copy 1

Interactive Computer Programs for the Computer Aided Design of Linear Microwave Circuits

Author

Brian G. Marchent, B.Sc. (Hons.)

A thesis submitted at the University of Warwick for the degree of Doctor of Philosophy.

Date : July 1973

and almost any microwave circuit could be handled by the method of analysis used in this program.

During the development of the programs, as each new program was developed, more facilities were included for the interactive on-line use of the programs. In the MICRO3 program a list processing approach to the data structure in the program and a full syntactical analysis of the data was included. This made it possible to generate very complex data structures in the program thus making the program ideal for interactive on-line use on a computer. The data for this program was more like a programming language than normal data and any data error could be correct on-line during the run of the program. The program could be used on batch processing, on a remote teletype interactively or with a graphical display to display the results of an analysis in graphical form.

An investigation into the equations describing microwave components and optimisation techniques was carried out. The MICRO3 program was prepared for these facilities but there was not sufficient time to include these facilities completely.

Preface

The idea for the research work cont ied in this Ph. D. thesis originated from a suggestion to the author by Mr. M.K. McPhun at the University of Warwick in November 1968 that research work in the field of the Computer Aided Design of Microwave Circuits would be a very useful research topic for a Ph. D. thesis. The research work on this topic started in May 1969 when the author started working with Microwave Associated Ltd., Luton whilst waiting for a Science Research Council grant to start in October 1969. Then a move was made to the School of Engineering Science at the University of Warwick where the main part of the research work was carried out up to October 1972. Also in February 1970 the Science Research Council grant was changed to an industrial studentship grant between the Science Research Council and Redac Software Ltd.. The writing of this thesis was completed in the author's spare time whilst working with Redac Software Ltd., Tewkesbury after October 1972.

At Microwave Associates Ltd., Luton experience was obtained in the theorectical design of microwave integrated circuits. To assist with this work a remote teletype terminal for a time sharing computer was used at Luton for microwave circuit synthesis whilst the computer program written by McPhun$^{(0.1)}$ using chain matrix analysis was used for circuit analysis on the Elliott 4130 computer at the University of Warwick. Two problems were investigated with Microwave Associates Ltd.. The first problem consisted of checking and, if possible, improving the design of an L-band microwave integrated circuit switch$^{(0.2)}$ as described in Appendix A.1. The second problem consisted of carrying out a theorectical investigation into various types of microwave phase shifter circuits for their possible production in the form of microwave integrated circuits$^{(0.3)}$ as described in Appendix A.2. For the second problem it was found that McPhun's computer program(0.1) was not suitable for the types of circuits to be analysed. Thus a new computer program, BCMA(0.4), had to be written by the author for this work to extend the ideas contained in McPhun's program.

On arriving at the University of Warwick in October 1969 a year was spent almost entirely on extending the appication of the chain matrix method of analysis for the analysis of general microwave circuits. During this period the CHAIN1 program(0.5) was written by the author using chain matrix analysis for the analysis of general microwave circuits. This program analysed a circuit consisting of an assembly of 2-port networks and it broke this assembly down into a simpler assembly of cascades of 2-port networks forming a main link between the input and output port of the circuit and branch arms, loop paths and parallel paths built to any level on this main path.

Towards the completion of the work on developing the CHAIN1 program it was realised that the use of the chain matrix method of analysis had been taken just about to its limit but it could still not analyse all microwave circuits. Also at that time the use of the chain matrix method of analysis for general microwave circuits was being critised by Dr. Larcombe, University of Warwick and Mr. Wolfendale, Redac Software Ltd.. Thus at that time it became necessary to completely rethink the method of analysis being used. The result of this was that it was decided to use a previously untried method for the analysis of microwave circuits.

The new method used was the mixed matrix method of analysis for general microwave circuits considered as an assembly of n-port networks(0.6). This method was found to be far more suitable. The MICRO2 computer program(0.7) was written by the author to use this method of analysis. This program took about a year to write and was later replaced by the MICRO3 computer program which incorporated more facilities (see Appendix A.5).

During the course of the research work for this thesis there was a considerable amount of interest shown by the University of Warwick in the use of computer programs in an interactive mode. Thus a large portion of the time spent in writing the BGMA, CHAIN1, MICRO2 and MICRO3 programs was spent on developing interactive applications for these programs with each program including more interactive facilities than the last one. This work involved developing techniques for a full syntax analysis of the data, the use of complex,but very versatile,data structures and the use of the program interactively on a remote teletype possibly with a graphical display to display the results of a circuit analysis in graphical form. The result was that in the MICRO3 program all these facilities were included and the data for the program had more the form of a computer programming language that the normal data for a program.

Some time was spent during the course of the research for this thesis investigating the range of microwave components which should be included in a computer program for the analysis of microwave circuits. Also an investigation into how optimisation could be included in this type of program was carried out.

The organisation of this thesis is as follows. In chapter 1 various types of microwave circuits are looked at to decide what problems are involved in the analysis of microwave circuits on a computer and the way these problems should be tackled. In chapter 2 three possible methods of analysis are discussed with the intention of deciding on the best method for the analysis of microwave circuits. Chapter 3 forms a supporting chapter for chapter 2 describing the various matrix methods involved in analysis and the best way these can be organised. Chapters 4 and 5 describe the CHAIN1 program with chapter 4 describing the path topological analysis used to break the circuit into paths consisting of cascades of 2-port networks and chapter 5 describes the organisation of the CHAIN1 program. Chapters 6 and 7 describe the MICRO2 and MICRO3 programs with chapter 6 describing the data structure used and chapter 7 describing the organisation of the program. Chapter 8 describes the interactive facilities included in the programs whilst chapters 9 and 10 describe the work done on microwave components and optimisation respectively. Finally chapter 11 gives the conclusions of the whole thesis.

References

0.1) M.K. McPhun, 'A Computer Program for the Analysis of Branched Distributed and Lumped Circuits', IEE Conference Publication No. 23, 1966, pp. 89-124

0.2) B.G. Marchent, 'Computer Aided Design of an L-Band M.I.C. Switch', Microwave Associates Ltd., Luton, 1969, Technical Report No. 160, (see also Appendix A.1)

0.3) B.G. Marchent, 'A Comparative Study of the Iterative, Hybrid Ring and Switched Line Microwave Phase Shifters', Microwave Associates Ltd., Luton, 1969, private communication, (see also Appendix A.2)

0.4) B.G. Marchent, 'A Computer Program (BGMA) for the Analysis of Lumped and Distributed Networks', University of Warwick, Nov. 1969, School of Engineering Science Report No. 50

0.5) B.G. Marchent, 'CHAIN1, A Frequency Domain Circuit Analysis Program using Chain Matrices', University of Warwick, July 1970, School of Engineering Science Report No. 64, (see also Appendix A.4)

0.6) B.G. Marchent, 'The Computer Aided Design of Microwave Circuits', IREE Czechoslovakia, Summer School on Circuit Theory 1971, Short Contributions Vol. 2, pp.255-263 (see also Appendix A.6)

0.7) B.G. Marchent, 'MICRO2, A Frequency Domain Circuit Analysis Program for Microwave Circuits using Mixed Matrices', University of Warwick, June 1971, School of Engineering Science Report No. 65

ACKNOWLEDGEMENTS

The author would like to thank the following for their assistance during the course of the research work and preparation of this thesis.

1) Mr. M.K. McPhun, University of Warwick
For the initial idea for a Ph. D. research topic on the Computer Aided Design of Microwave Circuits and his supervision of the research work.

2) Microwave Associates Ltd., Luton
For supplying the author with experience in the design of microwave integrated circuits, and also an income, whilst waiting for a Science Research Council grant to start.

3) Science Research Council
For their financial support during the course of the research work for the author's maintenance and costs for the author to attend conferences.

4) Redac Software Ltd., Tewkesbury
For their financial support and the use of their experience in the design of general purpose interactive computer programs for electronic circuit design.

5) Dr. M.H.E. Larcombe, University of Warwick
For his advice on syntax analysis, a list processing approach to data structure and matrix analysis.

6) Dr. W.R. Laxon, Lanchester Polytechnic
For his ideas on the organisation of a list processing approach to data structure for complex structures.

7) Mrs. Sonia Marchent, wife
For her support during the writing and typing of this thesis through many lost weekends and holidays.

8) Lucy Jane Marchent, daughter, born 25th October 1972
One of the conclusions to this thesis.

Contents

CONTENTS

Chapter 1

SPECIFICATION OF THE PROBLEM FOR THE COMPUTER AIDED DESIGN OF MICROWAVE CIRCUITS

Chapter 2

CIRCUIT ANALYSIS FOR MICROWAVE CIRCUITS

Chapter 3

MATRIX METHODS FOR CIRCUIT ANALYSIS page

Chapter 4

PATH TOPOLOGICAL ANALYSIS FOR CHAIN MATRIX ANALYSIS PROGRAM page

Chapter 5

CHAIN MATRIX ANALYSIS CHAIN1 PROGRAM page

Chapter 6

LIST PROCESSING APPROACH TO DATA STRUCTURE FOR MIXED MATRIX ANALYSIS PROGRAM

Chapter 9

MICROWAVE COMPONENTS IN CIRCUIT ANALYSIS PROGRAM page

Chapter 10

OPTIMISATION IN MICROWAVE ANALYSIS PROGRAM page

Chapter 11

CONLUSIONS OF THESIS page

Appendix A.1

L-BAND MICROWAVE INTEGRATED CIRCUIT SWITCH page

Appendix A.2

COMPARATIVE STUDY OF MICROWAVE INTEGRATED CIRCUIT PHASE SHIFTERS page

Appendix A.3

COMPUTATION TIMES ON ELLIOTT 4130 COMPUTER page

Appendix A.4

USER MANUAL FOR CHAIN MATRIX ANALYSIS CHAIN1 PROGRAM page

page

Appendix A.5

USER MANUAL FOR MIXED MATRIX ANALYSIS

MICRO3 PROGRAM

page

Chapter 1

Specification of the Problem for the Computer Aided Design of Microwave Circuits

1.1. MICROWAVE CIRCUITS

1.1.1. Definition

The first question to ask is "What is a microwave circuit ?". The first answer to this question could be that a microwave circuit is an assembly of components which carry electrical signals in the normal microwave region, i.e. 1GHz to 100GHz. This may appear to be a fairly good description but it is not really sufficient for this thesis.

For this thesis a microwave circuit could better be described as an assembly of components which require microwave techniques for their analysis. At low frequencies all electronic components can be characterised, at least for a small signal analysis, by an assembly of lumped components consisting of resistors, inductors, capacitors, dependant and independant voltage and current generators. This is not possible at microwave frequencies when disturbances in the circuit take a long time, comparable to the period of oscillation at the frequency being considered, to reach every part of the circuit. The microwave components thus become distributed in their nature and will propagate electromagnetic waves within their physical size and/or radiate electromagnetic energy into the surrounding medium. The terms voltage and current no longer have any well defined meaning and energy will travel to every part of the microwave circuit in the form of electromagnetic waves.

The definition of a microwave circuit given by Montgomery(1.1) states : "A microwave circuit is a region enclosed by a metallic wall of any shape and communicates with the exterior by way of a number of transmission lines or waveguides which may be called the terminals of the circuit.". This is illustrated for a 3-port microwave circuit in Fig. 1.1. At this stage we are dealing with a very generalised microwave circuit.

The next problem is to provide a complete description of the performance of the microwave circuit between the terminals, or ports, of the circuit. At low frequencies it is usually possible to relate voltages and currents at all the nodes in the circuit but for a microwave circuit, in general, this is not possible. The problem is that a microwave circuit communicates with the exterior by means of time varying electric and magnetic fields in the form of electromagnetic waves propagating in the lines or waveguides connecting the circuit to the exterior. Thus it is necessary to describe the circuit performance in terms of these fields on the boundary of the microwave circuit with the exterior. Thus the analysis of a microwave circuit involves the solution of a dynamic field problem where it is necessary to solve Maxwell's

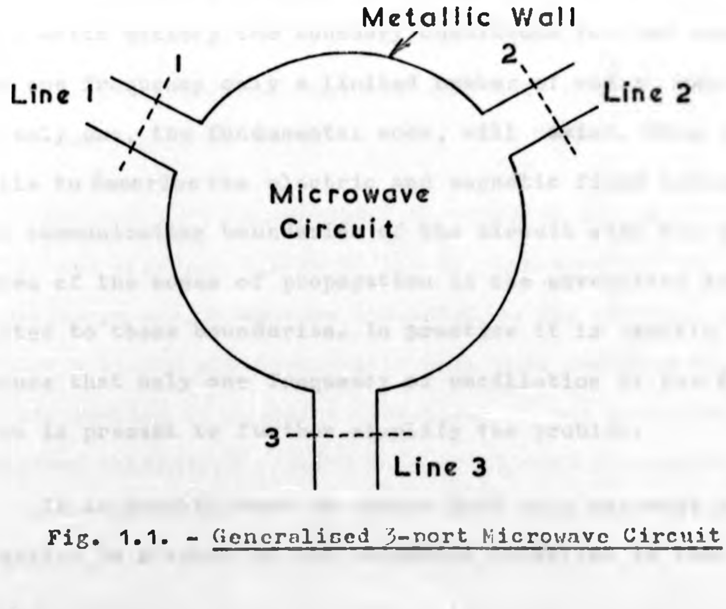


Fig. 1.1. - Generalised 3-port Microwave Circuit

Equations(1.2), within the metallic walls forming the boundary of the microwave circuit, to give a relationship between the time varying electric and magnetic fields on all the communicating boundaries of the circuit with the exterior. In practice this dynamic field problem, except for trivial microwave circuits, is impossible to solve and thus it is necessary to try to simplify the problem so it can be solved.

1.1.2. Circuit Ports

So far it has been necessary to talk in terms of the field patterns across the boundaries of the microwave circuit and the exterior and this requires some simplification. A microwave circuit usually communicates with the exterior via waveguides of some kind, e.g. coaxial lines, normal waveguides, striplines, etc., and we can usually assume that these lines extend to infinity outside the microwave circuit and have a uniform cross section. The field patterns in any waveguide of this type can be simplified into a summation of a number of modes, possibly an infinite number, which satisfy the boundary conditions for the waveguide. At any one frequency only a limited number of modes exist and often only one, the fundamental mode, will exist. Thus it is possible to describe the electric and magnetic field patterns on the communicating boundaries of the circuit with the exterior in terms of the modes of propagation in the waveguides externally connected to these boundaries. In practice it is usually necessary to assume that only one frequency of oscillation of the field pattern is present to further simplify the problem.

It is usually best to assume that only one mode of propagation is present in the waveguide connected to this circuit with the orientation of that mode defined but it may not be possible to completely define the circuit response in this way. An example of this is a circular waveguide which, in general, may carry, at one frequency, a number of modes at any orientation. This problem may be overcome by considering this type of problem as the superposition of a number of separate ports on the circuit one port for each mode of propagation in the waveguides connected to the circuit. This is illustrated in Fig. 1.2 where in the exterior circular waveguides an E_{01} mode and an H_{11} mode may be present with the latter mode having a horizontal, H^{h}_{11}, and vertical, H^{v}_{11}, component. Thus the 2-port circuit in Fig. 1.2 could be considered as a 6-port circuit with one port for each mode of propagation in the circular waveguides as shown in Fig. 1.3.

1.1.3. Description of Circuit Performance

So far we have arrived at describing a microwave circuit as an n-port network with each port carrying a given mode of propagation in the waveguides connecting it to the exterior. Now it is necessary to define the amplitudes of each mode and to describe the connection between all these modes on all the ports of the microwave circuit.

Initially it is necessary to consider the amplitudes of the electric and magnetic fields of the electromagnetic waves propagating in the waveguides connected to the circuit. In this case a single electromagnetic wave mode consists of two waves propagating in opposite directions in the waveguide with propagation constant, γ, which, in general, will be complex to include the attenuation of the waveguide or the presence of evanscent modes. Also the electric and magnetic fields transverse

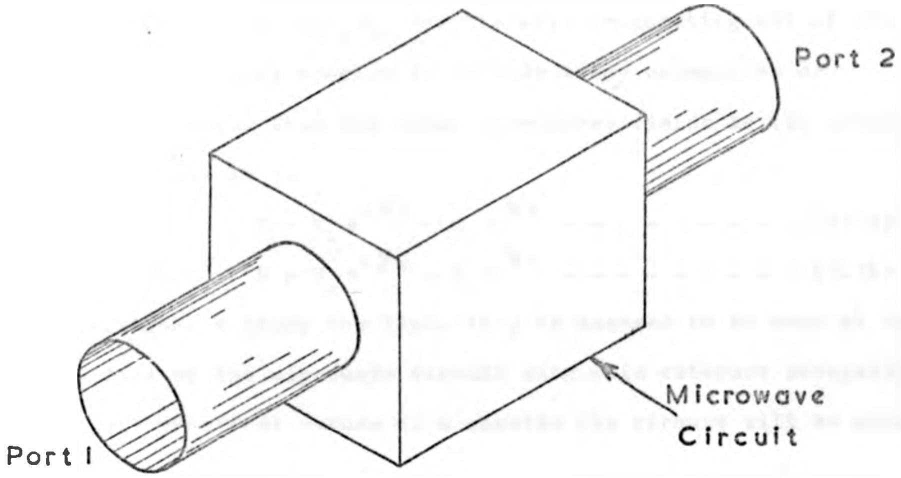


Fig. 1.2. - Microwave Circuit Connecting Circular Waveguides

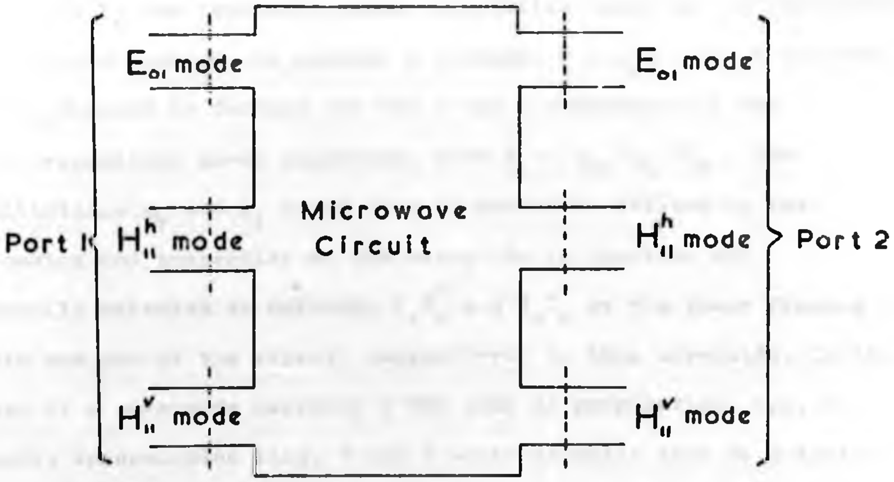


Fig. 1.3. - Equivalent Circuit for Microwave Circuit in Fig. 1.2.

to the direction of propagation are related by the wave impedance for the waveguide, Z_w, so that :-

$E_+ = Z_w H_+$ for the wave propagating into the circuit

and $E_- = Z_w H_-$ for the wave propagating out of the circuit

with Z_w in general complex to include lossy waveguides or evanscent modes. Thus the total transverse fields in the waveguide for this mode is :-

$$E = E_+ e^{-\gamma x} + E_- e^{\gamma x} \text{ - - - - - - - - - - } (1.1a)$$

and $$H = H_+ e^{-\gamma x} - H_- e^{\gamma x} \text{ - - - - - - - - - - } (1.1b)$$

for distance x along the line. If x is assumed to be zero at the interface of the microwave circuit with this exterior waveguide then all practical values of x outside the circuit will be negative.

A more convenient way to write equations (1.1) for a circuit analysis is :-

$$V = V_+ e^{-\gamma x} + V_- e^{\gamma x} \text{ - - - - - - - - - - - } (1.2a)$$

and $$I = I_+ e^{-\gamma x} - I_- e^{\gamma x} \text{ - - - - - - - - - - - } (1.2b)$$

where $V_+ = Z_o I_+$ and $V_- = Z_o I_-$

V, I and Z_o now represent scalar quantities which may be represented by complex numbers. In general a voltage, $V = g_v E$, and a current, $I = g_i H$,could be defined for the + and - components of the two propagating waves separately with $Z_o = (g_v / g_i) Z_w$. The multipliers g_v and g_i would then be constants defined by the geometry and properties of the waveguide in question and normally selected to maintain $V_+ I_+^*$ and $V_- I_-^*$ as the power flowing into and out of the circuit respectively in this waveguide. In the case of a waveguide carrying a TEM mode of propagation, i.e. a simple transmission line, V and I would normally also be selected as the voltage and current respectively on this line by suitable selection of g_v and g_i.

The most convenient way to describe a microwave circuit is to describe the electromagnetic wave or mode amplitudes flowing out of the microwave circuit as a function of the wave or mode amplitudes flowing into the circuit in every waveguide connected to the circuit. This is termed the scattering parameter description of the circuit and this will be described in detail in section 1.2.2.

1.1.4. Practical Mircowave Circuits

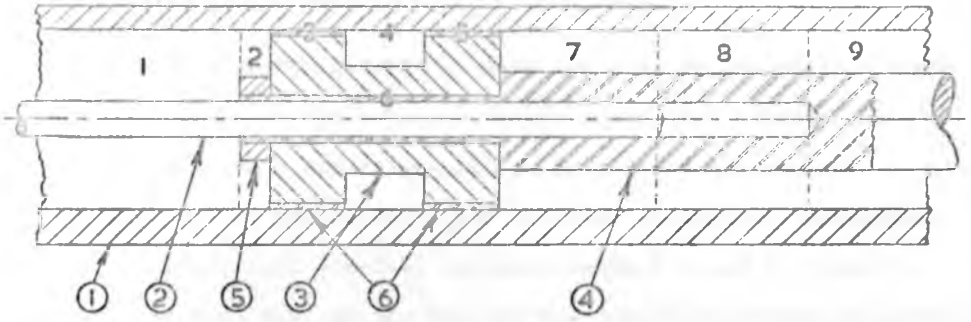
In this section a number of practical microwave circuits will be described from which a better understanding of the problems involved in the analysis of microwave circuits can be arrived at.

1.1.4.1. Practical Experience in Circuit Synthesis and Analysis

The first 5 months on the thesis was spent at Microwave Associates Ltd., Luton and during this period experience in the synthesis and analysis of Microwave Integrated Circuits was obtained. The first problem involved the design of an L-band microwave integrated circuit switch(1.3) and this is described in Appendix A.1. The second problem involved a comparative study of various types of microwave integrated circuit phase shifters(1.4) and this is described in Appendix A.2.

1.1.4.2. Coaxial Plunger

Fig. 1.4 shows a possible design for an adjustable coaxial plunger which is radially symmetric about its centre line and Fig. 1.5 shows its equivalent circuit for a TEM mode analysis. In the equivalent circuit the problem for analysis has been broken down into 9 separate sections consisting of a short section



N.B. The numbers in circles are the part numbers in the assembly whilst the others are for the equivalent circuit

Fig. 1.4. - Adjustable Microwave Coaxial Plunger

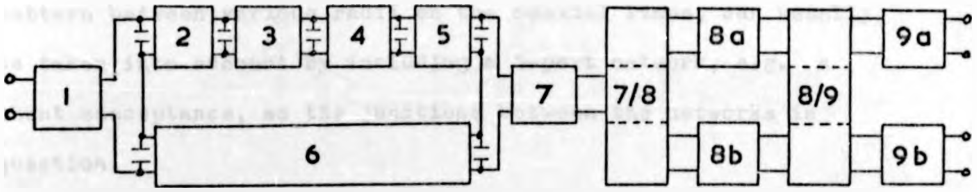


Fig. 1.5. - Equivalent Circuit for Adjustable Microwave Coaxial Plunger

from an infinite length of coaxial line, for sections 1 to 7, and an inhomogenously filled circular waveguide, for sections 8 and 9. Each of these sections can be considered as a simple distributed transmission line within a 2-port network.

For a predominately TEM mode analysis in the coaxial lines the connections of the networks in the equivalent circuit can be considered as a direct connection of voltage and current from the port of one network into the next as the wave propagates down the line, e.g. junctions between network 2 and 3, 3 and 4, 4 and 5 in Fig. 1.5. In the case of the junctions between networks 1 and 2 and 6, 5 and 6 and 7 the wave propagating down the line has to split between two networks. Also at the end of network 7 the line changes from a coaxial line into an inhomogenously filled circular waveguide in which it is assumed that two modes may propagate. Thus the 3-port network between network 7 and 8 will split the TEM mode wave into the appropriate amplitudes of these two modes. Similarly the 4-port network between networks 8 and 9 will modify the amplitudes of these two modes as the inhomogenous filling of the waveguide changes. The effects of the discontinuities in the circuit, e.g. change in the field pattern between various radii on the coaxial lines, can usually be taken into account by including a 1-port network, e.g. a shunt susceptance, on the junctions between the networks in question.

1.1.4.3. Microwave Integrated Circuit Phase Shifter

A practical microwave integrated circuit phase shifter is shown in Fig. 1.6. with its equivalent circuit in Fig. 1.7.

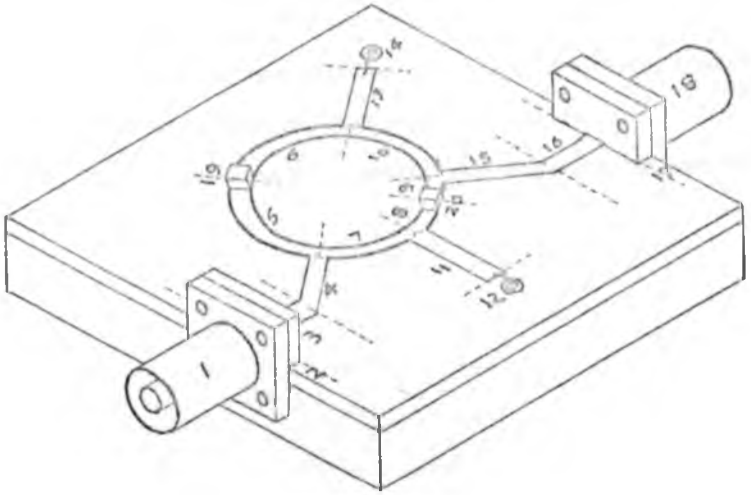


Fig.1.6.- Microwave Integrated Circuit Phase Shifter

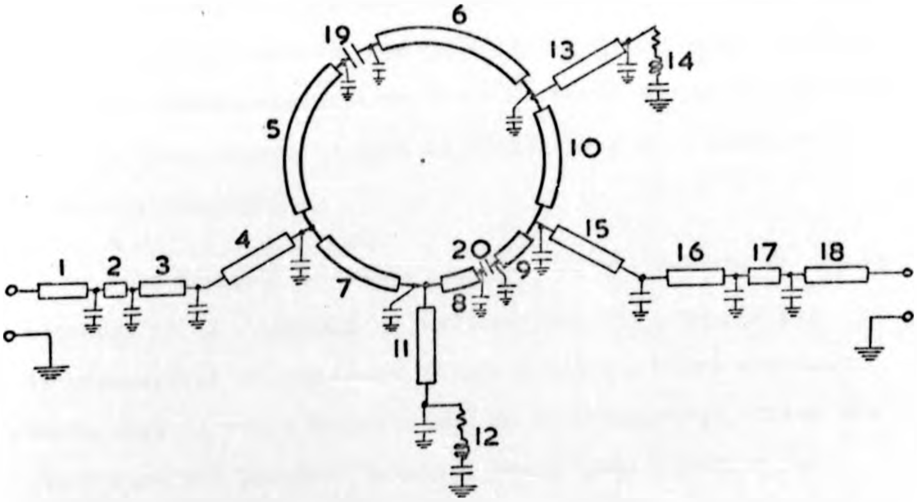


Fig.1.7.- Equivalent Circuit of Phase Shifter

This equivalent circuit consists of the following :-

1) Microstrip transmission lines for networks 2 to 11, 13 and 15 to 17. N.B. It is often assumed that only a TEM mode of propagation is present on the microstrip line.

2) p.i.n. diodes for networks 12 and 14.

3) d.c. isolating capacitors for networks 19 and 20.

1.2. MICROWAVE CIRCUIT ANALYSIS

1.2.1. Assemblies of n-port Networks

In section 1.1.4. various types of microwave circuits were presented and broken down for analysis. The complete circuit consisted of a single n-port network which could be broken down into an assembly of n-port networks each one of which was simpler to analyse on its own than the complete circuit. In a number of cases these n-port networks have a simple equivalent or approximate to a simple equivalent circuit, e.g. Tables 1.1 and 1.2. It should be noted that in a number of cases an equivalent circuit may only represent correctly the microwave component over a limited frequency range and often the component values in the equivalent circuit may vary in a complex manner with frequency.

In the analysis of assemblies of n-port networks, and in the final circuit, we are only interested in the amplitudes of the modes of propagation of the electromagnetic waves on each port of an n-port network. In practice it may be necessary to include an ideal transformer with unity turns ratio in cascade with each port on every n-port network to arrive at a more accurate equivalent circuit.

To analyse microwave circuits as an assembly of n-port networks it is necessary to consider how the networks are interconnected at junctions of the networks. There are two basic ways in which networks may be interconnected. These are the series and parallel junction connections as shown in Fig. 1.8 and 1.9 respectively. At a series junction the current is common and the voltages sum to zero whilst at a parallel junction the voltage is common and the currents sum to zero.

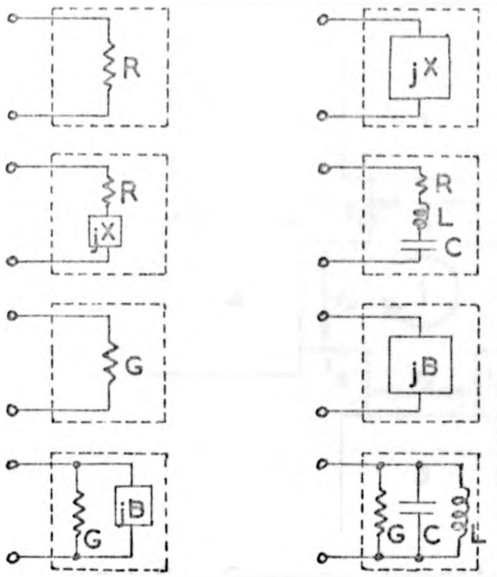
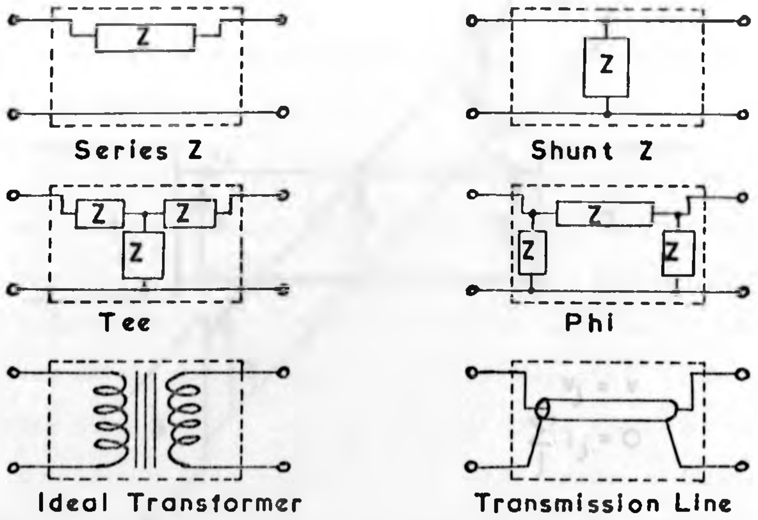


Table.1.1.- Simple 1-Port Networks


Series Z

Shunt Z

Tee

Phi

Ideal Transformer

Transmission Line

Table.1.2.- Simple 2-Port Networks

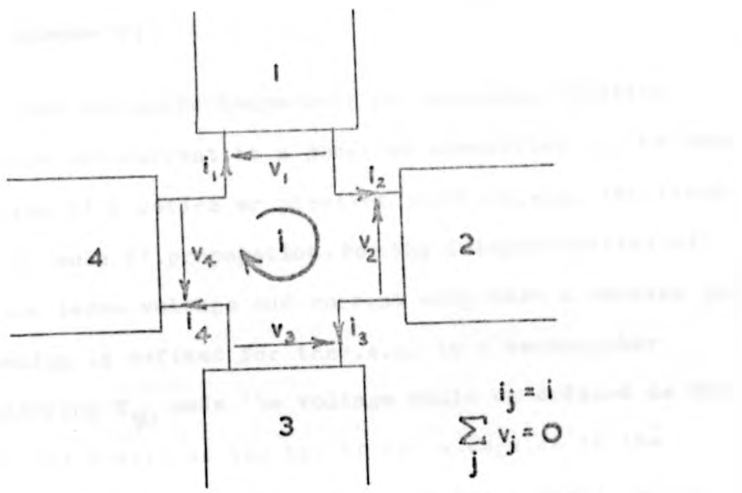


Fig.1.8.- Series Junction Connection

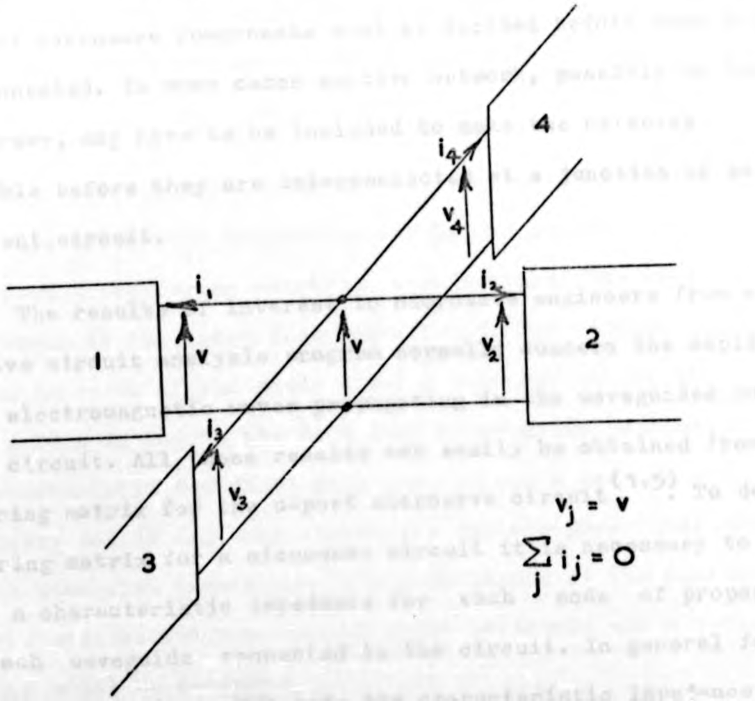


Fig.1.9.- Parallel Junction Connection

There may be some types of junctions which may be a mixture of these two types but they can always be broken up into separate series and parallel junctions using 2-port connecting link networks as necessary.

For some microwave components an immediate relation between voltage and current at a junction connection can be seen either in terms of a series or parallel junction, e.g. for lines carrying a TEM mode of propagation. For the interconnection of waveguides the terms voltage and current only have a meaning in so far as a meaning is defined for them, e.g. in a rectangular waveguide carrying H_{10} mode the voltage could be defined as the voltage from the centre of the top of the waveguide to the centre of the bottom of the waveguide with the current defined so that $v\,i^*$ represents the power flowing in the waveguide at that point. Thus the compatibility of the equivalent circuits of a number of microwave components must be decided before they are interconnected. In some cases another network, possibly an ideal transformer, may have to be included to make the networks compatible before they are interconnected at a junction in an equivalent circuit.

The results of interest to microwave engineers from a microwave circuit analysis program normally concern the amplitudes of the electromagnetic waves propagating in the waveguides connected to the circuit. All these results can easily be obtained from the scattering matrix for the n-port microwave circuit(1.5). To derive the scattering matrix for a microwave circuit it is necessary to define a characteristic impedance for each mode of propagation in each waveguide connected to the circuit. In general for a waveguide carrying a TEM mode the characteristic impedance just defines the ratio of voltage to current between the two conductors of the waveguide. For other modes in waveguides the definition of the characteristic impedance is more arbitary but it must be compatible with the voltage and current defined for that mode in the waveguide. In general the characteristic impedance may be a complex number to allow for lossy propagation in the waveguide or evanscent modes of power transfer.

1.2.2. Wave and Power Scattering Matrices

For the microwave engineer it is necessary to present the results of a microwave circuit analysis in the form of a scattering matrix for an n-port network for, in general, a different complex characteristic impedance for each port. The resulting scattering matrix is then in the form :-

$$b = S\ a$$

where b = column vector of reflected wave amplitudes

a = column vector of incident wave amplitudes

S = $n \times n$ scattering matrix

A problem arises in describing the scattering matrix for complex characteristic impedances due to the possibility of two different scattering matrices. One describes the amplitudes of the waves in the waveguides connected to the circuit and the other is in terms of the power flowing into and out of the circuit. This is due to the fact that a waveguide terminated in its characteristic impedance will not reflect a wave at its termination but it may not absorb the maximum power available whilst a waveguide terminated in the conjugate of its characteristic impedance will absorb the maximum power available but a reflected wave may still be produced.

1.2.2.1. Wave Scattering Matrix

For the wave scattering matrix the scattering parameters refer to the amplitudes of the electromagnetic modes of propagation propagating separately in the two directions in the waveguides connected to the microwave circuit, i.e. :-

$$b = S\,a \qquad (1.3)$$

where, in terms of matrices, :-

$$a = \tfrac{1}{2} R_o^{\frac{1}{2}} (Z_o^{-1} V + I) \qquad (1.4a)$$

$$b = \tfrac{1}{2} R_o^{\frac{1}{2}} (Z_o^{-1} V - I) \qquad (1.4b)$$

or

$$I = R_o^{-\frac{1}{2}} (a - b) \qquad (1.5a)$$

$$V = Z_o R_o^{-\frac{1}{2}} (a + b) \qquad (1.5b)$$

Equations (1.4) and (1.5) have been arranged so that the waves propagating in opposite directions can be separated out as :-

$V_+ = (Z_o R_o^{-\frac{1}{2}})a$ and $I_+ = R_o^{-\frac{1}{2}} a$ for the incident wave.

$V_- = (Z_o R_o^{-\frac{1}{2}})b$ and $I_- = R_o^{-\frac{1}{2}} b$ for the reflected wave.

1.2.2.2. Power Scattering Matrix

For power scattering the scattering parameters refer to the amplitude of the reflected power out of each port as a ratio of the maximum power available from the waveguides connected to the circuit. This gives $|S'_{ij}|^2$ as the transducer gain between ports j and i of the microwave circuit(1.6). Thus :-

$$b' = S' a' \qquad (1.6)$$

where, in terms of matrices, :-

$$a' = \tfrac{1}{2} R_o^{-\frac{1}{2}} (V + Z_o I) \qquad (1.7a)$$

$$b' = \tfrac{1}{2} R_o^{-\frac{1}{2}} (V - Z_o^{*} I) \qquad (1.7b)$$

or

$$I = R_o^{-\frac{1}{2}} (a' - b') \qquad (1.8a)$$

$$V = Z_o^{*} R_o^{-\frac{1}{2}} a' + Z_o R_o^{-\frac{1}{2}} b' \qquad (1.8b)$$

From equation (1.8) the power into port i (subscript i) is:-

$$= \mathrm{Re}(V_i I_i^*)$$

$$= |a_i'|^2 - |b_i'|^2$$

If the input impedance of port i is Z_{oi}^* then $b_i' = 0$ and thus for any input impedance on port i :-

$|a_i'|^2$ = incident power on port i

$|b_i'|^2$ = reflected power on port i

1.2.2.3. Wave/Power Scattering Conversion

Equating equations (1.5) and (1.8) and rearranging gives :-

1) Wave to Power Conversion

$$\begin{bmatrix} a' \\ b' \end{bmatrix} = \begin{bmatrix} R_o Z_o & 0 \\ j R_o^{-1} X_o & I \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix} \text{ - - - - - - (1.9)}$$

substituting b and b' from equations (1.4b) and (1.7b) gives :-

$$S' = (S R_o + j X_o) Z_o^{-1} \text{ - - - - - - - - - (1.10)}$$

2) Power to Wave Conversion

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} Z_o^{-1} R_o & 0 \\ -j Z_o^{-1} X_o & I \end{bmatrix} \cdot \begin{bmatrix} a' \\ b' \end{bmatrix} \text{ - - - - - - - (1.11)}$$

substituting b and b' from equations (1.4b) and (1.7b) gives :-

$$S = (S' Z_o - j X_o) R_o^{-1} \text{ - - - - - - - (1.12)}$$

It should be noted that if all the terms in Z_o are real, i.e. $Z_o = R_o$ and $X_o = 0$, then both the wave and power scattering parameters are the same, i.e. $a = a'$, $b = b'$ and $S = S'$.

1.2.3. Presentation of Results

The main circuit performance results of interest to the microwave engineer are shown in Table 1.3 with the formulae for

	From Wave Scattering Matrix	From Power Scattering Matrix
Wave Scattering Parameter, S_{ij}	S_{ij}	$S'_{ij} Z_{oj} R^{-1}_{oj}$ for $i \neq j$ (for $i = j$ see Γ_i)
Wave Reflection Coefficient, Γ_i	S_{ii}	$(S'_{ii} Z_{oi} - j X_{oi}) R^{-1}_{oi}$
Voltage Standing Wave Ratio, $VSWR_i$	$\frac{1 + \lvert S_{ii}\rvert}{1 - \lvert S_{ii}\rvert}$	$\frac{1 + \lvert S'_{ii}\rvert}{1 - \lvert S'_{ii}\rvert}$
Input Impedance, Z_{INi}	$Z_{oi} \frac{1 + S_{ii}}{1 - S_{ii}}$	$R_{oi} \frac{1 + S'_{ii}}{1 - S'_{ii}} - j X_{oi}$
Power Scattering Pararameter, S'_{ij}	$S_{ij} R_{oj} Z^{-1}_{oj}$ for $i \neq j$ (for $i = j$ see Γ'_i)	S'_{ij}
Reflection Coefficient, Γ'_i	$(S_{ii} R_{oi} + j X_{oi}) Z^{-1}_{oj}$	S'_{ii}
Transducer Gain, P_{Tij}	$\lvert R_{oj} Z^{-1}_{oj} S_{ij}\rvert^2$	$\lvert S'_{ij}\rvert^2$
Return Loss, R_{Li}	$\lvert(S_{ii} R_{oi} + j X_{oi}) Z^{-1}_{oj}\rvert^2$	$\lvert S'_{ii}\rvert^2$
Power Gain, P_{Gij}	$\frac{P_{Tij}}{1 - R_{Li}}$	$\frac{\lvert S'_{ij}\rvert^2}{1 - \lvert S'_{jj}\rvert^2}$
Available Gain, P_{Aij}	$\frac{P_{Tij}}{1 - R_{Lj}}$	$\frac{\lvert S'_{ij}\rvert^2}{1 - \lvert S'_{ii}\rvert^2}$
Insertion Gain, P_{Iij}	$\left\lvert \frac{Z_{oi} + Z_{oj}}{Z_{oi} - Z^*_{oj}} P_{Tij} \right\rvert^2$	$\left\lvert \frac{Z_{oi} + Z_{oj}}{Z_{oi} - Z^*_{oj}} S'_{ij} \right\rvert^2$

Insertion Phase, Θ_{Iij}	Arg(S_{ij}) - Arg(Z_{oj})
	+ Arg($Z_{oi} + Z_{oj}$)
	for $i \neq j$
		Arg(S'_{ij}) + Arg($Z_{oi} + Z_{oj}$)

N.B. The results of gain and return loss are often quoted in dB. To obtain these in dB then use 10 log(power ratio)

Table 1.3. - Results Obtainable from the Scattering Matrices

their derivation in terms of both the wave and power scattering parameters. For the evaluation of these parameters all the ports of the microwave circuit are terminated in the characteristic impedance for those ports unless otherwise stated.

The definitions of the results in Table 1.3 are as follows :-

1) wave scattering parameter, $S_{ij} = \dfrac{\text{amplitude of wave reflected on port i}}{\text{amplitude of wave incident on port j}}$

2) wave reflection coefficient, $\Gamma_i = \dfrac{\text{amplitude of wave reflected on port i}}{\text{amplitude of wave incident on port i}}$

3) voltage standing wave ratio, $VSWR_i = \dfrac{\text{maximum electric field along line on port i}}{\text{minimum electric field along line on port i}}$

4) input impedance, $Z_{INi} = \dfrac{\text{voltage on port i}}{\text{current into port i}}$

5) power scattering parameter, $S'_{ij} = \dfrac{\text{amplitude of power wave reflected on port i}}{\text{amplitude of power wave incident on port j}}$

6) reflection coefficient, $\Gamma'_i = \dfrac{\text{amplitude of power wave reflected on port i}}{\text{amplitude of power wave incident on port i}}$

7) transducer gain, $P_{Tij} = \dfrac{\text{power out of port i}}{\text{power available from port j}}$

8) return loss, $R_{Li} = \dfrac{\text{power out of port i}}{\text{power available from port i}}$

9) power gain, $P_{Gij} = \dfrac{\text{power out of port i}}{\text{power into port j}}$

10) available gain, $P_{Aij} = \dfrac{\text{maximum power available from port i for any termination}}{\text{power available from port j}}$

11) insertion gain, $P_{Iij} = \dfrac{\text{power out of port i with network inserted}}{\text{power out of port i with ports i and j connected directly}}$

12) insertion phase, Θ_{Iij} = (phase of current out of port i with the network inserted) - (phase of current out of port i with ports i and j connected directly)

1.3. COMPUTER AIDED DESIGN OF MICROWAVE CIRCUITS

1.3.1. Introduction

If we look at the possible requirements of a prospective user of computer aided design he would usually like the computer to design a microwave circuit to fulfil a set of performance requirements. In fact ideally he would like the computer to produce a suitable circuit design from a set of performance requirements. This can be done for special cases, e.g. for some filter circuits, where the equations for the synthesis are well defined for a given type of circuit but for general circuit design this is not possible. In practice the user would have to define or guess an initial circuit topology with the component values in that circuit. Then he could analyse this circuit on the computer to see if it fulfilled his performance requirements. If it did not then the circuit component values could be modified either manually or automatically in an optimisation routine to try to improve the circuit performance. In very complex circuits the user may prefer to design and analyse each part of the circuit in turn before analysing the entire circuit.

1.3.2. Microwave Circuit Description

Ideally one would like to include provision for the entire range of microwave components in a computer program for the small signal linear analysis of microwave circuits. The main problem here is the very large number of microwave components which could be included most of which could best be described by their physical dimensions. In section 1.1.4. each microwave component was represented by an n-port network and thus this is the way microwave components should be entered in a computer program for the analysis of microwave circuits.

1.3.3. Circuit Analysis Procedure

The objective of a circuit analysis procedure would be to provide results describing the circuit performance and, for microwave circuit analysis, it would have to consist of 2 parts as follows :-

1) From the data describing each microwave component form its description into an n-port network either in terms of a single matrix describing that n-port network or a simple equivalent circuit within an n-port network.
2) Use a circuit analysis method to analyse the circuit which now consists of an assembly of n-port networks.

For a circuit analysis on its own it is usually best to supply the user with a table of results of the circuit performance printed out for a set of frequencies defined by the user. Alternatively if an optimisation procedure is to be used then it is necessary to analyse the circuit and compare its performance with the desired performance of the circuit from which the optimisation procedure can adjust the circuit component values to try and improve the performance of the circuit.

1.3.4. Optimisation

In a circuit design it is usually impractical to let the user adjust the circuit component values manually to improve the circuit performance, particularly if a large number of component values may be adjusted. For this type of work there are a large number of optimisation procedures available which could be used and the main problem is to choose the best one(s) to use for the optimisation of microwave circuits.

1.3.5. The Computer Program

The simplest way to use a computer for computer aided design is on batch processing using the card or paper tape reader for the input data and the line printer for the results. Unfortunately the user often meets with problems in using this type of program. He will often make errors in preparing his data and he will normally take a long time to find and correct these errors. Thus often 2 or 3 runs are required on each problem and each run may take several hours, or even days, between the time it is handed in for a run on the computer and the time it is return with the results of the run. Also all he will get is the results in tabular from from which he will often have to plot graphs before he can decide whether the circuit performance is acceptable. Then, if the results of the circuit performance are not satisfactory, he will have to modify the circuit description and rerun the program with the new data.

An improvement to this method is the use of the program interactively on an on-line remote teletype. This has the advantage that the user can type all his data in directly to the program and have his results printed out directly on the remote teletype. This is extremely slow in terms of real time but if the computer is a multi-access machine then only about 2 to 3 times the processor time on batch processing will be used and the user will be able to see the results immediately after which he can decide to accept the present design or to try and improve the circuit design.

The remote teletype may seem very desirable but the user of it will soon ask for more interactive facilities from the computer program. In using a remote teletype the user would often make errors in the data and he would like to be able to correct errors without terminating the run. Thus the user would like data checking to be included in the program to enable him to do this. Also he may wish to compose the input data as he types it in. In this case the data would have to be very meaningful to him and ideally he may prefer to use a complete problem orientated programming language for microwave circuit analysis which would be very easy for him to read and understand the data and would print out the details of any errors in the data for their immediate correction.

So far the results would still be printed out in a tabular form . This is not very convenient as it would normally take a long time to decide if the results are acceptable or to decide what to do to improve them by altering the circuit description. This problem could be overcome by the use of a visual display, normally placed next to the remote teletype, on which graphs of the circuit performance could be plotted.

In the programs described in this thesis, particularly the graphical display version of the MICRO3 program, most of these objectives have been achieved and the MICRO3 program can be used, with full data checking, on batch, interactively on a remote teletype, or on a remote teletype with a graphical display next to it.

1.4. PRESENT PROGRAMS FOR MICROWAVE CIRCUIT ANALYSIS

1.4.1. Introduction

The programs previously written for the analysis of microwave circuits neatly fall into two classes.

The first class consists of the programs written for the analysis of microwave circuits which consist mainly of a cascade of 2-port networks. A large number of the simpler microwave circuits fall into this class as most microwave circuits involve the propagation of a wave or signal from the input port of the circuit to the output port with the signal being 'processed' by each part of the cascade in turn. It is usually fairly easy to transform this type of problem into a cascade of 2-port networks so a chain matrix method of analysis(1.7) can be used. In a number of programs using chain matrix analysis the method of analysis has been extended to include branch arms and/or parallel paths of cascades of 2-port networks on top of the main cascade of 2-port networks in the circuit.

The second class consists of programs written entirely for the analysis of lumped element circuits. In this case the circuit consists of a number of branch components connected between nodes in the circuit. Each branch may include a resistor, inductor, capacitor, independant and dependant voltage and current generators, mutual coupling for inductors etc.. The methods of analysis used are usually a nodal analysis(1.8) or, more recently, a mixed mesh and cutset analysis(1.9) in which a matrix equation relating all the variables in the circuit, e.g. node voltages and currents, is set up and reduced to give a set of equations relating the voltages and currents in the independant voltage and current generators in the circuit. To maintain continuity some programs of this type have been extend for use in the microwave region by replacing the microwave components in the circuit by their lumped element equivalent circuits.

The main points of most of the previous programs for microwave circuit analysis have been outlined here but there is still a need to consider the ways in which the programs can be used, e.g. data format, results provided, interactive use, etc..

1.4.2. Chain Matrix Analysis Programs

N.B. In this thesis the level of a program refers to the facilities provided by the program for design work.

1.4.2.1. MRMcPH

This program was written by McPhun(1.10) and was designed for the analysis of microwave reflection amplifiers and later extended for other circuits. It is a low level program written in Algol for an Elliott 503 computer. It includes 2-port networks only containing single R, L or C components, stub transmission lines and tunnel diode equivalent circuits. The program will analyse circuits consisting of a cascade of 2-port networks with branch arms to any level either connected to the main cascade in series or shunt.

The order in which the data is read defines the topology of the circuit and this data consists entirely of numbers, except for one title string. Thus the program is difficult to use and very sensitive to data errors. The results are in tabular form giving the frequency response over a frequency range defined by the user with up to 10 possible options included consisting of input impedance, reflection coefficient and insertion loss.

1.4.2.2. FILTAN

This program was written by Green[1.11]. It is a medium level program written for a time sharing computer using a remote teletype and the user can interact to a certain extent with the running of the program. The program includes 2-port networks only containing transmission lines, stub lines and series and shunt R, L, C combinations of components. The program will handle up to 60 networks assembled in a cascade between the input and output ports of the circuit with branch arms, series or shunt connected, and parallel paths, shunt connected only, to a level of 3 on the main cascade. The order of the input data defines the topology of the circuit and the input data includes 3 letter words to describe the network types and their topology. Command words are used in the data to control the running of the program and to edit the circuit description. Thus the program is easy to use and not too sensitive to data errors. The output is in tabular form over a frequency range defined by the user with a set number of output options, i.e. VSWR, transmission loss and phase and input resistance and reactance.

1.4.2.3. DIPNET

This program was written by Parker[1.12]. It is a low to medium level program written for a time sharing computer using a remote teletype terminal and the user can interact to a certain extent with the running of the program. The main advantage of this program is the large number of network types in the program including 2-port networks containing current and voltage generators, ideal transformers, uniform and radial coaxial lines, strip lines, coaxial step discontinuities, composite lines, R, L and C components, chain and S parameters. The circuit may include branch arms and parallel paths connected either in series or shunt. The order in which the data is read defines the topology of the circuit and the data consists of a title followed by a list of numbers. Thus the data is at a low level with a small amount of data checking in the program. The results output at each frequency consists of the frequency value followed by up to three selected output options printed out at selected points in the network.

1.4.3. Electronic Circuit Analysis Programs

1.4.3.1. ECAP

This was possibly the original general purpose electronic circuit analysis program, written by IBM(1.13), designed, at least initially, for batch processing. It is a medium level program and it is complete package including linear, d.c., a.c. and transient analysis for a lumped element circuit and uses a nodal method for the a.c. analysis. The data for the program includes a number of words and data checking is included. Thus the program is simple to use and the circuit is defined by a number of branches connected between nodes in the circuit with each branch consisting of lumped components. The results from the program consist of all the voltages at every node and all the currents in every branch in the circuit. In addition extra facilities are available, e.g. sensitivity analysis, worst case analysis.

1.4.3.2. REDAP31

This program was developed by Redac Software Ltd.(1.14) for the analysis of electronic circuits. It is a high level program designed to analyse any meaningful lumped element electronic circuit. The program uses a mixed mesh and cutset analysis developed by Branin[1.9]. The circuit to be analysed consists of lumped components , transmission lines included as 3-terminal, 2-port networks described by an admittance or scattering matrix and several electronic components, e.g. transistor equivalent circuits, from a data bank may be include. The input data is at a high level including a large number of words with full data checking and the program is thus easy to use. The output is in tabular form over a frequency range defined by the user. Most of the output options included are those of interest to the designer of electronic circuits. Also there are a number of extra facilities included, e.g. sensitivity analysis, worst case analysis, monte carlo analysis and optimisation. A recent addition is that the program can be used on-line interactively on a remote teletype.

1.4.3.3. BELNAP

This program was written by Davieau[1.15,1.16] and it appears to be a medium to high level program. It is designed for the analysis of a circuit consisting of a number of sub-circuits each one of which is described as an n-terminal device described by an admittance, impedance, scattering, h-parameter or chain matrix. Normal lumped components are also included. A nodal method of analysis is used to obtain a frequency response of the circuit over a defined frequency range. The data includes a numbers a number of words and thus the program is easy to use and the output is in tabular form or as a graph produced on a microfilm plotter.

1.5. CONCLUSIONS

The objective of this chapter was to outline the way in which the problem of the Computer Aided Design of Microwave Circuits should be tackled.

Firstly from an investigation into the general form of microwave circuits it was found that the entire microwave circuit could best be described as an n-port network. The results of interest to the microwave engineer in terms of the performance of this circuit is the relationship between the amplitudes of the modes of propagation on the waveguides or transmission lines connected to the ports of the circuit. These amplitudes can be best described in terms of the scattering parameters of the circuit.

The analysis of the circuit itself can be best tackled by breaking the general microwave circuit, as an n-port network, into an assembly of n-port networks. Each of these n-port networks representing a part or component used in the microwave circuit. Thus it is necessary to analyse an assembly of n-port networks and provide the results in terms of the scattering parameters of the circuit. The methods used so far for the analysis of electronic circuits including, in some cases, microwave circuits has been the chain matrix method of analysis or the nodal method of analysis. Thus it will be necessary to decide on the best method of analysis to use for microwave circ its.

The organisation of a computer program for microwave circuit analysis will be decided by the types of microwave circuits to be handled and the method of analysis used but in addition to this it is necessary to provide interactive facilities in the program so it is an interactive design program and not simply an analysis program.

References

1.1) C.G. Montgomery, R.H. Dicke, E.M. Purcell, 'Principles of Microwave Circuits', Dover, 1965

1.2) S. Ramo, J.R. Whinnery, T. Van Duzer, 'Fields and Waves in Communication Electronics', Wiley, 1965, Chapter 4, pp. 228-269

1.3) B.G. Marchent, 'Computer Aided Design of an L-Band M.I.C. Switch', Microwave Associates Ltd., Luton, 1969, Technical Report No. 160, (see also Appendix A.1)

1.4) B.G. Marchent, 'A Comparative Study of the Iterative, Hybrid Ring and Switched Line Microwave Phase Shifters', Microwave Associates Ltd., Luton, 1969, private communication, (see also Appendix A.2)

1.5) 'S-Parameters - Circuit Analysis and Design', Hewlett Packard, Application Note No. 95, Sept. 1968

1.6) L. Weinberg, 'Fundamentals of Scattering Matrices', Electro-Technology (USA), Vol. 80, July 1967, pp. 55-72

1.7) D.A. Calahan, 'Computer Aided Network Design', McGraw-Hill, 1968, pp. 36-39

1.8) E. Wolfendale, 'Computer-Aided Design Techniques', Iliffe, 1970, Chapter 2, pp. 15-50

1.9) F.H. Branin, 'Computer Methods of Network Analysis', Chapter 3 in F.F. Kuo, W.G. Magnuson, 'Computer Oriented Circuit Design', Prentice-Hall, 1969, pp. 71-122

1.10) M.K. McPhun, 'A Computer Program for the Analysis of Branched Distributed and Lumped Circuits', IEE Conference Publication No. 23, 1966, pp. 89-124

1.11) P.E. Green, 'General Purpose Programs for the Frequency Domain Analysis of Microwave Circuits', IEEE Trans. on Microwave Theory and Techniques, Vol. MTT-17, No. 8, August 1969, pp. 506-514

1.12) W.H. Parker, 'DIPNET: A General Distributed Parameter Network Analysis Program', IEEE Trans. on Microwave Theory and Techniques, Vol. MTT-17, No 8, August 1969, pp 495-505

1.13) H.M. Wall, C.O. Harbourt, 'ECAP: A Matrix-Oriented Approach to Network Analysis', Chapter 1 in G.W. Zobrist, 'Network Computer Analysis', MacDonald, 1969, pp. 1-77

1.14) 'REDAP31: General Circuit Analysis Program (Redacal Version)', Redacal Manual, Vol. 1, Redac Software Ltd., Tewkesbury

1.15) G.J. Herskowitz, 'Computer-Aided Integrated Circuit Design', McGraw-Hill, 1968, Chapter 4

1.16) M.A. Murray-Lasso, E.B. Kozemchak, 'Microwave Circuit Design by Digital Computer', IEEE trans. Microwave Circuits and Techniques, Vol. MTT-17, No. 8, Aug. 1969, pp. 514-526

Chapter 2

Circuit Analysis for Microwave Circuits

2.1. INTRODUCTION

In this chapter the methods of circuit analysis which may be used for the small signal linear analysis of microwave circuits in the frequency domain are considered. The objective is to choose the best method of analysis which :-

1) will handle all microwave circuits and components,
2) gives the correct, or at least an accurate, solution for any meaningful circuit,
3) uses the minimum number of operations and computation time on the computer and
4) is easy to program.

A flow diagram for a microwave circuit analysis program which will produce a table of results of the circuit performance is shown in Fig. 2.1. The method of analysis is not defined here but the flow diagram is applicable to most methods of analysis. The problems discussed in this chapter are the analysis part of the flow diagram shown within the dotted lines in Fig. 2.1.

The previous programs for the computer aided design of microwave circuits, section 1.4, have either used a chain matrix method of analysis or a nodal method of analysis. In section 2.2 the chain matrix method of analysis is described and in section 2.3 the nodal method of analysis is described for microwave circuit analysis. In chapter 1 the problems involved in microwave circuit analysis were described in a very general way as the analysis of an assembly of n-port networks. In section 2.4 a new method of analysis termed mixed matrix analysis is described which was developed for the analysis of n-port networks. In this thesis the chain matrix method of analysis was used in the CHAIN1 program, described in chapter 5, and the mixed matrix method of analysis was used in the MICRO3 program described in chapter 7.

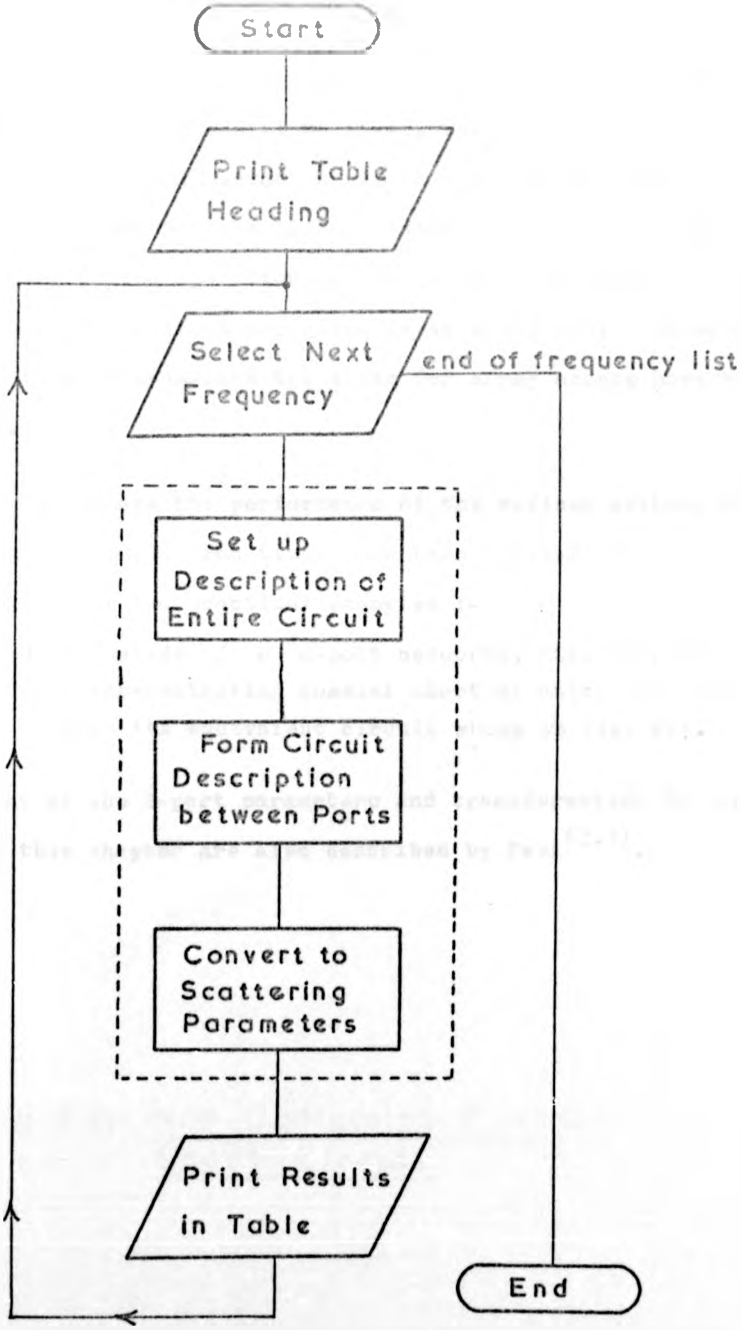

Start
Print Table Heading
Select Next Frequency
end of frequency list
Set up Description of Entire Circuit
Form Circuit Description between Ports
Convert to Scattering Parameters
Print Results in Table
End

Fig. 2.1.-Flow Diagram for Microwave Circuit Analysis

In this chapter reference will often be made to the number of operations required in a calculation. For this an equation using the term CALC on the left hand side and a list of operatiohs on the right hand side will be given. The CALC equation may also be followed by a TIME equation giving the time for this set of operations in μs for the Elliott 4130 computer with a $2\mu s$ store. The meaning of the operations given in the CALC equations and their computation times are given in Appendix A.3. To simplify some of these expressions the times for array access have been ignored.

To compare the performance of the various methods of analysis the computation times have been derived for each method of analysis for two practical examples :-

1) a cascade of n 2-port networks, Fig. 2.4, and
2) a non-contacting coaxial short circuit, Fig. 2.2, with its equivalent circuit shown in Fig. 2.3.

N.B. Most of the 2-port parameters and transformation for networks in this chapter are also described by Paul(2.1).

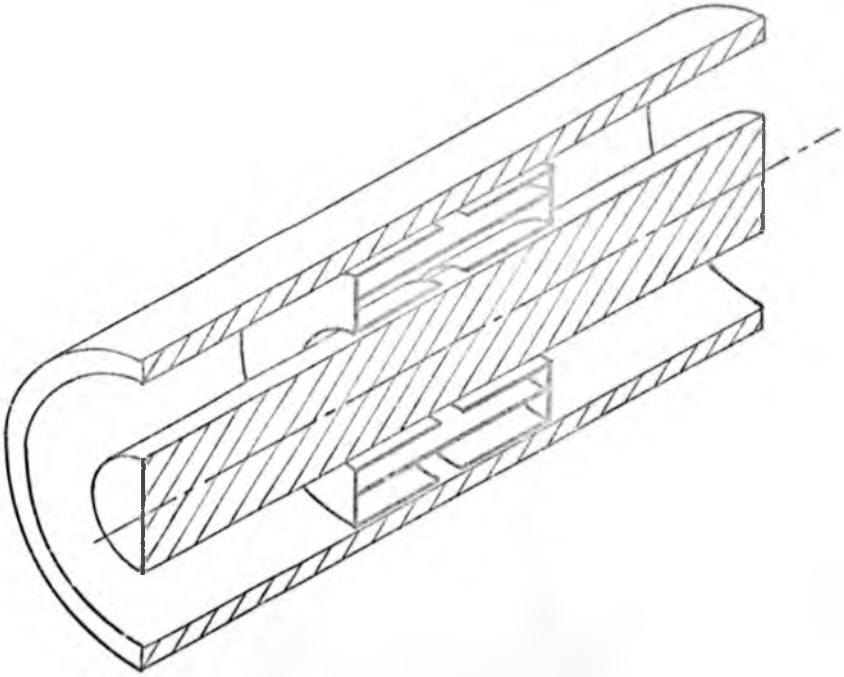
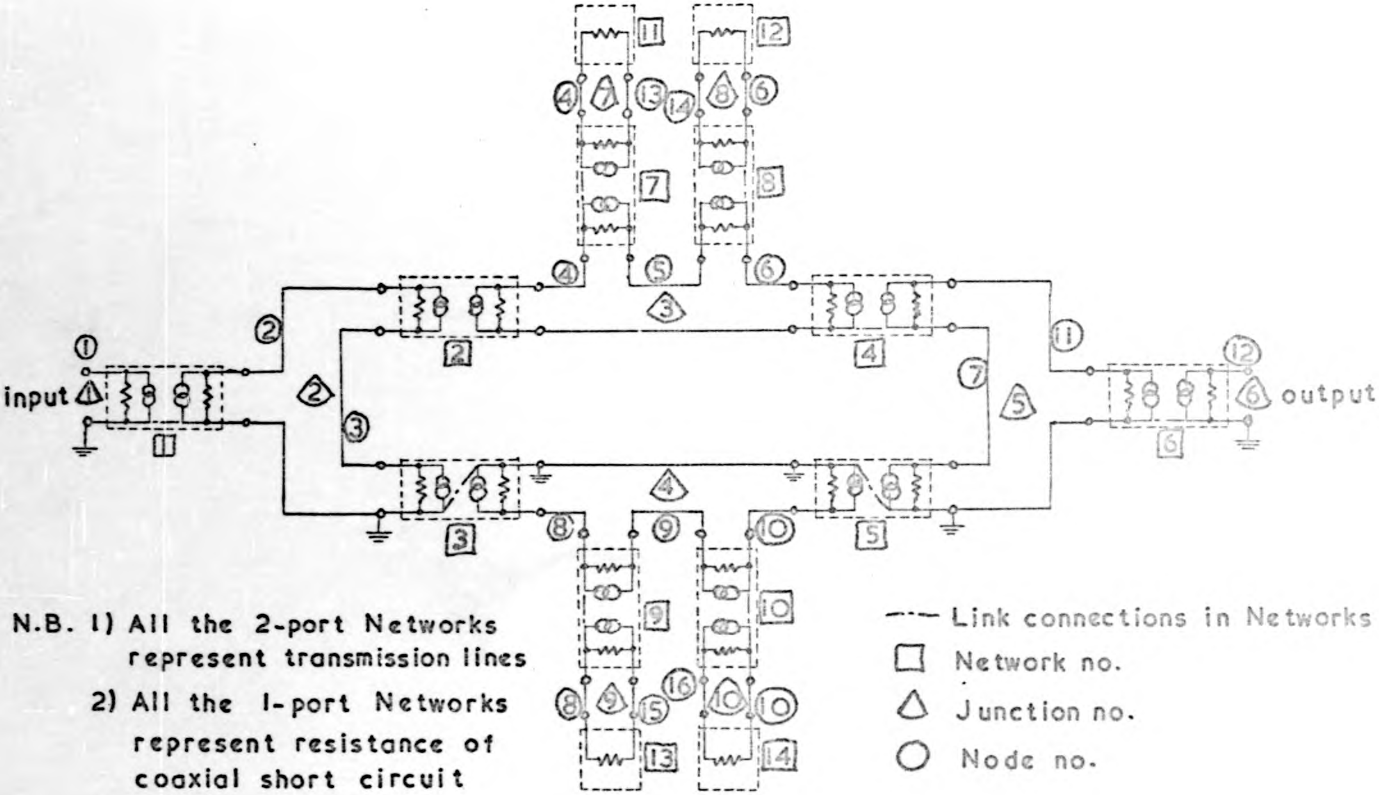


Fig. 2.2.- Non Contacting Coaxial Short Circuit



N.B. 1) All the 2-port Networks represent transmission lines

2) All the 1-port Networks represent resistance of coaxial short circuit

Fig. 2.3.- Equivalent Circuit for Coaxial Short Circuit

2.2. CHAIN MATRIX ANALYSIS

2.2.1. Introduction

The objective of a chain matrix method of analysis is to analyse a circuit as far as possible in terms of chain matrices. Chain matrix analysis is designed basically for the analysis of simple cascades of 2-port networks but it is possible to extend it application to more generalised circuits.

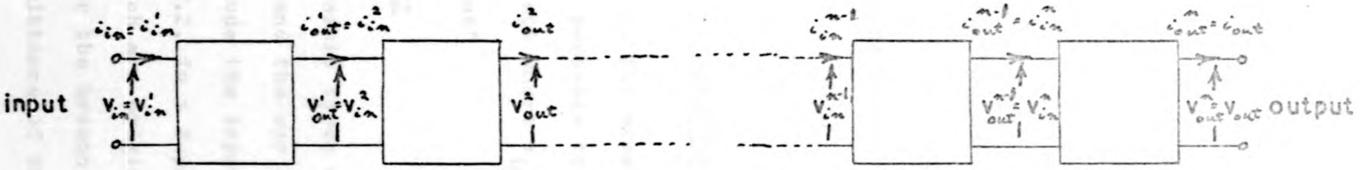
For a chain matrix analysis the circuit must consist initially of an assembly of 2-port networks. It is then necessary to break this assembly down into a simple cascade of 2-port networks between the input and output ports of the circuit. On top of this path, between the input and output ports, further paths of a similar form can be connected on this main path to any level. In section 2.2 the basic equations for a chain matrix analysis are described with the equations for all the path connection types as follows :-

1) simple cascade of 2-port networks (Fig. 2.4),
2) branch arms (Table 2.2),
3) parallel paths (Table 2.3) and
4) loop paths (Table 2.5).

2.2.2. Simple Cascade of 2-port Networks

The basic objective of chain matrix analysis is to analyse a circuit consisting of a cascade of 2-port networks, Fig. 2.4. The method used is to set up the 2 x 2 complex number chain matrix for each network in the cascade as :-

$$\begin{bmatrix} v^j_{in} \\ i^j_{in} \end{bmatrix} = \begin{bmatrix} a^j_{11} & a^j_{12} \\ a^j_{21} & a^j_{22} \end{bmatrix} . \begin{bmatrix} v^j_{out} \\ i^j_{out} \end{bmatrix} \quad \text{- - - - - - -} \quad (2.1)$$



N B To set up the chain matrix of a cascade of n 2-port networks given the chain matrix for all the networks:-

$$\text{CALC} = (n-1)(8\,\text{MULT}_j + 4\,\text{ADD}_j + 2\,\text{ASS}_j)$$

and $$\text{TIME} = (n-1)\,4744$$

Fig. 2.4.-Cascade of n 2-Port Networks

Thus on connecting the networks together in a cascade :-

$$v_{out}^{j} = v_{in}^{j+1} \quad \text{and} \quad i_{out}^{j} = i_{in}^{j+1} \quad \text{for } j = 1 \text{ to } (n-1).$$

For n 2-port networks in cascade, Fig.2.4, :-

$$\begin{bmatrix} v_{in} \\ i_{in} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} . \begin{bmatrix} v_{out} \\ i_{out} \end{bmatrix} \quad - - - - - (2.2)$$

where

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \prod_{j=1}^{n} \begin{bmatrix} a_{11}^{j} & a_{12}^{j} \\ a_{21}^{j} & a_{22}^{j} \end{bmatrix} \quad - - (2.3)$$

To set up equation (2.3) requires (n - 1) complex 2x2 matrix multiplication for a cascade of n 2-port networks. Thus the total number of operations are :-

$$\text{CALC} = (n-1)\ (8\,\text{MULT}_j + 4\,\text{ADD}_j + 2\,\text{ASS}_j)$$

and $\text{TIME} = 4744\ (n-1).$

2.2.3. Simple 2-port Networks

Chain matrices for some simple 2-port networks are given in Table 2.1. It is possible to set up a chain matrix for all 2-port networks except where v_{in} and i_{in} are not linear functions of v_{out} and/or i_{out}.

2.2.4. Branch Arms

The two possible types of branch arm connections are shown in Table 2.2 and the way in which these are included in the analysis is to include the input impedance/admittance of the branch arm, Table 2.2, in a 2-port network in the main cascade from which the branch arm originated. In the CHAIN1 program the chain matrix for the branch was first formed before the input impedance/admittance of the branch arm was calculated. It is possible to use the equation in Table 2.2 to transform the

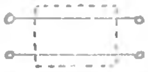
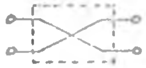
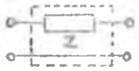
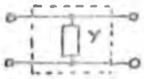
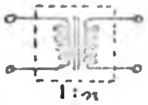
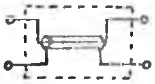
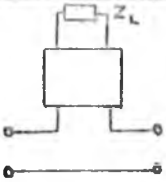
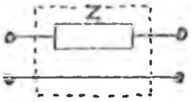
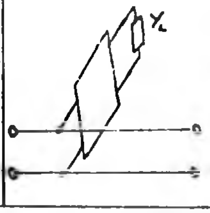
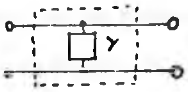
Network	Circuit	Chain Matrix	CALC	TIME
Link		$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	2 ASS_j 2 CLS_j	336
Reversed Link		$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$	2 ASS_j 2 CLS_j	336
Series Z		$\begin{bmatrix} 1 & Z \\ 0 & 1 \end{bmatrix}$	3 ASS_j CLS_j	436
Shunt Y		$\begin{bmatrix} 1 & 0 \\ Y & 1 \end{bmatrix}$	3 ASS_j CLS_j	436
Ideal Transformer		$\begin{bmatrix} 1/n & 0 \\ 0 & n \end{bmatrix}$	DIV_r 2 ASS_j 2 CLS_j	422
Transmission Line		$\begin{bmatrix} \cosh \gamma l & Z_o \sinh \gamma l \\ Y_o \sinh \gamma l & \cosh \gamma l \end{bmatrix}$ n.b. 1) $Y_o = Z_o^{-1}$ 2) Z_o and γl may be complex	$SINH_j$ $COSH_j$ EXP_j $MULT_j$ DIV_j ASS_j	8058

Table 2.1. - Chain Matrices for Simple 2-port Networks

Branch Type	Connection	Equivalent 2-port	Equations
Series			$Z = \dfrac{a_{11} Z_L + a_{12}}{a_{21} Z_L + a_{22}}$ n.b. $Z = \dfrac{a_{11}}{a_{21}}$ if $Z_L = \infty$
Parallel			$Y = \dfrac{a_{21} + a_{22} Y_L}{a_{11} + a_{12} Y_L}$ n.b. $Y = \dfrac{a_{22}}{a_{12}}$ if $Y_L = \infty$

n.b. The TIME for the calculations for a branch arm is :-

$$CALC = 2\ MULT_j + 2\ ADD_j + DIV_j$$

and $$TIME = 2075$$

Table 2.2. - Branch Arm Path Types

load impedance/admittance through one network at a time in the branch arm from the end of the branch arm to the connection point for the branch arm. This method only uses half the computation time but it was not used in the CHAIN1 program as it was simpler to design the program to first form the chain matrix for the branch arm on its own.

Assuming that the chain matrix for the branch arm has been formed then the computation time for a single branch arm is :-

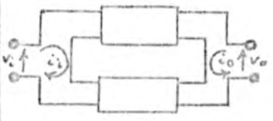
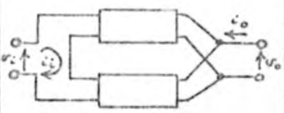
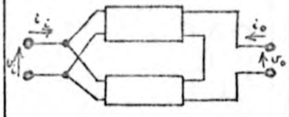
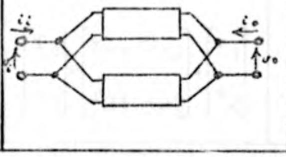
TIME = TIME to calculate input Z/Y for branch arm (Table 2.2)
+ TIME to set up chain matrix from Z/Y (Table 2.1)

for which TIME = 2511.

2.2.5. Parallel Paths

The 4 possible types of parallel path connections are shown in Table 2.3. It should be noted that the 2-port networks in each path could be the result of a reduction of a cascade of 2-port networks into the description of a single 2-port network. The method used to analyse parallel paths is to first transform the chain matrices for each of the arms of the parallel path into Z, H, G or Y parmaters, Table 2.4, as appropriate to the path type. The Z, H, G or Y parameters for all the arms of the parallel paths are added together, Table 2.3, and the result transformed back, Table 2.4, to the chain matrix for the entire parallel path. This chain matrix can then be included as a network in the cascade of 2-port netorks from which the parallel paths originated.

In the CHAIN1 program parallel paths were considered in pairs and the total computation time for 2 paths in parallel is :-

Parallel Path Type	Connection	Matrix Addition
Series to Series		$\begin{bmatrix} v_i \\ v_o \end{bmatrix} = \left\{ \begin{bmatrix} z^1_{11} & z^1_{12} \\ z^1_{21} & z^1_{22} \end{bmatrix} + \begin{bmatrix} z^2_{11} & z^2_{12} \\ z^2_{21} & z^2_{22} \end{bmatrix} \right\} . \begin{bmatrix} i_i \\ i_o \end{bmatrix}$
Series to Parallel		$\begin{bmatrix} v_i \\ i_o \end{bmatrix} = \left\{ \begin{bmatrix} h^1_{11} & h^1_{12} \\ h^1_{21} & h^1_{22} \end{bmatrix} + \begin{bmatrix} h^2_{11} & h^2_{12} \\ h^2_{21} & h^2_{22} \end{bmatrix} \right\} . \begin{bmatrix} i_i \\ v_o \end{bmatrix}$
Parallel to Series		$\begin{bmatrix} i_i \\ v_o \end{bmatrix} = \left\{ \begin{bmatrix} g^1_{11} & g^1_{12} \\ g^1_{21} & g^1_{22} \end{bmatrix} + \begin{bmatrix} g^2_{11} & g^2_{12} \\ g^2_{21} & g^2_{22} \end{bmatrix} \right\} . \begin{bmatrix} v_i \\ i_o \end{bmatrix}$
Parallel to Parallel		$\begin{bmatrix} i_i \\ i_o \end{bmatrix} = \left\{ \begin{bmatrix} y^1_{11} & y^1_{12} \\ y^1_{21} & y^1_{22} \end{bmatrix} + \begin{bmatrix} y^2_{11} & y^2_{12} \\ y^2_{21} & y^2_{22} \end{bmatrix} \right\} . \begin{bmatrix} v_i \\ v_o \end{bmatrix}$

n.b. 1) The operations for the matrix additions is :-

$\text{CALC} = 4\ \text{ADD}_j$ and $\text{TIME} = 916$

2) If there are n paths in parallel then there will be n matrices to be added together and :-

$\text{TIME} = 916\ (\ n - 1\ )$

Table 2.3. - Parallel Paths Types

Matrix	Definition	Conversion from chain matrix	Conversion to chain matrix
Z matrix	$\begin{bmatrix}v_1\\v_2\end{bmatrix}=\begin{bmatrix}z_{11}&z_{12}\\z_{21}&z_{22}\end{bmatrix}\cdot\begin{bmatrix}i_1\\i_2\end{bmatrix}$	$Z=\frac{1}{a_{21}}\begin{bmatrix}a_{11}&\Delta a\\1&a_{22}\end{bmatrix}$	$A=\frac{1}{z_{21}}\begin{bmatrix}z_{11}&\Delta z\\1&z_{22}\end{bmatrix}$
H matrix	$\begin{bmatrix}v_1\\i_2\end{bmatrix}=\begin{bmatrix}h_{11}&h_{12}\\h_{21}&h_{22}\end{bmatrix}\cdot\begin{bmatrix}i_1\\v_2\end{bmatrix}$	$H=\frac{1}{a_{22}}\begin{bmatrix}a_{12}&\Delta a\\-1&a_{21}\end{bmatrix}$	$A=\frac{1}{h_{21}}\begin{bmatrix}-\Delta h&-h_{11}\\-h_{22}&-1\end{bmatrix}$
G matrix	$\begin{bmatrix}i_1\\v_2\end{bmatrix}=\begin{bmatrix}g_{11}&g_{12}\\g_{21}&g_{22}\end{bmatrix}\cdot\begin{bmatrix}v_1\\i_2\end{bmatrix}$	$G=\frac{1}{a_{11}}\begin{bmatrix}a_{21}&-\Delta a\\1&a_{12}\end{bmatrix}$	$A=\frac{1}{g_{21}}\begin{bmatrix}1&g_{22}\\g_{11}&\Delta g\end{bmatrix}$
Y matrix	$\begin{bmatrix}i_1\\i_2\end{bmatrix}=\begin{bmatrix}y_{11}&y_{12}\\y_{21}&y_{22}\end{bmatrix}\cdot\begin{bmatrix}v_1\\v_2\end{bmatrix}$	$Y=\frac{1}{a_{12}}\begin{bmatrix}a_{22}&-\Delta a\\-1&a_{11}\end{bmatrix}$	$A=\frac{1}{y_{21}}\begin{bmatrix}-y_{22}&-1\\-\Delta y&-y_{11}\end{bmatrix}$

n.b. 1) Δx = the determinant of X

2) For any of the above conversions :-

$\text{CALC} = 6\ \text{MULT}_j + \text{ADD}_j + \text{INV}_j$

and TIME = 3368

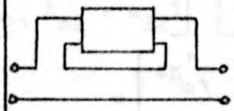
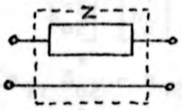
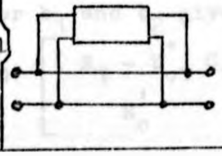
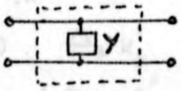
Table 2.4. - Conversions Chain/Z,H,G or Y Matrices

TIME = TIME to convert 2 chain to Z, H, G or Y matrices (Table 2.4) + TIME to add 2 matrices together (Table 2.3) + TIME to convert a Z, H, G or Y matrix to a chain matrix (Table 2.4)

TIME = 11020

2.2.6. Loop Paths

Loop paths, Table 2.5, are basically the same as the parallel paths, Table 2.3, except that one side of the parallel path is a direct link. The method of analysis for a loop path is to convert the chain matrix for the loop path into a Z or Y matrix, Table 2.4, appropriate to the path type. Then the input impedance/admittance for the loop can be caloulated, Table 2.5, and included in the path of origin of the loop path as a 2-port network, Table 2.1.

Loop Type	Connection	Equivalent 2-port	Equations
Series			set up Z matrix for for loop and then $Z = z_{11} - z_{12} - z_{21} + z_{22}$
Parallel			set up Y matrix for for loop and then $Y = y_{11} + y_{12} + y_{21} + y_{22}$

n.b. The maximum TIME to calculate Z or Y for a loop is :-

CALC = $ADD_j + 2\ SUB_j$ and TIME = 695

Table 2.5. - Loop Path Types

The computations for a loop path require :-

TIME = TIME to convert a chain to a Z/Y matrix (Table 2.4)
+ TIME to calculate the input Z/Y for the loop (Table 2.5) + TIME to set up a chain matrix from the input Z/Y for the loop (Table 2.1)

TIME = 4499

2.2.7. Conversion of Chain to Scattering Matrix

From equations (1.8), to transform equation (2.2) for the circuit to scattering parameters, using subscript 1 for the input port and subscript 2 for the output port and working in terms of the power scattering matrix :-

$$\begin{bmatrix} v_{in} \\ i_{in} \end{bmatrix} = \begin{bmatrix} Z^*_{o1}\ R^{-\frac{1}{2}}_{o1}\ a'_1 + Z_{o1}\ R^{-\frac{1}{2}}_{o1}\ b'_1 \\ R^{-\frac{1}{2}}_{o1}\ (\ a'_1 - b'_1\ ) \end{bmatrix} \quad - - - - (2.4)$$

and :-

$$\begin{bmatrix} v_{out} \\ -i_{out} \end{bmatrix} = \begin{bmatrix} Z^*_{o2}\ R^{-\frac{1}{2}}_{o2}\ a'_2 + Z_{o2}\ R^{-\frac{1}{2}}_{o1}\ b'_2 \\ R^{-\frac{1}{2}}_{o2}\ (\ a'_2 - b'_2\ ) \end{bmatrix} \quad - - - - (2.5)$$

Substituting equations (2.4) and (2.5) into (2.2) and rearranging gives :-

$$\begin{bmatrix} Z_{o1} & -(a_{11}\ Z_{o2} + a_{12}) \\ 1 & (a_{21}\ Z_{o2} + a_{22}) \end{bmatrix} \cdot \begin{bmatrix} R^{-\frac{1}{2}}_{o1} & 0 \\ 0 & R^{-\frac{1}{2}}_{o2} \end{bmatrix} \cdot \begin{bmatrix} b'_1 \\ b'_2 \end{bmatrix}$$

$$= \begin{bmatrix} -Z^*_{o1} & (a_{11}\ Z^*_{o2} - a_{12}) \\ 1 & -(a_{21}\ Z^*_{o2} - a_{22}) \end{bmatrix} \cdot \begin{bmatrix} R^{-\frac{1}{2}}_{o1} & 0 \\ 0 & R^{-\frac{1}{2}}_{o2} \end{bmatrix} \cdot \begin{bmatrix} a'_1 \\ a'_2 \end{bmatrix} - - - (2.6)$$

Solving for b_1 and b_2 gives :-

$$\begin{bmatrix} b'_1 \\ b'_2 \end{bmatrix} = \Delta_s^{-1} \begin{bmatrix} (\ A_1 - Z^*_{o1}\ C_1\ ) & \Delta_a\ R'_o \\ R'_o & (\ B_1 - Z^*_{o2}\ D_1\ ) \end{bmatrix} \cdot \begin{bmatrix} a'_1 \\ a'_2 \end{bmatrix} - - (2.7)$$

where 1) $A_1 = a_{11}\ Z_{o2} + a_{12}$ 2) $B_1 = a_{22}\ Z_{o1} + a_{12}$
3) $C_1 = a_{21}\ Z_{o1} + a_{22}$ 4) $D_1 = a_{21}\ Z_{o1} + a_{11}$
5) $\Delta_a = a_{11}\ a_{22} - a_{12}\ a_{21}$ 6) $\Delta_s = A_1 + Z_{o1}\ C_1$
7) $R'_o = 2\ R_{o1}\ R_{o2}$

If the conjugate sign is omitted from equation (2.7) then the conversion from chain to wave scattering matrix is obtained. The result in terms of a power scattering matrix for the 2-port circuit is thus :-

$$\begin{bmatrix} b'_1 \\ b'_2 \end{bmatrix} = \begin{bmatrix} S'_{11} & S'_{12} \\ S'_{21} & S'_{22} \end{bmatrix} . \begin{bmatrix} a'_1 \\ a'_2 \end{bmatrix} \text{ - - - - - - - - - - } (2.8)$$

The operations to convert from chain to scattering matrix using equation (2.7) requires :-

$$\text{CALC} = \text{INV}_j + 13\,\text{MULT}_j + 5\,\text{ADD}_j + 3\,\text{SUB}_j + 3\,\text{MULT}_r + \text{ADD}_r + \text{SQRT}_r$$

and TIME = 14297

2.2.8. Practical Computation Times

1) Cascade of n 2-port Networks (Fig. 2.4)

TIME for computation = 4744 (n - 1) see Fig. 2.4

2) Non-Contacting Coaxial Short Circuit (Fig. 2.2)

TIME for main path as cascade of 3 networks = 4744 * (3-1) (Fig. 2.4)

TIME for parallel path on main path = 11020 (section 2.2.5)

TIME for 2 sides of parallel path as cascade of 4 networks (Fig. 2.4) = 2*4744*(4-1)

TIME for 4 branch arms terminated in a load = 4*2511 (section 2.2.4)

TIME for computation = 59016

2.3. LUMPED ELEMENT ANALYSIS

2.3.1. Lumped Equivalent Circuits for n-port Networks

To analyse a microwave circuit using a lumped element analysis it is necessary to transform the individual components if the circuit into their lumped element equivalent circuit. Thus it is necessary to consider the lumped element equivalent circuit for an n-port network. In general a matrix equation relating the voltages and currents on all the ports on an n-port network could be derived to describe the performance of the n-port network and stated in the general form :-

$$\begin{bmatrix} v_s \\ i_p \end{bmatrix} = \begin{bmatrix} Z & C \\ D & Y \end{bmatrix} . \begin{bmatrix} i_s \\ v_p \end{bmatrix} \text{ - - - - - - - - - - - - } (2.9)$$

where 1) i_s and v_p are both column vectors of the port currents and voltages chosen as the independant variables.

2) v_s and i_p are both column vectors of the port voltages and currents corresponding to i_s and v_p respectively.

3) Z, C, D and Y are matrix partitions.

If we use subscript i for port i of the n-port network then :-

1) if port i has i_{si} as the independant variable then :-

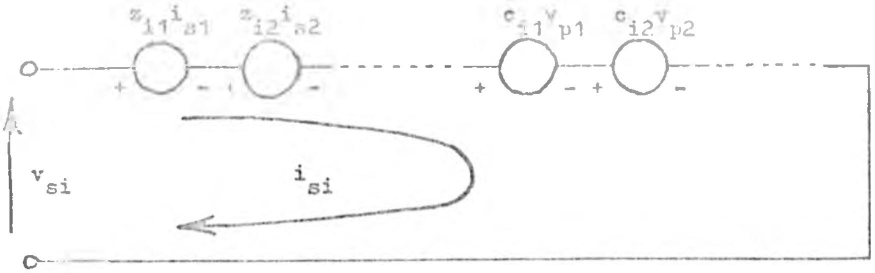
$$v_{si} = \sum_j z_{ij} \, i_{sj} + \sum_k c_{ik} \, v_{pk} \text{ - - - - - } (2.10)$$

2) if port i has v_{pi} as the independant variable then :-

$$i_{pi} = \sum_j d_{ij} \, i_{sj} + \sum_k y_{ik} \, v_{pk} \text{ - - - - - } (2.11)$$

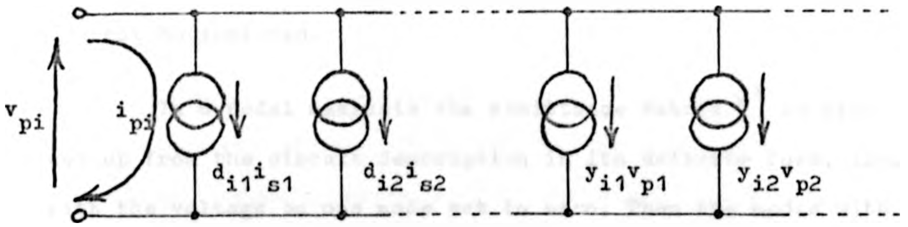
n.b. The equivalent circuits for these two cases are shown in Fig. 2.5 and 2.6 respectively.

It is possible that the n-port network being considered already consists of a lumped element equivalent circuit in which case this lumped element equivalent circuit can be used directly.



n.b. The voltage generator with voltage $Z_{ii}\, i_{si}$ can be represented as a simple impedance of Z_{ii}

Fig. 2.5. - Equivalent Circuit for a Single Port of an n-port Network with Current as the Independant Variable



n.b. The current generator with current $Y_{ii}\, v_{pi}$ can be represented as a simple admittance of Y_{ii}

Fig. 2.6. - Equivalent Circuit for a Single Port of an n-port Network with Voltage as the Independant Variable

2.3.2. Nodal Analysis

2.3.2.1. Basic Principles

The simplest method for the analysis of lumped element circuits in a computer program is to use nodal analysis(2.2). The principle is to set up a nodal admittance matrix for the entire circuit in the form :-

$$I = Y' V \text{ - - - - - - - - - - - - - - - - - - - } (2.12)$$

where V = vector of voltages on all the nodes in the circuit
I = vector of currents into all the nodes in the circuit
Y' = indefinite admittance matrix relating I to V

If one of the nodes is omitted from equation (2.12), i.e. the voltage on that node is set to zero, then the voltages on all the nodes in the circuit can be derived for a given I vector provided Y' is non-singular. There are some limitations in this method of analysis for the components in the circuit, e.g. voltage generators can not be included except in the V vector, independant current generators can not be included, and direct links between nodes can not be included.

In a nodal analysis the admittance matrix Y' is first set up from the circuit description in its definite form, i.e. with the voltage on one node set to zero. Then the nodes with no current flowing into them are eliminated from the matrix Y' to leave the nodal admittance matrix between the nodes in the circuit connected to the external voltage generators. This gives the admittance matrix of the circuit as an n-terminal network which can be translated to an n-port network description for the admittance matrix for each of the pairs of ports connected to an independant voltage generator.

2.3.2.2. N-port Networks in Nodal Analysis

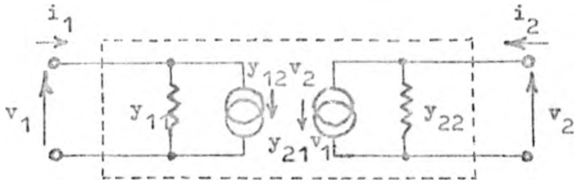
In a nodal analysis n-port networks are normally described by their n-port admittance matrix, i.e. in equation (2.9) only the partition Y is present so that :-

$$i_p = Y\, v_p \text{ - - - - - - - - - - - - - - - - - } (2.13)$$

n.b. A 2-port admittance matrix for a 2-port network is shown in Fig. 2.7.

To include n-port networks in a nodal analysis it is thus necessary to generate the lumped element equivalent circuit for the n-port networks as described in section 2.3.1, e.g. a practical example for a nodal analysis is shown in Fig. 2.3. For the analysis of an assembly of n-port networks initially the circuit will consist of a separate part, in the equivalent circuit, for each junction of the n-port networks in the circuit. It is possible to solve this problem provided only the voltage difference between pairs of nodes on the same part of the circuit are required but in practice it is usually better to connect links between the separate parts of the circuit. The condition on connecting each link is that it must be possible to construct a cutset through the link which does not cut any other current carrying path in the circuit. Thus the voltage across opposite ends of the links will all be zero and no current can flow in these links. The result of connecting these links in the analysis is that each link will merge two nodes in the circuit and reduce the number of nodes in the analysis.

A simplification to the nodal analysis will also result if the datum node is chosen wisely. It is best to arrange all the links in the circuit so that all the lower terminals on all the external ports of the circuit are connected together. This will reduce the number of non-zero elements in the nodal admittance



$$\begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

Fig. 2.7. - 2-port Admittance Matrix for a 2-port Lumped Element Network

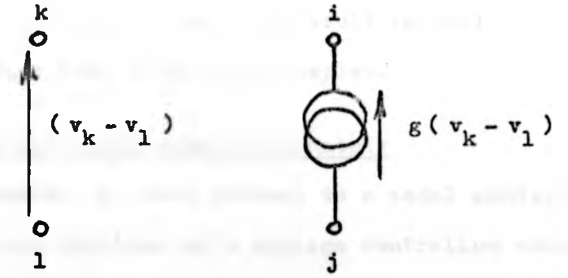


Fig. 2.8. - Basic Element for a Nodal Analysis

matrix for the circuit as the datum node should now have a very large number of the components in the circuit connected to it. Also the analysis can then proceed to derive the voltages and currents in all the upper terminals on all the external ports of the circuit only.

2.3.2.3. Computation Times and Matrix Filling

Number of Nodes

In an assembly of n-port networks :-

1) At a series junction - no. of nodes = max(c_s or 2)

2) At a parallel junction - no. of nodes = 2

where c_s = no. of networks connected to the series junction in question

Also for n_s series junctions and n_p parallel junctions in the circuit there will be $(n_s + n_p)$ separate parts of the circuit and to form the circuit into one part will require $(n_s + n_p - 1)$ links. Thus in the final equivalent circuit, ignoring the datum node, :-

$$\text{no. of nodes} = \sum_{i=1}^{n_s} (\max\ (c_s \text{ or } 2) - 1) + n_p \text{ - - - - - } (2.14$$

Matrix filling is a little more difficult to define and it is better to compare some practical examples.

To set the nodal admittance matrix

The addition of each element in a nodal analysis can be considered as the addition of a voltage controlled current generator to the nodal admittance matrix as shown in Fig. 2.8. To add this to the nodal admittance matrix requires the addition of the following elements to the matrix :-

$$\begin{array}{r|ccccc|}
 & & \text{column } k & & \text{column } l & \\
\text{row } i & \cdots & +g & \cdots & -g & \\
\text{row } j & \cdots & -g & \cdots & +g & \\
\end{array}$$

Thus to add one current generator to the nodal admittance matrix for the circuit, in its definite form, requires the following operations :-

No. of nodes connected to datum node	Operations	TIME
0	2 ADD_j + 2 SUB_j	924
1 (any node)	ADD_j + SUB_j	462
2 (not i and j or k and l)	ADD_j	229

2.3.2.4. Practical Computation Times

1) Cascade of n 2-port Networks (Fig. 2.4)

No. of Nodes	No.
Networks not interconnected	4 n
After connection at (n+1) junctions	2(n+1)
After connection of links	(n+2)
In definite admittance matrix	(n+1)

Matrix Filling

mean no. of off diagonal elements	$\frac{2n}{n+1}$

Computation Time	TIME
To clear admittance matrix	$99(n+1)^2$
To set up admittance matrix	916n
Mean time to eliminate internal nodes (Fig. 3.9)	$\approx 4.4*10^{4}$ for n=10 $\approx 2.1*10^{5}$ for n=20
Total Time	$\approx 6.5*10^{4}$ for n=10 $\approx 2.8*10^{5}$ for n=20

2) Non-Contacting Coaxial Short Circuit (Fig. 2.2)

No. of Nodes	No.
Networks not interconnected	52
After connection at 4 series and 6 parallel junctions	26
After connection of links	17
For definite admittance matrix	16

Matrix Filling

mean no. of off diagonal elements	3.1875

Computation Time	TIME
To clear admittance matrix	25344
To set up admittance matrix	29536
Mean time to eliminate internal nodes (Fig. 3.9)	$\gtrsim$ 322000
Total Time	$\approx 3.77*10^5$

2.3.3. Mixed Mesh and Cutset Analysis

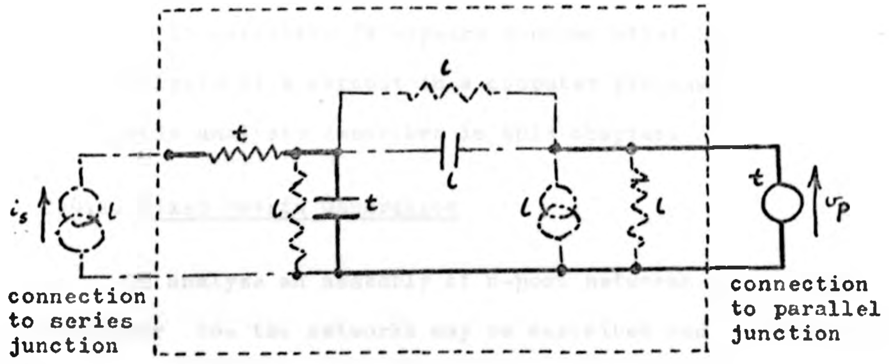
The mixed mesh and cutset method of analysis$^{(2.3)}$ is another possible way of analysing lumped element circuits. It is not of much help in the analysis of microwave circuits but there could be some n-port networks in a microwave circuit which consist of a complex lumped element equivalent circuit, e.g. a microwave transistor. In this section a very brief account of the way in which this method of analysis can be used in this case to provide a description of an n-port network for the mixed matrix analysis in section 2.4 will be described.

In the mixed mesh and cutset analysis for a lumped element circuit it is necessary first to find a tree in the circuit which includes all the voltages generators but no current generators. The remaining elements are then considered as mesh elements. The circuit analysis is then carried out in terms of the tree voltages and the mesh currents. In a microwave circuit if a single n-port network was found to have a fairly complex lumped element equivalent circuit then the mixed mesh and cutset analysis could be used to analyse this n-port network. For this method of analysis to give the required matrix for insertion in the mixed matrix analysis described in section 2.4 it is necessary to consider every port of the n-port network connected to a series junction as an independant current generator and every port connected to a parallel junction as an independant voltage generator, e.g. Fig. 2.9. The result of this method of analysis after the elimination of the internal variables in the n-port network is in the form :-

$$\begin{bmatrix} i_t \\ v_l \end{bmatrix} = \begin{bmatrix} Y & D \\ C & Z \end{bmatrix} . \begin{bmatrix} v_t \\ i_l \end{bmatrix} \text{ - - - - - - - - - - - - } (2.15)$$

where v_t vector of independant voltage generators
i_l vector of independant current generators
i_t currents in components of v_t
v_l voltages across components of i_l

In section (2.4) it will be seen that equation (2.15) is in the correct format for an n-port network for insertion directly in the mixed matrix analysis for assemblies of n-port networks.


t
l
l
l
t
l
l
i_s
l
t
v_p
connection to series junction
connection to parallel junction

t = tree element
1 = link element

Fig. 2.9. - Typical Tree and Link Branches in a Lumped Element Network

2.4. MIXED MATRIX ANALYSIS

2.4.1. Introduction

This section describes the mixed matrix method of analysis of assemblies of n-port networks. In section 1.2 a generalised microwave circuit was broken down into an assembly of n-port networks. Various authors have analysed electronic circuits as an assembly of lumped components but no one has analysed circuits as a general assembly of n-port networks. The mixed matrix analysis described in this section has been designed specially for the analysis of microwave circuits in terms of an assembly of n-port networks and the variables in the analysis are the voltages and currents in the junction connections of the n-port networks. Huelsman(2.4) has described the analysis of some assemblies of 2-port networks but only so far as a circuit consisting of 2 or 3 networks in parallel. It appears that no other work has been done in the analysis of a circuit in a computer program using the mixed matrix analysis described in this chapter.

2.4.2. Mixed Matrix Generation

To analyse an assembly of n-port networks it is necessary to consider how the networks may be described and how they are interconnected. The points of interconnections of the networks are termed junctions and there are two possible types of junctions, the series and parallel type of junction, as described in section 1.2.1. Thus :-

1) At a series junction the current i_s is common and :-

$$i_j = i_s \text{ for all } j$$

and $$v_s = \sum_j v_j \quad .$$

2) At a parallel junction the voltage v_p is common and :-

$$v_j = v_p \text{ for all } j$$

$$\text{and} \qquad i_p = \sum_j i_j \ .$$

where j includes all the networks connected to the junction in question. Thus we can set up a matrix equation relating these summations for all the junctions in the circuit as follows :-

$$\begin{bmatrix} v_s \\ i_p \end{bmatrix} = \begin{bmatrix} Z & C \\ D & Y \end{bmatrix} . \begin{bmatrix} i_s \\ v_p \end{bmatrix} \text{ - - - - - - - - - - - - (2.16)}$$

where i_s, v_p, v_s and i_p are now column vectors
suffix s and p refer to all the series and parallel junctions in the circuit respectively
Z, C, D and Y are matrix partitions

A flow diagram for a computer program using mixed matrix analysis is shown in Fig. 2.10. In this flow diagram initially the matrix in equation (2.16) is cleared, i.e. set to a zero matrix. This implies that all the series junctions are short circuited and all the parallel junctions are open circuited. Then a mixed matrix , in a similar but reduced format to equation (2.16), is set up in a buffer and then added to the matrix in equation (2.16) for each network in the circuit in turn. (N.B. the reduced format means that only the junctions to which the network in question is connected are included in the matrix equation). On the internal junctions in the circuit there are no current or voltage generators and thus, after equation (2.16) has been set up, the terms in v_s and i_p for these internal junctions can be set to zero. These variables can then be eliminated from equation (2.16), as described in section (3.3), to give a reduced mixed matrix equation describing the circuit performance between its external ports only in the form :-

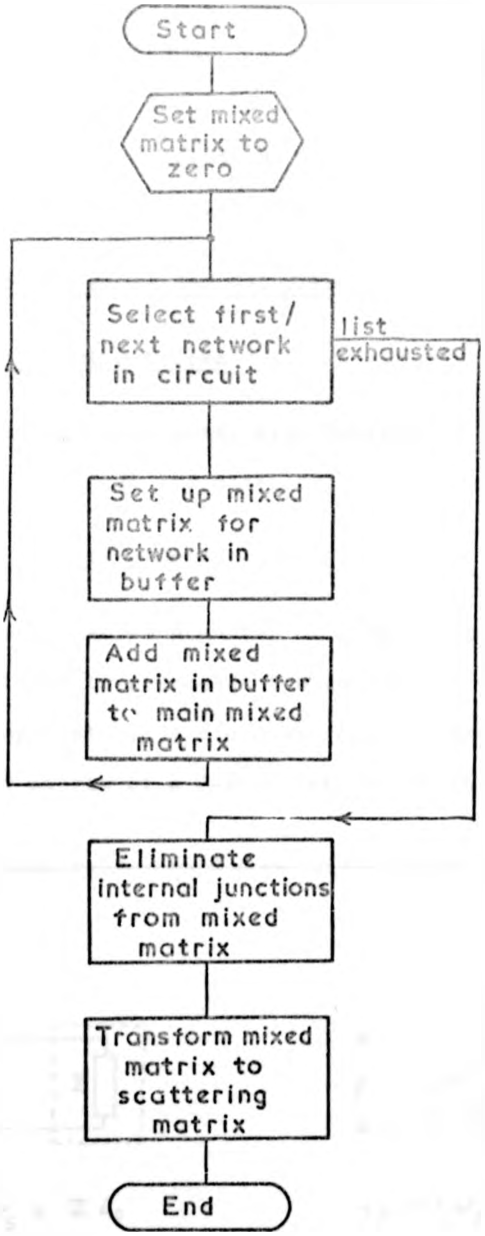

Start
Set mixed matrix to zero
Select first/next network in circuit
list exhausted
Set up mixed matrix for network in buffer
Add mixed matrix in buffer to main mixed matrix
Eliminate internal junctions from mixed matrix
Transform mixed matrix to scattering matrix
End

Fig. 2.10. - Flow Diagram for Mixed Matrix Analysis

$$\begin{bmatrix} v'_s \\ i'_p \end{bmatrix} = \begin{bmatrix} Z' & C' \\ D' & Y' \end{bmatrix} . \begin{bmatrix} i'_s \\ v'_p \end{bmatrix} \quad ----------- (2.17)$$

In both equations (2.16) and (2.17) the mixed matrix is shown neatly partitioned but in practice the rows and columns for the same junction may occur in any order. For the elimination process it is usually convenient to group the junctions connected to the exterior at the bottom right or top left of equation (2.16).

2.4.3. Mixed Matrices for Networks

2.4.3.1. 1-port Networks

For a 1-port network, e.g. Table 2.6, it is only necessary to calculate the input impedance or admittance of the network.

2.4.3.2. 2-port Networks

Table 2.7 shows a number of simple 2-port networks which could be included to represent the equivalent circuits of various microwave components in a microwave circuit analysis program. For the mixed matrix of a 2-port network it is necessary to set

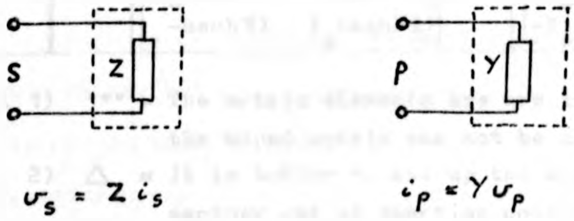
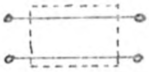
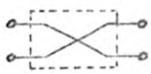
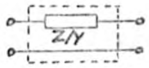
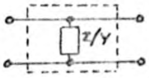
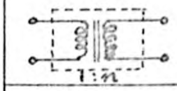
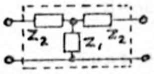
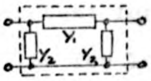
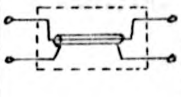


Table 2.6. - Mixed Matrix for 1-port Networks

Network	Junction Types			
	Series to Series Z matrix	Series to Parallel H matrix	Parallel to Series G matrix	Parallel to Parallel Y matrix
	$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = Z . \begin{bmatrix} i_1 \\ i_2 \end{bmatrix}$	$\begin{bmatrix} v_1 \\ i_2 \end{bmatrix} = H . \begin{bmatrix} i_1 \\ v_2 \end{bmatrix}$	$\begin{bmatrix} i_1 \\ v_2 \end{bmatrix} = G . \begin{bmatrix} v_1 \\ i_2 \end{bmatrix}$	$\begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = Y . \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$
	***	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$	***
	***	$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$	***
	***	$\begin{bmatrix} Z & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 \\ 1 & Z \end{bmatrix}$	$\begin{bmatrix} Y & -Y \\ -Y & Y \end{bmatrix}$
	$\begin{bmatrix} Z & Z \\ Z & Z \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ -1 & Y \end{bmatrix}$	$\begin{bmatrix} Y & -1 \\ 1 & 0 \end{bmatrix}$	***
	***	$\begin{bmatrix} 0 & 1/n \\ -1/n & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -n \\ n & 0 \end{bmatrix}$	***
	$\begin{bmatrix} Z_1+Z_2 & Z_1 \\ Z_1 & Z_1+Z_2 \end{bmatrix}$	△	△	△
	△	△	△	$\begin{bmatrix} Y_1+Y_2 & -Y_1 \\ -Y_1 & Y_1+Y_2 \end{bmatrix}$
	$\begin{bmatrix} Z_o \coth\gamma l & Z_o \operatorname{csch}\gamma l \\ Z_o \operatorname{csch}\gamma l & Z_o \coth\gamma l \end{bmatrix}$	$\begin{bmatrix} Z_o \tanh\gamma l & \operatorname{sech}\gamma l \\ -\operatorname{sech}\gamma l & Y_o \tanh\gamma l \end{bmatrix}$	$\begin{bmatrix} Y_o \tanh\gamma l & -\operatorname{sech}\gamma l \\ \operatorname{sech}\gamma l & Z_o \tanh\gamma l \end{bmatrix}$	$\begin{bmatrix} Y_o \coth\gamma l & -Y_o \operatorname{csch}\gamma l \\ -Y_o \operatorname{csch}\gamma l & Y_o \coth\gamma l \end{bmatrix}$

N.B. 1) *** = The matrix elements are are infinite and thus the mixed matrix can not be set up.

2) △ = It is better to set up the mixed matrix for another set of junction connection types and use an exchange of variables algorithm, section 3.1, to obtain the required mixed matrix.

3) For a transmission line Z_o and γ may be complex.

Table 2.7. - Mixed Matrices for Simple 2-port Networks

up the Z, H, G or Y parameters for a 2-port network, e.g. Table 2.4, depending on the junction types to which the network is connected. Most of the mixed matrix equations are fairly simple and they can be programmed easily in an analysis program taking advantage of similarities in the mixed matrices whenever possible to reduce the size of the program. For some of the more complex circuits it may be more convenient to set up one type of mixed matrix only and to use an exchange algorithm, section 3.1, to derive the required mixed matrix for the set of junction connections of the network.

2.4.3.3. Transmission Lines

Table 2.7 includes the mixed matrices for a transmission line but these are fairly complex and would require a large amount of code in the program to provide all the 4 types of mixed matrices. A simpler and more convenient way of deriving these mixed matrices will be described in section 2.4.6. At this point it should be noted that in general characteristic impedance and propagation constant for a transmission line will both be complex numbers to allow for lossy propagation and evanscent modes in the line.

2.4.3.4. n-port Networks

For an n-port network it is best to set up the mixed matrix which is the simplest to form, e.g. the impedance or admittance matrix for that network. Then an exchange algorithm, section 3.1, can be used to obtain the required mixed matrix. In some cases, particularly the results of practical measurement on a microwave component, it may be easiest to set up the scattering matrix for the n-port network and then to convert this to the required mixed matrix.

2.4.4. Conversion Mixed to Scattering Matrix

2.4.4.1. Power Scattering

Once the mixed matrix of the entire circuit between its external ports has been obtained then it is necessary to convert it to the scattering matrix for the circuit. The mixed matrix for the entire circuit is shown in equation (2.17) and to convert to a power scattering matrix it is necessary to substitute for v'_s, i'_p, i'_s and v'_p using equation (1.8) adding the appropriate s or p suffix to give :-

$$\begin{bmatrix} Z^*_{os} R^{-\frac{1}{2}}_{os} a'_s + Z_{os} R^{-\frac{1}{2}}_{os} b'_s \\ R^{-\frac{1}{2}}_{op} (a'_p - b'_p) \end{bmatrix} = \begin{bmatrix} Z' & C' \\ D' & Y' \end{bmatrix} \cdot \begin{bmatrix} R^{-\frac{1}{2}}_{os} (a'_s - b'_s) \\ Z^*_{op} R^{-\frac{1}{2}}_{op} a'_p + Z_{op} R^{-\frac{1}{2}}_{op} b'_p \end{bmatrix}$$

Rearranging to group b'_s with b'_p and a'_s with a'_p gives :-

$$\begin{bmatrix} Z' + Z_{os} & - C' Z_{op} \\ - D' & \Phi + Y' Z_{op} \end{bmatrix} \cdot \begin{bmatrix} R^{-\frac{1}{2}}_{os} & 0 \\ 0 & R^{-\frac{1}{2}}_{op} \end{bmatrix} \cdot \begin{bmatrix} b'_s \\ b'_p \end{bmatrix}$$

$$= \begin{bmatrix} Z' - Z^*_{os} & C' Z^*_{op} \\ - D' & \Phi - Y' Z^*_{op} \end{bmatrix} \cdot \begin{bmatrix} R^{-\frac{1}{2}}_{os} & 0 \\ 0 & R^{-\frac{1}{2}}_{op} \end{bmatrix} \cdot \begin{bmatrix} a'_s \\ a'_p \end{bmatrix} \quad --- \quad (2.18)$$

The solution of which will give the power scattering matrix for the circuit as :-

$$\begin{bmatrix} b'_s \\ b'_p \end{bmatrix} = \begin{bmatrix} S'_{ss} & S'_{sp} \\ S'_{ps} & S'_{pp} \end{bmatrix} \cdot \begin{bmatrix} a'_s \\ a'_p \end{bmatrix}$$

To obtain the power scattering matrix from the mixed matrix it is best, in a computer program, to set up the numerical form of equation (2.18) in the form :-

$$B\ b = A\ a$$

and use an algorithm to solve this type of equation, section 3.2, to give :-

$$b = B^{-1} A\ a \quad \text{or} \quad S' = B^{-1} A$$

Equation (2.18) is a very generalised equation and could be simplified if :-

1) all the terms in Z_o are real only,
2) all the real parts of Z_o are equal, or
3) only the partition Z' or Y' is present in the mixed matrix.

Typical times for the most general conversion are shown in the following table.

No. of Junctions		TIME			
n_s	n_p	Set up equation (2.18)	Multiply by $R_{os}^{-\frac{1}{2}}$ and $R_{op}^{-\frac{1}{2}}$	Solution to give S'	Total
1	0	931	829	1061	2821
0	1	1821	829	1061	3711
2	0	1862	2106	7149	11117
0	2	5422	2106	7149	14677
1	1	3642	2106	7149	12897
3	0	2793	3831	23585	30209
0	3	10803	3831	23585	38219

2.4.4.2. Wave Scattering

The conversion from mixed matrix to wave scattering matrix may be derived in the same way as for the conversion to power scattering matrix except that equations (1.5) are used for the substitution instead of equations (1.8). The equation equivalent to equation (2.18) is then :-

$$\begin{bmatrix} Z' & -C'Z_{op} \\ -D' & \Phi + Y'Z_{op} \end{bmatrix} \cdot \begin{bmatrix} R_{os}^{-\frac{1}{2}} & 0 \\ 0 & R_{op}^{-\frac{1}{2}} \end{bmatrix} \cdot \begin{bmatrix} b_s \\ b_p \end{bmatrix} = \begin{bmatrix} Z' - Z_{os} & C'Z_{op} \\ -D' & \Phi - Y'Z_{op} \end{bmatrix} \cdot \begin{bmatrix} R_{os}^{-\frac{1}{2}} & 0 \\ 0 & R_{op}^{-\frac{1}{2}} \end{bmatrix} \cdot \begin{bmatrix} a_s \\ a_p \end{bmatrix} \quad \text{-- (2.19)}$$

2.4.5. Conversion Scattering To Mixed Matrix

2.4.5.1. Power Scattering

To obtain the conversion from power scattering to mixed matrix the scattering matrix for the n-port network must first be partitioned for the ports connected to series and parallel junctions as follows :-

$$\begin{bmatrix} b'_s \\ b'_p \end{bmatrix} = \begin{bmatrix} S'_{ss} & S'_{sp} \\ S'_{ps} & S'_{pp} \end{bmatrix} . \begin{bmatrix} a'_s \\ a'_p \end{bmatrix} \text{ - - - - - - - - - - - - } (2.20)$$

Substituting equations (1.7) into equation (2.20) gives :-

$$\begin{bmatrix} \frac{1}{2} R_{os}^{-\frac{1}{2}} (v_s - Z^*_{os} \, i_s) \\ \frac{1}{2} R_{op}^{-\frac{1}{2}} (v_p - Z^*_{op} \, i_p) \end{bmatrix} = \begin{bmatrix} S'_{ss} & S'_{sp} \\ S'_{ps} & S'_{pp} \end{bmatrix} . \begin{bmatrix} \frac{1}{2} R_{os}^{-\frac{1}{2}} (v_s + Z_{os} \, i_s) \\ \frac{1}{2} R_{op}^{-\frac{1}{2}} (v_p + Z_{op} \, i_p) \end{bmatrix}$$

Rearranging to group v_s with i_s and i_s with v_p gives :-

$$\begin{bmatrix} \Phi - S'_{ss} & - S'_{sp} Z_{op} \\ S'_{ps} & Z^*_{op} + S'_{pp} Z_{op} \end{bmatrix} . \begin{bmatrix} R_{os}^{-\frac{1}{2}} & 0 \\ 0 & R_{op}^{-\frac{1}{2}} \end{bmatrix} . \begin{bmatrix} v_s \\ i_p \end{bmatrix}$$

$$= \begin{bmatrix} Z^*_{os} + S'_{ss} Z_{os} & S'_{sp} \\ - S'_{ps} Z_{os} & \Phi - S'_{pp} \end{bmatrix} . \begin{bmatrix} R_{os}^{-\frac{1}{2}} & 0 \\ 0 & R_{op}^{-\frac{1}{2}} \end{bmatrix} . \begin{bmatrix} i_s \\ v_p \end{bmatrix} \text{ - - - } (2.21)$$

The solution of which gives :-

$$\begin{bmatrix} v_s \\ i_p \end{bmatrix} = \begin{bmatrix} Z & C \\ D & Y \end{bmatrix} . \begin{bmatrix} i_s \\ v_p \end{bmatrix}$$

and this solution can be obtained numerically as described for the conversion mixed to power scattering matrix in section 2.4.4.1.

2.4.5.2. Wave Scattering

The conversion from wave scattering to mixed matrix may be derived in the same way as the conversion from power scattering except that equations (1.4) are used for the substitution instead of equations (1.7). The equation equivalent to equation (2.21) is then :-

$$\begin{bmatrix} (\Phi - S_{ss}) Y_{os} & - S_{sp} \\ S_{ps} Y_{os} & (\Phi + S_{pp}) \end{bmatrix} . \begin{bmatrix} R_{os}^{-\frac{1}{2}} & 0 \\ 0 & R_{op}^{-\frac{1}{2}} \end{bmatrix} . \begin{bmatrix} v_s \\ i_p \end{bmatrix}$$

$$= \begin{bmatrix} (\Phi + S_{ss}) & S_{sp} Y_{op} \\ - S_{ps} & (\Phi - S_{pp}) Y_{op} \end{bmatrix} . \begin{bmatrix} R_{os}^{-\frac{1}{2}} & 0 \\ 0 & R_{op}^{-\frac{1}{2}} \end{bmatrix} . \begin{bmatrix} i_s \\ v_p \end{bmatrix} \text{ - - - } (2.22)$$

Typical times for the most generalised conversion are given in the following table.

No. of junctions $n_s + n_p$	TIME			
	Set up equation (2.22)	Multiply by $R_{os}^{-\frac{1}{2}}$ and $R_{op}^{-\frac{1}{2}}$	Solution to give mixed matrix	Total
1	1376	829	1061	3266
2	3642	2106	7149	12897
3	6798	3831	23585	34214
4	10844	6004	55766	72614
5	15780	8625	109107	133512

2.4.6. Mixed matrices for Transmission Lines

The wave scattering matrix for a transmission line can be set up as :-

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 0 & e^{-\gamma l} \\ e^{-\gamma l} & 0 \end{bmatrix} . \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \text{ - - - - - - - - - - } (2.23)$$

where γ = propagation constant of the transmission line and the scattering matrix is based on the characteristic impedance of the transmission line.

This wave scattering matrix can then be converted to the required mixed matrix using the conversion described in section 2.4.5.2. The advantage of this method of setting up the mixed matrix for a transmission line is that the process is simple and common for all the types of mixed matrices required using a conversion procedure which will have to be supplied anyway for n-port networks described by a scattering matrix.

The total TIME to set up the mixed matrix for a transmission line this way is 17577 compared to 9106 to 9575 to set up the mixed matrix directly.

2.4.7. Practical Computation Times

1) Cascade of n 2-port networks (Fig. 2.4)

	No.
No. of Variables in mixed matrix	(n+1)
Mean no. of off diagonal elements in mixed matrix	$\frac{2\,n}{n+1}$
Computation Times	TIME
To clear mixed matrix	$99(n+1)^2$
To set up mixed matrix	916 n
Mean time to eliminate internal variables (Fig. 3.9)	$\approx 4.4*10^4$ for n=10 $\approx 2.1*10^5$ for n=20
Total time	$6.5*10^4$ for n=10 $2.8*10^5$ for n=20

2) Non-Contacting Coaxial Short Circuit (Fig. 2.2)

	No.
No. of variables in mixed matrix	10
Mean no. of off diagonal elements in mixed matrix	2.0
Computation Times	TIME
To clear mixed matrix	9900
To set up mixed matrix	10076
Mean time to eliminate internal variables	52212
Total time	72188

2.5. CONCLUSIONS

The main purpose of this chapter has been to outline the basis of various methods which could be used for the analysis of microwave circuits and to choose the most suitable one(s) for use in a computer program for the analysis of microwave circuits. The methods of analysis in this chapter are basically not new. In this chapter they have been fully described and compared with a comparison of the computation times involved. These are summarised in Tables 2.8 and 2.9.

The chain matrix method of analysis is simple to program and requires very little core store during the analysis. It was also the fastest of the 3 methods considered, Table 2.9. It is an analysis in terms of transmission through a cascade of 2-port networks which microwave circuits often consist of. The disadvantage was that it is very difficult to apply to generalised microwave circuits as it will only handle cascades of 2-port networks. It can be extended to include branch arms, parallel paths and loop paths but it can not handle general assemblies of n-port networks. Also it is necessary to form the circuit into an assmbly of cacades of 2-port networks, branch arms, parallel paths and loop paths before this method of analysis can be used.

The nodal method of analysis is designed for the analysis of lumped element circuits but it can be extended to include n-port networks by transforming the n-port networks into their equivalent lumped element circuits. The result of this is that often a large number of nodes are generated in the circuit particularly if series junction connections are involved. This involves setting up a large matrix for the analysis which greatly increases the computation time even if sparse matrix operations are used. There has been a large amount of work done on the theory of the analysis of lumped element circuits but almost all of this work is not applicable for the analysis of an assembly of n-port networks. The main advantage of using a lumped element analysis program for the analysis of microwave circuits is that the same program could be used for the analysis of all circuits from d.c. up to microwave frequencies.

The mixed matrix method of analysis is simple to organise and it will analyse the microwave circuit in the way it should be analysed, i.e. directly as an assembly of n-port networks. In this method one pair of variables is used for each junction connection in the circuit and thus a much smaller matrix will have to be set up. One problem is that it is necessary to provide for a number of different mixed matrix types for each n-port network. Provided all the microwave components can be broken down into a fairly small set of equivalent circuits then this should not be too much of a problem. In the text the mixed matrix for a transmission line was derived in a very general way, section 2.4.6, but it was later thought that it would be better to supply all the mixed matrix options for a transmission line to minimise the computation time, as included in Table 2.8.

One point to remember in microwave circuit analysis is that a large number of microwave components require a complex transformation to their equivalent circuit. Thus more time could be used in deriving these equivalent circuits than to analyse the resulting equivalent circuit. Even for the simple transmission lines in the example in Table 2.9 the time to set up the transmission line matrices about doubles the analysis time for problem 2 in Table 2.9.

Thus the result is that the chain matrix method of analysis is best for the analysis of simple microwave circuits which consist mainly of simple cascades of 2-port networks whilst the mixed matrix analysis is best for the analysis of generalised microwave circuits considered as a general assembly of n-port networks.

Chain Matrix Analysis	TIME
cascade of n 2-port networks	4744*(n - 1)
connection of branch arm	2511
connection of parallel path	11020
connection of loop path	4499
convert to S matrix	14297
chain matrix for transmission line	8058
Nodal Analysis	**TIME**
clear main Y matrix (n x n)	99 n^2
add single lumped component to Y matrix	924
elimination of internal nodes	see Fig. 3.7, 3.8, 3.9
convert to S matrix (2-port)	14677
Mixed Matrix Analysis	**TIME**
clear main mixed matrix (n x n)	99 n^2
add network mixed matrix (m x m)	229 m^2
elimination of internal variables	see Fig. 3.7, 3.8, 3.9
mixed matrix for transmission line	9106 to 9575
convert to S matrix (2-port)	$\leq$14677

Table 2.8. - Summary of Computation Times for Methods of Analysis

Analysis	Problem 1 (Fig. 2.4)		Problem 2 (Fig. 2.2 and Fig. 2.3)
	n=10	n=20	
Chain matrix	43 (57)	90 (104)	59 (73)
Nodal	65 (80)	280 (295)	377 (392)
Mixed matrix	65 (80)	280 (295)	72 (87)

N.B. 1) All times are in ms.

2) Time given is for analysis only not including time to set up network matrices.

3) Time in brackets includes conversion to S matrix

Table 2.9. - Computation Times for Practical Problems for Analysis

References

2.1) R.J.A. Paul, 'Two Port Parameters', Lecture 2 in IEE Vacation School on Circuit Theory, 5- 10 July 1971

2.2) E. Wolfendale, 'Computer-Aided Design Techniques', Iliffe, 1970, Chapter 2, pp. 15-50

2.3) F.H. Branin, 'Computer Methods of Network Analysis', Chapter 3 in F.F. Kuo, W.G. Magnuson, 'Computer Oriented Circuit Design', Prentice-Hall, 1969, pp. 71-122

2.4) L.P. Huelsman, 'Circuits Matrices and Linear Vector Spaces', Electronic Engineering Series, McGraw-Hill, 1963

Chapter 3

Matrix Methods for Circuit Analysis

3.1. EXCHANGE OF VARIABLES

For some of the mixed matrices for networks using the mixed matrix analysis, section 2.4.3, it is often simpler to set up one type of mixed matrix only and to use an exchange of variables algorithm to obtain the required mixed matrix. If we consider the matrix equation :-

$$\begin{bmatrix} y_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} A & b \\ c^t & d \end{bmatrix} . \begin{bmatrix} x_1 \\ y_2 \end{bmatrix} \text{- - - - - - - - - - - - -} \tag{3.1}$$

where A is a square matrix

y_1, x_1, b and c are column vectors

x_2, d, y_2 are scalar quantities

then it may be necessary to exchange the variables x_2 and y_2 to give :-

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} (A - b\,d^{-1}c^t) & b\,d^{-1} \\ -d^{-1}c^t & d^{-1} \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \text{- -} \tag{3.2}$$

This is termed an exchange of variables and the methods to do this are described by Skwirzynski(3.1). In equation (3.1) the variables to be exchanged were both at the bottom of the column vectors on both sides of this equation but in general the variables to be exchanged may be in any position in these column vectors provided they refer to the same junction. If it is necessary to exchange more than one variable then the variables should be exchanged one at a time, with the variable which would give the largest value of d being exchanged first to ensuring the greatest accuracy in the resulting equation.

The times for the exchange of a one pair of variables is given in the following table where n is the size of the matrix in equation (3.1).

n	1	2	3	4	5	6	7	8
TIME	469	2037	4961	9241	14877	21869	30217	39921

3.2. SOLUTION OF A SET OF LINEAR EQUATIONS

3.2.1. Introduction

In circuit analysis it is often necessary to solve a set of linear equation, e.g. sections 2.4.4 and 2.4.5, in the form :-

$$A\,y = B\,x \text{ - - - - - - - - - - - - - - - - - } (3.3)$$

where A and B, in all the cases considered in this thesis, are both $n \times n$ square complex number matrices

y and x are both column vectors of length n

If the matrix A is non-singular then the required solution will be given by :-

$$y = A^{-1} B\,x \text{ - - - - - - - - - - - - - - - - } (3.4)$$

In this thesis the size of the matrices, n, is typically 2,3 or 4. Due to the small size of these matrices and the fact that they will almost always be full of non-zero elements there is no point in using sparse matrix techniques but row interchanges will still be necessary to ensure that an accurate result is obtained. The solution of sets of linear equations is well documented(3.2) and the process consists of a forward elimination, normally using Guass elimination, followed by a back substitution.

3.2.2. Forward Elimination

For a forward elimination using Guass elimination the matrix equation (3.3) is multiplied by a series of pre-multiplying matrices to transform matrix A into an upper triangular matrix U in the form :-

$$U\,y = B'\,x \text{ - - - - - - - - - - - - - - - - } (3.5)$$

The flow diagram for the forward elimination process is shown in Fig. 3.1. If we consider the process at the start of the k th step then equation (3.3) has already been transformed into the

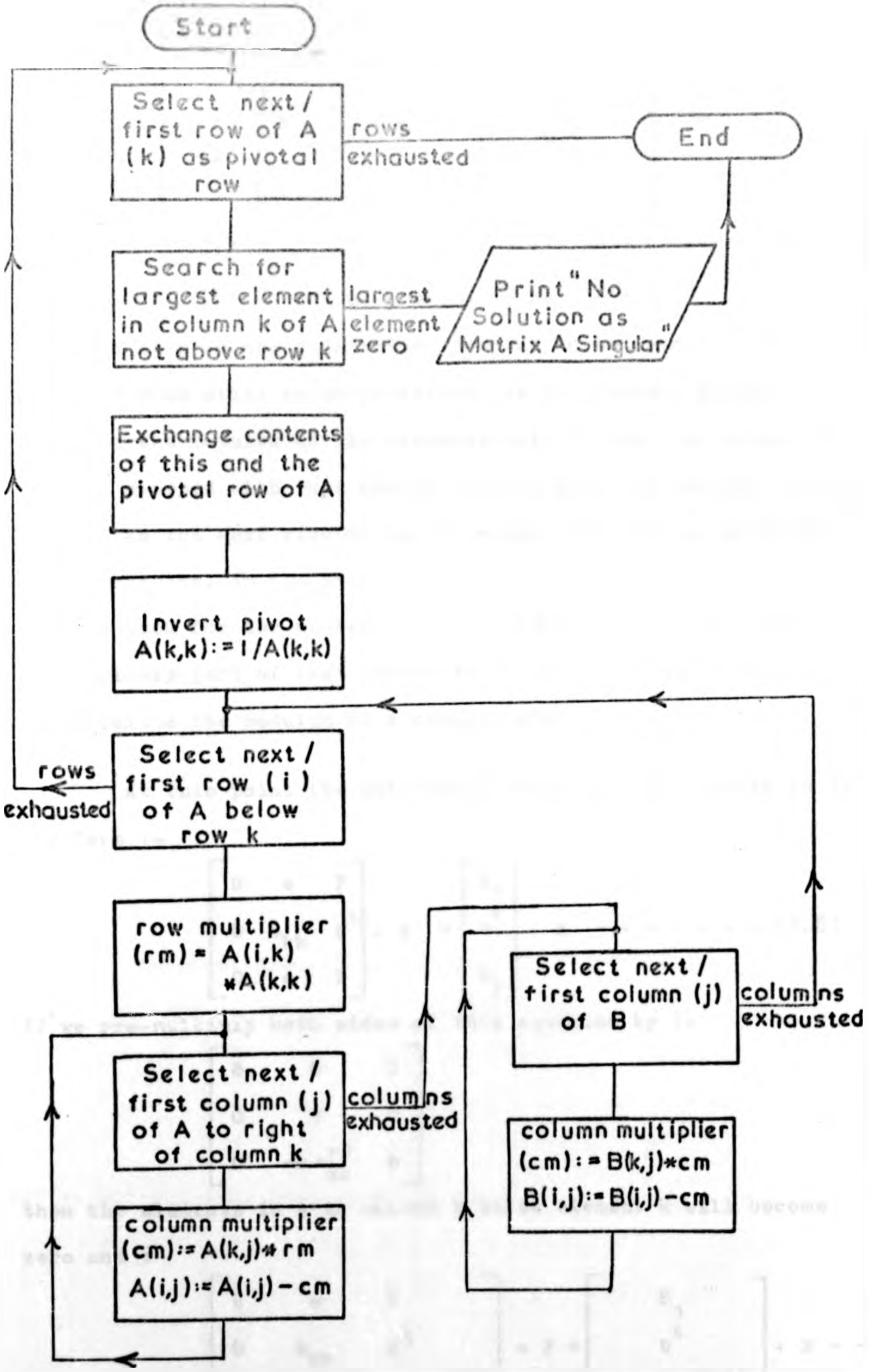
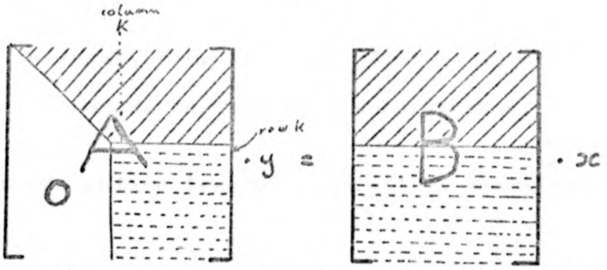

Start
Select next / first row of A (k) as pivotal row
rows exhausted
End
Search for largest element in column k of A not above row k
largest element zero
Print "No Solution as Matrix A Singular"
Exchange contents of this and the pivotal row of A
Invert pivot A(k,k):=1/A(k,k)
Select next / first row (i) of A below row k
rows exhausted
row multiplier (rm)= A(i,k) *A(k,k)
Select next / first column (j) of A to right of column k
columns exhausted
column multiplier (cm):=A(k,j)*rm
A(i,j):=A(i,j)-cm
Select next / first column (j) of B
columns exhausted
column multiplier (cm):=B(k,j)*cm
B(i,j):=B(i,j)-cm

Fig.3.1.- Flow Diagram for Forward Elimination in the Solution of a Linear Set of Equations

form :-



where //// = rows of A in which the elimination is complete

≡ = rows of A still to be processed

The next step in the elimination process is to search down column k in tre rows still to be processed for the largest element (tak_ng the modulus of the elements only). Row k in both A and B is then swapped with this row so the row with the largest element is used as the next pivotal row to ensure the maximum accuracy(3.3) in the results. In the case of complex number matrices it is best to look for the number with the largest modulus for the real or imaginary part of that number as it is a slow and long process to calculate the modulus of a complex number.

At this point the matrices in equation (3.3) would be in the form :-

$$\begin{bmatrix} U & e & F \\ 0 & a_{kk} & g^t \\ 0 & c & D \end{bmatrix} \cdot y = \begin{bmatrix} B_1 \\ b^t \\ B_2 \end{bmatrix} \cdot x \quad \text{- - - - - - - -} \quad (3.6)$$

If we pre-multiply both sides of this equation by :-

$$\begin{bmatrix} \Phi & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -c\,a_{kk}^{-1} & \Phi \end{bmatrix}$$

then the elements in A in column k below element k will become zero and :-

$$\begin{bmatrix} U & e & F \\ 0 & a_{kk} & g^t \\ 0 & 0 & (D - c\,a_{kk}^{-1} g^t) \end{bmatrix} \cdot y = \begin{bmatrix} B_1 \\ b^t \\ (B_2 - c\,a_{kk}^{-1}\,b^t) \end{bmatrix} \cdot x \quad \text{- -} \quad (3.7)$$

In the flow diagram, Fig. 3.1, the pivot is inverted and stored in a_{kk} for its use during the back substitution.

For the entire forward elimination process on equation (3.3) :-

a) essential operations without row exchanges :-

$$\text{CALC} = \sum_{k=1}^{n} \left\{ \text{INV}_j + (n-k)\text{MULT}_j + ((n-k)^2 + n(n-k))(\text{MULT}_j + \text{SUB}_j) \right\}$$

$$= n\,\text{INV}_j + \tfrac{1}{6} n(n-1) \left\{ (5n+2)\,\text{MULT}_j + (5n-1)\,\text{SUB}_j \right\}$$

b) additional operations for the row exchanges :-

$$\text{CALC} = \sum_{k=1}^{n} \left\{ (n-k+1)(\text{AMAX}_j + \text{COMP}_r) + \left(1 - \frac{1}{n-k+1}\right)\text{EXROW} \right\}$$

$$= \tfrac{1}{2} n(n+1)(\text{AMAX}_j + \text{COMP}_r) + \left(n - \sum_{j=1}^{n} \frac{1}{j}\right) \text{EXROW}$$

3.2.3. Back Substitution

The result of the forward elimination process is equation (3.5) and it is necessary to carry out a back substitution on this equation to solve for the values of x. This is done by multiplying both sides of equation (3.5) by a series of pre-multiplying matrices to form U in equation (3.5) into a unit matrix. The flow diagram for this process is shown in Fig. 3.2 and the process starts at the last row of U in equation (3.5) with k=n and works towards k=1 one row at a time. If we consider this process just before the operations on row k then equation (3.5) is in the form :-

$$\begin{bmatrix} U & e & 0 \\ 0 & a_{kk} & 0 \\ 0 & 0 & I \end{bmatrix} . y = \begin{bmatrix} B_1' \\ b^{t'} \\ B_2' \end{bmatrix} . x \quad ------ (3.8)$$

If we pre-multiply this equation by :-

$$\begin{bmatrix} I & -e\,a_{kk} & 0 \\ 0 & a_{kk}^{-1} & 0 \\ 0 & 0 & I \end{bmatrix}$$

then column k above element k will become zero in equation (3.5)

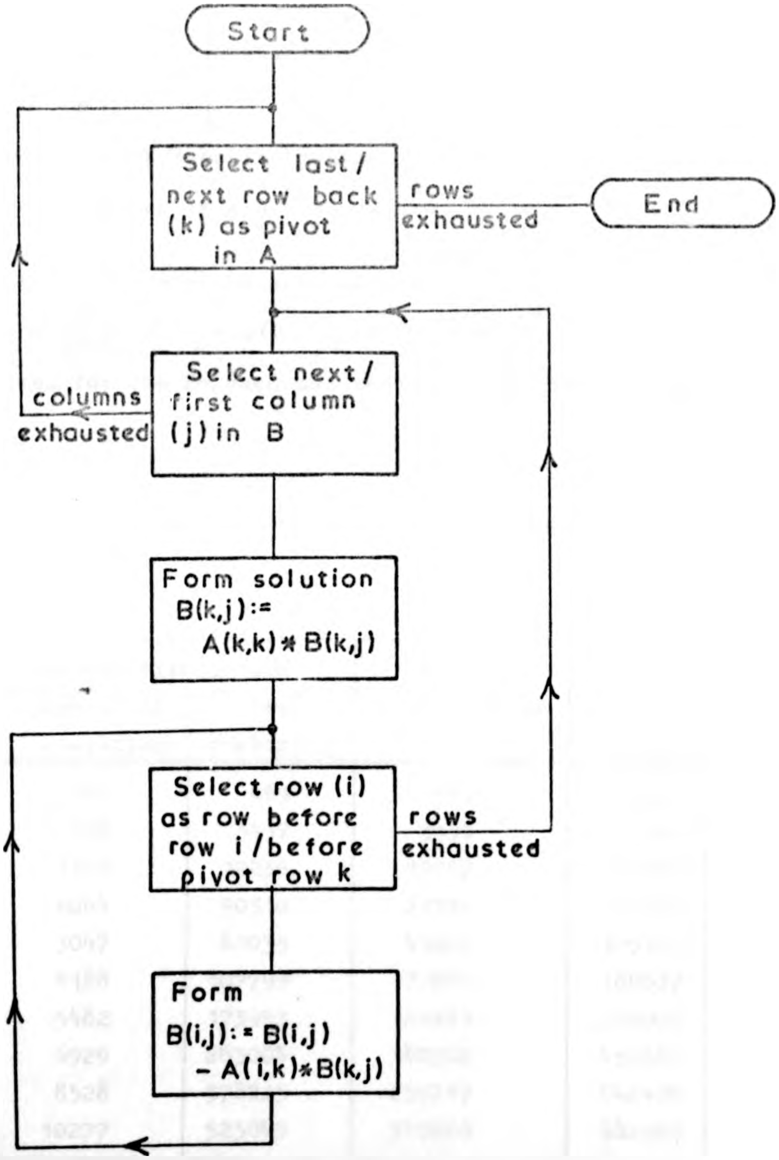

Start
Select last/ next row back (k) as pivot in A
rows exhausted
End
Select next/ first column (j) in B
columns exhausted
Form solution B(k,j):= A(k,k)*B(k,j)
Select row (i) as row before row i/before pivot row k
rows exhausted
Form B(i,j):= B(i,j) - A(i,k)*B(k,j)

Fig.3.2.- Flow Diagram for Back Substitution in the Solution of a Linear Set of Equations

to give :-

$$\begin{bmatrix} U & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & e \end{bmatrix} . y = \begin{bmatrix} (B_1' - e\,a_{kk}^{-1}\,b^{t'}) \\ a_{kk}^{-1}\,b^{t'} \\ B_2' \end{bmatrix} . x \quad ------ (3.9)$$

At the end of the back substitution :-

$$e\,y = B''\,x \quad \text{or} \quad y = B''\,x \quad \text{where} \quad B'' = A^{-1}\,B$$

The operations for the back substitution are :-

$$\text{CALC} = \sum_{k=1}^{n} \left\{ n\,\text{MULT}_j + n\,(k-1)\,(\text{MULT}_j + \text{SUB}_j) \right\}$$

The total times for the forward and back substitution are given in Table 3.1.

Matrix Size n	Forward elimination		Back Substitution	Total
	essential operations	row exchanges		
1	147	469	445	1061
2	596	3417	3136	7149
3	1244	12234	10107	23585
4	2064	30310	23392	55766
5	3047	61035	45025	109107
6	4188	107799	77040	189027
7	5482	173992	121471	300945
8	6929	263004	180352	450285
9	8528	378225	255717	642470
10	10277	523045	349600	882922

Table 3.1. - Computation Times for Solution of a Set of Linear Equations

3.3. ELIMINATION OF INTERNAL VARIABLES IN ANALYSIS

3.3.1. Introduction

In circuit analysis it is necessary first to relate all the variables in the circuit, and then all the internal variables must be eliminated from this relationship to leave a relationship of the variables on the external ports or nodes of the circuit. In a chain matrix analysis, section 2.2, these internal variables are eliminated as the analysis proceeds down a cascade of 2-port networks whilst in a nodal analysis, section 2.3, or a mixed matrix analysis, section 2.4, a matrix equation is first set up to relate all the variables in the circuit in the form :-

$$\begin{bmatrix} y_i \\ y_e \end{bmatrix} = \begin{bmatrix} A_{ii} & A_{ie} \\ A_{ei} & A_{ee} \end{bmatrix} \cdot \begin{bmatrix} x_i \\ x_e \end{bmatrix} \text{ - - - - - - - - - (3.10)}$$

where y and x are column vectors
A is a partitioned square matrix
i and e refer to the internal and external variables in the circuit respectively

For all the internal variables in the circuit the components of y_i are all zero and the relation between y_e and x_e is required. This can be obtained from :-

$$y_e = (A_{ee} - A_{ei} A_{ii}^{-1} A_{ie}) \, x_e \text{ - - - - - - (3.11)}$$

but this is not the most efficient way of deriving this relationship in terms of speed and accuracy.

The problem considered in this section is thus the best method for the elimination of the internal variables in the circuit to give :-

1) A solution for cases where a solution is obtainable
2) A correct solution or a solution with the maximum accuracy
3) A solution with the least computation time remembering

3.3. ELIMINATION OF INTERNAL VARIABLES IN ANALYSIS

3.3.1. Introduction

In circuit analysis it is necessary first to relate all the variables in the circuit and then all the internal variables must be eliminated from this relationship to leave a relationship of the variables on the external ports or nodes of the circuit. In a chain matrix analysis, section 2.2, these internal variables are eliminated as the analysis proceeds down a cascade of 2-port networks whilst in a nodal analysis, section 2.3, or a mixed matrix analysis, section 2.4, a matrix equation is first set up to relate all the variables in the circuit in the form :-

$$\begin{bmatrix} y_i \\ y_e \end{bmatrix} = \begin{bmatrix} A_{ii} & A_{ie} \\ A_{ei} & A_{ee} \end{bmatrix} \cdot \begin{bmatrix} x_i \\ x_e \end{bmatrix} \text{ - - - - - - - - - } (3.10)$$

where y and x are column vectors
A is a partitioned square matrix
i and e refer to the internal and external variables in the circuit respectively

For all the internal variables in the circuit the components of y_i are all zero and the relation between y_e and x_e is required. This can be obtained from :-

$$y_e = (A_{ee} - A_{ei} A_{ii}^{-1} A_{ie}) \, x_e \text{ - - - - - } (3.11)$$

but this is not the most efficient way of deriving this relationship in terms of speed and accuracy.

The problem considered in this section is thus the best method for the elimination of the internal variables in the circuit to give :-

1) A solution for cases where a solution is obtainable
2) A correct solution or a solution with the maximum accuracy
3) A solution with the least computation time remembering

that the A matrix in equation (3.10) will be full of mainly zero elements, typically only 15 to 40% of the elements in A are non-zero.

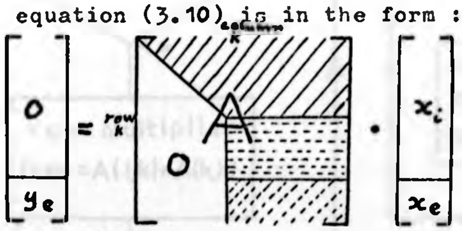
3.3.2. Elimination Process

After an investigation of the various methodsof elimination available the method of Aitken(3.4) was found to be the most suitable. This method is basically the same as Gauss elimination for a set of linear equations, section 3.2.2, except that the triangularisation or forward elimination process is only carried out on the matrix A in equation (3.10) and the process is stopped half way when all the variables in x_i in equation (3.10) have been eliminated. After this process on equation (3.10) this equation is in the form :-

$$\begin{bmatrix} 0 \\ y_e \end{bmatrix} = \begin{bmatrix} U & A'_{ie} \\ 0 & A'_{ee} \end{bmatrix} \cdot \begin{bmatrix} x_i \\ x_e \end{bmatrix} \text{ - - - - - - - - - } (3.12)$$

where $A'_{ee} = (A_{ee} - A_{ei} A^{-1}_{ii} A_{ie})$ as in equation (3.11)

The flow diagram for Aitkens elimination on equation (3.10), Fig. 3.3, is very similar to the flow diagram for a forward elimination on a set of linear equations, Fig. 3.1, except for the points described below. If we consider the elimination of the k th variable in x_i then just before the elimination of this variable equation (3.10) is in the form :-



where ▨ = rows which have already been eliminated

░ = rows still to be processed in x_i

▓ = rows for variables assigned as external variables

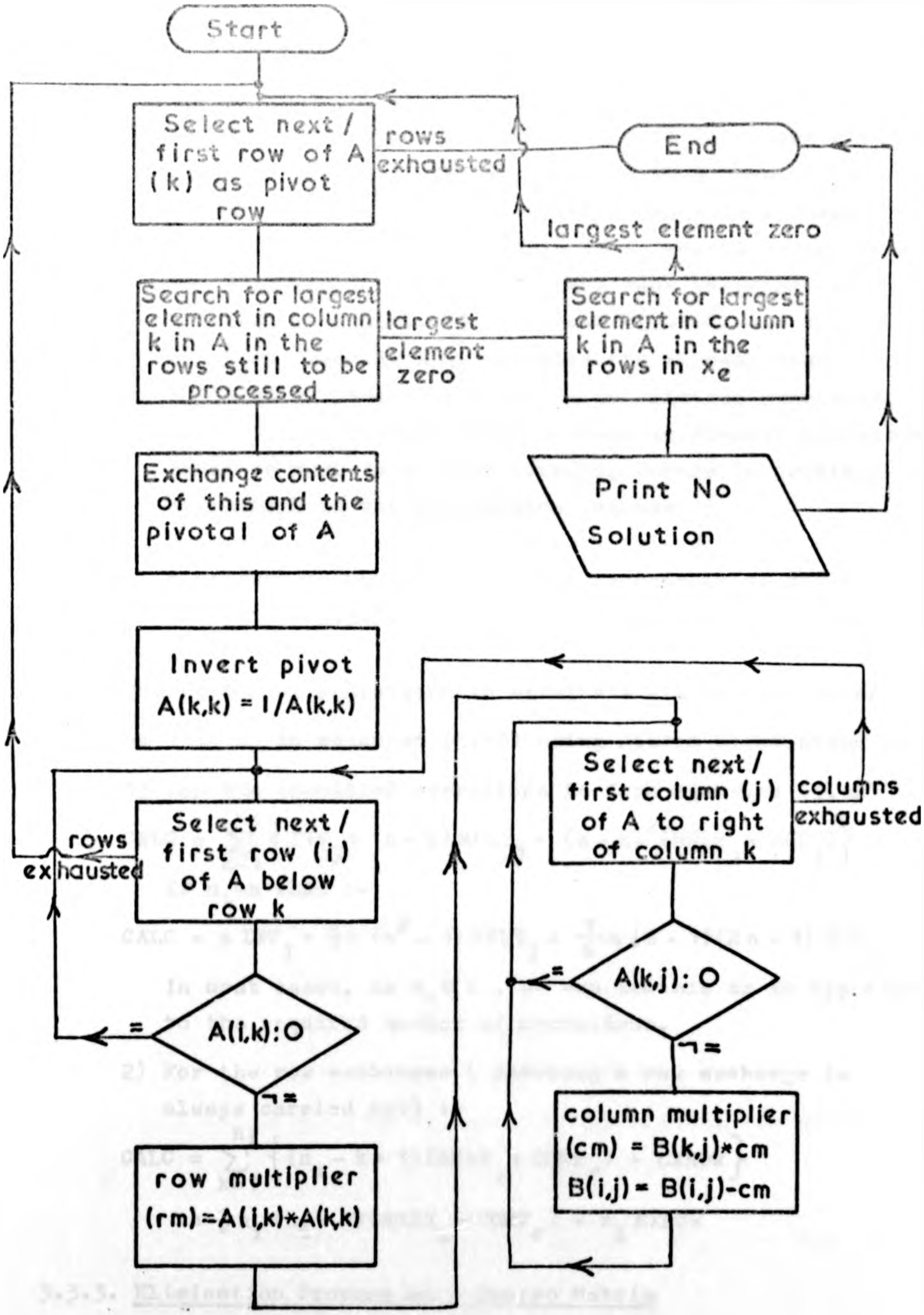

Start
Select next / first row of A (k) as pivot row
rows exhausted
End
largest element zero
Search for largest element in column k in A in the rows still to be processed
largest element zero
Search for largest element in column k in A in the rows in x_e
Exchange contents of this and the pivotal of A
Print No Solution
Invert pivot A(k,k) = 1/A(k,k)
Select next / first column (j) of A to right of column k
columns exhausted
Select next / first row (i) of A below row k
rows exhausted
A(k,j): O
=
¬=
A(i,k):O
=
¬=
column multiplier (cm) = B(k,j)*cm B(i,j) = B(i,j)-cm
row multiplier (rm)=A(i,k)*A(k,k)

Fig. 3.3.- Flow Diagram for Aitkens Elimination on a Sparse Matrix

The differences in the Aitken elimination are :-

1) The process is performed on the A matrix only
2) Row interchanges are only allowed in the rows still to be processed in x_i
3) If there are no non-zero elements left in column k in A for the rows still to be processed in x_i then this could mean that the k th variable has already been eliminated. To check for this it is necessary to check the elements in column k for the rows assigned as external variables. If all these elements are also all zero then the k th variable has already been eliminated. Otherwise there is no solution available. (N.B. a zero element could include an element approximately equal to zero to include rounding errors on previous operations in the elimination process.)
4) Matrix A will be fairly sparsely filled and it will be necessary to use sparse matrix techniques as described in section 3.3.3.

The number of operations to eliminate all the variables in x_i from 1 to n_i in equation (3.10) using Aitken elimination is :-

1) For the essential operations to form equation (3.12) :-

$$\text{CALC} = \sum_{k=1}^{n_i} \left\{ \text{INV}_j + (n-k)\,\text{MULT}_j + (n-k)^2(\text{MULT}_j + \text{SUB}_j) \right\}$$

If $n_i = n$ then :-

$$\text{CALC} = n\,\text{INV}_j + \tfrac{1}{3}n(n^2-1)\,\text{MULT}_j + \frac{1}{6}n(n-1)(2n-1)\,\text{SUB}_j$$

In most cases, as $n_i \approx n$, we can use this as an approximation to the required number of operations.

2) For the row exchanges (assuming a row exchange is always carried out) :-

$$\text{CALC} = \sum_{k=1}^{n_i} \left\{ (n_i - k + 1)(\text{AMAX}_j + \text{COMP}_r) + \text{EXROW} \right\}$$

$$= \tfrac{1}{2}n_i\,(n_i+1)(\text{AMAX}_j + \text{COMP}_r) + n_i\,\text{EXROW}$$

3.3.3. Elimination Process on a Sparse Matrix

After the pivotal row has been selected in the elimination process on the k th row it is necessary to modify all the elements on all the rows still to be processed and the rows assigned to the external variables in equation (3.10) so that :-

$$a_{ij} = a_{ij} - a_{ik}\, a_{kk}^{-1}\, a_{kj}$$

In the flow diagram in Fig. 3.3 each row (subscript i) is considered in turn and a multiple of the pivotal row (subscript k) is subtracted from this row to make $a_{ik}=0$. During this process the following saving can be made if various elements are zero :-

1) Zero pivot - this was described in section 3.3.2.

2) Zero row multiplier (i.e. $a_{ik}=0$)

extra CALC = ZERO_j

saved if $a_{ik}=0$ CALC = $\text{MULT}_j + (n - k)(\text{MULT}_j + \text{SUB}_j)$

3) Zero column multiplier (i.e. $a_{kj}=0$)

extra CALC = ZERO_j

saved if $a_{kj}=0$ CALC = $\text{MULT}_j + \text{SUB}_j$

4) Zero element (i.e. $a_{ij}=0$)

extra CALC = ZERO_j

saved if $a_{ij}=0$ CALC = SUB_j

It was not considered that any advantage would be gained in practice by checking for zero element (i.e. $a_{ij}=0$)

3.3.4. Elimination of a Single Variable

To estimate the forward elimination time for practical matrices in circuit analysis it is simpler to consider the elimination of a given variable which is connected to n_c other variables in the circuit. This is illustrated for the resistive network in Fig. 3.4 for which the non-zero elements in the pivotal row and column in the admittance matrix for the circuit are :-

$$\begin{bmatrix} i_{10} \\ i_{15} \\ i_{26} \\ i_{31} \end{bmatrix} = \begin{bmatrix} y_{10,10} \cdots y_{10,15} \cdots y_{10,26} \cdots y_{10,31} \\ y_{15,10} \\ y_{26,10} \\ y_{31,10} \end{bmatrix} \cdot \begin{bmatrix} v_{10} \\ v_{15} \\ v_{26} \\ v_{31} \end{bmatrix}$$

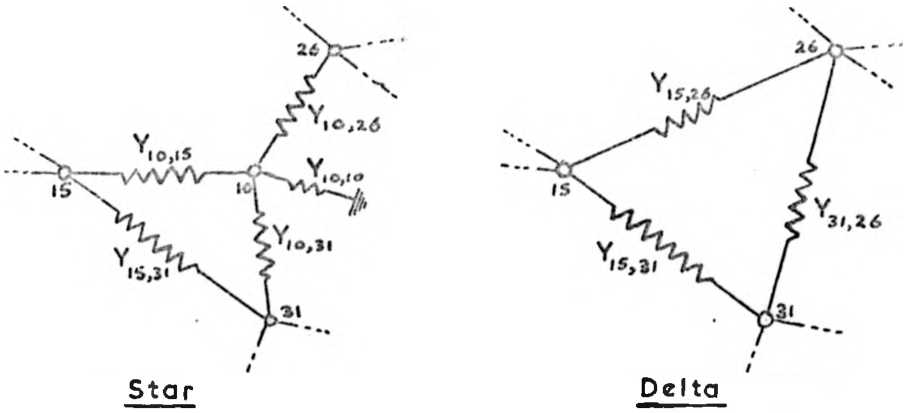


Fig. 3.4. - Star-Delta Transformation on Elimination of a Variable

The elimination of a variable from the circuit could be considered as a star-delta transformation, e.g. Fig. 3.4, where each internal variable is eliminated from the circuit in turn to leave a relationship between the external variables in the circuit.

From the flow diagram for the forward elimination, Fig. 3.3, the number of operations for the elimination of a variable connected to n_c other variables is :-

1) For the row exchanges if necessary :-

$$CALC = (n - k + 1)(AMAX_j + COMP_r) + (1 - 1/n_c)EXROW$$

where n = original size of matrix
k = variable row no. being eliminated

2) Checking for zero elements in a sparse matrix :-

$$CALC = (n - k)(1 + n_c)ZERO_j$$

3) Essential operations to eliminate variable :-

$$CALC = INV_j + n_c MULT_j + n_c^2(MULT_j + SUB_j)$$

If we consider just the essential operations then typical times are given in the following table :-

n_c	1	2	3	4	5	6	7
TIME	1592	4071	7906	13097	19644	27547	36806

Thus it is best to eliminate a variable with the least number of other variables connected to it first in the forward elimination.

Fig. 3.5 shows a typical interconnection of variables for which the best order of elimination of the internal variables would give an elimination time of 62303 whilst the worst order of elimination would give an elimination time of 182720, i.e. an increase in elimination time by about 3 times. Thus the best organisation of the elimination process would be to order the variables to be eliminated in the rows of the A matrix in equation (3.10) in order of increasing number of other variable connected to them. Thus row interchanges would only be allowed on rows with the same number of elements on the row unless a non-zero element could not be found for the pivot. It is possible whilst the A matrix is being set up to define the order in which the variables should be eliminated to give the minimum computation time. Also a map could be made of the position of the non-zero elements during the elimination but it would be difficult to include row interchanges during the elimination process to ensure the maximum accuracy in the resulting reduced matrix.

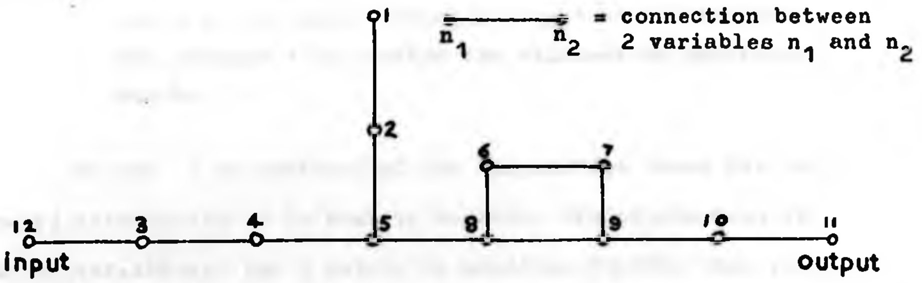


Fig. 3.5. - Typical Interconnection of Variables in a Circuit

3.3.5. Mean Forward Elimination Time for a Sparse Matrix

To estimate the time for a forward elimination on a sparse matrix it is best to consider the mean number of operations on a matrix in which there is a defined probability of a non-zero element being present in an off diagonal position in the matrix whilst assuming that all the diagonal elements are present.

Initially we can assume that there is a probability equal to p_1 of an off diagonal element being present in the A matrix in equation (3.10). During the elimination of the k th variable, using a_{kk} as the pivot, before the elimination of this variable there will be a probability equal to p_k of a element being non-zero in an off diagonal position in the A matrix in a_{ij} for $i > k$, $j > k$ and $i \neq j$. After the elimination of the k th variable these elements, a_{ij}, become :-

$$a_{ij} := a_{ij} - a_{ik}\, a_{kk}^{-1}\, a_{kj}$$

Thus the new value of p for these elements, a_{ij}, becomes :-

$$p_{k+1} = 1 - (1 - p_k)(1 - p_k^2) \quad - - - - - - - - \quad (3.13)$$

Thus, given the initial value for p_1, all the values of p_k for k = 1 to n, for n rows in the A matrix in equation (3.10) can be obtained. Alternatively :-

$$c_k = (n - k)p_k \quad - - - - - - - - - - - - - - \quad (3.14)$$

where c_k = the number of elements in off diagonal positions in row k in the upper triangular part of the A matrix for columns 1 to n after the elimination process on row k.

To obtain an estimate of the computation times for the forward elimination it is best to consider the elimination of all the variables in the A matrix in equation (3.10). This will only give a small error in the computation times and the resulting equations will be much easier to derive for the computation times for a general case. Thus the operations are as follows :-

1) Checking for zero row and cloumn multipliers :-

$$\text{CALC} = \sum_{k=1}^{n} \left\{ a_r (n-k) \text{ZERO}_j + a_c (n-k)^2 \text{ZERO}_j \right\}$$

2) For the essential operations for the elimination :-

$$\text{CALC} = \sum_{k=1}^{n} \left\{ \text{INV}_j + a_k (n-k) \text{MULT}_j + a_k b_k (n-k)^2 (\text{MULT}_j + \text{SUB}_j) \right\}$$

where a) If sparseness is not taken into account :-

$a_r = b_c = 0$ and $a_k = b_k = 1$

b) If a check for zero row multipliers is included :-

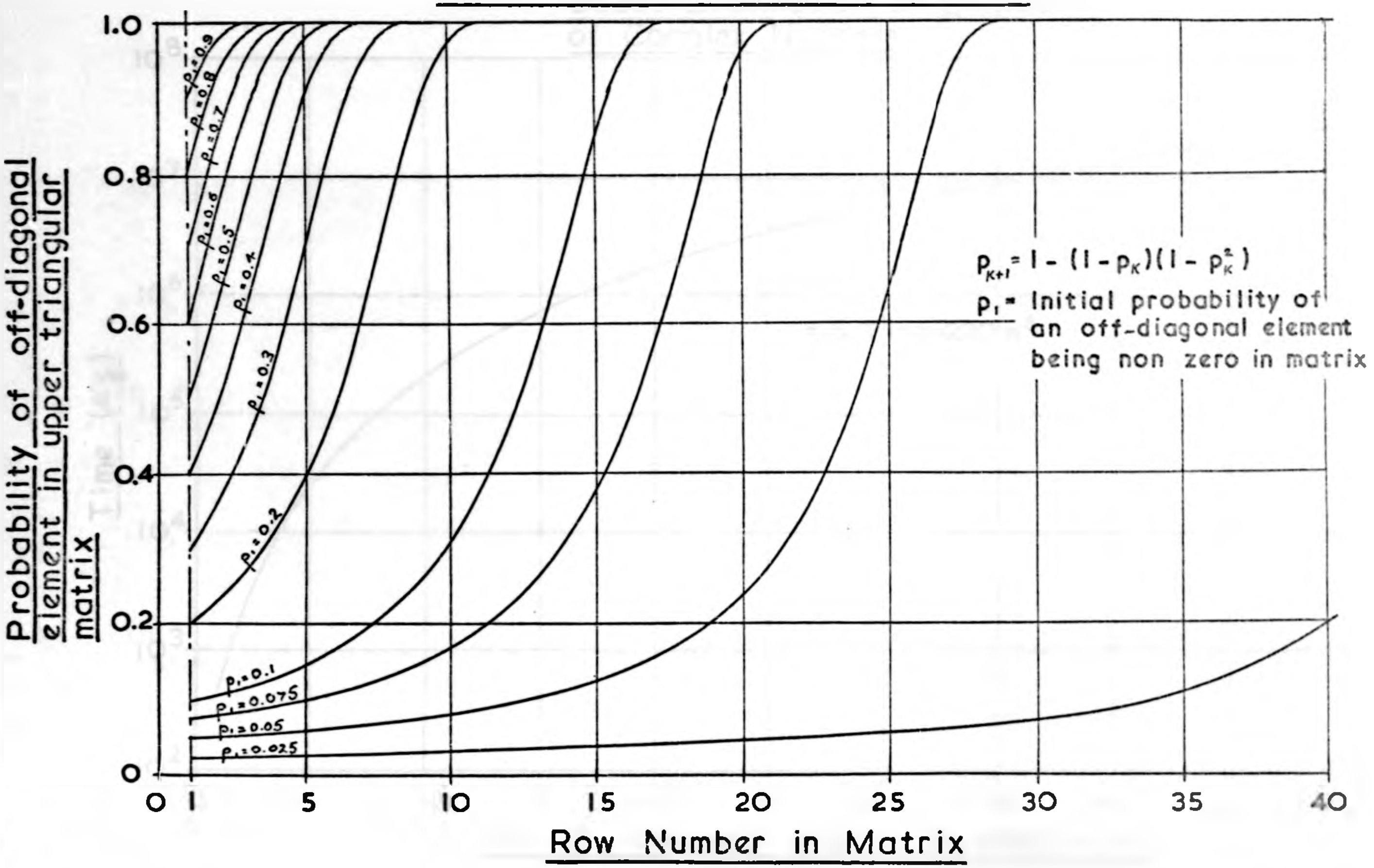
$a_r = 1$, $b_c = 0$, $a_k = p_k$ and $b_k = 1$

c) If a check for zero row and column multiplier is included :-

$a_r = b_c = 1$ and $a_k = b_k = p_k$

A simple computer program was written to derive the values of the above probabilties and computation times and the results of this is the graphs shown in Fig. 3.6 to 3.9. The first graph, Fig. 3.6, is a plot of equation (3.13) for various initial values of p_1. As one would expect the probability of an off diagonal element being present becomes greater as the elimination proceeds. Possibly towards the end of the elimination it may be better not to check for zero elements. Fig. 3.7 shows the computation time if a check for zero elements is not included. This computation time increases by the cube of the number of rows and columns in the matrix. This time, if there is only a mean of about 2 or 3 variables connected to each variable in the circuit, is out of all proportion to the actual or minimum number of operations for the reduction, section 3.3.4. The main graphs of interest are Fig. 3.8 and 3.9 giving the reduction in computation time for sparse matrix operations. From these it is obvious that it is an advantage to check for zero row and column multiplier for typical values of c of 2 to 4.

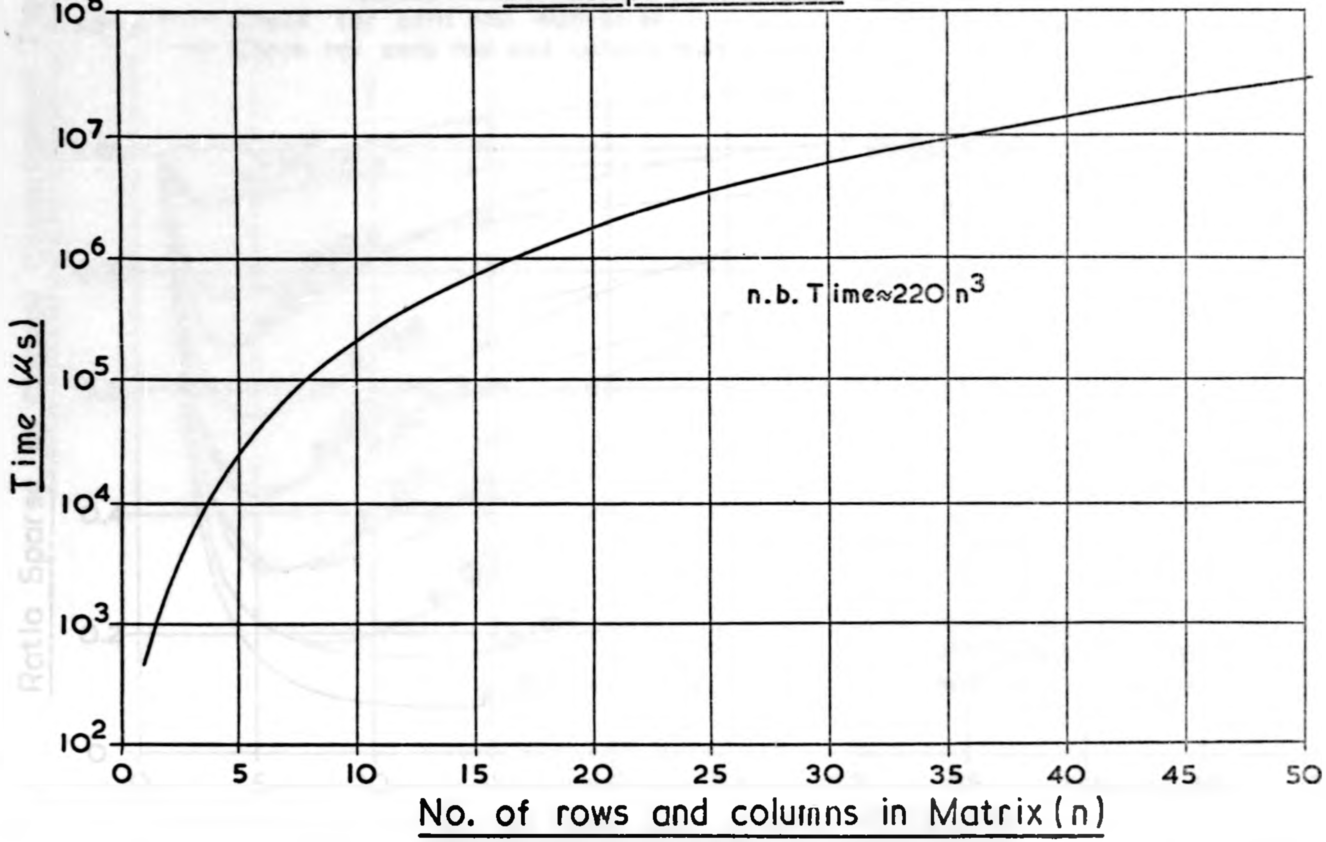
Fig. 3.6.- Variation of Matrix Sparsity During and After Forward Guass Elimination



$p_{K+1} = 1 - (1 - p_K)(1 - p_K^2)$

p_1 = Initial probability of an off-diagonal element being non zero in matrix

Fig. 3.7.- Computation Time for a Forward Guass Elimination on a Matrix of Complex Numbers



n.b. Time≈220 n^3

Fig. 3.8.- Reduction in Computation Time for Forward Guass Elimination using Sparse Techiques for an Initial Probability of an off diagonal Element being non Zero

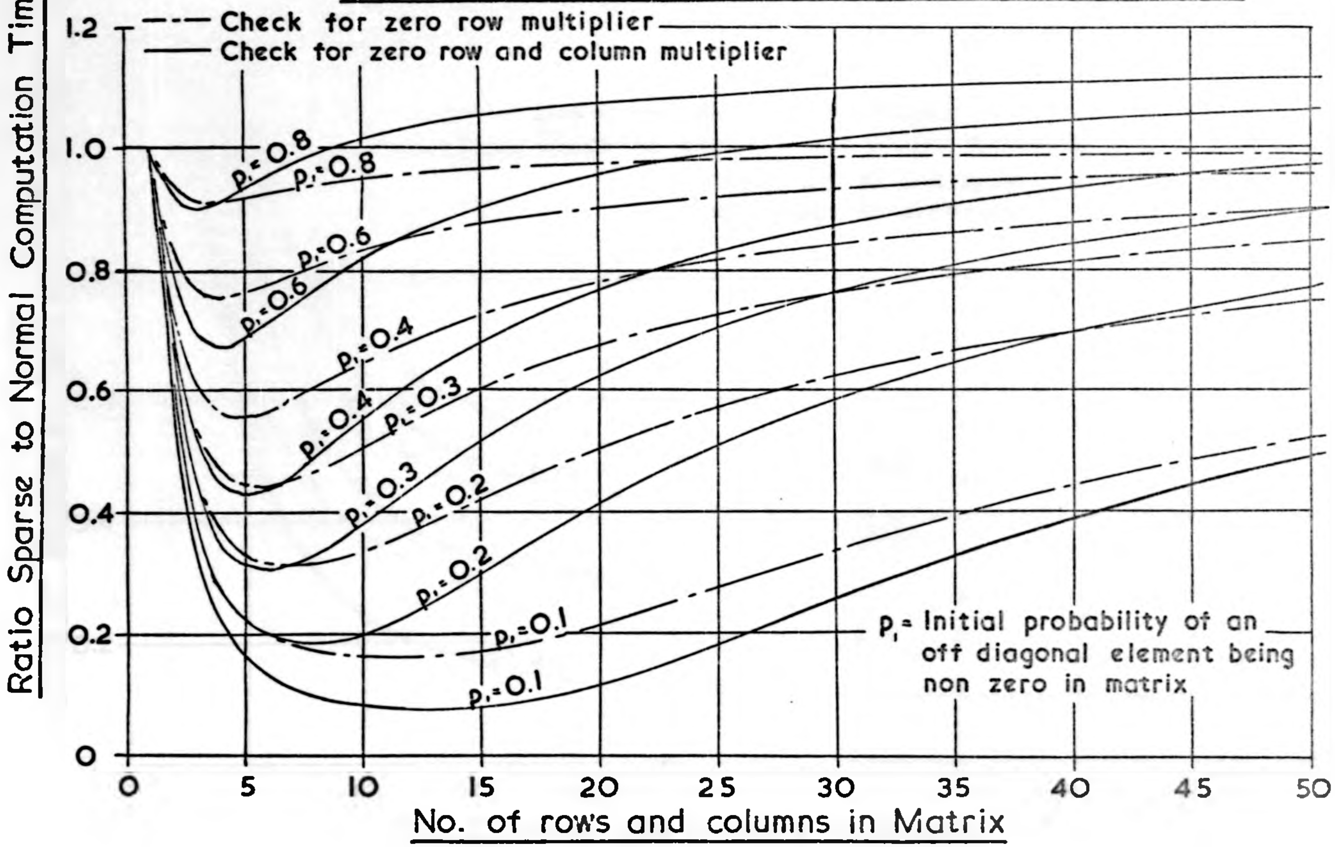
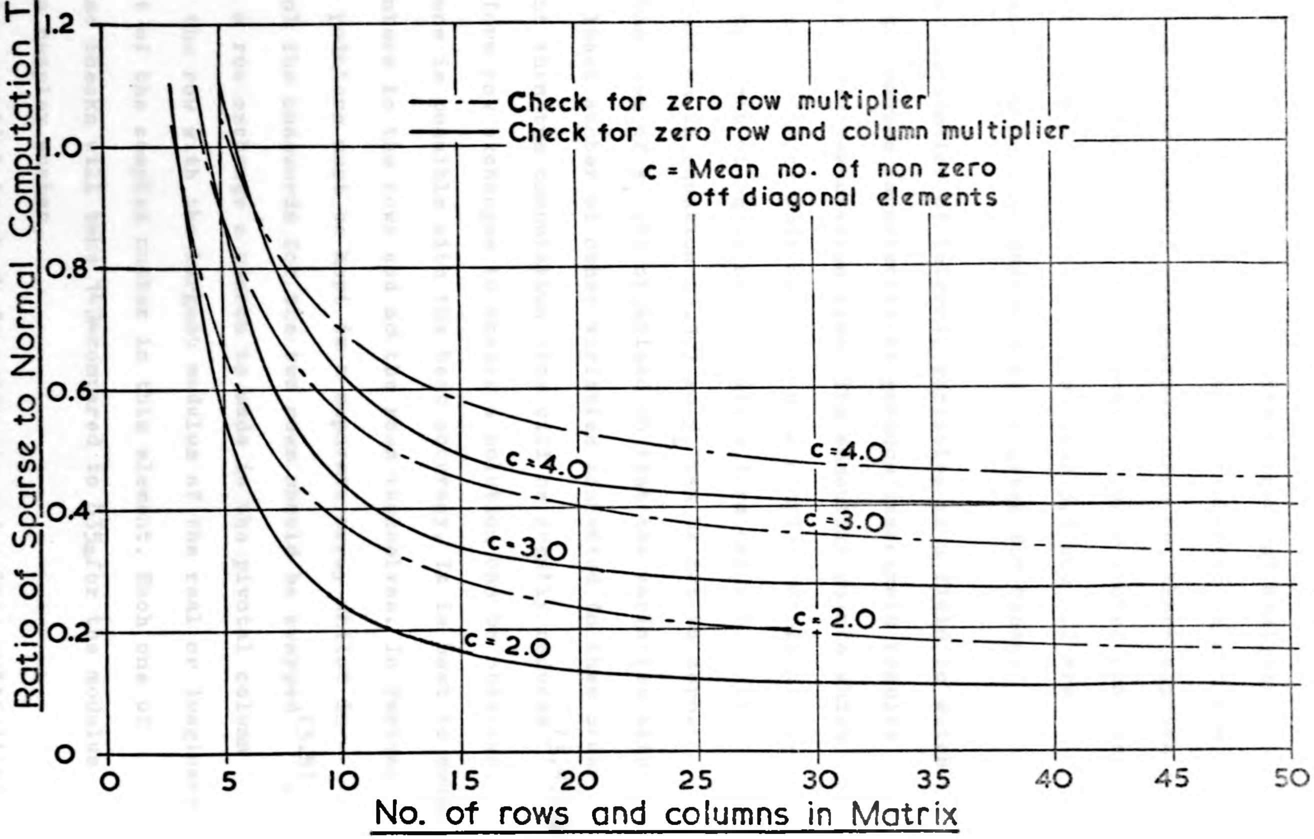


Fig. 3.9.-Reduction in Computation Time for Forward Guass Elimination using Sparse Techiques for an Initial Mean no. of off-diagonal non Zero Elements



3.4. CONCLUSIONS

The purpose of this chapter was to describe the matrix methods used in circuit analysis for the methods of analysis described in chapter 2. The contents of this chapter is not new except for the comparison of computation times but there appears to be no document describing the operations for a sparse complex number matrix in a single document. Sections 3.1 and 3.2 are fairly obvious and no conclusions will be given for these.

The elimination of internal variables is a field in which it is easy to program incorrectly to produce inaccurate results and/or a very long computation time. The essential points which must be followed for circuit analysis on a complex matrix are :-

1) Set up a set of equations in matrix form describing the circuit as in equation (3.10) so y_i can be set to zero. If the rows of y_i are organised so that the variables with the least number of other variables connected to them occur first then the computation time will be greatly reduced(3.5,3.6).
2) Perform row exchanges to ensure a solution can be obtained if one is possible with the best accuracy. It is best to swap pointers to the rows and not the rows themselves. In Fortran the pointers must be kept in a separate array whilst in Algol the codewords for the two rows should be swapped(3.8). For a row exchange a search is made in the pivotal column for the row with the largest modulus of the real or imaginary part of the complex number in this element. Each one of these checks will take 147μs compared to 835μs for the modulus of a complex number.
3) It is essential to check for zero row and column multipliers during the elimination process, see Fig. 3.9, to avoid numerous multiplications by zero for a sparse matrix.

References

3.1) J.K. Skwirzynski, 'Linear and Nonlinear Programming', IEE Vacation School on Circuit Theory, Lecture 20, 5 - 10 July 1971

3.2) L. Fox, 'An Introduction to Numerical Linear Algebra', Clarendon Press Oxford, 1964

3.3) J.H. Wilkinson, 'The Algebraic Eigenvalue Problem', Clarendon Press Oxford, 1965

3.4) as reference 3.1

3.5) M. Silverberg, 'Near Optimal Ordering of Electronic Circuit Equations', IEEE trans. on Computing, Vol. C-17, No.12, Dec. 1968, pp. 1173-1174

3.6) R.D. Berry, 'An Optimal Ordering of Electronic Circuit Equations for a Sparse Matrix Solution', IEEE trans. on Circuit Theory, Vol. CT-18, No. 1 , Jan 1971, pp. 40-50

3.7) J.K. Reid, 'Large Sparse Sets of Linear Equations', Academic Press, 1971

3.8) 4100 Technical Manual, ICL, Vol. 2 - Programming Information, Part 2 - NEAT and NICE, Section 6 - NICE Supplementary Routines, Chapter 8 - SPR (Storage Plan Routines)

Chapter 4

Path Topological Analysis for Chain Matrix Analysis Program

4.1. INTRODUCTION

In all chain matrix analysis programs previously written, section 1.4, it has always been necessary to input the data in a very precise order to define the circuit topology. This type of data is not suitable for the more advanced, possibly interactive, use of the program. For this type of application it is necessary to be able to add or delete any part of the circuit from the circuit description during a computer run on the program.

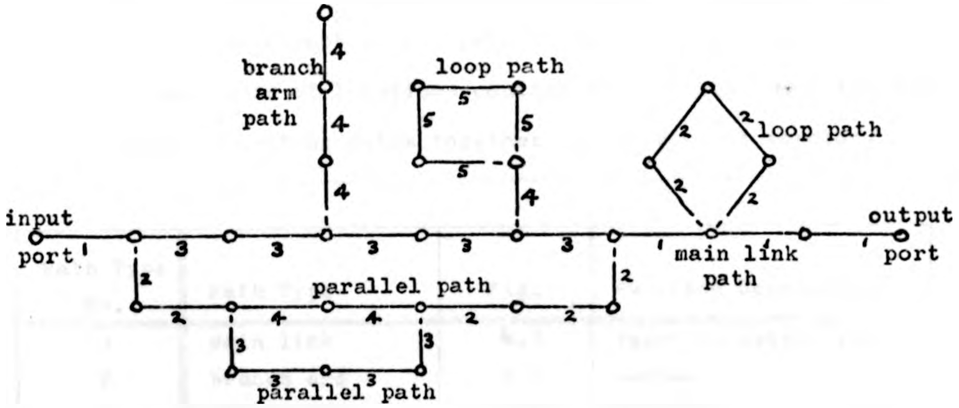
The objective in the CHAIN1 program (written by the author during the course of this thesis) was to make it possible to describe the circuit topology in a simple way in the data so that it could later be modified by additions or deletions to any part of the data. In the CHAIN1 program the data consist of a list of 2-port networks, given in any order, with the junctions to which the networks are connected. Two junctions are assigned, in the data, as the input and output ports of the circuit.

The CHAIN1 program will, just before an analysis, perform a topological analysis on this circuit description to form paths of 2-port networks in cascade in the circuit e.g. Fig. 4.1. These paths will form the main link between the input and output ports of the circuit on top of which other paths may be built to any level. This is necessary so that the chain matrix method of analysis, section 2.2, can be used for the analysis of the circuit. The possible path types are as follows :-

1) main link path between input and output ports of the circuit, Fig. 2.4, which must always be present.
2) branch arms, Table 2.2.
3) parallel paths, Table 2.3, considered in pairs.
4) loop paths, Table 2.5.

In some cases it may not be possible to break up the entire circuit into paths of these types in which case the circuit can not be analysed entirely using chain matrix analysis as described in section 2.2.

In this chapter the method of storage of the path topology, after the topological analysis, is described in section 4.2 whilst the formation of the path topology from the initial data is described in section 4.3. The specific application of the path topological analysis in this chapter is for the analysis of microwave circuits but the same analysis could be used in other fields, e.g. the analysis of water flow in a complex network of pipes, etc..



N.B. o = junction connection
o——o = 2-port network connecting 2 junctions
—-o = connection of a sub-path to its path of origin
numbers on networks = path levels (section 4.2.4)

Fig. 4.1. - Typical Path Topology

4.2. PATH TOPOLOGY STRUCTURE

4.2.1. Single Path

In the CHAIN1 program the path topology is stored entirely in 2 integer arrays A and B (n.b. these arrays as a pair may be refered to as the STORE arrays in parts of this chapter). These arrays are considered in parallel, Fig. 4.2, as we are interested in pairs of elements in the two arrays with the same subscript address. A number of consecutive pairs of locations in these arrays, termed a bead, may form a path description. The body of this path description contains a list of the 2-port networks in order as they occur in this path. In between networks in this list other paths may radiate out from this path and to accommodate these a pointer may be inserted in the path description to point to the description of this path in the path topology. In the path bead the first and last pair of locations contain information about the path and pointers connecting paths together in the circuit topology.

Path Type No.	Path Type	Fig.	Junction Connections
1	main link	4.3	input to output ports
2	branch arm	4.4	series
3	branch arm	4.4	parallel
4	loop path	4.5	series
5	loop path	4.5	parallel
6	parallel path	4.6	series to series
7	parallel path	4.6	series to parallel
8	parallel path	4.6	parallel to series
9	parallel path	4.6	parallel to parallel

Table 4.1. - Path Types

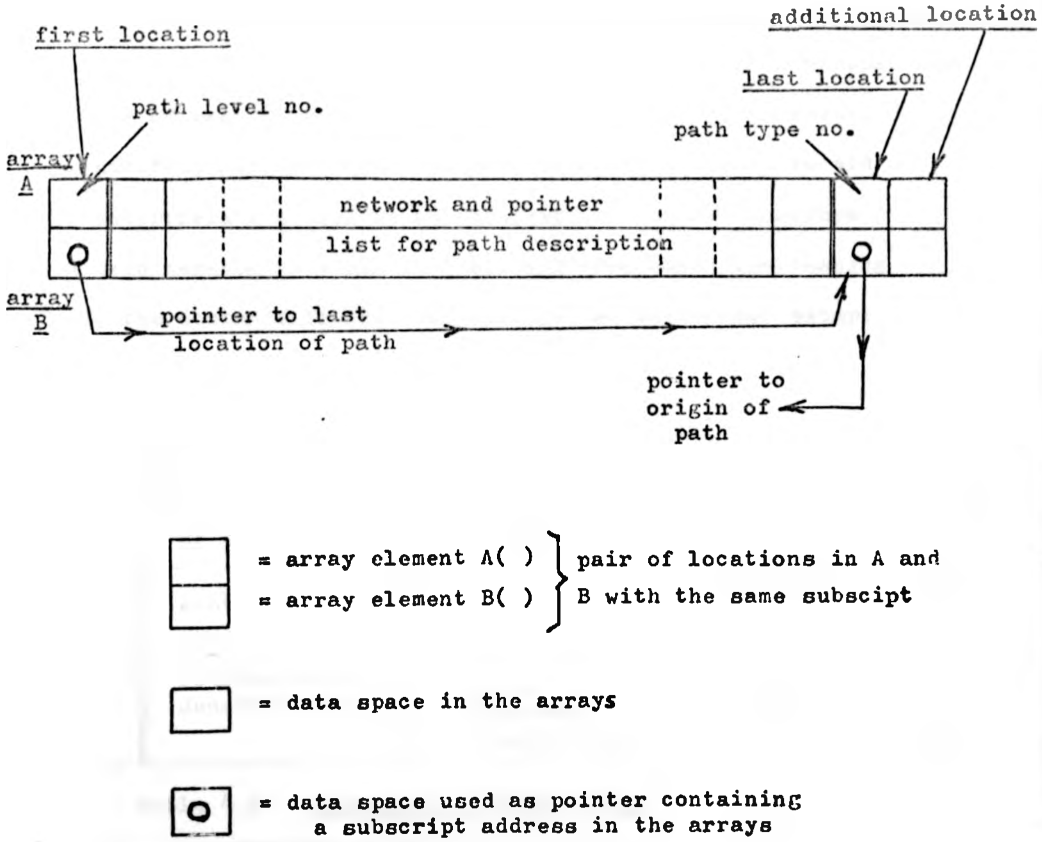

first location
additional location
last location
path level no.
path type no.
array A
array B
network and pointer
list for path description
pointer to last location of path
pointer to origin of path
= array element A( )
= array element B( )
pair of locations in A and B with the same subscipt
= data space in the arrays
= data space used as pointer containing a subscript address in the arrays

Fig. 4.2. - <u>Single Path Bead for a Path in the Data Structure in the Path Topology of a Circuit</u>

4.2.2. Path Types

All the possible path types are given in Table 4.1 with diagrams of the bead structure for these paths given in Fig. 4.3 to 4.6. In the use of the path topology, either during its formation or during the circuit analysis, it is necessary to know to what the contents of each pair of locations in the STORE arrays refer, e.g. pointer, network, start of path description etc.. To aid this interpretation some of the contents of the STORE arrays have been negated as shown in Table 4.2 from which the meaning of all the contents in the STORE arrays can be decided except the additional location on the end of a path description.

	$A(i) < 0$	$A(i)=0$	$A(i) > 0$
$B(i) < 0$	First Location of path	Network Number	Last Location of path
$B(i)=0$	empty	empty	
$B(i) > 0$	Pointer to Junction Number	Network Number (reverse connection)	Pointer to Path on Top of this path

Table 4.2. - Contents of STORE arrays

4.2.3. Use of Path Topology During Analysis

In section 2.2 the equations for the chain matrix analysis of a cascade of 2-port networks including branch arms, loops and parallel paths were given . This section contains a short description of how the results of the topological analysis can be used to assist in the organisation of the chain matrix analysis. The path levels, e.g. Fig. 4.1, are concerned with the analysis part of the program and indicate at any one time during the analysis

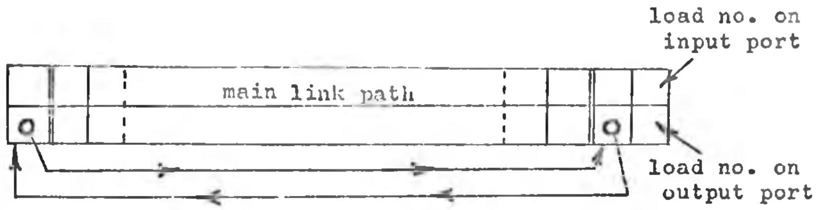


Fig. 4.3. - Main Link Path Bead Structure

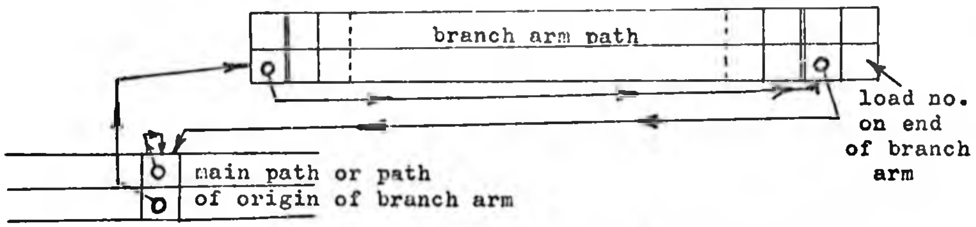


Fig. 4.4. - Branch Arm Path Bead Structure

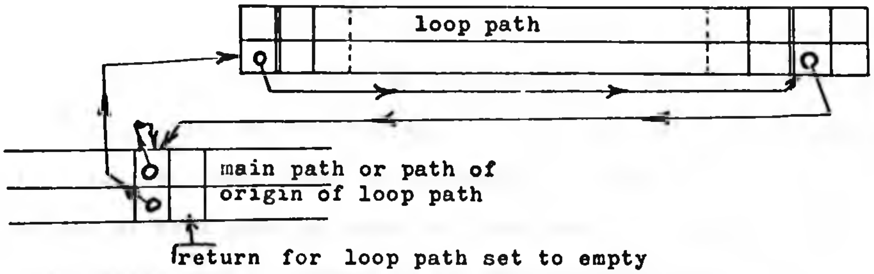


Fig. 4.5. - Loop Path Bead Structure

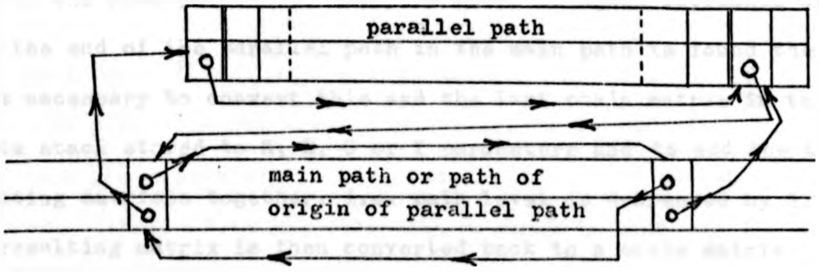


Fig. 4.6. - Parallel Path Bead Structure

the number of matrices held in a stack as intermediate results.

If we consider first the chain matrix analysis of the main link path then this can be analysed as a simple cascade of 2-port networks with path level of 1, i.e. only one chain matrix is required for the storage of the intermediate results. The chain matrix analysis will proceeed down the main link from the input to the output port of the circuit. If a branch arm or loop path is detected on this main link path, or on any other path, then the chain matrix for this path must be stored in a stack. Then the chain matrix for the branch arm or loop is formed in the same way as for the main link path. When the chain matrix for the branch arm or loop has been formed its equivalent circuit can be included in the main path as a simple 2-port network. During this process on the branch arm or loop path it is necessary to increase the path level by 1 on this path for the chain matrix which has to be temporarily stored in the matrix stack. If a parallel path is detected on the main link, or on any other path, then the chain matrix for the main path must again be stored in the matrix stack so that the analysis can proceed down the parallel path, i.e. the path level is increased by 1 for this path. When the end of this parallel path is found then the parallel path chain matrix must be stored in the matrix stack whilst the chain matrix analysis proceeds along the other side of the parallel path in the main path, i.e. the path level is again increased by 1. When the end of the parallel path in the main path is found then it is necessary to convert this and the last chain matrix in the matrix stack stored to Z, H, G or Y parameters and to add the two resulting matrices together, i.e. path level is decreased by 1. The resulting matrix is then converted back to a chain matrix for inclusion in the main path as a 2-port network, i.e. the path level again decreased by 1. Thus the analysis of the circuit will proceed storing intermediate matrices in a stack as required to finally give a single chain matrix describing the performance of the circuit between the input and output ports of the circuit. During the path topological analysis the maximum path level found is recorded and this is used to set up the size of the stack to hold the chain matrices during the circuit analysis.

4.2.4. Path Nesting

In the original data for the circuit any number of networks or ports could be connected to any junction in the circuit. Thus, for the analysis, it is necessary to nest, in the path topology, the paths radiating out from a junction correctly. This problem makes the path topological analysis more complex than would be expected in practice but it is necessary for a very general application of the topological analysis.

The paths radiating out from the main path on a junction are nested, Fig. 4.7, so that the pointers on each junction in the path topology will occur in the following order :-

1) Return pointers from parallel paths
2) Pointers to the start of parallel paths
3) Pointers to branch arms and loop paths

In both 1) and 2) the paths must be nested correctly for the circuit analysis. In the case of the return pointers from parallel paths a parallel path with a high origin address in the main path must occur first. In the case of the pointers to the start of parallel paths a path with a high return address in the main path must occur first. It is possible that a parallel path may not start and finish on the same main path. In this case the pointer to this parallel path is placed just before the pointer to the path on which this path returns. If more than one parallel path ends on the same path from this junction, even though it may not be the path of origin of these parallel paths, they must still be nested correctly. In theory these parallel paths should start and finish on the same path but this is far more difficult to organise in the path topological analysis.

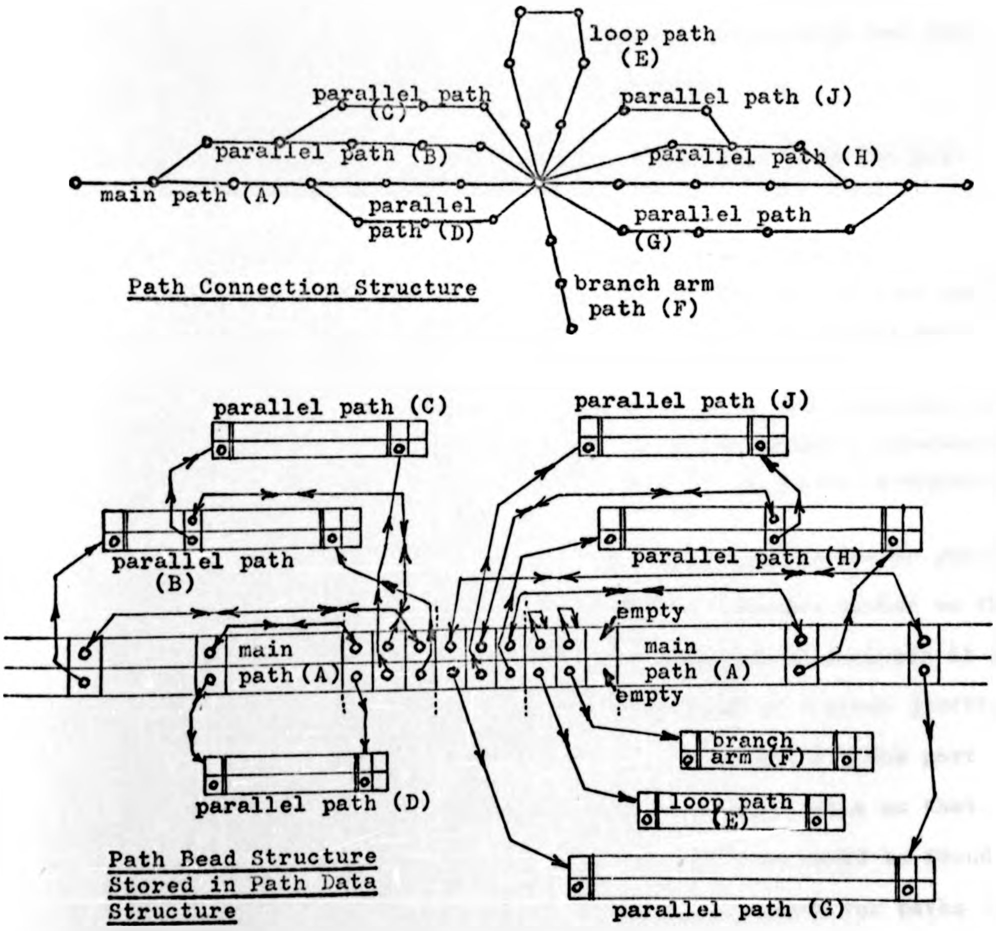


Fig. 4.7. - Complex Path Nesting Problem

4.3. PATH TOPOLOGICAL ANALYSIS

4.3.1. Objective

The objective of the path topological analysis is to break a circuit consisting of an assembly of 2-port networks into a number of paths as defined in section 4.2 so a chain matrix analysis can be used to analyse the circuit. The topological analysis uses a trial and error search procedure first to find the main link path in the circuit between the input and output ports. Then, on top of this path, parallel paths, branch arms and loop paths are searched for by a similar process.

The data from which the path search must form the path topology is as follows :-

1) A list of the 2-port networks in the circuit
2) The two junction connections on each end of each network
3) The junctions assigned as the input and output ports of the circuit
4) A list of loads connected to some of the junctions in the circuit for their use as characteristic impedances for the input and output ports or as loads terminating branch arms
5) A list of all the junction types i.e. series or parallel.

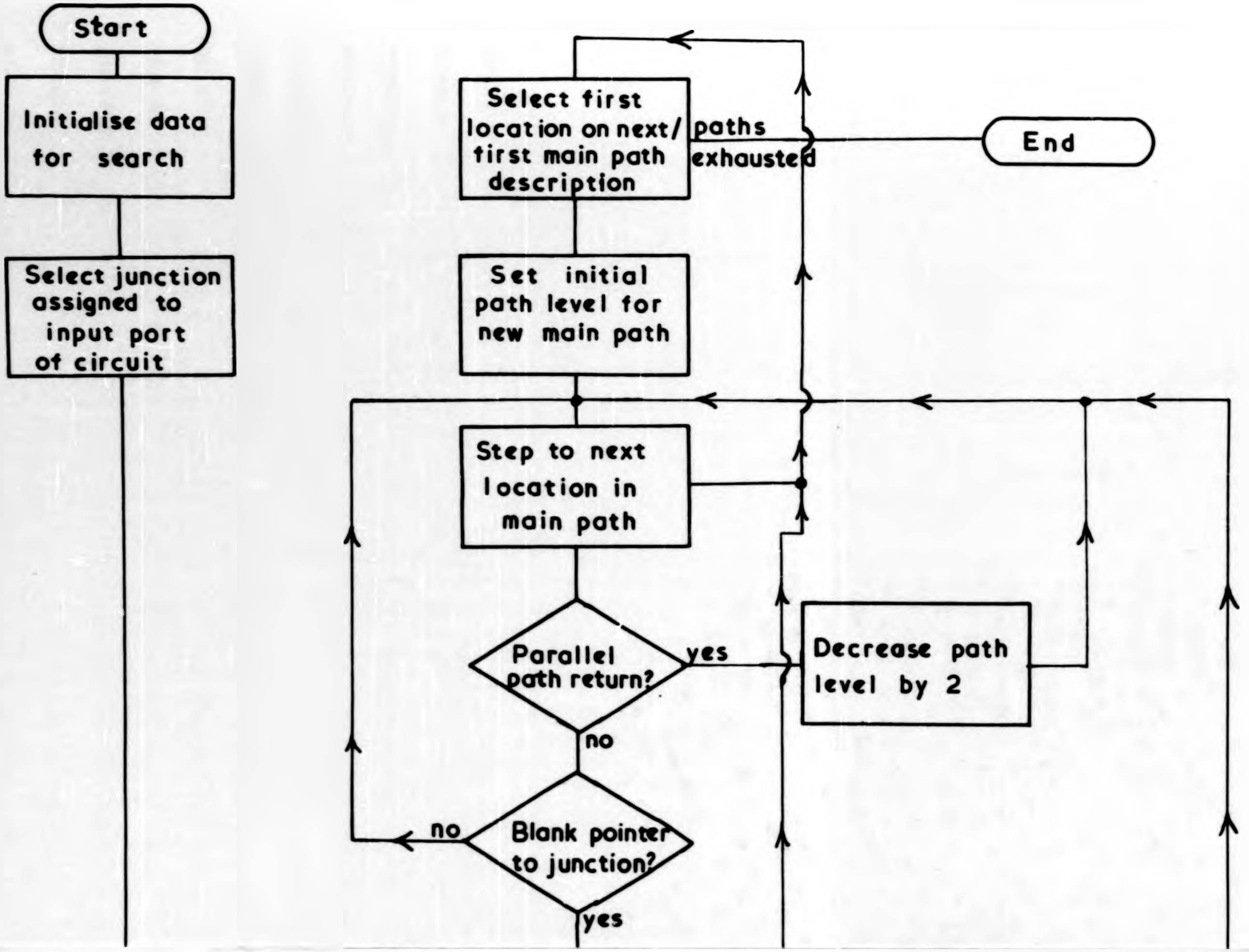
From this data it is very easy to find the junction number on the ends of a given network but during the topological analysis it is also necessary to find the networks connected to a given junction. To do this an open hash table(4.1) was set up and all the port and network junction connections entered in this table so that the networks and ports connected to any junction could be found quickly and simply. During a trial and error search for paths in the circuit it is necessary to keep a record of the paths which have already been tried or stored as a path. To do this two lists were set up: one for all the junctions in the circuit and one for all the networks in the circuit. In these two lists the following values were used :-

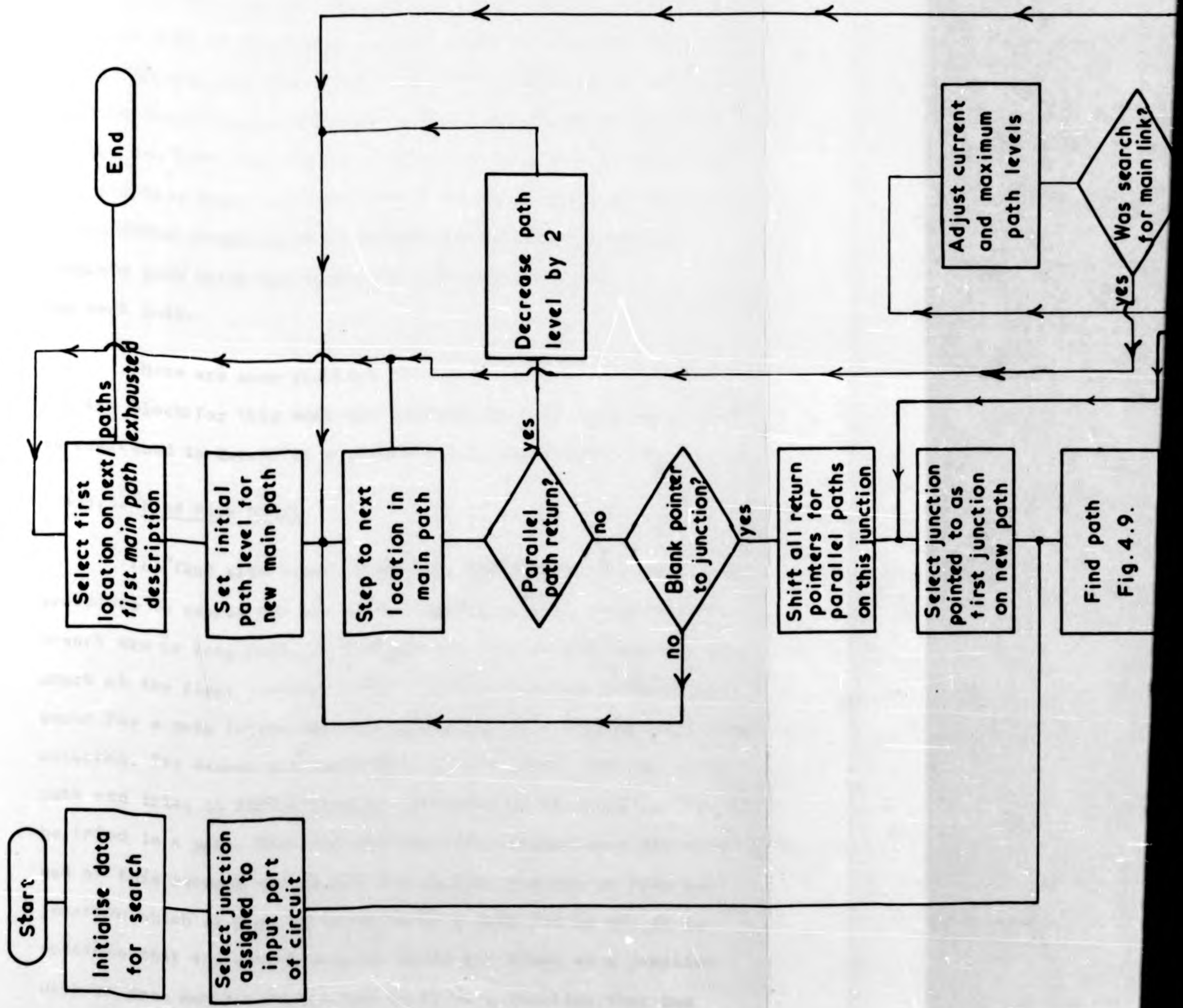
1) +ve = junction/network has been tried in a path
2) -ve = junction/networks has been stored as part of a path
3) zero = junction/network is free to be tried in a path

4.3.2. Path Search Organisation

The organisation of the 'path search' procedure is shown in Fig. 4.8. First the data structure is initialised and the junction assigned as the input port is taken as the first junction on the start of the main link path in the circuit. The procedure then enters the 'find path' block with the objective of finding a main link path between the input and output ports of the circuit. If there is a main link in the circuit somewhere then the find path will always find it and at the same time store the sequence of networks and pointers to other paths in the STORE arrays as described in section 4.2.1. If there is no main link present then the 'find path' block will detect a branch arm instead but this will be rejected as an error in the 'complete path' block. After the main path has been found and formed in the STORE arrays then the 'complete path' block will set the contents of the first and last pair of locations in the path description in the STORE arrays. Then the data structure is reset so that further paths can be searched for in the circuit and the maximum path level adjusted for this new path.

The path building process continues by selecting the first location in the main link path description in the STORE arrays or, later, the next path description in the STORE arrays. Then the current path level is set to the path level for this path and a step is made down each location in the description of this path


Start
Initialise data for search
Select junction assigned to input port of circuit
Select first location on next/ first main path description
paths exhausted
End
Set initial path level for new main path
Step to next location in main path
Parallel path return?
yes
no
Decrease path level by 2
Blank pointer to junction?
no
yes


Start
Initialise data for search
Select junction assigned to input port of circuit
Select first location on next/first main path description
paths exhausted
End
Set initial path level for new main path
Step to next location in main path
Parallel path return?
yes
no
Decrease path level by 2
Blank pointer to junction?
yes
no
Shift all return pointers for parallel paths on this junction
Select junction pointed to as first junction on new path
Find path Fig.4.9.
Adjust current and maximum path levels
Was search for main link?
yes

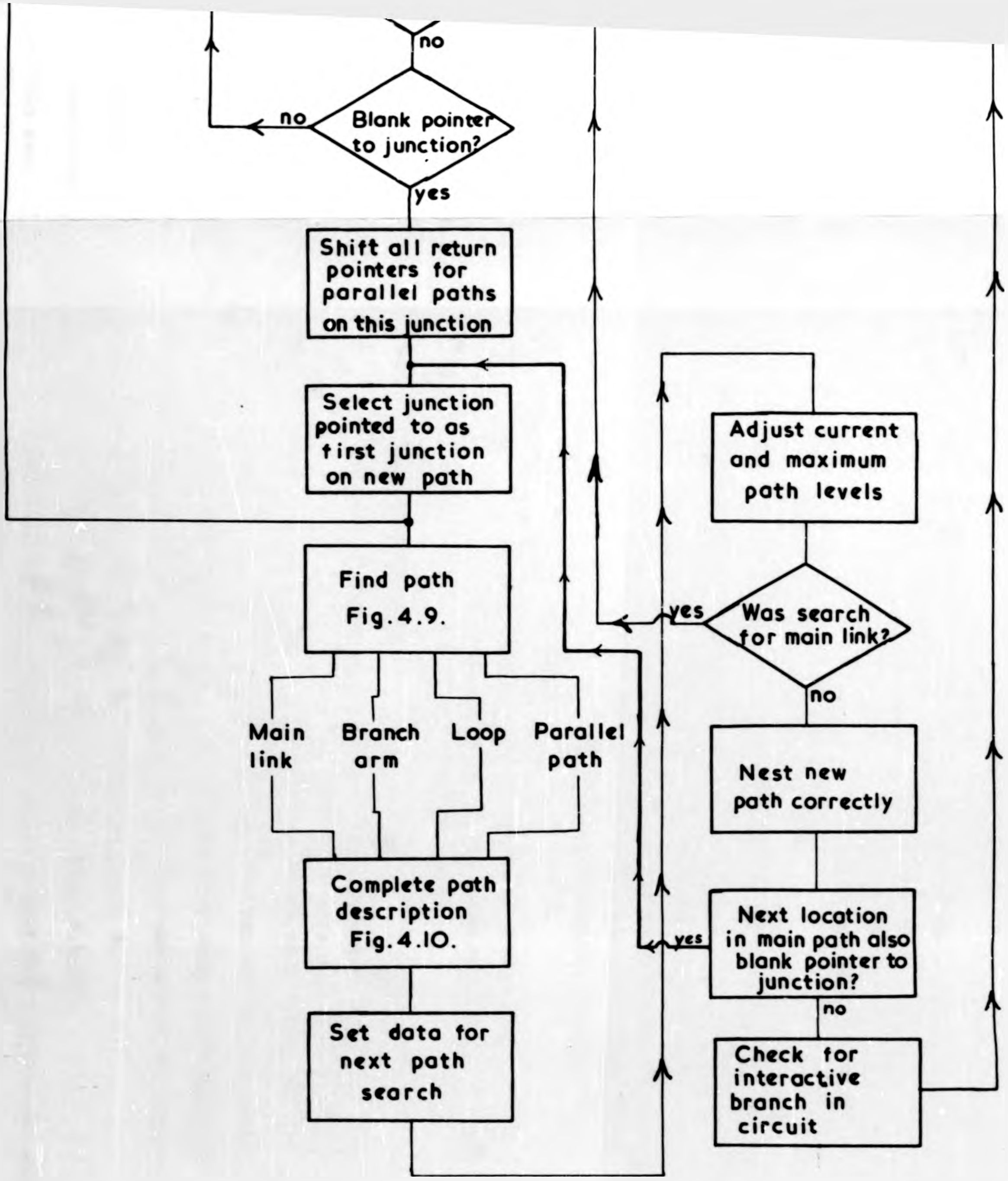

no
Blank pointer to junction?
no
yes
Shift all return pointers for parallel paths on this junction
Select junction pointed to as first junction on new path
Find path Fig.4.9.
Main link
Branch arm
Loop
Parallel path
Complete path description Fig.4.10.
Set data for next path search
Adjust current and maximum path levels
Was search for main link?
yes
no
Nest new path correctly
Next location in main path also blank pointer to junction?
yes
no
Check for interactive branch in circuit

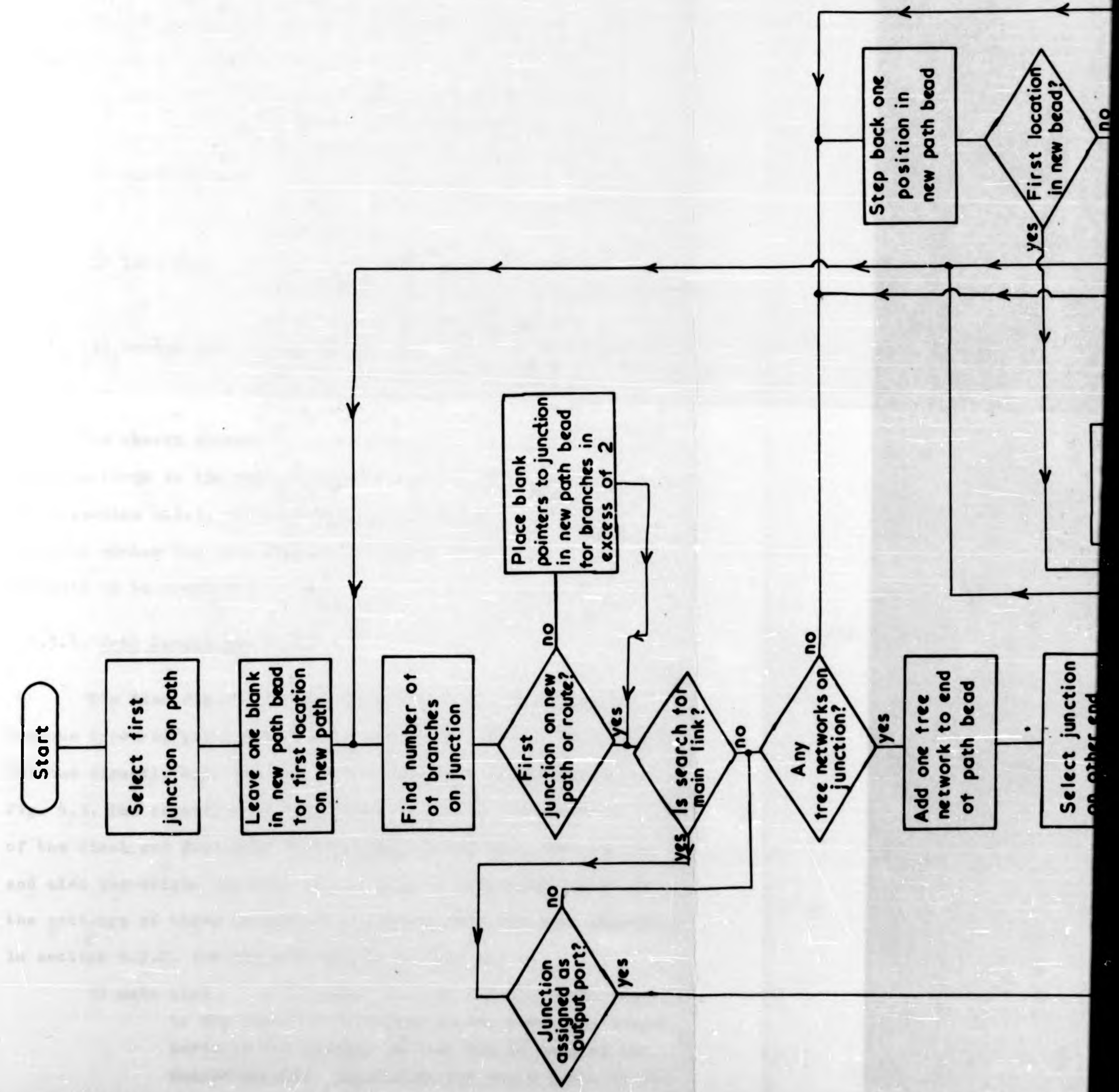
Fig.4.8.- Flow Diagram for Path Search in CHAINI Program

in the STORE arrays. If a pointer for the return of a parallel path is found then the current path level is decreased by 2 whilst if a blank pointer for the start of a path radiating out from this path is found then a search will be made for this path in the circuit. The search for such a path consists of using the junction connection in this blank pointer as the first junction on this new path. The 'find path' block is then used to find this path and this block will also store the description of this path in the STORE arrays as it is formed and complete it in the 'complete path' block and reset the data structure for the next path.

There are some problems with path nesting and checking and the blocks for this work are included in Fig. 4.8. These will be described in detail in sections 4.3.5. and 4.3.6.

4.3.3. Find Path Block

The 'find path' block, Fig. 4.9, was used in the path search procedure to search for any path, i.e. main link, parallel path, branch arm or loop path, in the circuit. The method used was to start at the first junction on the path and search by trial and error for a path in the circuit consisting of a cascade of 2-port networks. The search process starts at the first junction on the path and tries to find a network connected to it which is free to be tried in a path. Then the process will advance onto the other end of this network and search for another network on this new junction which is free to be tried in a path and so on. It is possible that the search process could get stuck at a junction with no free networks connected to it or a junction that has already been tried in a path. In this case the process reverses back down the path it has just formed looking for a junction which still has a free network leading out of it. If one is found then


Start
Select first junction on path
Leave one blank in new path bead for first location on new path
Find number of of branches on junction
First junction on new path or route?
yes
no
Place blank pointers to junction in new path bead for branches in excess of 2
Is search for main link?
yes
no
Junction assigned as output port?
yes
no
Any tree networks on junction?
yes
no
Add one tree network to end of path bead
Select junction on other end
Step back one position in new path bead
First location in new bead?
yes
no

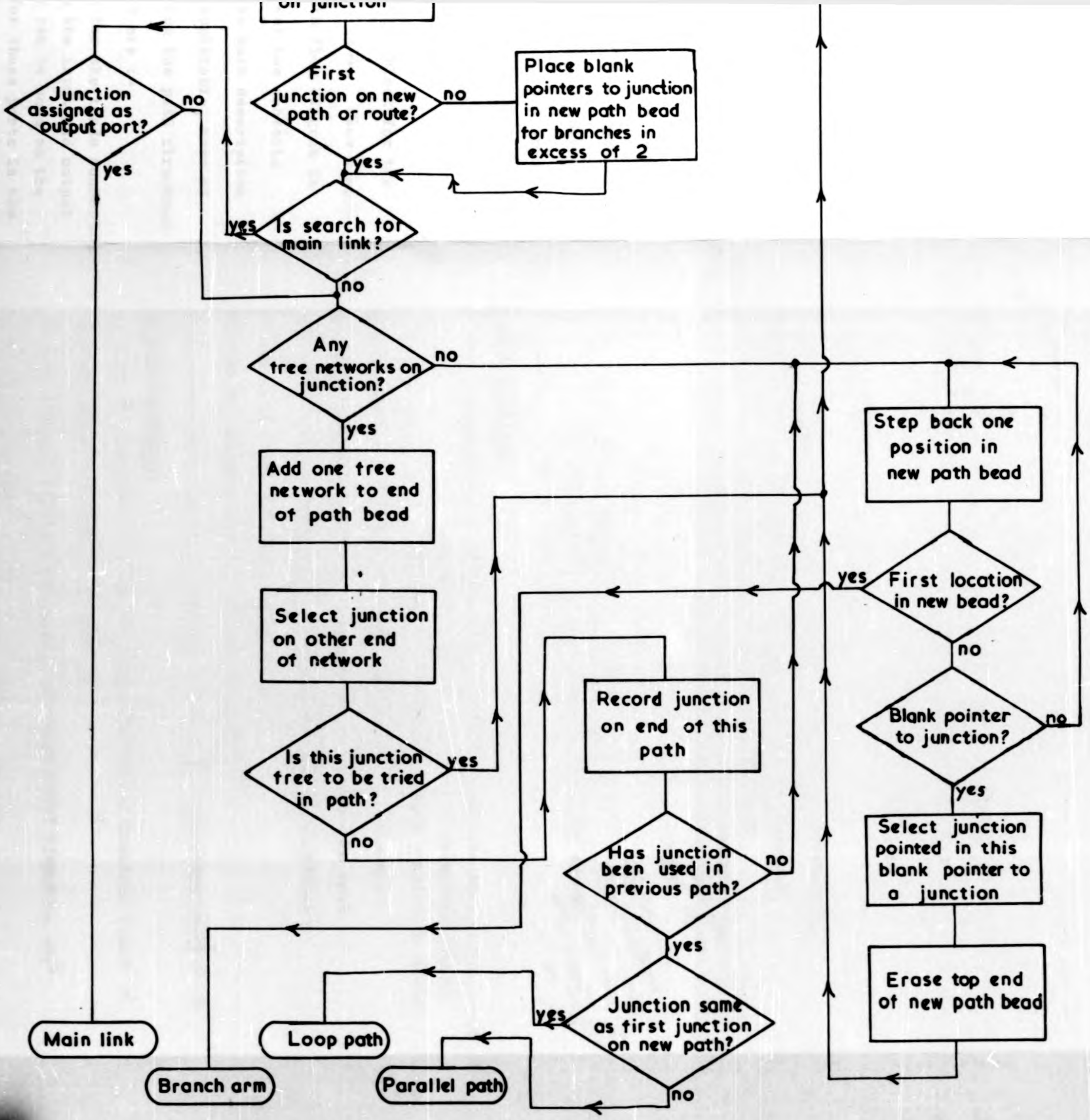

on junction
First junction on new path or route?
no
yes
Place blank pointers to junction in new path bead for branches in excess of 2
Junction assigned as output port?
no
yes
Is search for main link?
yes
no
Any tree networks on junction?
no
yes
Add one tree network to end of path bead
Select junction on other end of network
Is this junction free to be tried in path?
yes
no
Step back one position in new path bead
First location in new bead?
yes
no
Blank pointer to junction?
no
yes
Select junction pointed in this blank pointer to a junction
Erase top end of new path bead
Record junction on end of this path
Has junction been used in previous path?
no
yes
Junction same as first junction on new path?
yes
no
Main link
Branch arm
Loop path
Parallel path

Fig.4.9.- Flow Diagram for Find Path in CHAINI Program

the top part of the path above this junction is deleted and a new path is formed using this new free network. This search process terminates when a suitable path has been found as follows :-

1) main link - when the path reaches the junction assigned as the output port (n.b. on the search for the main link only).
2) parallel path - when the search reaches a junction which has previously been stored as part of a path.
3) loop path - same as a parallel path except that the path returns to the same junction number as the origin point of the path.
4) branch arm - when the process has tried all paths but has not been able to form any of them into a parallel path or loop path.

During the search process it is necessary to record the list of networks found in the path description in the STORE arrays, as in section 4.2.1, with blank spaces containing the current junction number for each uncompleted path radiating out from this new path to be completed later.

4.3.4. Path Completion Block

The flow diagram for the 'path completion' block for the various types of paths is shown in Fig. 4.10. This flow diagram follows directly onto the end of the find path flow diagram in Fig. 4.9. The objective of this block is to set the contents of the first and last pair of locations in the path description and also the origin location of the path if applicable. Most of the settings of these locations is obvious from the path structure in section 4.2.2. but the main points to note are :-

1) main link - A search is made for the loads connected to the junctions assigned as the input and output ports of the circuit so they can be used as the characteristic impedances for these ports in the

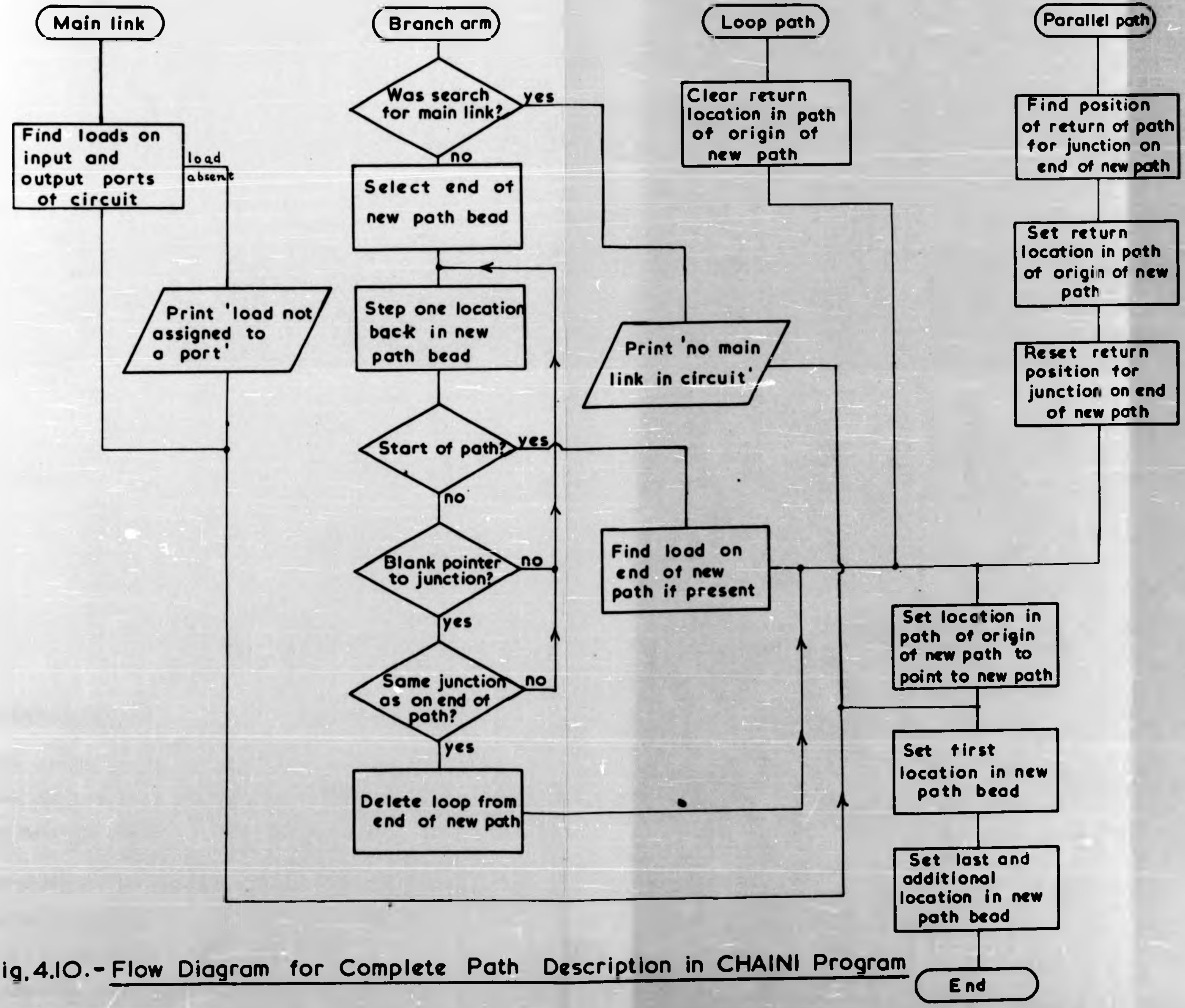

Main link
Find loads on input and output ports of circuit
load absent
Print 'load not assigned to a port'
Branch arm
Was search for main link?
yes
no
Select end of new path bead
Step one location back in new path bead
Start of path?
yes
no
Blank pointer to junction?
no
yes
Same junction as on end of path?
no
yes
Delete loop from end of new path
Print 'no main link in circuit'
Find load on end of new path if present
Loop path
Clear return location in path of origin of new path
Parallel path
Find position of return of path for junction on end of new path
Set return location in path of origin of new path
Reset return position for junction on end of new path
Set location in path of origin of new path to point to new path
Set first location in new path bead
Set last and additional location in new path bead
End

Fig. 4.10.- Flow Diagram for Complete Path Description in CHAINI Program

circuit analysis. If either of these loads is missing then an error is printed out.

2) branch arm - If the search was for the main link then an error is printed out. Otherwise it is possible that the path is not a true branch arm but a branch arm with a loop on its end. To check for this a search is made to see if the end junction on this path is the same junction as any other junction in the description of this path. If it is then there is a loop on the end of this path and the loop is deleted from the end of this path adding an additional blank pointer to the end of this path for this loop path. Finally a search is made to see if there is a load connected onto the end of this branch arm.

3) loop path - The location in the STORE arrays on the end of the block of blank pointers on the origin of this path is cleared to delete the blank pointer for the return of this loop path.

4) parallel path - For this type of path it is also necessary to know the return address for the parallel path in the path topology. During the search process for previous paths a list of the highest address in the path description for each junction as a blank pointer to a path is recorded. This is used to give the return address of a possible parallel path detected later.

4.3.5. Path Nesting

In section 4.2.4. the problems of path nesting were introduced and in this section methods used to ensure that the paths are nested correctly are described.

During the path nesting process it is necessary to swap the pointers to various path descriptions to nest them correctly. To do this the following procedure was used :-

```
procedure SWAP(e,f); value e,f; integer e,f;
```

circuit analysis. If either of these loads is missing then an error is printed out.

2) branch arm - If the search was for the main link then an error is printed out. Otherwise it is possible that the path is not a true branch arm but a branch arm with a loop on its end. To check for this a search is made to see if the end junction on this path is the same junction as any other junction in the description of this path. If it is then there is a loop on the end of this path and the loop is deleted from the end of this path adding an additional blank pointer to the end of this path for this loop path. Finally a search is made to see if there is a load connected onto the end of this branch arm.

3) loop path - The location in the STORE arrays on the end of the block of blank pointers on the origin of this path is cleared to delete the blank pointer for the return of this loop path.

4) parallel path - For this type of path it is also necessary to know the return address for the parallel path in the path topology. During the search process for previous paths a list of the highest address in the path description for each junction as a blank pointer to a path is recorded. This is used to give the return address of a possible parallel path detected later.

4.3.5. Path Nesting

In section 4.2.4. the problems of path nesting were introduced and in this section methods used to ensure that the paths are nested correctly are described.

During the path nesting process it is necessary to swap the pointers to various path descriptions to nest them correctly. To do this the following procedure was used :-

```
procedure SWAP(e,f); value e,f; integer e,f;
```

where e and f are the addressess of the pointers to the two paths to be swapped. These may be pointers to the start of a path description, return pointers for a parallel path or blank pointers. In all cases the pointers will be swapped and any other pointers associated with them reset for the new position of these pointers.

If there are a number of blank pointers in a path description then the return pointers for previous parallel paths will be set to the top end of this block. In this way the parallel paths starting at the lowest addressess will be formed first and thus the parallel path return pointers will be nested correctly, e.g. Fig. 4.11. Then on entering this block in the path building process all the parallel path return pointers will be first shifted down to the bottom of this block of pointers decreasing the path level by 2 for each parallel path return pointer.

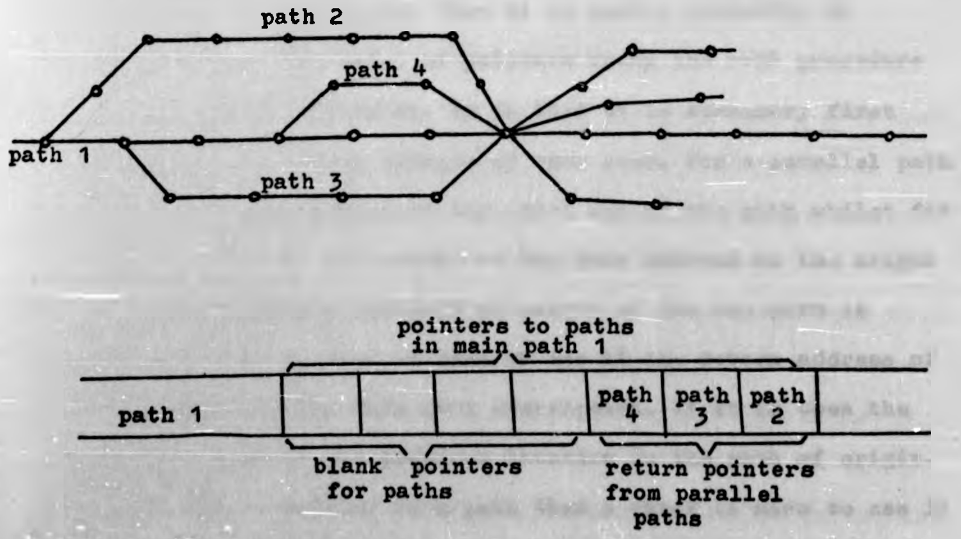


Fig. 4.11. - Nesting of Parallel Path Return Pointers

The path building process continues by jumping onto the new position of the first blank pointer to a junction in the path description and the 'find path block' is used, as before to search for a path radiating out from this point. Once a path has been found then it is necessary to nest it correctly for the path structure described in section 4.2.4. The pointer structure for a possible example on a junction is shown in Fig. 4.12 half way through the path building process on this junction. In this pointer block the pointers are in the order :-

1) Return pointers from previous parallel paths nested correctly.
2) Pointers to the start of parallel paths nested correctly.
3) Pointers to branch arms and loop paths in the order in which they were found.
4) Blank pointers not yet formed into pointer to paths.
5) Empty pointers for loop path returns no longer required.

Initially a new pointer to a path is added to the first blank pointer in this block. Then it is nested correctly by shifting it down this block of pointers using the SWAP procedure until it is nested correctly. To do this it is necessary first to determine the return address of this path. For a parallel path this is the return address of the other end of the path whilst for a loop or branch arm this would be the same address as the origin of the path. Initially the path of origin of the new path is selected and then a check is made to see if the return address of the new path is within this path description. If it is then the program will look at the previous location in the path of origin. If this is also a pointer to a path then a check is made to see if this path returns onto the same path as the new path. If it does then the process will compare the return addressess for these two paths and if the return address for this path is lower than the return address for the new path then the pointers to these two paths are swapped. This process is continued until the path has been nested correctly. So far it has been assumed that the return address for the new path is on the path of origin of this path. If it is not then the process will look at the previous location in the path of origin of this new path. If this is a pointer to a path then the bounds of this path are recorded and the new path swapped with this path. A comparison is made to see if the new path returns within the bounds of this path. If it does then the process will proceed to nest the new path correctly with all the other paths with return addressess within the bounds of this path as above. If it does not return within the bounds of this path then the same process will be carried out on the next path back in the path of origin of this path and so on. If the last path radiating out from this junction has been tried then an error is printed as the topological analysis can not handle this type of circuit.

4.3.6. Check for Parallel Paths Crossing

Once all the paths radiating out from a junction have been completed it is necessary to check all the paths returning to higher addressess in the main path to see if any of these paths cross each other indicating that a chain matrix analysis can not be used for the analysis. An example of this is shown in Fig. 4.13 in which path 1 is the main path. In the path search on junction A two parallel paths no. 2 and 3 will be found and a path search on junction B will give two parallel paths no. 4 and 5 but path 3 crosses path 4. The check for parallel paths crossing is carried out after all the paths leading out of a junction have been completed by stepping one position at a time from the current to the last address in the main path. Initially a marker is set at the current address in the main path . If a return pointer for a parallel path is found during this process then this marker is compared to the origin address of this parallel path. If it is less than this origin address then this marker is set to this origin address otherwise a crossing parallel path has been detected in the circuit and an error is printed out.

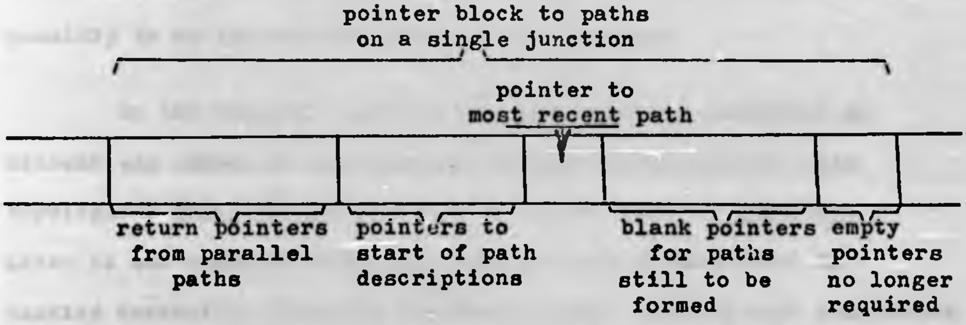


Fig. 4.12. - Pointer Structure for Paths on a Junction

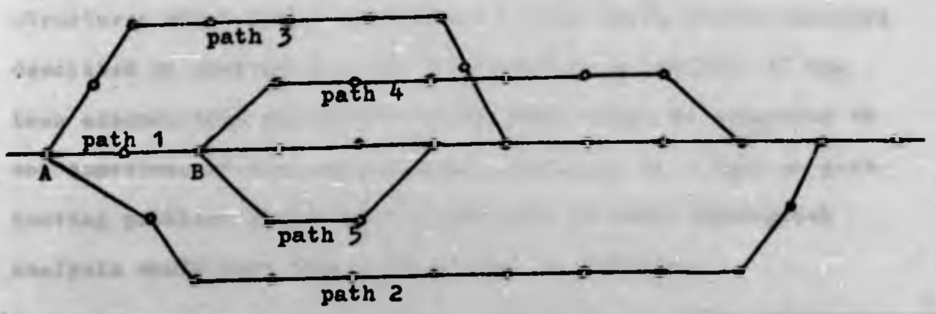


Fig. 4.13. - Crossing Parallel Paths in a Circuit

4.4. CONCLUSIONS

The path topological analysis described in this chapter is quite definitely new work. The result of this work is that it is possible to greatly extend the application of chain matrix analysis for the analysis of circuits consisting of an assembly of cascades of 2-port networks. Thus a computer program can be written using chain matrix analysis, see chapter 5, which is very versatile, i.e. the program will be easier to use for design work possibly in an interactive mode on the computer.

In the original work on the path topology described no account was taken of the problem of path nesting and the path topological analysis was found to be fairly easy to organise. Later it was realised that some path structures would not be handled correctly. Thus the problems of path nesting were considered and the topological analysis redesigned to take this into account. It was found that it became more and more difficult to include all the possible path structures in the program. The present topological analysis will handle virtually all practical path structures which could be analysed by the chain matrix analysis described in section 2.2. In the topological analysis it has been assumed that any number of networks could be connected to any junction. If this number were restricted to 3 then no path nesting problems would have arisen and the path topological analysis would have been much simpler to organise.

References

4.1) F.R.A. Hopgood, 'Compiling Techniques', MacDonald, 1969

4.4. CONCLUSIONS

The path topological analysis described in this chapter is quite definitcly new work. The result of this work is that it is possible to greatly extend the application of chain matrix analysis for the analysis of circuits consisting of an assembly of cascades of 2-port networks. Thus a computer program can be written using chain matrix analysis, see chapter 5, which is very versatile, i.e. the program will be easier to use for design work possibly in an interactive mode on the computer.

In the original work on the path topology described no account was taken of the problem of path nesting and the path topological analysis was found to be fairly easy to organise. Later it was realised that some path structures would not be handled correctly. Thus the problems of path nesting were considered and the topological analysis redesigned to take this into account. It was found that it became more and more difficult to include all the possible path structures in the program. The present topological analysis will handle virtually all practical path structures which could be analysed by the chain matrix analysis described in section 2.2. In the topological analysis it has been assumed that any number of networks could be connected to any junction. If this number were restricted to 3 then no path nesting problems would have arisen and the path topological analysis would have been much simpler to organise.

References

4.1) F.R.A. Hopgood, 'Compiling Techniques', MacDonald, 1969

Chapter 5

Chain Matrix Analysis

CHAIN1 Program

5.1. INTRODUCTION

This chapter describes the computer program, CHAIN1, written for the analysis of microwave circuits using chain matrix analysis as described in section 2.2. The objective of this program was to make the application of the chain matrix analysis more versatile. In previous programs using chain matrix analysis the path structure of the circuit, in terms of cascades of 2-port networks, always had to be defined in a very precise way in the data. In the CHAIN1 program each 2-port network in the circuit is connected between two junctions and the networks can be included in the data at any point and in any order. Two junctions must be assigned, in the data, as the input and output ports of the circuit. Loads could be attached to any junction in the circuit but these loads are only included in the analysis if they terminate a branch arm or are connected to the input or output ports of the circuit.

The main flow diagram for the CHAIN1 program is shown in Fig. 5.1. In this flow diagram the data is read in until an analysis statement is read in the data. Then the circuit is first processed by the path topological analysis to break the circuit up into a path structure as described in chapter 4. Then the circuit analysis, using chain matrix analysis, is used to analyse the circuit to produce a table of results of the circuit performance over a frequency range defined in the data.

Only the organisation of the CHAIN1 program is described in this chapter. A report on the use and preparation of the data for this program is given in Appendix A.4 and the reader may prefer to read this first to understand the facilities offered by the program.

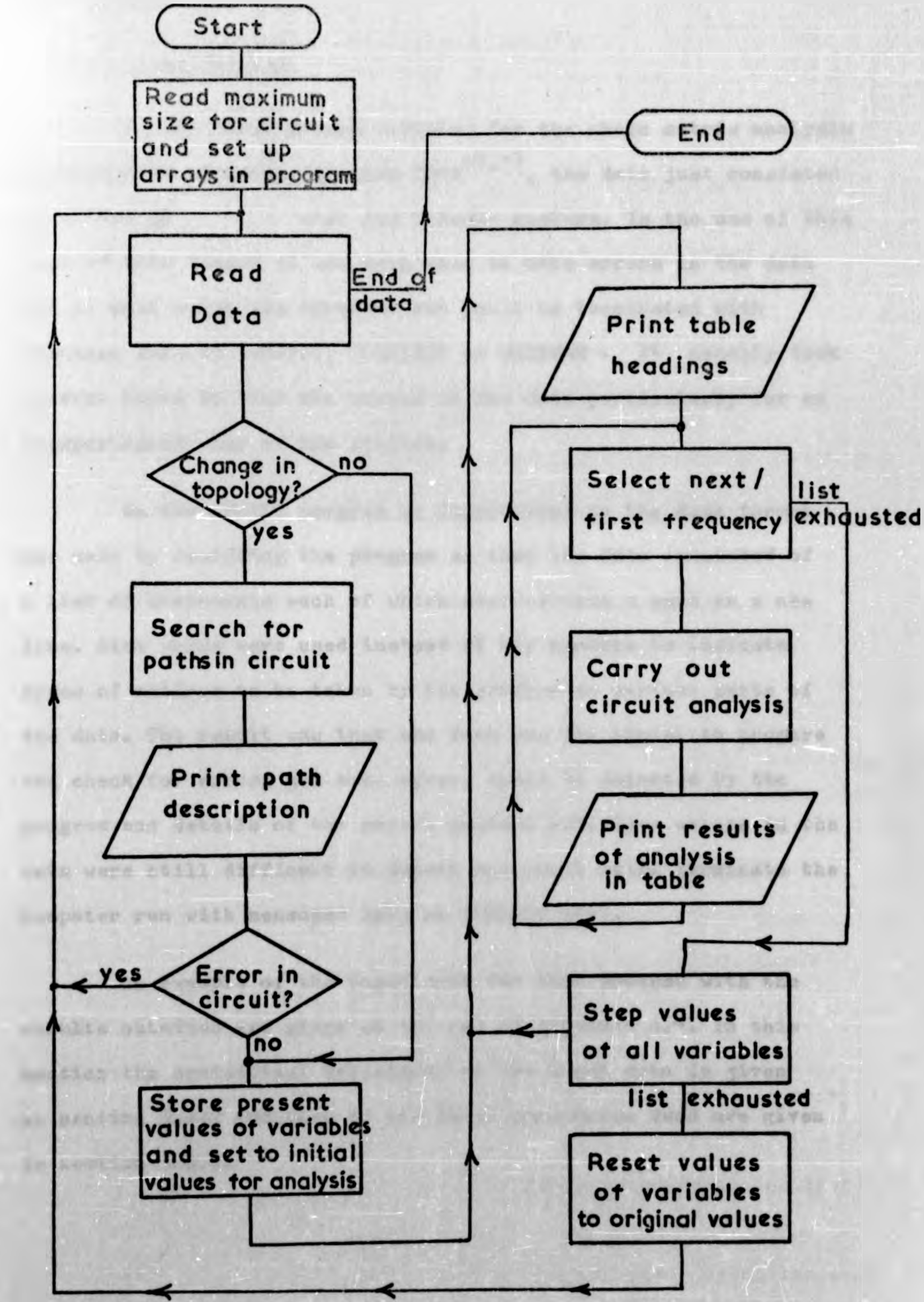

Start
Read maximum size for circuit and set up arrays in program
Read Data
End of data
End
Change in topology?
no
yes
Search for pathsin circuit
Print path description
Error in circuit?
yes
no
Store present values of variables and set to initial values for analysis
Print table headings
Select next / first frequency
list exhausted
Carry out circuit analysis
Print results of analysis in table
Step values of all variables
list exhausted
Reset values of variables to original values

Fig.5.1.- Flow Diagram for CHAINI Program

5.2. DATA FORMAT

5.2.1. Introduction

In the first program written for the chain matrix analysis of microwave circuits, program BGMA(5.1), the data just consisted of a list of real and integer numbers. In the use of this type of data format it was very easy to make errors in the data and in most cases the computer run would be terminated with messages such as SUBOFLO, BOUNDERR or READERR . It usually took several hours to find the errors in the data particularly for an inexperienced user of the program.

In the CHAIN1 program an improvement to the data format was made by designing the program so that the data consisted of a list of statements each of which started with a word on a new line. Also words were used instead of key numbers to indicate types of actions to be taken by the program in various parts of the data. The result was that the data wac far easier to prepare and check for errors and some errors could be detected by the program and details of the errors printed out. Some errors in the data were still difficult to detect and could still terminate the computer run with messages such as SUBOFLO etc..

An example of the input data for this program with the results obtained are given at the end of Appendix A.4. In this section the syntactical definition of the input data is given in section 5.2.2 and some of the basic procedures used are given in section 5.2.3.

5.2.2. Syntactical Definition of Data for CHAIN1

```
<entire input data> ::= <job title string> ;
                        <maximum size of circuit>
                        <input data>

<maximum size of circuit> ::= <maximum no. of junctions>
                              <maximum no. of networks>
                              <maximum no. of loads>
                              <maximum no. of variables>

<input data> ::= <statement list>

<statement list> ::= <statement> | <statement list> <statement>

<statement> ::= <network statement> | <junction statement> |
                <load statement> | <ports statement> |
                <output options statement> | <frequency statement> |
                <variable statement> | <circuit statement> |
                <title statement> | <analysis statement> |
                <termination statement>
```

N.B. 1) Each statement starts on a new line.

2) All spaces and line feeds are ignored except to serve as separators.

Network Statement

```
<network statement> ::= NETWORK <network no.> <junction connection list>
                        <network type no.> <network parameter list> |
                        NETWORK <network no.> <empty connection list>

<network no.> ::= <integer >0 and ≤maximum no. of networks>

<junction connection list> ::= <junction no.> <junction no.>

<junction no.> ::= <integer >0 and ≤maximum no. of junctions>

<network type no.> ::= <see Table A.4.1 >

<network parameter list> ::= <see Table A.4.1>

<empty connection list> ::= <junction no.> 0 | 0 <junction no.> | 0 0
```

Junction Statement

```
<junction statement> ::= JUNCTION <junction type> <junction list> 0

<junction type> ::= SERIES | PARALLEL

<junction list> ::= <junction no.> | <junction list> <junction no.>
```

Load Statement

```
<load statement> ::= LOAD <load nc.> <junction no.> <load type no.>
                          <load parameter list>

<load no.> ::= <integer >0 and ≤maximum no. of loads >

<load type no.> ::= <see section A.4.3.4 >

<load parameter list> ::= <see section A.4.3.4 >
```

Port Statement

```
<port statement> ::= PORT <port no.> <junction no.>

<port no.> ::= 1 | 2
```

Output Options Statement

```
<output options statement> ::= OPTION <options list> 0

<options list> ::= <option no.>| <option list> <option no.>

<option no.> ::= <see Table A.4.2 and A.4.3 >
```

Frequency Statement

```
<frequency statement> ::= FREQUENCY <lower frequency value >
                                <step frequency value><upper frequency value>
```

Variable Statement

```
<variable statement> ::= VARIABLE <variable no.> <variable data>

<variable no.> ::= <integer >0 and ≤maximum no. of variables >

<variable data> ::= <variable type> <variable range> |
                    NETWORK 0 | LOAD  0

<variable type> ::=  NETWORK <network no.> <network parameter no.>|
                     LOAD <load no.> <load parameter no.>

<network parameter no.> ::= <see Table A.4.1 >

<load parameter no.> ::= <see section A.4.3.4>

<variable range> ::= <lower value> <step value> <upper value>
```

Circuit Statement

```
<circuit statement> ::= CIRCUIT
```

Title Statement

```
<title statement> ::= TITLE <character string not including a ;> ;
```

Termination Statement

```
<termination statement> ::=  END
```

Load Statement

```
<load statement> ::= LOAD <load no.> <junction no.> <load type no.>
                      <load parameter list>

<load no.> ::= <integer >0 and ≤maximum no. of loads >

<load type no.> ::= <see section A.4.3.4 >

<load parameter list> ::= <see section A.4.3.4 >
```

Port Statement

```
<port statement> ::= PORT <port no.> <junction no.>

<port no.> ::= 1 | 2
```

Output Options Statement

```
<output options statement> ::= OPTION <options list> 0

<options list> ::= <option no.> | <option list> <option no.>

<option no.> ::= <see Table A.4.2 and A.4.3 >
```

Frequency Statement

```
<frequency statement> ::= FREQUENCY <lower frequency value >
                           <step frequency value><upper frequency value>
```

Variable Statement

```
<variable statement> ::= VARIABLE <variable no.> <variable data>


<variable no.> ::= <integer >0 and ≤maximum no. of variables >

<variable data> ::= <variable type> <variable range> |
                    NETWORK 0 | LOAD 0

<variable type> ::= NETWORK <network no.> <network parameter no.> |

                    LOAD <load no.> <load parameter no.>

<network parameter no.> ::= <see Table A.4.1 >

<load parameter no.> ::= <see section A.4.3.4>

<variable range> ::= <lower value> <step value> <upper value>
```

Circuit Statement

```
<circuit statement> ::= CIRCUIT
```

Title Statement

```
<title statement> ::= TITLE <character string not including a ;> ;
```

Termination Statement

```
<termination statement> ::= END
```

5.2.3. Syntax Analysis Procedures for CHAIN1

The procedures used in the CHAIN1 program for the processing of the basic input data used the following procedures :-

1) procedure ERROR(n); value n; integer n;

This procedure will print the message CHAIN1 ERROR followed by the value of n, i.e. the number of the error detected, and the fault indicator will be set. This procedure was used for syntax and other errors in the data and the user is usually supplied with a list of the data errors for each error number.

2) procedure TITLE;

This procedure was used to read in a character string in the data. The string is read in one character at a time with the advantage that SUBOFLO can be trapped if the string is too long or contains some illegal characters.

3) procedure WORD;

This procedure will read in the data until a character which is not a line feed or space is found. Then the characters in the data will be read in until a space or line feed is found. The first 4 of these characters are packed into a single location in the computer store. This set of 4 characters in this word in the data is compared to the list of acceptable first 4 characters for all the words in the data packed into single locationsin the computer store. If a match for this character string is found in this list then the procedure returns 'p' as the position of the match in this list otherwise an error message is printed out followed by the first 4 characters of the word read.

4) Real and integer numbers in the data

All the integer and real numbers in the data were read in using Algol read statements in the program.

5.3. CHAIN MATRIX ANALYSIS PROCEDURE

5.3.1. Introduction

The objective of the chain matrix analysis procedure, Fig. 5.2, is to form the chain matrix of the entire circuit between its input and output ports using the method of analysis described in section 2.2 and the result of the path topological analysis described in chapter 4.

In the analysis the chain matrix for each 2-port network is first formed in MATRIX(0) using the procedure AANET, section 5.3.7. and the chain matrix for each path as it is formed is stored in MATRIX(1). During the analysis it may be necessary to store the chain matrix in MATRIX(1) for a given path whilst the analysis proceeds up another path, then this chain matrix is stored on the top of the stack in MATRIX(2), MATRIX(3) etc..

5.3.2. Main Link Path (refer to Fig. 4.3 also)

The analysis of the main link path consists of forming the chain matrix for each 2-port network in the path in turn in MATRIX(0) and forming the chain matrix for the path so far processed in MATRIX(1). Thus each 2-port network chain matrix is added to the chain matrix for the path in MATRIX(1) by the multiplication MATRIX(1):=MATRIX(1)*MATRIX(0) or MATRIX(1):=MATRIX(0) if MATRIX(1) is empty as described in section 2.2.2.

The above process terminates when the end of the path description of the main link in the STORE arrays is reached. Then the chain matrix of the entire circuit between its input and output ports has been formed in MATRIX(1). The results of the analysis are then calculated and stored in the MATRIX stack, Table 5.1, as

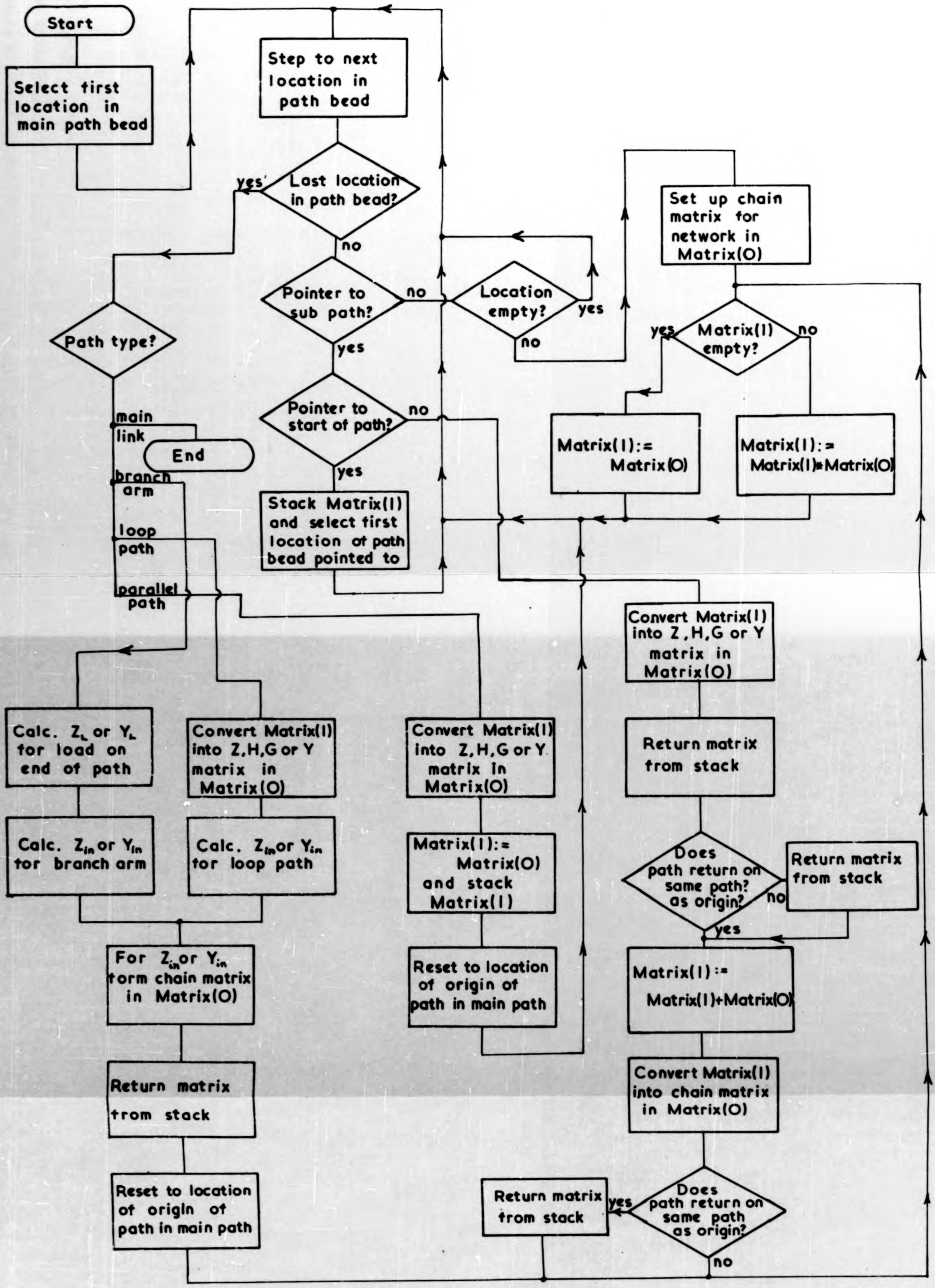

Start
Select first location in main path bead
Step to next location in path bead
Last location in path bead?
yes
no
Pointer to sub path?
no
yes
Location empty?
yes
no
Set up chain matrix for network in Matrix(O)
Matrix(I) empty?
yes
no
Matrix(I):= Matrix(O)
Matrix(I):= Matrix(I)*Matrix(O)
Pointer to start of path?
no
yes
Stack Matrix(I) and select first location of path bead pointed to
Path type?
main link
End
branch arm
loop path
parallel path
Convert Matrix(I) into Z,H,G or Y matrix in Matrix(O)
Return matrix from stack
Does path return on same path? as origin?
no
yes
Return matrix from stack
Matrix(I):= Matrix(I)+Matrix(O)
Convert Matrix(I) into chain matrix in Matrix(O)
Does path return on same path as origin?
yes
no
Return matrix from stack
Calc. Z_L or Y_L for load on end of path
Calc. Z_in or Y_in for branch arm
Convert Matrix(I) into Z,H,G or Y matrix in Matrix(O)
Calc. Z_in or Y_in for loop path
For Z_in or Y_in form chain matrix in Matrix(O)
Return matrix from stack
Reset to location of origin of path in main path
Convert Matrix(I) into Z,H,G or Y matrix in Matrix(O)
Matrix(I):= Matrix(O) and stack Matrix(I)
Reset to location of origin of path in main path

Fig.5.2.-Flow Diagram for Circuit Analysis in CHAINI Program

follows :-

1) Calculate the characteristic impedances for the input and output ports of the circuit.
2) Calculate the numerator and denominator parts of the input and output impedances/admittances of the circuit.
3) Calculate the scattering matrix for the circuit between its input and output ports as described in section 2.2.7.

MATRIX( )	Use	Equivalent subscript in real array AA	Meaning (see section 2.2.7)
0	characteristic impedances	8, 9	Z_{o1}
		10,11	Z_{o2}
		12,13	Z_{o1}^{*}
		14,15	Z_{o2}^{*}
1	chain matrix for circuit	16,17	A
		18,19	B
		20,21	C
		22,23	D
2	input and output impedances and admittances	24,25	A_1
		26,27	C_1
		28,29	B_1
		30,31	D_1
3	scattering matrix	32,33	S'_{11}
		34,35	S'_{12}
		36,37	S'_{21}
		38,39	S'_{22}

Table 5.1. - Results from CHAIN1 program

5.3.3. Sub-paths (ref. to Fig. 4.2 also)

In the main link path, or in any other path, there may be pointers to other paths. If a pointer to the start of a path is found during the analysis of this main path it is necessary to store the chain matrix for the main path so far formed on the top of a matrix stack. Then a jump is made to the start of the path pointed to in the path topology and the chain matrix for this path can be formed in the same way as for the main path. When the end of this path is found then it is necessary to process this path as described in the following sections for the appropriate path type. A pointer in the last location of this path, Fig. 4.2, may be used at the appropriate time to jump back to the origin of this path in the main path.

5.3.4. Branch Arm (refer also to Fig. 4.4)

For the case of a branch arm the value of the load impedance or admittance terminating the branch arm, if one is present, is calculated. Then the input impedance or admittance of the branch arm is calculated, as described in section 2.2.4, and inserted in a 2-port network chain matrix in MATRIX(0). The chain matrix for the path of origin is returned from the matrix stack. Then a jump is made back into the main path at the point of origin of this branch arm and the chain matrix in MATRIX(0) is used for the next network in this main path.

5.3.5. Loop Path (refer also to Fig. 4.5)

The processing for a loop path is the same as for a branch arm except that the input impedance or admittance for the loop path is calculated differently, see section 2.2.6.

5.3.6. Parallel Path (refer also to Fig. 4.6)

For a parallel path with its return address in the main path the chain matrix for the parallel path in MATRIX(1) is converted to Z, H, G or Y matrix for the path, see section 2.2.5. Then this matrix is stored on top of the matrix stack and a jump made to the origin of the parallel path in the main path. Then a chain matrix for the main path between the start and return of this parallel path in the main path is derived in the same way as for the whole main path. When the return pointer for this parallel path is found in the main path the chain matrix for this section of the main path in MATRIX(1) is converted into Z, H, G or Y, as before. The other matrix of the same type for the parallel path is returned from the top of the stack and added to this matrix to give the total matrix for the parallel path pair in MATRIX(1). This is then converted to the chain matrix for a network in MATRIX(O) and the chain matrix from the top of the stack for the main path is returned into MATRIX(1). A jump is then made to the point of return of the parallel path in the main path to continue processing the main path including the network chain matrix in MATRIX(O) as the next network in this path.

(refer at this point to Fig. 4.7)

A complication arises if the parallel path does not return onto its path of origin. In theory this parallel path should be a sub-path on the path to which it returns but this was not easy to organise in the topological analysis. In this case the same process as above can be followed up to the point of return of the parallel path. The resulting matrices in the stack are then :-

1) The chain matrix for the original main path.
2) The Z, H, G or Y matrix for parallel path.
3) An empty matrix for the start of the main path in which this path returns.
4) The chain matrix for the section of the main path in which the parallel path returns.

The required result is that the chain matrix from 4) must be converted to a Z, H, G or Y matrix as before but then it is found that the matrices 2) and 3) have been stacked in the reverse order to the required order for removal. Thus matrix 3) is deleted to remove matrix 2) and when matrix 3) is required it is just set as empty in MATRIX(1).

5.3.7. Setting up Chain Matrices for Networks

The networks included in the CHAIN1 program and their chain matrices are given in Table 5.2. These matrices are set up in MATRIX(0) for a given network using the following :-

procedure AANET;

5.3.8. Conversion Chain to ZHGY Matrices

To analyse a parallel path it is necessary to be able to convert a chain matrix into a Z, H, G or Y matrix and visa-versa. The procedures to do this work are :-

1) procedure A TO ZHGY;

This procedure is entered with k set to the parallel path type as given in Table 4.1. The chain matrix in MATRIX(1) is then converted into the required Z, H, G or Y matrix in MATRIX(0).

2) procedure ZHGY TO A;

This procedure is also entered with k set to the parallel path type as given in Table 4.1. The Z, H, G or Y matrix in MATRIX(1) is then converted into a chain matrix in MATRIX(0).

5.3.9. Matrix Stack and Workspace

During the analysis it is necessary to store the intermediate results for sub-paths, as 2 x 2 complex matrices, in a stack as :-

MATRIX(0) = chain matrix for single network
MATRIX(1) = chain matrix for current path

$$\left.\begin{array}{l}\text{MATRIX(2)}\\ \text{MATRIX(3)}\\ \text{etc.}\end{array}\right\} = \text{matrix stack}$$

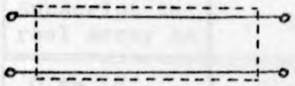
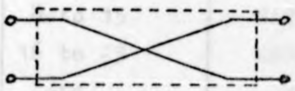
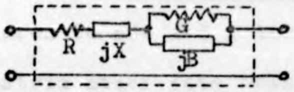
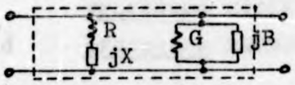
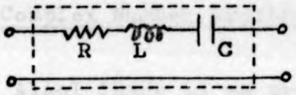
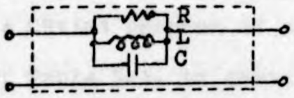
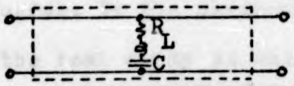
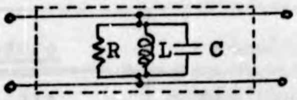
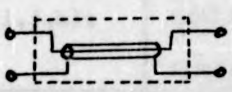
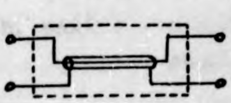
Network type no.	Network	Chain matrix	Comment
1		$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	
2		$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$	
3		$\begin{bmatrix} 1 & Z \\ 0 & 1 \end{bmatrix}$	use $Y := G + jB$ then $Z := R + jX$ if $Y = 0$ $Z := R + jX + 1/Y$ if $Y \neq 0$
4		$\begin{bmatrix} 1 & 0 \\ Y & 1 \end{bmatrix}$	use $Z := R + jX$ then $Y := G + jB$ if $Z = 0$ $Y := G + jB + 1/Z$ if $Z \neq 0$
5		$\begin{bmatrix} 1 & Z \\ 0 & 1 \end{bmatrix}$	$Z := R + j\omega L$ if $C = 0$ $Z := R + j(\omega L + 1/\omega C)$ if $C \neq 0$
6		$\begin{bmatrix} 1 & Z \\ 0 & 1 \end{bmatrix}$	use Y as network type 8 then $Z := 1/Y$
7		$\begin{bmatrix} 1 & 0 \\ Y & 1 \end{bmatrix}$	use Z as network type 5 then $Y := 1/Z$
8		$\begin{bmatrix} 1 & 0 \\ Y & 1 \end{bmatrix}$	use $G := 1/R$ if $R \neq 0$ $G := 0$ if $R = 0$ then $Y := G + j\omega C$ if $L = 0$ $Y := G + j(\omega C - 1/\omega L)$ if $L \neq 0$
9	 $Z_o, \lambda_o, f_o, \alpha_o$	$\begin{bmatrix} \cosh \gamma l & Z_o \sinh \gamma l \\ Y_o \sinh \gamma l & \cosh \gamma l \end{bmatrix}$	$Y_o := 1/Z_o$ and $\gamma l := \alpha l + j\beta l$ where $\alpha l := (\log_e 10/20)\ \lambda_o\ f/f_o$ $\beta l := \omega \lambda/f$ (λ in wavelengths)
10	 $Z_o, l, \epsilon_r, \alpha_o$	$\begin{bmatrix} \cosh \gamma l & Z_o \sinh \gamma l \\ Y_o \sinh \gamma l & \cosh \gamma l \end{bmatrix}$	$Y_o := 1/Z_o$ and $\gamma l := \alpha l + j\beta l$ where $\alpha l := ((\log_e 10)/20)\alpha l$ $\beta l := \omega l \sqrt{\epsilon_r}/c$ where c = speed of light

Table 5.2. - Chain Matrices for 2-port Networks in the CHAIN1 Program

In the program a one dimensional real array was used as follows :-

Subscript in real array AA	Use
0 to 7	Complex number workspace
8 to 15	MATRIX(0)
16 to 23	MATRIX(1)
etc.	etc.

The procedures to add and return the 2 x 2 complex matrices to and from the matrix stack into MATRIX(1) were :-

```
                procedure STACK;
        and     procedure RETURN;
```

5.3.10. Complex Number Arithmetic

In Algol there is no provision for complex number arithmetic. Thus in the CHAIN1 program it was necessary to include some procedures, Table 5.3, to carry out some of the complex arithmetic in this program. In the program all the complex numbers were stored in the real array AA which greatly simplified the procedures.

procedure	result
INVJ(i,j);	cAA(i):=c(1.0,0.0)/cAA(j);
MULTJ(i,j,k);	cAA(i):=cAA(j)*cAA(k); if i≠0 c(ar,ai):=cAA(j)*cAA(k); if i=0
MULADJ(i,j,k,l);	cAA(i):=cAA(j)*cAA(k) + cAA(l);

n.b. cAA( ) = complex number held in the 4 locations starting at location AA( )

Table 5.3. - Complex Number Procedures for CHAIN1

5.4. CONCLUSIONS

The result of the development of the CHAIN1 program was a computer program using chain matrix analysis which was far more versatile than any other program using chain matrix analysis. The main advantage of this program was the inclusion of a path topological analysis to break the circuit into a path structure consisting of cascades of 2-port networks which could be handled easily by the chain matrix method of analysis. Previously it had always been necessary to define the data in a very precise way to define the path structure of the circuit. In the CHAIN1 program it was not necessary to do this. Thus the CHAIN1 program could be used more in a interactive mode where it is possible to modify the data during the running of the program in a very simple way. The program did include words in the data and some diagnostics facilities which made the program easier to use. It could not be said that this program was completely suitable for interactive use on-line on the computer as many errors in the data would still cause a computer run of the program to be terminated.

The main advance in the CHAIN1 program was in respect of the ease of its use as discussed above and most of the time was spent on these improvements. Thus only trivial 2-port networks were included in the program. At the end of the development of the CHAIN1 program the use of chain matrix analysis was being critised by a few people. The problem with chain matrix analysis is that it can only handle 2-port networks in a 2-port circuit. The path topological analysis described in chapter 4 greatly extended the use of chain matrix analysis for circuit analysis, possibly to its limit, but the method could still not handle all microwave circuits. The problem is that generalised microwave circuits consist of an assembly of n-port networks not 2-port networks only.

References

5.1) B.G. Marchent, 'A Computer Program (BGMA) for the Analysis of lumped and Distributed Networks', University of Warwick, Nov. 1969, School of Engineering Science Report No. 50

References

5.1) B.G. Marchent, 'A Computer Program (BGMA) for the Analysis of lumped and Distributed Networks', University of Warwick, Nov. 1969, School of Engineering Science Report No. 50

Chapter 6

List Processing Approach to Data Structure for Mixed Matrix Analysis Program

6.1. INTRODUCTION

In the MICRO3 program,written during this thesis to use the mixed matrix method of analysis as described in section 2.4, it was decided to provide facilities for analysis, optimisation, sub-circuits and interactive use of the program. The implication of this was that the following data had to be stored in the program data structure :-

1) The descriptions of a number of circuits.
2) A list of the components in each circuit.
3) A list of the parameters associated with each component.
4) The component interconnections for each circuit.
5) The connection of each circuit with its exterior.
6) A list of the characteristic impedances for the analysis of any circuit.
7) A list of the frequencies to be included in an analysis.
8) A list of the output options to be included in the results of an analysis.
9) A list of channel assignments for input and output.
10) A list of the parameters assigned as variables in an optimisation.
11) A specification list for the circuit optimisation.
12) Number of interactions for an optimisation.

The number of arrays required for these lists would be much greater than for the CHAIN1 program and also it would be very difficult for the user to decide on the maximum length he would require for these arrays.

For the MICRO3,and the earlier version of this program as the MICRO2 program,it was decided to use a list processing approach to the storage of the data. In this method all the data is stored in blocks of locations in the computer store, termed beads, within a one dimensional integer array. The method of organisation of the data structure is not new. Work on this type of structure was carried by Ewing(6.1) and Williams(6.2) who have described techniques for the storage of complex data structures in terms of list, ring and tree structures. Ross(6.3,6.4) has developed the Advanced Engineering Design free storage package for complex data structure operations for the formation and processes required on this data structure. This package has been used by Thornhill et. al.(6.5) and this use is also described by Brackett(6.6). This work formed a basis of the techniques used for the data structure in the MICRO2 and MICRO3 programs. In the University of Warwick Larcombe(6.7) had developed the WARDEN package for operations on a complex data structure and this package was translated by the author from Fortran to Algol and modified to make it more suitable to the needs of the MICRO2 and MICRO3 programs. The WARDEN package developed by Larcombe was used by Laxon(6.8) in a computer program for the design of mechanical structures. The basic techniques used for the organistion of the storage of the circuit topology and the elimination of bead structures was a development of the ideas used by Laxon.

In this chapter the basic organisation of a list processing approach to the data storage is described in section 6.2. In section 6.3 the way in which this data structure was used in the MICRO2 and MICRO3 programs is described. In this chapter reference will often be made to the address in the data structure. This refers to the subscript in the one dimensional array used to hold the data structure.

6.2. A LIST PROCESSING APPROACH TO DATA STRUCTURE

6.2.1. Introduction

In general as the facilities offered by any computer program are extended the problems met in the storage of the data are as follows :-

1) As the facilities are improved the complexity of the data will increase.
2) Blocks of data of varying length will have to be stored with often, initially, no defined length.
3) Numerous blocks of data of varying types will have to stored.
4) It must be possible to access all the blocks of data associated with each other easily.
5) It may be necessary for blocks of data to be added and deleted at will.
6) It may be required to transfer the entire data structure to backing store for later retrieval.
7) It must be impossible to corrupt the data structure via any possible set of input data.

6.2.2. Bead Structure

In the data structure used each block of data was stored in a block of consecutive locations in a one dimensional array, termed a bead, as shown in Fig. 6.1. The first location in this block is a headerword, Table 6.1, and this contains the length of the bead, its priority and the number of pointers in this bead. In addition the bead will contain, for some applications, a number or name in its first data location so it can be refered from the data, e.g. a network number, a junction number, etc..

6.2.3. Bead Storage

Ideally one would like to be able to allocate a block of

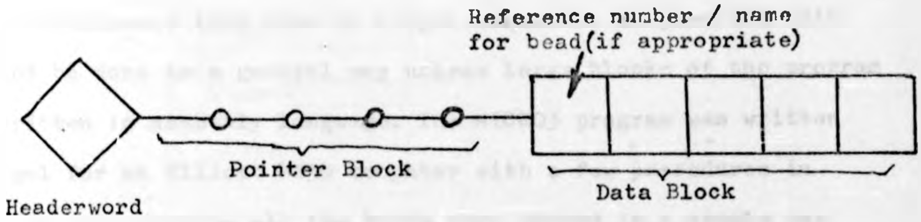


Fig. 6.1. - Bead Structure used in the List Processing Approach to Data Storage

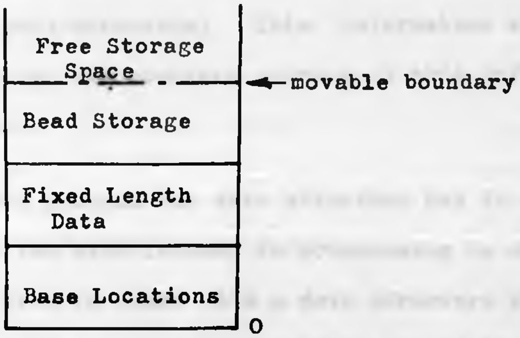
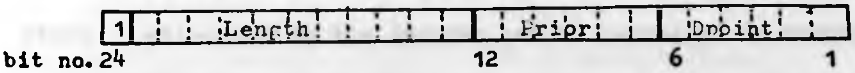


Fig. 6.2. - Space Allocation in Data Structure



Bits in header word	Use	Range
1 to 6	The number of pointers in the bead pointer block.	0 to 63
7 to 12	The priority of the bead, i.e. the type of bead structure of which if is part.	0 to 63
13 to 23	The total length of the pointer and data block in the bead.	0 to 2047
24	Always set to 1. It is used to indicate that a location contains a headerword for a bead (i.e. always negative) and not a pointer.	1

Table 6.1. - Contents of Headerword of Bead in Data Structure

locations in the computer store for each new bead as required and to deallocate them when no longer required. In practice this can not be done in a general way unless large blocks of the program are written in assembly language. The MICRO3 program was written in Algol for an Elliott 4130 computer with a few procedures in NEAT. In this program all the beads were stored in a single one dimensional integer array called the data structure array. In this data structure, Fig. 6.2, all the base pointers to lists in the data structure were stored, with the fixed length data, at the lower address end of the data structure. This information could thus be accessed directly given the absolute address of this information in the data structure.

In the MICRO3 program the data structure had to be of a fixed length due to the restrictions in programming in Algol. In practice one would like to start with a data structure as small as possible to conserve computer store and to expand the size of the data structure, using more free store, as required. The advantage with this method is that on a multi-access computer the size of core store allocated to the program can be increased as required.

6.2.4. Bead Interconnection

Each bead is allocated a priority number in its headerword and this indicates the type of bead structure it is part of, see Table 6.2, as follows :-

1) simple list (priority = 1)

For a simple list, Fig. 6.3, a pointer points to the start of the list and the beads in the list are linked through their first pointers.

2) ring connection (priority = 2 or 3)

A ring connection, Fig. 6.4, is the same as a simple list except that the last bead in the list points

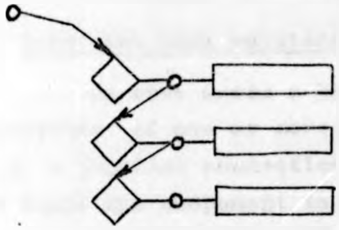


Fig. 6.3. - Simple List of Beads

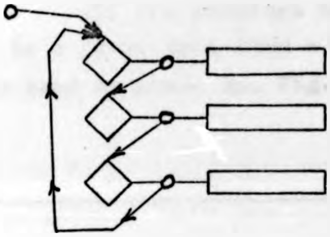


Fig. 6.4. - Ring Connection of Beads

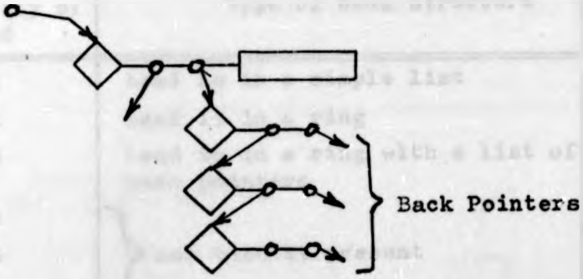


Fig. 6.5. - Bead with Back Pointers

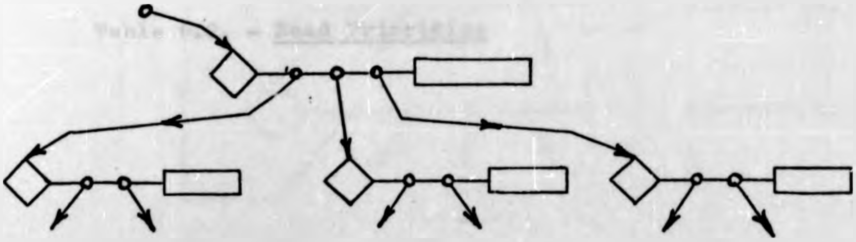


Fig. 6.6. - Tree Structure of Beads

back to the start of the list.

3) <u>bead with back pointers</u> (priority = 3)

In some cases a bead is only generated by the existence of one or more other beads in the circuit, e.g. a junction connection point is only required if at least one component is connected to that junction. This type of bead, Fig. 6.5, is still on a ring but it has a list of back pointers on its second pointer pointing back to the beads pointing to this bead.

4) <u>tree structure</u> (priority = 1, 2 or 3)

All the pointers not accounted for in 1), 2) and 3) in a given bead form a tree structure with this bead at its head as shown in, Fig. 6.6.

priority of bead	type of bead structure
1	bead is in a simple list
2	bead is in a ring
3	bead is in a ring with a list of back pointers
4	
5	not used at present
6	
7	bead is in a garbage list

Table 6.2. - <u>Bead Priorities</u>

6.2.5. Storage of Circuit Topology

In the MICRO3 program a pointer in a base location first points to a list of circuit beads, Fig. 6.7, and another pointer in a base location points to the current circuit being processed. The circuit topology then consists of the circuit bead on the circuit ring which points to a ring of components, junctions and ports for the circuit as shown in Fig. 6.8. In this way a new component, junction or port bead can be added to the appropriate ring in this circuit and also beads can be deleted as desired for these rings.

In the circuit, component and junction rings in the data structure the first data location contains the reference number of the bead, i.e. the circuit number etc.. In this way the bead for a given component, junction or circuit can be located from its number given in the data with component and junction numbers being local to one circuit only.

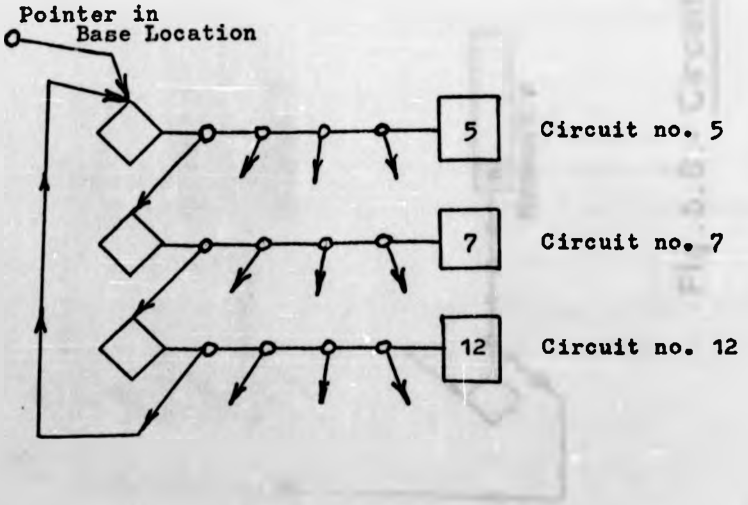


Fig. 6.7. - Circuit Ring

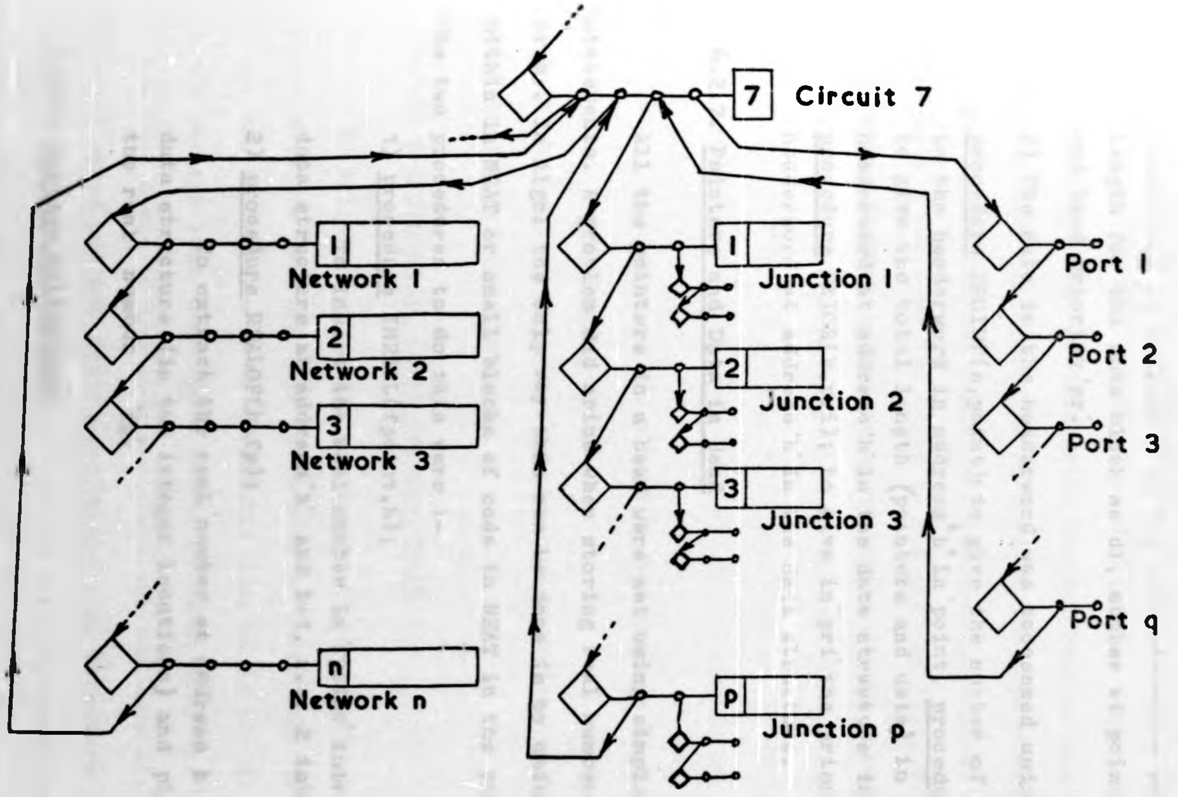


Fig.6.8.- Circuit Topology and Data Storage

6.2.6. Headerword Formation and Access

The procedures written for the formation of the headerword of a bead and to look at the data in the headerword are as follows:-

1) procedure KEY(h,dl,dp,pr);

This procedure will set the contents of the bead headerword at address 'h' in the data structure with the length for the data block as 'dl', number of pointers 'dp' and bead priority 'pr.'

2) The data in the headerword was accesssed using :-

procedure DPOINT(h,point); to give the number of pointers in the headerword in address 'h' in 'point', procedure LENGTH(h,leng) to give the total length (pointers and data) in the headerword at address 'h' in the data structure in 'leng' and procedure PRIOR(h,pri); to give in 'pri' the priority of the headerword at address 'h' in the data structure.

6.2.7. Pointers and Data in Bead

All the pointers in a bead were set using simple assignment statements. A problem did arise when storing real numbers in the array. In Algol the only way this can be done is by using procedures within in NEAT or small blocks of code in NEAT in the program. The two procedures to do this were :-

1) procedure INREAL(fpoint,h);

To insert the real number in 'fpoint' into the data structure at address 'h' and h+1, i.e. 2 integer locations

2) procedure REALOF(h,fp);

To extract the real number at address h and h+1 in the data structure (in two integer locations) and place it in the real number 'fp.'

6.2.8. Garbage Collection

When a bead is no longer required in the data structure it can be deleted, or disconnected, from that data structure and added to a garbage list. The beads in the garbage list still have a headerword but set with a priority of 7 and no pointers. The beads in the garbage list are then linked in a simple list via pointers in their first data location. The order of the beads in this garbage list was arranged to enable these beads to be used most efficiently as new beads in new structures. The way this was done in the MICRO3 program was to order the beads in the garbage list in order of increasing length. Thus on searching the garbage list for a bead to be used in the data structure the first bead of sufficient length is used and any excess length is formed into a new bead and added to the garbage list. This was found to be the most efficient organisation of the garbage list. In the use of the program beads added to the garbage list were soon reused.

The procedure to add a bead to the garbage list was :-

<u>procedure</u> DUMP(h);

so that the bead at address 'h' in the data structure would be added to the garbage list.

6.2.9. Allocation of Space for New Beads

To allocate space for a new bead it was first necessary to search down the garbage list for a bead of sufficient length to use for the new bead. If a suitable bead could not be found in the garbage list then space had to be taken from the bottom of of the free space in the data structure and the boundary, Fig.6.2, of the bead space and the free space in the data structure reset. If sufficient space was not left in the data structure for this new bead then it was necessary to print out an error message and terminate the computer run.

The procedure to allocate space for a new bead was :-

```
integer procedure DEFINE(dl,dp,pr);
```

This procedure will allocate space for a bead with length of the data block equal to 'dl', 'dp' pointers and priority 'pr'. The value of DEFINE will be returned as the address of the new bead.

6.2.10. Insert Bead in Data Structure

In some cases it is more convenient to store beads in a list or ring in order of increasing reference number or name in the first data locations in the beads. The procedure used to do this was :-

```
procedure INSERT(ring,bead,name);
```

This procedure will insert the value in 'name' in the first data location of the bead at address 'bead'. Then this bead will be inserted in the list or ring pointed to in location 'ring' in the data structure.

6.2.11. Locate Bead

If it is necessary to refer to a bead from the input data then it is necessary to search for the bead with the required reference number or name in its first data location in the appropriate list or ring. The procedure used to do this was :-

```
integer procedure LOCNAM(ring,name)
```

This procedure will search for the bead with the value 'name' in its first data location in the list or ring pointed to in address 'ring' in the data structure. If it is found then LOCNAM is returned as the address of this bead otherwise LOCNAM is returned as zero.

6.2.12. Bead Structure Elimination

In the use of a complex data structures in a program it is often necessary to build up various bead structures and later, when one of these structures is no longer required, it may be necessary to eliminate it from the data structure. The types of bead structures it may be necessary to eliminate from the data structure in the MICRO3 program are :-

1) A simple list, Fig. 6.3, e.g. a frequency list, output options list.
2) An element in a ring, Fig. 6.4, e.g. a network or junction.
3) A complete ring, Fig. 6.4, e.g. part of a circuit description.
4) A bead with back pointers, Fig. 6.5, e.g. junction.
5) A tree structure, Fig. 6.6, e.g. a complete circuit description.

In the case of the elimination of a junction bead, or any bead with a list of back pointers, if it is being eliminated directly then it is eliminated by eliminating all the beads pointed to in its list of back pointers and then eliminating the junction bead itself. If the junction bead is only involved in the eliminating because a bead has a pointer pointing to it then the back pointer to this bead is removed from the junction bead's list of back pointers. The junction bead itself is only then eliminated if there are no back pointers left on this bead.

The procedure for the elimination is :-

<u>procedure</u> ELIMIN(head);

where 'head' contains the address of the head bead in the bead structure to be eliminated from the data structure. If 'head' gives the address of a pointer then the bead pointed to in this pointer is used as the head bead instead.

The flow diagram for the ELIMIN procedure is shown in Fig. 6.9 and the elimination process is as follows :-

1) If the bead has a list of back pointers then if this is

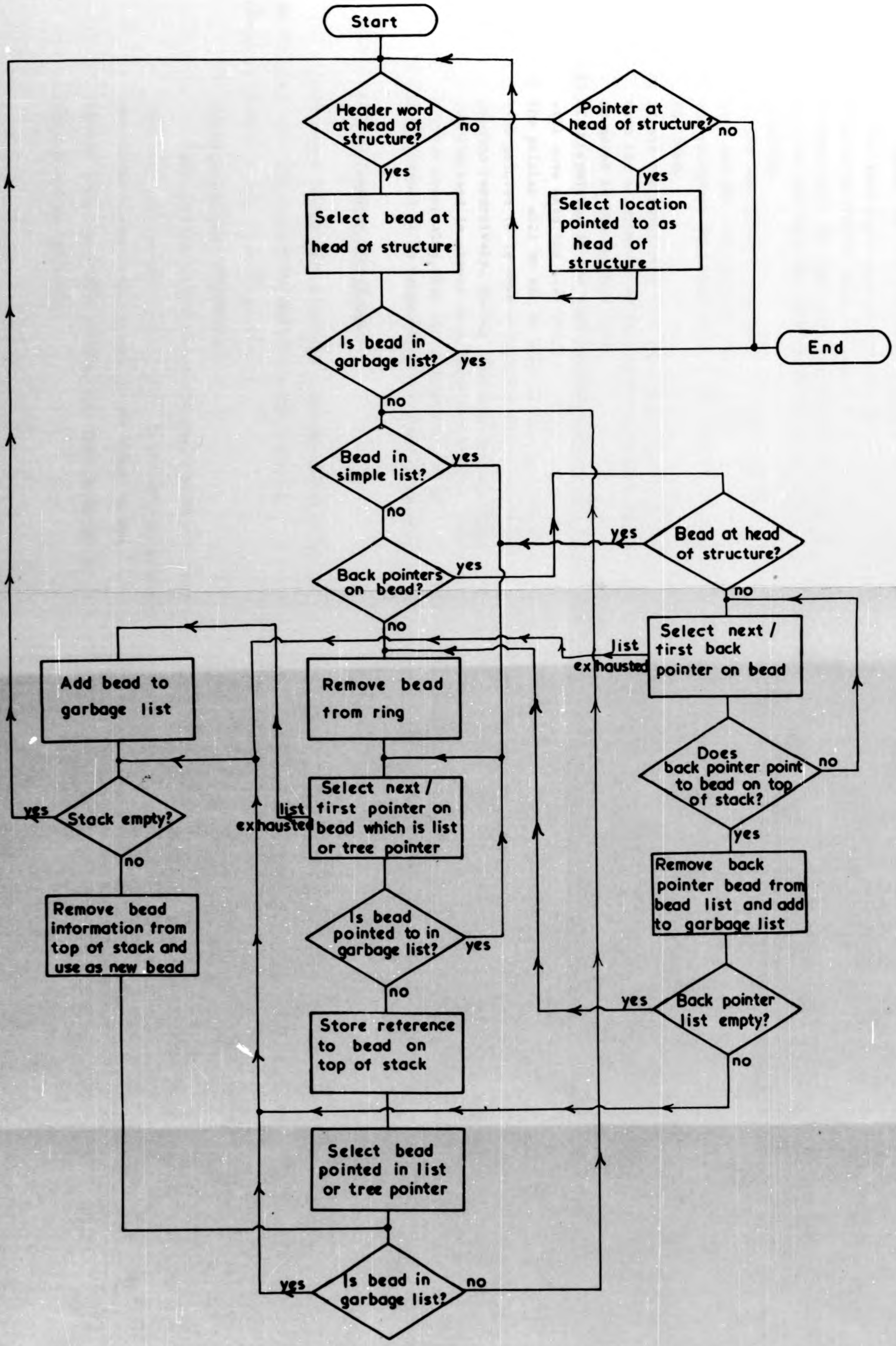

Start
Header word at head of structure?
no
yes
Pointer at head of structure?
no
yes
Select bead at head of structure
Select location pointed to as head of structure
Is bead in garbage list?
yes
End
no
Bead in simple list?
yes
no
Back pointers on bead?
yes
Bead at head of structure?
yes
no
no
Select next / first back pointer on bead
list exhausted
Add bead to garbage list
Remove bead from ring
Does back pointer point to bead on top of stack?
no
yes
Stack empty?
yes
no
list exhausted
Select next / first pointer on bead which is list or tree pointer
Remove back pointer bead from bead list and add to garbage list
Remove bead information from top of stack and use as new bead
Is bead pointed to in garbage list?
yes
no
Back pointer list empty?
yes
no
Store reference to bead on top of stack
Select bead pointed in list or tree pointer
Is bead in garbage list?
yes
no

Fig. 6.9.- Flow Diagram for Bead Structure Elimination

the head bead of the structure then all its pointers are treated as tree pointers. Otherwise the back pointer to the bead which pointed to this bead in the elimination process is removed from this bead and added to the garbage list. If there are still some back pointers left on this bead then the elimination of this bead is complete.

2) If the bead is on a ring then it is removed from that ring and its ring pointer set to empty.

3) The procedure will then look at the first pointer of the bead.

4) If all the pointers on this bead have been processed then the elimination of this bead is complete and it is added to the garbage list.

5) If this pointer in the bead is empty then the process will move onto the next pointer and go back to 4)

6) The pointer will be set to zero in the bead and the bead pointed to is then eliminated using the ELIMIN process recursively. During this process the address of the old bead is stored in a stack and returned for further processing, when the elimination of the bead pointed to is complete, with a jump back to 3)

6.2.13. Circuit Topology procedures

In addition to the data structure procedures described so far there were two procedures written to help form the circuit topology in Fig. 6.8. These were :-

1) integer procedure NEWCIT(cit)

This procedure will locate the bead for the circuit with reference number given in 'cit'. If one is not present then a new circuit bead is set up and added to the circuit ring. The value of NEWCIT is then returned as the address of this bead.

2) procedure JUNCT(hnet,jun,suf);

This procedure is used to set or change a junction connection on a network bead. It is entered with :-

hnet = address of network bead
jun = junction number
suf = position of pointer in network bead for junction connection

If the pointer for this junction connection is already set then this junction connection is removed, i.e. set to zero and the back pointer from the junction in question removed. Then this pointer is set to point to the new junction bead with number 'jun' setting a new back pointer on this junction bead to point back to this network. If during this process the junction bead can not be found for the new connection then a new junction bead is set up and added to the circuit topology. Also if during the removal of the old junction connection the junction is left with no back pointers then it is removed from the circuit topology and added to the garbage list.

6.3. DATA STRUCTURE FOR MICRO3

6.3.1. Use of Base Locations

The use of the base locations in the data structure is shown in Table 6.3. Initially all the pointers to lists are set to empty and pointers to rings set to empty. The pointer to the last location used is set to point to the last address of the base locations and an empty string is entered for the label string.

6.3.2. Circuit Bead

The circuit beads, Fig. 6.10, are held in a ring pointed to in one of the base locations and in addition a base location points to the current circuit being analysed. The format of the circuit topology in the data structure is shown in Fig. 6.8.

Address	Use of Location	Initial Value
1	pointer to last address used	25
2	pointer to garbage list	0
3	pointer to current circuit being processed	0
4	pointer to circuit ring	3
5	pointer to characteristic impedance ring	4
6	pointer to frequency list	0
7	pointer to output options list	0
8	pointer to specification list	0
9	pointer to vary list	0
10 to 14	workspace	
15 to 25	label string	empty string

Table 6.3. - Use of Base Locations in Data Structure

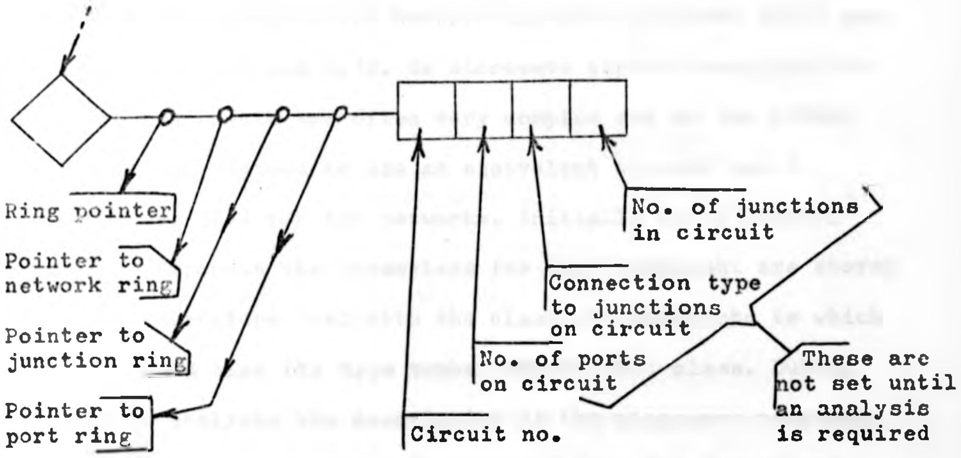

Ring pointer
Pointer to network ring
Pointer to junction ring
Pointer to port ring
Circuit no.
No. of ports on circuit
Connection type to junctions on circuit
No. of junctions in circuit
These are not set until an analysis is required

Fig. 6.10. - Circuit Bead

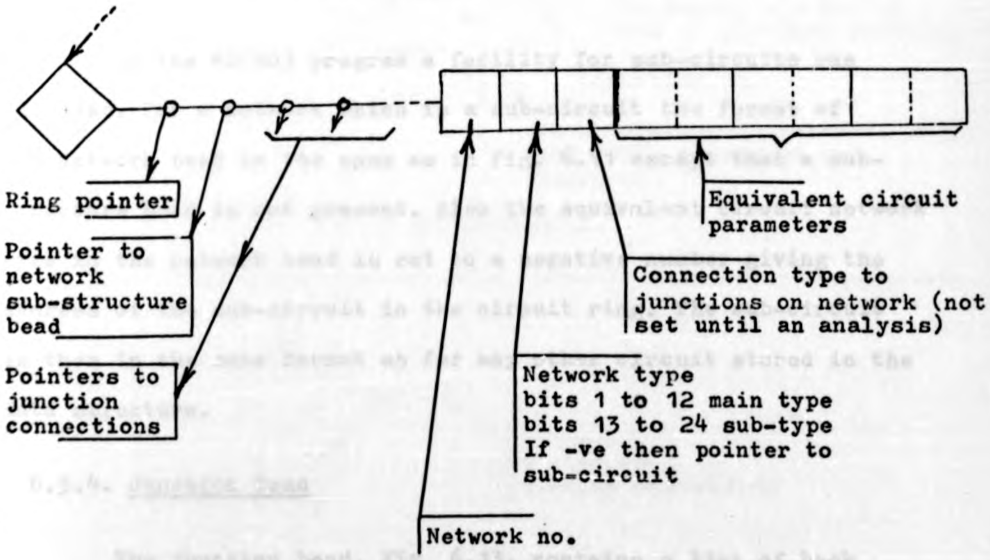

Ring pointer
Pointer to network sub-structure bead
Pointers to junction connections
Network no.
Network type bits 1 to 12 main type bits 13 to 24 sub-type If -ve then pointer to sub-circuit
Connection type to junctions on network (not set until an analysis)
Equivalent circuit parameters

Fig. 6.11. - Network Bead

6.3.3. Network Bead

The contents of the network and sub-structure beads are shown in Fig. 6.11 and 6.12. In microwave circuit analysis the microwave components are often very complex and in the MICRO3 program it was decided to use an equivalent circuit and a sub-structure bead for the networks. Initially for a complex microwave component the parameters for that component are stored in the sub-structure bead with the class of components to which it belongs and also its type number within that class. During the circuit analysis the description of the microwave component in the sub-structure bead is transformed into its equivalent circuit parameters in the network bead for which the equivalent circuit type has previously been set in this bead. Then the circuit analysis is carried out entirely using the equivalent circuits in all the network beads.

In the MICRO3 program a facility for sub-circuits was included. For a network which is a sub-circuit the format of the network bead is the same as in Fig. 6.11 except that a sub-structure bead is not present. Also the equivalent circuit network type in the network bead is set to a negative number giving the address of the sub-circuit in the circuit ring. The sub-circuit is then in the same format as for any other circuit stored in the data structure.

6.3.4. Junction Bead

The junction bead, Fig. 6.13, contains a list of back pointers to all the networks and ports within the current circuit connected to that junction. In the program the junction beads are never set up directly and they are only set up when a network or

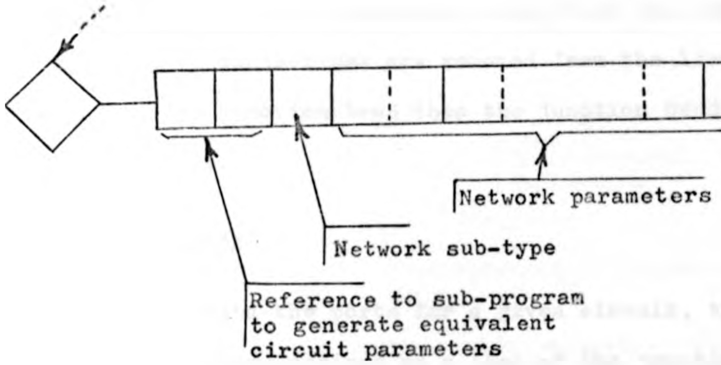


Fig. 6.12. - Network Sub-structure Bead

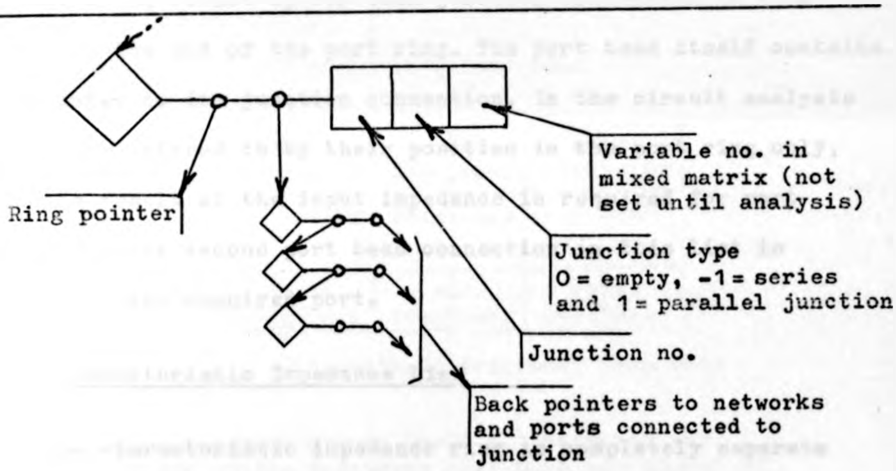


Fig. 6.13. - Junction Bead

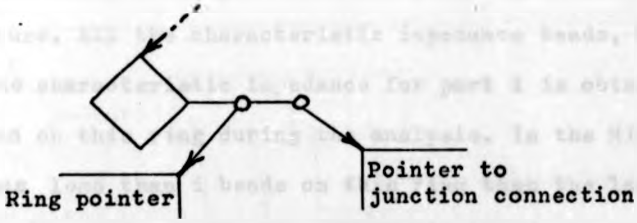


Fig. 6.14. - Port Bead

port is connected to the junction. Also if at any time all the network or port connections are removed from the list of back pointers on the junction bead then the junction bead is deleted from the circuit.

6.3.5. Port Bead

In the data the ports for a given circuit, to connect it to the exterior, are defined as a list of the junctions in order to be defined as port connections for the circuit. Thus a port bead, Fig. 6.14, is set up for each junction connection in turn and added to the end of the port ring. The port bead itself contains just a pointer to its junction connection. In the circuit analysis the ports are refered to by their position in the port ring only, i.e. if the result of the input impedance is required for port number 2 then the second port bead connection in this list is assumed to be the required port.

6.3.6. Characteristic Impedance List

The characteristic impedance ring is completely separate from the circuit description, i.e. it can be used in the analysis of any circuit, and it is pointed to in a base location in the data structure. All the characteristic impedance beads, Fig. 6.15, are on a ring and the characteristic impedance for port i is obtained from the i th bead on this ring during the analysis. In the MICRO3 program if there was less than i beads on this ring then the last bead on the ring is used for the characteristic impedance on port i.

6.3.7. Frequency Bead

The three types of frequency beads are shown in Fig. 6.16 on a typical frequency list. In practice any number of these frequency

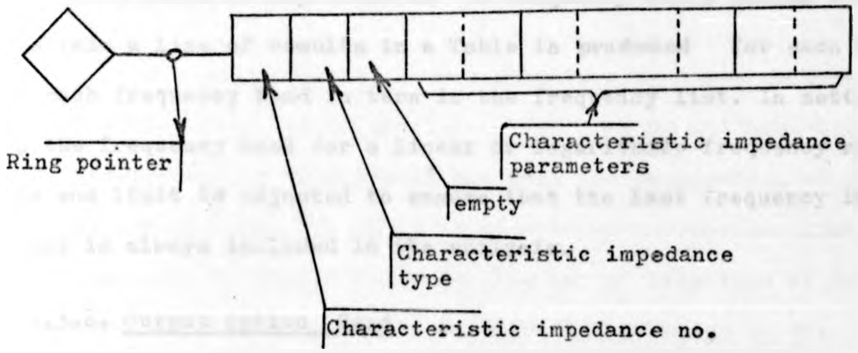
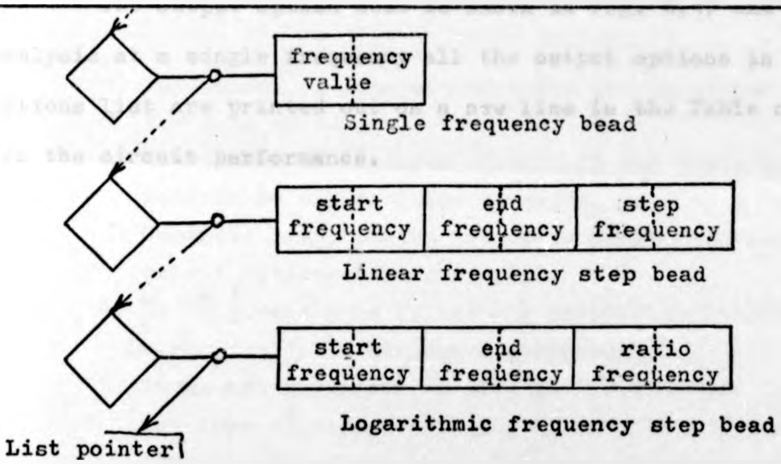

Ring pointer
Characteristic impedance parameters
empty
Characteristic impedance type
Characteristic impedance no.

Fig. 6.15. - Characteristic Impedance Bead



n.b. The above frequency beads may occur in any order in the frequency list with any number of each kind.

Fig. 6.16. - Frequency Bead

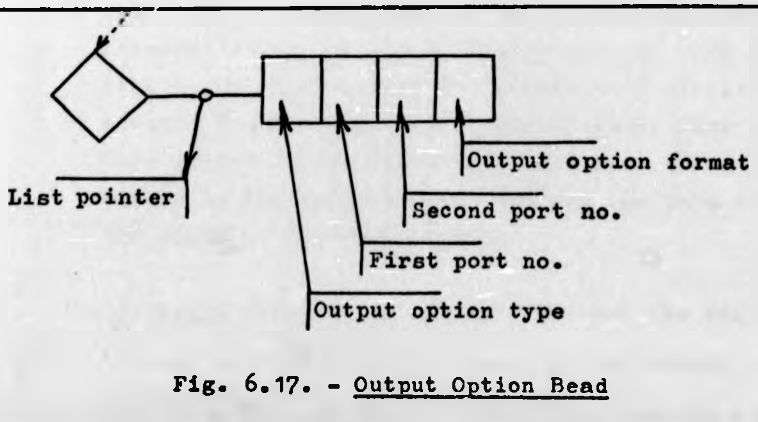


Fig. 6.17. - Output Option Bead

beads may occur in any order in the frequency list. During a circuit analysis a line of results in a Table is produced for each frequency in each frequency bead in turn in the frequency list. In setting up the frequency bead for a linear or logarithmic frequency range the end limit is adjusted to ensure that the last frequency in this range is always included in the analysis.

6.3.8. Output Option Bead

The output option bead is shown in Fig. 6.17 and after an analysis at a single frequency all the output options in the output options list are printed out on a new line in the Table of results for the circuit performance.

6.4. CONCLUSIONS

The organisation of a data structure in terms of beads organised in lists, rings and tree structures is not new. The main development in this thesis has been the application of this type of structure to the analysis of microwave circuits considered as an assembly of n-port networks. The use of this type of data structure has made it possible to include facilities in the program which it would be impossible, or very difficult, to provide in any other way. The facilities areas follows :-

1) Any number of separate circuits can be stored in the data structure.
2) Any number of networks, junctions and ports can be defined in any of these circuits.
3) Separate lists can be set up to store frequency lists, output options lists, etc..
4) It is possible to extend any network to include a complex structure for a microwave component.
5) It is not necessary to give the program any details of the type of structure to be stored for a given set of circuits to be set up in the data structure during a program run.
6) Additions to or deletions from the data structure can be made at any point in the data and to any circuit description.
7) This type of data structure is very suitable for an interactive use of the program where the user would like to be able to describe a number of circuits and analyse them in any order. Also he would like to be able to modify the circuits stored and, if he had made errors in the data, then to correct the data structure and continue the program run.

The original sub-routine package used was the WARDEN package(6.7) which was translated to Algol so the MICRO2 and MICRO3 programs could be written in Algol. During this process a number of alterations were made to make the package more suitable for the way in which it would be used for the MICRO2 and MICRO3 programs. Towards the end of the development of these programs problems were being met with the use of the data structure procedures. A number of the procedures, i.e. INREAL, REALOF, KEY, DPOINT, LENGTH, PRIOR, carried out very simple operations in a block of NEAT code which took about 50μs but they were in an Algol procedure block which took about 300μs to enter and exit from. Thus it was decided to use ML1[6.9,6.10] as a macro-generator to replace all the calls of these procedures in the program source code by their equivalent NEAT code block. This was found to greatly speed up the program by a factor of 2 to 3 on its run time. Also at that time it was considered if it would have been better to program the MICRO2 and MICRO3 programs in Fortran. The result of this was that it probably would have been better to write them in Fortran as it is simpler to operate on the data structure in Fortran. The data structure array can be used both as an integer and real array by the use of an equivalence statement. Also the space above the bead structure in the data structure could be used for the arrays for the circuit analysis by entering a subroutine with an array parameter passed through the subroutine as an address in the free space in the data structure array. This is assuming that a compress integer and logical mode can be used in the compilation of the Fortran program.

References

6.1) D.K. Ewing, 'Implementation of Data Structures for Engineering Design Problems', IEE Conf. Publication No. 51, Computer Aided Design, 15-18 April 1969, pp.600-607

6.2) R. Williams, 'On the Application of Graph Theory to Computer Data Structures', International Symposium on Computer Graphics 70, Wednesday 15 April 1970, Section 3

6.3) D.T. Ross, 'The AED Free Storage Package', Communications of ACM, Vol. 10, No. 8, Aug. 1967, pp. 481-492

6.4) D.T. Ross, 'The AED Approach to Generalised Computer Aided Design', Proceedings of ACM 22nd National Conf., 1967, pp.367-385

6.5) D.E. Thornhill, R.H. Stotz, D.T. Ross, J.E. Ward, 'An Integrated Hardware-Software System for Computer Graphics in Time-sharing', Project MAC Technical Report No. 56, Project MAC, Massachusetts Institute of Technology, Dec. 1968

6.6) J.W. Brackett, 'Case Study in Interactive Graphics Programming : A Circuit Drawing and Editing Program for use with a Storage-tube Display Terminal', 3 day seminar on Software for Interactive Computer Graphics, Brunel University, 8-10 Sept. 1970

6.7) M.H.E. Larcombe, 'The WARDEN System, A Technique for Data Exchange within a Suite of Programs', Proceeding of International Symposium on Computer Aided Structural Design, Vol. 2, University of Warwick, 10-14 July 1972

6.8) W.R. Laxon, 'An Investigation into the use of Computer Graphics for the Design of Skeletal Structures', University of Warwick, Ph. D. thesis, June 1972

6.9) C. Strachey, 'A General Purpose Macrogenerator', Computer Journal, Vol. 8, 1965-66, pp. 225-241

6.10) 'ML1 Users Manual', University Mathematical Laboratory, Cambridge University

Chapter 7

Mixed Matrix Analysis

MICRO3 Program

7.1. INTRODUCTION

The objective of the MICRO3 program was to enable the mixed matrix method of analysis to be used for the analysis of microwave circuits. This method of analysis was far simpler to implement for a generalised microwave circuit than the chain matrix method of analysis used in the CHAIN1 program. In practice the MICRO3 program was larger because far more facilities were included as follows :-

1) A full syntax analysis of the data
2) An extremely versatile format for the data
3) A list processing approach to the data storage
4) Ability to call sub-programs to derive the equivalent circuits for complex networks
5) Interactive use of the program
6) Extension for the use of the visual display

The MICRO3 program was segmented in the computer store to save core store using the following segments with segments 1, 2 and 3 being overlaid :-

1) Outer block - global procedures and main program block
2) Segment 1 - read data and form data structure
3) Segment 2 - perform analysis or optimisation
4) Segment 3 - visual display work

The flow diagram for the MICRO3 program is shown in Fig. 7.1. Initially the channels for the files to be used during the program run are assigned to channel numbers. Then the sub-programs required during the program run are loaded into core store, Table 7.1, and the standard input/output channels are assigned as follows :-

data channel	- channel 50 (this channel must be assigned to a suitable input device before the program run, e.g. card reader, teletype, etc.)
message channel	- lineprinter (for messages and errors)
results channel	- lineprinter (for the results of an analysis)

The data structure is cleared so that the program can start to read in data.

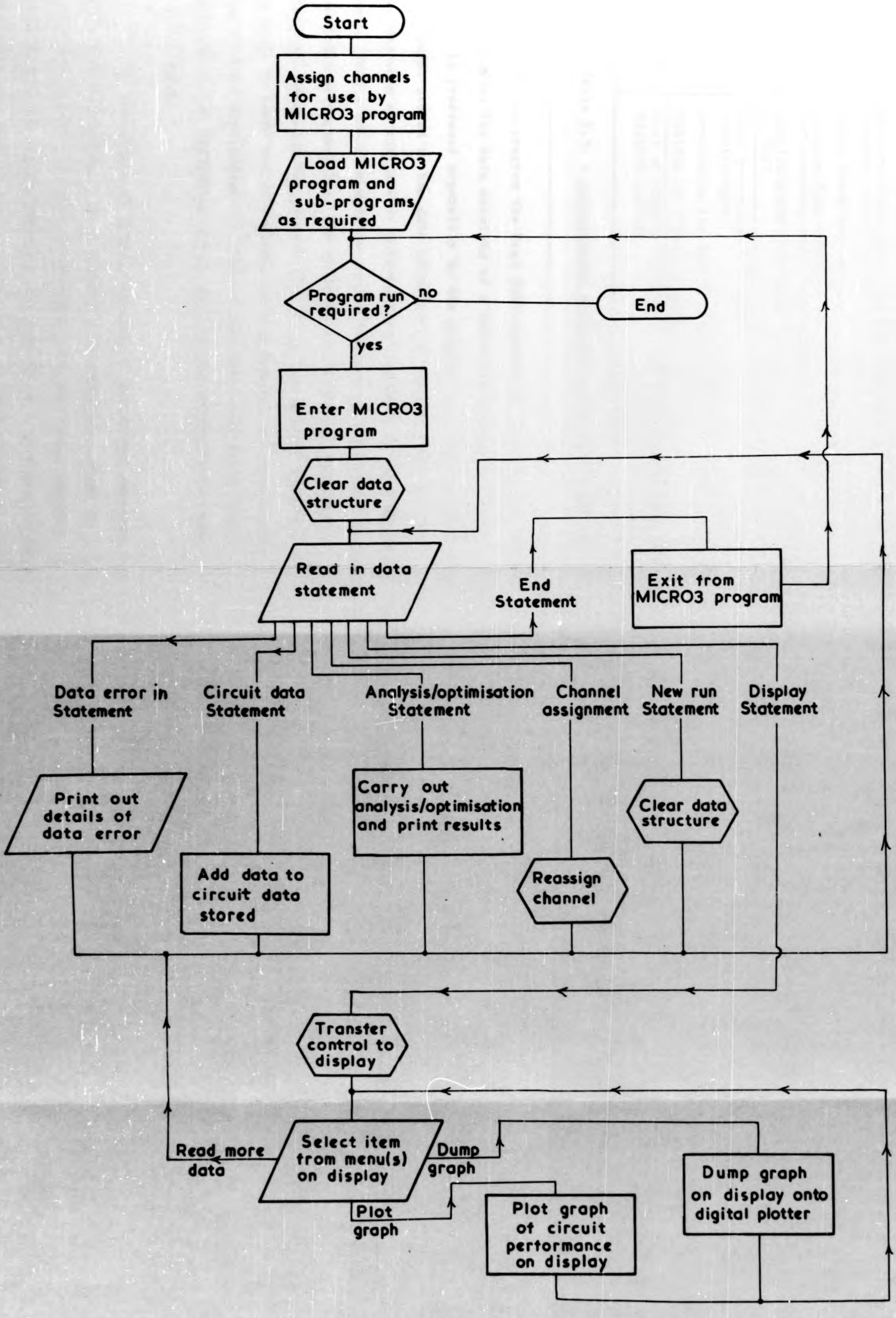

Start
Assign channels for use by MICRO3 program
Load MICRO3 program and sub-programs as required
Program run required?
no
End
yes
Enter MICRO3 program
Clear data structure
Read in data statement
End Statement
Exit from MICRO3 program
Data error in Statement
Circuit data Statement
Analysis/optimisation Statement
Channel assignment
New run Statement
Display Statement
Print out details of data error
Add data to circuit data stored
Carry out analysis/optimisation and print results
Reassign channel
Clear data structure
Transfer control to display
Select item from menu(s) on display
Read more data
Dump graph
Plot graph
Plot graph of circuit performance on display
Dump graph on display onto digital plotter

Fig.7.1.- General Flow Diagram for MICRO3/MIC3D Programs

Program Name	Use of Program (see ref. 7.1)	Comment
ACIO	Channel Assignments	Always required
DRO	Basic Algol Procedures	
SPR	Storage Plan Routines for swapping array codewords	
CMPLEX	Complex number routines in NEAT	
DR10	Disc procedures for array input/output	Only if visual display used
DR50	Procedures for visual display	
DISMAN	Control of visual display	
FIPLOT	Plot of display file on digital plotter	Only if hard copy of visual display picture required

Table 7.1. - Sub-programs required by the MICRO3 program

In the program the Read Data segment is first used to read in the data. The data consists of a series of statements each of which is processed separately by the program and the data in that statement stored in the data structure. A pause in this process occurs when an analysis or optimisation statement is found in the data. Then a check is made to see if the data is sufficient for an analysis or optimisation and if it is not then this statement is ignored. Otherwise the program will enter the Circuit Analysis segment to carry out an analysis or optimisation and then return to the Read Data segment to read in more data statements. The program run is terminated when a termination statement is read in the data.

In the graphical display version of the MICRO3 program, i.e. the MIC3D program, if a statement to transfer control to the visual display is read in the data then the Visual Display segment will be entered provided the results of a previous circuit analysis have been stored on a disc file. In this segment it is possible to plot graphs of the circuit performance, stored on disc during the last circuit analysis, on the visual display on the Elliott 4130 computer. The graph it is required to plot is selected by pointing the light pen at a series of menus on the display. An exit from the display segment to read more data using the Read Data segment can be made by pointing the light pen at a word in one of the menus on the display screen.

In this chapter only the organisation of the MICRO3 program is described. The additional facilities included in the program are described in other chapters as follows :-

Chapter 6 - data structure used
Chapter 8 - interactive facilities for remote teletype on line possible with a graphical display
Chapter 9 - microwave components
Chapter 10 - optimisation.

A report on the use of the MICRO3 program is given in Appendix A.5 with examples of the data and results for the program. The reader may wish to read this report first to gain an understanding of the facilities offered by the program.

7.2. SYNTAX ANALYSIS

7.2.1. Introduction

In the MICRO3 program (and also the MICRO2 program) a very versatile method of reading the data was used. In this method the entire data was read in one character at a time by the program and formed into groups of characters to define a word, integer number, real number and other special characters recognised by the program. In the data if the next group of these characters were not acceptable next in the data then an error message would be printed out and the data up to the next marker would be ignored. This marker would normally be the end of the current statement in the data.

The procedures and the format of the input data are described in section 7.2 and this type of work is applicable to a wide range of applications for the processing of complex data. The data for the program is arranged in the form of statements each one of which starts with a key word and terminates in a suitable terminating character. If an error is detected in the format of the data then the program will print out an error message and search for the start of the next statement in the data. Extensions of the procedures have been made for interactive work as described in chapter 8. For batch processing the data is normally on cards or paper tape and the results and error messages are printed out on the line printer. With this the input data stream is also printed out so that the error messages occur in this stream in the position that they are detected in the data. Each error message print out is followed by the last character group read which caused that error to be detected.

7.2.2. Statement Structure of Data

For a full syntax analysis of the data it is necessary to read in all the data one character at a time. The whole data could be defined as a string of characters which could be broken up as follows :-

```
<input data> ::= <statement list>
<statement list> ::= <statement>| <statement list> <statement>
<statement> ::= <character group list> <statement terminator>
<character group list> ::= <character group >|
                           <character group list><character group>
                           <empty >
< character group > ::= <line feed >|<space>|<erase>|<erase syntax>|
                        <erase syntax and statement>|
                        <erase statement>|<syntax>|<symbol>
< erase syntax > ::= <syntax > <erase >
< erase syntax and statement> ::= <syntax><erase><erase>
< erase statement >::= <erase><erase>
< syntax > ::= <word or identifier>|<integer>| <real>
```

In the MICRO3 program :-

```
1) <statement terminator > ::=   ;
2) <symbol> ::= <any single character which does not start
                any other character group or statement
                terminator >
3) <erase> ::= %
4) <word or identifier > ::= < see Bull(7.2) page 244 >
5) <integer> ::= < see Bull(7.2) page 245 >
6) <real> ::= <see Bull(7.2) page 245 >
```

7.2.3. Syntax Analysis Procedures

The procedure used to read the next character group or statement terminator in the data was :-

<u>procedure</u> SYNTAX(types,err)

This procedure is entered with :-

types = character group(s) which are acceptable next in the data with the bits in the value of types set as follows :-

bit	set to	meaning	integer value
1	1	statement terminator acceptable	1
2	1	integer acceptable	2
3	1	real acceptable	4
4	1	word or identifier acceptable	8
5	0	exit via jump out of procedure if statement terminator found to read next statement in data	16 (if bit set to 1)

err = the syntax error number to be printed out if an acceptable syntax or statement terminator is not found next in the data. In this case the DATERR procedure will be used and this will cause a jump to the start of the processing of the next statement in the data.

A block schematic of the SYNTAX procedure is shown in Fig. 7.2. This is not a precise block schematic but it was very useful in preparing the flow diagrams for the syntactical analysis of the data statements. The dotted lines are not of too great an interest except as a continuous monitoring facility and these are normally omitted in the use of the block schematic.

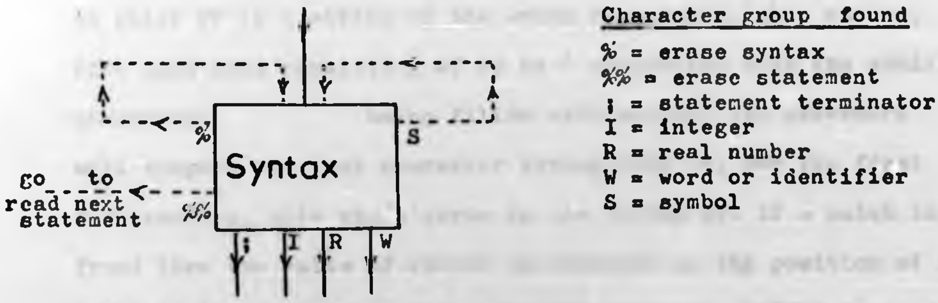


Fig. 7.2. - Block Schematic of Procedure SYNTAX

The points to note in the use of the SYNTAX procedure are as follows :-

1) If an integer is found in the data as the next character group when an integer is not acceptable then it is converted into a real to see if that is acceptable.
2) As each character is accepted as part of a character group then it is stored on the end of a string for later string comparisons or for printing out after a syntax error if the character group is not acceptable.
3) For the exponent part of a real, 'E' is used to replace $_{10}$, e.g. 1.29E9 or .02E-10, as the power $_{10}$ was not available on the teletypes used. A real must not start with E otherwise the E is recognised as a word.

7.2.4. String Comparisons

In the full syntax analysis of the data numerous words are included in the data and it is often necessary to compare a character string representing a word or identifier, obtained from the procedure SYNTAX, with a list of the acceptable words or identifiers possible at that point in the data. The procedure written to do this work was :-

integer procedure FNDWRD(n,ST,err);

in which ST is a string of the words acceptable,'n' in number, with each word consisting of up to 8 characters with the additional characters being filled with spaces. The procedure will compare the last character string read in, for the first 8 characters, with the 'n' words in the string ST. If a match is found then the value of FNDWRD is returned as the position of this match in the string ST. Otherwise the procedure DATERR is used, provided 'err≠0, to print out the syntax error 'err' and to start processing the next statement in the data. If 'err=0 and a match has not been found FNDWRD is returned as zero.

7.2.5. Data Errors

During the syntax analysis of the input data and checking the data before an analysis or optimisation it is necessary to print out the details of the errors found if any. This work was done in the :-

procedure DATERR(n);

This procedure could be used in two ways :-

1) syntax errors, n ⩾ 0

The message <** SYNTAX ERROR> is printed out on a new line followed by the value of 'n' and the character group read last in the data. Then the data is skipped up to the start of the next statement in the data. Then a jump is made out of the procedure to start processing the next statement in the data.

2) circuit error, n < 0

The message <** CIRCUIT ERROR> is printed out on a new line followed by the value of '-n'. The procedure will then return through its normal return link.

For both syntax and circuit errors it is necessary to supply the user with a printed list of the possible errors in the data for each syntax and circuit error number.

7.3. DATA PROCESSING

7.3.1. Introduction

In this section the processing of the input data is described using the network statement as an example. The full syntactical analysis of the input data for the MICRO3 program is given in section 7.3.2 and the processing of most of these data statements, using the syntax procedures in section 7.2 and the data structure described in section 6.3, is obvious from this definition. An example of the data for the MICRO3 program is given in Table 7.2.

The processing of all the statements in the data is carried out in the Read Data segment of the program and the general flow diagram for the processing of the data statements is shown in Fig. 7.3. Initially the data structure is cleared and as each statement is read from the data it is checked against its syntactical definition and at the same time the data from each statement is stored in the data structure. It should be noted that an error could terminate the processing of a statement and leave the data structure connected with this statement partly formed. In writing the program care was taken to ensure that an interupt to the processing of a statement would not produce a corrupt data structure

7.3.2. Syntactical Definition of Data

Statement Groups

```
<statement> ::= <statement group><statement terminator>|
                <statement terminator>
<statement terminator> ::= ;
<statement group> ::= <topology statement>|<analysis statement>|
                      <program control statement>|
                     .<channel assignment  statement>
                      <optimisation statement>|<empty>
```

```
&JOB;  <job number >  ;
&LOAD; MICRO3; DC; 10; ALGOL;
&ASSIGN; 50; CR;

LABEL CIRCULAR HYBRID RING 90 DEG PHASE SHIFTER;
NET 1 LINE WL 70.71 0.25 2E9 0.1 JUNCT 1 2 ;
NET 2 LINE WL 70.71 0.25 2E9 0.1 JUNCT 2 3 ;
NET 3 LINE WL 70.71 0.25 2E9 0.1 JUNCT 3 4 ;
NET 4 LINE WL 70.71 0.75 2E9 0.1 JUNCT 4 1 ;
NET 5 LINE WL 22.584 0.10927 2E9 0.1 JUNCT 2 5 ;
NET 6 LINE WL 76.869 0.33765 2E9 0.1 JUNCT 4 6 ;
NET 7 SRLC 1.0 0.1E-9 0 JUNCT 5 ;
NET 8 SRLC 1.0 0.1E-9 0 JUNCT 6 ;
PORTS 1 2 ;
ZO 1 R 50 ;
JUNCT PARALLEL 1 2 3 4 5 6 ;
OUTPUT ZIN 1 CMPX   SPAR 1 1 MODARG   SPAR 2 1 MODARG
       SPAR 1 1 DBARG   SPAR 2 1 DBARG   VSWR 1 ;
FREQ STEPLIN 1.5E9 2.5E9 20 ;
LABEL DIODES FORWARD BIASSED WITH LINE LOSSES OF 0.1 DB/WL ;
ANALYSE ;
NET 7 PAR 3 0.5E-12 ;
NET 8 PAR 3 0.5E-12 ;
LABEL DIODES REVERSE BIASSED WITH LINE LOSSES OF 0.1 DB/WL ;
ANALYSE ;
END;

&END;
```

Table 7.2. - Typical Data for MICRO3 Program

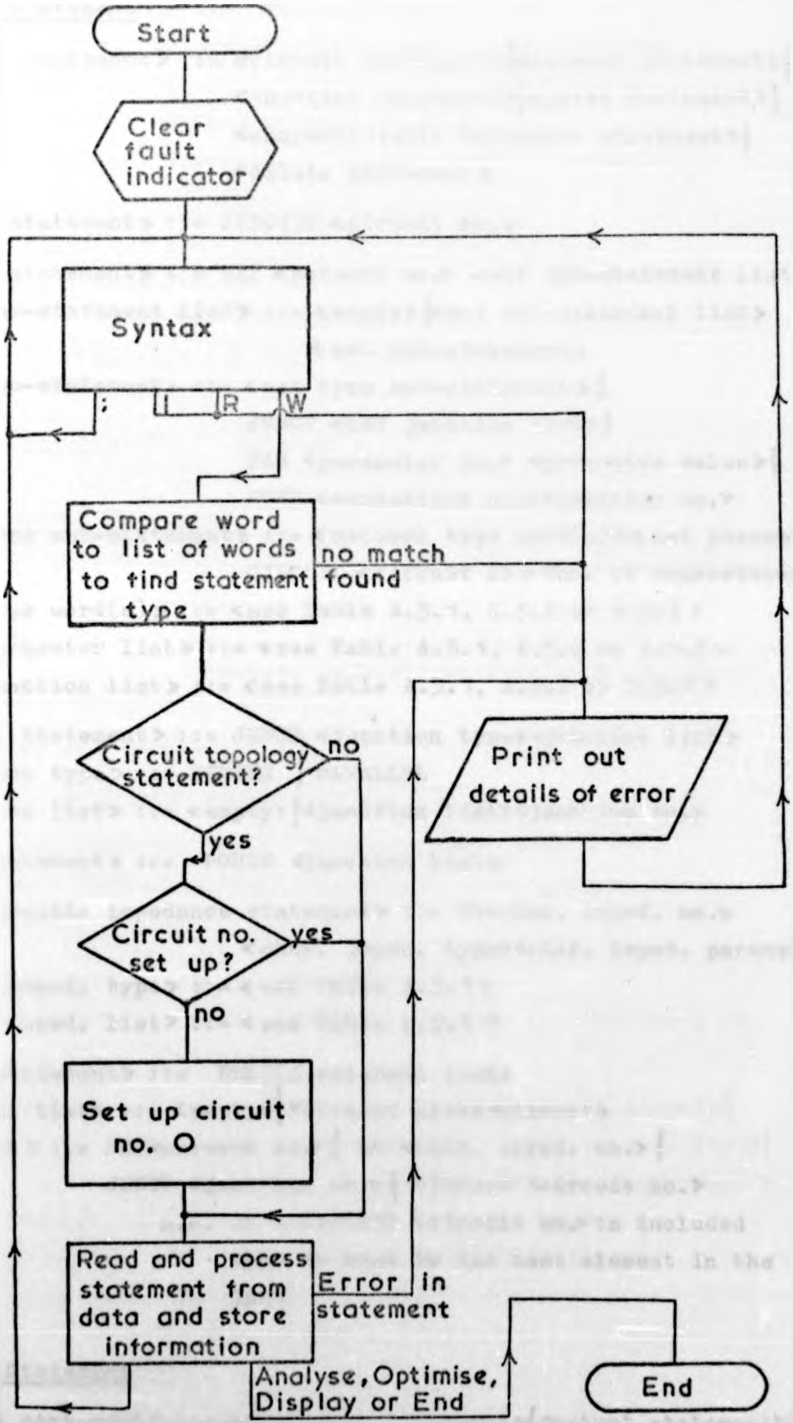

Start
Clear fault indicator
Syntax
;
I
R
W
Compare word to list of words to find statement type
no match found
Circuit topology statement?
no
yes
Circuit no. set up?
yes
no
Set up circuit no. 0
Read and process statement from data and store information
Error in statement
Print out details of error
Analyse, Optimise, Display or End
End

Fig.7.3.- Flow Diagram for Read Data in MICRO3 Program

Topology Statement

```
<topology statement> ::= <circuit statement>|<network statement>|
                         <junction statement>|<ports statement>|
                         <characteristic impedance statement>|
                         <delete statement>

<circuit statement> ::= CIRCUIT <circuit no.>

<network statement> ::= NET <network no.> <net sub-statement list>
  <net sub-statement list> ::= <empty>|<net sub-statement list>
                               <net sub-statement>
  <net sub-statement> ::= <net type sub-statement>|
                          JUNCT <net junction list>|
                          PAR <parameter no.> <parameter value>|
                          CONN <connection no.><junction no.>
  <net type sub-statement> ::= <network type word(s)><net parameter list
                          CIRCUIT <circuit no.><no. of connections>
  <net type word(s)> ::= <see Table A.5.1, A.5.2 or A.5.3 >
  <net parameter list> ::= <see Table A.5.1, A.5.2 or A.5.3>
  <net junction list> ::= <see Table A.5.1, A.5.2 or A.5.3 >

<junction statement> ::= JUNCT <junction type><junction list>
  <junction type> ::= SERIES | PARALLEL
  <junction list> ::= <empty>|<junction list><junction no.>

<ports statement> ::= PORTS <junction list>

<characteristic impedance statement> ::= ZO<char. imped. no.>
                          <char. imped. type><char. imped. parameter list
  <char. imped. type> ::= < see Table A.5.1>
  <char. imped. list> ::= < see Table A.5.1 >

<delete statement> ::= DELETE <element list>
  <element list> ::= <empty>|<element list><element>
  <element> ::= NET<network no.> | ZO <char. imped. no.> |
                JUNCT <junction no.> | CIRCUIT <circuit no.>
                  n.b. If a CIRCUIT <circuit no.> is included
                       then it must be the last element in the
                       list
```

Analysis Statement

```
<analysis statement> ::= <frequency statement>|<output statement>|
                         <analyse statement>
```

```
<frequency statement> ::= FREQ <frequency list>
  <frequency list> ::= <empty>|<frequency list><frequency>
  <frequency> ::= <single frequency>| STEPLIN  <frequency range>|
                STEPLOG <frequency range>
  <frequency range> ::= <start frequency><end frequency><no. of steps>

<output statement> ::= OUTPUT <option list>
  <option list> ::= <empty>|<option list><option>
  <option> ::= <see Table A.5.4  >
              n.b. the full option format, if applicable, is :-
            <option> ::= <option word><port no.>
                          <port no.> <option format>

<analyse statement> ::=  ANALYSE
```

Program Control Statement

```
<program control statement> ::= <new run statement>|<termination statement
                     <reset fault indicator statement>|
                     <program load statement>|<transfer to display>|
                     <output data structure>|<label statement>
<new run statement> ::=  NEWRUN

<termination statement> ::= END

<reset fault indicator statement> ::= NOFAULT

<program load statement> ::=  LOAD <program list>
    <program list> ::= <empty>|<program list><program name>

<transfer to display> ::= DISPLAY

<output data structure> ::= STRUCTURE
<label statement> ::= LABEL <character string not including ;>
```

Channel Assignment Statement

```
<channel assignment statement> ::= <data channel assignment>|
                        <message channel assignment>|
                        <results channel assignment >
  <data channel assignment> ::= DATA <channel no.>
  <message channel assignment> ::= ERROR <channel no.>
  <results channel assignment> ::= RESULT <channel no.>
```

Optimisation Statement

Refer to section 10.2.1.

7.3.3. Statement Processing

1) Circuit Statement - This will find the bead for the given circuit number if it is present in the data structure otherwise a new circuit bead with the given circuit number will be set up. This circuit bead will be inserted in the circuit ring and the pointer to the current circuit being processed will be set to point to this circuit bead.

2) Junction Statement - This will set the junction types in the junction beads for all the junctions given in the junction list.

3) Network Statement - This will set up a new network bead structure or modify the parameters or connections of a present bead structure for the network, see section 7.3.4.

4) Ports Statement - This will add a bead to the end of the ports ring for the current circuit for each junction in the junction list in turn and set the pointer in this bead to point to the appropriate junction bead. Any previous ports ring will be deleted.

5) Characteristic Impedance Statement - This will set up a new characteristic impedance bead and add it to the characteristic impedance ring. Then the parameters will be entered into this new characteristic impedance bead.

6) Delete Statement - This will delete, in turn, the elements given in the element list from the circuit topology.

7) Frequency Statement - This will enter all the frequencies in the frequency list into the frequency bead list.

8) Output Statement - This will enter all the output options in the output options list into the output option bead list.

9) Analyse Statement - This will allow a circuit analysis to be carried out on the circuit topology for the current circuit being processed including all its sub-circuits. Before the analysis the following is checked :-

the fault indicator has not been set
the data and circuit topology is sufficient for analysis

10) New run Statement - This will clear the data structure but the channel assignments will be left unchanged.
11) Termination Statement - This will terminate the program run.
12) Program Load Statement - This will load into the computer store the programs given in the program list.
13) Structure Statement - This will dump out the entire data structure onto the result channel. It is intended to assist a programmer to detect faults in the data structure.
14) Channel Assignment Statement - This will reassign the channel no. for a data, message or results channel.
15) Label Statement - This will store the label string for printing out at the start of an analysis or optimisation.
16) Reset Fault Indicator Statement - This will reset the fault indicator in the program to allow later analysis or optimisation statements to be obeyed.

7.3.4. Example of Data Processing on a Network Statement

The main flow diagram for the processing of the network statement is shown in Fig. 7.4. In this flow diagram after a network statement has been detected in the data the network no. is read in. Then a search is made, in the current circuit description in the data structure, for a bead with this network no.. The rest of the network statement then consists of a list of sub-statements each one of which is processed separately in the order it occurs in this list. The sub-statement type can be recognised by the first word of the sub-statement as follows :-

PAR - reset the value of a single network parameter.
CONN - reset a single junction connection on the network.
JUNCT - set all the junction connections for the network.
any other word(s) - network type, possibly consisting of two words, followed by a list of network parameters.

The processing of each sub-statement for the network statement is shown in Fig. 7.4 with an additional flow diagram in Fig. 7.5 and also in Fig. 7.6 for the network type sub-statement.

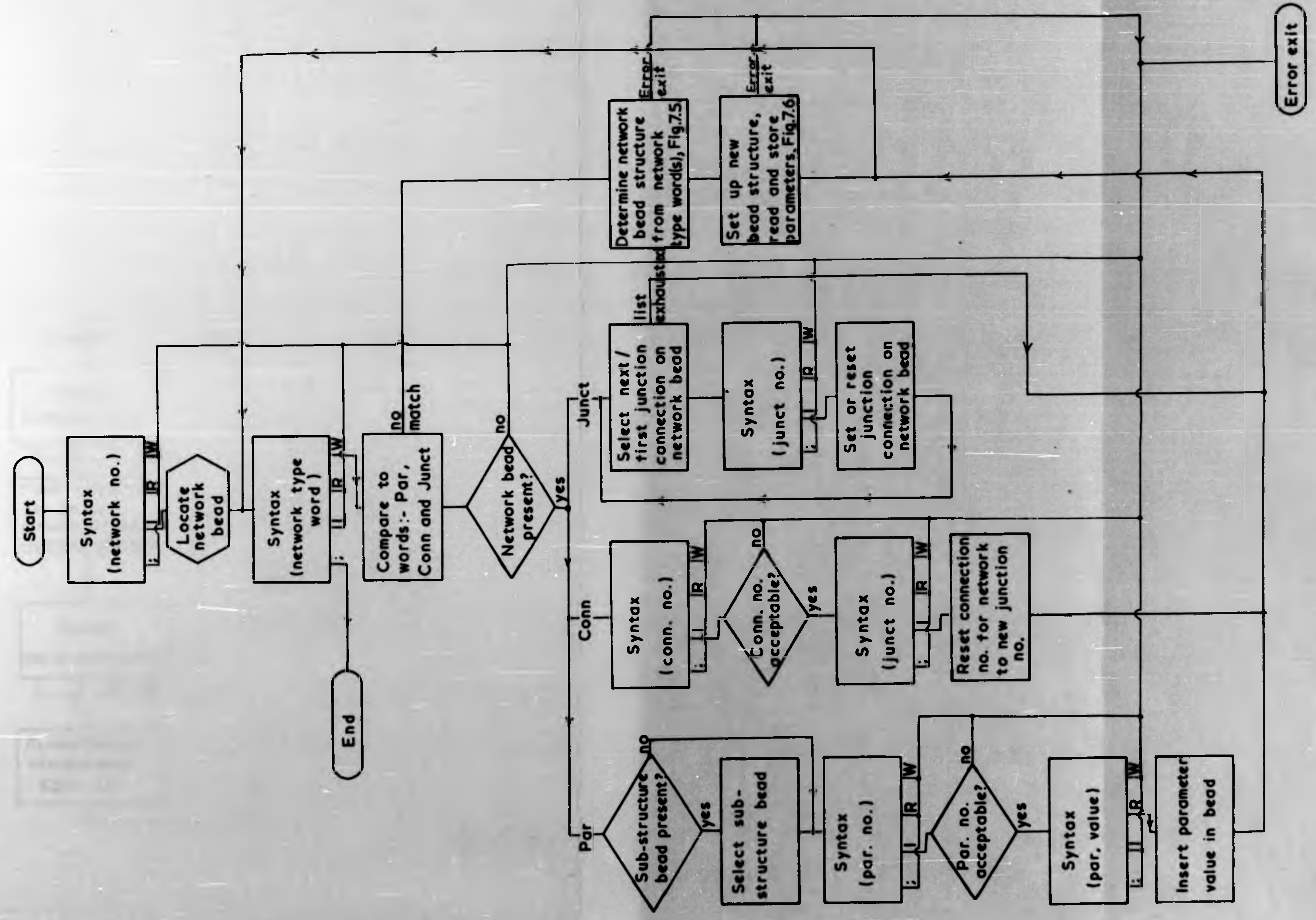
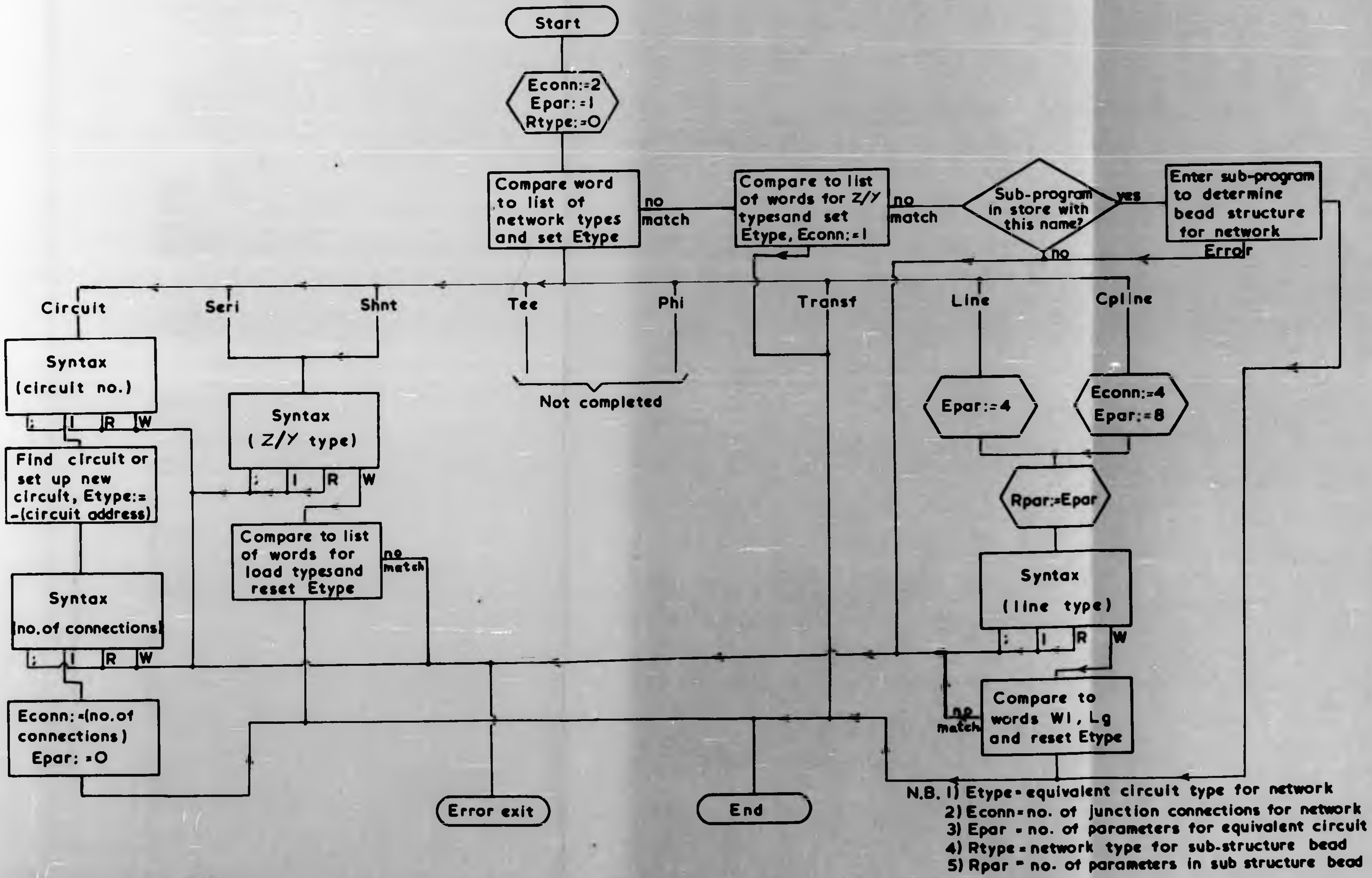

Start
Syntax (network no.)
; I R W
Locate network bead
Syntax (network type word)
; I R W
Compare to words:- Par, Conn and Junct
no match
End
Network bead present?
no
yes
Junct
Conn
Par
Select next / first junction connection on network bead
list exhausted
Syntax (junct no.)
; I R W
Set or reset junction connection on network bead
Syntax (conn. no.)
; I R W
Conn. no. acceptable?
no
yes
Syntax (junct no.)
; I R W
Reset connection no. for network to new junction no.
Sub-structure bead present?
no
yes
Select sub-structure bead
Syntax (par. no.)
; I R W
Par. no. acceptable?
no
yes
Syntax (par. value)
; I R W
Insert parameter value in bead
Determine network bead structure from network type word(s), Fig.7.5
Error exit
Set up new bead structure, read and store parameters, Fig.7.6
Error exit
Error exit

Fig. 7.4.- Flow Diagram for Processing of Network Statement


Start
Econn:=2
Epar:=1
Rtype:=O
Compare word to list of network types and set Etype
no match
Compare to list of words for Z/Y typesand set Etype, Econn:=1
no match
Sub-program in store with this name?
yes
no
Enter sub-program to determine bead structure for network
Error
Circuit
Seri
Shnt
Tee
Phi
Transf
Line
Cplinе
Syntax (circuit no.)
; I R W
Find circuit or set up new circuit, Etype:= -(circuit address)
Syntax [no.of connections]
; I R W
Econn:=(no.of connections) Epar:=O
Syntax (Z/Y type)
; I R W
Compare to list of words for load typesand reset Etype
no match
Not completed
Epar:=4
Econn:=4 Epar:=8
Rpar:=Epar
Syntax (line type)
; I R W
Compare to words Wl, Lg and reset Etype
no match
Error exit
End

N.B. 1) Etype = equivalent circuit type for network
2) Econn = no. of junction connections for network
3) Epar = no. of parameters for equivalent circuit
4) Rtype = network type for sub-structure bead
5) Rpar = no. of parameters in sub structure bead

Fig.7.5.- Flow Diagram for Determining Bead Structure from Network Type

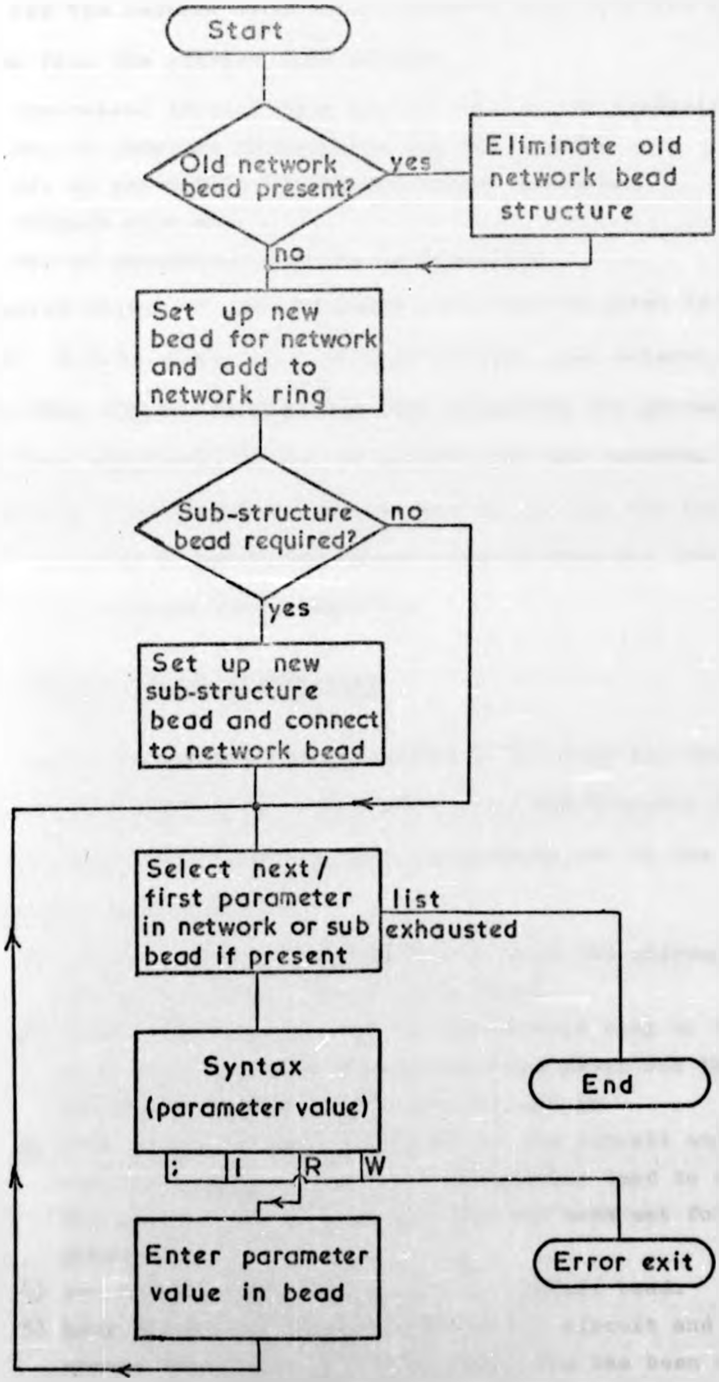

Start
Old network bead present?
yes
Eliminate old network bead structure
no
Set up new bead for network and add to network ring
Sub-structure bead required?
no
yes
Set up new sub-structure bead and connect to network bead
Select next / first parameter in network or sub bead if present
list exhausted
End
Syntax (parameter value)
;
I
R
W
Enter parameter value in bead
Error exit

Fig.7.6.-Flow Diagram for Setting up Bead Structure for Network

For the network type sub-statement, Fig. 7.5, the following is set up from the network type word(s) :-

equivalent circuit type no. for use in the analysis
no. of junction connections for the network
no. of parameters in its equivalent circuit
network type no.
no. of parameters for the network type

The required values of some of these variables is given in Table A.5.1, A.5.2 or A.5.3. Once these parameters have been determined then the flow diagram in Fig. 7.6 will eliminate the present network bead structure, if one is present for the network. Then a new network bead structure will be set up for the new network type and the list of parameter values read in from the data and stored in the network bead structure.

7.3.5. Check Circuit Description

Before an analysis it is necessary to check the circuit topology to be analysed to ensure that it is sufficiently defined for an analysis. The circuit checks are carried out in the CHCIT procedure as follows :-

1) Enter the procedure CHCIT(bead) with the address of the circuit bead structure in 'bead'.
2) Locate the circuit bead on the circuit ring to ensure that it is defined as a circuit and print out the message <** CIRCUIT> and the circuit no.
3) Loop around the junction ring on the circuit and set the variable no. for each junction bead to zero and ensure that a junction type has been set for each junction.
4) Set the no. of junctions in the circuit bead.
5) Loop around the network ring on the circuit and ensure that every junction connection has been set. Also, for each network, set bit i in the network connection type in the network bead, Fig. 6.11, to junction connection type for the i th junction connection on the

network bead for all the network junction connections. For each network the sub-program call in the network bead is set to 1 if the network includes a sub-structure bead.

6) Loop around the port ring and set the variable no. in the junction connection on each port to the port no., i.e. the position in the port ring. Also set bit i in the connection type in the circuit bead, Fig. 6.10, for type of junction connection on port i for all ports.
7) Insert the no. of ports in the circuit bead.
8) Check the no. of ports as >0 and ≤8 (n.b. this limit of 8 can be adjusted by altering the matrix sizes set up in the program).
9) The variable no. is set to zero in a counter and a step is made around each junction in the junction ring in turn. For each junction bead, if the variable no. is empty, then the variable counter is incremented by one and the variable no. in the junction set to this variable counter no.. Otherwise the junction is connected to a port and the variable no. in the junction bead is increased by the difference between the no. of junctions and no. of ports in the circuit. In this way the junctions assigned as ports are placed in the bottom right of the mixed matrix during an analysis.
10) Loop around the network ring and if any of the networks point to a sub-circuit for the network description then enter the procedure CHCIT recursively to check these sub-circuits.

7.3.6. Check Data for Analysis

The procedure CHOPTS(optim) is entered, for a simple analysis, with optim set to false and it will check that sufficient additional information has been supplied for the analysis as follows :-

1) Check that at least one characteristic impedance is present.
2) Check that a frequency list is present for the analysis.
3) Check that an output options list is present for the results.
4) Check that both the port no.s for all the output options are acceptable.

7.4. CIRCUIT ANALYSIS

7.4.1. Introduction

During the Read Data segment of the program the data is read, stored and checked and on leaving the Read Data segment the data structure is ready for an analysis and contains :-

1) A number of circuit descriptions with the circuit to be analysed pointed to in address 3 in the data structure.
2) A ring of characteristic impedance beads.
3) A frequency list for the analysis.
4) An output options list for the results to be included in the table of results.

From this information the Circuit Analysis segment will analyse the circuit at each frequency in the frequency list in turn to give a line of results, in a table, for all the output options included in the output option list. The Circuit Analysis segment will also carry out an optimisation, if location 10 in the data structure is not set to zero, but this will be described in chapter 10.

The flow diagram for the Circuit Analysis segment is shown in Fig. 7.7. Initially the arrays are set up to hold the results of the analysis as follows :-

<u>array</u> MIXZY(,) - for the mixed matrix of the circuit.
<u>array</u> SPAR(,) - for the power scattering matrix for the circuit.
<u>array</u> ZO( ) - for the characteristic impedances for the circuit.

On the results channel the label string is first printed out followed by the table headings for all the results in the output options list. Then the first bead in the frequency list is selected and the first frequency in this bead selected. An analysis is then carried out at this frequency and the results given in the output options list are printed, with the frequency, in the next line in the table of

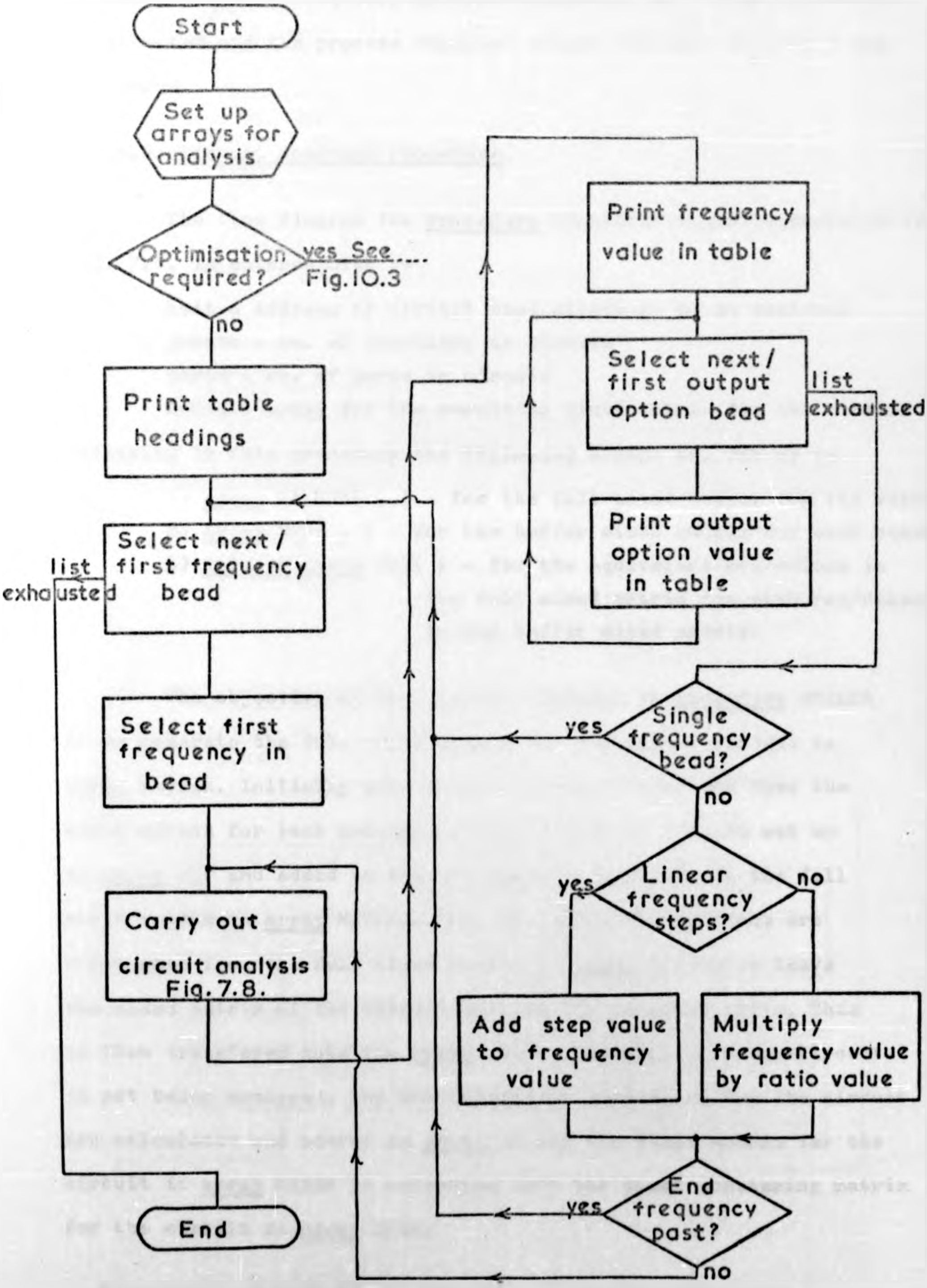

Start
Set up arrays for analysis
Optimisation required?
yes See Fig.10.3
no
Print table headings
Select next / first frequency bead
list exhausted
Select first frequency in bead
Carry out circuit analysis Fig.7.8.
End
Print frequency value in table
Select next / first output option bead
list exhausted
Print output option value in table
Single frequency bead?
yes
no
Linear frequency steps?
yes
no
Add step value to frequency value
Multiply frequency value by ratio value
End frequency past?
yes
no

Fig.7.7.- Flow Diagram for Analysis in MICRO3 Program

results. The next frequency in the current or next frequency bead is selected and the process repeated unless the last frequency has been used.

7.4.2. Circuit Analysis Procedure

The flow diagram for procedure ANALYS(hcit,juncts,ports,AAMIX), Fig. 7.8, is entered with :-

hcit = address of circuit bead structure to be analysed
juncts = no. of junctions in circuit
ports = no. of ports in circuit
AAMIX = array for the resulting mixed matrix for the circuit

Initially in this procedure the following arrays are set up :-

1) array MATRIX(,) - for the full mixed matrix for the circuit
2) array BUF(,) - for the buffer mixed matrix for each network
3) integer array ROW( ) - for the equivalent row/column in the full mixed matrix for each row/column in the buffer mixed matrix.

The objective of the circuit analysis in procedure ANALYS is to generate the full mixed matrix for the entire circuit in array MATRIX. Initially this matrix is set to zero and then the mixed matrix for each network in the circuit in turn is set up in array BUF and added to the appropriate positions in the full mixed matrix in array MATRIX. Then the internal junctions are eliminated from the full mixed matrix in array MATRIX to leave the mixed matrix of the circuit between its external ports. This is then transfered into the array MIXZY. Finally, if a sub-circuit is not being analysed, the characteristic impedances for the circuit are calculated and stored in array ZO and the mixed matrix for the circuit in array MIXZY is converted into the power scattering matrix for the circuit in array SPAR.

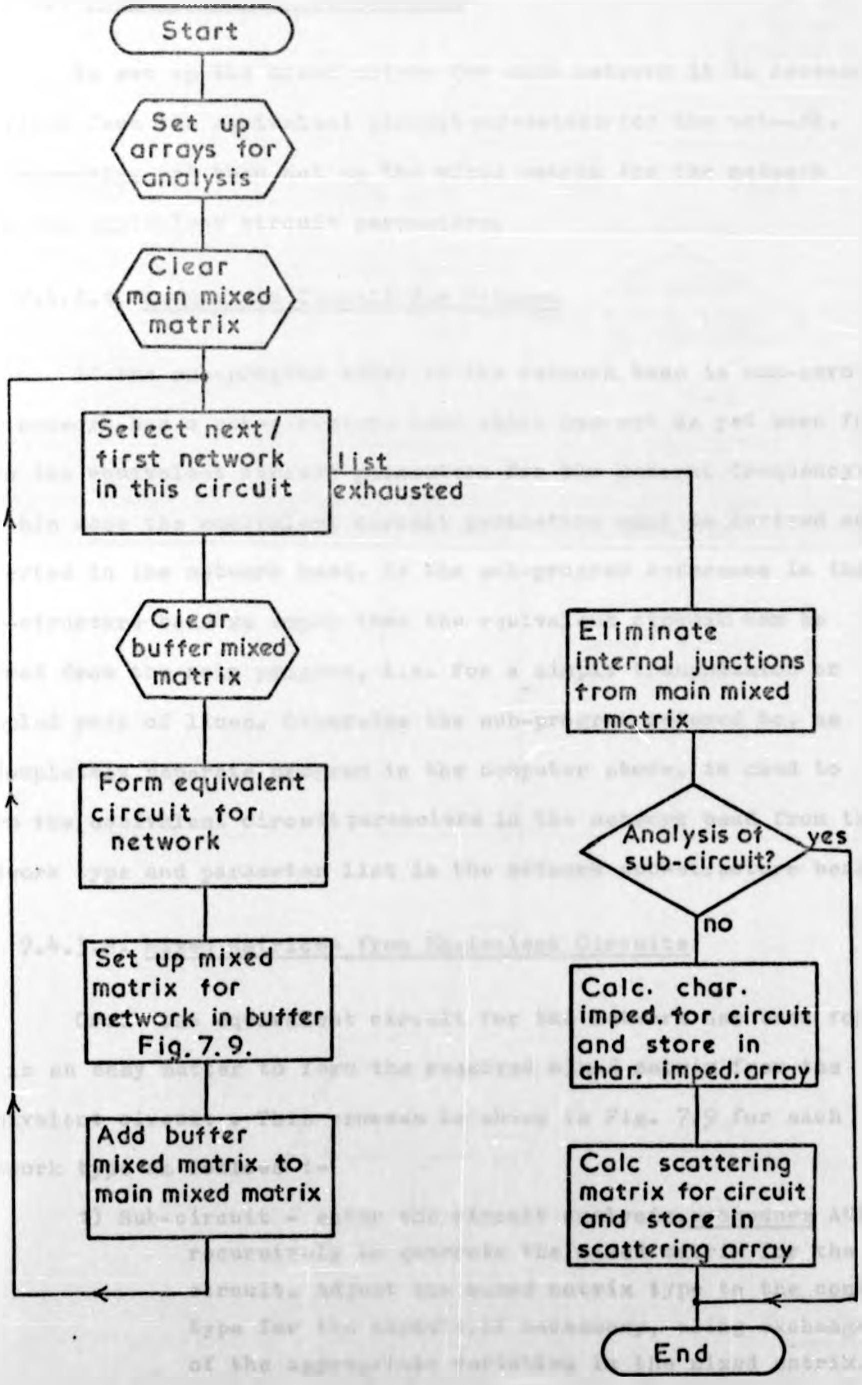

Start
Set up arrays for analysis
Clear main mixed matrix
Select next / first network in this circuit
list exhausted
Clear buffer mixed matrix
Form equivalent circuit for network
Set up mixed matrix for network in buffer Fig. 7.9.
Add buffer mixed matrix to main mixed matrix
Eliminate internal junctions from main mixed matrix
Analysis of sub-circuit?
yes
no
Calc. char. imped. for circuit and store in char. imped. array
Calc scattering matrix for circuit and store in scattering array
End

Fig. 7.8.-Flow Diagram for Circuit Analysis in MICRO3 Program

7.4.3. Mixed Matrices for Networks

To set up the mixed matrix for each network it is necessary to first form the equivalent circuit parameters for the network, if necessary, and then set up the mixed matrix for the network from the equivalent circuit parameters.

7.4.3.1. Equivalent Circuit for Network

If the sub-program entry in the network bead is non-zero then the network has a sub-structure bead which has not as yet been formed into the equivalent circuit parameters for the current frequency. In this case the equivalent circuit parameters must be derived and inserted in the network bead. If the sub-program reference in the sub-structure bead is empty then the equivalent circuit can be formed from the main program, i.e. for a simple transmission or coupled pair of lines. Otherwise the sub-program refered to, as a completely separate program in the computer store, is used to form the equivalent circuit parameters in the network bead from the network type and parameter list in the network sub-structure bead.

7.4.3.2. Mixed Matrices from Equivalent Circuits

Once the equivalent circuit for the network has been formed it is an easy matter to form the required mixed matrix from the equivalent circuit . This process is shown in Fig. 7.9 for each network type as follows :-

1) Sub-circuit - enter the circuit analysis, procedure ANALYS, recursively to generate the mixed matrix for the circuit. Adjust the mixed matrix type to the connection type for the circuit, if necessary, using exchange of the appropriate variables in the mixed matrix.

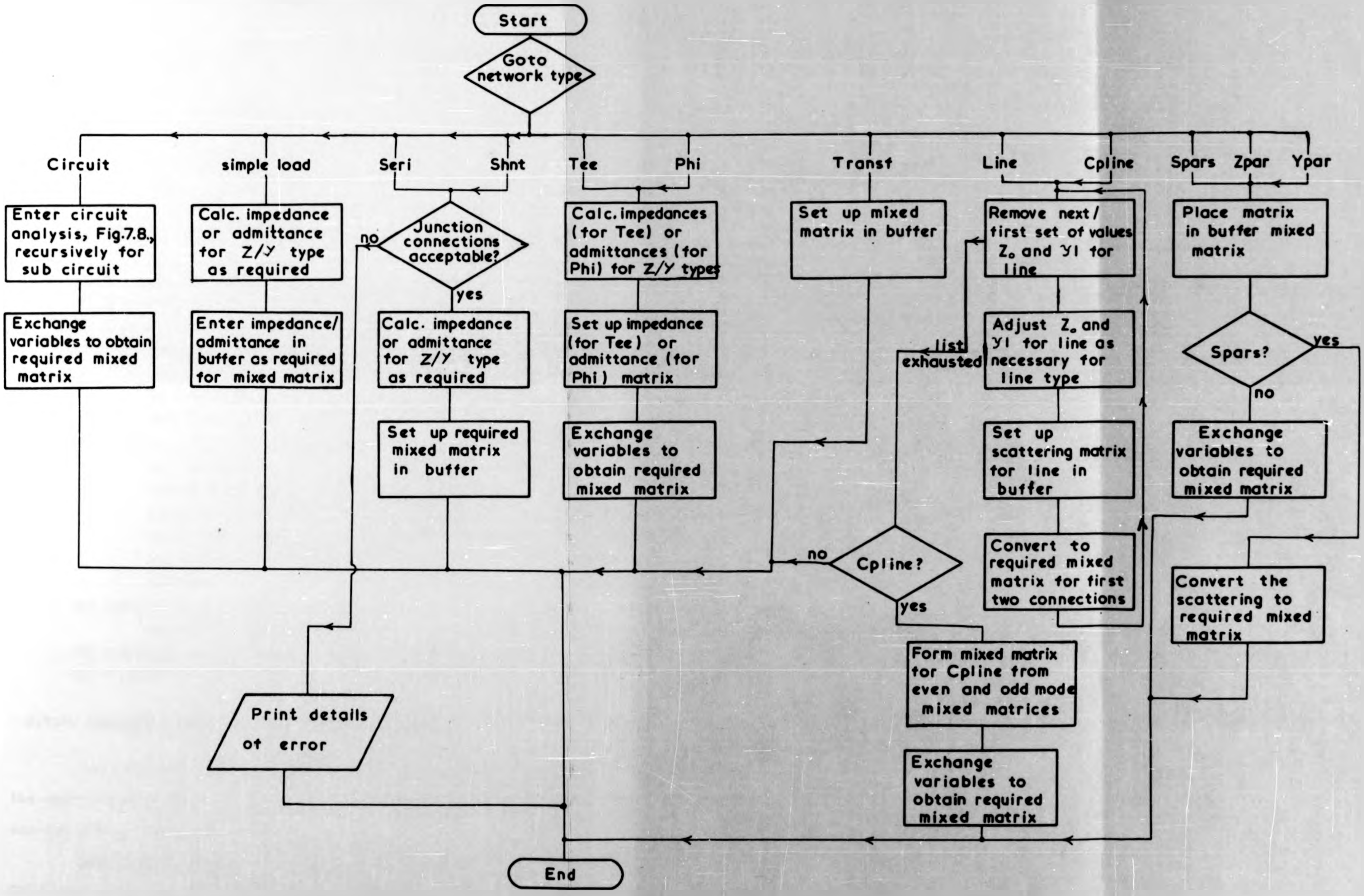

Start
Goto network type
Circuit
simple load
Seri
Shnt
Tee
Phi
Transf
Line
Cpline
Spars
Zpar
Ypar
Enter circuit analysis, Fig.7.8., recursively for sub circuit
Exchange variables to obtain required mixed matrix
Calc. impedance or admittance for Z/Y type as required
Enter impedance/admittance in buffer as required for mixed matrix
Junction connections acceptable?
no
yes
Calc. impedance or admittance for Z/Y type as required
Set up required mixed matrix in buffer
Calc. impedances (for Tee) or admittances (for Phi) for Z/Y types
Set up impedance (for Tee) or admittance (for Phi) matrix
Exchange variables to obtain required mixed matrix
Set up mixed matrix in buffer
Remove next/first set of values Z₀ and Yl for line
list exhausted
Adjust Z₀ and Yl for line as necessary for line type
Set up scattering matrix for line in buffer
Convert to required mixed matrix for first two connections
Cpline?
no
yes
Form mixed matrix for Cpline from even and odd mode mixed matrices
Exchange variables to obtain required mixed matrix
Place matrix in buffer mixed matrix
Spars?
yes
no
Exchange variables to obtain required mixed matrix
Convert the scattering to required mixed matrix
Print details of error
End

Fig. 7.9.-Flow Diagram for Setting up Mixed Matrix for Network in MICRO3 Program

2) Load - set up the impedance/admittance for the load.

3) Series Z/Y - calculate the impedance/admittance of the Z/Y and set up the mixed matrix from this (Table 2.7)

4) Shunt Z/Y - as for Series Z/Y.

5) Tee - set up the impedances for the two Z/Y's in the Tee and from these set up the Z matrix for the Tee (Table 2.7). Exchange variables in the Z matrix to obtain the required mixed matrix.

6) Phi - as for Tee except that the admittances for the two Z/Y's in the Phi are set up and then the Y matrix for the Phi.

7) Transformer - set up mixed matrix as in Table 2.10.

8) Line - adjust the characteristic impedance and propagation constant for the line as necessary, depending on the line sub-type, and set up the wave scattering matrix for the line. Transform the wave scattering matrix into the required mixed matrix using procedure STOZY.

9) Coupled Lines - set up the mixed matrices for the even and odd modes on the lines separately, as for a normal line, using the connection type for the first two junction connections only. Then set up the 4-port mixed matrix from these two mixed matrices and exchange variables to obtain the required mixed matrix.

10) Scattering Parameters - transform to the required mixed matrix (section 2.4.5.1)

11) Z Matrix - convert to the required mixed matrix.

12) Y Matrix - convert to the required mixed matrix.

7.4.4. Variable Elimination and Equation Solution

The elimination of the internal variables, section 3.3, and the solution of a set of linear equations, section 3.2, were both carried out in the program in :-

procedure CMPXEQ(AA,elim,rows,cols,backsb,BB);

This procedure was used for these two applications as described below.

1) Elimination of Internal Variables - the matrix equation for the elimination is shown in equation (3.10) and the procedure CMPXEQ is entered with :-

AA = the identifier for matrix A in equation (3.10) or the array MATRIX in the program, i.e. the full mixed matrix for the circuit

elim = the number of variables to be eliminated in x_i, i.e. the number of internal junctions in the circuit

rows = the number of rows in A, i.e. the total number of junctions in the circuit

cols = the number of columns in A, i.e. the total number of junctions in the circuit

backsb = false to give no back substitution

BB = the identifier of the matrix in which the resulting reduced matrix, A'_{ee} in equation (3.12), must be placed. This is the array MIXZY in the program for the mixed matrix for the circuit.

In the elimination process row exchanges as necessary are performed using the SPR program$^{(7.1)}$ to swap the pointers to pairs of rows in the matrix AA. Also a check is made for zero pivot, row and column multipliers. After the elimination is complete the reduced matrix is transfered into the array BB.

2) Solution of a Set of Linear Equations - the objective is to solve the set of linear equations in equation (3.3) to give the result in equation (3.4). To do this the procedure CMPXEQ is entered with :-

AA = the identifier of the matrix containing both A and B, equation (3.3), in the form $[A \vdots B]$

elim = no. of rows in $[A \vdots B]$

rows = no. of rows in $[A \vdots B]$

cols = no. of columns in $[A \vdots B]$

backsb = true to include a back substitution on all rows of $[A \vdots B]$

BB = the identifier of the array into which the result, equation (3.4), must be placed

7.4.5. Exchange of Variables

The objective of the exchange of variables, section 3.1, is to enable pairs of variables refering to the same junction in a mixed matrix in array BUF to be swapped to obtain the required mixed matrix from another mixed matrix which is simpler to set up. This was carried out in :-

procedure EXCHGE;

entered with :-

stype = connection type for the mixed matrix in array BUF
typcon = the required connection type for the mixed matrix in array BUF
conn = no. of rows/columns in the mixed matrix.

The procedure EXCHGE will look at each bit in the values of 'stype' and 'typcon' in turn for bits 1 to 'conn'. If the values of the two bits are not the same then the procedure will swap the i th variable between the opposite sides of the mixed matrix equation in array BUF for the i th bit in the comparison.

7.4.6. Convert Scattering to Mixed Matrix

To convert from wave scattering to mixed matrix, section 2.4.5.2, :-

procedure STOZY(line);

was used and this could be used in two ways as follows.

1) Convert Wave Scattering to Mixed Matrix - the procedure STOZY is entered with :-

line = false
wave scattering matrix in array BUF
conn = no. of rows/columns in scattering matrix
typcon = required mixed matrix type (bit i = i th connection type)
c(zc,zci) = complex characteristic impedance for scattering matrix (equal for all ports), i.e. the characteristic impedance of the line.

The procedure will then convert the scattering matrix in array BUF into the required mixed matrix in array BUF, section 2.4.5.2.

2) Mixed Matrix for Transmission Line - the procedure STOZY is entered with :-

line = true
conn = 2
typcon = required mixed matrix type
c(zc,zci) = complex characteristic impedance for line
c(att,ang) = complex propagation constant for line

The procedure will then set up the wave scattering mixed for the line and then convert it to the required mixed matrix, section 2.4.6.

7.4.7. Complex Number Procedures

In the MICRO3 program initially all the complex number operations were carried out using long and very slow procedures, Table 7.3, to do even the very simple complex number operations. This enabled the program to be written with ease, tested and checked. Then the complete source code of the program was scaned using suitable instructions written for the ML1 macrogenerator(7.3,7.4) program. This replaced all the complex number procedure calls with the equivalent block of NEAT code using the CMPLEX program to carry out the complex arithmetic. In the program all the complex numbers were held in 4 locations as pairs of real numbers in store as simple identifiers or array elements. In this way the final comlex number operations were fast with little overheads in terms of time or object code. There were some rules to follow as follows :-

For the macrogenerator to be successful :-

1) The real part of every complex number must be an identifier or an identifier followed by a list of array subscripts with each array subscript being an identifier or unsigned integer.

2) All the parameters in a procedure call which are not part of a complex number must be a single identifier.

For the equivalent NEAT code to give the same results :-

1) All the parameters in the procedure calls must be of the type given in the parameter list for the procedure

2) Every complex number must be equivalent to 4 consecutive locations in the computer store.

procedure	result

ZEROJ(a,ja,zero)	zero:=0 if c(a,ja)=c(0.0,0.0)
	zero:=1 otherwise

AMAXJ(a,ja,max)	max= larger of |a| or |ja|
ASSJ(a,ja,b,jb)	c(b,jb):= c(a,ja)
ADDJ(a,ja,b,jb,c,jc)	c(c,jc):= c(a,ja) + c(b,jb)
SUBJ(a,ja,b,jb,c,jc)	c(c,jc):= c(a,ja) - c(b,jb)
INVJ(a,ja,b,jb)	c(b,jb):= c(1.0,0.0)/c(a,ja)
MULTJ(a,ja,b,jb,c,jc)	c(c,jc):= c(a,ja) * c(b,jb)
DIVJ(a,ja,b,jb,c,jc)	c(c,jc):= c(a,ja) / c(b,jb)
MODJ(a,ja,mod)	mod:= |c(a,ja)|
ARGJ(a,ja,arg)	arg:= argument of c(a,ja) in degrees
DBJ(a,ja,db)	db:= modulus of c(a,ja) in dB

n.b. c(x,y) is the complex number represented by the real numbers x and y

Table 7.3. - Complex Number Procedures for MICRO3

7.5. CONCLUSIONS

In the MICRO2 and MICRO3 programs it was found that the mixed matrix analysis was much easier to organise than the chain matrix analysis as a path topological analysis was not required. The problem of setting up the various types of mixed matrices for all the possible n-port networks was not found to be very difficult as it was assumed that all microwave components could be broken down into a small set of equivalent circuits in n-port networks. Problems were met in these programs because in Algol 60 there is no provision for complex numbers. Thus initially long and slow procedures had to be used for the complex number operations and later replaced, in the source code of the program, by blocks on assembly code to make the program run efficiently.

In the MICRO2 and MICRO3 programs a full syntactical analysis of the data was included to make the programs suitable for interactive use, chapter 8. Thus the format of the data was more like a programming language than normal data for a program. The procedures used for this work were translated from the WARDEN package(7.5), which was in Fortran, into Algol with modifications, as necessary, for the needs of the MICRO2 and MICRO3 programs. It may have been better to program the whole or parts of the programs in machine code so that the basic techniques for a full syntax analysis, e.g. Hopgood(7.6), could be used but this was outside the scope of this thesis. Generalised work on syntax analysis is also given in references 7.7 and 7.8.

References

7.1) 4100 Technical Manuals, ICL, Vol. 2 - Programming Information

7.2) G. Bull, 'Computation Methods and Algol', Harrap, 1966

7.3) C. Strachey, 'A General Purpose Macrogenerator', Computer Journal, Vol. 8, 1965-66, pp. 225-241

7.4) 'ML1 Users Manual', University Mathematical Laboratory, Cambridge University

7.5) M.H.E. Larcombe, 'The WARDEN System, A Technique for Data Exchange within a Suite of Programs', Proceedings of International Symposium on Computer Aided Structural Design, Vol. 2, University of Warwick, 10-14 July 1972

7.6) F.R.A. Hopgood, 'Compiling Techniques', MacDonald, 1969

7.7) D.B. Worfman, 'A Compiler Generator', Prentice Hall, 1970, Chapter 4, Section 4

7.8) R.I. Chaplin, R.E. Crosbie, J.L. Hay, 'A Graphical Representation of the Backas-Naur Form', Computer Journal, Vol. 16, No. 1, Feb. 1973, pp. 28-29

Chapter 8

Interactive use of MICRO3 Program

8.1. INTRODUCTION

During this thesis considerable interest was shown by the School of Engineering Science in the use of computer programs in an interactive mode so that the user of the program can use it on-line. Thus the MICRO3 program was designed so it could be used in this mode. The MICRO3 program can be used on batch processing, on-line on a (remote) teletype, or on-line with a (remote) teletype and graphical display.

For batch processing the user could prepare his data on cards or paper tape. The computer would read in the data on batch and produce a copy of the data on the line printed with any errors detected in the position in which they were detected in this data. If no errors had been detected before an analysis or optimisation statement then the analysis or optimisation would be carried out otherwise the rest of the data would be checked for errors only. The results of the analysis or optimisation would then be printed out on the line printer.

In the case of the use of the program on a remote teletype the program was designed so that the user could type in his data directly on the teletype. If he makes any errors in the data then the program will immediately respond by printing out the message <** SYNTAX ERROR -> followed by an error number and the last group of characters read which caused that error to be detected. The user would then have to refer to a printed list of the possible errors, for each syntax or circuit error, supplied to him to decide why that error was detected in the data. He could then type in a new statement in the data to correct that error. Once he thinks that he has sufficient data read into the program he can carry out an analysis or optimisation by typing in the appropriate statement.

If there are no errors in the circuit description or other data then an analysis or optimisation would be carried out and the results printed either on the line printer or remote teletype. Any errors detected at this stage are printed out as the message <** CIRCUIT ERROR -> followed by an error number.

The on-line use of the graphical display is very similar to the use of the program on the remote teletype except that the display must be placed next to the remote teletype and the MIC3D version of the program used. This program will store the results of an analysis on a disc file and later, after an analysis, a statement to transfer control to the graphical display must be typed in on the remote teletype. Then the light pen can be used on the graphical display to select various results to be plotted in graphical form on the display from the last set of results stored from an analysis.

In general, in on-line work, it is very easy for the user to enter an initial circuit design, analyse it, display the results of the analysis on the graphical display and then, if the circuit performance is not satisfactory, to return to the teletype to modify the circuit to try and improve its performance.

8.2. TELETYPE ON-LINE

8.2.1. Objective

The objective in writing a general computer program for use interactively on-line on a remote teletype could be summerised as follows :-

1) Versatility - the program should be able to handle as wide a range of problems as possible, or practical, within the field defined for the application of the program.
2) Data Errors - all possible data errors should be detected and trapped in the program and the details of these errors printed out so the user can correct them.
3) Error Correction - it must be possible to correct any data errors on-line after they have been detected by the program so that the program run can be continued.
4) Data Modification - it must be possible to change the data stored by additions, deletions or modifications and re-analyse the circuit as many times as desired.

8.2.2. Data Processing

For the MICRO3 program the syntax analysis, section 7.2, and the data processing, section 7.3, have already been described and this program is already well prepared for interactive use. The facilities available in the MICRO3 program already described include, for interactive use, the following :-

1) The syntax analysis will detect any syntactical error in the data and print out the message <** SYNTAX ERROR > followed by an error number and the last group of characters read from the data.
2) The data will not be corrupted by any error in the data.
3) Further statements in the data can add to, overwrite, delete or correct the data stored from previous statements.

4) After a syntax error has been detected in the data the program will search for the end of the current statement in the data before starting to process the next statement in the data.
5) Before an analysis or optimisation the circuit topology to be analysed and the additional data stored for the analysis or optimisation is checked to ensure that it has been sufficiently and correctly defined for the analysis or optimisation. Any errors thus detected, or previous syntax errors uncleared, will cause the analysis or optimisation to be ignored.
6) The statements in the data can occur in almost any order and an analysis or optimisation performed at any point.
7) Any number of completely separate circuit topologies can be defined in the data and stored and analysed as desired at any point in the data. Also any circuit can be used as a sub-circuit of any other circuit.
8) Facilities are available in the syntax analysis to erase the last character group read in the data or to abandon the processing of the current statement in the data.

The use of the MICRO3 program on a remote teletype is not exactly the same as on batch processing. In the use of the remote teletype for the program, e.g. Table 8.1, the data starts with the message <** DATA> printed out on the teletype and a <↑> on a new line to request more data to be typed in by the user. The user must then type in a line of data, as defined by the syntactical definition of the data in section 7.3.2. If he makes a syntactical error in the data then the program will print out the message <** SYNTAX ERROR-> followed by an error number and the last group of characters read from the data. The processing of the current statement in the data will be abandoned and the program will print a <↑> on a new line to request the next line of data to be typed in. If an error is made in typing a character group then it may be erased by typing a <%> immediately following it. Also, if

```
** DATA
↑ LABEL EXAMPLES OF SYNTAX AND CIRCUIT ERRORS IN DATA ;
↑ NOT
** SYNTAX ERROR    2 - NOT
↑ NET 1 SERI SRLC 4.5 0.1E-9 JUNCT
** SYNTAX ERROR   15 - JUNCT
↑ NET 1 SERI SRLC  4.5  0.1E-9  5.6E-12  JUNCT  2  3 ;
↑ NET 2 LINE WL  50.0  0.3  5E9  0.1  JUNCT  1  2 ;
↑ NET 3 LINE WL  40.0  0.15  5E9  0.2  JUNCT  3  4 ;
↑ PORTS 1 4 ;
↑ ANALYSE ;
** CIRCUIT ERROR   2
** CIRCUIT ERROR  21
** CIRCUIT ERROR  22
** CIRCUIT ERROR  23
↑ ZO 1 SRX 40 ;
** SYNTAX ERROR   48 - ;
↑ ZO 1 SRX 40 25 ;
↑ ZO 2 R 60 ;
↑ JUNCT PARALLEL 1 2 3 4 5 ;
** SYNTAX ERROR  34 - 5
↑ FREQ  5.5E9  STOPLIN
** SYNTAX ERROR  52 - STOPLIN
↑ FREQ  5.5E9  STEPLIN 1E9 10E9  STEPLOG
** SYNTAX ERROR  55 - STEPLOG
↑ FREQ  5.5E9  STEPLIN 1E9 10E9 9  STEPLOG 1E6 1E12 6  1.2E9  2.2E9 ;
↑ OUTPUT  ZIN CMPX
** SYNTAX ERROR  63 - CMPX
↑ OUTPUT  ZIN 1 CMPX  ZIN 2 CMPX  SPAR 2 1 DBARG  VSWR 1 ;
↑ NOFAULT;  ANALYSE ;
** ANALYSIS
** DATA
↑ END ;
** RUNEND
```

Table 8.1. - Data Checking in the Interactive MICRO3 Program

desired,the processing of the current statement can be abandoned at any time by typing <%%> in the data.

8.2.3. Channel Assignment

In the MICRO3 program 3 channels were used for the input and output streams for the program, excluding work files, and these were :-

data channel (input only)
message channel (output only)
result channel (output only)

In a versatile program for interactive use it is necessary to be able to assign these channels to any suitable channel number, including disc files, during the running of the program. Typical channel assignments for various modes of use of the program are shown below.

1) Batch Processing

data channel = card or paper tape reader
message channel = line printer
result channel = line printer

2) Partial Interactive Work

data channel = file containing the data
message channel = remote teletype
result channel = line printer

The paper tape or card reader could be used for the data channel but for interactive work, on a time sharing computer, it is not normally convenient to tie up either of these two channels for too long. The advantage with a disc file is that it can be used as many times as desired and editted as required. If a graphical display is available then this could be used next to the remote teletype with the MIC3D version of the MICRO3 program.

3) Full Interactive Work

data channel = remote teletype
message channel = remote teletype
result channel = remote teletype

This method of interactive work is very useful to the user because he can see his results printed out in tabular form on the teletype after an analysis. The disadvantage of this method is that it takes a long time to type in the data by hand and a long time to print out the results on the teletype. During all this time the computer, and possibly a telephone line, would have to be connected to the remote teletype.

In the MICRO3 program the data, message and result channel can be assigned to any suitable channel but it is necessary to detect, in the program,the way in which the program is being used and print out, on the various channels, appropriate information. The points included in the syntax analysis to include all these points are :-

1) A record of the input data is printed on the result channel provided this is not the same as the data channel.
2) All the syntax and circuit error messages are printed out on the message and result channel but only on one channel if both of these are on the same channel.
3) After a syntax error has been detected the program will set the fault indicator and start processing the next statement in the data. If the data and message channels are not the same, i.e. for batch work, the program will skip the data up to and including the current statement terminator, i.e. a <;> . Otherwise the program is being used interactively and it will print a <↑> on a new line on the message channel to request the next statement to be typed in.

4) If the message and results channel are not the same, i.e. not for full interactive work, then a <↑> is printed on the message channel as each line feed is read from the data. This will indicate that a line of data has been processed or as a request for the next line of data to be typed in.

5) Various messages are printed out on the message channel during a program run, section A.5.12, and these indicate to the user information as to the running or state of the program.

8.3. USE OF GRAPHICAL DISPLAY

8.3.1. Use of Display

The version of the MICRO3 program to use the graphical display is the MIC3D program which can be obtained by using an edit stream to edit the MICRO3 program source code into the source code for the MIC3D program. This edit stream will add, to the MICRO3 program, code to do the following :-

1) Store the results of a circuit analysis on a work file on disc (channel 25).
2) Add a statement to the syntactical definition of the data to transfer control to the Visual Display segment of the program,
3) Add a complete Visual Display segment to the program to read the results of the last analysis stored on the disc work file and plot graphs, as required, from them on the graphical display.

During a program run on the MIC3D program to display the results of an analysis on the graphical display it is first necessary to carry out an analysis on the circuit in question using a normal analysis statement. Then the statement <transfer to display> , i.e. the statement <DISPLAY ;> must be typed in on the teletype. Then control of the program is entirely from the lightpen or keys on the graphical display.

The control of the Visual Display segment of the program by the user is by the use of menus on the screen of the display to which the user must point the light pen to the desired item in the menu or press a key equal in number to the position of the desired item in the menu list. The menus, if applicable, are displayed in order, Table 8.2, with the response for each word selected from the menu shown in this table.

Menu no.	Words in menu	Response	Next menu
1	DATA	transfer to Read Data segment	teletype
	PLOT	plot new graph	2
	PLOT ADD	add another graph to present ones	3
	DUMP	dump graphs on digital plotter	1
		n.b. a <.> must be typed on the teletype to accept the dump	
2	RECT	plot graph on cartesian grid against frequency	3
	POLAR	plot graph on polar grid of complex number over frequency range	3
	SMITH	plot graph on smith chart of scattering parameter (SPAR) over frequency range	4
3	ZY	plot element in mixed matrix	4
	SPAR	plot power scattering parameter	4
	ZO	plot characteristic impedance	5
	ZIN	plot input impedance	5
	YO	plot characteristic admittance	5
	YIN	plot input admittance	5
	VSWR	plot V.S.W.R. (RECT grid only)	5
4	1 2 etc.	select port number for option to be plotted	5
5	1 2 etc.	select port number for option to be plotted	6
6	n.b. menu 6 omitted for POLAR/SMITH grid or VSWR option		7
	REAL	plot real part	7
	IMAG	plot imaginary part	7
	MOD	plot modulus	7
	ARG	plot argument (degrees)	7
	DB	plot modulus (dB) (SPAR only)	7
7	empty	plot graph on screen or add to graphs at present on screen	1

Table 8.2. - Menus used on Graphical Display for MIC3D program

After the grid type and graph to be plotted has been selected from the set of menus, Table 8.2, the required graph, with automatic scaling of the grid, is plotted on the display. Also included is the title of the graph, i.e. the graph no. followed by the option being plotted, and the label string from the data given in the last <label statement> . After this the program returns to menu 1, Table 8.2, where the user may return control to the teletype, plot another new graph, add another graph onto the same grid (up to 5 graphs only) or dump the graphs at present on the display onto the digital plotter to obtain a hard copy of the graphs. If the DUMP option is chosen then first the message <**> is typed on the teletype after which the user must type in a <.> . Then the message <** DISPLAY DUMP> is typed out and the message <** END DUMP> when the dump is complete.

8.3.2. Results on Graphical Display

Plate 8.1 shows the graphical display next tc the remote teletype and Plate 8.2 shows the graphical display in use. Plates 8.3, 8.4 and 8.5 show the results of an analysis on a phase shifter on a rectangular grid, smith chart and polar grid respectively. Plates 8.6 and 8.7 show the results of an analysis of a non-contacting coaxial short circuit on a rectangular and smith chart grid respectively. On 8.3 to 8.7 the format on the display consists of a label string at the top of the picture. Below this is a list of up to 5 graphs at present displayed. For the graphs the scales are automatically set for the graphs (except for a smith chart) and a number is printed on the end of each graph to indicate its number. The menus are displayed on the right hand side of the screen.

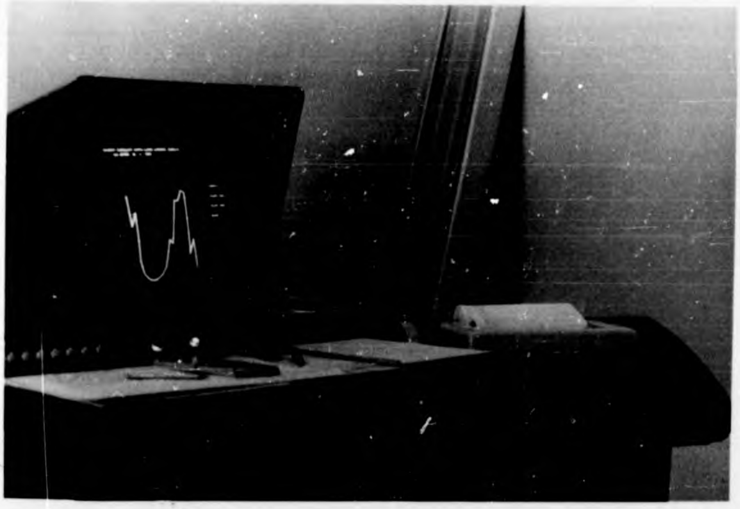
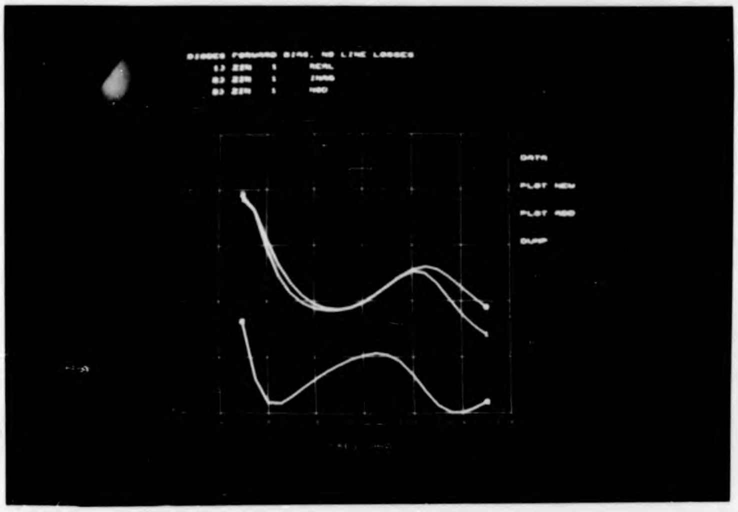


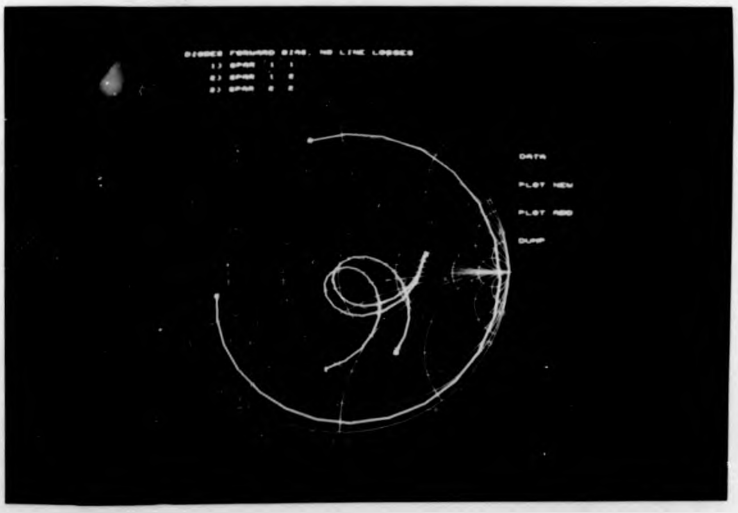
Plate 8.1. - Graphical Display and Remote Teletype

Plate 8.2. - Graphical Display and Remote Teletype in Use



```
DIODES FORWARD BIAS, NO LINE LOSSES
    1) ZIN  1     REAL
    2) ZIN  1     IMAG
    3) ZIN  1     MOD
```

Plate 8.3. - Phase Shifter Results of Input Impedance ZIN in REAL, IMAG and MOD on Rectangular Grid



```
DIODES FORWARD BIAS, NO LINE LOSSES
    1) SPAR  1  1
    2) SPAR  1  2
    3) SPAR  2  2
```

Plate 8.4. - Phase Shifter Results of Scattering Parameters 11, 12 and 22 on a Smith Chart

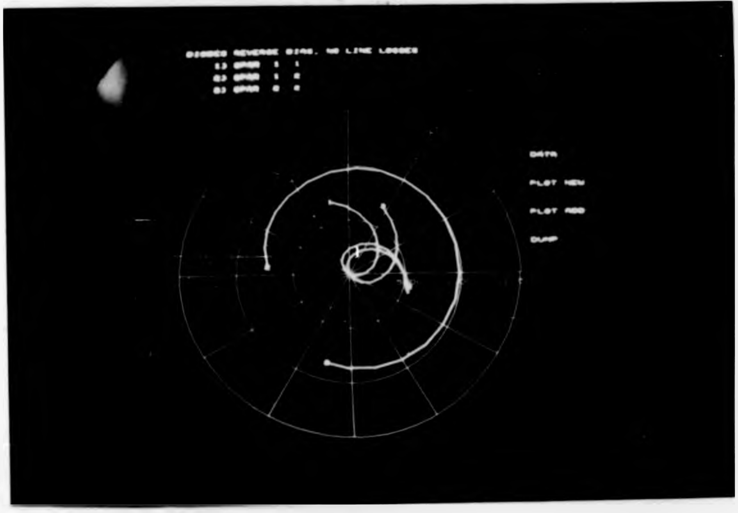


Plate 8.5. - Phase Shifter Results of Scattering Parameters 11, 12 and 22 on a Polar Grid

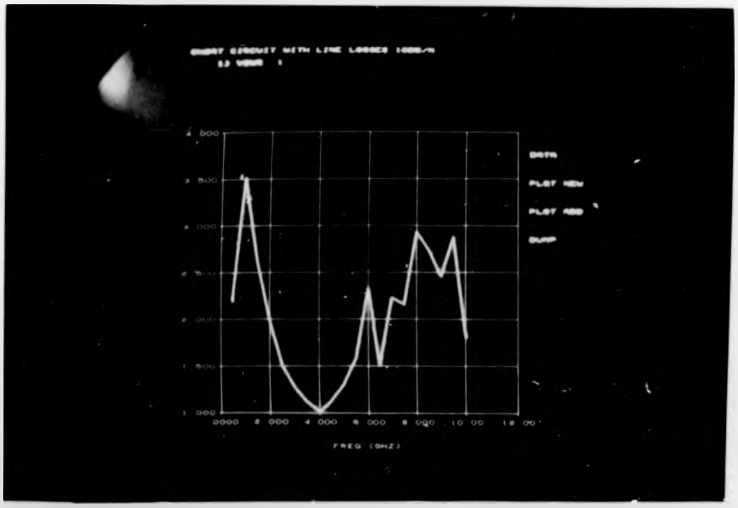


Plate 8.6. - Short Circuit Results of Input VSWR on a Rectangular Grid

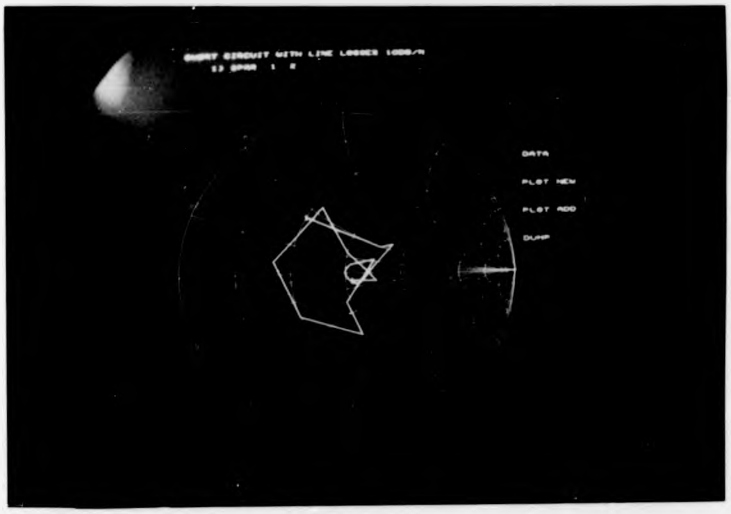


Plate 8.7. - Short Ci~~r~~cuit Results of Scattering Parameter 11 on a Smith Chart

8.4. THE COMPUTER FOR INTERACTIVE WORK

The type of use of the computer during interactive use of the computer for circuit analysis are as follows :-

1) Very little processor time but a large amount of real time is used whilst the user types in his data.
2) The processor is used very efficiently whilst the computer carries out the circuit analysis.
3) No processor time is used, but a large amount of real time is used, whilst the user decides what to do next.
4) Go back to 1).

On interactive work it is quite common for the user to use one hour or more on the computer in terms of real time but possibly less than one minute or less processor time. Thus it is absolutely essential that the computer can do other work at the same time. Otherwise interactive use of a computer would be out of the question for this type of work due to the excessive cost.

To summarise, the main points in selecting a suitable computer for interactive work are :-

1) Possible use in a multi-programming mode so the central processor can do other work at the same time.
2) Possible use in a multi-access mode where the core store can be used by other uses at the same time.
3) Dynamic allocation of core store as required or as demanded by each user.
4) A good executive program to assist in the writing of interactive programs.
5) A computer programming language which is easy to use, e.g. Algol or Fortran, with additional facilities (in fast and efficient procedures) for syntax analysis, channel allocation, data structure, real time program interrupts, etc..
6) Immediate response from a remote terminal.
7) If a telephone line has to be paid for then its cost must be minimised.

The computer used at the University of Warwick during the period of the research was an Elliott 4130, used for interactive work on the DES2 time sharing multi-programming system. This was better than the use of a dedicated machine for interactive work but was far from ideal. The main problems were :-

1) A slave of a given core size and a number of peripheral devices had to loaded manually for the interactively work.
2) The core size of the slave could not be changed once loaded.
3) A multi-programming mode was possible but not a multi-access mode.
4) The executive program for the system was not designed for interactive programming on a remote teletype.
5) A suitable interactive programming language was not available for this machine.

During the final year of this thesis the SIGMA5 computer became available in the School of Engineering Science. This computer, if it had more core store, would be more suitable for interactive work but there was not time in the course of this research work to look into this. Also there is more and more interest in the use of computers for interactive work as time proceeds and to meet this demand better and better computers will be developed for this application.

Many computers now have visual displays of some kind. The one on the Elliott 4130 was a refresh type of visual display where the picture has to be continually redrawn and a display file has to remain in the computer core store all the time the display was being used. This type of display was convenient to use because a light pen could be used but it is very expensive to use in terms of running costs. Some computers are now supplied with storage tube displays, e.g. the SIGMA5, where the picture only has to be drawn once. The advantage of this is that the computer can be used in a multi-access mode and crosshairs and joystick may fulfill the function of the light pen.

Chapter 9

Microwave Components in Circuit Analysis Program

9.1. INTRODUCTION

In this thesis, so far, only R, L, C, transmission lines and coupled lines have been included in the computer programs for the Computer Aided Design of Microwave Circuits. It could hardly be said that these components form the basis for a true microwave circuit. In practice microwave circuits may include a wide range of microwave components, e.g. waveguides of various types, circulators, isolators, aerials, etc., and the components so far included in the programs could only be used as an equivalent circuit for these components often at one frequency point only. In practice the number of microwave components which could be used in a microwave circuit run into many hundreds and each component has a complex set of equations to derive its equivalent circuit with many of them including elliptic integrals, bessel functions, summations of series to infinity, etc.. Even though research work to include a number of these components in a computer program would be extremely useful there is not sufficient time in this thesis to look into this problem except in so far as to suggest how it could be tackled.

The MICRO3 program was designed for its possible extension for microwave components, of a complex nature, by designing the bead structure in the data structure for each network, section 6.3.3, in two parts as follows :-

1) The main network bead holding the data and connections for the equivalent circuit, as an n-port network, for the network.
2) The network sub-structure bead holding the data for the microwave component, e.g. the dimensions of the component.

Thus, during an analysis, the data for the microwave component in

9.1. INTRODUCTION

In this thesis, so far, only R, L, C, transmission lines and coupled lines have been included in the computer programs for the Computer Aided Design of Microwave Circuits. It could hardly be said that these components form the basis for a true microwave circuit. In practice microwave circuits may include a wide range of microwave components, e.g. waveguidesof various types, circulators, isolators, aerials, etc., and the components so far included in the programs could only be used as an equivalent circuit for these components often at one frequency point only. In practice the number of microwave components which could be used in a microwave circuit run into many hundreds and each component has a complex set of equations to derive its equivalent circuit with many of them including elliptic integrals, bessel functions, summations of series to infinity, etc.. Even though research work to include a number of these components in a computer program would be extremely useful there is not sufficient time in this thesis to look into this problem except in so far as to suggest how it could be tackled.

The MICRO3 program was designed for its possible extension for microwave components, of a complex nature, by designing the bead structure in the data structure for each network, section 6.3.3, in two parts as follows :-

1) The main network bead holding the data and connections for the equivalent circuit, as an n-port network, for the network.
2) The network sub-structure bead holding the data for the microwave component, e.g. the dimensions of the component.

Thus, during an analysis, the data for the microwave component in

9.1. INTRODUCTION

In this thesis, so far, only R, L, C, transmission lines and coupled lines have been included in the computer programs for the Computer Aided Design of Microwave Circuits. It could hardly be said that these components form the basis for a true microwave circuit. In practice microwave circuits may include a wide range of microwave components, e.g. waveguidesof various types, circulators, isolators, aerials, etc., and the components so far included in the programs could only be used as an equivalent circuit for these components often at one frequency point only. In practice the number of microwave components which could be used in a microwave circuit run into many hundreds and each component has a complex set of equations to derive its equivalent circuit with many of them including elliptic integrals, bessel functions, summations of series to infinity, etc.. Even though research work to include a number of these components in a computer program would be extremely useful there is not sufficient time in this thesis to look into this problem except in so far as to suggest how it could be tackled.

The MICRO3 program was designed for its possible extension for microwave components, of a complex nature, by designing the bead structure in the data structure for each network, section 6.3.3, in two parts as follows :-

1) The main network bead holding the data and connections for the equivalent circuit, as an n-port network, for the network.
2) The network sub-structure bead holding the data for the microwave component, e.g. the dimensions of the component.

Thus, during an analysis, the data for the microwave component in

in the network sub-structure bead is first translated into the data for the equivalent circuit in the main network bead. Then the circuit analysis is carried out entirely on the equivalent circuit of the microwave circuit.

It is possible to include blocks of code in the program to tranform the microwave components into their equivalent circuits but the number of microwave components would soon make the program far to large to handle and most of the program, about 90%, would not be used for a given circuit. Thus it was decided that the code to generate these equivalent circuit• would be best contained in separate sub-programs and only the sub-programs required for a given microwave circuit would be loaded into the computer core store.

9.2. SUB-PROGRAMS FOR MICROWAVE COMPONENTS

9.2.1. General Principles

The sub-programs to translate a microwave component into its equivalent circuit should be supplied with 3 possible entry points as follows :-

Entry

0 check if the network sub-type is in the program and determine the sizes required for the network bead structure for this microwave component

1 calculate the parts of the equivalent circuit which are not a function of frequency

2 complete the equivalent circuit for a point frequency

The equivalent circuit for a given microwave component could be independant of frequency or it could be obtained only from a very complex set of equations some parts of which may be independant of frequency. Thus it is best to calculate the parts of the equivalent circuit which are independant of frequency once only, particularly if they are complex functions.

In the MICRO3 program some experimentation with sub-programs was carried out, with all the programs in Algol, but Algol on the Elliott 4130 computer was never designed for such an application and the results were not satisfactory. It took a long time to 'fiddle' the use of Algol, with blocks of NEAT, to make this work for some cases.

9.2.2. Entry 0 - Check Network Type

During the syntax analysis of the data in the MICRO3 program it is possible that a network type sub-statement, section 7.3.2, may have a network type word that is not acceptable within the

9.2. SUB-PROGRAMS FOR MICROWAVE COMPONENTS

9.2.1. General Principles

The sub-programs to translate a microwave component into its equivalent circuit shouldbe supplied with 3 possible entry points as follows :-

Entry	
0	check if the network sub-type is in the program and determine the sizes required for the network bead structure for this microwave component
1	calculate the parts of the equivalent circuit which are not a function of frequency
2	complete the equivalent circuit for a point frequency

The equivalent circuit for a given microwave component could be independant of frequency or it could be obtained only from a very complex set of equations some parts of which may be independant of frequency. Thus it is best to calculate the parts of the equivalent circuit which are independant of frequency once only, particularly if they are complex functions.

In the MICRO3 program some experimentation with sub-programs was carried out, with all the programs in Algol, but Algol on the Elliott 4130 computer was never designed for such an application and the results were not satisfactory. It took a long time to 'fiddle' the use of Algol, with blocks of NEAT, to make this work for some cases.

9.2.2. Entry 0 - Check Network Type

During the syntax analysis of the data in the MICRO3 program it is possible that a network type sub-statement, section 7.3.2, may have a network type word that is not acceptable within the

MICRO3 program itself. In this case it is necessary to see if this network type is contained in a sub-program. For this case the format for the network type in the data would consist of 2 words as follows :-

<net type word(s)> ::= <sub-prog name><net sub-type>

In this case the MICRO3 program would check to see if the program with the name <sub-prog name> is in the computer core store or can be loaded into the core store. If it is then this program is then entered at entry 0,with the data structure set as in Table 9.1, to see if the <net sub-type> is acceptable for a network type within this program. If it is then the sub-program will return to the main MICRO3 program giving the data, Table 9.1, for the bead structure for the network to be set up and the parameters for the network to be read into the sub-structure bead, Fig. 7.5.

Address in array IST	Entry to sub-program	Return form sub-program
10	ref. to sub-program	network type no.
11		no. of parameter for network
12	network sub-type word	equivalent circuit type no.
13		no. of junction connections
14	0	no. of parameters for equivalent circuit (if ≤0 then syntax error)

Table 9.1. - Data Structure for Entry 0 to Microwave component Sub-program

MICRO3 program itself. In this case it is necessary to see if this network type is contained in a sub-program. For this case the format for the network type in the data would consist of 2 words as follows :-

<net type word(s)> ::= <sub-prog name><net sub-type>

In this case the MICRO3 program would check to see if the program with the name <sub-prog name> is in the computer core store or can be loaded into the core store. If it is then this program is then entered at entry 0,with the data structure set as in Table 9.1, to see if the <net sub-type> is acceptable for a network type within this program. If it is then the sub-program will return to the main MICRO3 program giving the data, Table 9.1, for the bead structure for the network to be set up and the parameters for the network to be read into the sub-structure bead, Fig. 7.5.

Address in array IST	Entry to sub-program	Return form sub-program
10	ref. to sub-program	network type no.
11		no. of parameter for network
12	network sub-type word	equivalent circuit type no.
13		no. of junction connections
14	0	no. of parameters for equivalent circuit (if ⩽0 then syntax error)

Table 9.1. - Data Structure for Entry 0 to Microwave component Sub-program

9.2.3. Entry 1 - Intermediate Results for Equivalent Circuit

The objective of this entry to the sub-program for a microwave component is to calculate, from the parameters in the sub-structure bead, the parts of the equations for the equivalent circuit for the microwave component which are not a function of frequency. This entry to the sub-program could be used in two ways as follows :-

1) Directly for the Equivalent Circuit - If the equivalent circuit parameters are not a function of frequency then the equivalent circuit parameters would be derived directly and stored in the main network bead. The sub-program entry in the bead structure for the network would then be set to zero. The equivalent circuit would not then be derived again unless a component value in the network sub-bead is changed.

2) Indirectly for the Equivalent Circuit - If the functions for the equivalent circuit are a complex function of frequency then this entry point in the sub-program could be used to derive the parts of these functions which are not a function of frequency and store them in the network main or sub-structure. After this the sub-program entry in the bead structure is set to 2 so that these parts of the functions for the equivalent circuit are not calculated again. Finally entry 2 in the sub-program is also entered.

The setting of the Data structure before an entry to the sub-program for entry 1 and 2 is shown in Table 9.2.

9.2.4. Entry 2 - Complete Equivalent Circuit

This entry to the sub-program is used to complete the equivalent circuit after entry 1 (part 2 in section 9.2.3) in the main network bead. The parts of the equivalent circuit which are only applicable at one frequency point will be derived in this part of the sub-program.

Address in array IST	Entry to sub-program	Return from sub-program
10	reserved for optimisation	reserved for optimisation
11		
12		
13	address of equivalent circuit parameters	
14	address of microwave component parameters	

Table 9.2. - Data Structure for Entry 1 and 2 to Microwave Component Sub-program

9.3. PROBLEMS OF INCLUDING MICROWAVE COMPONENTS

During the course of the reseach work some time was spent on investigating the theory of microwave components and considering how a wide variety of microwave components could be included in a circuit analysis computer program including optimisation. The main difficulty involved was the complexity of the theory of some of the microwave components and the equations for their equivalent circuit parameters. Often the equations would take a large amount of core store in the computer and a large amount of computer time to evaluate often involving summations to infinity, bessel functions, elliptic integrals, etc.. All these functions would have to be programmed and entered in the computer program for analysis. Often it would take longer in computer time to derive the parameter values for an equivalent circuit for a microwave circuit than to analyse that equivalent circuit. In practice there may be several ways to derive the component values for an equivalent circuit e.g. a rough equation to give a fast but approximate value, a complex equation to give an accurate value but it will take a long time to evaluate, a relaxation method to solve a field pattern in a component which will take a very long time to evaluate, etc.. Thus it may be necessary to select a suitable method considering that most users would expect a computer program to give a very accurate result but remembering that these users would not be willing to pay too much for an analysis.

In a computer program for the analysis of microwave circuits it is necessary to derive the equivalent circuit from the physical dimensions and properties of the microwave component. Thus the program would have to check these items to ensure they are acceptable and that the equations for the equivalent circuit are correct for the order of these parameters. Often an equivalent circuit component values will only be correct at one frequency point or over a limited frequency range. Thus it would be necessary to check the validity of the equivalent circuit at each frequency point and to generate a new equivalent circuit as required for each new frequency point.

In practice the designer of a microwave circuit may first design a rough approximation to the circuit he requires. Then he would analyse the circuit to give its performance and then adjust the equivalent circuit parameters, either manually or automatically in an optimisation, to try and improve the circuit performance. He would then try and set the dimensions of his microwave components to give this new equivalent circuit. This method would be fast in terms of computation time but it would be necessary to include procedures in the program for the generation of the microwave component dimensions from the equivalent circuit as well as the generation of the equivalent circuit from the microwave component dimensions.

The result of the preliminary work done on including microwave components in an analysis program was to realise the complexity of the problems involved. Unfortunately there was not sufficient time in the course of the research work to complete this work as it would have taken at least a year to complete. In fact it could constitute a Ph. D. thesis on its own. In section 9.4 a brief outline of the microwave components which could be included in microwave component sub-programs has been given with some of the problems involved. The techniques for writing these sub-programs were investigated but no sub-programs were actually written.

9.4. MICROWAVE COMPONENT SUB-PROGRAMS

9.4.1. Rectangular Waveguide, Program WGRECT (Table 9.3)

The basic theory behind rectangular waveguide is well described by Collin(9.10)p191 and Marcuvitz(9.8) p57. The important point to note is that in general the wave impedance and propagation constant for the guide are both complex numbers to include losses in the walls and dielectric for the waveguide. Recently waveguides have been used below cut-off in the evanescent mode, Craven(9.22) and thus it is necessary to provide an equivalent circuit for the waveguide which is applicable below as well as above cut-off. A waveguide section would normally be considered as an equivalent length of transmission line and it is necessary to define some characteristic impedance for this line. This value is fairly arbitary provided it is consistant for all sections of the waveguides in the circuit or if suitable transformations between sections are included. It may be convenient to supply facilities for higher order modes in the rectangular guide, above H_{10}, but these would rarely be used.

There are numerious discontinuities used in rectangular waveguide, e.g. step in guide height for transformers, windows or posts to form filters, etc., and it is necessary to supply equivalent circuits for these. These discontinuities are described by Marcuvitz(9.8) and their equations derived. The boundary conditions for these discontinuities can usually be satisfied by the summation of a number of higher order modes. The discontinuity equations are complex but only involve usually about 3 or 4 terms in the summation series.

Network sub-type	Network	Ref.	Para-meters	No. of conns.	Equiv. cit. type
H10	H10 propagation in lossless guide	9.10 p189	a,b,l,ϵ_r	2	LINE
H10L	H10 propagation with losses in guide walls	9.10 p189	a,b,l,ϵ_r, σ	2	LINE
H<m><n>	H<m><n> propagation in lossless guide	9.10 p189	a,b,l,ϵ_r	2	LINE
E<m><n>	E<m><n> propagation in lossless guide	9.10 p189	a,b,l,ϵ_r	2	LINE
ESTEPS	E plane STEP in guide Symmetric case	9.8 p307	a,b,b'	1	C
ESTEPA	E plane STEP in guide Asymmetric case	9.8 p308	a,b,b'	1	C
HSTEPS	H plane STEP in guide Symmetric case	9.8 p296	a,b,a'	1	L
HSTEPA	H plane STEP in guide Asymmetric case	9.8 p298	a,b,a'	1	L
CAPWF2	CAPacitive Window thin of 2 obstacles	9.8 p218	a,b,d',d	1	C
CAPWF1	CAPacitive Window thin of 1 obstacle	9.8 p219	a,b,d'	1	C
CAPOFS	CAPacitive Obstacle thin Symmetric	9.8 p221	a,b,d'	1	C
INDWFS	INDuctive Window thin Symmetric	9.8 p221	a,b,d'	1	L
INDWFA	INDuctive Window thin Asymmetric	9.8 p224	a,b,d'	1	L
INDOFS	INDuctive Obstacle thin Symmetric	9.8 p227	a,b,d'	1	L
CAPWT2	CAPacitive Window Thick of 2 obstacles	9.8 p248	a,b,d',l	2	PHI C L
CAPWT1	CAPacitive Window Thick of 1 obstacle	9.8 p252	a,b,d',l	2	PHI C L
CAPOTS	CAPacitive Obstacle Thick Symmetric	9.8 p252	a,b,d',l	2	PHI C L
INDWT2	INDuctive Window Thick of 2 obstacles	9.8 p255	a,b,d',l	2	TEE L C

Continued on next page.

Continued from last page.

Network sub-type	Network	Ref.	Parameters	No. of conns.	Equiv. cit. type
INDWT1	INDuctive Window Thick of 1 obstacle	9.8 p256	a,b,d',l	2	TEE L C
INDOTS	INDuctive Obstacle Thick Symmetric	9.8 p257	a,b,x,d	2	TEE L C

N.B. 1)In all cases except for H<m><n> or E<m><n> H_{10} propagation is assumed in the waveguide.

2)For H10 or H10L components the characteristics of the waveguide below cutoff may be included.

Table 9.3. - Possible Network Sub-types for Rectangular Waveguide Sub-program, WGRECT

Network sub-type	Network	Ref.	Parameters	No. of conns.	Equiv. cit. type
H<m><n>	H<m><n> propagation in lossless guide	9.10 p197	R,l,ϵ_r	2	LINE
E<m><n>	E<m><n> propagation in lossless guide	9.10 p197	R,l,ϵ_r	2	LINE
H<m><n> L	H<m><n> propagation with losses in guide walls	9.10 p197	R,l,ϵ_r, σ	2	LINE
E<m><n> L	E<m><n> propagation with losses in guide walls	9.10 p197	R,l,ϵ_r, σ	2	LINE
INDWFA	INDuctive Window thin and Annular, H_{01} mode	9.8 p247	R,r,d	1	L
CAPOFA	CAPacitive Window thin and Annular, E_{01} mode	9.8 p248	R,r,d	1	C
INDWTA	INDuctive Window Thick and Annular, H_{11} mode	9.8 p273	R,d,t	2	TEE L C
RESOTA	RESonant Obstacle Thick and Annular, H_{11} mode	9.8 p275	R,r,d', d''	2	TEE SRLC X

N.B. 1) For H<m><n>, E<m><n>, H<m><n> L or E<m><n> L components the characteristics of the waveguide below cutoff may be included.

Table 9.4. - Possible Network Sub-types For Circular Waveguide Sub-program, WGCIRC

9.4.2. Circular Waveguide, Program WGCIRC (Table 9.4)

The basic theory behind circular wavguide is well described by Collin[9.10] p197 and Marcuvitz[9.8] p66. In the case of cicular waveguide it is necessary to consider far more modes of propagation. In the case of rectangular waveguide usually only H_{10} mode is used but for circular waveguide it is common to use the E_{01} and H_{11} mode and in the case of the H_{11} mode it is also necessary to consider the orientation of that mode in the guide. For the circular waveguide most of the points for the rectangular waveguide, section 9.4.1, still apply. The main difference in the equations for its equivalent circuit is that the circular symmetry means that in addition bessel functions are also included in these equations.

9.4.3. Coaxial Line, Program COAX (Table 9.5)

The basic equations for coaxial line are very simple and easy to implement as normally only a TEM mode of propagation is used. Discontinuities effects are described by Marcuvitz[9.8] and King[9.13]. Here again due to the circular symmetry of the problem bessel functions and summations of series are involved for these discontinuities.

9.4.4. Two Wire Line, Program WIRE2

The two wire line is an important component used mainly for radio communications below the microwave region. The two wire line and some of its discontinuity effect are described by King[9.13].

Network sub-type	Network	Ref.	Para. meters	No. of conns.	Equiv. cit. type
LINE	TEM mode line		a,b,l, ϵ_r	2	LINE
LINEL	TEM mode line with losses in walls and dielectric		a,b,l, ϵ_r,tanδ	2	LINE
STEPI	STEP on Inner conductor of line	9.8 p310	a,b,b$'$	1	C
STEPO	STEP on Outer conductor of line	9.8 p311	a,b,a$'$	1	C
CAPGT	CAPacitive Gap Termination of line	9.8 p178	a,b,d	1	C
CAPDFI	CAPacitive Disc thin on Inner	9.8 p229	a,b,d$'$	1	C
CAPDFO	CAPacitive Disc thin on Outer	9.8 p234	a,b,d$'$	1	C
SQI	SQuare Inner and circular outer	9.21 p95	a,b,l, ϵ_r	2	LINE
SQO	SQuare outer and circular inner	9.21 p95	a,b,l, ϵ_r	2	LINE
SQIO	SQuare inner and outer	9.21 p95	a,b,l, ϵ_r	2	LINE
RECTIO	RECTangular Inner and Outer	9.21 p98	a,b,l, ϵ_r,a$'$,b$'$	2	LINE

N.B. 1) TEM mode propagation is assumed in all the cases listed above.

Table 9.5. - Possible Network Sub-types for Coaxial Waveguide Sub-program, COAX

9.4.5. Microstrip, Program MICSTRIP (Table 9.6)

The main advance in the field of microwave circuits over the last 10 years has been in the production of microwave circuits in the form of Microwave Integrated Circuits using Microstrip lines. Thus it would be essential to include microstrip lines in a program for the analysis of microwave circuits. The characteristics of the basic microstrip line on a dielectric substrate are normally solved using the method of conformal mapping e.g. Ramo et. al.(9.14). The original work in this field was carried out by Wheeler(9.23 and 9.24) and by Assadourian and Rimai(9.25) for a finite strip thickness. Another important component in microstrip line is the coupled lines. For this case it is necessary to derive the odd and even mode impedances for the coupled lines using conformal mapping as for a single line. Coupled lines were included in the MICRO3 program but only so far as including them in terms of their odd and even mode line parameters.

Some microstrip discontinuities have been characterised but only by a vcry approximate method and there is still a lot of work to be done in this field.

9.4.6. Triplate, Program TRIPLATE (Table 9.7)

The components in triplate, developed before microstrip, are similar to the microstrip components but it is much simpler to derive their equivalent circuits due to the simpler form of the problem with two planes of symmetry instead of one. The basic equations for the impedance of a triplate line is obtained by the process of conformal mapping and the solution gives an equation

Network sub-type	Network	Ref.	Parameters	No. of conns.	Equiv. cit. type
STRIPF	STRIP thin on dielectric base	9.21 p135	h,w,ϵ_r	2	LINE
STRIPT	STRIP Thick on dielectric base	9.21 p135	h,w,ϵ_r, t	2	LINE
CPLINE	CouPled LINE with thin strips	9.21 p133 p149	h,w_1,w_2 s,ϵ_r	4	CPLINE

Table 9.6. - Possible Network Sub-types for Microstrip Line Sub-program, MICSTRIP

Network sub-type	Network	Ref.	Parameters	No. of conns.	Equiv. cit. type
STRIPF	conducting STRIP thin		b,w,ϵ_r	2	LINE
STRIPT	conducting STRIP Thick	9.21 p117	b,w,ϵ_r,t	2	LINE
CPLINE	CouPled LINE with thin strips	9.21 p129	b,w_1,w_2, s,ϵ_r	4	CPLINE
GAP	GAP in inner strip	9.21 p138	b,w,ϵ_r, s	2	PHI B B

Table 9.7. - Possible Network Sub-types for Triplate Line Sub-program, TRIPLATE

Network sub-type	Network	Ref.	Parameters	No. of conns.	Equiv. cit. type
S<n>L<l>	l entries of S matrix of size n x n for char. imped. Z_o at frequency point f.		f,Z_o, S matrix	n	SPAR
Y<n>L<l>	l entries of Y matrix of size n x n at frequency f.		f, Y matrix	n	YPAR
Z<n>L<l>	l entries of Z matrix of size n x n at frequency f.		f, Z matrix	n	ZPAR

Table 9.8. - Possible Network Sub-types for Matrix Table Sub-program, TABLE

in terms of elliptic integrals, e.g. Collin[9.10] p137 and Black[9.26].

Some triplate discontinuities have been characterised in triplate e.g. Oliner[9.27].

9.4.7. Table of Parameters, Program TABLE (Table 9.8)

It is often necessary to carry out measurements on a microwave component when it is impossible or too difficult to characterise that component mathematically from its physical dimensions. Thus there would have to be facilities in the computer program to read in sets of results. For microwave components these measurements would normally be in terms of the scattering matrix of the microwave component at a set of frequency points. It would thus be necessary to read in this table of scattering parameters , store in the computer store, and, for an analysis, to interpolate this table to obtain the scattering or other matrix at the frequency point defined for analysis. The problem here is that the table often would have a large number of entries and it could take up a large amount of computer store.

References

Mathematical Functions and Subroutines

9.1) H.B. Dwight, 'Tables of Integrals and Other Mathematical Data', MacMillan, 1961

9.2) M. Abramovitz, I.A. Stegun, 'Handbook of Mathematical Functions', Dover, 1965

9.3) 'System 360 Scientific Subroutine Package', Programmers Manual, IBM, Application Package

Microwave Textbooks

9.4) C.G. Montgomery, R.H. Dicke, E.M. Purcell, 'Principles of Microwave Circuits', McGraw-Hill, 1948

9.5) T. Moreno, 'Microwave Transmission Design Data', McGraw-Hill, 1948

9.6) G.L. Ragan, 'Microwave Transmission Circuits', McGraw-Hill,1948

9.7) L. Lewin, 'Advanced Theory of Waveguides', Iliffe, 1951

9.8) N. Marcuvitz, 'Waveguide Handbook', No. 10 Radiation Laboratory Series, McGraw-Hill, 1951

9.9) G.C. Southworth, 'Principles and Applications of Waveguide Transmission', D. Van Nostrand Co., Princeton, N.J., 1950

9.10) R.E. Collin, 'Field Theory of Guided Waves', McGraw-Hill, 1960

9.11) A.F. Harvey, 'Microwave Engineering', Academic Press, 1963

9.12) G.N. Ghose, 'Microwave Circuit Theory and and Analysis', McGraw-Hill, 1963

9.13) R.W.P. King, 'Transmission Line Theory', Dover, 1965

9.14) S. Ramo, J.R. Whinnery, T. Van Duzer, 'Fields and Waves in Communication Electronics', Wiley, 1965

9.15) R.E. Collin, 'Foundations of Microwave Engineering', McGraw-Hill, 1966

9.16) E. Argence, T. Kahan, 'Theory of Waveguides and Cavity Resonators', Blackie, 1967

9.17) D.M. Kerns, R.W. Beatty, 'Basic Theory of Waveguide Junctions and Introductory Microwave Network Analysis', Pergamon, 1967

9.18) J. Schwinger, D.S. Saxon, 'Discontinuities in Waveguide', Gordon and Breach, 1968

9.19) P. Grivet, 'The Physics of Transmission Lines at High and very High Frequency', Academic Press, 1970

9.20)H.A.Waston, 'Microwave Semiconductor Devices and their Circuit Applications', McGraw-Hill, 1969

9.21) 'Microwave Engineers Technical and Buyers Guide', Microwave Journal, Lastest Edition

Other references

9.22) G.F. Craven, C.K. Mok, 'The Design of Evanscent Mode Waveguide Bandpass Filters for a Prescribed Insertion Loss Characteristic', IEEE trans. Microwave Theory and Techniques, Vol. MTT-19, No. 3, March 1971, pp. 295-308

9.23) H.A. Wheeler, 'Transmission Line Properties of Parallel Wide Strips by Conformal Mapping Approximations', IEEE trans. Microwave Theory and Techniques, Vol. MTT-12, No. 3 , May 1964, pp. 280-289

9.24) H.A. Wheeler, 'Transmission Line Properties of Parallel Strips Separated by Dielectric Sheet', IEEE trans. Microwave Theory and Techniques, Vol. MTT-13, No. 2, March 1965, pp. 172-185

9.25) F. Assodourian, E. Rimai, 'Simplified Theory of Microstrip Transmission Systems', Proceedings of IRE, Vol. 40, No. 12, Dec. 1952, pp. 1651-1657

9.26) K.G. Black, T.J. Higgins, 'Rigorous Determination of the Parameters of Microstrip Transmission Lines', IRE trans. Microwave Theory and Techniques, Vol. MTT-3, No. 2, March 1955

9.27) A.A. Oliner, 'Equivalent Circuits for Discontinuities in in Balanced Strip Transmission Lines', IRE trans. Microwave Theory and Techniques, Vol. MTT-3, No. 2, March 1955

Chapter 10

Optimisation in Microwave Analysis Program

10.1. INTRODUCTION

This chapter describes the facilities for optimisation provided in the MICRO3 program. In this program all the data statements, data structure and circuit analysis were supplied for optimisation but the optimisation packages themselves, i.e. to vary the parameters in the circuit for the optimisation, must be included in programs completely separate from the MICRO3. In this way it is possible to experiment with various optimisation sub-programs without altering the main program. Due to the lack of time available in the course of the research work for this thesis it was not possible to carry out a complete study of all the optimisation methods available to select the best one(s) for the optimisation of microwave circuits. At least the MICRO3 program has been prepared for this work which could be carried out by another research student.

In the MICRO3 program the objective of including facilities for optimisation was to define :-

1) A set of parameters in the circuit to be varied during th^ optimisation with each parameter assigned an optional lower and upper bound.
2) A set of desired circuit performance values, with optional weighting factors, at a set of frequency points. (A extension of this, not so far included, could be to also define a lower and upper bound for the desired circuit performance values.)
3) The name of the optimisation sub-program, from the data, to be used for the optimisation.
4) The maximum number of iterations to be used during the optimisation.

10.2. INPUT DATA

10.2.1. Syntactical Definition

The syntactical definition of the input data for the MICRO3 program for optimisation is an extension of the syntactical definition of the data for the MICRO3 program described in section 7.3.2 as described below.

```
<optimisation statement> ::= <spec statement>|
                             <vary statement>|
                             <optimise statement>
<spec statement> ::= SPEC <spec list>
       <spec list> ::= <empty >|<spec list><spec>
       <spec> ::=  <spec  option> <spec option value list>
       <spec option> ::= <see Table A.5.4>
       <spec option value list> ::= <empty>|<spec option value list>
                                    <spec option value>
       <spec option value> ::= <frequency><option value><weight>
       <weight> ::= <empty>| WEIGHT <weight value>
<vary statement> ::= VARY <vary list>
       <vary list> ::= <empty>|<vary list> <vary element>
       <vary element> ::= <set circuit>|<vary parameter><par bounds>
       <set circuit> ::=  CIRCUIT <circuit no.>
       <vary parameter> ::= NET <network no.> PAR <net par no.>|
                            ZO <char imped no.> PAR <char imped par no.>
       <par bounds> ::= <empty>|<lower bound><upper bound>
<optimise statement> ::= OPTIMISE <optim prog name><no. of iterations>
```

10.2.2. Specification Statement

This statement is used, in the data, to define the desired performance of the circuit. The statement defines a list of output option values at frequency points. The statement starts with the word SPEC followed by a list of output options with each output option followed by a list of values of this output option defined at frequency points with an optional weighting factor.

```
e.g. SPEC SPAR 1 1 MOD 1.06E9 0.5
                       1.08E9 0.2 WEIGHT 2.0
                       1.10E9 0.6
          VSWR 2 1.06E9 1.10 WEIGHT 0.5
                 1.08E9 1.00
                 1.10E9 1.15 WEIGHT 0.5 ;
```

A new<spec statement> will overwrite any previous <spec statement>.

10.2.3. Vary Statement

A list of variables in the circuit(s) which may be varied during a circuit optimisation are defined in the <vary statement>. This statement starts with the word VARY followed by a list of the parameters which may be varied during a circuit optimisation.

```
e.g. VARY CIRCUIT 5   NET 4 PAR 2   NET 7 PAR 3 -10 50.2
                      ZO 2 PAR 3
          CIRCUIT 7   NET 1 PAR 4 1.26 2.5
                      ZO 3 PAR 1 0 50 ;
```

A new <vary statement> will overwrite any previous <vary statement>. The elements which may occur in the <vary statement> are as follows :-

1) Set circuit - this will set the circuit no. for the following parameters in the vary statement. If it is not included circuit no. 0 is assumed.

   e.g. CIRCUIT 5

2) Network parameter - this will set a parameter no. in the given network no. as a parameter which may be varied in an optimisation.

   e.g. NET 4 PAR 2

3) Characteristic impedance parameter - this will set a parameter no. in the given characteristic impedance as a parameter which may be varied in a optimisation.

   e.g. ZO 2 PAR 3

4) Lower and upper bounds - optional lower and upper bounds may be included after a network or characteristic impedance parameter.

   e.g. 1) NET 7 PAR 3 -10 50.2
        2) ZO 3 PAR 1 0 50

10.2.4. Optimise Statement

The optimise statement is used to initialise an optimisation. It consists of the word OPTIMISE followed by the name of the optimisation sub-program to be used and the maximum no. of iterations allowed during the optimisation.

e.g. OPTIMISE SIMPLEX 200 ;

After the optimise statement has been read a check is made to ensure that the optimisation sub-program is present in the computer core store or can be loaded into store. At the same time this program is entered and its data structure initialised. Before an optimisation can start the circuit desrciption(s) to be analysed are checked, as in section 7.3.5, and the data for the optimisation checked, as in section 10.3.3.

The optimisation starts with the program printing out the message <•• OPTIMISE> . During each interaction in the optimisation the program prints out a message giving the current value of the error sum. Finally when the optimisation is complete the values of all the parameters in the vary list are printed out.

10.3. DATA STRUCTURE

10.3.1. Specification Bead

The specification bead, Fig. 10.2, is similar to the bead for the output options, Fig. 6.17, except that the bead is longer and, in addition, holds the desired value of the output option, the point frequency value and the weighing factor. The specification statement is broken up in the data processing to form a list of these beads pointed to in address 8 in the data structure.

10.3.2. Vary Bead

The vary bead, Fig. 10.1, is contained in a list pointed to in address 9 in the data structure. During the processing of a network or characteristic impedance parameter in the vary statement a vary bead is set up for each parameter and added to the end of the new vary list. The contents of the vary bead give full details of the parameter , the circuit it is in and the lower and upper bounds for that parameter. If the parameter bounds are omitted in the data then they are set to $-\infty$ and $+\infty$ respectively (i.e. the closest approximation which can be stored in the computer as a single real number). The parameter address in the data structure is set in the vary bead just before an optimisation, section 10.3.3, so it can be accessed directly during the optimisation.

10.3.3. Check Data for Ontimisation

The procedure CHOPTS(optim) is entered, for an optimisation check, with optim set to true and it will check that sufficient additional information has been supplied for the optimisation as follows :-

1) Check that at least one characteristic impedance is present.

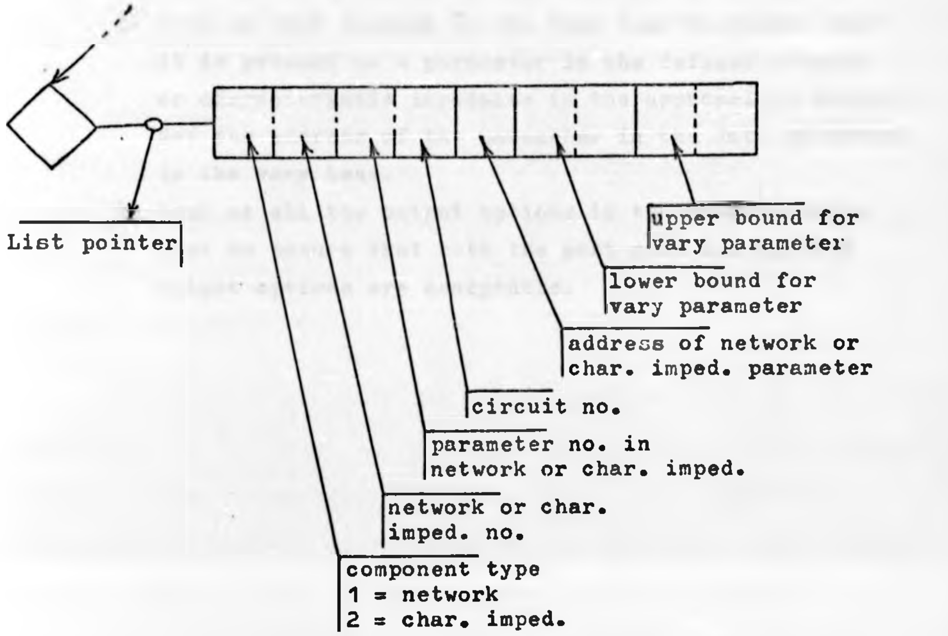


Fig. 10.1. - Vary Bead

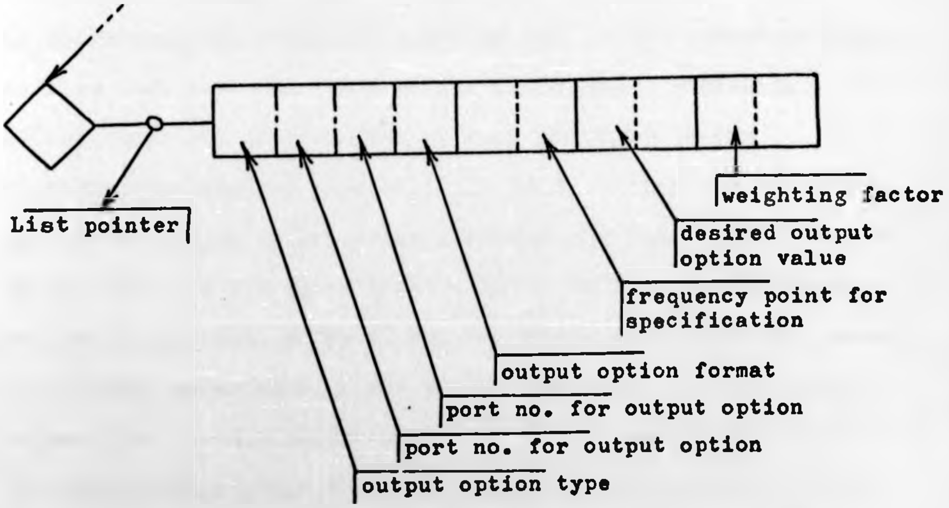


Fig. 10.2. - Specification Bead

2) Check that a vary list is present.
3) Look at each element in the vary list to ensure that it is present as a parameter in the defined network or characteristic impedance in the appropriate circuit. Set the address of the parameter in the data structure in the vary bead.
4) Look at all the output options in the specification list to ensure that both the port no.s for all the output options are acceptable.

10.4. OPTIMISATION

10.4.1. Circuit Analysis for Optimisation

The circuit analysis for the optimisation is contained in the Circuit Analysis segment of the MICRO3 program, as described in section 7.4, whilst the optimisation of the circuit parameters is carried out entirely in a sub-program written for the required method of optimisation.

The flow diagram for the optimisation is shown in Fig. 10.3 and this is an extension of the flow diagram for a simple circuit analysis shown in Fig. 7.7. The optimisation sub-program is initialised during the syntax analysis of the data . The Circuit Analysis segment of the MICRO3 program is entered with the reference to the sub-program for the optimisation contained in address 10 and 11 and the maximum number of iterations contained in address 12 in the data structure. The optimisation starts by decreasing the iteration count by one. If the iteration count is less than zero the optimisation is complete. Otherwise a circuit analysis is performed at each frequency in the specification list in turn whilst keeping a record of the error sum of the output option value obtained from the required output option value in the specification list. Then the optimisation program is entered, entry 0, and the sub-program will then adjust the circuit parameters, given in the vary list, in an attempt to improve the circuit performance towards the specification. If the optimisation program can not improve the performance of the circuit any further, or the specification has been met, then the iteration count is set to zero before the optimisation sub-program is left.

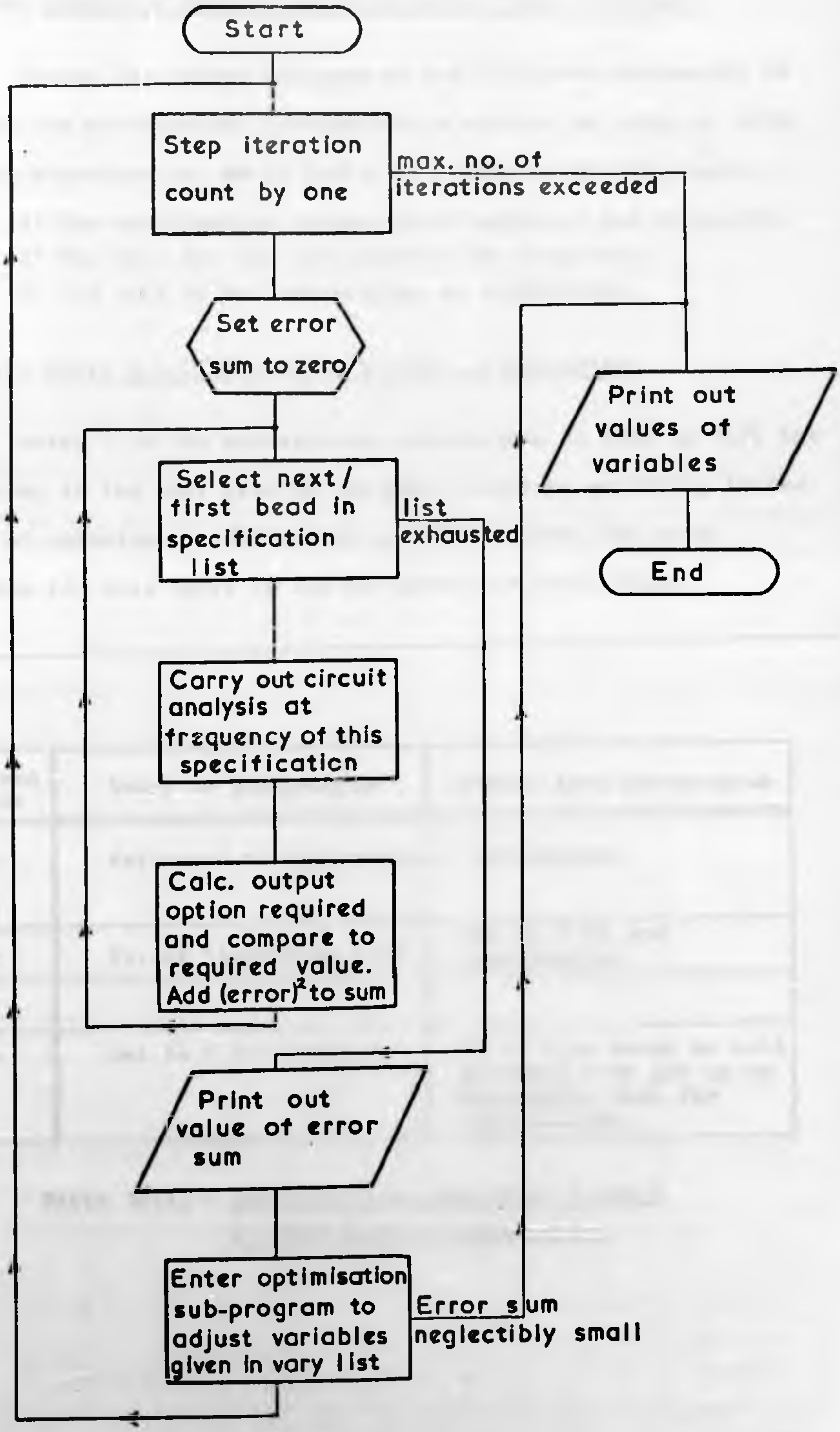

Start
Step iteration count by one
max. no. of iterations exceeded
Set error sum to zero
Print out values of variables
End
Select next/first bead in specification list
list exhausted
Carry out circuit analysis at frequency of this specification
Calc. output option required and compare to required value. Add (error)^2 to sum
Print out value of error sum
Enter optimisation sub-program to adjust variables given in vary list
Error sum neglectibly small

Fig.10.3.-Flow Diagram for Optimisation

10.4.2. Entry 0 - Initialise Optimisation Data Structure

During the syntax analysis of the <optimise statement> in the data the optimisation sub-program is entered at entry 0 with the data structure set as in Table 10.1. This entry will ensure :-

1) The optimisation program is in store or can be loaded.
2) The data for the optimisation is acceptable.
3) The data in the sub-program is initialised.

10.4.3. Entry 1 - Optimisation of Circuit Parameters

Entry 1 in the optimisation sub-program is used to vary the parameters in the vary list in the data structure according to the method of optimisation defined in the sub-program. The data structure for this entry is set as defined in Table 10.1.

Address in IST	Entry to sub-program	Return from sub-program
10	Reference to optimisation sub-program	
11		
12	No. of iterations left	Set to 0 to end optimisation
13		
14	Set to 0 for entry 0	If <0 then error in call of entry 0 to set up or initialise data for optimisation.

Table 10.1. - Data Structure for Entry 0 and 1 to Optimisation Sub-program

10.5. OPTIMISATION TECHNIQUES

During the course of this thesis a short time was spent investigating the techniques available for the optimisation of microwave circuit parameters in circuit design. The objective was to select or develop a suitable optimisation routine which would give the values of the circuit parameters to provide a required performance of the circuit. The list of parameters is defined in the VARY list, section 10.3.2, and the required circuit performance is defined in the SPEC list, section 10.3.1. The objective of the optimisation routine would then be to adjust the parameters in the VARY list to minimise the value of a suitable objective function, F. The most common objective function is :-

$$F = \sum_{i=1}^{n} \left\{ \frac{c_i - s_i}{s_i} \right\}^2 w_i$$

giving the value of F as the sum of the p.u. error squared of each part of the circuit performance from its required value.

where c_i = actual circuit performance value

s_i = required circuit performance value

w_i = weighting factor for this performance value

The advantage with this objective function is that F is always positive, the minimum value of F will give the optimum response and the function F will usually be continuous.

The derivatives of F for each parameter are not usually easily available. Thus the optimisation routines used are usually of the Direct Search Method e.g.

1) PARTAN(10.17) - search for minimum along two parallel planes at a time
2) Hooke and Jeeves(10.18) - try exporatory moves in all directions and then use a pattern move
3) Simplex(10.19,10.26) - find values of F at all corners of a simplex and reflect the corner with the largest error value through the centroid of the simplex

4) Orthogonal Directions(10.15) - find suitable set of n orthogonal directions and rotate these in the direction of steepest descent

5) Conjugate Directions(10.23,10.24,10.25)

During the process of a number of optimisation routines it is often necessary to find the minimum of a function along a line. This is fairly simple and usually a Fibonacci Search, Golden Search or Quadratic Interpolation is used to find the mimimum value of the objective function once the region along this line in which the minimum occurs has been tound.

The Gradient Search Methods have not been used for circuit optimisation because of the difficulty and time involved in calculating an accurate gradient vector for the objective function. It is necessary to change all the parameters in the optimisation by a small amount in turn to derive the change in the objective function and thus its gradient. This requires (n+1) circuit analyses, for n variables in the optimisation, to calculate, numerically, the gradient of the objective function. The changes required in the parameter values are very small and thus the value for the gradient will be very inaccurate due to rounding errors in the computer. Director and Rohrer(10.32,10.33) have described a method to derive the sensitivities of a lumped element circuit directly. This requires generating and analysing an Adjoint circuit based on the use of Tellegens theorem(10.12). If an equivalent Adjoint circuit for a microwave circuit considered as an assembly of n-port networks could be derived then the process of optimisation, using Gradient Methods, would be greatly speeded up. Some methods simply use the first order gradient of the objective function whilst others try to obtain an approximation to the second order derivatives in the form of the Hessian matrix of the objective function to speed up the optimisation.

In the practical optimisation of microwave circuits the following problems may be met in certain circuits :-

1) Parameter constraints - it may be necessary to restrict the circuit parameters to practical limiting values. In this case it would be necessary to apply some form of transformation to the objective function or parameter values to form an unconstrained optimisation.

2) Limits on the circuit performance values - in some cases the specification of the required circuit performance may just define limiting values and not point values. In this case the objective function would have to be changes so that the error sum of the circuit performance value outside these limits is taken and not the error from a given point value. In this case discontinuties in the objective function will result.

3) Resonant effects - resonant effects in the circuit may result in the objective function having very narrow valleys. Thus it would be necessary to locate these valleys and find the minimum value in the valley.

4) Multiple minima - with resonant components in the circuit, particulary with transmission lines which may resonant at an infinite number of different lengths, multiple minima will be produced in the objective function. Ideally in an optimisation, one would like to find the minimum value of all the minima but usually this is impractical. Almost all optimisation routines will only find the local minimum and it is necessary for the user to start with parameter values close to this minimum. In an optimisation with parameter constraints it may be possible to add a random effect to the optimisation to, if not find the minimum of the minima, then to at least increase the probability of finding it.

Due to lack of time it was only possible to define the problems involved in circuit optimisation for microwave circuits. There is a very large amount of literature already available on the topic as described in the references at the end of this chapter.

References

Text Books

10.1) K.I. Arrow, Hurwitz, Uzawa, 'Studies in Linear and Non-linear Programming', Stanford University Press, 1958

10.2) A. Lavi, T.P. Vogl, 'Recent Advances in Optimisation Techniques', Wiley, 1966

10.3) J. Abadie, 'Non-linear Programming', North Holland, 1967

10.4) D.J. Wilde, C.S. Beightler, 'Foundations of Optimisation', Prentice Hall, 1967

10.5) D.A. Calahan, 'Computer Aided Network Design', McGraw Hill, 1968

10.6) A.V. Fiacco, G.P. McCormick, 'Non-linear Programming : Sequential Unconstrained Minimisation Techniques', Wiley, 1968

10.7) J. Kowalik, M.R. Osborne, 'Methods for Unconstrained Optimisation Problems', Elsevier, 1969

10.8) D.J. Wilde, 'Optimum Seeking Methods', Prentice Hall, 1964

10.9) G.S.G. Beveridge, R.S. Schechter, 'Optimisation : Theory and Practice', McGraw Hill, 1970

10.10) R. Fletcher (editor), 'Optimisation : Symposium of the Institute of Mathematics and its Applications', Academic Press, 1969, (Symposium at University of Keele, 1968)

10.11) 'Optimisation Techniques in Circuit and Control Applications', IEE Conference Publication No. 66, June 1970

10.12) P. Penfield, R. Spence, S. Duinker, 'Tellegans Theorem and Electrical Networks', MIT Press, 1970

10.13) J.B. Rosen, O.L. Mangesaneon, K. Ritter, 'Non-linear Programming', Academic Press, 1971

10.14) S.H. Brooks, 'A Discussion of Random Methods of Seeking Maxima', Operations Research, Vol. 6, No. 2, March 1958, p244-251

10.15) H.H. Rosenbrock, 'An Automatic Method for Finding the Greatest of Least Value of a Function', Computer Journal, Vol. 3, 1960 - 61 pp 175-184

10.16) R.J. Buchler, B.V. Shah, O. Kempthorne, 'Some Properties of Steepest Ascent and Related Procedures for Finding Optimum Conditions', Iowa State University, Statistical Laboratory, No. 18, p8-10

10.17) R.J. Buchler, B.V. Shah, O. Kempthorne, ' The Method of Parallel Tangents (PARTAN) for Finding an Optimum', Technical Report No. 2, Office of Naval Research, Iowa State University, Statistical Laboratory, April 1961

10.18) R. Hooke, T.A. Jeeves, 'Direct Search Solution of Numerical and Statistical Problems', Association for Computing Machinery Journal, No. 8, 1961, p212

10.19) F.R. Himsworth, W. Spendley, G.R. Hext, 'Sequential Application of Simplex Design in Optimisation and Evolutionary Operations', Technometrics, No. 4, 1962, pp. 441

10.20) M.J.D. Powell, 'An Iterative Method for Finding Stationary Values of a Function of Several Variables', Computer Journal, Vol. 5, 1962 - 63, pp. 147-151

10.21) R. Fletcher, M.J.D. Powell, 'A Rapidly Convergent Descent Method for Minimisation', Computer Journal, Vol. 6, 1963 - 64, pp. 163 - 168

10.22) D.J. Wilde, 'Optimisation by the Methods of Contour Tangents', American Institute of Chemical Engineers Journal, Vol. 9, No. 2, March 1963, pp. 186-190

10.23) R. Fletcher, C.M. Reeves, 'Function Minimisation by Conjugate Gradients', Computer Journal, Vol. 7, 1964 - 65, pp. 149-154

10.24) M.J.D. Powell, 'An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives', Computer Journal, Vol. 7, 1964 - 65, pp. 155 - 162

10.25) M.J.D. Powell, 'A Method for Minimising a Sum of Squares of Non-linear Functions without Calculating Derivatives', Computer Journal, Vol. 7, 1964 - 65, pp. 303 - 307

10.26) J.A. Nelder, R. Mead, 'A Simplex Method for Function Minimisation', Computer Journal, Vol. 7, 1964 - 65, pp. 308 - 313

10.27) R. Fletcher, 'Function Minimisation without Evaluating Derivatives - A Review', Computer Journal, Vol. 8, 1965 - 66 , pp. 33 - 41

10.28) M.J. Box, 'A New Method of Constrained Optimisation and Comparison with Other Methods', Computer Journal, Vol. 8, 1965 - 66, pp. 42 - 52

10.29) M.J. Box, 'A Comparison of Several Current Optimisation Methods and the use of Transformations in Constrained Problems' Computer Journal, Vol. 9, 1966 - 67, pp. 67 - 77

10.30) W.I. Zangwill, 'Minimising a Function without Calculating Derivatives', Computer Journal, Vol. 10, 1967 - 68, pp. 293 - 296

10.31) M.J.D. Powell, 'Fortran Subroutine for Solving Systems of Non-linear Algebraic Equations', UKAEA, Research Report No. AERE-R5947, HMSO 1968

10.32) S.W. Director, R.A. Rohrer, 'The Generalised Adjoint Network and Network Sensitivities', IEEE trans on Circuit Theory, Vol. CT-16, No.3, Aug. 1969, pp. 318-323

10.33) S.W. Director, R.A. Rohrer, 'Automated Network Design : The Frequency Domain Case', IEEE trans. on Circuit Theory, Vol. CT-16, No. 3, Aug. 1969, pp. 330-337

10.34) M.R. Osborne, C.A. Watson, 'An Algorithm for Minimax Approximation in the Non-linear Case', Computer Journal, Vol. 12, 1969, pp. 63 - 68

10.35) J.R. Palmer, 'An Improved Procedure for Orthogonalising the Search Vectors in Rosenbrock and Swann Direct Search Optimisation Method', Computer Journal, Vol. 12, 1969, pp. 69 - 71

10.36) J.W. Bandler, 'Current Trends in Network Optimisation', IEEE trans. Microwave Theory and Techniques, Vol. MTT-18, No. 12, Dec 1970, pp. 1159-1170

10.37) L.P. Huelsman, 'GOSPEL : A General Optimisation Software Package', University of Arizona, Department of Electrical Engineering, 1970

10.38) S.N. Ghani, L. Barnes, 'Parameter Optimisation for Unconstrained Object Function - A Bibliography', Computer Aided Design, Vol. 4, No.5, Oct. 1972

10.39) M.J.D. Powell, 'Numerical Methods for Optimisation', IEE Conf. Publication No. 66, June 1970

10.40) W. Murray, 'Constrained Optimisation', IEE Conf. Publication No. 66, June 1970

10.41) P.C. Haarhoff, J.D. Buys, 'A New Method for the Optimisation of a Non-linear Function Subject to Non-linear Constraints', Computer Journal, Vol. 13, 1970, pp. 178 - 184

10.42) B.A. Murtagh, R.W.H. Sargent, 'Computational Experience with Quadratically Convergent Minimisation Methods', Computer Journal, Vol. 13, 1970, pp. 185 - 194

10.43) G. Peckham, 'A New Method for Minimising a Sum of Squares without Calculating Gradients', Computer Journal, Vol. 13, 1970, pp. 418 - 420

10.44) S.W Director, 'Survey of Circuit Orientated Optimisation Techniques', IEEE trans. Circuit Theory, Vol. CT-18, No. 1, Jan 1971, pp. 3-10

10.45) J.K. Skwirzynski, 'Non-Linear Programming', IEE Vacation School on Circuit Theory, July 1971

10.46) G.A. Richards, 'The Adjoint Concept and Sensitivities', IEE Vacation School on Circuit Theory, July 1971

Chapter 11

Conclusions of Thesis

Chapter II

Conclusions of Thesis

11.1. CONCLUSIONS

The objective of this thesis was to develop the basic techniques and computer programs for the interactive on-line computer aided design of linear microwave circuits.

Initially it was necessary to consider all the possible types of microwave circuits to see how they could be analysed in a computer program. The result of this was that it was found that the best way to analyse a microwave circuit was first to break it down into an assembly of n-port networks so that the characteristics of each n-port network could be easily derived. Then the microwave circuit could be analysed as an assembly of n-port networks with the networks being connected together at junctions in the circuit and the entire circuit consisting of a single n-port network. At microwave frequencies it is necessary to describe the performance of the circuit in terms of the electromagnetic waves on the ports of the circuit. This can best be done in the form of the scattering matrix but it should be remembered that for complex characteristic impedances on the ports on the circuit two different types of scattering matrices exist. One describes the amplitudes of the electric fields in the waves and the other the amplitudes of the power transmitted in the waves. In general the microwave engineer may be interested in both types of scattering parameters. In this thesis all the equations have been set out to derive these scattering parameters for an n-port network.

For the analysis of a microwave circuit various methods of analysis were compared in detail for their method and optimum computation times. The result of this was that the chain matrix method was found to be fastest but of limited application whilst the mixed matrix analysis was able to analyse any microwave circuit. In previous programs for the analysis of microwave circuits the main method of analysis used was the chain matrix analysis. This was because it was very simple to program for simple circuits consisting of an assembly of 2-port networks in cascade. The mixed matrix is very basic and simple to understand but it has not before been described for general circuit analysis nor used in a computer program for circuit analysis.

The first programs developed by the author were the programs using chain matrix analysis, the main one of these being the CHAIN1 program. This program advanced the use of the chain matrix method of analysis just about to its limit. The main improvement in this program was the path topological analysis to break a circuit consisting of an assembly of 2-port networks into a path structure which could be processed by a chain matrix method of analysis. In this way the program was easy to use and suitable for use in an interactive on-line mode. Previously it had always been necessary to define the path structure of the circuit in a very precise way in the data. Unfortunately the limitation in this program was that the chain matrix analysis could only analyse a circuit consisting of an assembly of 2-port networks. A lot of time on developing the path topological analysis was spent making it work for all possible structures but still some little used circuits could not be analysed correctly and the topological analysis would reject the circuit.

The final two programs developed by the author used the mixed matrix method of analysis. This was found to be far more versatile and any microwave circuit consisting of an assembly of n-port networks could be handled by this method of analysis.

One problem was that a square matrix, of size given by the number of junctions in the circuit, must be set up and the internal junction variables eliminated from this matrix. Even though sparse matrix operations were used this method was slower but not by an excessive amount. In the mixed matrix analysis there was no need for a complex path topological analysis.

It is certain that the best method of analysis for general microwave circuits is the mixed matrix method of analysis described in this thesis. It may be necessary to design a computer program to analyse all circuits for a linear analysis from d.c. up to microwave frequency. For this purpose n-port networks in the circuit could be defined as a lumped element network. The lumped element analysis would then derive the required mixed matrix for the networks using a mixed mesh and cutset analysis and the mixed matrix analysis would analyse the final circuit, if necessary, as an assembly of n-port networks.

During the development of the BGMA, CHAIN1, MICRO2 and MICRO3 programs by the author as each new program was developed more facilities were included for the interactive on-line use of the program. The result in the final MICRO3 program was a very versatile program which could be used interactively on-line with a full syntax analysis of the data.

In the MICRO3 program a list processing approach was used for the data structure in the program and thus it was possible to set up very complex structures in the data structure of the program. The result of this was that it was possible to define any number of separate circuits in the data structure with each circuit containing any number of components and junctions. Also other lists of data for output options, frequency lists, specification list, etc. could be included at the same time in the same data structure. The components in the circuits could themselves represent complex structures to be able to describe any microwave component. This type of data structure is ideal for interactive on-line use of the program where it is necessary to define a circuit in the data, analyse it and then to add to, modify or delete parts of the circuit description or other lists in the data and then to reanalyse as many times as desired.

The data for the MICRO3 program was more like a programming language, complete with a syntactical definition, than normal data for a program. To analyse this data a full syntax analysis was used. In this method all the data is read in one character at a time and numbers, words etc. formed by procedures in the program so that errors in the data could be trapped. Also the program could be routed depending on the next item in the data i.e. a word, number , terminating character etc.. This of course is much slower than the normal method of reading data but it is possible to trap all possible errors in the data and on-line, in an interactive mode, to allow these errors to be corrected by the user typing in further data to correct or overwrite the error. The techniques developed could be used for most interactive programs.

The MICRO3 program could be used on batch processing, interactively on a remote teletype or on a remote teletype with a graphical display. In an interactive mode it was possible to detect all data errors as they were detected in the data and, with the graphical display, it was possible to see the results of a circuit analysis plotted out in graphical form. The use of the program in an interactive mode was found to be very useful but very expensive in terms of computer time. The problem is in this mode the processor on the computer is only used at less than 1% efficiency usually. Thus it is essential to use the computer in a multi-access mode where both the processor and core store can be used for other work at the same time. The use of a storage tube for the display instead of a refresh display would assist with this.

Initially in the MICRO3 program a larger number of procedure calls were used to ease the programming effort required. These made the program very slow and it was necessary to use a macrogenerator to replace all these procedure calls with their equivalent block of NEAT assembly code. At this point it was realised that the program should have been written in Fortran and not in Algol. Algol is very versatile but this versatility makes the program run slowly particularly if a large number of procedure calls are involved. Also there are no facilities for complex numbers in Algol which are essential for microwave linear circuit analysis. In Fortran it is also quite easy to use the same data structure for storing data describing the circuit and during analysis the free space may be used as workspace for the analysis. There are still a large number of basic checks in the program, e.g. array overflow, which in the complete program should not be necessary. If this type of check could be removed the analysis would be speeded up.

The list processing approach to the data structure in the MICRO3 program made it easy to add microwave components of a complex nature and optimisation to the program. The MICRO3 program was initially designed so that the microwave components and optimisation routines would be held in completely separate programs due to the large number of microwave components and optimisation routines which may be required. Thus only the programs required by a given circuit analysis would be loaded into core store. This method is good in theory but in practice it was very difficult to implement and the result was that this system would only work in some cases. There is still a very large amount of work to be done to provide these programs for the wide variety of possible microwave components to be included and optimisation routines to be tested for their efficiency. This work could consitute a possible future Ph. D. topic.

After the completion of the research work for this thesis the author started a new project with Redac Software Ltd., Tewkesbury, This project consisted of rewriting their REDAP31 general circuit analysis program to include sub-circuits. Due to the poor data structure and excessive size of the program it was decided to completely rewrite it. During this process the techniques of data structure, syntax analysis, circuit analysis, matrix analysis etc. descibed in this thesis were used in the new program. The result was that the new program(11.1) was far more versatile, the data structure was better organised, the syntax analysis was less susceptable to errors in the data to give corruption of the program, the core store used by the program was greatly reduced and the program was easier to extend with further facilities as desired at a later data.

References

11.1) 'REDAP51 General Circuit Analysis Program : Program Description', Redac Software Ltd., 1973, (to be completed)

programs due to the large number of microwave components and optimisation routines which may be required. Thus only the programs required by a given circuit analysis would be loaded into core store. This method is good in theory but in practice it was very difficult to implement and the result was that this system would only work in some cases. There is still a very large amount of work to be done to provide these programs for the wide variety of possible microwave components to be included and optimisation routines to be tested for their efficiency. This work could consitute a possible future Ph. D. topic.

After the completion of the research work for this thesis the author started a new project with Redac Software Ltd., Tewkesbury, This project consisted of rewriting their REDAP31 general circuit analysis program to include sub-circuits. Due to the poor data structure and excessive size of the program it was decided to completely rewrite it. During this process the techniques of data structure, syntax analysis, circuit analysis, matrix analysis etc. descibed in this thesis were used in the new program. The result was that the new program(11.1) was far more versatile, the data structure was better organised, the syntax analysis was less susceptable to errors in the data to give corruption of the program, the core store used by the program was greatly reduced and the program was easier to extend with further facilities as desired at a later data.

References

11.1) 'REDAP51 General Circuit Analysis Program : Program Description', Redac Software Ltd., 1973, (to be completed)

copy 1

Interactive Computer Programs for the Computer Aided Design of Linear Microwave Circuits (Appendix)

Author
Brian G. Marchent, B. Sc.(Hons.)

A thesis submitted at the University of Warwick for the degree of Doctor of Philosophy.

Date : July 1973

Appendix A.1

L-Band Microwave Integrated Circuit Switch

A.1.1. INTRODUCTION

At the start of the research work for this Ph. D. thesis 5 months were spent working at Microwave Associates Ltd., Luton. This Appendix describes the first of two design studies(A.1.1) carried out during this period.

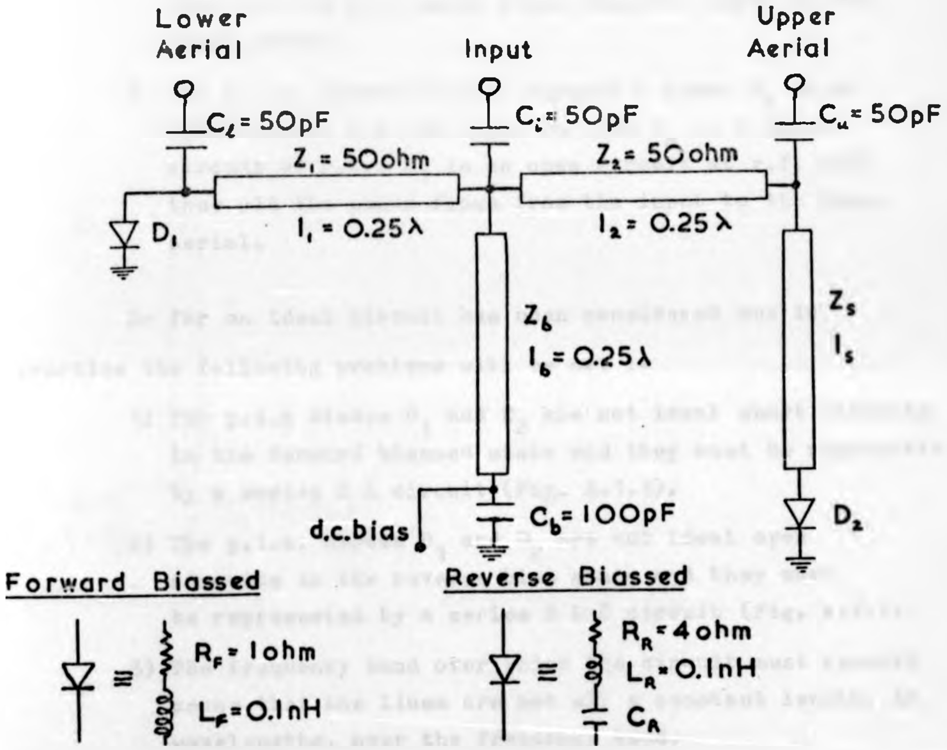
Before this design study was started a microwave integrated circuit (M.I.C.) switch had been designed by Microwave Associates at Luton and a few had been constructed and tested. During this design the MACAP computer program(A.1.2) for microwave circuit analysis had been used to assist in the design. The intention of the author was to carry out a complete design and analysis of the M.I.C. switch with the aid of the MKMcPH computer program(A.1.3) written for the analysis of microwave circuits to at least prove the results obtained by the MACAP program and, if possible, to improve the design of the M.I.C. switch.

The required performance for the microwave integrated circuit L-band switch was as follows :-

1) Frequency band	1.02 to 1.10 GHz
2) Aerial Isolation	20dB
3) Aerial Insertion Loss	0.5dB
4) Voltage Standing Wave Ratio	1.2

A.1.2. OLD SWITCH DESIGN

The old M.I.C. switch design, Fig. A.2.1, consisted of an assembly of approximately quarter wavelength microstrip transmission lines, i.e. lines Z_1, Z_2, Z_b and Z_s, p.i.n. diodes D_1 and D_2 (both always biassed in the same direction), capacitor C_b (to give a short circuit at r.f.) and 3 lower frequency decoupling capacitors C_l, C_i and C_u. In an idealised circuit the d.c. (or lower frequency bias) for the p.i.n. diodes is supplied through



Model 1A

C_R = 0.25pF, Z_b = 100 ohm, Z_s = 50ohm,
l_s = 0.19, 0.21, 0.23, 0.25 λ

Model 1B

C_R = 0.5 pF, Z_b = 75ohm,

Z_s =	20.0	28.3	34.7	40.0	44.9	49.0 ohm
l_s =	0.241	0.235	0.232	0.229	0.226	0.224 λ
L =	0.2	0.4	0.6	0.8	1.0	1.2 nH

(L in series with D)

Fig.A.1.1.- Old M.I.C. L-Band Switch Equivalent Circuit

line Z_b terminated in an r.f. short circuit. Thus for :-

1) The p.i.n. diodes forward biassed - diode D_1 is a short circuit and the input to line Z_1, at r.f., is an open circuit. Also diode D_2 is a short circuit and the input to line Z_s is an open circuit at r.f.. Thus all the r.f. power flows from the input to the upper aerial.

2) The p.i.n. diodes reverse biassed - diode D_2 is an open circuit and the input to line Z_s is a short circuit at r.f.. D_1 is an open circuit at r.f. and thus all the power flows from the input to the lower aerial.

So far an ideal circuit has been considered but in practice the following problems will be met :-

1) The p.i.n diodes D_1 and D_2 are not ideal short circuits in the forward biassed state and they must be reprsented by a series R L circuit (Fig. A.1.1).

2) The p.i.n. diodes D_1 and D_2 are not ideal open circuits in the reverse bias state and they must be represented by a series R L C circuit (Fig. A.1.1).

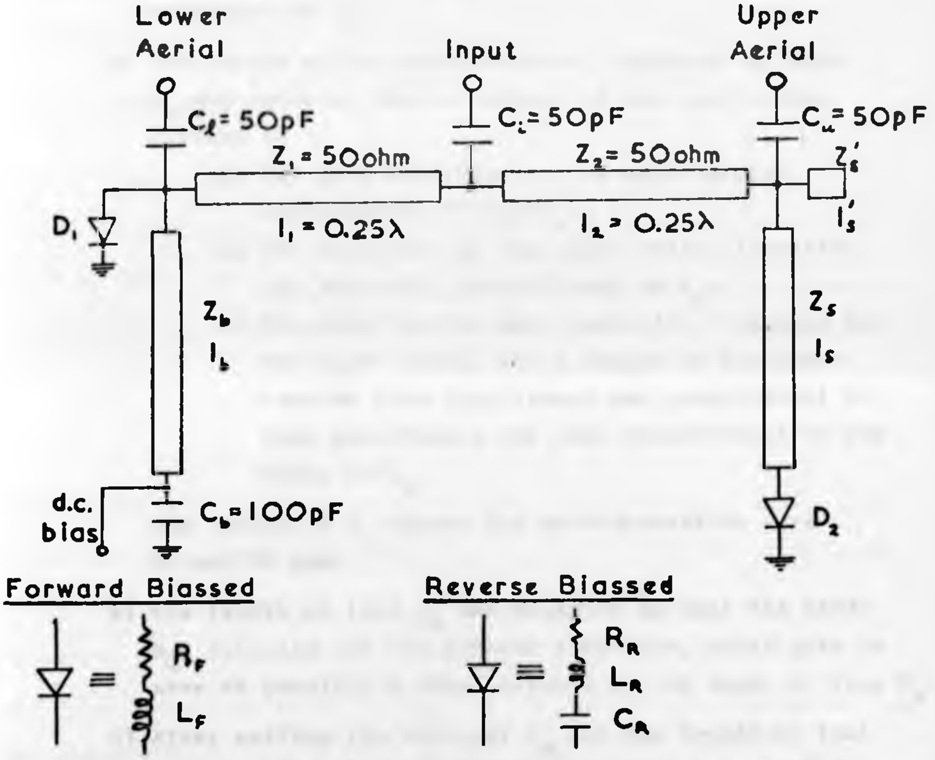
3) The frequency band over which the circuit must operate means that the lines are not all a constant length, in wavelengths, over the frequency band.

4) The bias capacitor, C_b, will have a finite reactance.

5) The low frequency decoupling capacitors C_l, C_i and C_u will not have a zero reactance at r.f..

A.1.3. NEW DESIGN PROCEDURE

To redesign the M.I.C. switch a synthesis procedure was used, with the help of a remote teletype time sharing TELCOMP(A.1.4) terminal, to look at the performance of each part, in turn, of the switch design. From this work a new switch design, Fig. A.1.2, was developed incorporating the following points to improve



Model 2A

Z_b = 75ohm , Z_s = 50ohm , Z_s' = 50 ohm ,
l_b = 0.211λ , l_s = 0.2237λ, l_s' = 0.0263λ

Model 2B

Z_b = 75ohm , Z_s = 25ohm , Z_s' = 50ohm ,
l_b = 0.213λ , l_s = 0.2368λ, l_s' = 0.0179λ

Fig. A.1.2.- New M.I.C. L-Band Switch Equivalent Circuit

the circuit performance at r.f. :-

1) The susceptance of diode D_1 in the reverse biassed state was balanced out by the input reactance of line Z_b adjusted in length to give this effect.

2) Line Z_b was shortened slightly to accommodate the reactance of C_b.

3) The choice of the characteristic impedance of line Z_s was selected from a balance of two conflicting factors :-

   a) The peak isolation on the upper aerial decreased by $40*\log(Z_s)$.

   b) The bandwidth of the upper aerial isolation was inversely proportional to Z_s.

   c) The shift in the peak isolation frequency for the upper aerial for a change in the diode reverse bias capacitance was proportional to this capacitance but also proportional to the value of Z_s.

   The values of Z_s chosen for experimentation were 25 and 50 ohm.

4) The length of line Z_s was adjusted so that the diode D_2, allowing for its reverse reactance, would give as near as possible a short circuit at the input of line Z_s.

5) After setting the value of Z_s and the length of that line,for the diode D_2 forward biassed, the input to line Z_s still contained a small susceptance. This was balanced out by the input susceptance of the stub line Z_s'.

6) The decoupling capacitors C_1, C_i and C_u should be as close as possible to a quarter wavelength apart, i.e. as close as possible to the ends of lines Z_1 and Z_2, so that the effects of there reactances will cancel out. Also these capacitors should be as large as possible, provided they do not effect the switching of the bias for the diodes D_1 and D_2.

A.1.4. SWITCH PERFORMANCE

The new design of the M.I.C. L-band switch was finally analysed using the MKMcPH computer program and the results of the circuit analysis of the switch are shown in Fig. A.1.3 to A.1.9. A summary of the circuit performance is also given in Table A.1.1.

A.1.5. CONCLUSIONS

Initially the results of the circuit analysis on the MKMcPH computer program for the old circuit design of the switch was found to give the same results as the MACAP computer program. In addition to this the new circuit design was found to give a much better circuit performance.

In the new circuit design the performance of the switch was improved by including the inductance of the bias line Z_b and the short stub line Z_s' in the circuit to match out the mismatches in the insertion loss for the two states of the circuit. In the isolation loss states these matching elements are short circuited and thus have no effect. The new circuit gave a performance within the specification for a stub line impedance, Z_s, of 25 to 50 ohm but the specification was not met, for the reverse bias isolation and V.S.W.R., for a stub line of 50 ohm and a reverse bias capacitance of the diodes of 0.3 to 0.7 pF.

At present the best stub line impedance, Z_s, would appear to be about 30 ohm but if the line losses are included, and these are appreciable, then it may be necessary to increase this impedance to about 40 ohm (or maybe 50 ohm) to reduce the insertion loss of the upper aerial although the isolation of this aerial would also be reduced. Suitable impedances and lengths for the lines Z_s and Z_s' are given in Table A.1.2.

Stub line impedance, Z_s	25 ohm	50 ohm
Lower Aerial Isolation, R_{F1}=0.2 ohm R_{F1}=1.0 ohm	36 to 37 dB 32 to 33 dB	36 to 37 dB 32 to 33 dB
Lower Aerial Insertion, C_{R1}=0.5 pF C_{R1}=0.3 to 0.7 pF	0.03 dB 0.04 dB	0.05 dB 0.06 dB
Upper Aerial Insertion, R_{F2}=0.2 ohm R_{F2}=1.0 ohm	0.16 dB 0.44 dB	0.10 dB 0.18 dB
Upper Aerial Isolation, C_{R2}=0.5 pF C_{R1}=0.3 to 0.7 pF	30 dB 26 dB	24 dB 18 dB
Input V.S.W.R., diodes forward bias	1.13	1.05
Input V.S.W.R., diodes reverse bias C_R=0.5 pF C_R=0.3 to 0.7 pF	 1.08 1.10	 1.10 1.17

N.B. Frequency band for results 1.02 to 1.10 GHz and the worst case results are given in the table

Table A.1.1. - Summary of Circuit Performance for L-band M.I.C. Switch

Z_s ohm	l_s wavelengths	Z_s' ohm	l_s' wavelengths
25	0.2368	50	0.0179
30	0.2342	50	0.0204
40	0.2289	50	0.0241
50	0.2237	50	0.0241

Table A.1.2. - Suitable Values for the lines Z_s and Z_s' in the New Switch Design in Fig. A.1.2.

Fig. A.1.3.-Lower Aerial Isolation for M.I.C. Switch with Diodes Forward Biassed

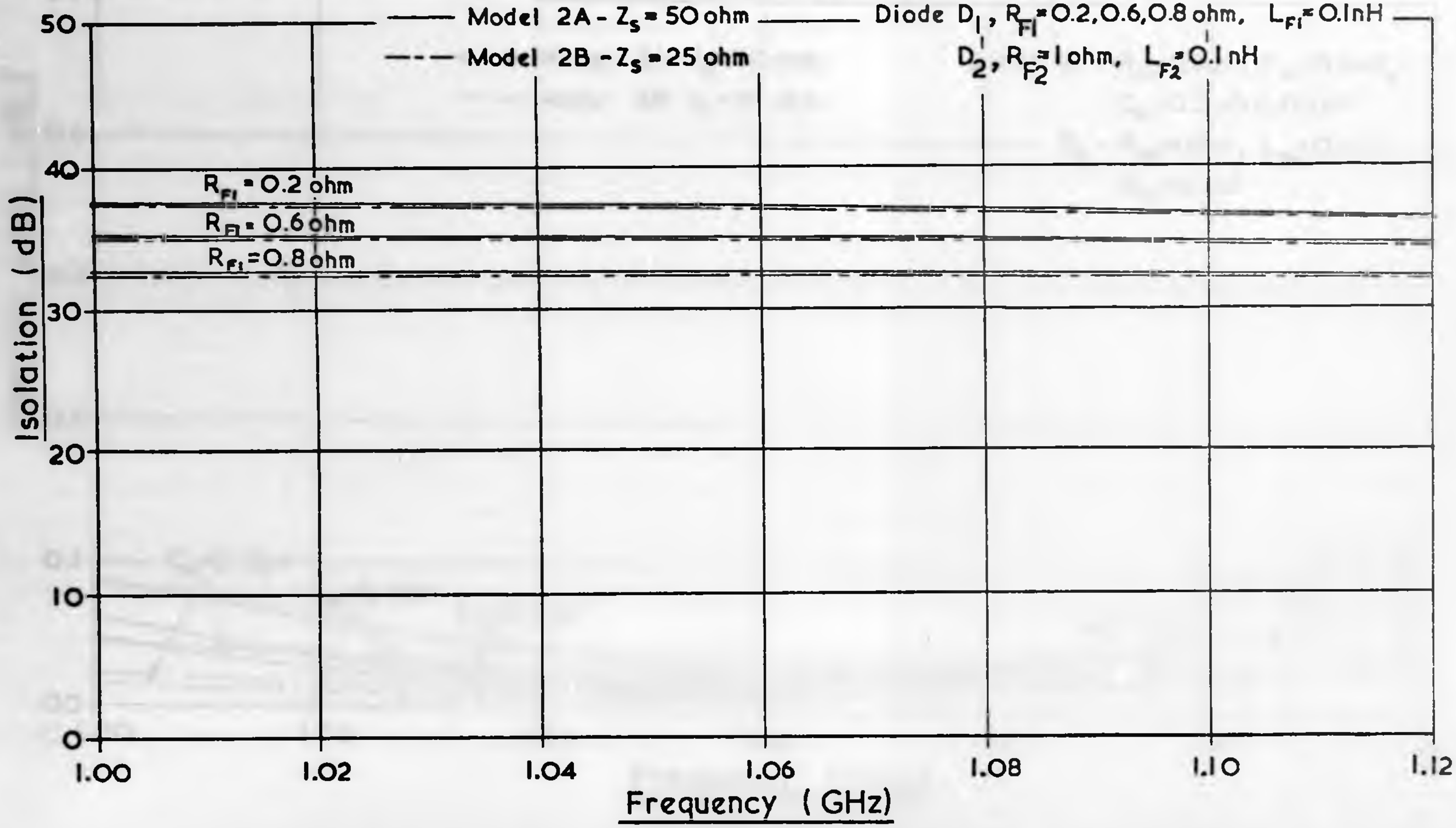


Fig. A.I.4.- Lower Aerial Insertion Loss for M.I.C. Switch with Diodes Reverse Biassed

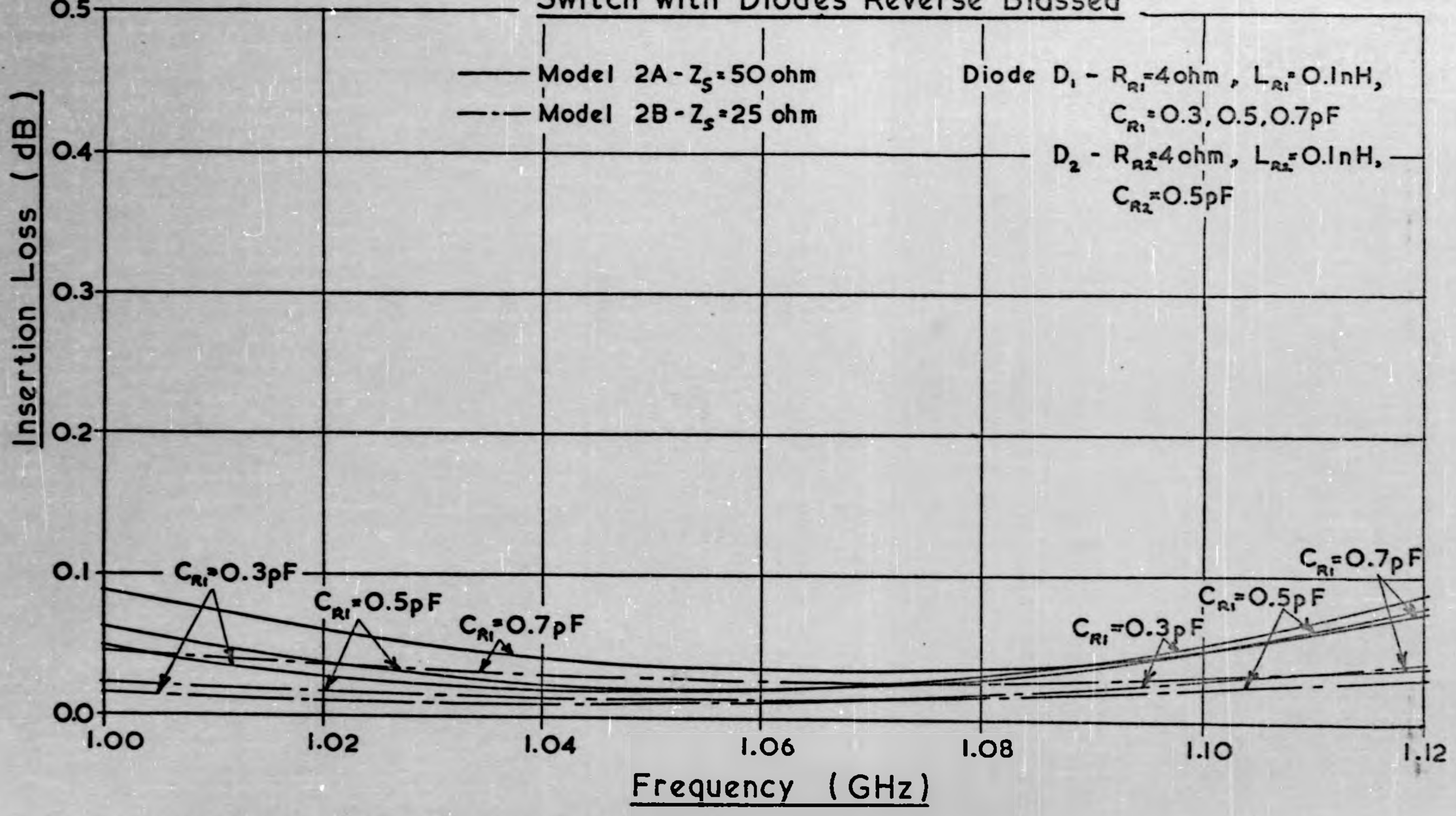


Fig. A.I.5. - Upper Aerial Insertion Loss for M.I.C. Switch with Diodes Forward Biassed

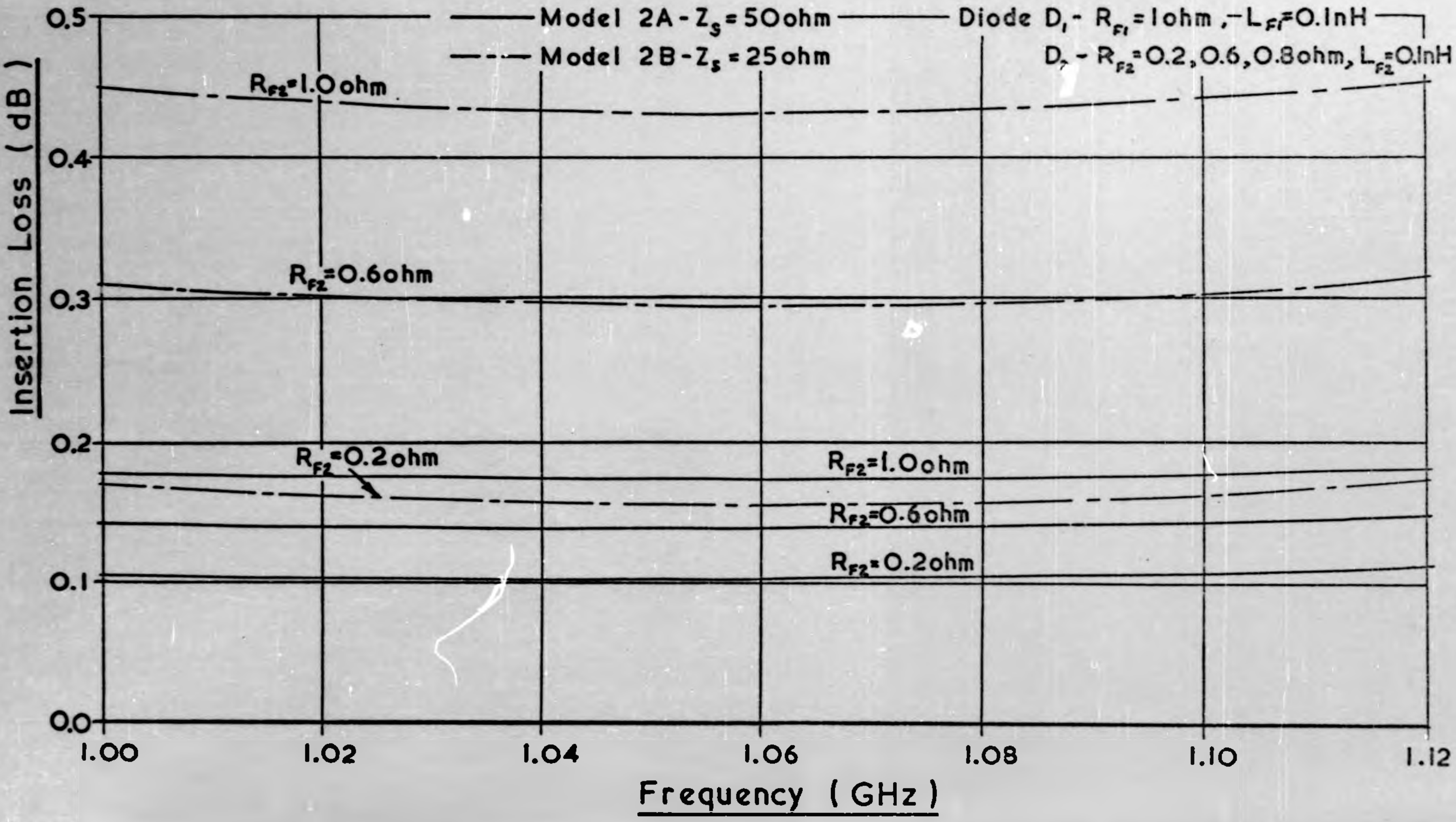


Fig.A.1.6.-Upper Aerial Isolation for M.I.C. Switch with Diodes Reverse Biassed

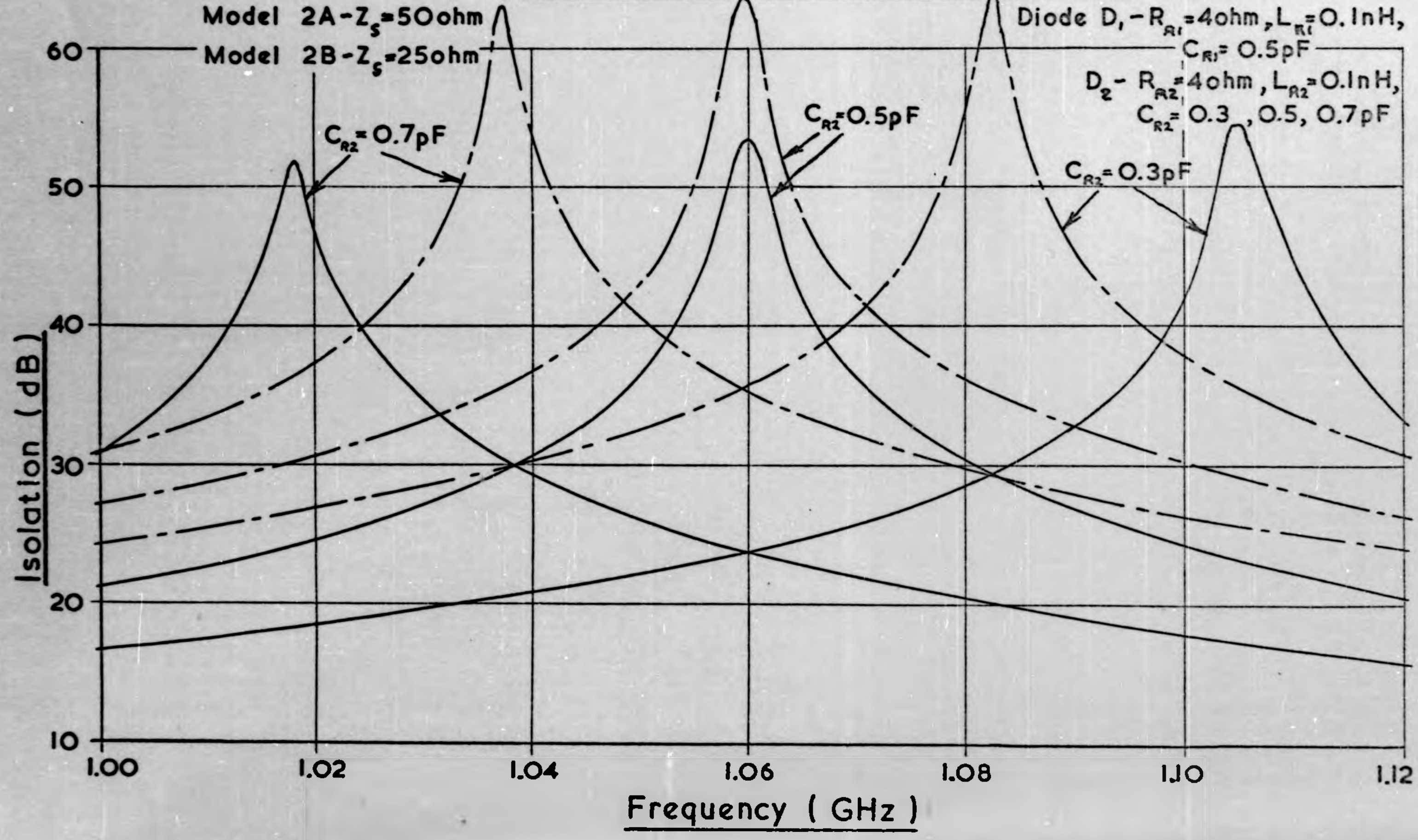


Fig.A.I.7. - Input V.S.W.R. for M.I.C. Switch with Diodes Forward Biassed

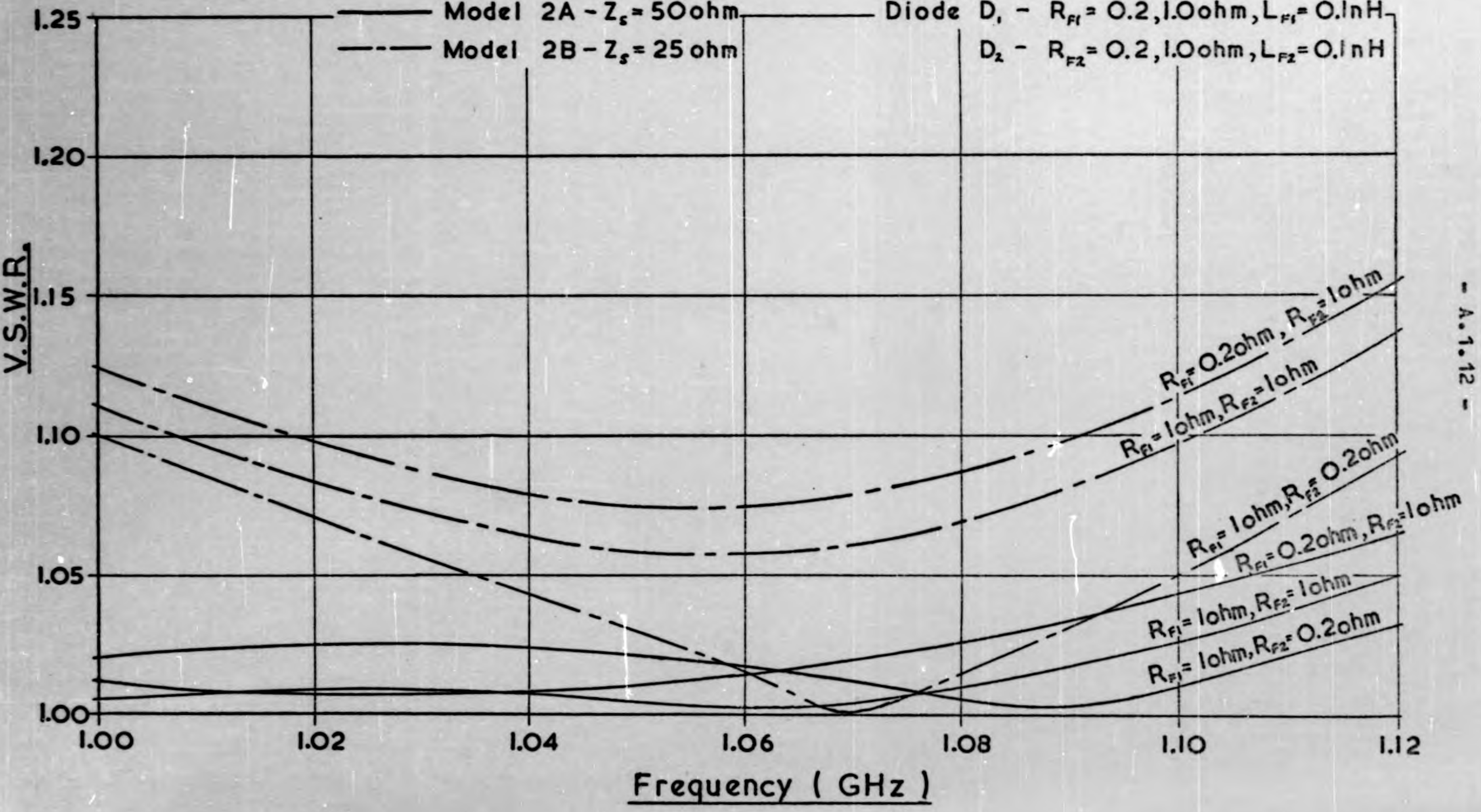


Fig. A.1.8.-Input V.S.W.R. for M.I.C. Switch with Diodes Reverse Biassed

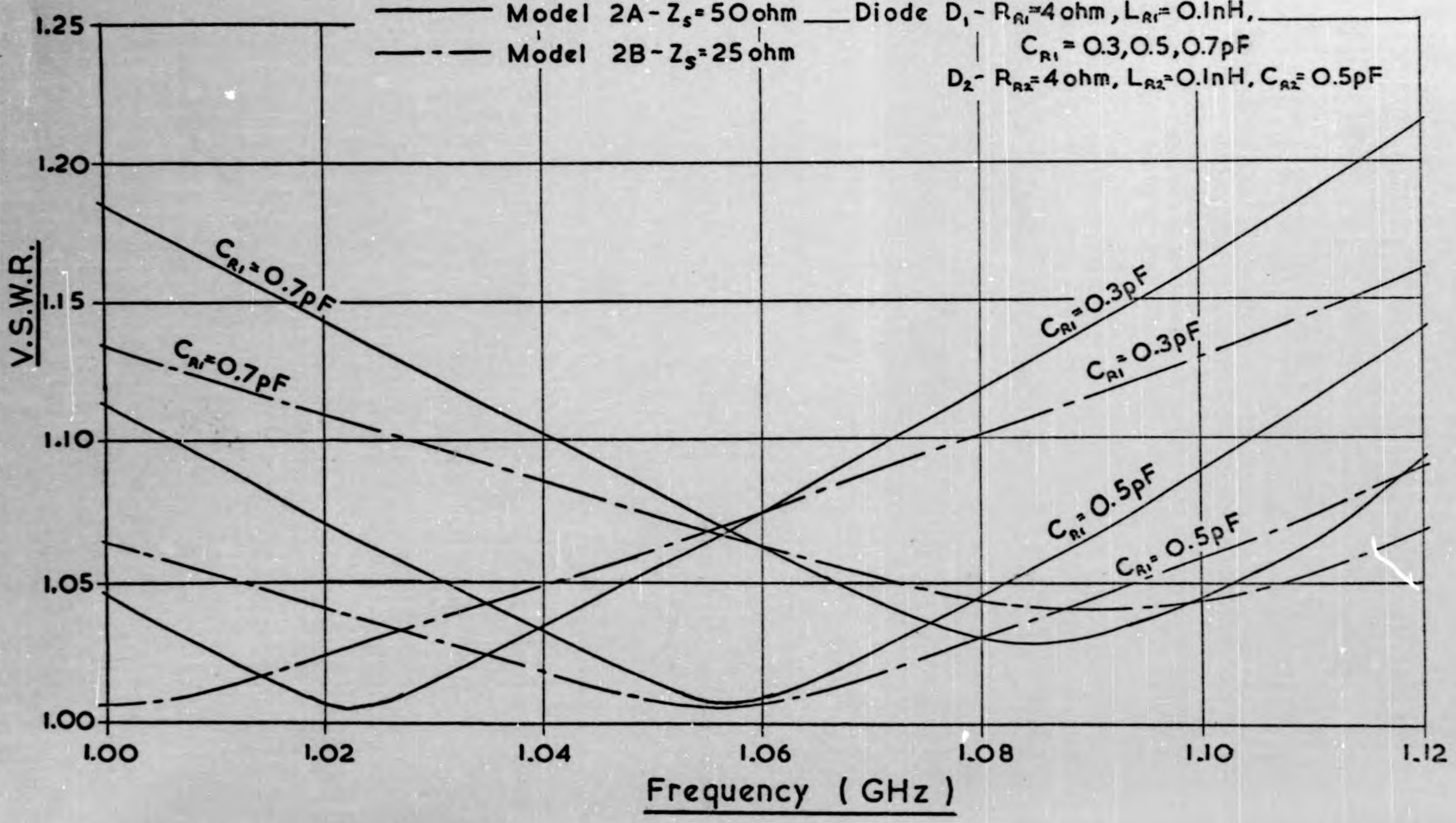
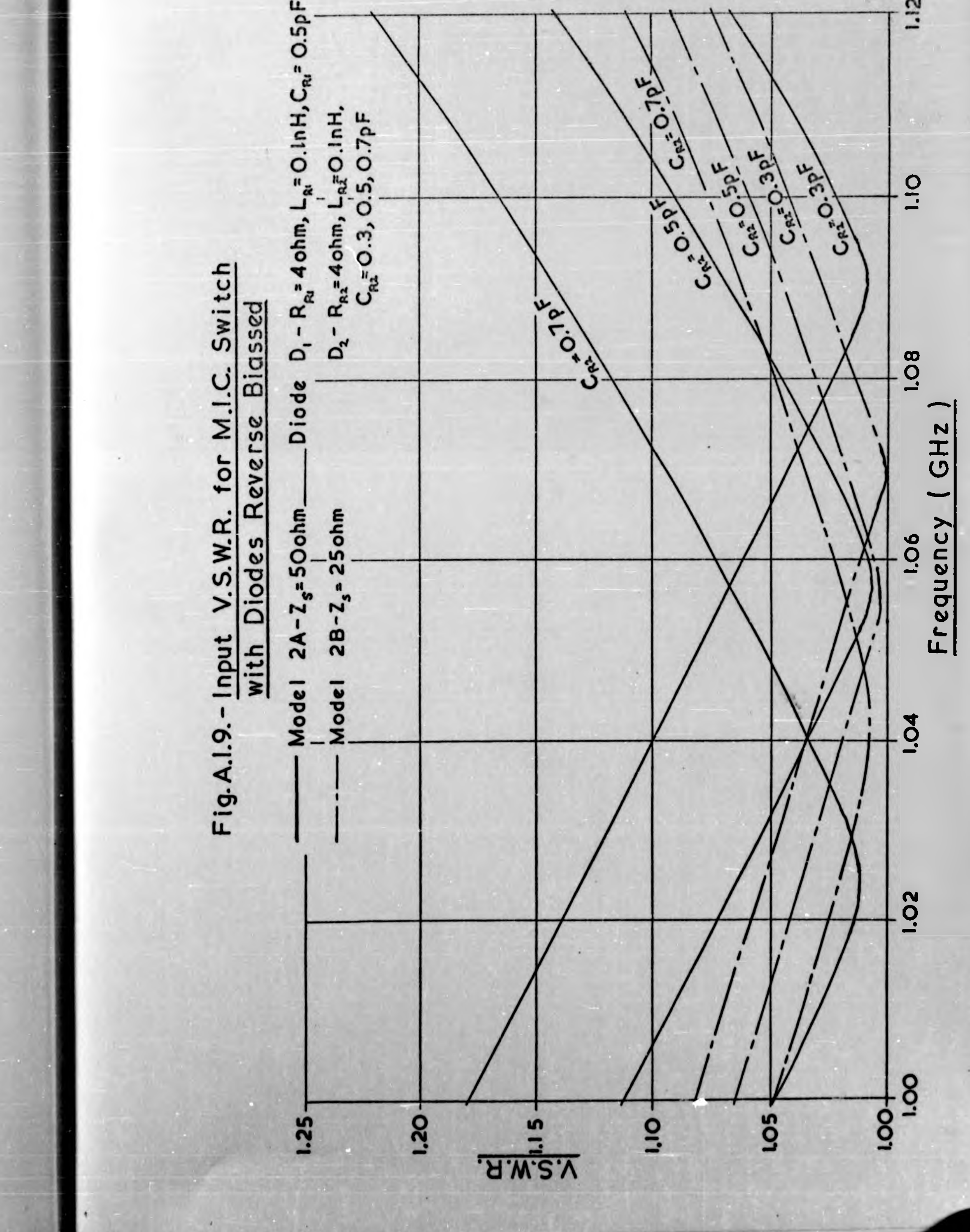


Fig.A.I.9.- Input V.S.W.R. for M.I.C. Switch with Diodes Reverse Biassed



References

A.1.1) B.G Marchent, 'Computer Aided Design of an L-Band M.I.C. Switch', Microwave Associates Ltd., Luton, 1969, Technical Report No. 160

A.1.2) 'MACAP Circuit Analysis Program', Microwave Associates Ltd., Burlington, U.S.A.

A.1.3) M.K. McPhun, 'A Computer Program for the Analysis of Branched Distributed and Lumped Circuits', IEE Conference Publication No. 23, 1966, pp. 89-124

A.1.4) 'TELCOMP Time Sharing UsersManual'

Appendix A.2

Comparative Study of Integrated Circuit Phase Shifters

Appendix A.2

Comparative Study of Integrated Circuit Phase Shifters

A.2.1. INTRODUCTION

At the start of the research work for this Ph. D. thesis 5 months were spent working at Microwave Associates Ltd., Luton. This Appendix describes the second of two design studies(A.2.1) carried out during this period.

There are at present three basic methods which may be used in the design of a microwave integrated circuit (M.I.C.) phase shifter. These methods are the iterative (Fig. A.2.1), hybrid ring (Fig. A.2.4) and the switched line (Fig. A.2.9) digital phase shifters. The objective of the design study described in this Appendix was to look at the design and performance of these three types of phase shifters and to compare the performance which could be obtained from each type of phase shifter. In this work first a synthesis of the design of each type of phase shifter was carried out and after this the performance of each type of phase shifter, in a practical circuit, was obtained using the computer program written for this work, i.e. the BGMA computer program(A.2.2).

In this Appendix all values are based on a per unit system (p.u.) and for a practical circuit it is necessary to multiply by the appropriate centre design values, i.e. for a system designed for a centre frequency of 2.3GHz and a characteristic impedance of 50 ohm all the frequencies must be multiplied by 2.3GHz and all the resistances/impedances must be multiplied by 50 ohm .

A.2.2. ITERATIVE PHASE SHIFTER

The iterative phase shifter(A.2.3,A.2.4), Fig. A.2.1, consists of a short length of transmission line, approximately quarter wavelength long, with a shunt reactance across each end

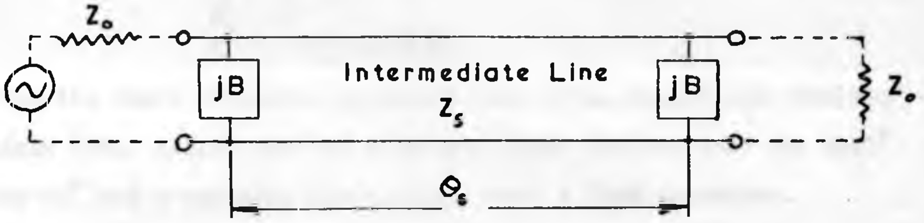


Fig. A.2.1. - Iterative Phase Shifter

of the line. To obtain the required phase shift the values of the shunt reactances must be stepped between two values as follows :-

1) ∞ to jX
2) ∞ to $-jX$
or 3) $-jX$ to jX.

From an investigation of the insertion loss bandwidth for various phase shift values it was decided that a shunt reactance stepped from $-jX$ to $+jX$ give the widest bandwidth. For this case, Fig. A.2.1, to obtain a phase shift of ϕ required :-

$$Z_s = Z_o * \cos(\tfrac{1}{2}\phi)$$
$$\phi_s = \tfrac{1}{2}$$
$$X = Z_o * \cot(\tfrac{1}{2}\phi)$$

From the input V.S.W.R., insertion loss (Fig. A.2.2) and absolute phase (Fig. A.2.3) for the iterative phase shifter only the 22.5° and 45° had a suitably low V.S.W.R. over a ±10% bandwidth.

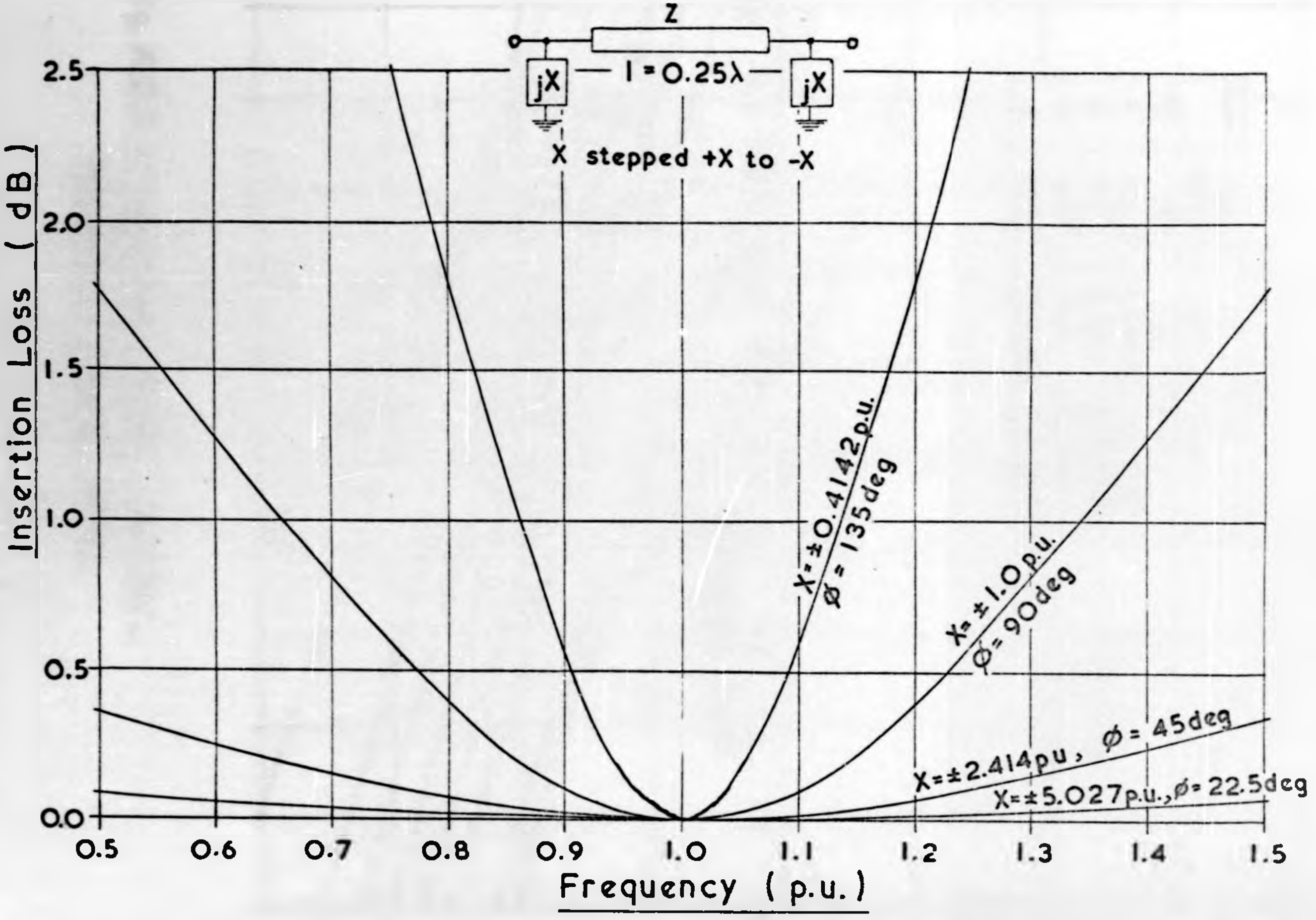
An investigation of the performance of a number of 22.5° and 45° practical iterative phase shift bits in cascade gave the following results for a ±10% bandwidth with line losses of 0.1dB/λ :-

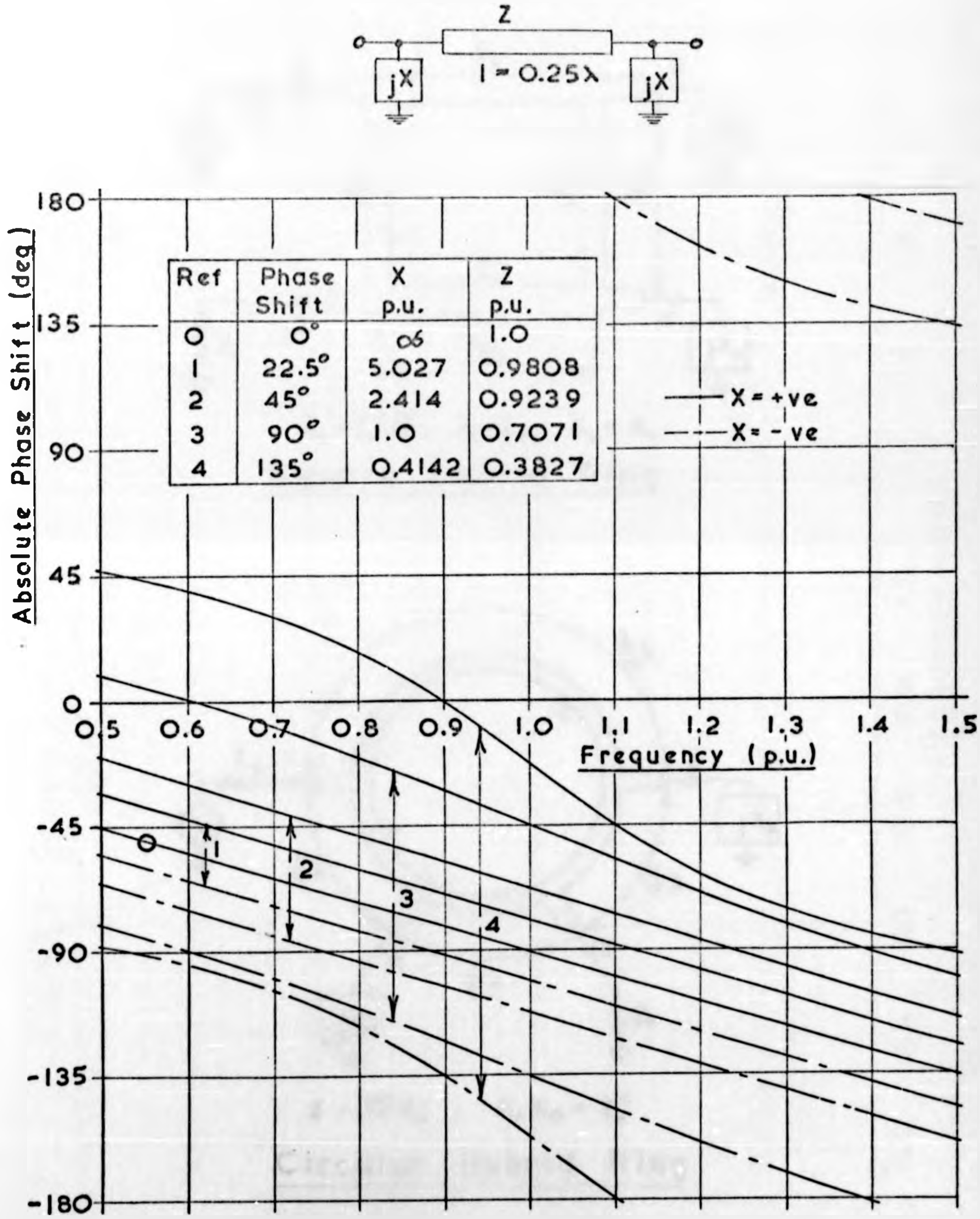
	22.5° bits	45° bits
V.S.W.R for any no. in cascade	1.20	1.30
insertion loss (mainly line losses)	0.04dB/bit	0.04dB/bit
phase shift error on switching	4.0°/bit	1.0°/bit

A.2.3. HYBRID RING PHASE SHIFTER

The two types of hybrid rings which may be used in a phase shifter are the square and circular hybrid rings, Fig. A.2.4. The hybrid ring in this case is a power splitter, i.e. with all the ports of the hybrid ring matched if power is feed into say port 1 then half the power will flow out of each of ports 2 and 4 whilst port 3 is isolated. From the basic performance of the square and circular hybrid rings, Fig. A.2.5 and A.2.6, it was easy to see

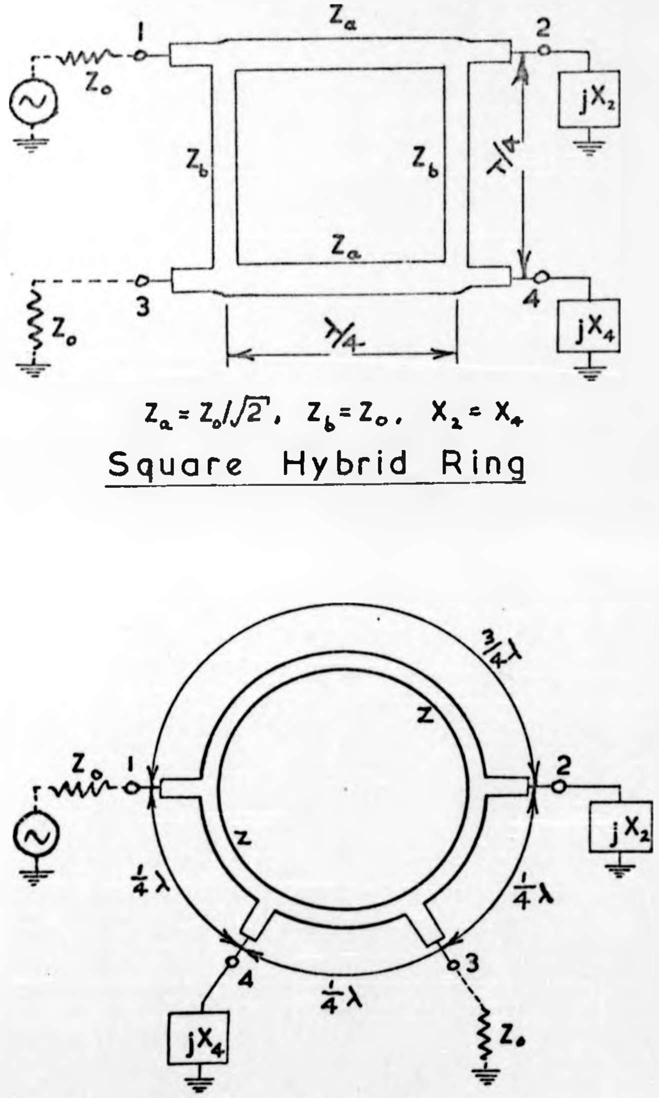
Fig.A.2.2 -Iterative Phase Shifter, Insertion Loss





Ref	Phase Shift	X p.u.	Z p.u.
0	0°	∞	1.0
1	22.5°	5.027	0.9808
2	45°	2.414	0.9239
3	90°	1.0	0.7071
4	135°	0,4142	0.3827

Fig. A.2.3.- Iterative Phase Shifter, Absolute Phase Shift



$Z_a = Z_o/\sqrt{2}$, $Z_b = Z_o$, $X_2 = X_4$

Square Hybrid Ring

$Z = \sqrt{2}\,Z_o$, $X_2 X_4 = Z_o^2$

Circular Hybrid Ring

Fig. A.2.4. - Hybrid Ring Phase Shifter

Fig.A.2.5.-Square 3dB Hybrid Ring, Basic Response

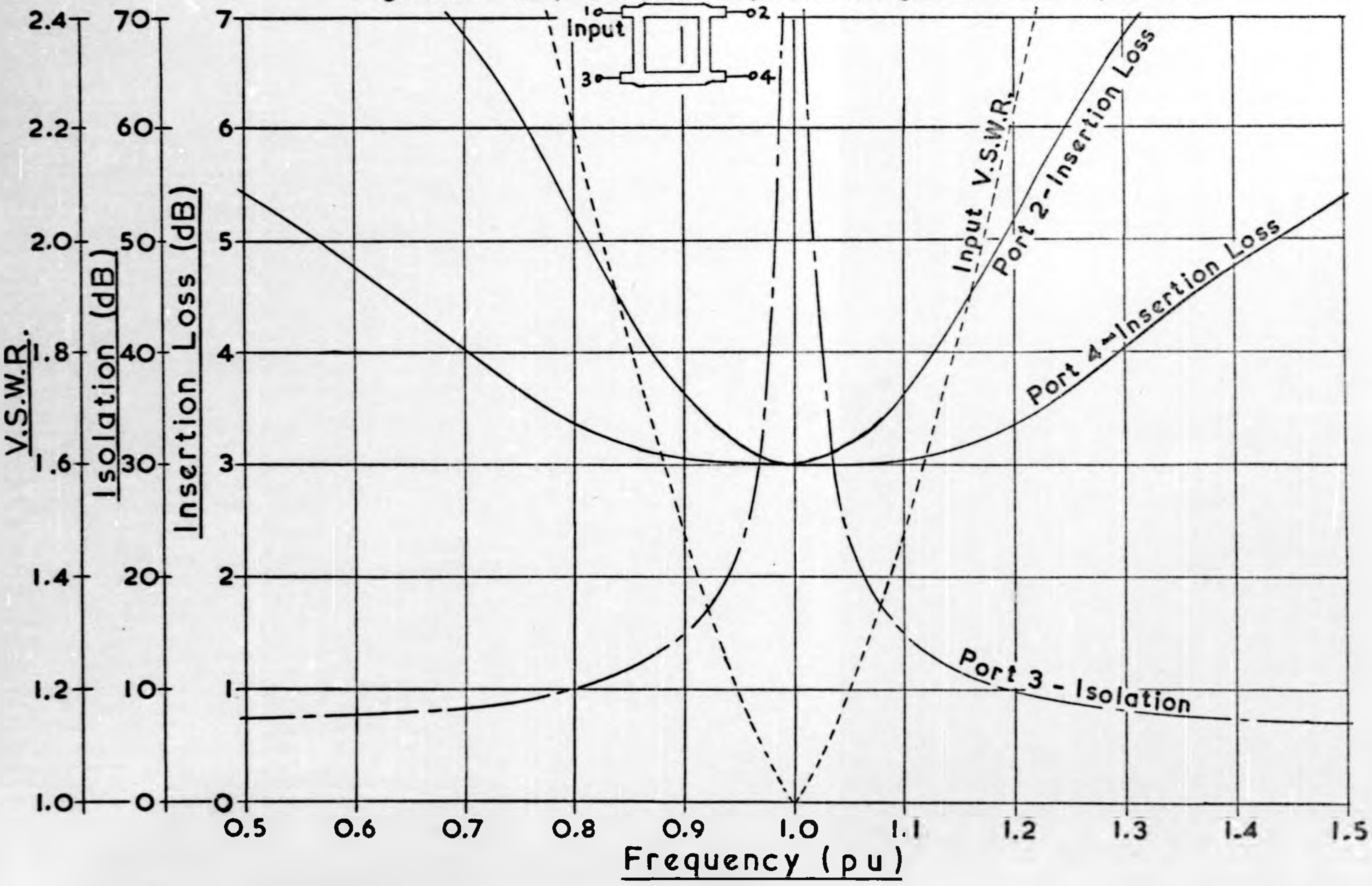
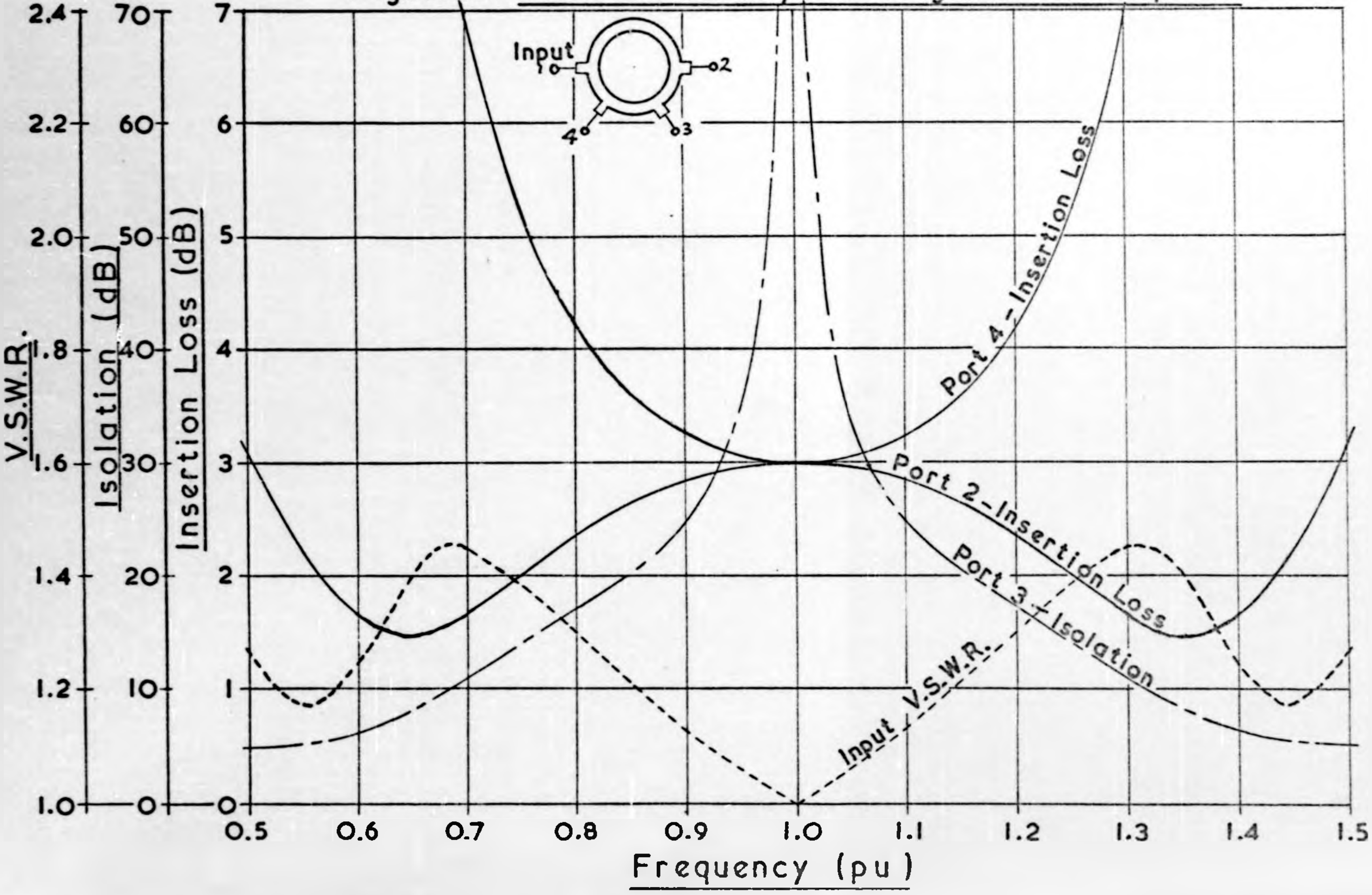


Fig.A.2.6.-Circular 3dB Hybrid Ring , Basic Response



that the circular hybrid ring had a far better performance over a frequency band.

The hybrid ring phase shifter was formed by terminating ports 2 and 4 in a shunt reactance stepped between two values, i.e. a circuit containing a p.i.n. diode switched between its forward and reverse biassed states to obtain the two reactances. The values for X_2 and X_4, Fig. A.2.4, must be set as follows for matching :-

1) Square hybrid ring -

$$X_2 = X_4$$

$$S_{31} = \frac{1}{X_2^2 + Z_o^2} (- 2 * X_2 * Z_o + j (X_2^2 - Z_o^2))$$

2) Circular hybrid ring -

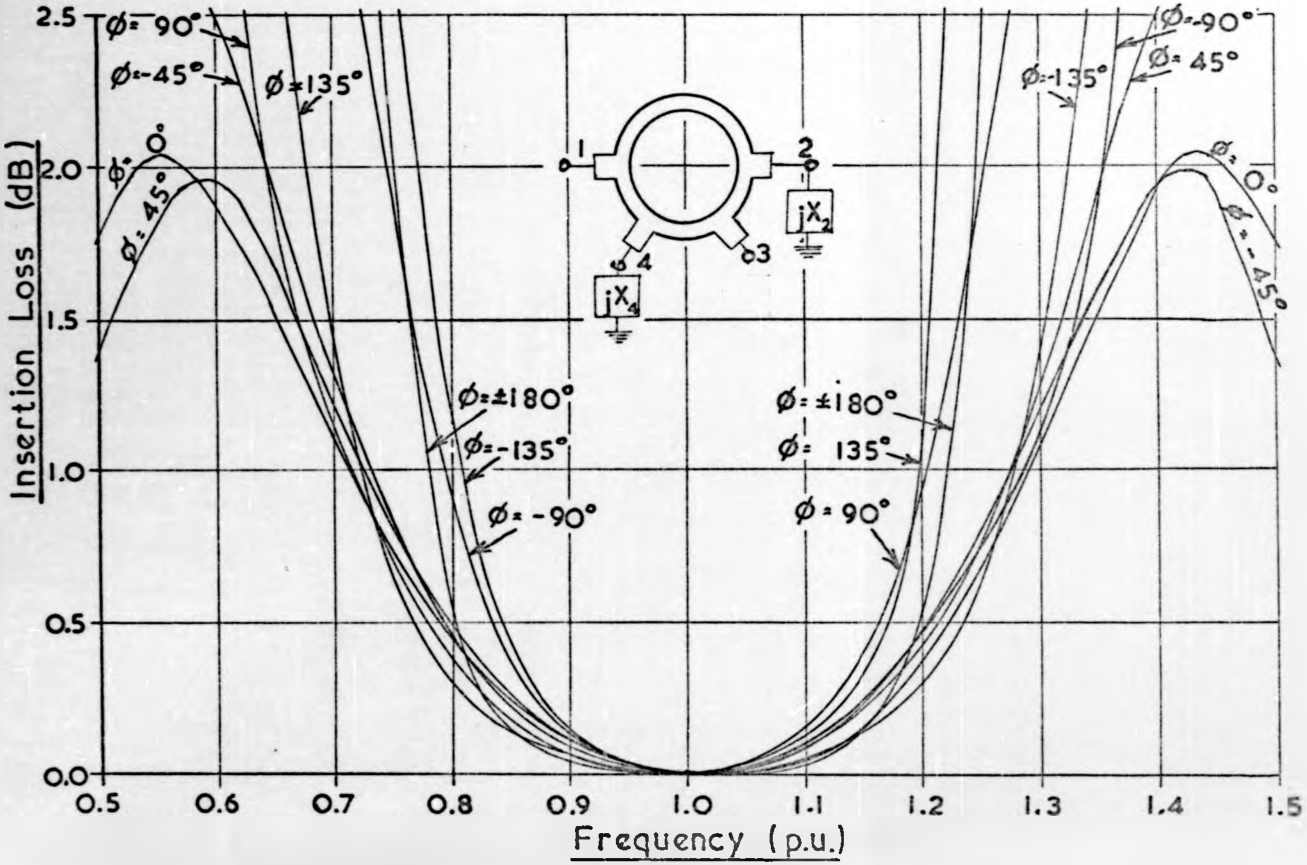
$$X_2 * X_4 = - Z_o^2$$

$$S_{31} = \frac{1}{X_2^2 + Z_o^2} ((X_2^2 - Z_o^2) + j \, 2 * X_2 * Z_o)$$

From these equations two values of X_2 can be selected to produce the change in phase shift to give two different arguements for S_{31} giving the required phase shift.

From the performance curves of the insertion loss, Fig. A.2.7, and absolute phase, Fig. A.2.8, (shown in the figures for the circular hybrid ring only) the circular hybrid ring gave a much wider band width. It gave a more constant phase shift with frequency over this band width but the absolute value of the phase of the power transmitted changed very rapidly with frequency. From comparison with other types of phase shifters it was decided that the circular hybrid ring phase shifter would be best for the larger values of phase shift, e.g. 90° to 180°, provided the rapidly changing absolute phase was acceptable.

Fig.A.2.7.-Circular Hybrid Ring Phase Shifter, Insertion Loss



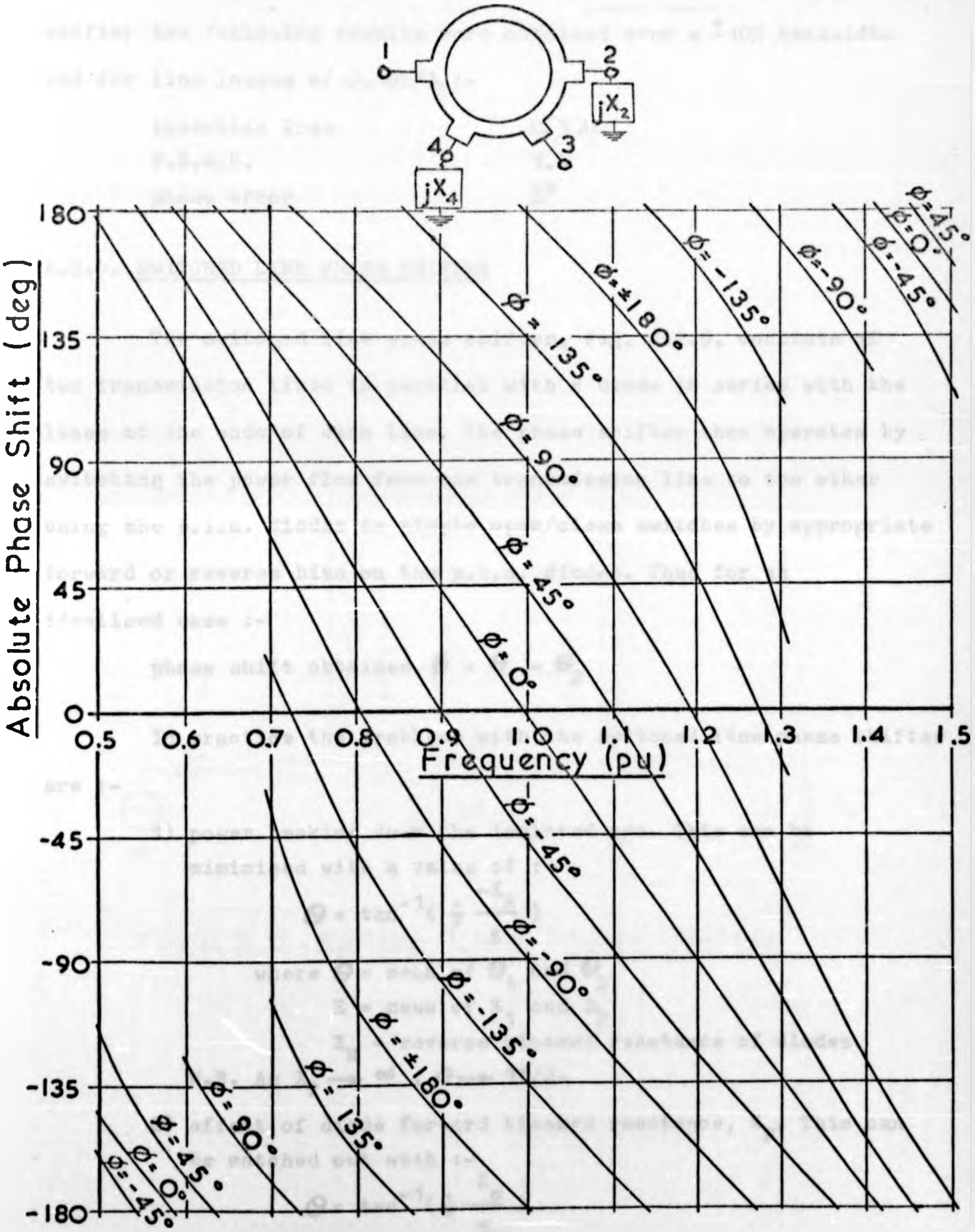


Fig.A.2.8.-Circular Hybrid Ring Phase Shifter, Absolute Phase Shifter

From the practical design of a circular hybrid ring phase shifter the following results were obtained over a ±10% bandwidth and for line losses of 0.1dB/λ :-

insertion loss	0.3 dB
V.S.W.R.	1.4
phase error	8°

A.2.4. SWITCHED LINE PHASE SHIFTER

The switched line phase shifter, Fig. A.2.9, consists of two transmission lines in parallel with a diode in series with the lines at the ends of each line. The phase shifter then operates by switching the power flow from one transmission line to the other using the p.i.n. diodes as simple open/close switches by appropriate forward or reverse bias on the p.i.n. diodes. Thus for an idealised case :-

phase shift obtained, $\phi = \theta_1 - \theta_2$

In practice the problems with the switched line phase shifter are :-

1) power leaking down the isolated arm. This can be mimimised with a value of :-

$$\theta = \tan^{-1}\left(\tfrac{1}{2} \frac{-X_R}{Z} \right)$$

where θ = mean of θ_1 and θ_2
Z = mean of Z_1 and Z_2
X_R = reverse biassed reactance of diodes

N.B. As $X_R \rightarrow \infty$, $\theta \rightarrow \pi/2$.

2) effect of diode forward biassed reactance, X_F. This can be matched out with :-

$$\theta = \tan^{-1}\left(\tfrac{1}{2} \frac{Z_o}{X_F} \right)$$

N.B. As $X_F \rightarrow 0$, $\theta \rightarrow \pi/2$.

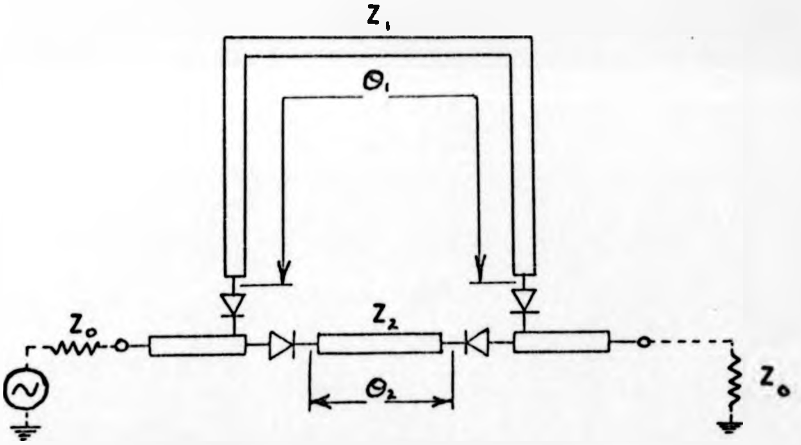


Fig.A.2.9.-Switch Line Phase Shifter

The mean length of θ_1 and θ_2 was set to quarter wavelength initially for an idealised case and then reduced to the minimum mean value given by 1) or 2) above.

The effects of the p.i.n. diodes reverse and forward reactances were plotted out separately and Fig. A.2.10 shows the effect of the diodes reverse reactance on the insertion loss whilst Fig. A.2.11 shows the effect of the diodes reverse reactances on the absolute phase error obtained from the expected absolute phase. From these results it was found that the highest acceptable diode reverse susceptance would be j0.2 p.u.. At this value the phase error is large but it would be similar for both arms of the phase shifter and thus the phase shift error would be <1.5° if all the diodes have similar reverse reactance values. From the consideration of the effect of the diodes forward reactance the maximum acceptable forward reactance for the diodes was found to be j0.2 p.u.. The phase error was again large but constant on both arms of the phase shifter and thus the phase shift error would be <1° if all the diodes have similar forward reactances.

A.2.5. CONCLUSIONS

The best type of iterative phse shifter is one in which the shunt reactance is stepped from -jX to +jX using phase shift bits of 22.5° or 45° for a ±10% bandwidth. The best type of hybrid ring phase shifter uses a circular hybrid ring for which the main uses are for the larger phase shift values of 90° and 180°.

The comparison of the practical 22.5° and 45° iterative phase shifters showed that the 45° bits gave a lower phase shift error and insertion loss than the 22.5° bits for the same phase

Fig.A.2.10.-Switched Line Phase Shifter, Effect of Isolated Arm

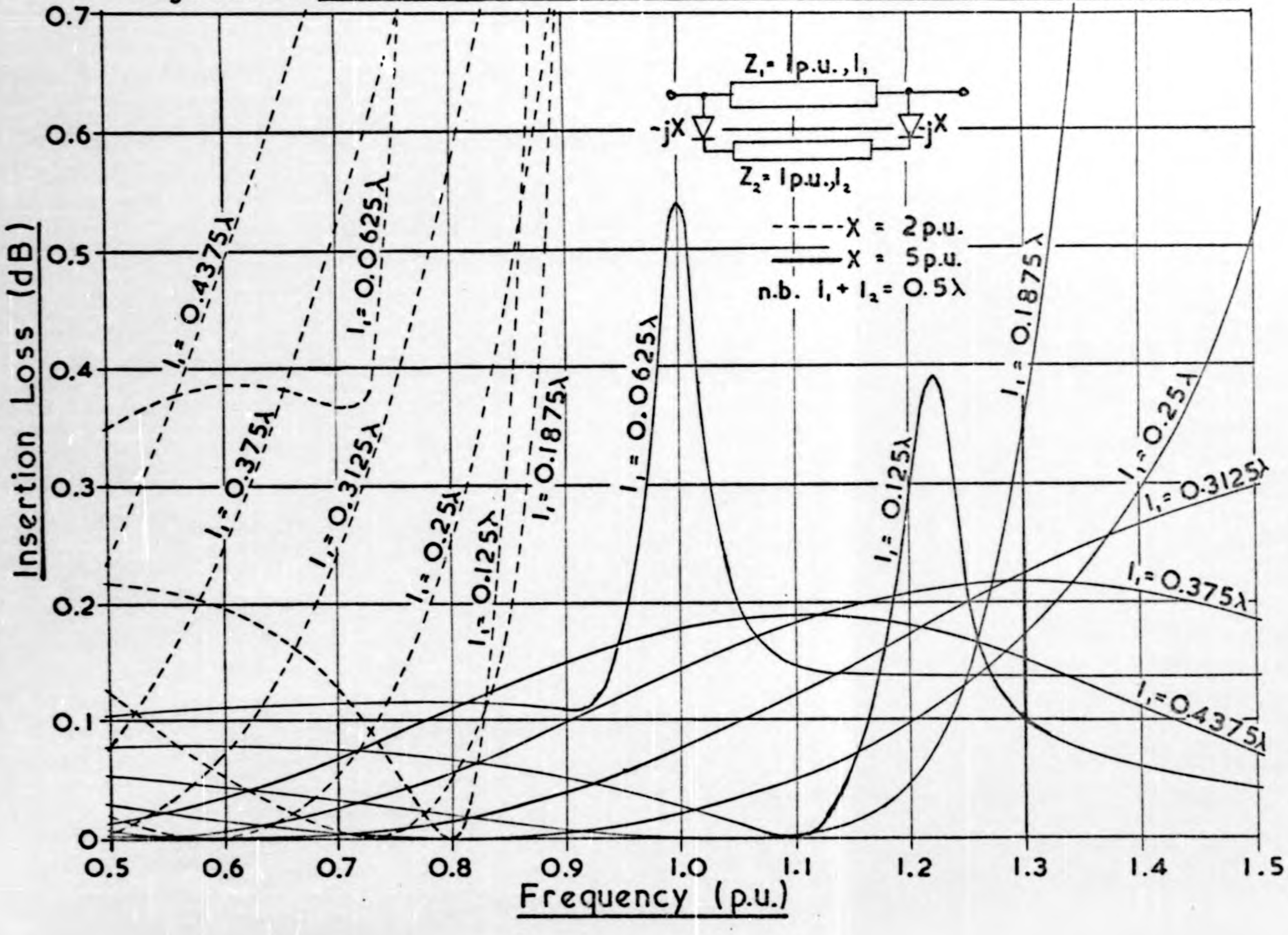
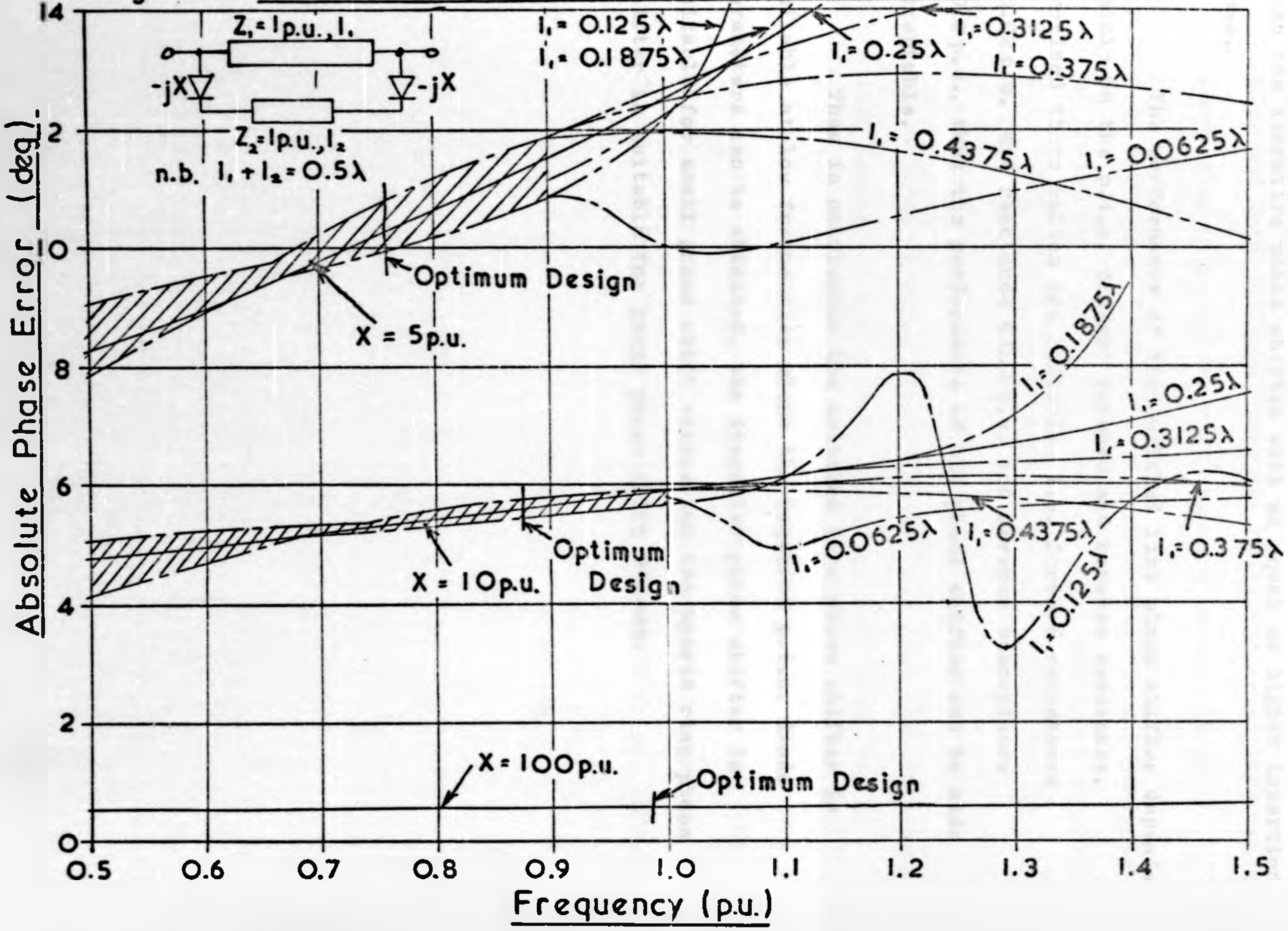


Fig.A.2.11.-Switched Line Phase Shifter, Effect of Isolated Arm



shift but gave a higher input V.S.W.R. The practical circular hybrid ring phase shifter gave a higher V.S.W.R. and phase error than the iterative phase shifter with an equal or higher insertion loss.

The performance of the switched line phase shifter depends mainly on the p.i.n. diodes' forward and reverse reactance. Provided these values are suitable, i.e. forward resistance <0.02 p.u. and reactance <0.2 p.u. and reverse susceptance <0.2 p.u., then the performance of the phase shifter can be made acceptable.

Thus in conclusion the switched line phase shifter is suitable at low frequencies where the required p.i.n. diode parameters can be obtained, the iterative phase shifter is suitable for small phase shift values and the hybrid ring phase shifter is suitable for large phase shift values.

References

A.2.1) B.G. Marchent, 'A Comparative Study of the Iteraitve, Hybrid Ring and Switched Line Microwave Phase Shifters', Microwave Associates Ltd., Luton, 1969, private communication

A.2.2) B.G. Marchent, 'A Computer Program (BGMA) for the Analysis of Lumped and Distributed Networks', University of Warwick, Nov. 1969, School of Engineering Science Report No. 50

A.2.3) F.L. Opp, W.F. Hoffman, 'Design of Digital Loaded-Line Phase Shift Networks for Microwave Thin Film Applications', IEEE Journal of Solid State Circuits, Vol. SC-3, No. 2, June 1968, pp. 124-130

A.2.4) J.F. White, 'High Power, p.i.n. Diode Controlled Microwave Transmission Phase Shifters', IEEE trans. Microwave Theory and Techniques, Vol. MTT-13, March 1965, pp. 233-242

Appendix A.3

Computation Times on Elliott 4130 Computer

A.3.1. COMPUTATION TIMES

A.3.1.1. Introduction

In this thesis it has often been necessary to compare computation times for various operations to decide on the most suitable method for the fastest computation time. The computer used in the research work was the Elliott 4130 computer with a 2μs store , i.e. this computer had floating point hardware. The figuares given below are the computation times obtained from the manuals for this machine(A.3.1). In practice they would not be the true times due to the large overheads and book-keeping involved but they do at least give an indication of the relative times between different methods for solving a given problem.

In the text of this thesis the abbreviations given below have to used in the CALC equations to define the operations required and the actual times given have been used in the TIME equations.

A.3.1.2. Integer Operations

Abbr.	Time	Oper.	Meaning
CLS_i	5	k:=0	Clear integer store
ASS_i	10	k:=i	Set k equal to i
$COMP_i$	12	$i<j$	Compare i to j
ADD_i	14	k:=i+j	Set k to sum of i and j
SUB_i	14	k:=i-j	Set k to i minus j
$MULT_i$	36	k:=i*j	Set k to product of i and j
DIV_i	43	k:=i/j	Set k to i divided by j

N.B. 1) i, j and k are all integers.

A.3.1.3. Real Operations

Abbr.	Time	Oper.	Meaning
CLS_r	11	$z:=0$	Clear real number in store
ASS_r	16	$z:=x$	Set z equal to x
$COMP_r$	21	$x < y$	Compare x to y
ADD_r	31	$z:=x+y$	Set z to sum of x and y
SUB_r	31	$z:=x-y$	Set z to x minus y
$MULT_r$	56	$z:=x*y$	Set z to product of x and y
DIV_r	86	$z:=x/y$	Set z to x divided by y

N.B. 1) x, y and z are all real numbers.

A.3.1.4. Complex Operations

The complex operation times given below were derived for the CMPLX program used in NEAT for the complex number operations on the Elliott 4130 computer$^{(A.3.2)}$.

Abbr.	Time	Oper.	Meaning
CLS_j	34	$c:=0$	Clear complex number in store
ASS_j	134	$c:=d$	Set c equal to d
ADD_j	229	$c:=d+e$	Set c to sum of d and e
SUB_j	233	$c:=d-e$	Set c to d minus e
INV_j	469	$c:=1/d$	Set c to inverse of d
$MULT_j$	445	$c:=d*e$	Set c to product of d and e
DIV_j	727	$c:=d/e$	Set c to d divided by e
$ZERO_j$	99	$c=0$	Set ZERO to zero if c equals 0
$AMAX_j$	126		Set AMAX to maximum modulus of real or imaginary part of c
MOD_j	814 (193)	$\|c\|$ $\|c\|^2$	Set MOD to modulus of c (for MOD squared)
DB_j	2650	$\|c\|$dB	Set DB to 10 $\log_{10}\|c\|^2$

N.B. 1) c, d and e are all complex numbers.
2) The value of ZERO is an integer number and the values of AMAX, MOD and DB are real numbers.

A.3.1.5. Mathematical Functions

See ref. A.3.3

Abbr.	Time	Oper.	Meaning
SQRT_r	605	$z:=\sqrt{x}$	Set z to the square root of x
SIN_r	1500	$z:=\sin(x)$	Set z to sin of x
COS_r	1500	$z:=\cos(x)$	Set z to cos of x
TAN_r	2100	$z:=\tan(x)$	Set z to tan of x
LOG_r	2400	$z:=\log(x)$	Set z to log to base 10 of x
EXP_r	1500	$z:=\exp(x)$	Set z to e^x
ARCS_r	2800	$z:=\sin^{-1}(x)$	Set z to the angle who's sin is x
ARCC_r	2800	$z:=\cos^{-1}(x)$	Set z to the angle who's cos is x
ARCT_r	2500	$z:=\tan^{-1}(x)$	Set z to the angle who's tan is x
SINH_r	173	$z:=\sinh(x)$	Set z to sinh of x (given e^x)
COSH_r	173	$z:=\cosh(x)$	Set z to cosh of x (given e^x)
EXP_j	4612	$c:=\exp(d)$	Set c to e^d
SINH_j	1072	$c:=\sinh(d)$	Set c to sinh of d (given e^d)
COSH_j	1068	$c:=\cosh(d)$	Set c to cosh of d (given e^d)

N.B. 1) z and x are real numbers.

2) c and d are complex numbers.

A.3.1.6. Array Operations

See ref. A.3.1

Abbr.	Time	Oper.	Meaning
ARR_1	26	A(i)	Access array element with 1 dimensions
ARR_2	47	A(i,j)	Access array element with 2 dimensions
ARR_3	68	A(i,j,k)	Access array element with 3 dimensions
EXROW	310		Swap row order of 2 rows in a 2 dimensional array

A.3.1.7. Data Structure Access

Abbr.	Time	Meaning
INREAL	42	Enter real number into data structure
REALOF	42	Read real number out of data structure
DPOINT	39	Find number of pointers in a bead
LENGTH	48	Find length of bead
PRIOR	48	Find priority of bead
KEY	64	Set up keyword or header word for bead

N.B. 1) These procedures are described in section 6.2.6. and the times given are the times after the procedures have been replaced by their equivalent NEAT code in the program using the ML1 program(A.3.4).

References

A.3.1) 4100 Technical Manual, ICL, Vol. 2 - Programming Information, Part 4 - NEAT Assembly Language, Section 1 - Programming Guide, Appendix 1 - Summary of Instructions

A.3.2) 4100 Technical Manual, ICL, Vol. 2 - Programming Information, Part 4 - NEAT and NICE, Section 6 - NICE Supplementary Routines, Chapter 9 - Complex and Double

A.3.3) 4100 Technical Manual, ICL, Vol. 2 - Programming Information, Part 8 - Library Routines, Section - Mathematical Subroutines MATH, p. 13

A.3.4) 'ML1 Users Manual', University Mathematical Laboratory, Cambridge University

Appendix A.4

User Manual for Chain Matrix Analysis CHAINI Program

A.4.1. INTRODUCTION

The CHAIN1 program will determine the steady state sinusoidal input to output response of a circuit composed of lumped linear elements and transmission lines each of which is described in a simple 2-port network. The complete circuit may consist of an assembly of any number of these simple networks connected between junctions in the circuit. Any two junctions may be assigned as the external input and output ports. The one condition is that it must be possible to find a path between the input and output ports of the circuit. On this path parallel paths, branch arms and loop paths may be built to any level. The only limitation to this is that an arm across opposite sides of a parallel path must not be included. Any branch arm may be terminated in a load consisting of a series or parallel R L C circuit and the junctions in the circuit can be set as series or parallel to describe the way in which the networks are interconnected at the junctions.

The reader may find it useful whilst reading the description of the use of the CHAIN1 program to refer to the simple examples of a circuit description, its input data and the results obtained on the CHAIN1 program in sections A.4.13 and A.4.14.

A.4.2. DECLARATION OF CIRCUIT SIZE

A.4.2.1. Main Title

The data must start with a title. This title must consist of capital letters, digits, spaces or line feeds only with up to 240 characters. This corresponds to 3 cards with 80 characters per card. This title must be terminated with a <;>.

e.g. DATA FOR CIRCUIT NUMBER 1 ;

The title is reproduced on the results and serves as a heading for the results.

A.4.2.2. Maximum Size of Circuit

The maximum size of the circuit must follow the main title. This consists of 4 integer numbers giving the maximum number of junctions, networks, loads and variables in that order in the following circuit description. These must all be greater than zero.

e.g. 10 20 5 3

The number of junctions, networks, loads, variables and ports in the circuit is printed out next in the results.

A.4.3. CIRCUIT TOPOLOGY STATEMENTS

A.4.3.1. Type of Circuit Acceptable

For a circuit to be acceptable it must be possible to split it up into paths. The form of the circuits which can be analysed by the program are shown in Fig. A.4.1 to A.4.8 and an assembly of any number of these path types built to any level is also acceptable. If it is uncertain whether a circuit can be analysed then it should be attempted and if it is not acceptable the CHAIN1 program will reject it and print out the reason for its rejection.

The type of circuit which is not acceptable is one in which a path between opposite sides of a parallel path is included. This is illustated in an example in Fig. A.4.9. If an analysis of the circuit in Fig. A.4.9 is required between junctions 2 and 4 then the circuit is acceptable as it consists of 3 paths in parallel but if an analysis of the circuit is required between junctions

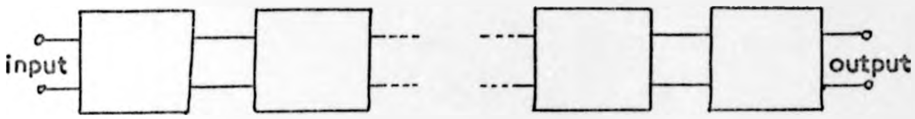


Fig. A.4.1. - Main Link Path

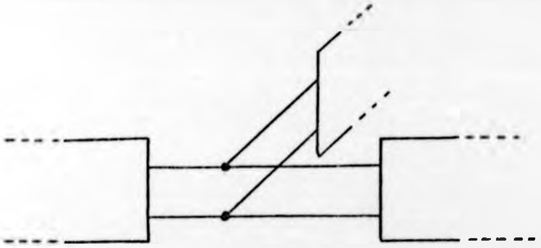


Fig. A.4.2.- Branch Arm on Parallel Junction

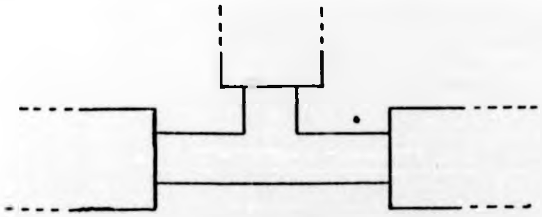


Fig. A.4.3. - Branch Arm on Series Junction

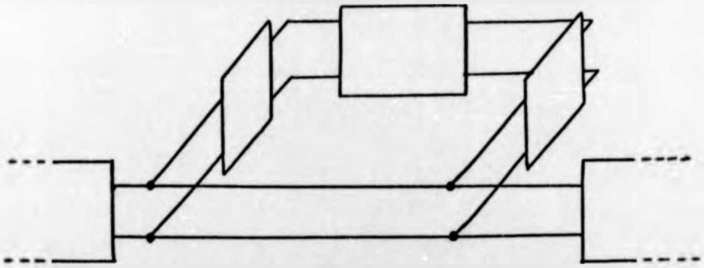


Fig. A.4.4.- Loop Path on Parallel Junction

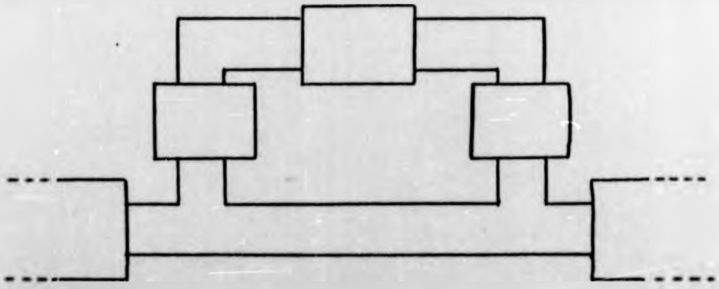


Fig. A.4.5.- Loop Path on Series Junction

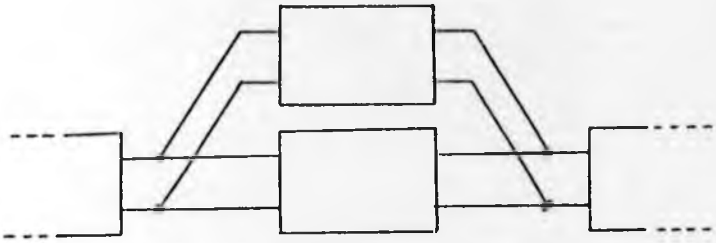


Fig.A.4.6.- Parallel Path, Parallel to Parallel Junction

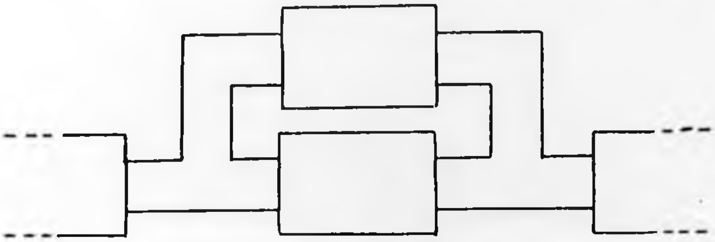


Fig.A.4.7. - Parallel Path, Series to Series Junction

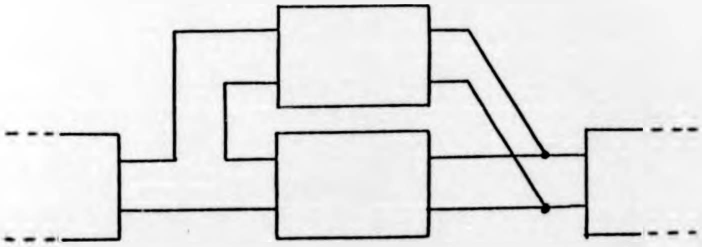


Fig.A.4.8.- Parallel Path, Series to Parallel Junction

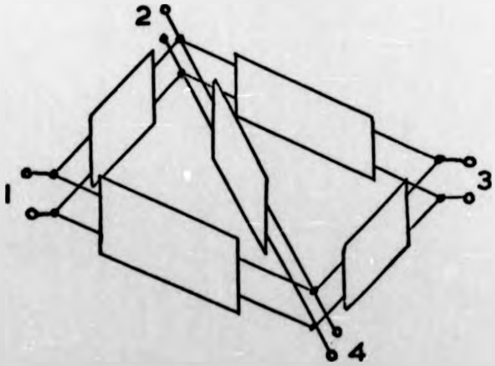


Fig.A.4.9.- Circuit with Possible Interactive Branch

1 and 3 then the circuit is not acceptable as it consists of 2 parallel paths with another path connected between the centres of these 2 paths.

In the case of a complicated circuit, using series and parallel junction connections it may be difficult to decide what the circuit actually consists of. It should be remembered that any current flowing into one terminal on any network must equal the current flowing out of the other terminal on the same port of the network. Also the program is only interested in the voltage differences between terminals on the same port of any network. In practice to realise the correct equivalent circuit it may be necessary to include an ideal transformer with unity turns ratio in cascade with all the ports on all the networks in the circuit.

A.4.3.2. Network Statement

The word NETWORK must be given followed by its number, the 2 junctions to which it is connected, the type of the network (see Table A.4.1) and the four parameters associated with that network. These four parameters are always read in even if all the parameters are not necessary. Provided the network is reciprocal then the junctions may be given in either order.

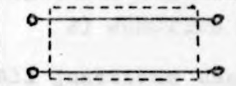
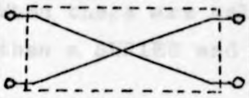
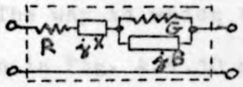
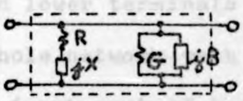
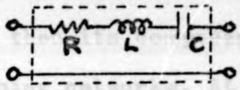
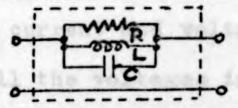
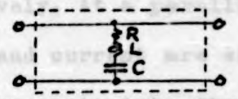
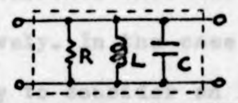
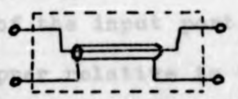
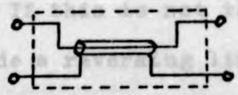
e.g. NETWORK 15 7 5 6 50.0 0.1 0.5 0.0

If it is required to delete a network previously used then one or both of its junction numbers must be set to zero. The network type and parameters are not then required.

e.g. NETWORK 11 0 2

A.4.3.3. Junction Statement

The word JUNCTION must be given followed by either the word SERIES or PARALLEL and the list of junctions required to be

Network type no.	Network	Parameter			
		1	2	3	4
1		*	*	*	*
2		*	*	*	*
3		R (ohm)	X (ohm)	G(1) (mmho)	B(1) (mmho)
4		R(2) (ohm)	X(2) (ohm)	G (mmho)	B (mmho)
5		R (ohm)	L (nH)	C(3) (pF)	*
6		R(4) (ohm)	L(4) (nH)	C (pF)	*
7		R (ohm)	L (nH)	C(3) (pF)	*
8		R(4) (ohm)	L(4) (nH)	C (pF)	*
9		Z_o (ohm)	λ_o (at f_o)	f_o (GHz)	α (dB/λ)
10		Z_o (ohm)	l (m)	ϵ_r (p.u.)	α (dB/m)

n.b. 1) If G and B are both zero they are assumed to be equal to a short circuit.

2) If R and X are both zero they are assumed to be equal to an open circuit.

3) If C is zero then it is assumed to be equal to a short circuit.

4) If R or L is zero then it is assumed to be equal to an open circuit.

Table A.4.1. - Network Types for the CHAIN1 Program

set to that state. This list must be terminated in a zero.

e.g. 1) JUNCTION SERIES 5 7 3 1 0
2) JUNCTION PARALLEL 9 4 7 2 0

N.B. 1) All junctions unset are assumed to be parallel
2) When there are only 2 networks connected on a junction then a SERIES and PARALLEL junction are both the same.

The way in which the networks are connected at a junction is shown in Fig. A.4.10 and A.4.11. To identify which are the upper and lower terminals of each network requires a consideration of the whole network path topology. Consider an incident wave from the input port of the circuit. If this wave reaches a multiway junction then its components of voltage and current are split between the outgoing networks. At a series junction, Fig. A.4.10, the incident current and voltage are equal to the current and the sum of all the voltages into the networks leaving that junction respectively. At a parallel junction, Fig. A.4.11, the incident voltage and current are equal to the voltage and the sum of all the currents into the networks leaving that junction respectively. In the case of the end of a parallel path it is necessary to consider an incident wave from the output port instead of the input port. Also the voltages and currents refer to the upper relative to the lower terminal on each side of a network. If this is not the case required then it is necessary to include a reversing link in the appropriate path.

A.4.3.4. Load Statement

In the complete circuit a load may be attached only to a junction with only one network connected to it, i.e. the end of a branch arm. If a load is connected to any other junction then it is completely ignored in the circuit analysis. An exception to this rule is the case of a load attached to the junctions assigned as

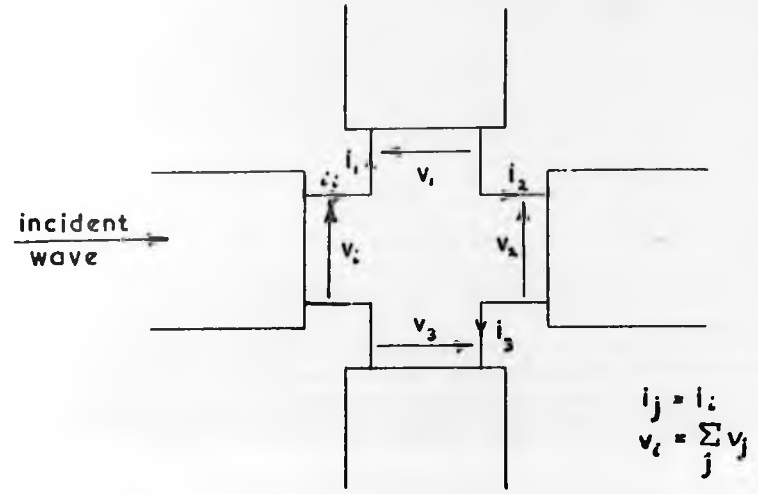


Fig. A.4.10. - Series Junction

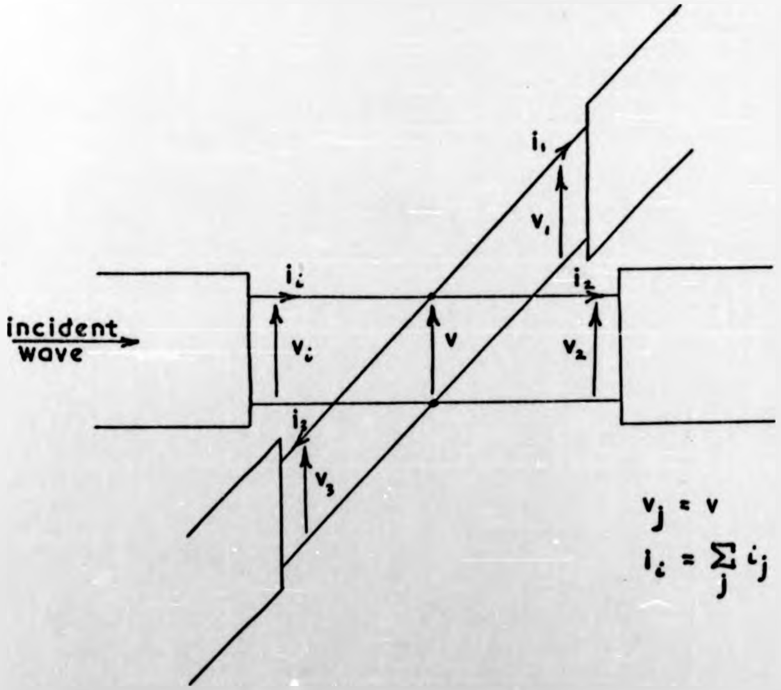


Fig. A.4.11. - Parallel Junction

the input and output ports of the circuit. In this case these loads are used for the characteristic impedances associated with the ports of the circuit.

Series RLC Load - The word LOAD must be given followed by the load number and the junction to which it is connected. Next the word SERIES must be given and the load resistance in ohms, inductance in nH and capacitance in pF. If no capacitance is required, i.e. a short circuit in its place, then a capacitancc of 0 should be given.

```
e.g. LOAD 4  7  SERIES 40.0  0.5  0.9
```

Parallel RLC Load - This is the same as a series load except that the word SERIES is replaced by the word PARALLEL. In this case if no resistance or inductance is required, i.e. an open circuit in its place, then the value of this parameter should be given as 0.

```
e.g. LOAD 2  9  PARALLEL  30.0  0.1  0.7
```

Short Circuit Load - The word LOAD must be given followed by the load number and the junction to which it is connected. Next the word SC must be given followed by 3 dummy parameter values.

```
e.g. LOAD 5  2  SC  0.0  0.0  0.0
```

Open Circuit Load - This is the same as the short circuit load except that the word SC is replaced by OC.

A.4.3.5. Ports Statement

The word PORT must be given followed by 1, for the input port, or 2, for the output port and the junction to be assigned to that port.

```
e.g. 1) PORT  1  5
     2) PORT  2  7
```

Any junction may be assigned as the input or output port of the circuit provided the circuit topology, section A.4.3.1, will be acceptable later by the analysis part of the program. The characteristic impedances for the input and output ports of the circuit is derived from the load impedances connected to the junctions assigned as the ports of the circuit.

A.4.4. OUTPUT STATEMENT

The choice of output options are given in Tables A.4.2 and A.4.3. A maximum of 7 of these may be selected for printing in the results for the circuit performance in tabular form. To select the required output options the word OPTION must be given followed by a list of not more than 7 of the output option numbers from Table A.4.2 or Table A.4.3. This list must be terminated in a 0.

e.g. OPTION 9 33 52 49 44 41 0

A.4.5. FREQUENCY STATEMENT

The word FREQUENCY must be given followed by the start, step and stop values of the frequency in GHz. During a circuit analysis a table of the circuit performance is produced starting at the start frequency and stepping the step frequency until the stop frequency.

e.g. FREQUENCY 9.8 0.1 10.9

There must not be more than 100 frequencies in this range.

A.4.6. VARIABLE STATEMENT

In some cases it may be necessary to vary one or more parameters in a circuit to determine the effect of these parameters on the circuit performance. This may be done using the VARIABLE statement to vary individual parameters in the circuit. Later in the analysis a number of tables of the circuit response will be produced. In the first table all the variable parameters will be set to their initial values and then for each new table all the

Output option no.	Output option	Form	Units
1	Characteristic impedance port 1	$R_{o1}+jX_{o1}$	ohm, ohm
2	Characteristic impedance port 1	$Z_{o1}\angle\theta_{o1}$	ohm, deg
3	Characteristic impedance port 2	$R_{o2}+jX_{o2}$	ohm, ohm
4	Characteristic impedance port 2	$Z_{o2}\angle\theta_{o2}$	ohm, deg
5	Characteristic admittance port 1	$G_{o1}+jB_{o1}$	mmho, mmho
6	Characteristic admittance port 1	$Y_{o1}\angle\phi_{o1}$	mmho, deg
7	Characteristic admittance port 2	$G_{o2}+jB_{o2}$	mmho, mmho
8	Characteristic admittance port 2	$Y_{o2}\angle\phi_{o2}$	mmho, deg
9	Input impedance port 1	$R_{in1}+jX_{in1}$	ohm, ohm
10	Input impedance port 1	$Z_{in1}\angle\theta_{in1}$	ohm, deg
11	Input impedance port 2	$R_{in2}+jX_{in2}$	ohm, ohm
12	Input impedance port 2	$Z_{in2}\angle\theta_{in2}$	ohm, deg
13	Input admittance port 1	$G_{in1}+jB_{in1}$	mmho, mmho
14	Input admittance port 1	$Y_{in1}\angle\phi_{in1}$	mmho, deg
15	Input admittance port 2	$G_{in2}+jB_{in2}$	mmho, mmho
16	Input admittance port 2	$Y_{in2}\angle\phi_{in2}$	mmho, deg
17	Input impedance port 1	$R_{in1}+jX_{in1}$	p.u. of Z_{o1}
18	Input impedance port 1	$Z_{in1}\angle\theta_{in1}$	p.u. of Z_{o1}, deg
19	Input impedance port 2	$R_{in2}+jX_{in2}$	p.u. of Z_{o2}
20	Input impedance port 2	$Z_{in2}\angle\theta_{in2}$	p.u. of Z_{o2}, deg
21	Input admittance port 1	$G_{in1}+jB_{in1}$	p.u. of Y_{o1}
22	Input admittance port 1	$Y_{in1}\angle\phi_{in1}$	p.u. of Y_{o1}, deg
23	Input admittance port 2	$G_{in2}+jB_{in2}$	p.u. of Y_{o2}
24	Input admittance port 2	$Y_{in2}\angle\phi_{in2}$	p.u. of Y_{o2}, deg

Table A.4.2. - Output Options (Impedances) for the CHAIN1 Program

Output option no.	Output option	Form	Units
25	Input/output voltage ratio, output open circuit	$A_{11r}+jA_{11i}$	p.u., p.u.
26	Input/output voltage ratio, output open circuit	$A_{11}\angle\alpha_{11}$	p.u., deg
27	Input voltage/output current ratio, output short circuit	$A_{12r}+jA_{12i}$	ohm, ohm
28	Input voltage/output current ratio, output short circuit	$A_{12}\angle\alpha_{12}$	ohm, deg
29	Input current/output voltage ratio, output open circuit	$A_{21r}+jA_{21i}$	mmho, mmho
30	Input current/output voltage ratio, output open circuit	$A_{21}\angle\alpha_{21}$	mmho, deg
31	Input/output current ratio, output short circuit	$A_{22r}+jA_{22i}$	p.u., p.u.
32	Input/output current ratio, output short circuit	$A_{22}\angle\alpha_{22}$	p.u., deg
33	Reflection at port 1	$S_{11r}+jS_{11i}$	p.u., p.u.
34	Reflection at port 1	$S_{11}\angle\beta_{11}$	p.u., deg
35	Transmission port 2 to port 1	$S_{12r}+jS_{12i}$	p.u., p.u.
36	Transmission port 2 to port 1	$S_{12}\angle\beta_{12}$	p.u., deg
37	Transmission port 1 to port 2	$S_{21r}+jS_{21i}$	p.u., p.u.
38	Transmission port 1 to port 2	$S_{21}\angle\beta_{21}$	p.u., deg
39	Reflection at port 2	$S_{22r}+jS_{22i}$	p.u., p.u.
40	Reflection at port 2	$S_{22}\angle\beta_{22}$	p.u., deg
41	Reflection at port 1	S_{11}	dB
42	as 41		
43	Transmission port 2 to port 1	S_{12}	dB
44	as 43		
45	Transmission port 1 to port 2	S_{21}	dB
46	as 45		
47	Reflection at port 2	S_{22}	dB
48	as 47		
49	Input VSWR port 1	$VSWR_1$	p.u., $\geqslant 1$
50	Input VSWR port 1	$VSWR_1$	p.u., $\leqslant 1$
51	Input VSWR port 2	$VSWR_2$	p.u., $\geqslant 1$
52	Input VSWR port 2	$VSWR_2$	p.u., $\leqslant 1$

Table A.4.3. - Output Options (Transmission) for the CHAIN1 Program

variable parameters will be stepped by their step values until the first variable parameter exceeds its stop value or when 10 tables have been produced.

Variable Network Parameter - The word VARIABLE must be given followed by the number of the variable, the word NETWORK, the network number and the parameter number to be varied in that network. Finally the start, step and stop values of that parameter must be given.

e.g. VARIABLE 2 NETWORK 12 1 20.0 10.0 50.0

Variable Load Parameter - This is the same as for the variable network parameter except that the word NETWORK is replaced by the word LOAD.

e.g. VARIABLE 4 LOAD 3 2 0.1 0.2 0.9

Reset Variable to Empty - The word VARIABLE must be given followed by the variable number, either the word NETWORK or LOAD and 0.

e.g. VARIABLE 3 NETWORK 0

A.4.7. CIRCUIT STATEMENT

The circuit description, as a path description, is printed out at the start of the next analysis if a circuit change has occurred in the data or if a circuit statement is included in the data. The circuit statement simply consists of the word CIRCUIT.

e.g. CIRCUIT

The circuit description consists firstly of the main link path between the input and output of the circuit including all the networks in that path. If a network number is included negated then the network is connected in the reverse direction in the path. At the end of the main path the loads attached to input and output ports of the circuit are printed out. The type number for each load is as follows :-

Type	Load
1	Parallel RLC
2	Series RLC
3	Short Circuit

N.B. An open circuit load is not printed out.

There may be parallel paths, branch arms and loops in the circuit and these would be printed out in a similar way to the main link with pointers to these paths from their paths of origin and visa-versa also included. In a new path description the first line states the line at which this path description ends and the level of the path. The last line of the path gives the path type and its point of origin. The pointers in the path of origin give the type of the path (see below) followed by the line in the circuit description at which its description starts and the line to which the path returns. In the case of a branch arm the return line is the same as the line of origin. Another type of pointer in the path of origin gives the return of a parallel path and states the line of origin of that parallel path

Path Type	Meaning
Z	Branch arm or loop on a series junction or a parallel path starting and finishing on series junctions.
Y	Branch arm or loop on a parallel junction or a parallel path starting and finishing on parallel junctions.
H	Parallel path starting on a series junction and finishing on a parallel junction.
G	Parallel path starting on a parallel junction and finishing on a series junction.

A.4.8. TITLE STATEMENT

At any point in the data the word TITLE may be given to start a statement and to include a title, i.e. a series of characters terminated in a <;>. This title string is printed out directly in the results. The character string must include capital letters, digits, spaces and line feeds only with up to 240 characters, i.e. this corresponds to 3 cards.

e.g. TITLE CHANGE OF LINE LENGTH ;

A.4.9. ANALYSIS STATEMENT

The calculation sequence is commenced by giving the word ANALYSE in the data but before an analysis is allowed the following must be present :-

1) The input and output ports must both be assigned to junctions.
2) There must be at least one output option selected.
3) A frequency range must have been given.
4) A load must be attached to both the input and output ports.
5) There must be a path between the input and output ports in the circuit.
6) The circuit topology must be acceptable.

A.4.10. TERMINATION STATEMENT

To terminate a run on the program the word END must be given.

A.4.11. EXECUTION OF PROGRAM

The input data for the program must be on punched cards with each new statement on a new card. The first card of this deck must be the &JOB card giving the job number and an optional title for the job. The next card is the &LOAD card and this will load the CHAIN1 program from disc and after this the &RUN card will run the program. This is followed by the data and then further &RUN cards or an &END card to terminate the job.

```
&JOB; <job number> ; <optional title> ;
&LOAD;  CHAIN1;  DC; <disc no.> ; ALGOL;
&RUN;
   ↑
   data for run no. 1
   ↓
&RUN;
   ↑
   data for run no. 2 etc.
   ↓
&END;
```

A.4.12. RUN TIME ERRORS

Errors that are detected in the program are printed out as error messages and, in some cases, the data which produced that error.

e.g. CHAIN1 ERROR 14 NETW 27

All the errors are listed in Table A.4.4 with the possible cause of that error. Once an error has been detected the program will not allow a further analysis statement to be obeyed and the rest of the data will be checked for syntax errors only.

Error no.	Message	Meaning	Statement
1		title too long	any title
2		number of nodes, networks, loads or variables ≤ 0	max. size of circuit
11		outshift character in word	any word
12	<word>	word not acceptable	any word
13	JUNC <no.>	junction no. not acceptable	JUNCTION, LOAD NETWORK or PORT
14	NETW <no.>	network no. not acceptable	NETWORK or VARIABLE
15	NETT <no.>	network type no. not acceptable	NETWORK
16	LOAD <no>	load no. not acceptable	LOAD, VARIABLE
17	PORT <no.>	port no. not acceptable	PORT
18	VARI <no.>	variable no. not acceptable	VARIABLE
19	PARA <no.>	parameter no. not acceptable	VARIABLE
20	OPTI <no.>	option no. not acceptable	OPTION
21		too many output options	OPTION
22		port not assigned to a junction	PORT
23		no output options	OPTION
24		too many frequency steps	FREQUENCY
101		no path between input and output ports	
102		no load on a port	
103	LINE <no.>	interactive branch in circuit from line <no.> in description	
104	LINE <no.>	interactive branch in circuit from line <no.> in description	

Table A.4.4. - Run Time Errors in CHAIN1

A.4.13. EXAMPLE, 3dB BRANCH ARM HYBRID RING

Problem

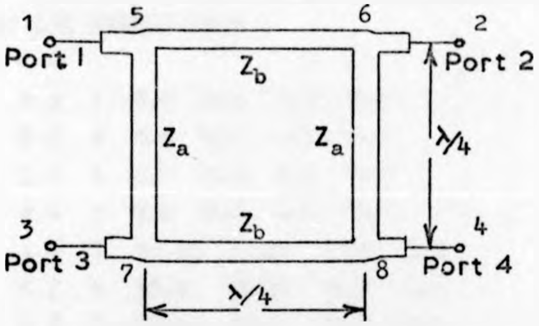


Fig. A.4.12. - 3dB Branch Arm Hybrid Ring

It is required to determine the input impedance and input reflection coefficient in complex form and the transmission, both in complex form and in dB, for the 3dB branch arm hybrid ring in Fig. A.4.12 between ports 1 and 2. The frequency range of interest is 5 to 15 GHz in steps of 1 GHz. The characteristic impedance for the system is 50 ohm (Z_o) and the centre design frequency is 10 GHz . The line impedances are $Z_a = Z_o$ and $Z_b = Z_o/\sqrt{2}$. An analysis is required with no line losses and with line losses of 0.1 dB/wavelength.

Data

In the circuit, as the load (equal to Z_o) can not be attached to a junction with more than one network connected to it, it is necessary to include dummy links in series with ports 3 and 4. In the example this is done also on ports 1 and 2 so that the input and output ports could be moved onto ports 3 or 4. The analysis of the circuit is carried out with port 1 as the input port and port 2 as the output port. The data for this problem is given in Table A.4.5.

```
&JOB;  <job number> ;  HYBRID RING ;
&LOAD; CHAIN1; DC; 10; ALGOL;
&RUN;
3 DB BRANCH ARM HYBRID RING ;
8  8  4  4
NETWORK 1   1 5   1   0.0   0.0   0.0   0.0
NETWORK 2   6 2   1   0.0   0.0   0.0   0.0
NETWORK 3   3 7   1   0.0   0.0   0.0   0.0
NETWORK 4   8 4   1   0.0   0.0   0.0   0.0
NETWORK 5   5 6   9   35.35   0.25   10.0   0.0
NETWORK 6   5 7   9   50.0    0.25   10.0   0.0
NETWORK 7   6 8   9   50.0    0.25   10.0   0.0
NETWORK 8   7 8   9   35.35   0.25   10.0   0.0
LOAD 1   1   SERIES   50.0   0.0   0.0
LOAD 2   2   SERIES   50.0   0.0   0.0
LOAD 3   3   SERIES   50.0   0.0   0.0
LOAD 4   4   SERIES   50.0   0.0   0.0
PORT 1   1
PORT 2   2
OPTION 9   33   38   45   49   0
FREQUENCY   5.0   1.0   15.0
VARIABLE 1   NETWORK 5   4   0.0   0.1   0.1
VARIABLE 2   NETWORK 6   4   0.0   0.1   0.1
VARIABLE 3   NETWORK 7   4   0.0   0.1   0.1
VARIABLE 4   NETWORK 8   4   0.0   0.1   0.1
TITLE INSERTION LOSS PORTS 1 TO 2 ;
ANALYSE
END
&END;
```

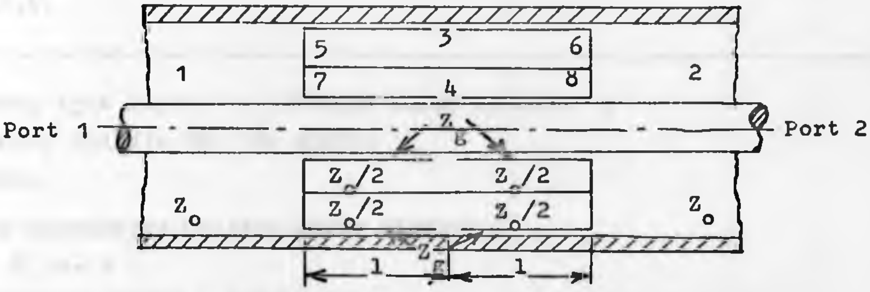
Table A.4.5. - Data for 3dB Branch Arm Hybrid Ring on CHAIN1

Results

```
3DB  BRANCH  ARM  HYBRID  RING;



NUMBER OF NODES      =    8
NUMBER OF NETWORKS   =    8
NUMBER OF PORTS      =    2
NUMBER OF LOADS      =    4
NUMBER OF VARIABLES  =    4

INSERTION  LOSS PORT  1  TO  2;



CIRCUIT DESCRIPTION


MAIN LINK
    0   END AT LINE    6  LEVEL    1
    1  NETWORK    1  NODES    1  TO    5  TYPE    1  PARAMETERS  .000000  .000000  .000000  .000000
    2    START OF  PARALLEL PATH TYPE Y  AT LINE    8  RETURN AT LINE    4
    3  NETWORK    5  NODES    5  TO    6  TYPE    9  PARAMETERS  35.3500  .250000  10.0000  .000000
    4    RETURN FROM LINE    2
    5  NETWORK    2  NODES    6  TO    2  TYPE    1  PARAMETERS  .000000  .000000  .000000  .000000
    6    END OF  MAIN LINK
PORT 1  LOAD    1  TYPE    2  PARAMETERS  50.0000  .000000  .000000
PORT 2  LOAD    2  TYPE    2  PARAMETERS  50.0000  .000000  .000000


PARALLEL PATH TYPE Y
    8   END AT LINE   14  LEVEL    2
    9  NETWORK    6  NODES    5  TO    7  TYPE    9  PARAMETERS  50.0000  .250000  10.0000  .000000
   10    START OF  BRANCH ARM TYPE Y  AT LINE   16  RETURN AT LINE   10
   11  NETWORK     6  NODES     7  TO     8  TYPE     9  PARAMETERS  35.3500  .250000  10.0000  .000000
   12     START OF  BRANCH ARM TYPE Y  AT LINE   20  RETURN AT LINE   12
   13  NETWORK    -7  NODES     8  TO     6  TYPE     9  PARAMETERS  50.0000  .250000  10.0000  .000000
   14     END OF  PARALLEL PATH TYPE Y  ORIGIN AT LINE    2

BRANCH ARM TYPE Y
   16    END AT LINE    18  LEVEL     3
   17  NETWORK    -3  NODES     7  TO     3  TYPE     1  PARAMETERS  .000000  .000000  .000000  .000000
   18     END OF  BRANCH ARM TYPE Y  ORIGIN AT LINE   10
LOAD     3  TYPE     2  PARAMETERS  50.0000  .000000  .000000


BRANCH ARM TYPE Y
   20    END AT LINE    22  LEVEL     3
   21  NETWORK     4  NODES     8  TO     4  TYPE     1  PARAMETERS  .000000  .000000  .000000  .000000
   22     END OF  BRANCH ARM TYPE Y  ORIGIN AT LINE   12
LOAD     4  TYPE     2  PARAMETERS  50.0000  .000000  .000000




VARIABLE     1  NETWORK     5  PARAMETER     4 =    .000000
VARIABLE     2  NETWORK     6  PARAMETER     4 =    .000000
VARIABLE     3  NETWORK     7  PARAMETER     4 =    .000000
VARIABLE     4  NETWORK     8  PARAMETER     4 =    .000000

   FREQ     INPUT IMPED  1      SPAR 1 1          SPAR 2 1          SPAR 2 1          VSWR 1
   (GHZ)        (OHM)             (PU)          (PU)    (DEG)         (DB)              (PU)
   5.0000   12.71 J 7.739  -.5706 J .1938    .4061   -42.20       -7.8272           4.0330
   6.0000   14.25 J 12.21  -.5016 J .2852    .4094   -44.34       -7.7567           3.7283
   7.0000   18.05 J 17.42  -.3685 J .3474    .4538   -46.42       -6.6631           3.0523
```

```
 8.0000    28.30 J 21.30   -.1892 J .3234    .5498  -53.63     -5,1953            2.1985
 9.0000    42.69 J 16.52   -.0456 J .1864    .6593  -69.15     -3,6188            1,4748
 10.000    49.98 J .0000   -.0002 J .0000    .7072  -90.00     -3,0090            1.0003
 11.000    42.69 J-16.52   -.0456 J-.1864    .6593 -110.85     -3,6188            1.4748
 12.000    28.30 J-21.30   -.1892 J-.3234    .5498 -126.37     -5,1953            2.1985
 13.000    18.65 J-17.42   -.3685 J-.3474    .4538 -133.58     -6,8631            3,0520
 14.000    14.28 J-12.21   -.5016 J-.2852    .4094 -135.66     -7,7567            3.7283
 15.000    12.71 J-7.739   -.5706 J-.1938    .4061 -137.80     -7,8272            4,0330

VARIABLE     1  NETWORK    5  PARAMETER     4 =    .100000
VARIABLE     2  NETWORK    6  PARAMETER     4 =    .100000
VARIABLE     3  NETWORK    7  PARAMETER     4 =    .100000
VARIABLE     4  NETWORK    8  PARAMETER     4 =    .100000

 FREQ     INPUT IMPED  1      SPAR 1 1        SPAR 2 1           SPAR 2 1           VSWR 1
 (GHZ)         (OHM)            (PU)        (PU)     (DEG)         (DB)              (PU)
 5.0000    12.75 J 7.732   -.5698 J .1934    .4059  -42.19     -7,8317            4,0214
 6.0000    14.33 J 12.18   -.5007 J .2841    .4093  -44.37     -7,7591            3,7133
 7.0000    18.72 J 17.34   -.3681 J .3453    .4534  -46.53     -6,8703            3,0380
 8.0000    28.32 J 21.11   -.1904 J .3208    .5482  -53.81     -5,2206            2,1902
 9.0000    42.50 J 16.30   -.0485 J .1848    .6557  -69.28     -3,6663            1.4723
 10.000    49.64 J .0000   -.0036 J .0000    .7023  -90.00     -3,0691            1,0072
 11.000    42.46 J-16.25   -.0491 J-.1844    .6549 -110.69     -3,6768            1.4718
 12.000    28.32 J-21.01   -.1910 J-.3195    .5474 -126.11     -5,2333            2,1861
 13.000    18.77 J-17.27   -.3678 J-.3435    .4531 -133.37     -6,8766            3.0262
 14.000    14.40 J-12.14   -.4995 J-.2826    .4091 -135.58     -7,7625            3.6936
 15.000    12.32 J-7.720   -.5681 J-.1927    .4055 -137.83     -7,8407            3,9983


STCRE LEFT 39971  USED 13784
TIME =  0000  20
  A
```

A.4.14. EXAMPLE, COAXIAL SHORT CIRCUIT



n.b. 1) The cross section of this short circuit across A-A is radial about its centre line.

2) All solid lines represent conducting boundaries.

Fig. A.4.13. - Coaxial Short Circuit

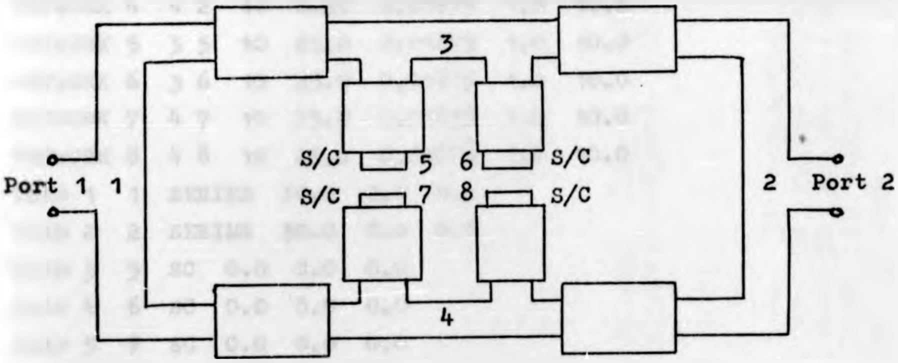


Fig. A.4.14. - Equivalent Circuit of Coaxial Short Circuit

It is required to determine the input impedance as p.u. of Z_o and the input reflection coefficient (in modulus and argument form) the transmission (both in complex form and in dB) and the input V.S.W.R. ($\geqslant 1$) for the above coaxial short circuit. The frequency range of interest is 0.5 to 10 GHz in steps of 0.5 GHz with a centre design frequency of 4.0 GHz. The characteristic impedance for the system is 50 ohm (Z_o) . The gap impedance Z_g is 6.25 ohm and the length l is 1.875 cm. The lines are all air spaced with losses of 10 dB/metre.

Data

The data for the coaxial short circuit is shown in Table A.4.6.

```
&JOB; <job number> ;  COAXIAL SHORT CIRCUIT  ;
&LOAD; CHAIN1; DC; 10; ALGOL;
&RUN;

NON CONTACTING COAXIAL SHORT CIRCUIT ;
8  8  6  1
JUNCTION SERIES 1 2 3 4 0
NETWORK 1  1 3  10  6.25  0.01875  1.0  10.0
NETWORK 2  3 2  10  6.25  0.01875  1.0  10.0
NETWORK 3  1 4  10  6.25  0.01875  1.0  10.0
NETWORK 4  4 2  10  6.25  0.01875  1.0  10.0
NETWORK 5  3 5  10  25.0  0.01875  1.0  10.0
NETWORK 6  3 6  10  25.0  0.01875  1.0  10.0
NETWORK 7  4 7  10  25.0  0.01875  1.0  10.0
NETWORK 8  4 8  10  25.0  0.01875  1.0  10.0
LOAD 1  1  SERIES  50.0  0.0  0.0
LOAD 2  2  SERIES  50.0  0.0  0.0
LOAD 3  5  SC  0.0  0.0  0.0
LOAD 4  6  SC  0.0  0.0  0.0
LOAD 5  7  SC  0.0  0.0  0.0
LOAD 6  8  SC  0.0  0.0  0.0
PORT 1  1
PORT 2  2
OPTION  18  34  37  41  49  0
FREQUENCY  0.5  0.5  10.0
ANALYSE
END

&END;
```

Table A.4.6. - Data for Coaxial Short Circuit for CHAIN1

Results

```
NCN CONTACTING  COAXIAL  SHORT  CIRCUIT :

NUMBER OF NODES      =    8
NUMBER OF NETWORKS   =    8
NUMBER OF PORTS      =    2
NUMBER OF LOADS      =    6
NUMBER OF VARIABLES  =    1

CIRCUIT DESCRIPTION


MAIN LINK
   0  END AT LINE    7  LEVEL    1
   1    START OF  PARALLEL PATH TYPE Z  AT LINE    9  RETURN AT LINE    6
   2  NETWORK    1  NODES    1  TO    3  TYPE   10  PARAMETERS  6.25000  .018750  1.00000  10.0000
   3    START OF  BRANCH ARM TYPE Z  AT LINE   16  RETURN AT LINE    3
   4    START OF  BRANCH ARM TYPE Z  AT LINE   20  RETURN AT LINE    4
   5  NETWORK    2  NODES    3  TO    2  TYPE   10  PARAMETERS  6.25000  .018750  1.00000  10.0000
   6    RETURN FROM LINE    1
   7    END OF MAIN LINK
PORT 1  LOAD    1  TYPE    2  PARAMETERS  50.0000  .000000  .000000
PORT 2  LOAD    2  TYPE    2  PARAMETERS  50.0000  .000000  .000000


PARALLEL PATH TYPE Z
   9  END AT LINE   14  LEVEL    2
  10  NETWORK    3  NODES    1  TO    4  TYPE   10  PARAMETERS  6.25000  .018750  1.00000  10.0000
  11    START OF  BRANCH ARM TYPE Z  AT LINE   24  RETURN AT LINE   11
  12    START OF  BRANCH ARM TYPE Z  AT LINE   28  RETURN AT LINE   12
  13  NETWORK    4  NODES    4  TO    2  TYPE   10  PARAMETERS  6.25000  .018750  1.00000  10.0000
  14    END OF  PARALLEL PATH TYPE Z  ORIGIN AT LINE    1
```

```
BRANCH ARM TYPE Z
   16   END AT LINE    18  LEVEL    4
   17  NETWORK     6  NODES     3  TO     6  TYPE   10  PARAMETERS  25.0000  .018750  1.00000  10.0000
   18     END OF  BRANCH ARM TYPE Z  ORIGIN AT LINE      3
LCAD     4  TYPE     3  PARAMETERS  .000000  .000000  .000000

BRANCH ARM TYPE Z
   20   END AT LINE    22  LEVEL    4
   21  NETWORK     5  NODES     3  TO     5  TYPE   10  PARAMETERS  25.0000  .018750  1.00000  10.0000
   22     END OF  BRANCH ARM TYPE Z  ORIGIN AT LINE      4
LCAD     3  TYPE     3  PARAMETERS  .000000  .000000  .000000

BRANCH ARM TYPE Z
   24   END AT LINE    26  LEVEL    3
   25  NETWORK     8  NODES     4  TO     8  TYPE   10  PARAMETERS  25.0000  .018750  1.00000  10.0000
   26     END OF  BRANCH ARM TYPE Z  ORIGIN AT LINE     11
LCAD     6  TYPE     3  PARAMETERS  .000000  .000000  .000000

BRANCH ARM TYPE Z
   28   END AT LINE    30  LEVEL    3
   29  NETWORK     7  NODES     4  TO     7  TYPE   10  PARAMETERS  25.0000  .018750  1.00000  10.0000


   30     END OF  BRANCH ARM TYPE Z  ORIGIN AT LINE     12
LCAD     5  TYPE     3  PARAMETERS  .000000  .000000  .000000
```

```
   FREQ      INPUT IMPED  1       SPAR 1 1          SPAR 2 1          SPAR 1 1       VSWR 1
   (GHZ)     (PU)     (DEG)    (PU)    (DEG)         (PU)               (DB)          (PU)
   .50000    .5656   -23.19   .3445 -146.77      .4857 J-.6851       -9.2558        2.0512
   1.0000    .8971    -9.70   .1007 -122.85     -.3885 J-.7529       -19.942        1.2239
   1.5000    .6435   -77.45   .8183 -115.00     -.3863 J .1057       -1.7413        10.009
   2.0000    .3300   -85.51   .9544 -143.56     -.0970 J .1062       -.40495        42.906
   2.5000    .2007   -86.63   .9775 -157.34     -.0295 J .0567       -.19722        88.087
   3.0000    .1195   -86.21   .9846 -166.40     -.0097 J .0292       -.13522        128.47
   3.5000    .0566   -83.45   .9872 -173.57     -.0030 J .0126       -.11181        155.38
   4.0000    .0061     0.00   .9879    0.00     -.0013 J .0000       -.10544        164.76
   4.5000    .0566    83.45   .9872  173.57     -.0030 J-.0126       -.11181        155.38
   5.0000    .1195    86.21   .9846  166.40     -.0097 J-.0292       -.13522        128.47
   5.5000    .2007    86.63   .9775  157.34     -.0295 J-.0567       -.19722        88.087
   6.0000    .3300    85.51   .9544  143.56     -.0970 J-.1062       -.40495        42.906
   6.5000    .6435    77.45   .8183  115.00     -.3863 J-.1057       -1.7413        10.009
   7.0000    .8971     9.70   .1007  122.85     -.3885 J .7529       -19.942        1.2239
   7.5000    .5656    23.19   .3445  146.77      .4857 J .6851       -9.2558        2.0512
   8.0000    .8969     0.00   .0532  180.00      .8943 J .0000       -25.474        1.1125
   8.5000    .5656   -23.19   .3445 -146.77      .4857 J-.6851       -9.2558        2.0512
   9.0000    .8971    -9.70   .1007 -122.85     -.3885 J-.7529       -19.942        1.2239
   9.5000    .6435   -77.45   .8183 -115.00     -.3863 J .1057       -1.7413        10.009
   10.000    .3300   -85.51   .9544 -143.56     -.0970 J .1062       -.40495        42.906


STORE LEFT 39968  USED 13787


 &END;                                                                                 TIME =   0000  20
TIME =  0000  20
   A
```

Appendix A.5

User Manual for Mixed Matrix Analysis MICRO3 Program

A.5.1. INTRODUCTION

The MICRO3 program will calculate the steady state sinusoidal response of a circuit made up of an assembly of n-port networks. For the analysis a number of these networks are defined as being connected to the external ports of the circuit.

In the program a network is defined as a circuit complete within itself, except for its ports which are used to connect it with other networks in the circuit. The network types available in the present program are given in Tables A.5.1,A.5.2 and A.5.3. In the equivalent circuit of a network each port consists of a terminal pair. In the circuit the current entering one terminal of such a pair must equal the current leaving the other terminal of the same pair. Also in the analysis the program is only interested in the voltage difference between the two terminals on a terminal pair defining a port. To realise this in the equivalent circuit correctly it may be necessary to include an ideal transformer with unity turns ratio in cascade with the ports on some if not all the networks in the circuit.

To make a circuit suitable for analysis using the MICRO3 program the circuit must first be broken down into an assembly of n-port networks as shown in the examples in sections A.5.13 and A.4.14. Once the circuit has been broken down into n-port networks it will be observed that the networks will have ports connected together at junctions in either a series or parallel connection as shown in Fig. A.5.1 and A.5.2. If a junction is a mixture of these two types then dummy networks, i.e. decoupling transformers with unity turns ratio, must be included to form two or more junctions of an acceptable type. A number of junctions in the

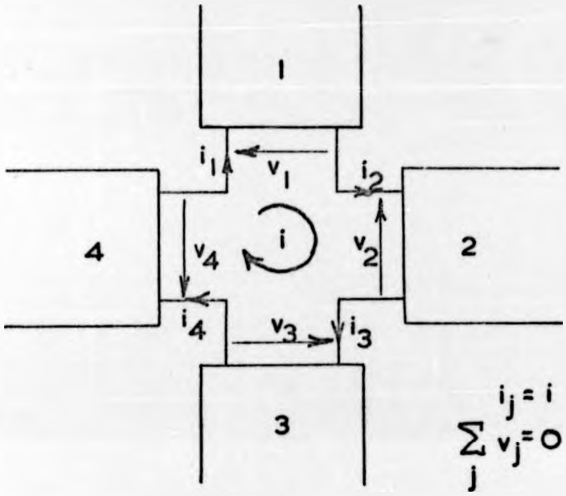


Fig.A.5.1.- Series Junction

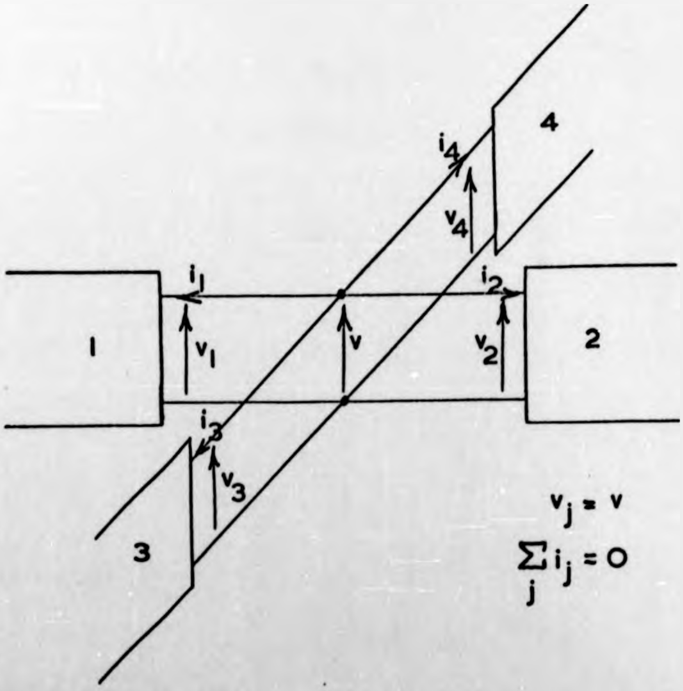


Fig.A.5.2.- Parallel Junction

circuit can then be assigned as external ports for the circuit. In the analysis there must be a characteristic impedance associated with each external port in the circuit. This characteristic impedance is required to derive the scattering matrix for the circuit.

The data for the program consists of a number of statements each one of which is terminated in a <;> . Some of these statements could be further divided into sub-statements or lists of simpler data. Full syntax and diagnostic checking of the data is included in the program. If a syntax error is detected in the data then the program will search for the next <;> in the data to terminate the current statement. After this the program will check the rest of the data for syntax errors only.

During the running of the program there are three character streams used by the program :-

data channel
message channel
result channel

The result channel, normally the line printer, consists of a copy of the input data , including syntax errors as they are detected, and the results of any circuit analysis or optimisation. The message channel is used to print out messages for the users information during the running of the program. This is normally assigned to the line printer but could be assigned to say a remote teletype for interactive use of the program. In general all the channels mentioned above can be assigned to any suitable device or channel including files. The MICRO3 program is suitable for use in an interactive mode as well as on batch in which case it is possible to correct errors detected in the data online.

Facilities for optimisation, but not the optimisation packages, were included as a later addition to the program. Also the MIC3D version of the program enabled the results of an analysis to be displayed on the graphical display on the Elliott 4130 computer. The use of the optimisation part of the program is described in chapter 10 and the graphical display version of the program is described in section 8.3.

A.5.2. CIRCUIT TOPOLOGY STATEMENTS

A.5.2.1. Circuits

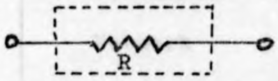
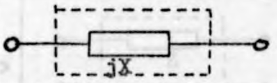
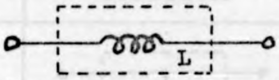
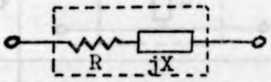
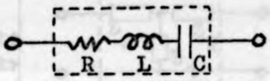
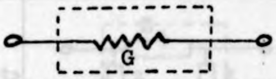
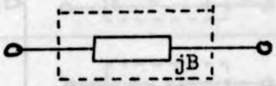
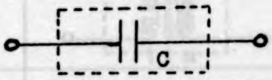
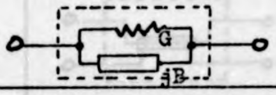
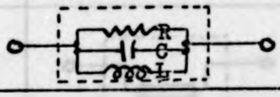
If it is required to store more than one circuit topology in the computer then at the start of the description of each circuit the circuit number must be given. This is given as a circuit statement starting with the word CIRCUIT followed by the circuit number. Then all the topology statements in the data will refer to this circuit until another circuit statement is given. Any number of circuit topologies can be stored , analysed and modified as desired in any order.

e.g. CIRCUIT 15 ;

There are two places in the data when the computer does not know which circuit is being referred to. This is at the start of the data and following the deletion of a circuit. At these two points circuit number 0 is assumed until another circuit statement is found. Networks, junctions and ports are associated with one circuit only and thus the same network, junction or port numbers could be used in several different circuits.

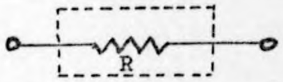
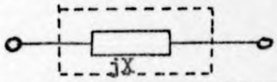
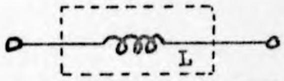
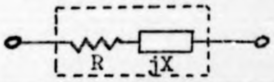
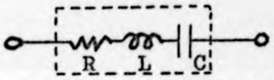
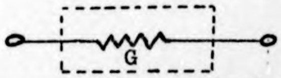
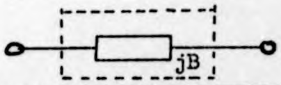
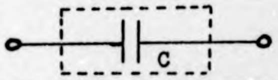
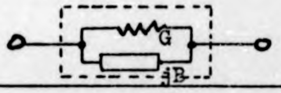
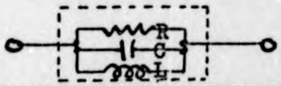
A.5.2.2. Networks

n.b. Tables A.5.1, A.5.2 and A.5.3 gives all the network types so far included in the MICRO3 program.

Network type	Network	Parameter	
		No. of	Values
R		1	R (ohm)
X		1	X (ohm)
L		1	L (H)
SRX		2	R (ohm) X (ohm)
SRLC		3	R (ohm) L (H) ** C (F)
G		1	G (mho)
B		1	B (mho)
C		1	C (F)
PGB		2	G (mho) B (mho)
PGCL		3	G (mho) C (F) ** L (H)

n.b. 1) ** if the value is set as zero then an infinite value is assumed.

Table A.5.1. - 1-port Networks for the Networks and Characteristic Impedances in MICRO3 Program

Network type	Network	Parameter	
		No. of	Values
R		1	R (ohm)
X		1	X (ohm)
L		1	L (H)
SRX		2	R (ohm) X (ohm)
SRLC		3	R (ohm) L (H) ** C (F)
G		1	G (mho)
B		1	B (mho)
C		1	C (F)
PGB		2	G (mho) B (mho)
PGCL		3	G (mho) C (F) ** L (H)

n.b. 1) ** if the value is set as zero then an infinite value is assumed.

Table A.5.1. - 1-port Networks for the Networks and Characteristic Impedances in MICRO3 Program

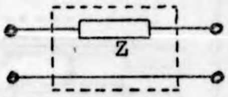
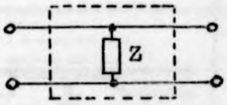
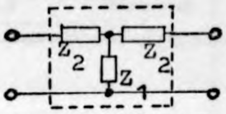
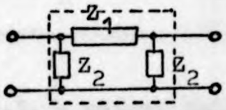
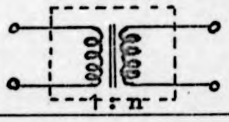
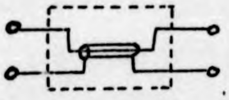
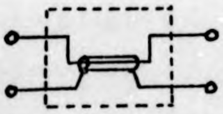
Network type	Network	Parameters		Comment
		No. of	Values	
SERI <Z format>		As Z in Table A.5.1.		not series to series junctions
SHNT <Z format>		As Z in Table A.5.1.		not parallel to parallel junctions
TEE <Z_1 format> <Z_2 format>		As Z_1 and Z_2 in Table A.5.1.		
PHI <Z_1 format> <Z_2 format>		As Z_1 and Z_2 in Table A.5.1.		
TRANSF		1	n (p.u.)	not series to series or parallel to parallel junctions
LINE LG		4	Z_o (ohm) l^o(m) v.r. (p.u.) α (dB/m)	
LINE WL		4	Z_o (ohm) λ^o (wavelength at f_o) f_o (Hz) α(dB/λ at f_o)	α assumed proportional to $\sqrt{f/f_o}$

Table A.5.2. - 2-port Networks for MICRO3 Program

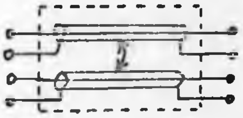
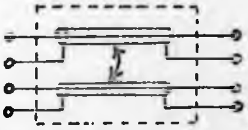
Network type	Network	Parameters		Comments
		No. of	Values	
CPLINE LG		8	Even mode as LINE LG plus Odd mode as LINE LG	4-port
CPLINE WL		8	Even mode as LINE WL plus Odd mode as LINE WL	4-port
CIRCUIT		2	circuit no. no. of ports on circuit	n-port sub-circuit defined as for main circuit

Table A.5.3. - n-port Networks for the MICRO3 Program

A network statement is used to connect an n-port network of a specified type with the parameters for that network type between junctions in the circuit. The network statement must start with the word NET followed by the network number.

e.g. NET 27 LINE WL 50.0 0.25 1E+9 0.1 ;

The rest of the network statement consists of a list of the network sub-statements given below and these may be included in any order provided a network type sub-statement has been included for the network, possibly in a previous statement, before any other type of network sub-statement is given. If a new network type sub-statement is given then all the infor tion associated with the network is erased before the new network type is set up.

```
e.g. NET 52  LINE WL 50.0 0.25 1E+9 0.1 JUNCT 25 41 ;
     NET 52  PAR 1 75.0  PAR 3 1.5E9  CONN 2 7 ;
```

network type sub-statement

The network type sub-statement starts with the word(s) giving the network type followed by a list of the parameters for that network type as given in Table A.5.1, A.5.2 or A.5.3.

e.g. LINE LG 75.0 0.02 0.15 0.21

N.B. A network could refer to a sub-circuit defined in the same way as for the main circuit. For this the word CIRCUIT must be given followed by the circuit number and the number of ports on that circuit.

e.g. CIRCUIT 5 2

junction sub-statement

The junction sub-statement starts with the word JUNCT followed by a list of the junction numbers to which the network is to be connected. The number of junction connections is defined by the last network type sub-statement, Table A.5.1, A.5.2 or A.5.3.

e.g. JUNCT 11 13

parameter sub-statement

The parameter sub-statement may be used to alter the value of one of the network parameters. Its form is the word PAR followed the parameter number in the network parameter list, Table A.5.1, A.5.2 or A.5.3, and the new value for that parameter.

e.g. PAR 3 0.65

connection sub-statement

The connection sub-statement may be used to alter the junction connection on one junction connection of the network. Its form is the word CONN followed by the connection number in the list of junction connections for the network, Table A.5.1,A.5.2 or A.5.3, and the junction number to be assigned for that junction connection.

e.g. CONN 2 15

A.5.2.3. Junctions

In the circuit data structure junctions are created when a network or port is connected to the junction in question. Thus in the data it is only necessary to set the junction types to series or parallel but this can not be done until the junctions have been created in the data structure. The statement to do this is the junction statement which starts with the word JUNCT and then the junction type word, i.e. SERIES or PARALLEL,followed by a list of the junctions to be set to that type.

```
e.g. JUNCT SERIES   7 21 2 5 ;
     JUNCT PARALLEL 8 17 1 6 ;
```

A.5.2.4. Ports

The ports statement will assign a list of junctions as external ports for a circuit. Its starts with the word PORTS followed by a list of junctions to be assigned for the external ports. In the program data structure the external ports as refered by number. The junction connection for a given port, say port i , is obtained from the i th junction entry in the ports statement.

e.g. PORTS 5 6 2 ;

The program can handle at present up to an 8-port network or circuit.

A.5.2.5. Characteristic Impedances

The characteristic impedance statement is used to define a characteristic impedance for one of the ports of a circuit to enable the scattering matrix and other output options to be calculated after a circuit analysis. There is only one list of characteristic impedances stored and this list may be used for the analysis of any circuit stored in the data structure. The characteristic impedance statement starts with the word ZO followed by the characteristic impedance number, the impedance type and the parameters associated with that type. The impedance type and parameters are the same as for a 1-port network in Table A.5.1.

e.g. ZO 2 RJX 50 10 ;

In the circuit analysis the characteristic impedance for port i is taken as the characteristic impedance no. i. If there are .less than i characteristic impedances then the characteristic impedance with the largest number is used but there must always be at least one.

A.5.2.6. Deletions

Networks, junctions, characteristic impedances and circuits may be deleted from the data structure using the delete statement. This statement starts with the word DELETE followed by the list of elements to be deleted from the data structure.

e.g. DELETE NET 5 ZO 2 JUNCT 7 CIRCUIT 8 ;

Any number of elements may be deleted in a single delete statement but all the elements must refer to the current circuit. An exception to this is that any circuit may be included in the list of elements to be deleted but this must be the last element in the list and a new circuit statement must be included next in the data otherwise circuit no. 0 will be assumed for the following data.

In the case of the deletion of a junction the junction is not deleted directly but it is deleted by deleting, in turn, all the networks and ports connected to that junction.

A.5.3. FREQUENCY STATEMENT

The frequency statement defines the frequencies to be included in the table of results of the circuit analysis. It starts with the word FREQ and then a list of the frequencies, in Hz, required in the table of results. These frequencies may be single frequencies or blocks of frequencies as defined below.

```
e.g. FREQ  5E9 2.5E6 27E6 ;
or   FREQ  5E9  STEPLIN 1E9 20E9 19
           7E9  STEPLOG 1E5 1E9 4   20E9 ;
```

Only one frequency list is stored and a new frequency list will replace the last one.

single frequencies

Single frequencies, in Hz, may be included directly in the frequency statement.

```
e.g. FREQ  5E9  2.5E6  27E6 ;
```

incremental frequencies

A large number of frequencies may be included using the step facility. In this case a single frequency in the frequency list is replaced by the word STEPLIN, for linear increments, or the word STEPLOG, for logarithmic increments, followed by the lower and upper limits of the frequency range and the number of steps required.

```
e.g. FREQ  STEPLIN 1E9 20E9 19 ;
     FREQ  STEPLOG 1E5 1E9 4 ;
```

A.5.4. OUTPUT OPTIONS STATEMENT

The results to be included in the table of results from a circuit analysis are defined in the output options statement. This statement starts with the word OUTPUT followed by a list of the output options required as defined in Table A.5.4. The full format for each output option is the option word, port i, port j and the format required but if any of these are not applicable to a given output option then they must not be included.

```
e.g. OUTPUT  ZO 1 CMPX  ZIN 2 MODARG  SPAR 2 1 DBARG  VSWR 3 ;
```

The width of the line printer will only allow 7 output options to be printed on one line and if more are included then the results will run onto several lines. If any matrices are entered in the output options list then these are output after each line in the table of results but the format of the results, if both normal

Output option word	Output option	Acceptable formats	
		for OUTPUT statement	for SPEC statement
ZY	Element ZY(i,j) in mixed matrix	CMPX, MODARG	REAL, IMAG, MOD, ARG
SPAR	Element S(i,j) in scattering matrix	CMPX, MODARG, DBARG	REAL, IMAG, MOD, ARG, DB
ZO	Characteristic impedance for port i	CMPX, MODARG	REAL, IMAG, MOD, ARG
ZIN	Input impedance port i	CMPX, MODARG	REAL, IMAG, MOD, ARG
YO	Characteristic admittance port i	CMPX, MODARG	REAL, IMAG, MOD, ARG
YIN	Input admittance port i	CMPX, MODARG	REAL, IMAG, MOD, ARG
VSWR	Voltage Standing Wave Ratio port i	none	none
ZYMX	Complete mixed matrix	CMPX, MODARG	not acceptable
SPMX	Complete scattering matrix for circuit	CMPX, MODARG, DBARG	not acceptable

N.B. The output option format words are as follows :-

Output option format	format
CMPX	value output in complex form
MODARG	value output in modulus and argument form
DBARG	value output in modulus dB and argument form
REAL	real part of value
IMAG	imaginary part of value
MOD	modulus of value
ARG	argument of value
DB	modulus of value in dB

Table A.5.4. - Output Options for MICRO3 Program

and matrix options are included, is not very easy to decipher. Only one output option list is stored and a new output option list will replace an old one.

A.5.5. ANALYSIS

To start an analysis an analyse statement must be given in the data. This statement consists of the word ANALYSE.

e.g. ANALYSE ;

If the fault indicator is set then the analyse statement is ignored. Otherwise first a check is made to see if the circuit and data is sufficiently defined for an analysis. Any errors found are printed out . If no errors are found then an analysis of the circuit performance is carried out for all the frequencies in the frequency list and the required results, given in the output options list, are printed out in a table of results.

A.5.6. OTHER STATEMENTS

A.5.6.1. End Statement

The end statement consists of the word END and it will terminate the current run of the MICRO3 program.

e.g. END ;

A.5.6.2. Newrun Statement

The newrun statement consists of the word NEWRUN and it will cause the current data structure to be cleared so that a completely fresh start may be made in the data. The channels assigned for input and output for the program are left unchanged.

e.g. NEWRUN ;

A.5.6.3. Label Statement

The label statement consists of the word LABEL followed by a string of characters defining a label string and terminated by a <;> which also terminates the statement. This label string is stored in the data structure and it is printed out in the results before the results obtained from an analyse statement. Only one label string is stored and a new one will replace the last one stored.

e.g. LABEL DATA FOR CIRCUIT NUMBER 2 ;

A.5.6.4. Nofault Statement

After an error is detected in the data a fault indicator is set and further analyse statements are ignored. In an interactive use of the program it is possible to correct data errors. In this case the nofault statement can be used to clear the fault indicator. This statement consists of the word NOFAULT.

e.g. NOFAULT ;

A.5.6.5. Structure Statement

The structure statement will cause the entire data structure formed so far by the data read in to be printed out. This statement consists of the word STRUCTURE.

e.g. STRUCTURE ;

This statement is intended for the programmer interested in locating possible errors in the data structure during program development.

A.5.7. CHANNEL ASSIGNMENT

Initially the input and output channels for the program are set as follows :-

A.5.6.3. Label Statement

The label statement consists of the word LABEL followed by a string of characters defining a label string and terminated by a <;> which also terminates the statement. This label string is stored in the data structure and it is printed out in the results before the results obtained from an analyse statement. Only one label string is stored and a new one will replace the last one stored.

e.g. LABEL DATA FOR CIRCUIT NUMBER 2 ;

A.5.6.4. Nofault Statement

After an error is detected in the data a fault indicator is set and further analyse statements are ignored. In an interactive use of the program it is possible to correct data errors. In this case the nofault statement can be used to clear the fault indicator. This statement consists of the word NOFAULT.

e.g. NOFAULT ;

A.5.6.5. Structure Statement

The structure statement will cause the entire data structure formed so far by the data read in to be printed out. This statement consists of the word STRUCTURE.

e.g. STRUCTURE ;

This statement is intended for the programmer interested in locating possible errors in the data structure during program development.

A.5.7. CHANNEL ASSIGNMENT

Initially the input and output channels for the program are set as follows :-

data channel - assigned to channel 50
message channel - punch 4, line printer
result channel - punch 4, line printer

During the running of the program these character streams may be reassigned to any suitable input or output channel. The data channel is initially assigned to channel 50 and in this way channel 50 can be assigned before the program run to pick up the required channel for the data, e.g. card reader, paper tape reader, remote teletype etc..

The character streams may be reassigned during the running of the program using the following statements :-

DATA <channel no.> - reassign data channel
ERROR <channel no.> - reassign message channel
RESULT <channel no.> - reassign result channel

In each statement <channel no.> refers to the Algol device number, in Elliott 4100 Algol, for the required peripheral device or the channel no. previously assigned to the required device or file. A certain amount of care should be used in the use of the statements in case a channel, such as the data channel, is assigned to an unsuitable channel no. in which case control of the program could be lost.

A.5.8. JOB PREPARATION

A.5.8.1. Data

The data for the MICRO3 program consists of a set of statements as defined in sections A.5.2 to A.5.7 with each statement terminated in a <;> . These statements consist of an assembly in a defined order of the following items :-

words - a word is a group of upper case letters only but in addition an identifier is also accepted as a word, i.e.a group of letters and/or digits which

must start with a letter.
e.g. CIRCUIT NET WG16 PORTS

<u>integers</u> - an integer is a number which does not contain a decimal point or exponent. The number must not have more than 6 digits in it.
e.g. 276 543 -10 0 +1256

<u>reals</u> - a real number is a number with a decimal point and/or an exponent in it but in addition in the program an integer is also acceptable as a real number. The letter E must be used to represent the exponent to the base 10 of the number but the number must not start with an E.
e.g. 7.6 .62 76. 1.2E+9 7.5E-12
1E9 1E-7

<u>semi-colon</u> - a semi-colon is used as a statement terminator.

<u>erase</u> - A % or %% in the data is used as an erasing symbol as described in section A.5.9.

All other characters in the data are ignored and may be considered equivalent to spaces.

A.5.8.2. <u>Storage Requirements</u>

The storage required by the segmented version of the MICRO3 program is 19000 words. Disc systems DES1 or DES2 may be used but for interactive work the multi-programming DES2 system must be used on the Elliott 4130 computer.

A.5.8.3. <u>Batch Processing</u>

The job stream for batch processing using the MICRO3 program is as follows for input data on cards :-

```
&JOB; <job number> ;
&ASSIGN; 50; CR;
&LOAD; MICRO3; DC; <disc no.> ; ALGOL;
&RUN;
     ↑
     data for MICRO3 program
     ↓
&END;
```

A.5.8.4. Interactive use on Remote Teletype

The character stream which must be typed in on the remote teletype to start the program is as follows (n.b. all the characters not underlined are printed out by the computer) :-

```
**MES: DIN,ALLOC,10.
ALLOC
END
**MES: ALLOC,CL.
 &A;50;CTP;
 &A; < assign channel no.s to files as necessary >
 &A; !
END
**MES: DIN,MIC3D,10.
SEGMENTS
    4086
    3742
    2354
MIC3D
END
**MES: MIC3D.
DRO
DR10
DR50
ACIO
CMPLEX
SPR
DISMAN
ERROR 50;
** ASSIGNED
↑     ↑
↑     
    type in data for program MIC3D
↑     
↑     ↓
↑ END;
** RUNEND
END
**MES:
```

Initially a <!> must be typed in to obtain the message <**MES:> printed out on the teletype. If this can not be obtained then the teletype is not connected to the computer correctly. The first line typed in after the <**MES:> will load the ALLOC program and the next line typed in after a <**MES:> will run the ALLOC program. This program is run with the parameter <CL> to clear the current assignment table after which each channel number as required can be assigned after each <&A:> is typed out. It is essential to assign channel 50 to the CTP as the data stream for the program is on channel 50. The ALLOC program is terminated with a <!>and a new message <**MES:> will be printed out. Then the MIC3D program can be loaded from disc 10 and then run by typing in its name after another <**MES:>.

The first line of data for the program must be the statement <ERROR 50 ;> and this will assign the message channel for the program to the teletype. After this the message <** ASSIGNED>will be printed out. If this is not printed out then the program is not working correctly. The main data for the program can now be typed in on the teletype. If an error is detected in the data then the program will print out a suitable error message giving the nature of the error. After each line of data has been read in a ↑ will be printed on a new line to indicate that the program is ready to read in the next line of data. To terminate the program run the statement <END;> must be typed in. Alternatively the program can be abandoned at any time by typing in a <!>.

It is a very slow process to type in data on the teletype and it is suggested that the user should prepare his data on cards and read the cards into his own file. Then this data can be read in to the program at any time by typing in the statement <DATA <channel no. assigned to file>;>. If this file contains the statement <DATA 50 ;> in place of any<ANALYSE ;> statements then control of the program will return to the teletype so that any errors detected in the data can be corrected before an<ANALYSE ;> statement is typed in on the teletype.

A.5.9. DATA ERRORS

A.5.9.1. Data Preparation

If any errors are made whilst typing in or preparing the data for the program then it may be possible to erase these errors as described below.

Erase syntax

If it is required to erase a character group from the data then that group must be followed by the symbol % immediately following that character group.

```
e.g. NET LANE% LINE WL 75.0% 50.0 0.37 1E9 0.1 ;
```

Abandon statement

If it is required to terminate a statement before it is completed then a %% must be typed in. The statement will then be abandoned at that point and if the first % also follows the last character group then that character group will be ignored.

```
e.g. NET 8 LINE WL 50.0 0.1 %%
     NET 8 LINE LG 50.0 0.1 0.75 0.1 ;
```

It should be noted that the statement is only abandoned and not completely ignored. Thus the information which may have been stored from this statement must be deleted using the DELETE statement or overwritten using the next statement in the data.

A.5.9.2. Syntax Errors

All syntax errors are detected in the program and it is impossible to corrupt the program by errors in the data unless a channel assignment is made for the input or output of the program to an unsuitable channel. When a syntax error is detected then the message <** SYNTAX ERROR>followed by the error number and the last character group read in is printed out. The cause of each syntax error number is given in section A.5.10. A syntax error will not cause the current statement to be ignored and any data formed by this statement should be corrected by later deletion or by over writing in further statements in the data. All syntax errors can be corrected by further statements in the data but the fault indicator must be cleared, using the statement <NOFAULT ;> before the program will allow an analysis statement to be obeyed.

A.5.9.3. Circuit Errors

Following an<ANALYSE ;> statement in the data a check is made to ensure that the circuit to be analysed, including any sub-circuits used in this circuit, is sufficiently defined for analysis. If it is not then the message <** CIRCUIT ERROR> followed by an error number is printed out for each error detected and the ANALYSE statement is ignored. The cause of each error number is given in section A.5.11.

A.5.9.4. Messages

The messages given in section A.5.12 may be printed out during the running of the program. Some of these messages just indicate to the user that a particular statement has been accepted whilst others indicate errors in the data and two of the messages will be followed by the program being abandoned.

A.5.10. SYNTAX ERRORS

Error no.	Statement	Cause
1	any	statement does not start with a word
2	any	first word of statement not acceptable
3	any	circuit no. not integer
4	any	statement not terminated in <;>
5	any	more than 6 digits in an integer
6	any	no digits in a real number
11	network	network no. not integer
12	network	network sub-statement does not start with word
14	network	network type sub-statement not given before a JUNCT, PAR or CONN sub-statement
15	network	network parameter not real number
16	network	parameter no. not integer
17	network	parameter no. not acceptable
18	network	connection no. not integer
19	network	connection no. not acceptable
20	network	junction no. not integer
21	network	junction no. not integer
22	network	SERI or SHNT not followed by Z type word
23	network	sub-circuit no. not integer
24	network	no. of ports on sub-circuit not integer
25	network	LINE or CPLINE not followed by line type word
26	network	LINE or CPLINE line type word not acceptable
27	network	network type word not acceptable as network type or sub-program name
28	network	network sub-type word for sub-program not given.
31	junction	junction type not word
32	junction	junction type word not SERIES or PARALLEL
33	junction	junction no. not integer
34	junction	junction no. not present in circuit
41	ports	junction no. not integer
45	char.imped.	char. imped. no. not integer
46	char.imped.	char. imped. type not word
47	char.imped.	char. imped. type word not acceptable
48	char.imped.	char. imped. parameter not real number
49	char.imped.	char. imped. not terminate in <;>

Error no.	Statement	Cause
51	frequency	frequency not real number
52	frequency	word in frequency list not STEPLIN or STEPLOG
53	frequency	lower frequency in STEP not real number
54	frequency	upper frequency in STEP not real number
55	frequency	number of steps in STEP not integer
61	output/spec	option does not start with word
62	output/spec	option word not acceptable
63	output/spec	option port no. not integer
64	output/spec	option format not word
65	output/spec	option format word not acceptable
66	output/spec	option format DBARG or DB not acceptable
67	spec	frequency value not real number
68	spec	option value not real number
69	spec	weight not real number
71	delete	delete element does not start with word
72	delete	delete element word not acceptable
73	delete	circuit number not set
74	delete	delete element no. not integer
75	delete	delete element not present in circuit
81	data	channel no. not integer
82	error	channel no. not integer
83	result	channel no. not integer
101	vary	vary element not word
102	vary	vary element word not acceptable
103	vary	vary element no. not integer
104	vary	parameter no. not integer
105	vary	upper limit for variable not real number
106	vary	upper limit for variable less than lower limit
107	vary	word PAR not given for element
110	analyse/ optimise	fault indicator set
111	optimise	optimisation program name not given
113	optimise	no. of iterations not integer
114	optimise	optimisation program not in store
120	display	no results stored

N.B. All errors greater than 1000 are produced by the sub-programs used by the MICRO3 program.

A.5.11. CIRCUIT ERRORS

Error no.	Cause
1	circuit not in store
2	junction type not set on at least one junction
3	network connection not set on at least one network
4	no junctions or more than 8 junctions assigned as external ports for the circuit
6	no. of external ports on sub-circuit does not match no. of connections on call of sub-circuit
21	no char. imped. defined
22	no frequency list defined for analysis
23	no output or spec list defined
24	option port no. in output options list not acceptable for analysis
25	no vary list defined for optimisation
26	circuit no. for element in vary list for optimisation not defined
27	ZO or NET no. on vary element in vary list for optimisation not defined in the circuit
28	parameter no. for ZO or NET in vary list for optimisation not acceptable

A.5.12. RUN TIME MESSAGES

During Reading of the Data

Message	Meaning	Continuation
** SYNTAX ERROR <no.>-<syntax>	A syntax error has been detected in the data	Fault indicator set
** CIRCUIT ERROR <no.>	A circuit error has been detected whilst checking for an analysis or optimisation	Fault indicator set
** DATA	more input data is requested after an analysis, optimisation or at the start of the program run	to read more data
** CIRCUIT <no.>	This circuit no. is being checked for errors	to check circuit

Message	Meaning	Continuation
** ANALYSIS	An analysis has been started	to carry out a circuit analysis
** OPTIMISE	An optimisation has been started	to carry out a circuit optimisation
** ASSIGNED	An input/output channel for the program has been reassigned	with new channel set
** RESTART	The data structure has been cleared after a newrun statement	with a completely new start in the data but with the old channel assignments
** RUNEND	The program has been terminated	program run terminated
** IST OFLO	The data structure formed is too large to be stored	program run terminated
** DISPLAY PLOT	Display statement accepted	control of program transfered to visual display

During an Analysis or Optimisation

Message	Meaning	Continuation
** SINGULAR	A matrix operation can not be performed at the current frequency in the analysis or optimisation	next set of results at the current frequency incorrect
** EX SINGULAR	as for <** SINGULAR>on a different matrix operation	next set of results at the current frequency incorrect

During Dumping Graphs from the Visual Display

Message	Meaning	Continuation
**	Request for the program to dump a graph onto the digital plotter	If a <.> is typed in the dump continues otherwise if any other character dump ignored
** DISPLAY DUMP	A graph is being dumped onto the digital plotter	to dump graph
** END DUMP	The dump of a graph is complete	return to control from visual display

A.5.13. EXAMPLE, 3dB BRANCH ARM HYBRID RING

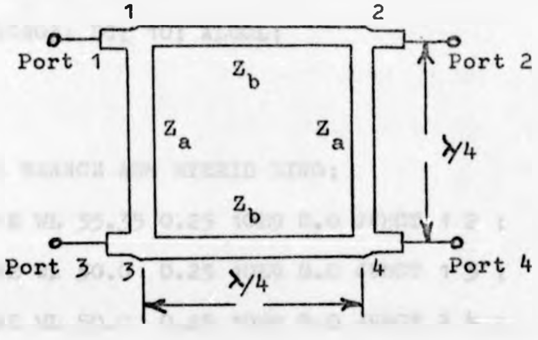


Fig. A.5.3.- 3dB Branch Arm Hybrid Ring

Problem

It is required to determine the input impedance and input reflection coefficient in complex form, the transmission between ports 1 and 2, 1 and 3, and 1 and 4 in dB and arguement, and the input V.S.W.R. for the 3dB branch arm hybrid ring in Fig. A.5.3. The frequency range of interest is 5 to 15 GHz in steps of 1 GHz. The characteristic impedance of the system is 50 ohm (Z_o) and the centre design frequency is 10 GHz. The line impedances $Z_a = Z_o$ and $Z_b = Z_o/\sqrt{2}$. An analysis is required with no line losses and with line losses of 0.1 dB/wavelength.

Data

The data for the 3dB branch arm hybrid ring is shown in Table A.5.3.

```
&JOB; <job number> ;

&ASSIGN; 50; CR;

&LOAD; MICRO3; DC; 10; ALGOL;

&RUN;


LABEL 3DB BRANCH ARM HYBRID RING;

NET 1 LINE WL 35.35 0.25 10E9 0.0 JUNCT 1 2 ;

NET 2 LINE WL 50.0  0.25 10E9 0.0 JUNCT 1 3 ;

NET 3 LINE WL 50.0  0.25 10E9 0.0 JUNCT 2 4 ;

NET 4 LINE WL 35.35 0.25 10E9 0.0 JUNCT 3 4 ;

JUNCT PARALLEL 1 2 3 4 ;

PORTS 1 2 3 4 ;

ZO 1 R 50.0 ;

FREQ STEPLIN 5E9 15E9 10 ;

OUTPUT  ZIN 1 CMPX  SPAR 1 1 CMPX  SPAR 2 1 DBARG

        SPAR 3 1 DBARG  SPAR 4 1 DBARG  VSWR 1 ;

LABEL LINES LOSSLESS ;

ANALYSE;

NET 1 PAR 4 0.1 ;   NET 2 PAR 4 0.1 ;

NET 3 PAR 4 0.1 ;   NET 4 PAR 4 0.1 ;

ANALYSE;

END;


&END;
```

Table A.5.3. - Data for 3dB Branch Arm Hybrid Ring

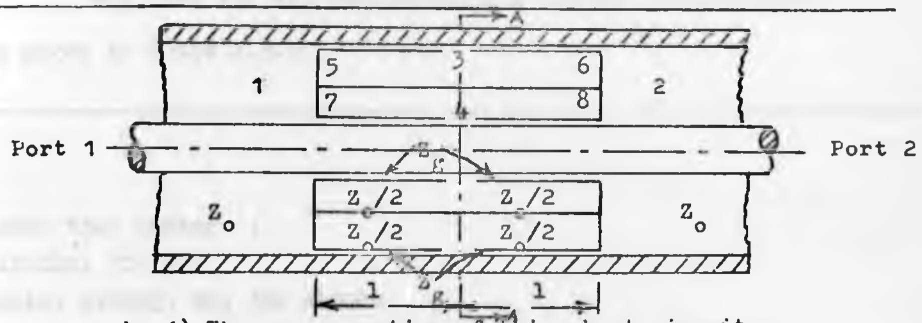
Results

```
** DATA

LABEL 3DB BRANCH ARM HYBRID RING;
NET 1 LINE WL 35,35 0,25 10E9 0 JUNCT 1 2 ;
NET 2 LINE WL 50,0  0,25 10E9 0 JUNCT 1 3 ;
NET 3 LINE WL 50,0  0,25 10E9 0 JUNCT 2 4 ;
NET 4 LINE WL 35,35 0,25 10E9 0 JUNCT 3 4 ;
JUNCT PARALLEL 1 2 3 4 ;
PORTS 1 2 3 4 ;
ZO 1 R 50 ;
FREQ STEPLIN 5E9 15E9 10;
OUTPUT ZIN 1 CMPX   SPAR 1 1 CMPX   SPAR 2 1 DBARG
         SPAR 3 1 DBARG   SPAR 4 1 DBARG   VSWR 1 ;
LABEL HYBRID RING WITH NO LINE LOSSES ;
ANALYSE;
** CIRCUIT      1
** ANALYSIS


 HYBRID RING WITH NO LINE LOSSES

  FREQ     SPAR      1   1   SPAR     2   1   SPAR     3   1   SPAR     4   1  INPUT IMPED  1     VSWR  1
  (GHZ)        (PU)          (DB)   (DEG)      (DB)   (DEG)     (DB)   (DEG)       (OHM)          (PU)
  5,0000  -,5716 J ,1938  -7,827  -42,20    -7,347  -66,59   -5,410  -79,35   12,71 J 7,739     4,0330
  6,0000  -,5016 J ,2352  -7,757  -44,34    -7,742  -81,24   -4,799  -95,77   14,28 J 12,21     3,7283
  7,0000  -,3615 J ,3474  -6,863  -46,42    -8,440  -99,09   -4,041 -114,20   18,65 J 17,42     3,0520
  8,0000  -,1832 J ,3234  -5,195  -53,63   -10,17 -121,87    -3,362 -135,28   28,30 J 21,30     2,1985
  9,0000  -,0435 J ,1864  -3,619  -69,15   -14,89 -149,66    -3,044 -157,94   42,69 J 16,52     1,4748
 10,000   -,0012 J ,0000  -3,009  -90,00   -76,42   90,00    -3,012  180,00   49,96 J ,0000     1,0013
 11,000   -,0436 J-,1364  -3,619 -110,85   -14,89  -30,32    -3,044  157,94   42,69 J-16,52     1,4748
 12,000   -,1832 J-,3234  -5,195 -126,37   -10,17  -58,13    -3,362  135,28   28,30 J-21,30     2,1985
 13,000   -,3615 J-,3474  -6,863 -133,58   -8,440  -60,91    -4,041  114,20   18,65 J-17,42     3,0520
 14,000   -,5016 J-,2352  -7,757 -135,66   -7,742  -98,75    -4,799   95,77   14,28 J-12,21     3,7283
 15,000   -,5716 J-,1938  -7,827 -137,80   -7,347 -113,41    -5,410   79,35   12,71 J-7,739     4,0330
```

```
** DATA

NET 1 PAR 4 0,1; NET 2 PAR 4 0,1 ;
NET 3 PAR 4 0,1 ; NET 4 PAR 4 0.1 ;
LABEL HYBRID RING WITH LINE LOSSES ;
ANALYSE;
** CIRCUIT      )
** ANALYSIS

 HYBRID RING WITH LINE LOSSES

  FREQ      SPAR      1  1    SPAR      2  1     SPAR     3  1    SPAR      4  1   INPUT IMPED  1      VSWR  1
  (GHZ)          (PU)           (DB)    (DEG)      (DB)    (DEG)      (DB)    (DEG)        (OHM)            (PU)
  5,0000   -,5594 J ,1933   -7,854   -42,18    -7,374   -66.56    -5,441   -79.39    12,77 J 7,730       4.0165
  6,0000   -,5034 J ,2358   -7,750   -44,36    -7,776   -81,20    -4,836   -95.84    14,35 J 12.17       3,7090
  7,0000   -,3611 J ,3446   -6,672   -46,55    -8,486   -98.99    -4,056  -114.27    18,73 J 17,32       3,0353
  8,0000   -,1916 J ,3205   -5,224   -53,33   -10,23  -121,58    -3,416  -135,34    28,32 J 21,08       2,1692
  9,0000   -,0436 J ,1847   -3,659   -69,29   -14,97  -148,79    -3,103  -157,95    42,49 J 16.29       1,4722
  10,000   -,0036 J ,0000   -3,069   -90,00   -49,65   -90.00    -3,072   180,00    49,64 J ,0000       1,0072
  11,000   -,0430 J-,1345   -3,674  -110,70   -14,98   -31.31    -3,110   157,95    42,47 J-16,27       1.4719
  12,000   -,1919 J-,3199   -5,250  -126,13   -10,24   -58.49    -3,430   135,35    28,32 J-21,04       2,1872




  13,000   -,3679 J-,3432   -6,875  -133,40    -8,593   -81.05    -4,103   114.30    18,76 J-17,29       3,0293
  14,000   -,4913 J-,2836   -7,762  -135,60    -7,794   -98,82    -4,855    95,87    14,38 J-12,15       3,6989
  15,000   -,5615 J-,1929   -7,838  -137,83    -7,394  -113.46    -5,464    79,42    12,80 J-7,723       4,0046
```

```
** DATA

 END;

 LAST  ADDR =   369

 ** RUNEND



STORE LEFT 49770  USED 16831


 &END;
TIME =  0000  21.335
   A
```

A.5.14. EXAMPLE, COAXIAL SHORT CIRCUIT



n.b. 1) The cross section of this short circuit across A-A is radial about its centre line.

2) All solid lines represent conducting boundaries.

Fig. A.5.4. - Coaxial Short Circuit

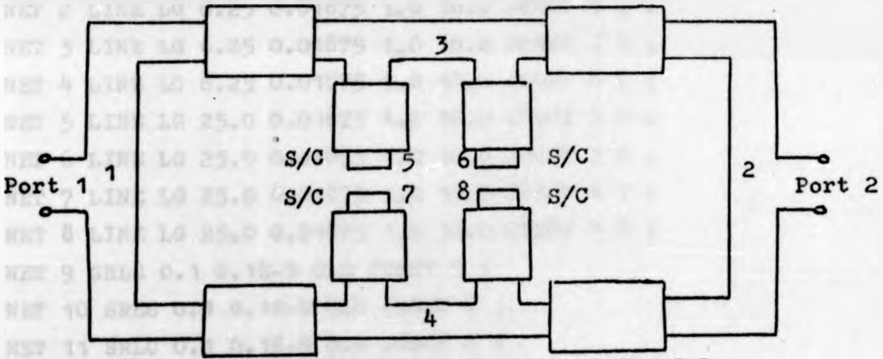


Fig. A.5.5. - Equivalent Circuit of Coaxial Short Circuit

Problem

It is required to determine the input impedance, in modulus and arguement, the input reflection coefficient both in modulus and arguement and in dB, and the transmission in complex form for the above non-contacting coaxial short circuit. The frequency range of interest is 0.5 to 10 GHz in steps of 0.5 GHz with a centre design frequency of 4 GHz. The characteristic impedance of the system is 50 ohm (Z_o). The gap impedances, Z_g, are 6.25 ohm and the length l is 1.875 cm .The lines are all air spaced with a loss of 10 dB/m and the short circuits at junctions 5, 6,7 and 8 are assumed equivalent to a series R L circuit of 0.1 ohm and 0.1 nH .

Data

The data for the non-contacting coaxial short circuit is given in Table A.5.5.

```
&JOB; <job number > ;
&ASSIGN; 50; CR;
&LOAD; MICRO3; DC; 10; ALGOL;
&RUN;

LABEL NON CONTACTING COAXIAL SHORT CIRCUIT ;
NET 1 LINE LG 6.25 0.01875 1.0 10.0 JUNCT 1 3 ;
NET 2 LINE LG 6.25 0.01875 1.0 10.0 JUNCT 3 2 ;
NET 3 LINE LG 6.25 0.01875 1.0 10.0 JUNCT 1 4 ;
NET 4 LINE LG 6.25 0.01875 1.0 10.0 JUNCT 4 2 ;
NET 5 LINE LG 25.0 0.01875 1.0 10.0 JUNCT 3 5 ;
NET 6 LINE LG 25.0 0.01875 1.0 10.0 JUNCT 3 6 ;
NET 7 LINE LG 25.0 0.01875 1.0 10.0 JUNCT 4 7 ;
NET 8 LINE LG 25.0 0.01875 1.0 10.0 JUNCT 4 8 ;
NET 9 SRLC 0.1 0.1E-9 0.0 JUNCT 5 ;
NET 10 SRLC 0.1 0.1E-9 0.0 JUNCT 6 ;
NET 11 SRLC 0.1 0.1E-9 0.0 JUNCT 7 ;
NET 12 SRLC 0.1 0.1E-9 0.0 JUNCT 8 ;
JUNCT SERIES 1 2 3 4 5 6 7 8 ;
PORTS 1 2 ;
ZO 1 R 50.0 ;
FREQ STEPLIN 0.5E9 10E9 19 ;
OUTPUT  ZIN 1 MODARG  SPAR 1 1 MODARG
        SPAR 2 1 CMPX  SPAR 1 1 DBARG  VSWR 1 ;
LABEL LINE LOSSES OF 10DB/M ;
ANALYSE;
END;

&END;
```

Table A.5.5. - Data for Coaxial Short Circuit

Results

```
** DATA

LABEL NON CONTACTING COAXIAL SHORT CIRCUIT ;
NET 1 LINE LG 6.25 0,01875 1.0 10,0 JUNCT 1 3 ;
NET 2 LINE LG 6.25 0,01875 1.0 10,0 JUNCT 3 2 ;
NET 3 LINE LG 6.25 0,01875 1.0 10,0 JUNCT 1 4 ;
NET 4 LINE LG 6.25 0,01875 1.0 10.0 JUNCT 4 2 ;
NET 5 LINE LG 23,0 0,01875 1.0 10,0 JUNCT 3 5 ;
NET 6 LINE LG 23,0 0,01875 1.0 10,0 JUNCT 3 6 ;
NET 7 LINE LG 23,0 0,01875 1.0 10.0 JUNCT 3 6 ;
NET 8 LINE LG 23,0 0,01875 1.0 10,0 JUNCT 4 8 ;
NET 9 SRLC 0,1 0,1E-9 0 JUNCT 5 ;
NET 10 SRLC 0,1 0.1E-9 0 JUNCT 6 ;
NET 11 SRLC 0,1 0.1E-9 0 JUNCT 7 ;
NET 12 SRLC 0.1 0.1E-9 0 JUNCT 8 ;
JUNCT SERIES 1 2 3 4 5 6 7 8 ;
PORTS 1 2 ;    Z0 1 R 50 ;
FREQ STEPLIN 0,5E9 10E9 19 ;
OUTPUT    ZIN 1 MODARG    SPAR 1 1 MODARG
          SPAR 2 1 CMPX    SPAR 1 1 DBARG    VSWR 1 ;
LABEL SHORT CIRCUIT WITH LINE LOSSES 10DB/M ;
ANALYSE;
** CIRCUIT       1
** ANALYSIS


 SHORT CIRCUIT WITH LINE LOSSES 10DB/M

   FREQ      SPAR     1  1    SPAR      2  1    SPAR     1  1   INPUT IMPED  1     VSWR  1
   (GHZ)     (PU)    (DEG)          (PU)          (DB)    (DEG)   (OHM)   (DEG)      (PU)
   ,50000    ,3770   -13,47   -,2318 J ,3493   -8,472   -13.47   107,1   -11,57     2,2104
   1,0000    ,5548   -27,63    ,2151 J ,1604   -5,086   -27.63   133,2   -36,82     3,5123
   1,5000    ,4437   -18,52    ,2917 J ,1753   -6,961   -18.52   121,0   -19,64     2,6278
   2,0000    ,3320   -64,18    ,0771 J-,0687   -9,577   -64.18   65,28   -33,89     1,9941
   2,5000    ,2014   -74,58    ,0194 J-,0344   -13,92   -74.58   55,44   -22.03     1,5044
   3,0000    ,1118   -79,15    ,0057 J-,0158   -18,50   -79.15   52,26   -13,32     1,2697
   3,5000    ,0514   -79,77    ,0017 J-,0048   -25,29   -79.77   50,97    -6,13     1,1151
   4,0000    ,0076    31,92    ,0011 J ,0038   -42,37    31.92   50,65     0,46     1,0153
 4,5000    ,0635   79,92    ,0031 J ,0132   -23,94    79.92   51,12    7,16      1.1357
 5,0000    ,1317   78,21    ,0101 J ,0274   -17,61    78.21   52,72   14,79      1,3035
 5,5000    ,2259   72,08    ,1351 J ,0545   -12,92    72.08   57,12   24.37      1,5836
 6,0000    ,3975   52,12    ,1773 J ,0843   -7,969    52.12   78,53   36,89      2,3306
 6,5000    ,2022   45,42    ,0071 J-,1749   -13.89    45.42   66,14   16,72      1,5560
 7,0000    ,3610    6,20    ,0973 J-,3786   -8,404     6.20   110,5    5,48      2,2259
 7,5000    ,3619   27,33  -,2791 J-,2680   -8,757    27.33   95,64   21.13      2,1490
 8,0000    ,4910    7,77  -,4340 J ,0738   -6,179     7.77   143,7    9,93      2,9292
 8,5000    ,4619   -3,38  -,1296 J ,4111   -6,652    -3.38   136,5   -4,00      2,7379
 9,0000    ,4215  -45,10    ,0217 J ,1191   -7,505   -45.10   87,22  -35,98      2,4569
 9,5000    ,4312  -48,23    ,1699 J-,0444   -6,299   -48.23   89,30  -43.34      2,8776
10,000     ,2912  -68,63    ,0403 J-,0447   -10,63   -68.63   61,07  -30,96      1,8339

** DATA

END;

LAST  ADDR =   175



** RUNEND



STORE LEFT 49682  USED 16919


 &END;
TIME =  0000  24,123
   A
```

Appendix A.6

The Computer Aided Design of Microwave Circuits

This Appendix is a reprint from the following paper :-

B.G. Marchent, 'The Computer Aided Design of Microwave Circuits', IREE Czechoslovakia, Summer School on Circuit Theory 1971, Short Contributions Vol. 2, pp. 255 - 263

THE COMPUTER AIDED DESIGN OF MICROWAVE CIRCUITS

by B.G. Marchent

This paper is concerned with the frequency domain analysis of linear microwave circuits. A method of analysis for microwave circuits using mixed matrices will be described with a computer program which uses this method of analysis.

All microwave components involve the transmission of a signal between one port of an n-port network and another port of the same network. Every microwave component can thus be completely described within an n-port network. Thus a microwave circuit can be considered as a general assembly of n-port networks. In n-port network theory we are only interested in the voltage and current at each port on each network. Thus, in practice, it may be necessary to include an ideal transformer with unity turns ratio in cascade with each port on each network to realise the correct equivalent circuit.

Previously, for the frequency domain analysis of microwave circuits, either a chain matrix(1 - 4) or a nodal(5) analysis has been used. The chain matrix analysis is simple to apply to a cascade of 2-port networks but it is very difficult, if not impossible, to apply it to a general assembly of n-port networks. On the other hand the nodal analysis can be applied to some assemblies of n-port networks but it is mainly intended for the analysis of lumped element circuits. In particular it is difficult to apply if series junction connections of the networks, see Fig. 1, are present in the circuit. The mixed matrix method of analysis which will be described in this paper has been specially designed for the analysis of general assemblies of n-port networks.

In an assembly of n-port networks two types of interconnection of the networks at junctions may be defined. These are the series and parallel connections of the networks as shown in Fig. 1 and 2. If a junction is a mixture of these two types then a dummy network, possibly an ideal transformer with unity turns ratio, must be included to break the junction into separate acceptable junctions. An example of the use of parallel junctions is shown in the microwave integrated phase shifter in Fig. 3. In this a plan view of the microstrip lines in the circuit is shown and below all these a ground plane is assumed to be present. An example of the use of series junctions is shown in the non-contacting coaxial short circuit in Fig. 4.

In Fig. 1, at a series junction, the current i_s is common and the voltage v_s is the sum of the input voltages into all the networksconnected to that junction. In Fig. 2, at a parallel junction, the voltage v_p is common and the current i_p is the sum of the input currents into all the networks connected to that junction. Thus we can set up the follow equation relating the sum to common variables for every junction in the circuit :-

$$\begin{bmatrix} V_s \\ I_p \end{bmatrix} = \begin{bmatrix} Z & C \\ D & Y \end{bmatrix} . \begin{bmatrix} I_s \\ V_p \end{bmatrix}$$

where V and I are column vectors and suffix 's' indicates all the series junctions and suffix 'p' indicates all the parallel junctions. The ZCDY mixed matrix is a square matrix but in general it will not be symmetric.

Initially a zero matrix is set up for the mixed matrix which implies that all the series junctions are short circuited and all the parallel junctions are open circuited. A mixed matrix is then set up for each network in the circuit in turn and added to this zero matrix. Once the full mixed matrix has been set up the internal junctions will have no current generator or voltage generatorpresent so v_s or i_p will be zero for these junctions and they may be eliminated from the full mixed matrix using the normal procedure of Guass elimination to give the reduced mixed matrix of the form :-

$$\begin{bmatrix} V'_s \\ I'_p \end{bmatrix} = \begin{bmatrix} Z' & C' \\ D' & Y' \end{bmatrix} . \begin{bmatrix} I'_s \\ V'_p \end{bmatrix}$$

where only the junctions which have been assigned as external ports, for the insertion or extraction of signals to or from the circuit, now remain in the mixed matrix. It is usual , in the Guass elimination, to employ sparse matrix techniques with row interchanges to improve the computation time and accuracy.

So far it has been assumed that a mixed matrix is available to describe each network in the circuit but in practice the required mixed matrix may be difficult to obtain. This is because it is necessary to get the correct voltage and current variables on the correct side of the mixed matrix equation for the network. In the case of a 2-port network it may be connected from a series to a series junction, from a parallel to a parallel junction, from a series to a parallel junction or from a parallel to a series junction which produces 4 possible mixed matrices for that network. In general for an n-port network there are 2^n possible mixed matrices one of which will be the one required for the given set of network connections. I will try to describe how this problem is overcome.

Fig. 5 shows some very simple one and two port lumped element networks. For these all the possible mixed matrix options may be provided for in a computer program taking advantage of similarities in the mixed matrix equations. For complex lumped element networks a separate method of circuit analysis for lumped element circuits must be used. A very suitable one for this application is the mixed mesh and cutset analysis described by Branin(6). Fig. 6 shows an example of a typical lumped element circuit. In the mixed mesh and cutset analysis a search is made for a tree in the circuit which must contain all the voltage generators but no current generators. All the other lumped elements which are not in the tree are then link elements. The analysis of the lumped circuit is then carried out in terms of the tree element voltages and the link element currents. In the lumped network the ports connected to series junctions, as current generators, will be considered as link elements whilst the ports connected to parallel junctions, as voltage genertors, will be considered as tree elements. Thus the mixed mesh and cutset analysis of a lumped element circuit will automatically produce the required mixed matrix for the network.

In the case of distributed networks it is usually fairly easy to set up the scattering matrix for the network in the form :-

$$\begin{bmatrix} b_s \\ b_p \end{bmatrix} = \begin{bmatrix} S_A & S_B \\ S_C & S_D \end{bmatrix} . \begin{bmatrix} a_s \\ a_p \end{bmatrix}$$

where the b and a terms are column vectors and the scattering matrix is assumed to be partitioned. The scattering matrix can be translated to the required mixed matrix using the following equation:-

$$\begin{bmatrix} V_s \\ I_p \end{bmatrix} = \begin{bmatrix} \Phi - S_A & -S_B R_{op} \\ S_C & (\Phi + S_D) R_{op} \end{bmatrix}^{-1} . \begin{bmatrix} (\Phi + S_A) R_{os} & S_B \\ -S_C R_{os} & \Phi - S_D \end{bmatrix} . \begin{bmatrix} I_s \\ V_p \end{bmatrix}$$

where Φ is a unit diagonal matrix and the extra suffix 'o' defines the characteristic impedance for the series and parallel junctions. In this equation all the values of R_o are real and equal.

Once the full mixed matrix has been set up for the entire circuit and reduced it may be translated to scattering parameters for the entire circuit between its external ports using the following equation :-

$$\begin{bmatrix} b_s \\ b_p \end{bmatrix} = \begin{bmatrix} R_{os}^{\frac{1}{2}} & 0 \\ 0 & R_{op}^{\frac{1}{2}} \end{bmatrix} . \begin{bmatrix} Z' + Z_{os} & -C' Z_{op} \\ +D' & -\Phi + Y' Z_{op} \end{bmatrix}^{-1} .$$

$$\begin{bmatrix} Z' - Z_{os}^{*} & C' Z_{op}^{*} \\ -D' & -\Phi - Y' Z_{op}^{*} \end{bmatrix} . \begin{bmatrix} R_{os}^{-\frac{1}{2}} & 0 \\ 0 & R_{op}^{-\frac{1}{2}} \end{bmatrix} . \begin{bmatrix} a_s \\ a_p \end{bmatrix}$$

where Z_o is of the form $R_o + j X_o$ and may be different at each external port and * indicates the complex conjugate.

The computer program written to use the mixed matrix method of analysis for the frequency domain analysis of linear microwave circuits is called MICRO2[7]. It does not as yet include all the facilities required for actual Computer Aided Design of Microwave Circuits but it has helped to prove, in a program with a very versitile structure, that this method of mixed matrix analysis does work and produces the correct results. The networks that are at present included in this program are very trival but it is hoped to improve the program in the near future with microwave components in waveguide, microstrip, coaxial line etc.

A practical example for this program is shown in the microwave integratec circuit phase shifter in Fig. 3 and the data for the computer program for this example is shown in Fig. 7. The data consists of a set of statements which may be given in any order with each one terminated in a semi-colon. The program provides complete diagnostic facilities and any errors in the data are printed out so they may be later corrected. The program may be used on batch processing or on-line on a teletype. In the latter case errors in the data may be corrected as they are detected. Also the results of a circuit analysis may be displayed, in graphical form, on an on-line CRT display on a rectangular grid, polar grid or on a smith chart with the option of dumping the graph onto a digital plotter. On the CRT display the graph which is to be plotted is selected by means of a light pen pointed at a menu on the CRT display.

REFERENCES

1) McPhun, M.K. : 'A Computer Program for the Analysis of Branched, Distributed and Lumped Circuits', IEE Conference Publication, 1966, No. 23, p89-124.

2) Green, P.E. : 'General Purpose Programs for the Frequency Domain Analysis of Microwave Circuits', IEEE Trans., 1969, MTT-17, No. 8, p506-514.

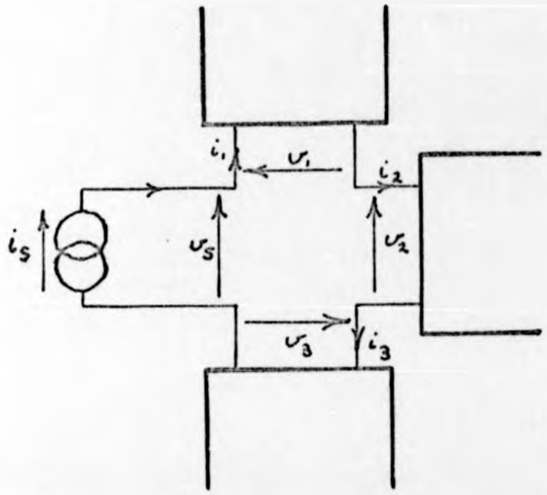
3) Parker, W.N. :' DIPNET : A General Distributed Parameter Network Analysis Program', IEEE Trans., 1969, MTT-17, No. 8, p495-505.

4) Marchent, B.G. : 'CHAIN1 : A Frequency Domain Circuit Analysis Program using Chain Matrices', University of Warwick, 1970, School of Eng. Science Report No. 64.

5) Murray-Lasso, M.A. : 'Microwave Circuit Design by Digital Computer ', IEEE Trans., MTT-17, No. 8 , p514-526.

6) Branin, F.H. : 'Computer Methods of Network Analysis', Proc. IEEE, 1967, Vol. 55, No. 11, p1787-1801.

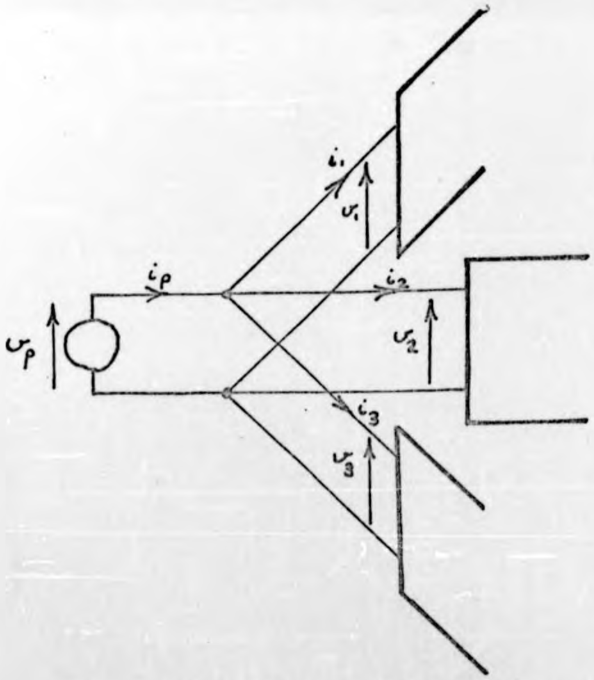
7) Marchent, B.G. : 'MICRO2 : A Frequency Domain Circuit Analysis Program for Microwave Circuits using Mixed Matrices', University of Warwick, 1971, School of Eng. Science Report No. 65.



$$i_j = i_s$$

$$v_s = \sum_j v_j$$

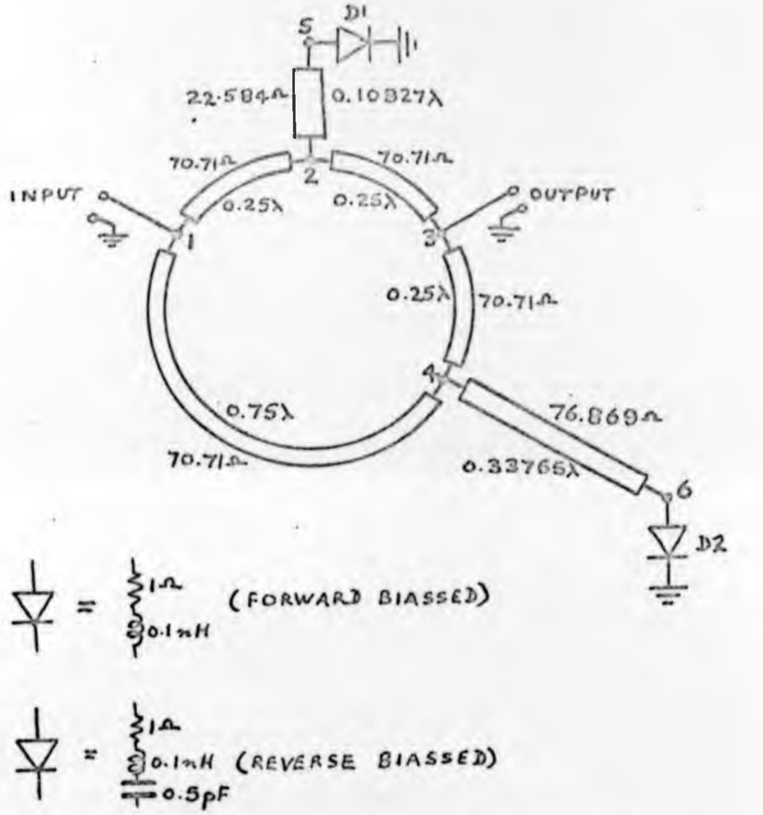
Fig. 1 - Series Junction



$$v_j = v_p$$

$$i_p = \sum_j i_j$$

Fig. 2 - Parallel Junction



n.b. 1) A ground plane is assumed present below all lines.

2) Centre design frequency is 2GHz

3) All line losses 0.1dB/wavelength

Fig. 3 - Practical Phase Shifter

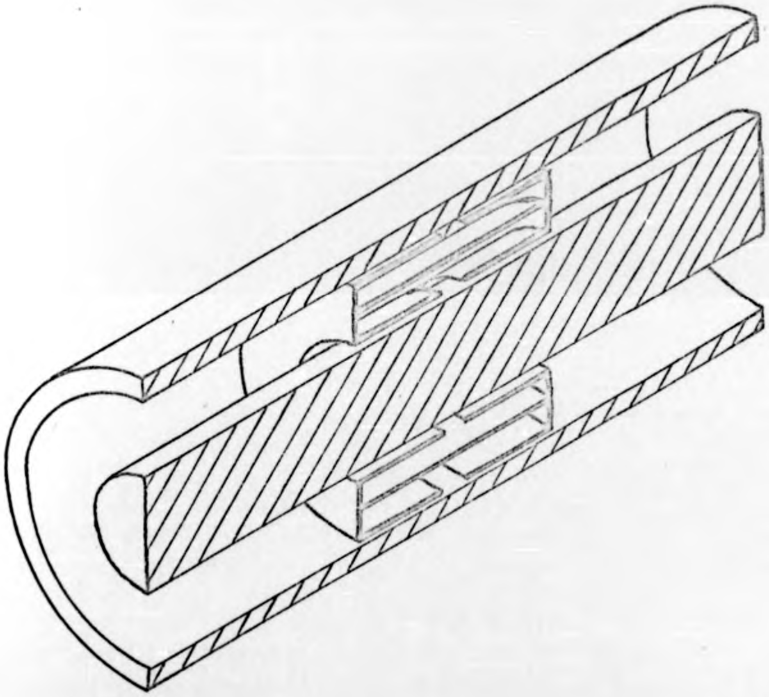


Fig. 4 - Non Contacting Short Circuit

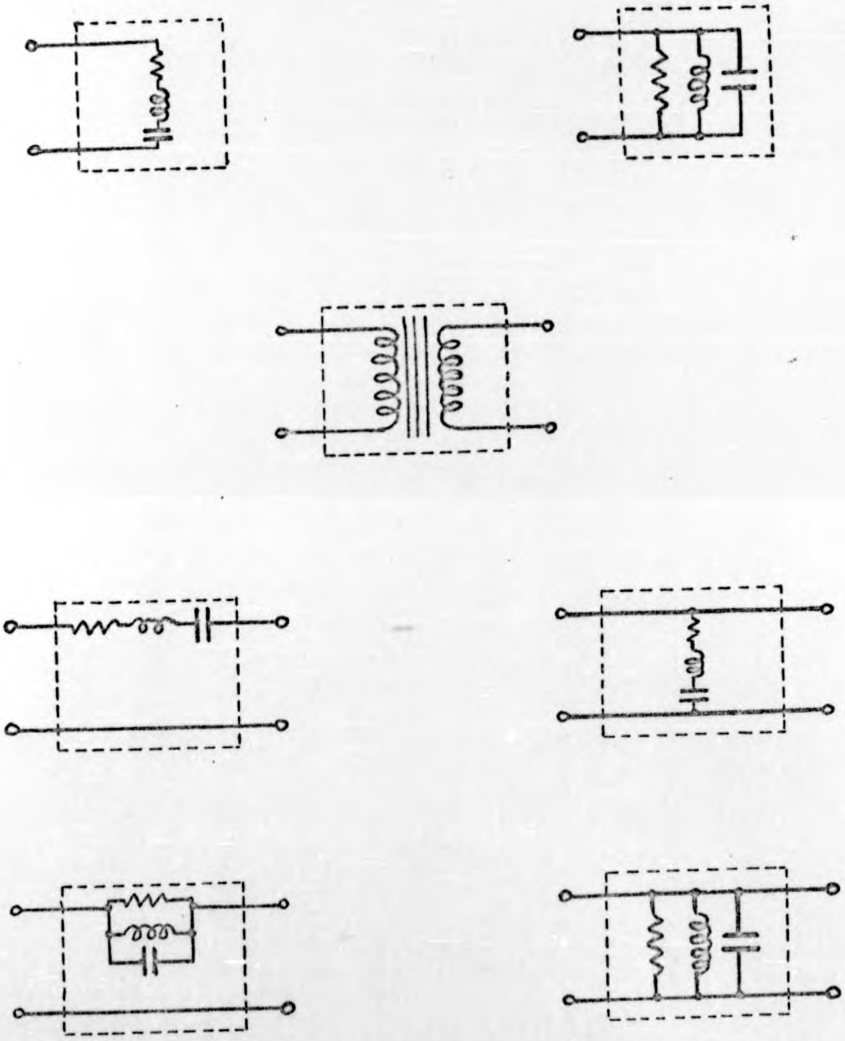
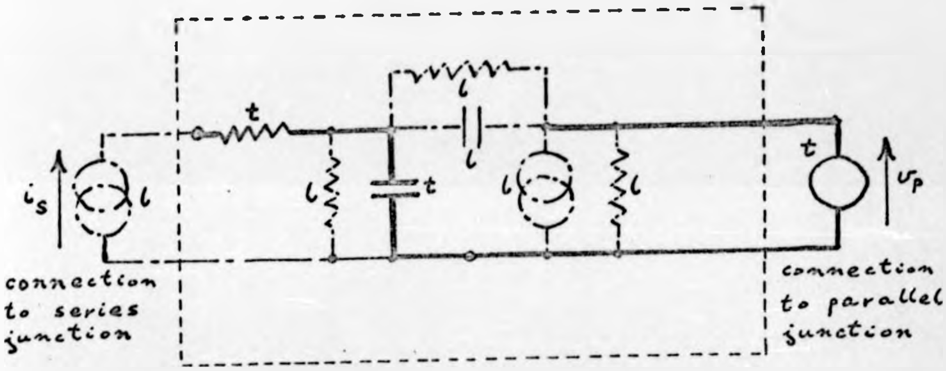


Fig. 5 - Simple Lumped Networks


t
l
l
i_s
l
l
t
l
l
t
v_p
connection to series junction
connection to parallel junction

t = tree element

l = link element

Fig. 6 - More Complex Lumped Network

```
&JOB;
&LOAD; MICRO2; DC; 10; ALGOL;
&ASSIGN; 50; CR;
&RUN;

LABEL  CIRCULAR HYBRID RING 90 DEG PHASE SHIFTER;
NETWORK  1  LINEWL  70.71  0.25  2E9  0.1  JUNCTION  1  2 ;
NETWORK  2  LINEWL  70.71  0.25  2E9  0.1  JUNCTION  2  3 ;
NETWORK  3  LINEWL  70.71  0.25  2E9  0.1  JUNCTION  3  4 ;
NETWORK  4  LINEWL  70.71  0.75  2E9  0.1  JUNCTION  4  1 ;
NETWORK  5  LINEWL  22.584  0.10927  2E9  0.1  JUNCTION  2  5 ;
NETWORK  6  LINEWL  76.869  0.33765  2E9  0.1  JUNCTION  4  6 ;
NETWORK  7  LOADSRLC  1.0  0.1E-9  0  JUNCTION  5 ;
NETWORK  8  LOADSRLC  1.0  0.1E-9  0  JUNCTION  6 ;
PORT  1  RJX  50  0  JUNCTION  1 ;
PORT  2  RJX  50  0  JUNCTION  3 ;
JUNCTION PARALLEL  1  2  3  4  5  6 ;
OUTPUT  ZIN 1 CMPX   SPAR 1 1 MODARG   SPAR 2 1 MODARG
            SPAR 1 1 DBARG   SPAR 2 1 DBARG   VSWR 1 ;
FREQUENCY  STEPLIN 1.5E9  2.5E9  20 ;
LABEL  DIODES FORWARD BIASSED WITH LINE LOSSES OF 0.1 DB/WL;
COMPUTE;
NETWORK  7  PARAMETER  3  0.5E-12 ;
NETWORK  8  PARAMETER  3  0.5E-12 ;
LABEL  DIODES REVERSE BIASSED WITH LINE LOSSES OF 0.1 DB/WL ;
COMPUTE;
END;

&END;
```

Fig. 7 - Data for Phase Shifter for MICRO2