file manager.

Save a graph to a new filename or to another filename.

Edit Cell
****

A file manager will show on the screen. It can access up to D: drive. Directory will have a symbol "< >", click the left button at the specific directory. Then it will change to that directory. If you find the right file, click on that file using the left button and click on "O.K.". You can also type the filename at the dialogue box and press <enter>. You are also allow to use wildcards. Click on the dialog box locate at the top and type any wildcards to filter the filename and press <enter>. Press "Cancel" or press <Esc> to quit the file manager.

Load a cell from a file to the screen and change the architecture of the cell. Remember that you have to minimize the cell area by the minimization of the outer lines.

Get Cell
****

A file manager will show on the screen. It can access up to D: drive. Directory will have a symbol "< >", click the left

button at the specific directory. Then it will change to that directory. If you find the right file, click on that file using the left button and click on "O.K.". You can also type the filename at the dialogue box and press <enter>. You are also allow to use wildcards. Click on the dialog box locate at the top and type any wildcards to filter the filename and press <enter>. Press "Cancel" or press <Esc> to quit the file manager.

Get a cell from the file and the shape of the cell will follow your mouse position. Click on the left button when the desired position is fixed. Click on the right button to cancel the function.

Save as Cell
****

A file manager will show on the screen. It can access up to D: drive. Directory will have a symbol "< >", click the left button at the specific directory. Then it will change to that directory. If you find the right file, click on that file using the left button and click on "O.K.". You can also type the filename at the dialogue box and press <enter>. You are also allow to use wildcards. Click on the dialog box locate at the top and type any wildcards to filter the filename and press <enter>. Press "Cancel" or press <Esc> to quit the file manager.

After you finished editing an existing cell or a new cell, you can save the cell to the file by using this command. You have to use this command if you want to save it as a cell.

Exit
****

Function Key : Ctrl+X

A window will show on the screen. Click the "Yes" button or press the "y" key to terminate the software or click the "No" button or press the "n" key to ignore the quit.

C.E.C. (Connection Error Check)
****

Check the graph connection whether it is connected properly. A window will show on the bottom of the screen. When any connection error occurred, an error message will show on the dialogue box of that window and the error element or line will change to green color. You can choose "Change" button to correct an error or choose "Proceed" button to continue the error check or choose "Cancel" to close the window and return to the graph.

Netlist
******

It will also perform the "Connection Error Check". When no error occurred, a netlist will be shown on the screen. You can use the scrollbar at the right to scroll up or down. When you press the 'Save' button, a file manager will shown the screen. You can save the common data file in any filename.

After you have saved the common data file, a choice panel will show on the screen. Click on the left mouse button to choose any five of the unique data file and you can choose more than one type. The unique data file will be saved as the same name of the common data file but with different extension.

1. Frequency Response - extension (.FRQ)
2. Group Delay & Slope of Mag. Resp. - extension (.DSM)
3. First Order Coefficient Sensitivity - extension (.FCS)
4. Noise - extension (.NOI)
5. Impulse Response - extension (.IMP)

Fill Window
****

Function Key : Alt+F

Normally, the size of the graph will be excceed the viewing window. This command will make a zoom out in order to view everything that you have drawn on the screen.

Zoom In
****

Function Key : Alt+I

Zoom in the graph.

Zoom Out
****

Function Key : Alt+O

Zoom out the graph.

Set Colors
****

A window will show on the screen. It allows you to change the foreground and background colors of the viewing window. Choose the desired color under the "background" and click on the left button. Then the specific color will be assigned. Doing the same thing under the "foreground". Press "O.K." or type <enter> to set the colors that you have changed. Press "Cancel" or press <Esc> to ignore the colors that you have changed.

Clear Window
****

Function Key : Ctrl+C

A window will show on the screen. Click the "Yes" button or press the "y" key to clear the window or click the "No" button or press the "n" key to ignore the command. Remember that the existing graph will be lost.

Appendix C

User Manual of Digital Network Analysis Software

DIGITAL NETWORK DESIGN AUTOMATION
(Network Analysis)
Version 1.0

User Manual

System Specification
****

i. Computer Type : IBM AT or compatible.

ii. System : 80386 with math co-processor or higher.

iii. Video Adapter : Video Graphics Array or higher.

iV. Keyboard Type : IBM Enchaned (101- or 102-key) keyboard.

V. Mouse Type : Microsoft compatible mouse.

Note: Mouse must be installed in order to use the software properly. A harddrive called c: must be installed. The software will use the base memory (640K RAM).

Compile the program
****

The DNDA (Network Analysis) has total of 4 programs which are

written in WATFOR and its VGAWAT graphics library.

1. SFGGRA.FOR
2. 2DPLOT.FOR
3. 3DPLOT.FOR
4. EXT.FOR

There is also 15 executable files which are written in Fortran 77 and use for calculate the output of different analysis.

1-D digital network analysis:

1. FR1DPMR.EXE
2. G1DPMR.EXE
3. S1DPMR.EXE
4. N1DPMR.EXE
5. IR1D2.EXE

2-D digital network analysis:

1. FR2DPMR.EXE
2. G2DPMR.EXE
3. S2DPMR.EXE
4. N2DPMR.EXE
5. IR2D2.EXE

3-D digital network analysis:

1. FR3DPMR.EXE
2. G3DPMR.EXE
3. S3DPMR.EXE
4. *N3DPMR.EXE
5. IR3D2.EXE

*-file is not working yet.

A WATFOR compiler called 'WATCOMP.EXE'.

Add a command line into the 'autoexec.bat' file:

set library=c:\watfor\watfor;c:\watfor\vgawat2;

In order to compile the programs, you have to include all the files (include executable files and one 'sfglogo.blk' file) in

the same directory. At the prompt, type 'WATCOMP/EXE SFGGRA.FOR'. An execution file called 'SFGGRA.EXE' and a list file called 'SFGGRA.LST' will be produced. You can type 'SFGGRA' at the prompt in order to run the software.

After you run the software, some files will be created automatically when you click the menu or using file manager. Those files will be used internally by the software and will be stored into your c: drive root directory. The following is the files that will be created.

These files will be used for file manager.
1. JUNK.TXT
2. FILE.TXT
3. DIR.TXT
4. END.TXT

These files will be used for storing the block that was hidden by the menus or windows.

1. ~TMP4.BLK
2. ~TMP7.BLK

About This Software
****

This software will use the netlist created by the software "Digital Network Design Automation (Signal Flow Graph)" and analyse the 1-D, 2-D and 3-D digital networks with Frequency Response, Group Delay and Slope of Magnitude Response, First Order Coefficient Sensitivity, Noise and Impulse Response.

Input Parameters
****

Dimension : Select the box for 1-D, 2-D or 3-D digital network and click on the left mouse button.

Netlist : A file manager will display. Choose a common netlist file that was created by the software "DNDA (Signal Flow Graph)" and click on "O.K." or press <enter>. The dimension of the netlist file must be match with the dimension that you have chosen, otherwise, an error may occur.

Type of
Analysis : An analysis choice window will display on the screen.
Choose any one of the analysis:
1. Frequency Response
2. Group Delay and Slope of Magnitude Response
3. First Order Coefficient Sensitivity
4. Noise
5. Impluse Response
Then press "O.K." for the correct one or press "Cancel" to quit the analysis choice window.

Command

Run : Run the selected analysis with selected dimension of the digital network. The button will shaded when the calculation is in progress.

Quit : Quit the software and return to DOS. (Ctrl+X)

Options

Grid : Enable or Disable grid lines in 2-D graph.
Enable or Disable grid lines and numeric values in 3-D graph.

Save : Save the output data of the analysis.

2-D Graph Window

The software can plot two graphs in the two different graph window at the same analysis. It can only plot the graph for the 1-D digital network.

Numerical Display

Move the mouse button within any of the two graph window, the relative numerical value will display at the message box under the "OPTION" panel.

Set Range

Within the graph window, click on the left mouse button to locate the begining point. When moving the mouse, a rectangular shape will follow your mouse. Click on the left mouse button again, the graph will zoom to the area inside the rectangle. You can plot the whole range by setting the beginning point at the upper left corner and setting the end point at the bottom right corner. That means the rectangular shape will cover the whole graph window.

Logarithm Plot

At the Top graph window, Click on the "LOG" button to show the logarithm of the Magnitude Response in Frequency Response Analysis.

Coefficient Sensitivity Plot

When first order coefficient sensitivity analysis was selected, the top graph window will display the graph of the first coefficient according to your netlist. The coefficient value will display on the top of the graph window. Click on the left button at the arrow and select other coefficients. The graph will be updated according to the selected coefficient.

3-D Graph Window

Plot the graph of 2-D or 3-D digital network.

When 2-D digital network was selected, the software will display one graph window, buttons for the choice of graph in the same analysis at the right and rotation angle, elevation angle, set range button at the bottom.
When 3-D digital network was selected, the software has the same display as in 2-D digital network with addition of selecting third frequency axis (f3).

Common Commands for 2-D and 3-D digital network

Rotation and Elevation

There is two integer value at the bottom. They are the angle for rotation and elevation. Click on the left button at the arrow

pointing up to increase the angle or click on the left button at the arrow pointing down to decrease the angle. You can also click on the left button at the box to input the angle using keyboard and press <enter>. After you have defined the rotation and elevation angle, click on the "Plot" button to update the graph.

"Zoom" button

This button will set the range of the 3-D graph. A window will display on the screen. There is four numeric input, two for the first frequency axis (f1) and two for the second frequency axis (f2). Input the range of f1 and f2, then the graph will update to range that you have already defined.

Choice of graph in 2-D digital network

Frequency Response

The graph window will display the magnitude response when you press "Run" and the "Mag." button will high-lite at the choice of graph panel at the right. You can select "Log" or "Phase" at the choice of graph panel and the graph window will be updated.

Group Delay and Slope of Magnitude Response

The graph will display the group delay with respect to f1 when you press "Run", the "Group" button and "f1" button will high-lite at the choice of graph panel.

Button pressed:
Group and f1 : Group Delay with respect to f1
Slope and f1 : Slope of Magnitude with respect to f1
Group and f2 : Group Delay with respect to f2
Slope and f2 : Slope of Magnitude with respect to f2

First Order Coefficient Sensitivity

The graph window will display the graph of the first coefficient according to your netlist. The coefficient value will display on the top of the graph window. Click on the left button at the arrow and select other coefficients. The graph will be updated according

to the selected coefficient.

Choice of graph in 3-D digital network
****

In 3-D digital network, there is three frequency axis to be plot. In 3-D graph, it can only show two of the axis. Therefore, one of the frequency will be fixed at one time. At the middle of the choice of graph panel, when you press at the arrow, the third frequency will be increase or decrease and the graph will be updated to that frequency. Every time when you press "Run", the third frequency will be set to -0.495.

Frequency Response

The graph window will display the magnitude response when you press "Run" and the "Mag." button will high-lite at the choice of graph panel at the right. You can select "Log" or "Phase" at the choice of graph panel and the graph window will be updated.

Group Delay and Slope of Magnitude Response

The graph will display the group delay with respect to f1 when you press "Run", the "Group" button and "f1" button will high-lite at the choice of graph panel.

Button pressed:
Group and f1 : Group Delay with respect to f1
Slope and f1 : Slope of Magnitude with respect to f1
Group and f2 : Group Delay with respect to f2
Slope and f2 : Slope of Magnitude with respect to f2
Group and f3 : Group Delay with respect to f3
Slope and f3 : Slope of Magnitude with respect to f3

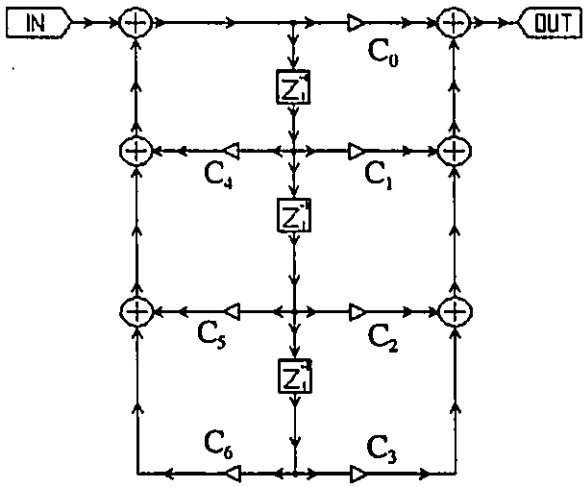
First Order Coefficient Sensitivity

The graph window will display the graph of the first coefficient according to your netlist. The coefficient value will display on the top of the graph window. Click on the left button at the arrow and select other coefficients. The graph will be updated according to the selected coefficient.

Appendix D

Examples of Analyzing 1-D Digital Networks

D1 Direct-Form II of Third-Order IIR Digital Filter



C_0=0.05634, C_1=-0.000935, C_2=-0.000935, C_3=0.05634, C_4=2.1291, C_5=-1.7839, C_6=0.54346

Common Netlist

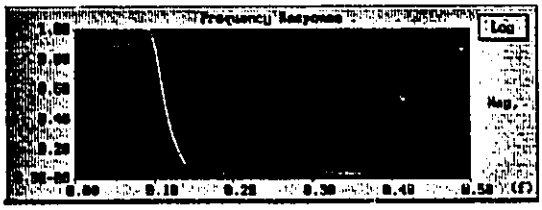
```
3       5       8       8       13
 1       7      1.000000000000000
 4       1      1.000000000000000
```

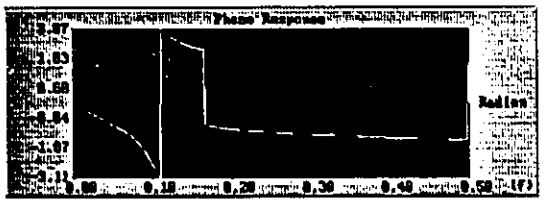
7	4	2.129100000000000
8	4	-0.000935000000000
2	4	1.000000000000000
5	2	1.000000000000000
7	5	-1.783900000000000
8	5	-0.000935000000000
6	3	1.000000000000000
7	6	0.543460000000000
8	6	0.056340000000000
3	5	1.000000000000000
8	7	0.056340000000000

Frequency Response Analysis

Unique Netlist

7 201 0

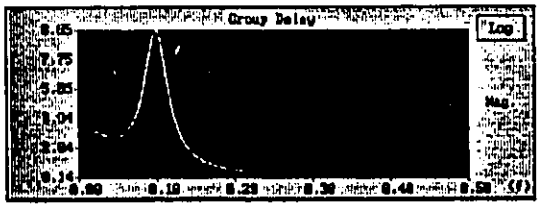


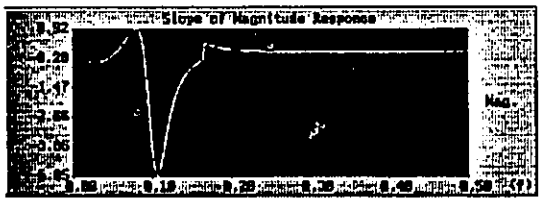


Group Delay and Slope of Magnitude Response Analysis

Unique Netlist

7 201



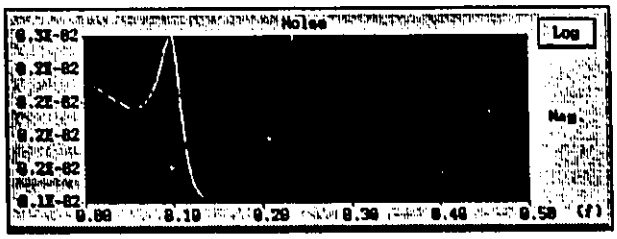


Noise Analysis

Unique Netlist

7	201	0.1000000	4
7			
8			
7			
8			

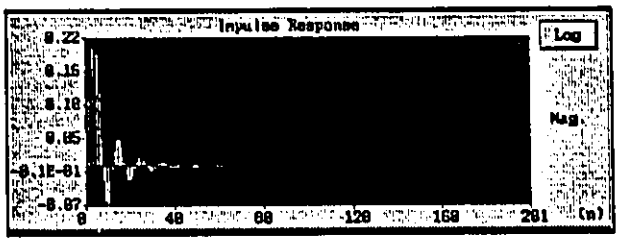
7
8
8



Impulse Response Analysis

Unique Netlist

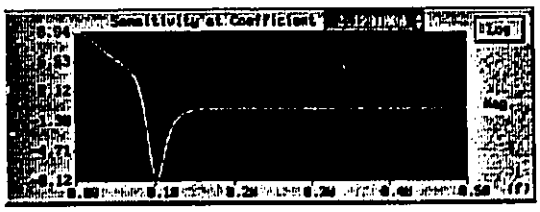
7 201

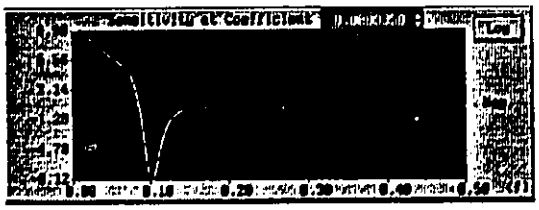


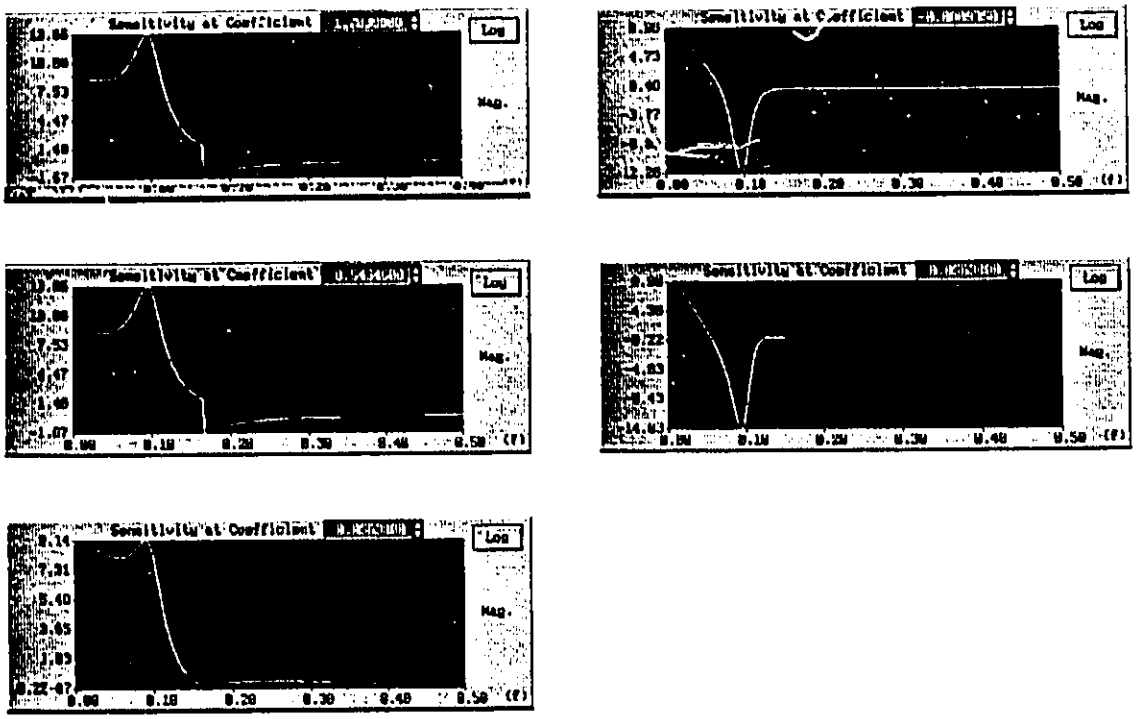
First Order Corfficient Sensitivity Analysis

Unique Netlist

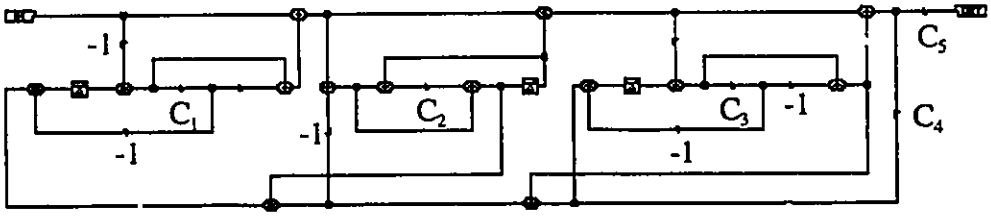
7	201	7
4	7	
4	8	
5	7	
5	8	
6	7	
6	8	
7	8	







D2 Third-Order VW Linear Transformed 1-D Chebyshev Low-Pass Digital Filter



C_1=0.4922, C_2=-0.3002, C_3=0.3716, C_4=0.2429, C_5=1.242772

Common Netlist

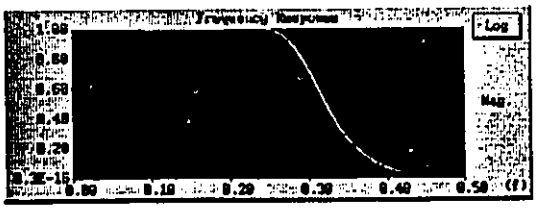
```
3       17      20      20      33
 1       18     1.000000000000000
 5        1     1.000000000000000
 5        4    -1.000000000000000
 6        5     0.492200000000000
 7        5     1.000000000000000
 7        6    -1.000000000000000
18        6    -1.000000000000000
```

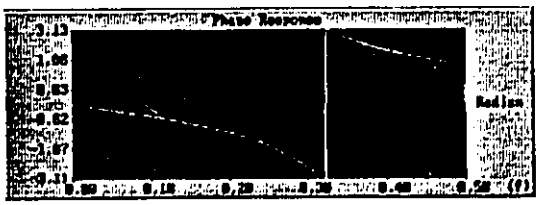
7	4	1.000000000000000
15	7	1.000000000000000
17	16	-0.300200000000000
16	15	1.000000000000000
17	15	1.000000000000000
2	17	1.000000000000000
8	2	1.000000000000000
16	2	1.000000000000000
8	7	1.000000000000000
18	17	1.000000000000000
18	14	1.000000000000000
15	14	-1.000000000000000
3	19	1.000000000000000
9	3	1.000000000000000
9	8	-1.000000000000000
10	9	0.371600000000000
11	9	1.000000000000000
11	10	-1.000000000000000
19	10	-1.000000000000000
12	11	1.000000000000000
12	8	1.000000000000000
14	11	1.000000000000000
13	12	0.242900000000000
20	12	1.242771846000000
14	13	1.000000000000000
19	13	1.000000000000000

Frequency Response Analysis

Unique Netlist

4 201 0

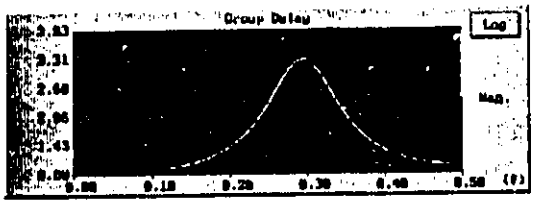


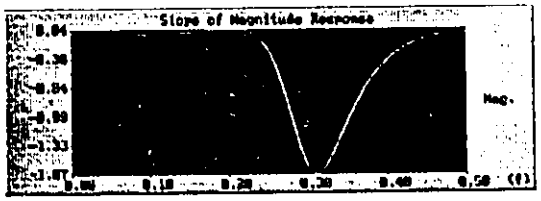


Group Delay and Slope of Magnitude Response Analysis

Unique Netlist

4 201

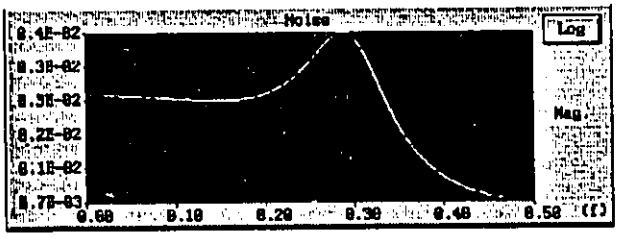




Noise Analysis

Unique Netlist

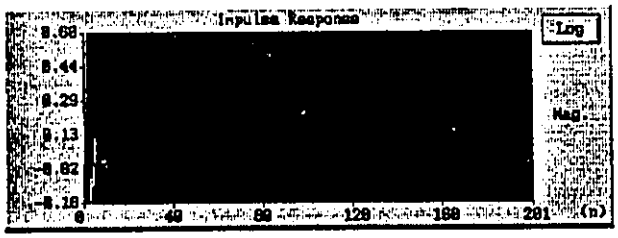
5	201	0.1000000	4
6			
17			
10			
13			
20			



Impulse Response Analysis

Unique Netlist

4	201

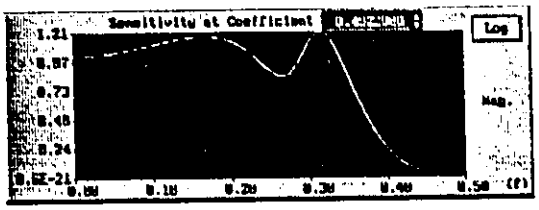
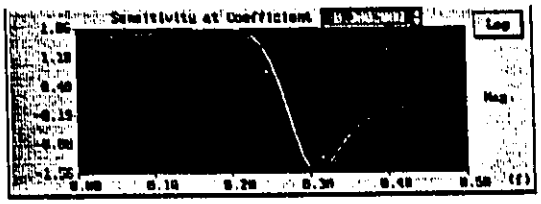
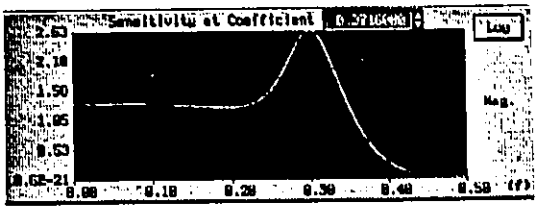
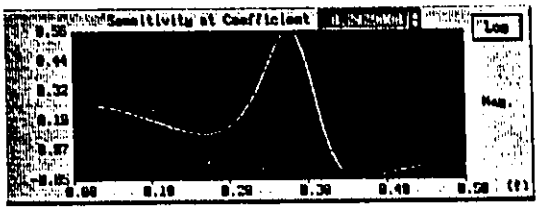
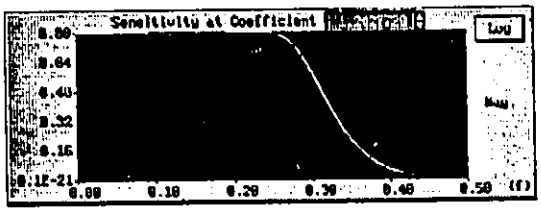


First Order Coefficient Sensitivity Analysis

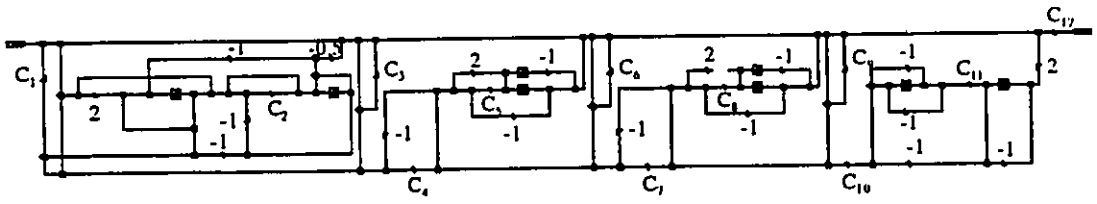
Unique Netlist

4	201	5
5	6	
16	17	

9 10
12 13
12 20

D3 Eighth-Order FDNPR in MTA Linear-Transformed 1-D Band-Pass Digital Filter



C_1=-0.107350, C_2=-0.018286, C_3=-0.935354, C_4=0.797202, C_5=0.544128, C_6=-0.884325, C_7=0.867268, C_8=0.372961, C_9=-0.948672, C_{10}=0.558206, C_{11}=0.806316, C_{12}=

Common Netlist

8 35 43 43 73
1 32 1.000000000000000

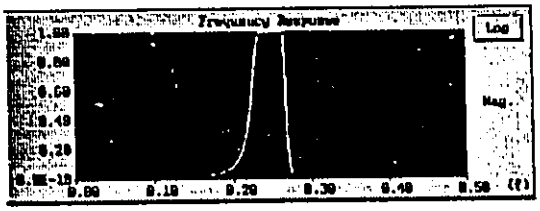
32	1	1.000000000000000
28	1	1.000000000000000
32	27	2.000000000000000
28	27	1.000000000000000
30	29	-0.018286000000000
25	1	1.000000000000000
29	28	1.000000000000000
30	28	1.000000000000000
9	2	1.000000000000000
25	9	-1.000000000000000
29	9	-1.000000000000000
31	9	1.000000000000000
2	30	1.000000000000000
31	30	1.000000000000000
33	31	-0.500000000000000
26	25	-0.107350000000000
34	33	1.000000000000000
34	24	1.000000000000000
25	24	1.000000000000000
35	34	-0.935354000000000
35	33	1.000000000000000
10	4	1.000000000000000
11	10	0.544128000000000
3	10	1.000000000000000
12	10	2.000000000000000
36	3	-1.000000000000000
4	36	1.000000000000000
12	11	1.000000000000000
36	11	-1.000000000000000
24	12	-1.000000000000000
36	35	1.000000000000000
24	23	0.797202000000000
23	12	1.000000000000000
38	37	-0.884325000000000
13	6	1.000000000000000
14	13	0.372961000000000
5	13	1.000000000000000
15	13	2.000000000000000
39	5	-1.000000000000000
6	39	1.000000000000000
15	14	1.000000000000000
39	14	-1.000000000000000
22	15	-1.000000000000000
39	38	1.000000000000000

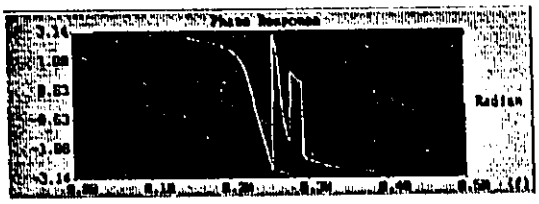
22	21	0.867268000000000
21	15	1.000000000000000
38	35	1.000000000000000
37	35	1.000000000000000
23	22	1.000000000000000

Frequency Response Analysis

Unique Netlist

26 201 0

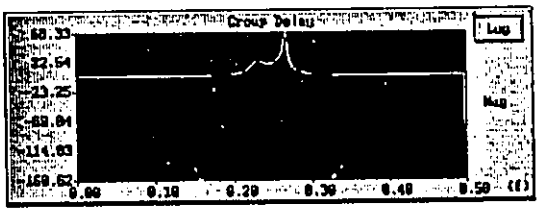


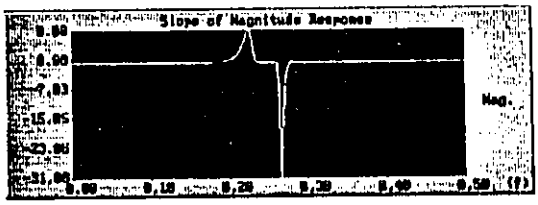


Group Delay and Slope of Magnitude Response Analysis

Unique Netlist

26 201



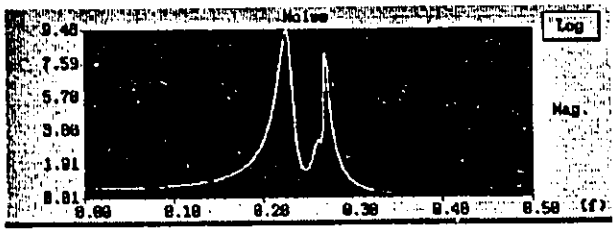


Noise Analysis

Unique Netlist

17	201	0.1000000	4
32			
30			
33			
26			
35			
11			
12			
24			
38			
14			

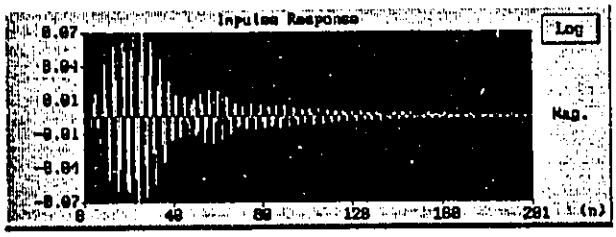
15
22
41
18
42
43
20



Impulse Response Analysis

Unique Netlist

26 201

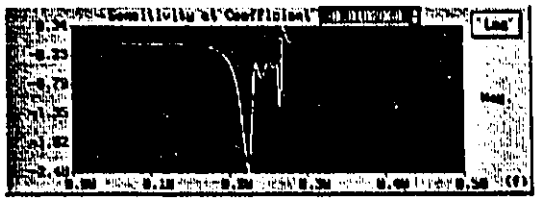
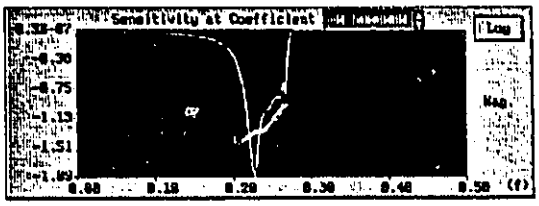
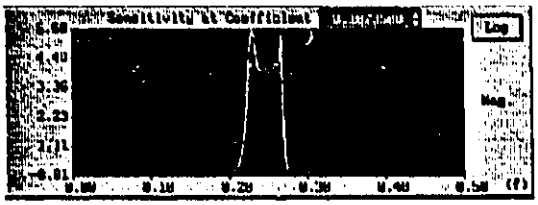
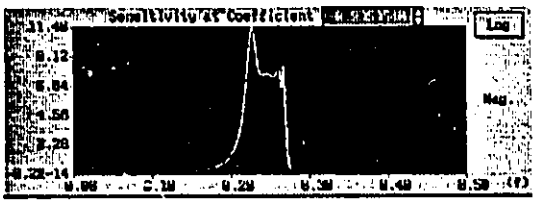
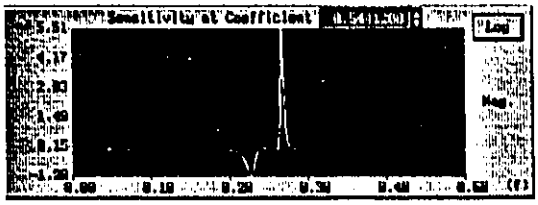
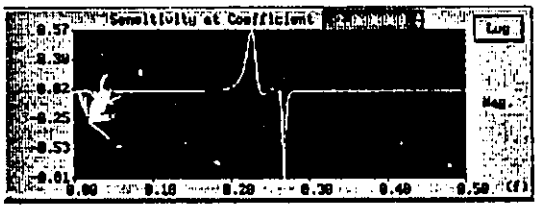
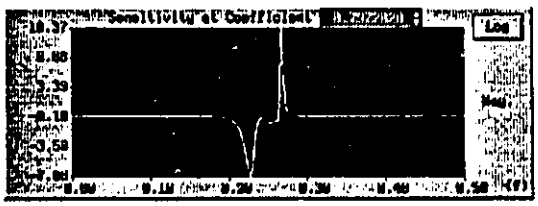
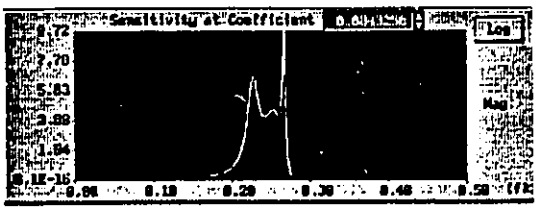
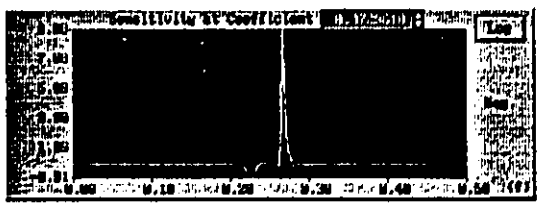


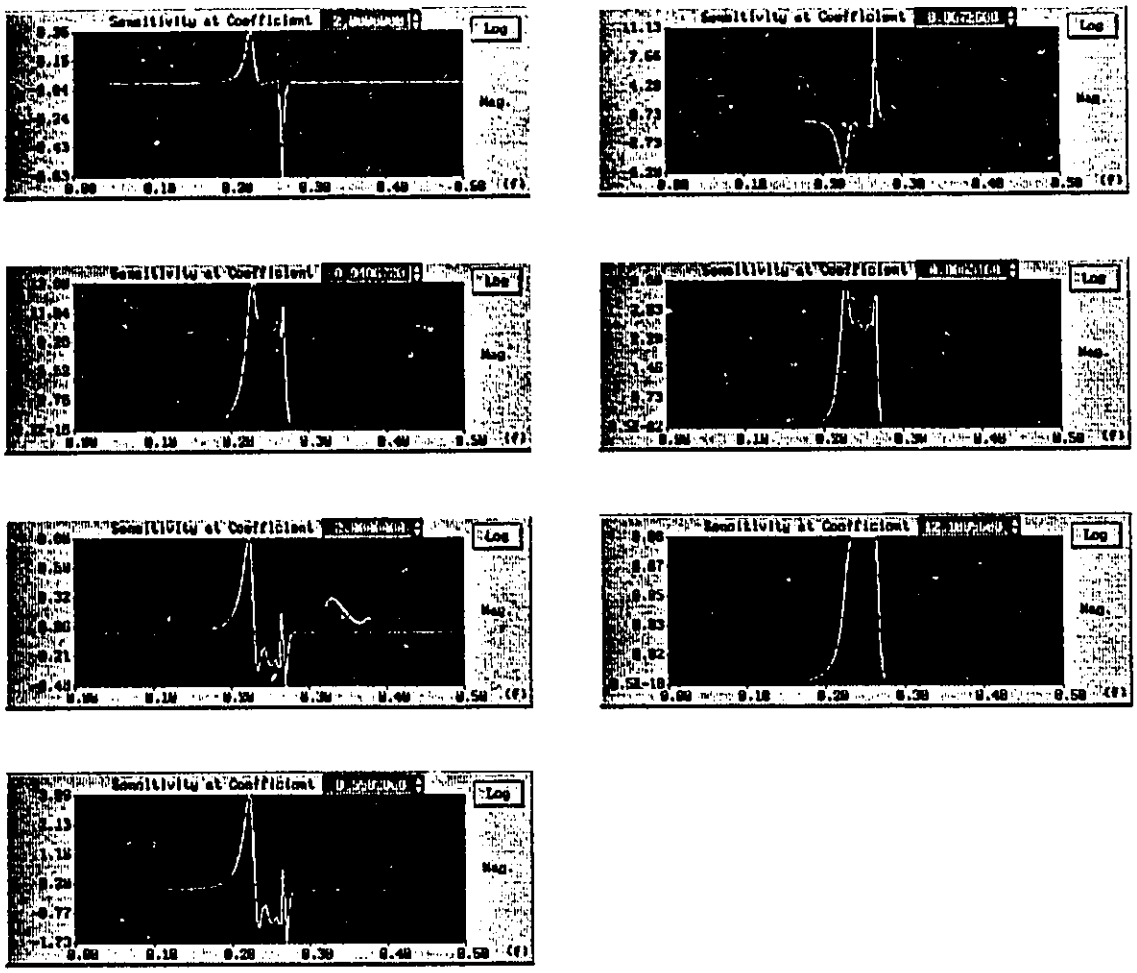
First Order Coefficient Sensitivity Analysis

Unique Netlist

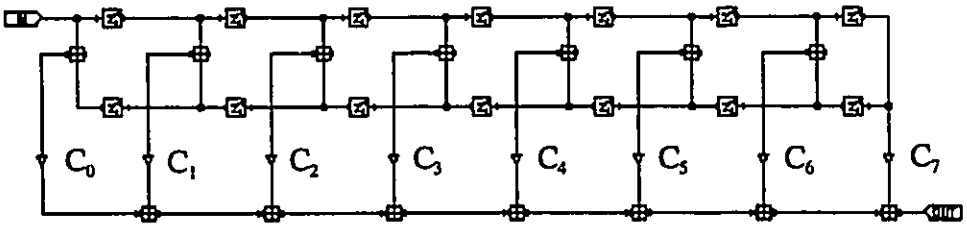
26	201	17
27	32	
29	30	
31	33	
25	26	
34	35	
10	11	
10	12	
23	24	
37	38	
13	14	
13	15	
21	22	
40	41	

17 18
41 42
41 43
19 20



D4 Linear-Phase FIR Digital Filter of Length N=15



C_0=-0.014112893, C_1=-0.001945309, C_2=0.04000004, C_3=0.01223454, C_4=-0.09138802, C_5=-0.01808986, C_6=0.3133176, C_7=0.52

Common Netlist

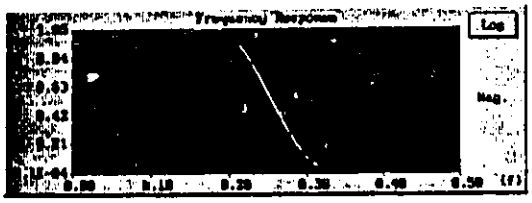
```
14      10      24      24      37
23      2       1.000000000000000
```

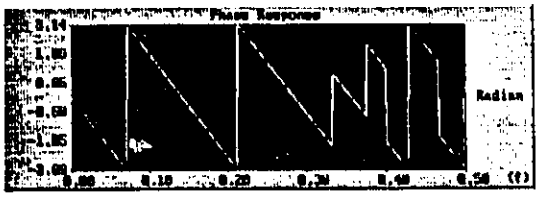
1	23	1.000000000000000
2	3	1.000000000000000
15	3	1.000000000000000
3	4	1.000000000000000
16	4	1.000000000000000
4	5	1.000000000000000
17	5	1.000000000000000
5	6	1.000000000000000
18	6	1.000000000000000
6	7	1.000000000000000
19	7	1.000000000000000
7	8	1.000000000000000
20	8	1.000000000000000
8	21	1.000000000000000
22	21	1.000000000000000
9	10	1.000000000000000
20	10	1.000000000000000
10	11	1.000000000000000
19	11	1.000000000000000
11	12	1.000000000000000
18	12	1.000000000000000
12	13	1.000000000000000
17	13	1.000000000000000
13	14	1.000000000000000
16	14	1.000000000000000
14	1	1.000000000000000
15	1	1.000000000000000
22	9	1.000000000000000
24	22	-0.014112893000000
24	20	-0.001945309000000
24	19	0.040000040000000
24	18	0.012234540000000
24	17	-0.091388020000000
24	16	-0.018089860000000
24	15	0.313317600000000
24	23	0.520000000000000

Frequency Response Analysis

Unique Netlist

21	201	0

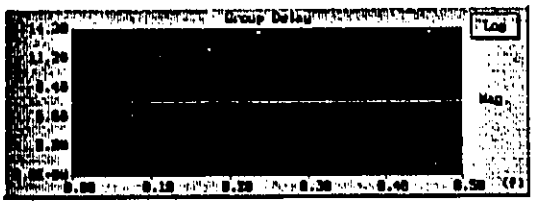




Group Delay and Slope of Magnitude Response Analysis

Unique Netlist

21 201

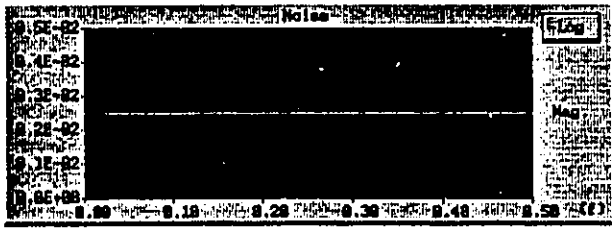


Noise Analysis

Unique Netlist

8 201 0.1000000 4

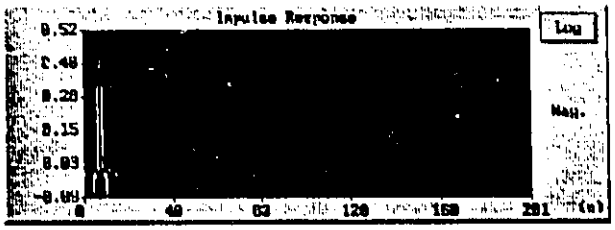
24
24
24
24
24
24
24
24



Impulse Repsonse Analysis

Unique Netlist

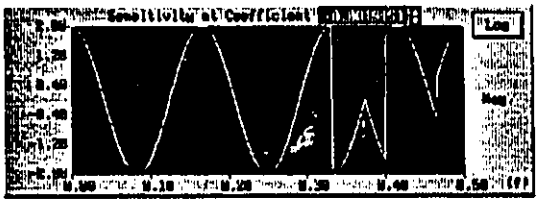
21 201

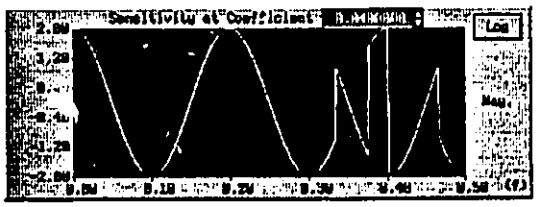


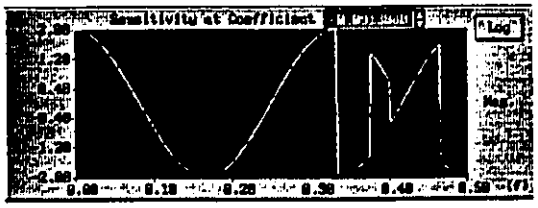
First Order Coefficient Analysis

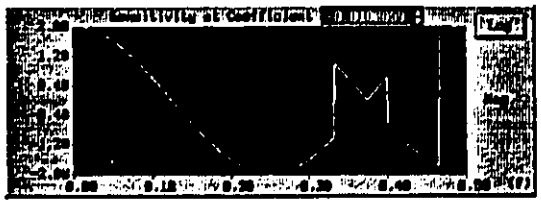
Unique Netlist

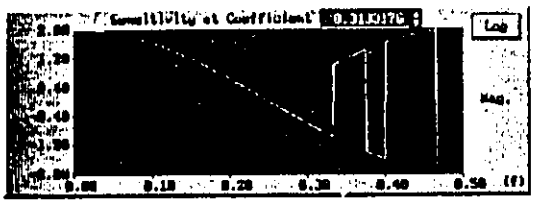
21	201	8
22	24	
20	24	
19	24	
18	24	
17	24	
16	24	
15	24	
23	24	

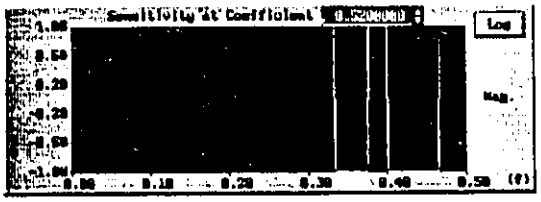




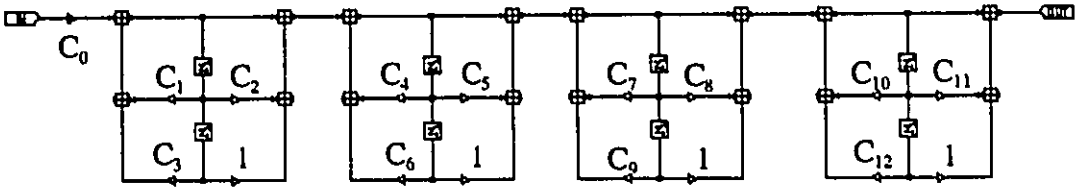








D5 Cascade-Form of Band-Pass Digital Filter



C_0=0.005656462366, C_1=0.2855823838, C_2=0.4512591755, C_3=-0.9116860616, C_4=0.4457342317, C_5=-1.09869455, C_6=-0.913078673, C_7=0.196901665, C_8=-0.0099665157, C_9=-0.9695353354, C_{10}=0.5515388641, C_{11}=-0.7304169706, C_{12}=-0.9705899009

Common Netlist

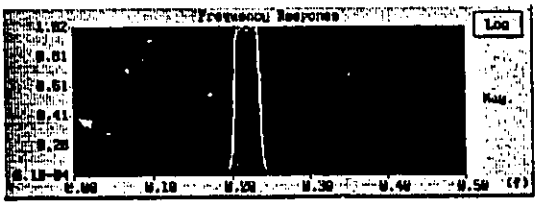
8	14	22	22	37
1	18	1.000000000000000		
9	1	1.000000000000000		
2	9	1.000000000000000		
18	9	0.285582383800000		
19	9	0.451259175500000		
10	2	1.000000000000000		
18	10	-0.911686061600000		
19	10	1.000000000000000		
19	18	1.000000000000000		
3	19	1.000000000000000		
11	3	1.000000000000000		
4	11	1.000000000000000		
19	11	0.445734231700000		
20	11	-1.098694550000000		
12	4	1.000000000000000		
19	12	-0.913078673000000		
20	12	1.000000000000000		
20	19	1.000000000000000		
5	20	1.000000000000000		
13	5	1.000000000000000		
6	13	1.000000000000000		

20	13	0.196901665000000
21	13	-0.009966515700000
14	6	1.000000000000000
20	14	-0.969535335400000
21	14	1.000000000000000
21	20	1.000000000000000
7	21	1.000000000000000
15	7	1.000000000000000
8	15	1.000000000000000
21	15	0.551538864100000
22	15	-0.730416970600000
16	8	1.000000000000000
21	16	-0.970589900900000
22	16	1.000000000000000
22	21	1.000000000000000
18	17	0.005656462400000

Frequency Response Analysis

Unique Netlist

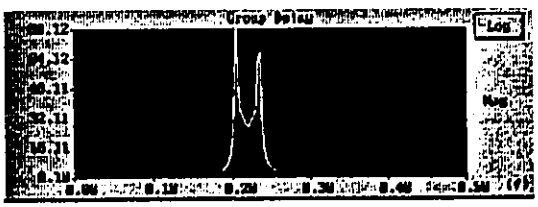
17 201 0

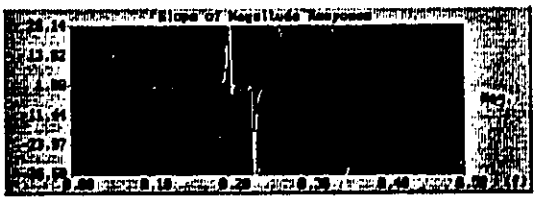


Group Delay and Slope of Magnitude Response Analysis

Unique Netlist

17 201

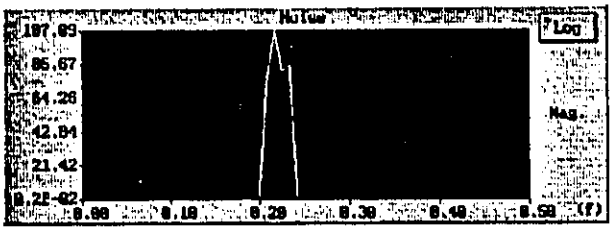




Noise Analysis

Unique Netlist

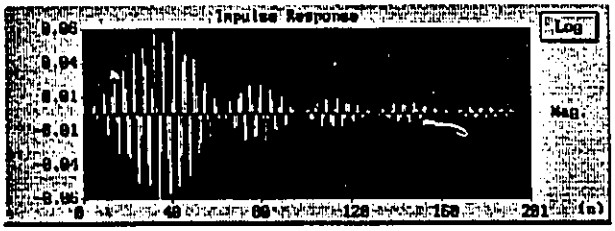
7	201	0.1000000	4
7			
8			
7			
8			
7			
8			
8			



Impulse Response Analysis

Unique Netlist

17	201



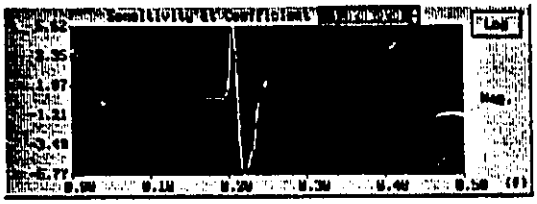
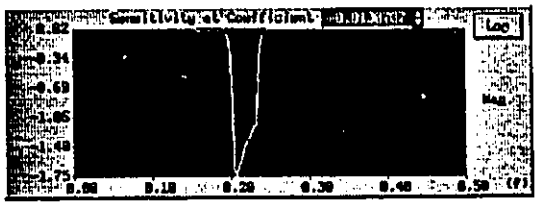
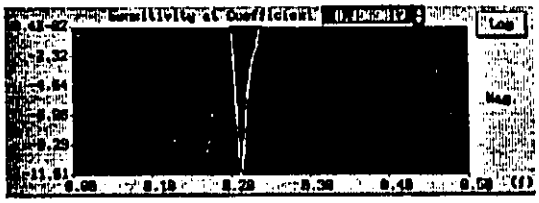
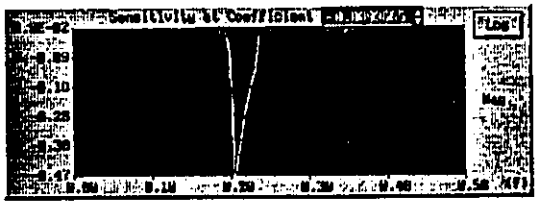
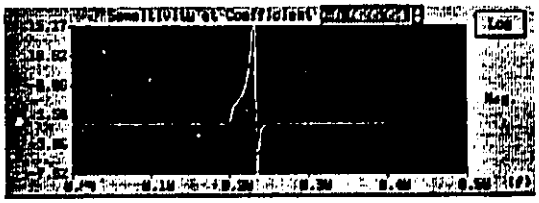
First Order Coefficient Sensitivity Analysis

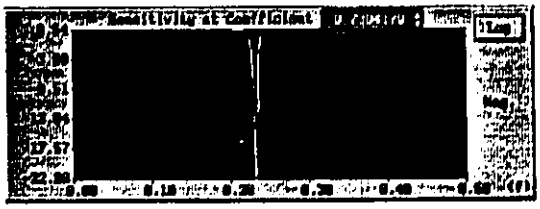
Unique Netlist

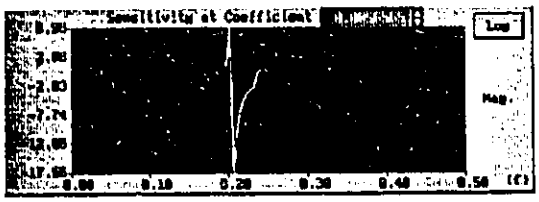
17	201	17
9	18	
9	19	
10	18	
10	19	
11	19	
11	20	
12	19	
12	20	

13	20
13	21
14	20
14	21
15	21
15	22
16	21
16	22
17	18

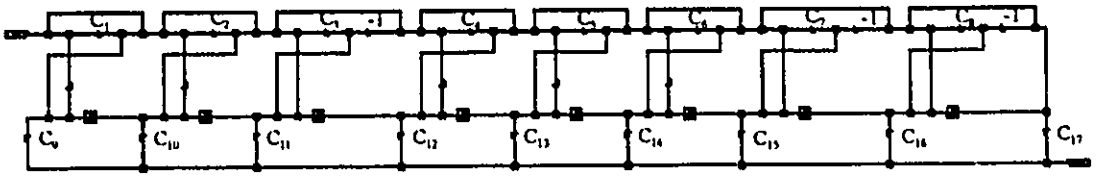






D6 Ladder Structure of Band-Pass Digital Filter by Gray and Markel



C_1=0.7833447265, C_2=-0.1885428229, C_3=0.9843741951, C_4=-0.1920962931, C_5=0.9936376827, C_6=-0.1890252997, C_7=0.9914182038, C_8=-0.1964790194, C_9=0.005656462366, C_{10}=0.0002916124024, C_{11}=-0.001699657657, C_{12}=-0.06980454614, C_{13}=0.2031623313, C_{14}=0.01061747099, C_{15}=-0.01641846145, C_{16}=-1.031187912, C_{17}=0.8446328267

Common Netlist

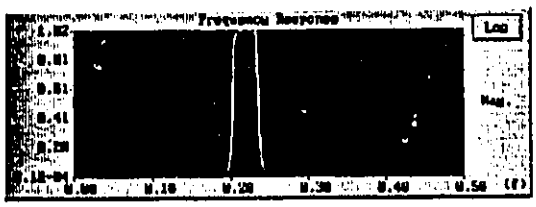
8	39	47 47 78
12	11	0.783344726500000
11	9	1.000000000000000
14	9	1.000000000000000
14	12	1.000000000000000
13	12	1.000000000000000
13	10	1.000000000000000
11	10	-1.000000000000000
10	8	1.000000000000000
47	13	0.005656462400000
17	16	-0.188542822900000
16	14	1.000000000000000
18	14	1.000000000000000
18	17	1.000000000000000
45	17	1.000000000000000
45	15	1.000000000000000
16	15	-1.000000000000000
15	7	1.000000000000000
47	45	2.916124000000000D-004
20	19	0.984374195100000
19	18	1.000000000000000

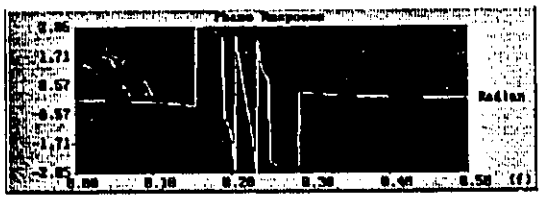
21	20	-1.000000000000000
44	20	1.000000000000000
44	6	1.000000000000000
47	44	-0.001699657700000
24	23	-0.192096293100000
23	21	1.000000000000000
25	21	1.000000000000000
25	24	1.000000000000000
43	24	1.000000000000000
43	22	1.000000000000000
23	22	-1.000000000000000
22	5	1.000000000000000
47	43	-0.069804546100000
21	18	1.000000000000000
19	6	1.000000000000000
28	27	0.993637682700000
27	25	1.000000000000000
29	25	1.000000000000000
29	28	1.000000000000000
42	28	1.000000000000000
42	26	1.000000000000000
27	26	-1.000000000000000
26	4	1.000000000000000
47	42	0.203162331300000
32	31	-0.189025299700000
31	29	1.000000000000000
33	29	1.000000000000000
33	32	1.000000000000000
41	32	1.000000000000000
41	30	1.000000000000000
31	30	-1.000000000000000
30	3	1.000000000000000
47	41	0.010617471000000
35	34	0.991418203800000
34	33	1.000000000000000
36	35	-1.000000000000000
40	35	1.000000000000000
40	2	1.000000000000000
34	2	1.000000000000000
36	33	1.000000000000000

Frequency Response Analysis

Unique Netlist

9 201 0

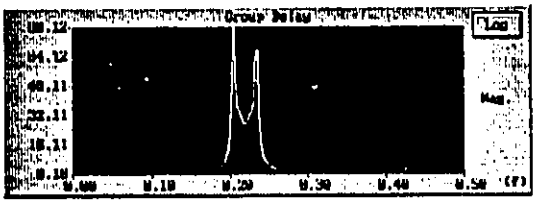


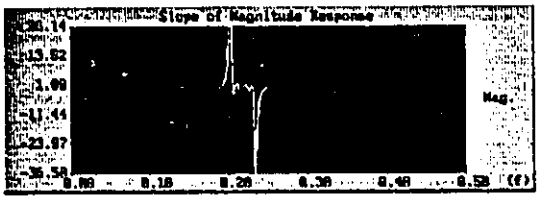


Group Delay and Slope of Magnitude Response Analysis

Unique Netlist

9 201





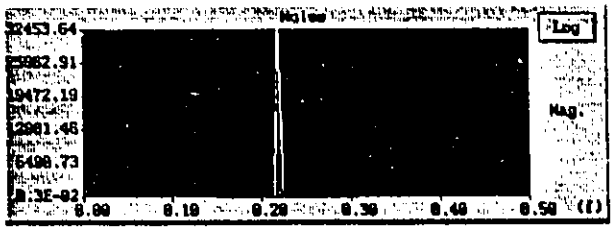
Noise Analysis

Unique Netlist

17 101 0.5000000 4

12
47
17
47
20
47
24
47
28
47
32
47
35
47
38
47

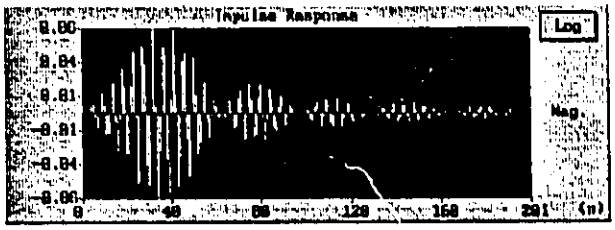
47



Impulse Response Analysis

Unique Netlist

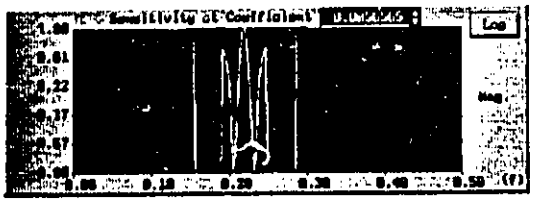
9 201

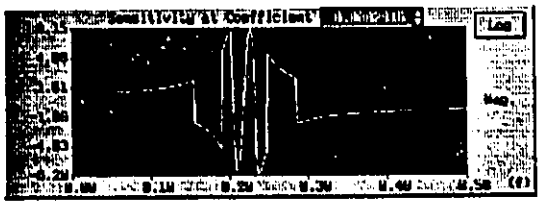


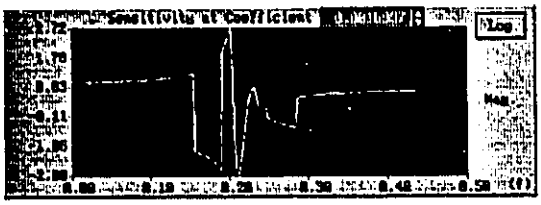
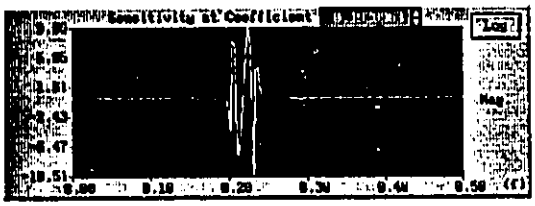
First Order Coefficient Sensitivity Analysis

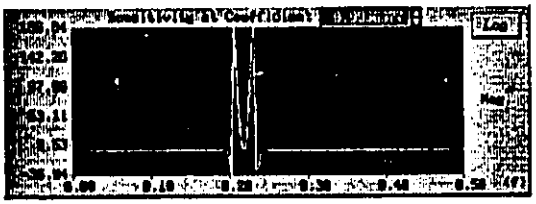
Unique Netlist

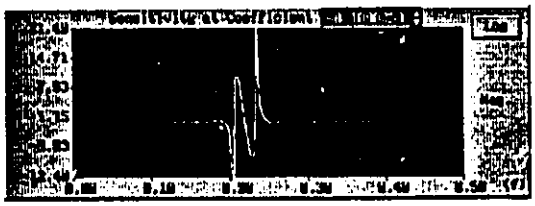
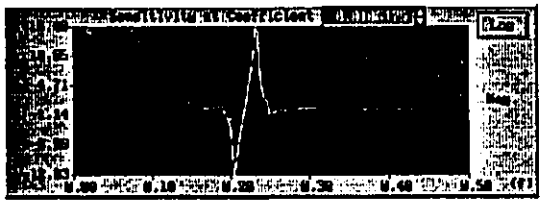
9	201	17
11	12	
13	47	
16	17	
45	47	
19	20	
44	47	
23	24	
43	47	
27	28	
42	47	
31	32	
41	47	
34	35	
40	47	
37	38	
39	47	
46	47	

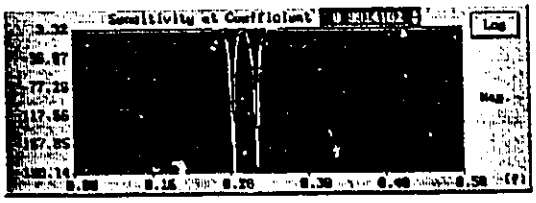
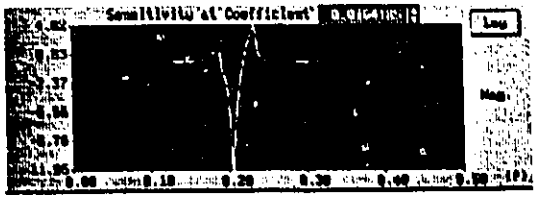
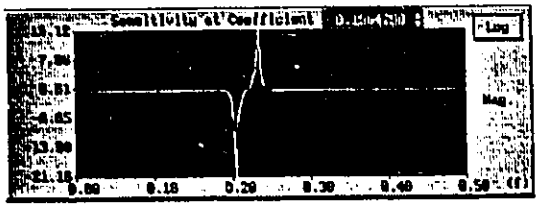


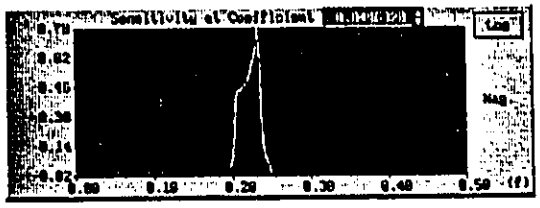



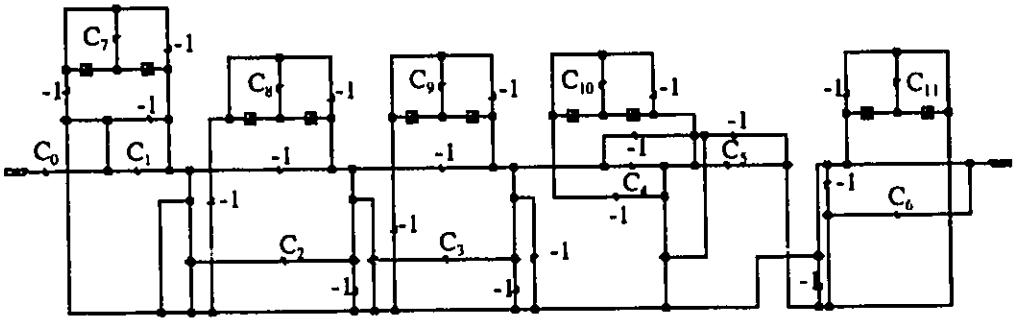




D7 Wave Digital Structure of Band-Pass Digital Filter



C_0=1.31194961, C_1=0.06502370645, C_2=0.2235692269, C_3=0.5809284606,
C_4=0.1626985222, C_5=0.007805575218, C_6=0.0574957168, C_7=-0.1915378541,
C_8=0.0003565321277, C_9=-0.3698270573, C_{10}=-0.1915378541, C_{11}=0.1915378541

Common Netlist

10	41	51	48	81
46	45	1.000000000000000		
47	46	-0.191537854100000		
47	1	1.000000000000000		
1	45	1.000000000000000		

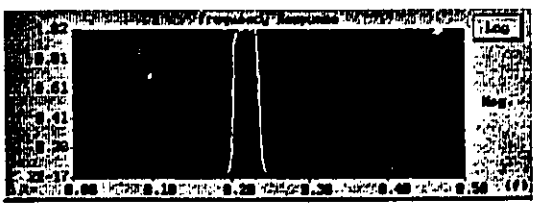
46	16	-1.000000000000000
19	18	0.065023709500000
18	11	1.311949610000000
2	35	1.000000000000000
37	2	1.000000000000000
36	35	1.000000000000000
37	36	3.565321000000000D-004
3	37	1.000000000000000
12	3	1.000000000000000
36	12	-1.000000000000000
20	12	1.000000000000000
4	39	1.000000000000000
41	4	1.000000000000000
40	39	1.000000000000000
41	40	-0.369827057300000
5	41	1.000000000000000
13	5	1.000000000000000
40	13	-1.000000000000000
21	13	1.000000000000000
6	25	1.000000000000000
27	6	1.000000000000000
26	25	1.000000000000000
27	26	-0.191537854100000
7	27	1.000000000000000
14	7	1.000000000000000
26	14	-1.000000000000000
24	14	1.000000000000000
24	22	0.162698522200000
22	14	-1.000000000000000
44	19	1.000000000000000
44	33	1.000000000000000
34	33	1.000000000000000
35	34	-1.000000000000000
32	31	-1.000000000000000
34	31	1.000000000000000
33	32	0.223569226900000
32	20	1.000000000000000
31	20	1.000000000000000
38	31	1.000000000000000
39	38	-1.000000000000000
31	30	0.580928460600000
38	29	1.000000000000000
30	29	-1.000000000000000
30	21	1.000000000000000

29	21	-1.000000000000000
25	24	-1.000000000000000
28	24	1.000000000000000
29	28	1.000000000000000
23	14	-1.000000000000000
28	14	1.000000000000000
43	42	-1.000000000000000
49	42	1.000000000000000
43	15	1.000000000000000
42	15	-1.000000000000000
42	28	1.000000000000000
8	49	1.000000000000000
51	8	1.000000000000000
51	50	0.191537854100000
9	51	1.000000000000000
15	9	1.000000000000000
48	43	0.057495716800000
50	15	1.000000000000000
50	49	-1.000000000000000
33	19	1.000000000000000
18	17	1.000000000000000
44	17	1.000000000000000
45	44	-1.000000000000000
20	19	-1.000000000000000
21	20	-1.000000000000000
22	21	1.000000000000000
49	48	1.000000000000000
10	47	1.000000000000000
16	10	1.000000000000000
17	16	-1.000000000000000
19	16	1.000000000000000
23	15	1.000000000000000
24	23	0.007805575200000

Frequency Response Analysis

Unique Netlist

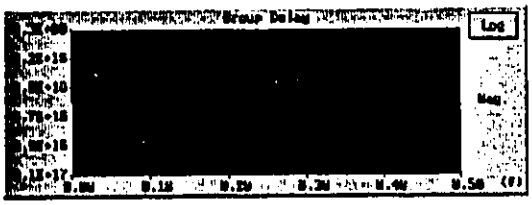
11 201 0

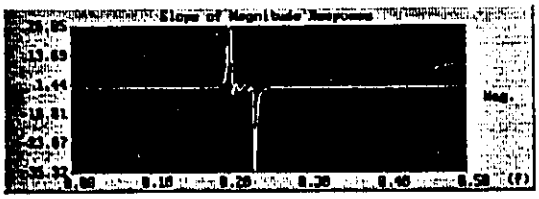


Group Delay and Slope of Magnitude Response

Unique Netlist

11 201





Noise Analysis

Unique Netlist

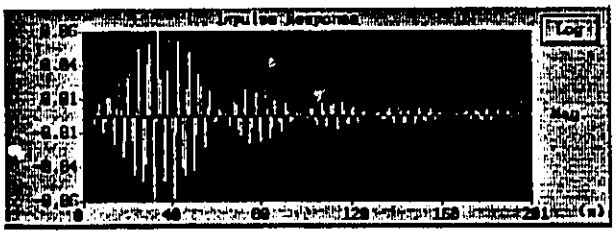
12 201 0.1000000 4
47
19
18
37
41
27
24
33
31
51
48
24

Graph cannot be plotted because the number of signal nodes in the common netlist are larger than the array size that defined in analysis module, n1dpmr.exe.

Impulse Response Analysis

Unique Netlist

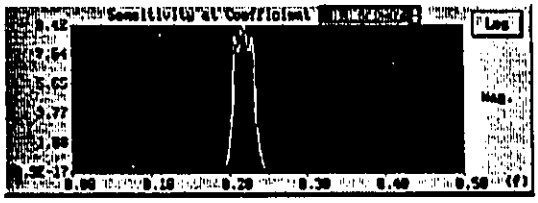
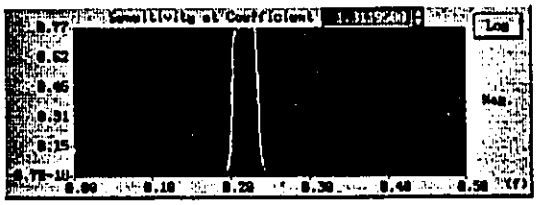
11 201

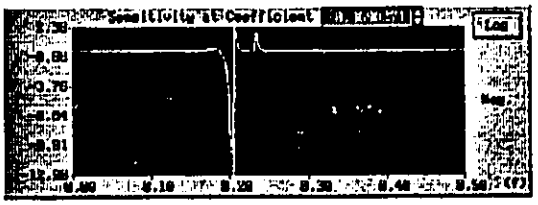
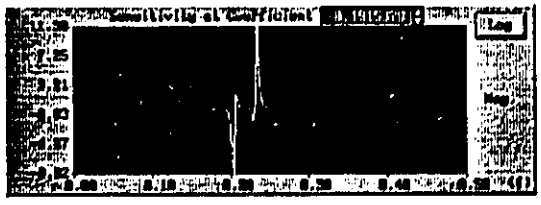


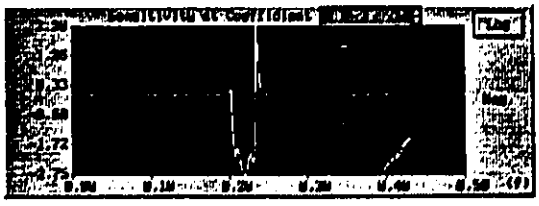
First Order Coefficient Sensitivity Analysis

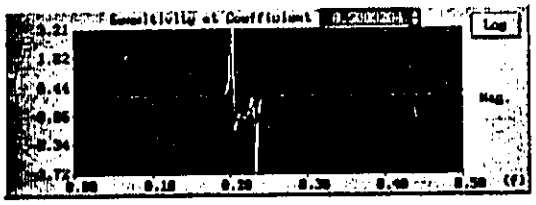
Unique Netlist

11	201	12
46	47	
18	19	
11	18	
36	37	
40	41	
26	27	
22	24	
32	33	
30	31	
50	51	
43	48	
23	24	

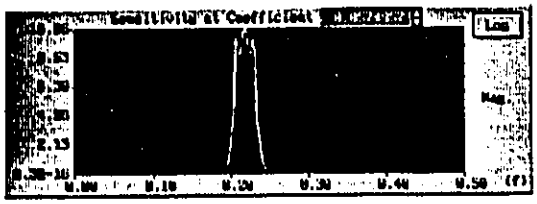




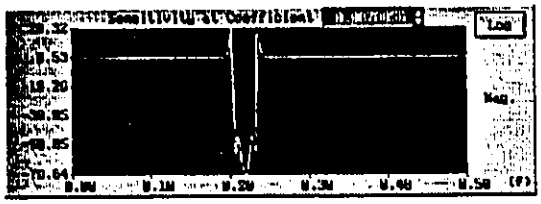






Log
Mag.
(f)


Log
Mag.
(f)

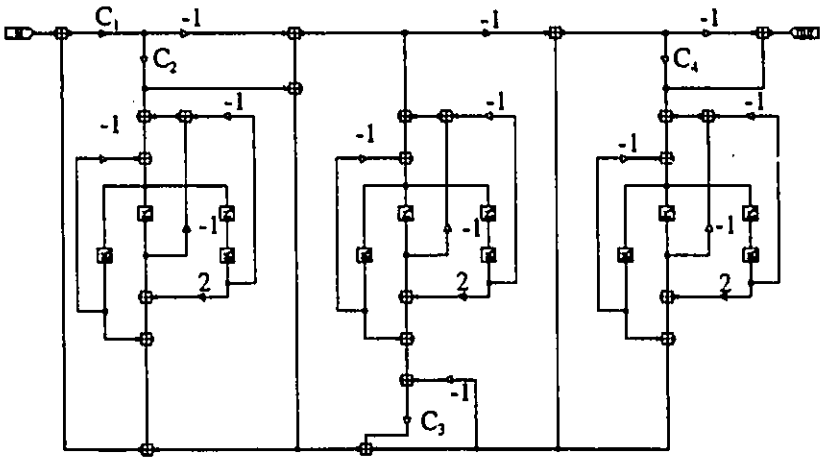

Log
Mag.
(f)


Log
Mag.
(f)

Appendix E

Examples of Analyzing 2-D Digital Networks

E1 Third-Order Linear Transformed 2-D Chebyshev Low-Pass Digital Filter



C_1=0.80346, C_2=0.91176, C_3=0.92370, C_4=0.78846

Common Netlist

6	6	22	34	34	56
27	26	0.911760000000000			
1	28	1.000000000000000			
7	1	1.000000000000000			
13	7	1.000000000000000			
28	13	-1.000000000000000			
25	13	2.000000000000000			
2	28	1.000000000000000			

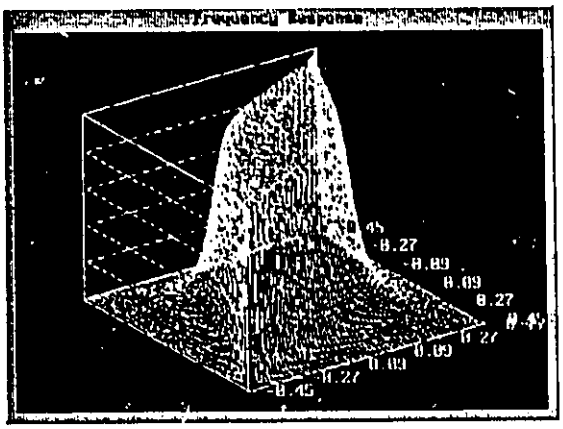
14	2	1.000000000000000
25	14	1.000000000000000
28	14	-1.000000000000000
8	28	1.000000000000000
15	8	1.000000000000000
25	15	1.000000000000000
28	15	-1.000000000000000
3	30	1.000000000000000
9	3	1.000000000000000
16	9	1.000000000000000
30	16	-1.000000000000000
23	16	2.000000000000000
4	30	1.000000000000000
17	4	1.000000000000000
23	17	1.000000000000000
30	17	-1.000000000000000
10	30	1.000000000000000
18	10	1.000000000000000
23	18	1.000000000000000
30	18	-1.000000000000000
32	31	0.788460000000000
5	33	1.000000000000000
11	5	1.000000000000000
19	11	1.000000000000000
33	19	-1.000000000000000
22	19	2.000000000000000
6	33	1.000000000000000
20	6	1.000000000000000
22	20	1.000000000000000
33	20	-1.000000000000000
12	33	1.000000000000000
21	12	1.000000000000000
22	21	1.000000000000000
33	21	-1.000000000000000
29	26	-1.000000000000000
28	27	1.000000000000000
24	23	0.923700000000000
25	24	1.000000000000000
29	27	1.000000000000000
29	24	1.000000000000000
23	22	-1.000000000000000
31	29	-1.000000000000000
31	22	1.000000000000000
34	31	-1.000000000000000

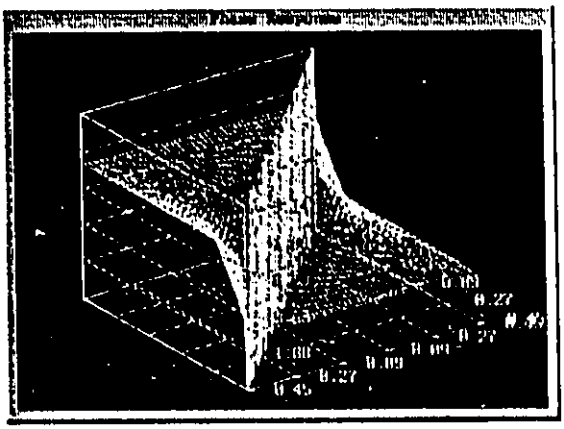
33	32	1.000000000000000
34	32	1.000000000000000
26	25	0.803460000000000
30	29	1.000000000000000
24	22	1.000000000000000

Frequency Response Analysis

Unique Netlist

25 61 0

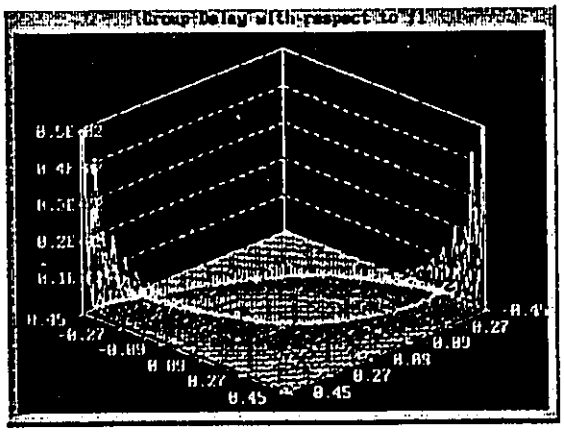


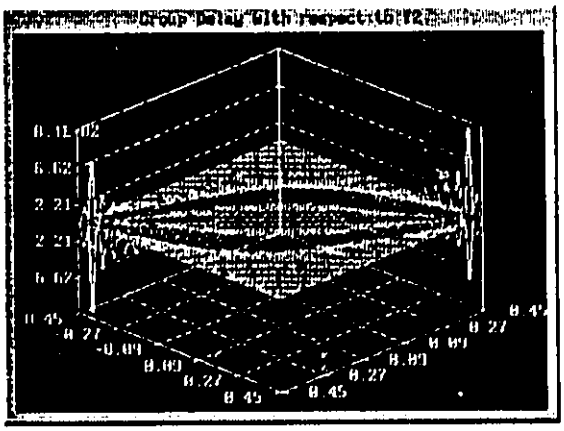


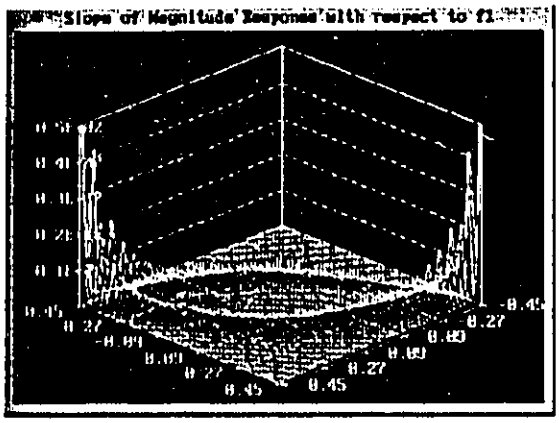
Group Delay and Slope of Magnitude Response Analysis

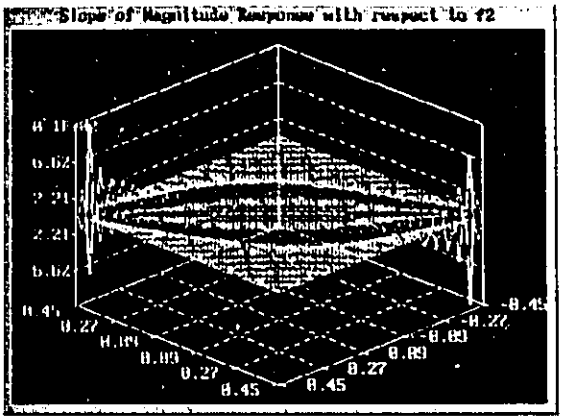
Unique Netlist

25 61





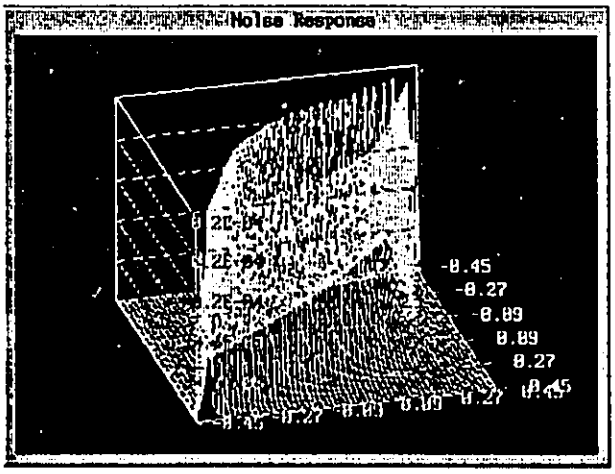




Noise Analysis

Unique Netlist

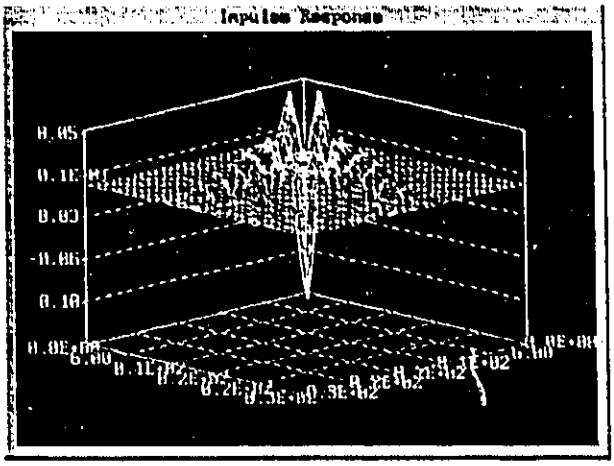
7 61 0.2000000 8
27
25
23
32
22
24
26



Impulse Response Analysis

Unique Netlist

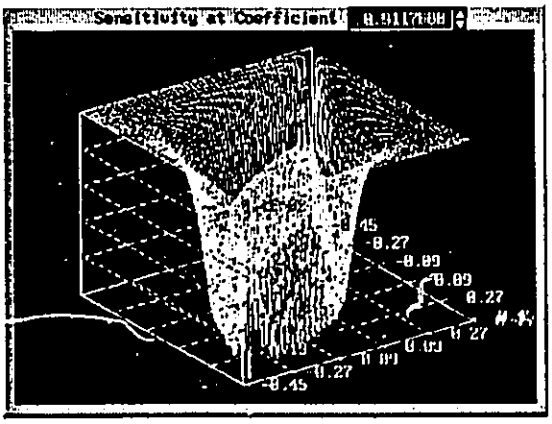
25 30 30

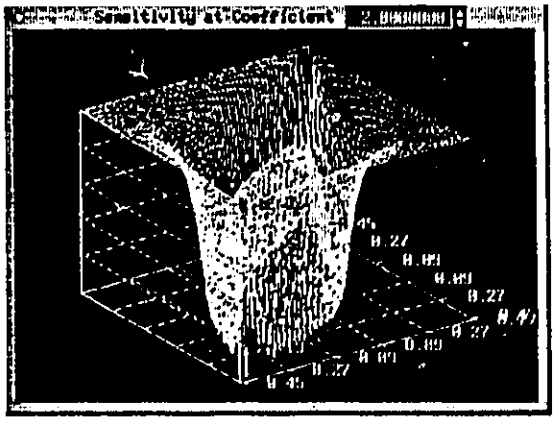


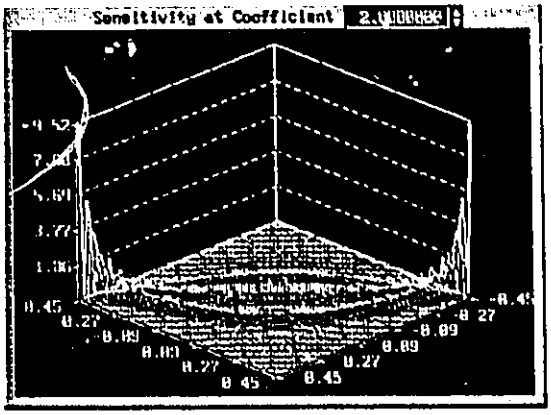
First Order Coefficient Sensitivity Analysis

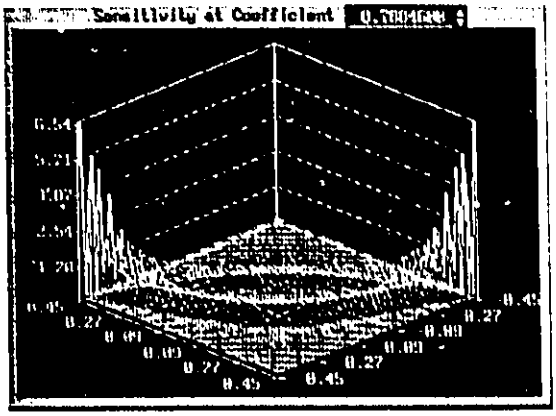
Unique Netlist

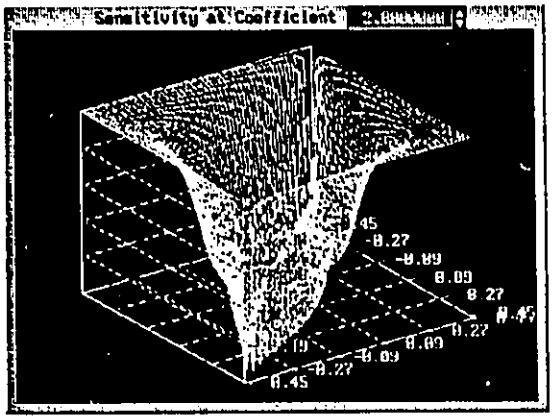
7 201 7
4 7
4 8
5 7
5 8
6 7
6 8
7 8

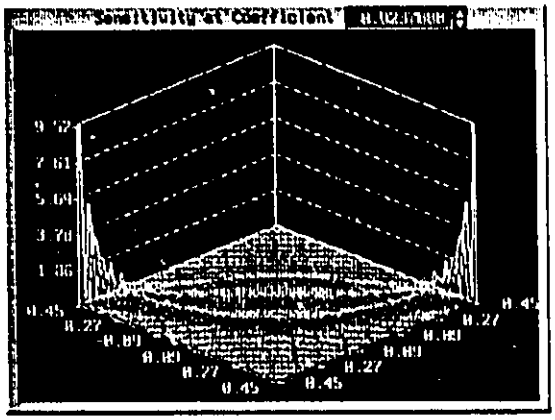


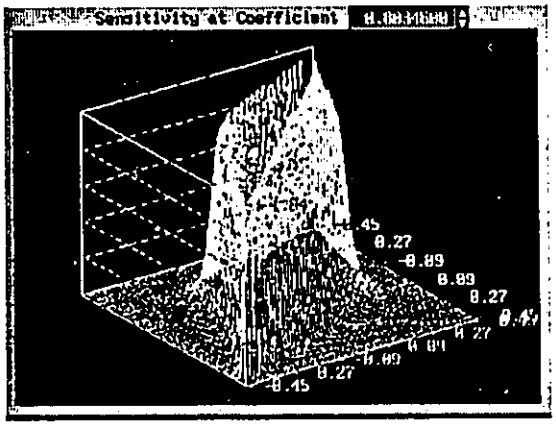



Sensitivity at Coefficient

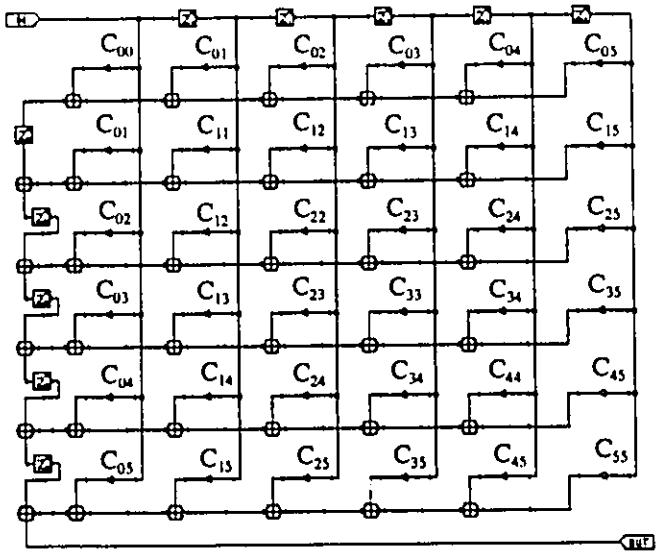

Sensitivity at Coefficient


Sensitivity at Coefficient


Sensitivity at Coefficient


Sensitivity at Coefficient

D2 2-D FIR Digital Network



C_{00}=0.137375027, C_{01}=0.213937491. C_{02}=0.0905000120, C_{03}=-0.0219999980, C_{04}=-0.0548124984, C_{05}=-0.0266874935, C_{11}=0.342062384, C_{12}=0.146749973, C_{13}=-0.0313749984, C_{14}=-0.0907499939, C_{15}=-0.0376249924, C_{22}=0.0608125031, C_{23}=-0.0110624973, C_{24}=-0.0391874984, C_{25}=-0.0173124969, C_{33}=0.00456250273, C_{34}=0.00768750440, C_{35}=0.00612500263, C_{44}=0.0264375042, C_{45}=0.0108124986, C_{55}=0.00768750207

Common Netlist

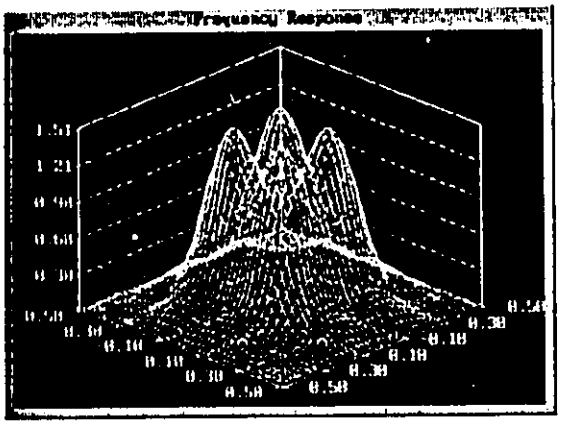
5	5	12	22	22	56
17	15	0.137375027000000			
18	15	0.213937491000000			
18	14	0.342062384000000			
19	15	0.090500012000000			
19	14	0.146749973000000			
19	13	0.060812503100000			
20	15	-0.021999998000000			
20	14	-0.031374998400000			
20	13	-0.011062497300000			
20	12	0.004562502700000			
21	15	-0.054812498400000			
21	14	-0.090749993900000			
21	13	-0.039187498400000			
21	12	0.007687504400000			
21	11	0.026437504000000			
22	15	-0.026687493500000			

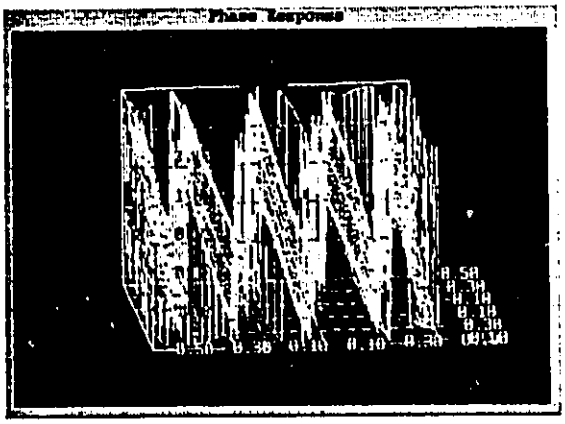
22	14	-0.037624992400000
22	13	-0.017312496900000
22	12	0.006125002600000
22	16	0.007687502100000
22	11	0.010812498600000
1	15	1.000000000000000
14	1	1.000000000000000
2	14	1.000000000000000
13	2	1.000000000000000
3	13	1.000000000000000
12	3	1.000000000000000
4	12	1.000000000000000
11	4	1.000000000000000
5	11	1.000000000000000
16	5	1.000000000000000
17	14	0.213937491000000
17	13	0.090500012000000
17	12	-0.021999998000000
17	11	-0.054812498400000
17	16	-0.026687493500000
18	13	0.146749973000000
18	12	-0.031374998400000
18	11	-0.090749993900000
18	16	-0.037624992400000
19	12	-0.011062497300000
19	11	-0.039187498400000
19	16	-0.017312496900000
20	11	0.007687504400000
20	16	0.006125002600000
21	16	0.010812498600000
6	18	1.000000000000000
7	19	1.000000000000000
8	20	1.000000000000000
9	21	1.000000000000000
19	6	1.000000000000000
20	7	1.000000000000000
21	8	1.000000000000000
22	9	1.000000000000000
10	17	1.000000000000000
18	10	1.000000000000000

Frequency Response Analysis

Unique Netlist

15 41 0

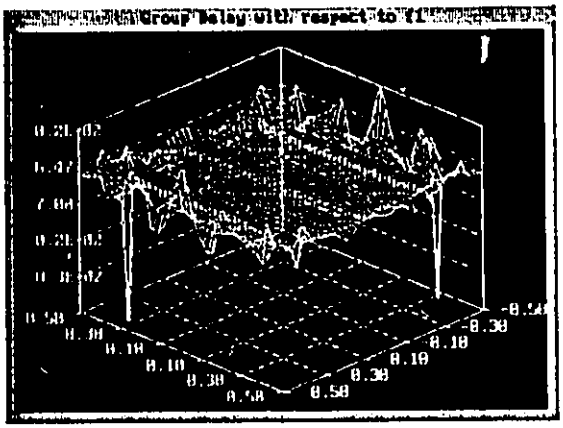


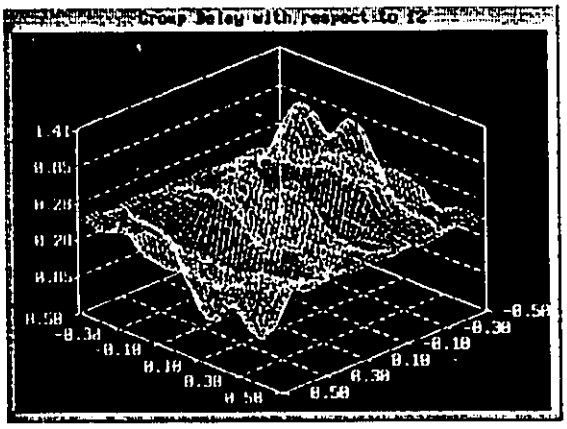


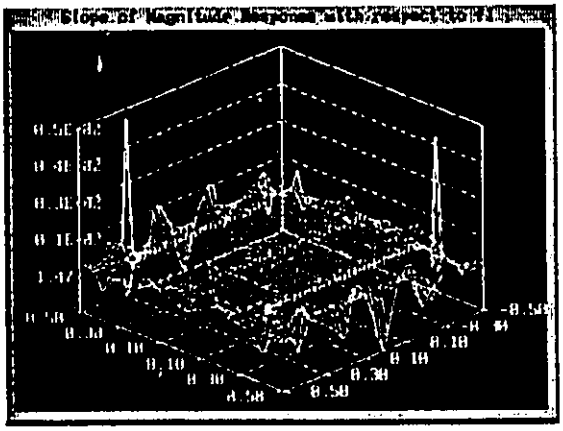
Group Delay and Slope of Magnitude Response Analysis

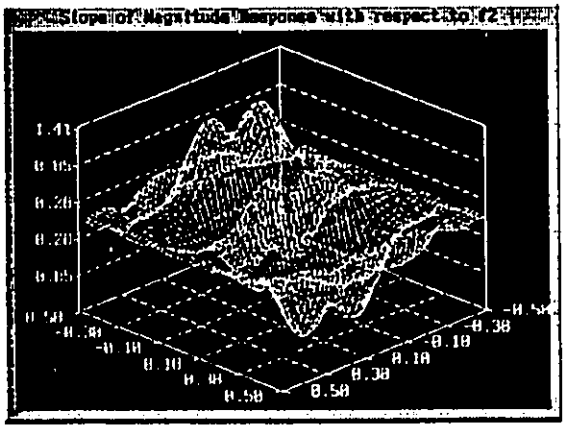
Unique Netlist

15 41





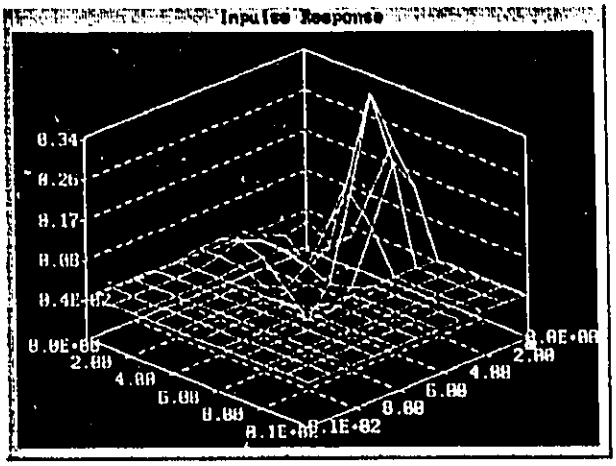




Impulse Response Analysis

Unique Netlist

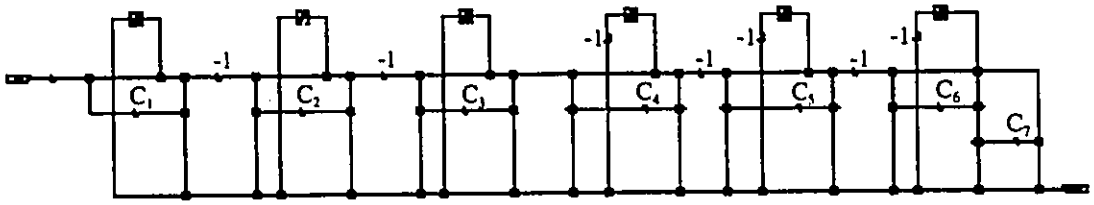
15 30 30



Appendix F

Example of Analyzing 3-D Digital Network

D1 3-D Digital Filter



C_1=0.322931054, C_2=0.596278407, C_3=0.712391977, C_4=0.061408892, C_5=0.515838536, C_6=-1.166123347, C_7=0.269284666

Common Netlist

2	2	2	30	36	30	59
9	8	-0.322931054000000				
1	34	1.000000000000000				
8	1	1.000000000000000				
10	1	1.000000000000000				
10	9	1.000000000000000				
34	9	1.000000000000000				
12	11	-0.596278407000000				
3	33	1.000000000000000				
11	3	1.000000000000000				
13	3	1.000000000000000				
13	12	1.000000000000000				
33	12	1.000000000000000				
34	33	1.000000000000000				

34	11	1.000000000000000
15	14	-0.712391977000000
5	32	1.000000000000000
14	5	1.000000000000000
16	5	1.000000000000000
16	15	1.000000000000000
32	15	1.000000000000000
33	32	1.000000000000000
33	14	1.000000000000000
11	10	-1.000000000000000
14	13	-1.000000000000000
8	7	-1.000000000000000
31	16	1.000000000000000
36	35	-1.000000000000000
2	36	1.000000000000000
17	2	1.000000000000000
17	16	1.000000000000000
26	17	1.000000000000000
31	26	-0.061408892000000
35	25	1.000000000000000
26	25	1.000000000000000
25	18	1.000000000000000
27	25	1.000000000000000
28	27	-1.000000000000000
4	28	1.000000000000000
19	4	1.000000000000000
19	18	1.000000000000000
24	19	1.000000000000000
25	24	-0.515838536000000
27	23	1.000000000000000
24	23	1.000000000000000
18	17	-1.000000000000000
21	19	-1.000000000000000
22	21	-1.166123347000000
22	20	1.000000000000000
21	20	1.000000000000000
23	21	1.000000000000000
23	22	1.000000000000000
29	22	-1.000000000000000
6	29	1.000000000000000
20	6	1.000000000000000
22	20	0.269284666000000
30	20	1.000000000000000
30	22	1.000000000000000

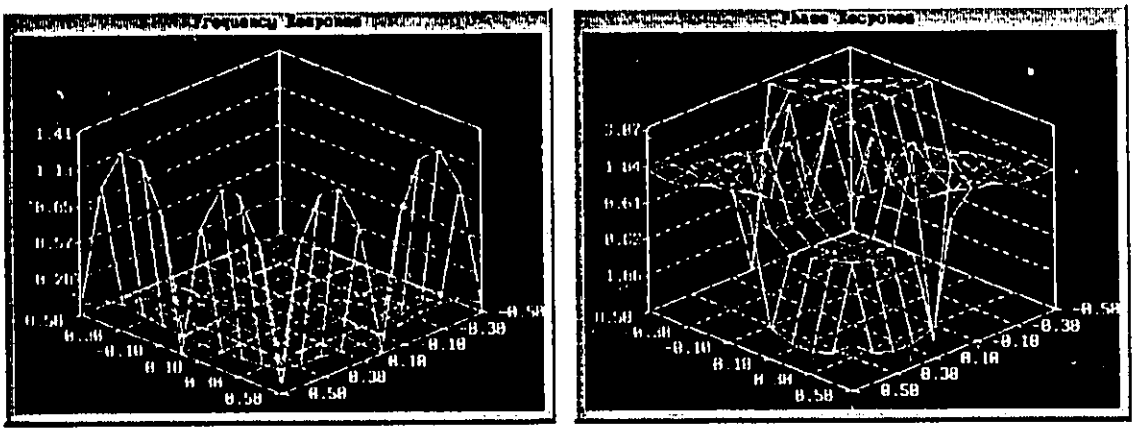
35	31	1.000000000000000
32	31	1.000000000000000

Frequency Response Analysis

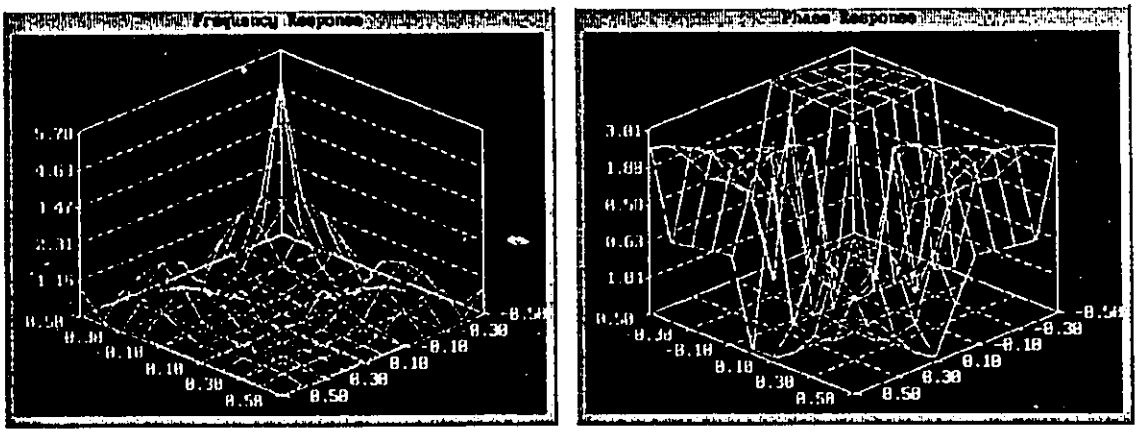
Unique Netlist

7	11	0

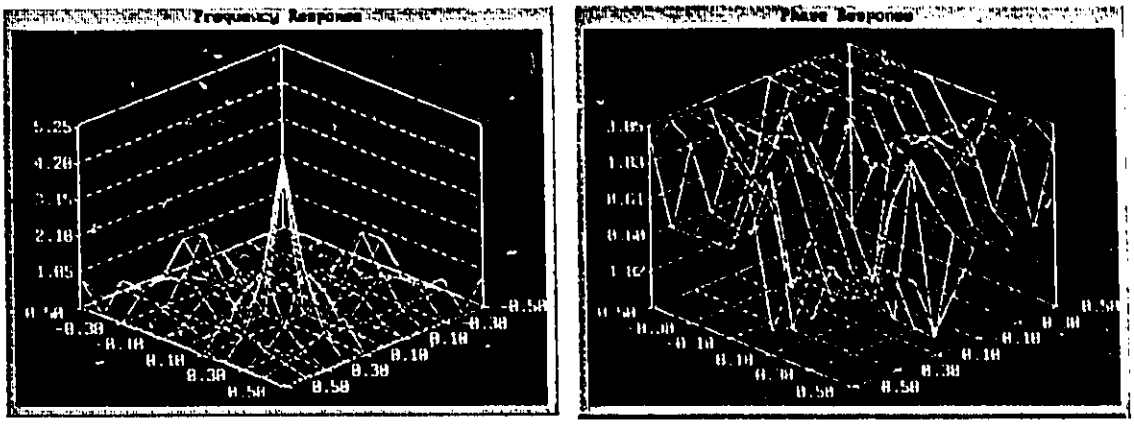
f3 = -0.495



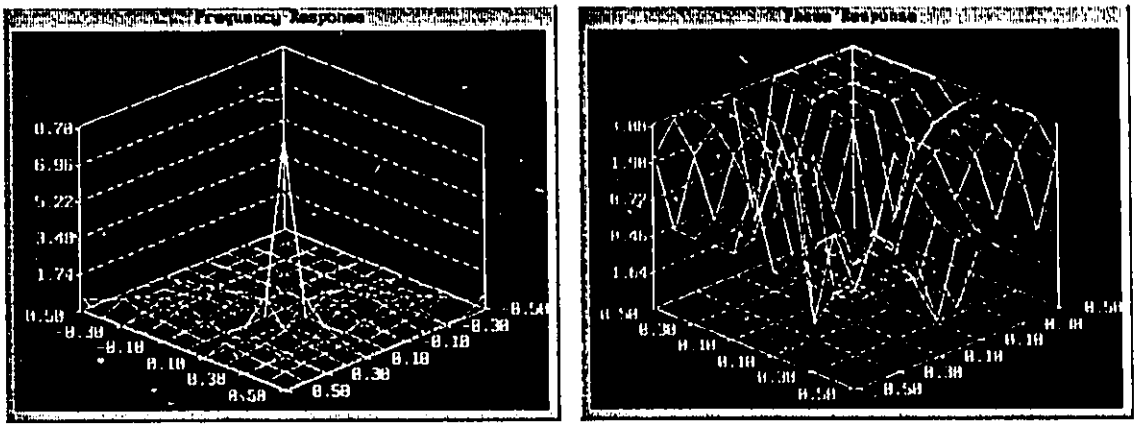
f3=-0.396



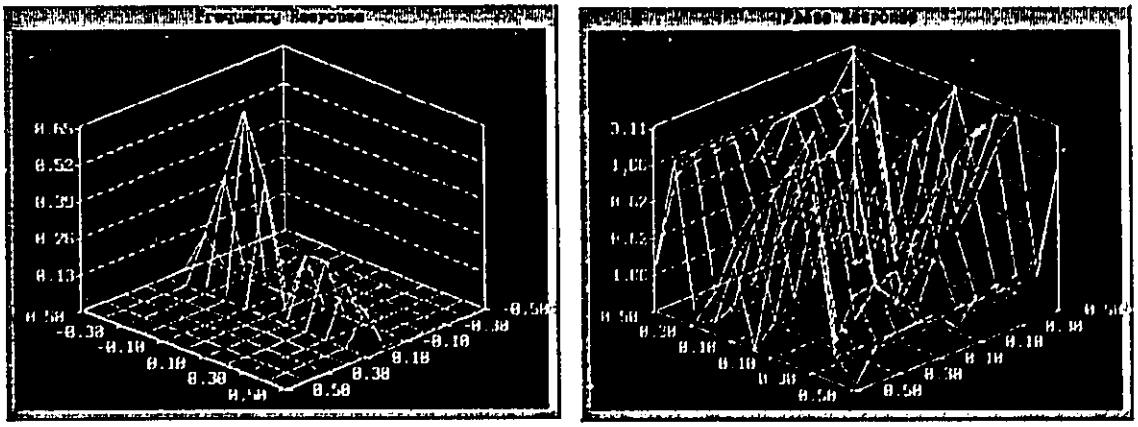
f3=-0.198



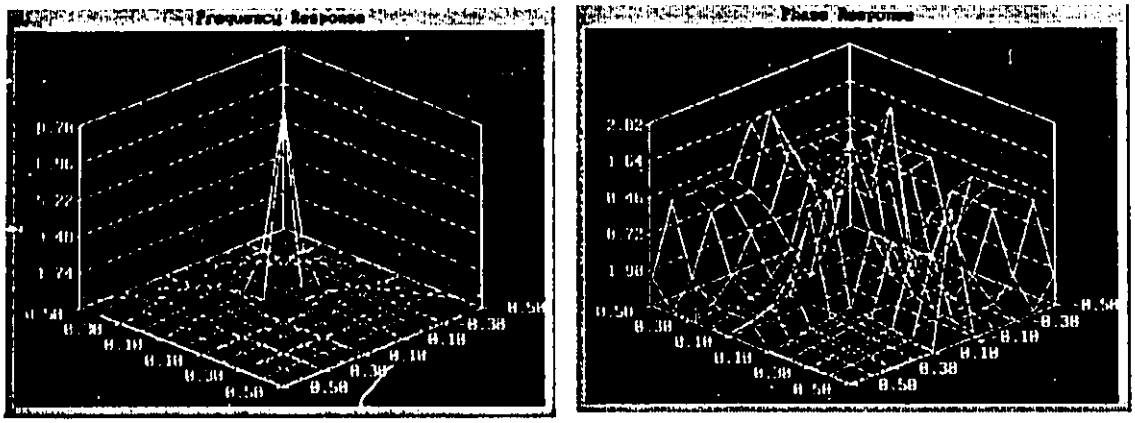
f3=-0.099



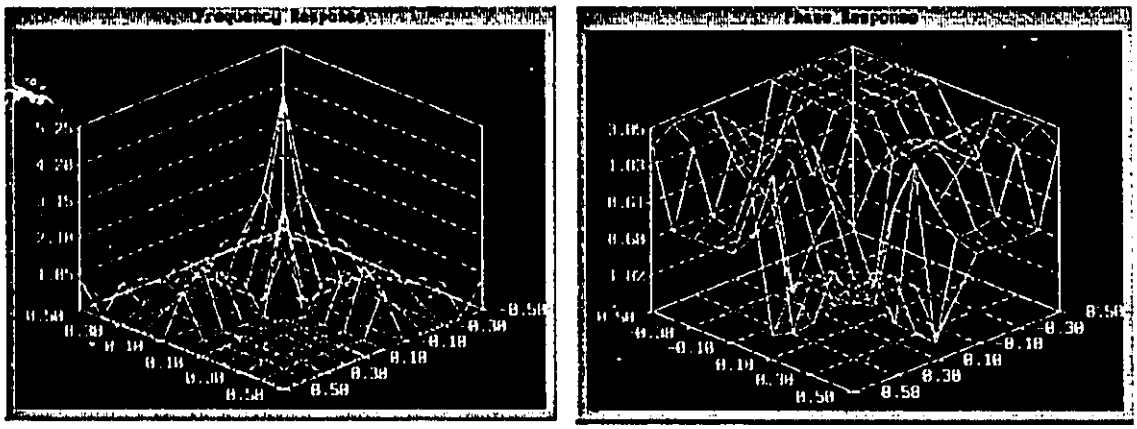
f3=0.000



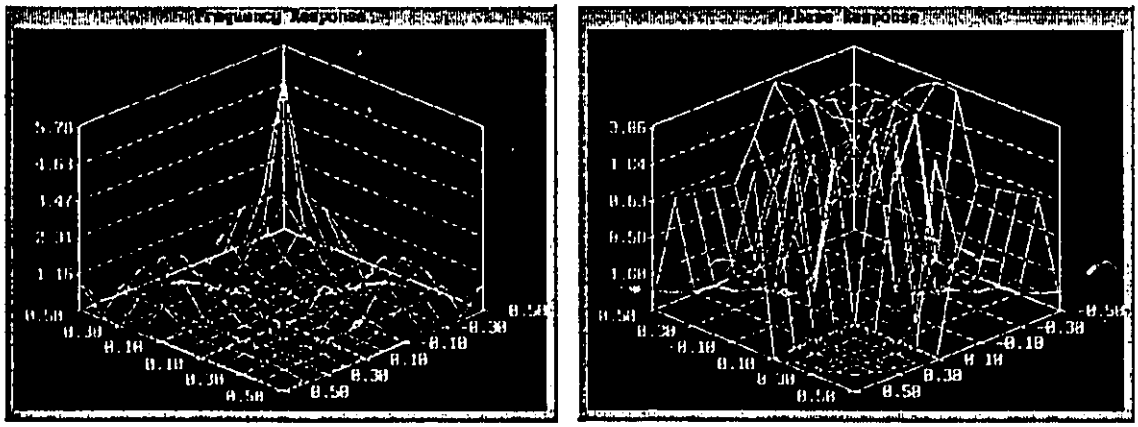
f3=0.099



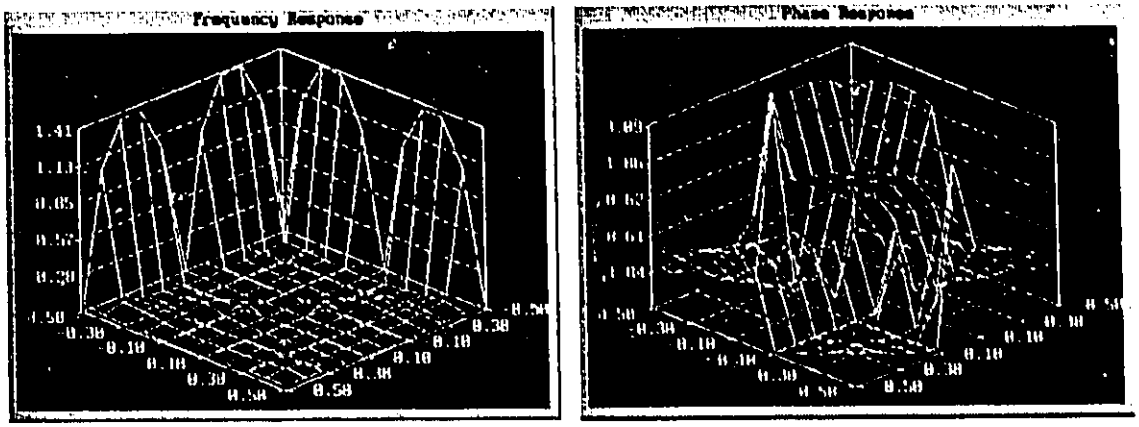
f3=0.198



f3=0.396



f3=0.495



Other types of analysis is not shown because too many graphs need to be displayed.

Vita Auctoris

Jacky Luk was born in Hong Kong in 1967. He came to Canada in 1986, and studied in F.J. Brennan High School, Windsor, and recieved his high school dilopma in 1987. He received the B.A.Sc. degree in electrical engineering from University of Windsor, Windsor, Ont., in 1991. He was admitted in the Master's program at Electrical Engineering Department, University of Windsor, in 1992. He is now a candidate of M.A.Sc. degree in Electrical Engineering Department, University of Windsor.