Graph model analysis of computer structures

Ömür Taşar

GRAPH MODEL ANALYSIS OF COMPUTER STRUCTURES

BY

ÕMÜR TAŞAR, 1948-

A THESIS

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI-ROLLA

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

1974

Approved by

[illegible] (Advisor) [illegible]

R. M. Rakestraw

PUBLICATION THESIS OPTION

This thesis has been prepared in the style utilized by the IEEE Computer. Pages 1-28 are accepted for publication in that journal. The remainder of the paper has been added for purposes normal to thesis writing.

ABSTRACT

Graph theory is applicable to the solving of problems in nearly every field of scientific study. The purpose of this thesis is to consider its applications in representing and analyzing digital computers. Fundamental graph theory definitions, the types and the properties of the directed graphs, the matrix representation, and several reduction techniques are discussed. The blocking gate method for diagnosing computer systems is described and applied to the Scientific Control Corporation (SCC) 650 for its fault-diagnosis.

Microprogramming has been a significant trend in hardware and software designs of computers. Microprogrammed computers are discussed in comparison to conventional computers. A general scheme utilizing four nodes generates directed graphs for both types of architecture. The directed graphs are studied with respect to the flexibility and cost parameters.

ACKNOWLEDGEMENT

The author wishes to express her appreciation to Dr. Paul D. Stigall for his continued guidance and advise throughout the course of this thesis.

Appreciation is also extended to Dr. Javin M. Taylor, Dr. Roy M. Rakestraw for their review of this thesis, and Mrs. Diane Frederickson for her excellent typing.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (cont.)

LIST OF ILLUSTRATIONS (cont.)

LIST OF TABLES

I. INTRODUCTION

Computers are one of the most complex systems that man has made. The research that led to the evolution of today's large scale computers is extensive and complex enough to create a science: computer science. The complexity is not only in the design stages, but also in the use and understanding of the systems. The large scale use of such complex systems have led some researchers to investigate tools not commonly used. In recent years, applications of graph theory to computers as well as other fields of study have given fruitful results and have attracted more and more scientists. The attempt here will be to review previous accomplishments on a fundamental level and to apply graph theory concepts to computer related problems.

A great advantage for anyone who works on graphs is to be able to transfer his problem to a computer. The graphs are represented by matrices that can be easily handled in computer programs. Useful programs, which are applicable to many engineering problems, are documented by Henley and Williams [1].

The theory relevant to the study of graphs is rigorously developed [2-3]. Because its applications are many, it is worth mentioning a few interesting papers. A Fortran recognizer, which is itself a Fortran program, has been modeled by a graph by Gonzalez and Ramamoorthy [4]. This graph is reduced by the techniques described in the following chapter. The information obtained from the reduced graph is useful in determining the suitability of a program for parallel processing. Another paper presents a discussion of the techniques for optimal scheduling of tasks in a multiprocessor system [5]. Given a set of computational tasks and the relationship between them, a graph is formed. The algorithm developed finds a schedule for tasks for which the total execution time and the minimum number of processors required to realize this schedule are minimum. Bruno and Altman [6] have modeled the control structure of an asynchronous digital system. Basic control modules are formed to perform single control functions. The obtained graph is used to find the necessary and sufficient conditions for a class of well-formed, control networks.

The discussion of graphs in this paper will not span the classical use of graphs, such as state diagrams which model finite-state sequential machines [7]. However, minicomputer, conventional and microprogrammed computer structures will be represented by directed graphs, that serve the analysis of fault-diagnosis and structural behavior.

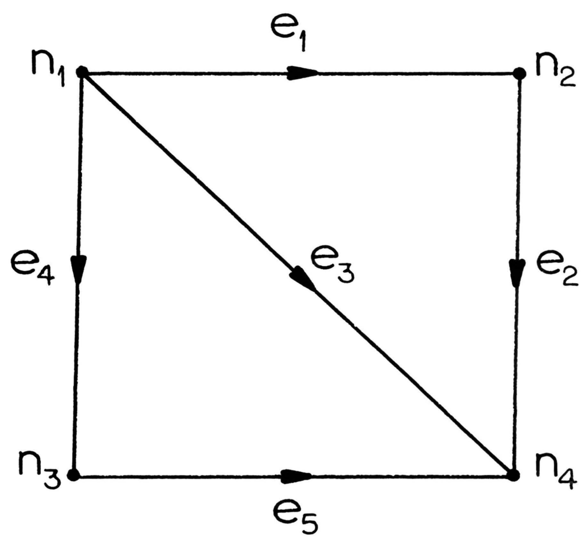
II. A REVIEW OF DIRECTED GRAPHS AS APPLIED TO COMPUTERS

A. Basic Definitions

A graph is simply a mathematical model of a system. It exhibits a relation or the absence of a relation among the elements of a set. The terms "point", "vertex", and "node" are frequently referred to as the elements of this set. A relation between these elements is usually called "line", "branch", "link", or "edge". In this paper, interest is concentrated on directed graphs, where the edges must be directed, and the terms "node" and "edge" will be used.

What is to be coordinated with nodes and edges is a matter for the problem in question. In the case presented here, a node may possibly represent a register, a flip flop, a gate, or a unit of the computer. An edge may represent a connection between two registers, or if it has a value associated, it may reflect a property of the system such as speed. Figure 1 shows a simple, directed graph.

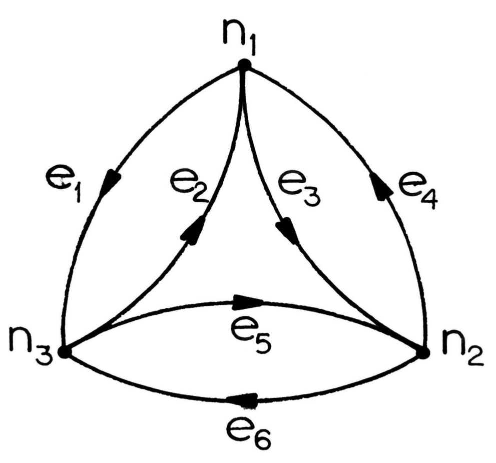
Graphs may have properties such as symmetry, reflexiveness, and completeness. A graph is _symmetric_ if every node satisfies the following condition: existence of an edge from node (a) to node (b) implies an edge directed from node (b) to node (a). A graph is _reflexive_ if every node has a loop on itself. A graph is _complete_ if every pair of nodes is connected in at least one direction. These properties are reflected in Figure 2.



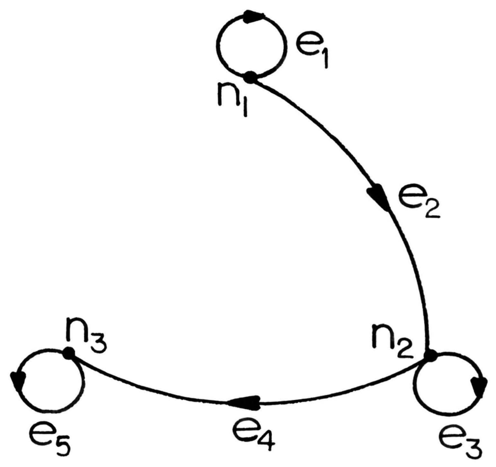
$$N = \{n_1, n_2, n_3, n_4\}$$

$$E = \{e_1, e_2, e_3, e_4, e_5\}$$

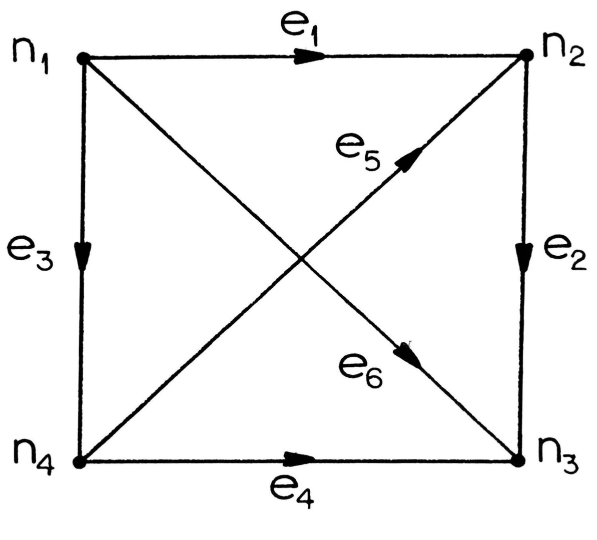
Figure 1. A Directed Graph.



a. Symmetric Graph



b. Reflexive Graph



c. Complete Graph

Figure 2. Properties of Directed Graphs.

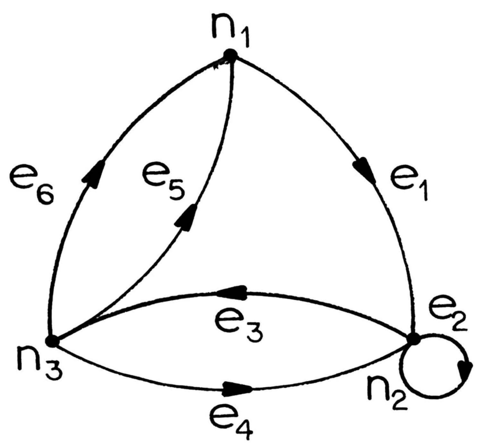
B. Types of Directed Graphs

The following set of directed graphs are not used throughout the rest of the paper but are included here for the sake of completeness.

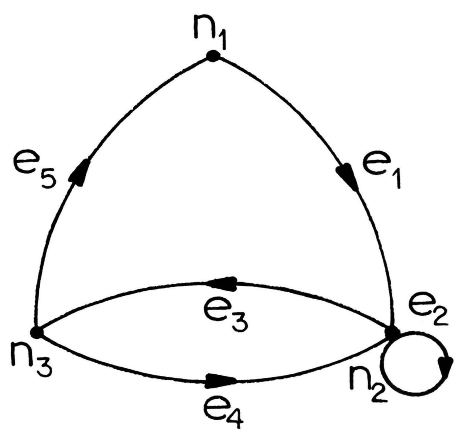
A net is a directed graph consisting of a set of nodes and edges. The set of nodes is finite and not empty. The set of edges is finite [3]. Hence a single node can constitute a trivial net. An example of a net is shown in Figure 3a. Petri nets, named after their inventor, C. A. Petri, are used to represent systems in which both static and dynamic conditions can exist simultaneously [8]. J. L. Bear introduced them as graph models for parallel computation in a survey in which multiprocessing was studied [9].

A relation is a directed graph which satisfies the above conditions but does not have any parallel edges. Two edges connecting the same two nodes are not considered parallel if they are oppositely directed. In Figure 3b, a relation obtained from Figure 3a is shown.

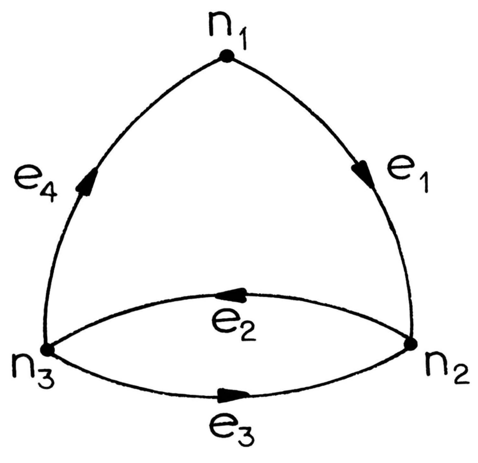
A digraph is an irreflexive relation [3]. Namely, it is a directed graph or a net having no parallel edges and loops. The theoretical studies and matrix representation of digraphs will be discussed in detail below. Figure 3c illustrates a digraph reduced from Figure 3b. Acyclic directed graphs form a special class of digraphs where no two nodes are mutually reachable [3,9].



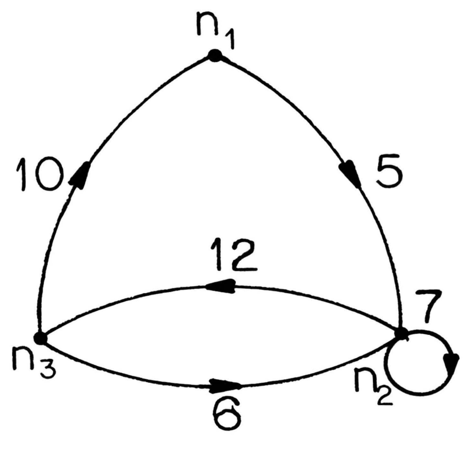
a. Net



b. Relation



c. Digraph



d. Network

Figure 3. Types of Directed Graphs.

A network is a relation in which the edges are assigned values [3]. If all the values on the edges are one, the graph is still a relation. In this sense, relations form a subset of networks. Systems dealing with frequencies, probabilities, and cost analysis can easily be represented by networks. A corresponding network of the relation in Figure 3b is shown in Figure 3d. In flow networks edges are interpreted to represent capacities of a flow, such as signals, cars, people, oil, and trade items. Then maximum flow considerations become important. Techniques and algorithms are developed to find a maximum flow in a network between any two nodes [2-3]. Network flows are formulated as linear programs. Maximum flow in relation to linear programming is discussed by T. C. Hu [10].

C. Matrix Representation

Matrix representation is a practical tool with which one can work on graphs. It allows algebraic manipulation and use of computer programming so that large dimensioned matrices, hence large graphs that have many nodes and edges, can be analyzed.

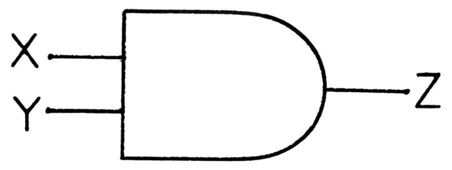
In forming the matrix, one row and one column for each node in the graph are assigned. Unless the assignment varies, a square matrix is generated. Depending on what is intended for the matrix, it is equally valuable to assign rows and columns to edges or rows to nodes and columns to edges. However, the case with nodes is discussed here, and it should be kept in mind that a similar approach may be taken for other cases.

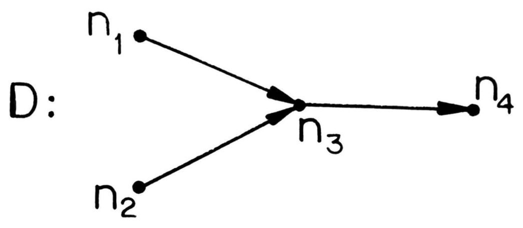
1. The Adjacency Matrix

The adjacency or connectivity matrix A has extensive use and is defined as follows: If the element a_{ij} of the matrix A is one, it indicates that there is an edge from node (i) to node (j). If a_{ij} equals zero, the graph does not contain an edge from node (i) to node (j) [3,11-12]. In other words, the nonzero elements of A show how many paths of length one, i.e., directed edges, exist between the corresponding nodes. Similarly, the elements of A^2, where A^2 is obtained by regular matrix multiplication of A by itself, indicate the number of possible paths of length two. The idea can be extended to find the number of possible paths of length n. These are illustrated in Figure 4. For example, there exists one possible path of length two from n_1 to n_4.

The row sum of node (i) of A is called the outdegree of node (i) and the column sum of node (i) of A is called the indegree. These figures give the total number of edges going out of and into the node, respectively.

The adjacency matrix has been effectively applied to computer areas. Ramamoorthy and Chang [11] have based an algorithm on the adjacency matrix to segment a large system into smaller subsystems with the purpose of diagnosing the system in parallel. Russel and Kime [12] used the adjacency





a. Logic Diagram for Z = XY.

b. Its Directed Graph

$$A = \begin{array}{c|cccc} & n_1 & n_2 & n_3 & n_4 \\ \hline n_1 & 0 & 0 & 1 & 0 \\ n_2 & 0 & 0 & 1 & 0 \\ n_3 & 0 & 0 & 0 & 1 \\ n_4 & 0 & 0 & 0 & 0 \end{array}$$

c. Its Adjacency Matrix A

$$A^2 = \begin{array}{c|cccc} & n_1 & n_2 & n_3 & n_4 \\ \hline n_1 & 0 & 0 & 0 & 1 \\ n_2 & 0 & 0 & 0 & 1 \\ n_3 & 0 & 0 & 0 & 0 \\ n_4 & 0 & 0 & 0 & 0 \end{array}$$

d. Square of the Adjacency Matrix

Figure 4. Adjacency Information of D Representing a Logic Diagram.

matrix as well as indegree and outdegree concepts in fault diagnosis of combinational networks. Kleir and Ramamoorthy [13] have used the adjacency matrix in optimization strategies for microprograms. It has also been used in the structural theory aspects of machine diagnosis [14].

2. The Reachability Matrix

The _reachability_ matrix, R, gives useful information about the behavior of a graph. The element r_{ij} equals one if node (i) reaches node (j) over any path regardless of its length and if i equals j. If r_{ij} equals zero, it indicates that there is no possible path whereby node (j) can be reached from node (i) [3,11-12]. The transpose of R, namely R^T, represents a graph in which the directions are reversed with respect to the original graph.

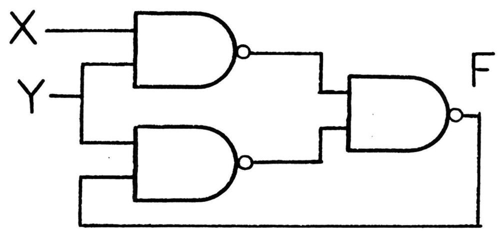
The elementwise product, $Q = R \times R^T$, is obviously a symmetric matrix. The nonzero elements, q_{ij}, of this matrix indicate that nodes (i) and (j) are mutually reachable. This information is useful for the purpose of reducing graphs if necessary. One reduction technique requires the selection of the nodes whose columns are equal. These are the nodes which are reachable from the same set of nodes and which reach the same set of nodes. Hence, they can be combined into one node. The set of nodes combined into one new node is called a _strong component_. If all columns of Q happen to be equal, the graph is said to be strongly connected, in which case every node is reachable from every other node. An example of the points illustrated above is given in Figure 5.

Hence, the graph has four strong components, $\{n_1\}$, $\{n_2\}$, $\{n_3\}$, and $\{n_4, n_5\}$, which are called N_1, N_2, N_3, and N_4, respectively. To find the reduced graph with nodes N_1, N_2, N_3, and N_4, the new edges must be found. The rule is as follows: There exists an edge from N_i to N_j if there is a path from any node in N_i to any node in N_j. The reduced graph is called a condensation of D and is shown in Figure 6.

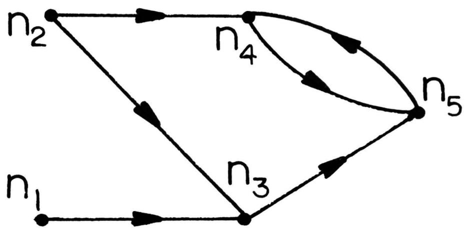
The reachability matrix has been applied to computer related problems [12,14-15]. It provides a powerful reduction technique. Most of the algorithms in fault diagnosis are derived for reduced graphs [16].

3. The Connectedness Matrix

A valuable measure for classifying graphs is connectedness. Earlier, a strongly connected graph and its strong components were defined. To explore the other possibilities, the following types of graphs can be briefly described with respect to connectedness. A disconnected graph implies that there exists at least one node or a set of strongly connected nodes that neither reaches nor is reached from any other node. In a strictly weak graph, there exists a sequence of edges between any two nodes; however, the directions are not



a. Logic Diagram for F = XY + Yf



b. Its Directed Graph

$$R = \begin{array}{c|ccccc} & n_1 & n_2 & n_3 & n_4 & n_5 \\ \hline n_1 & 1 & 0 & 1 & 1 & 1 \\ n_2 & 0 & 1 & 1 & 1 & 1 \\ n_3 & 0 & 0 & 1 & 1 & 1 \\ n_4 & 0 & 0 & 0 & 1 & 1 \\ n_5 & 0 & 0 & 0 & 1 & 1 \end{array}$$

c. Its Reachability Matrix R

$$Q = \begin{array}{c|ccccc} & n_1 & n_2 & n_3 & n_4 & n_5 \\ \hline n_1 & 1 & 0 & 0 & 0 & 0 \\ n_2 & 0 & 1 & 0 & 0 & 0 \\ n_3 & 0 & 0 & 1 & 0 & 0 \\ n_4 & 0 & 0 & 0 & 1 & 1 \\ n_5 & 0 & 0 & 0 & 1 & 1 \end{array}$$

d. $Q = R \times R^T$

Figure 5. Reachability Information of D Representing a Logic Diagram.

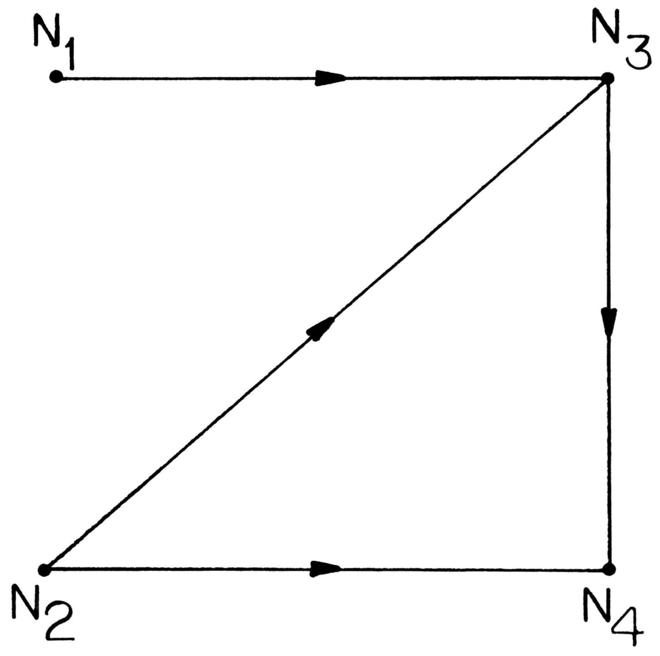


Figure 6. The Reduced Graph of Figure 5b.

continuous. A <u>strictly strong</u> graph has a directed path between any two nodes. The example in Figure 7 clarifies these descriptions.

It is an easy matter to deduce the idea of connectedness from the reachability information. Hence, a connectedness matrix, C, can be defined, which will indicate the above attributes, given two nodes. Let us say that 0, 1, 2, and 3 represent the four kinds of connectedness: disconnected, strictly weak, strictly strong, and strongly connected, respectively. A new matrix, J, is defined as follows:

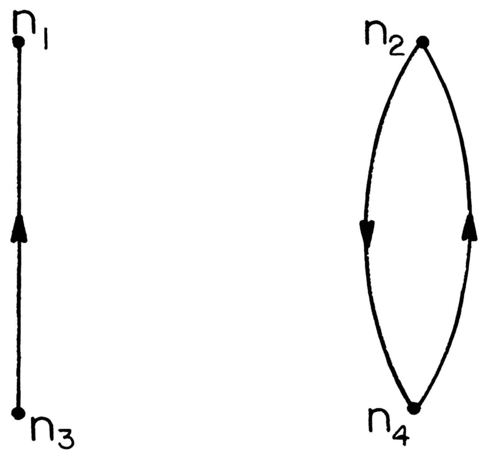
$$j_{kl} = \begin{cases} 0 \text{ if node (k) and node (l) are disconnected} \\ 1 \text{ otherwise} \end{cases}$$

Thus, the matrix representation for connectedness is simply $C = R + R^T + J$. For instance, in Figure 7a, because j_{14}, r_{14}, and r_{41} are zero, c_{14} equals zero implying a disconnected graph.

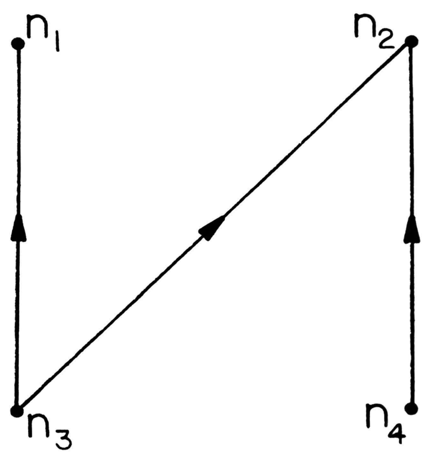
In case the graph is composed of disconnected subgraphs, D_1, D_2, ... D_n, the connectedness matrix for each subgraph can be found, $C(D_1)$, $C(D_2)$, ... $C(D_n)$. The connectedness matrix of the whole graph will have the forms shown in Figure 8.

4. The Value Matrix

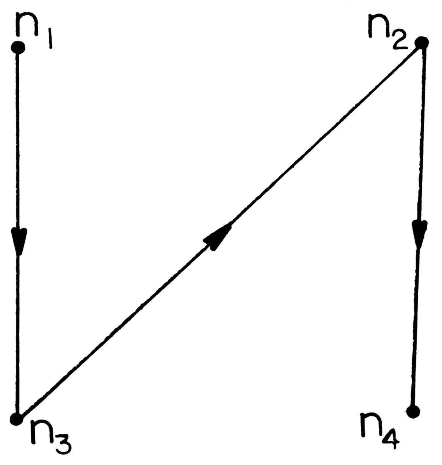
For a network, the adjacency matrix has scalar entries



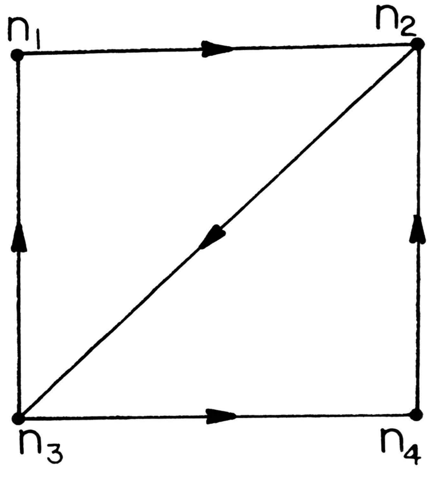
a. Disconnected Graph



b. Strictly Weak Graph



c. Strictly Strong Graph



d. Strongly Connected Graph

Figure 7. Types of Graphs with Respect to Connectedness.

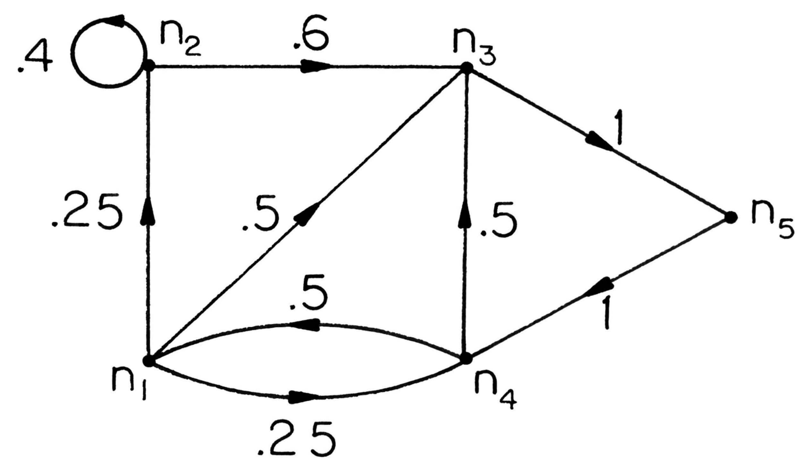
$$C = \begin{bmatrix} C(D_1) & 0 & 0 & \cdot & \cdot & 0 \\ 0 & C(D_2) & 0 & \cdot & \cdot & 0 \\ 0 & 0 & C(D_3) & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdot & C(D_n) \end{bmatrix}$$

Figure 8. Connectedness Matrix of a Disconnected Graph.

rather than ones and zeros. To distinguish the two matrices, the adjacency matrix for a network is called the value matrix, M. If the values on the edges are associated with probability or cost, the matrix is called a probability matrix, P, or a cost matrix, G, respectively. In cases where the outdegree of each node on a probability network is one, and the probabilities assigned to the edges are time-dependent, this particular type of graph is occasionally called a markov chain. M. A. Breuer modeled the statistics of intermittent faults in digital circuit by a first order Markov model [17]. The cost of going from node (i) to node (i+1) is infinite, if there is no edge from node (i) to node (i+1). Examples are shown in Figures 9 and 10.

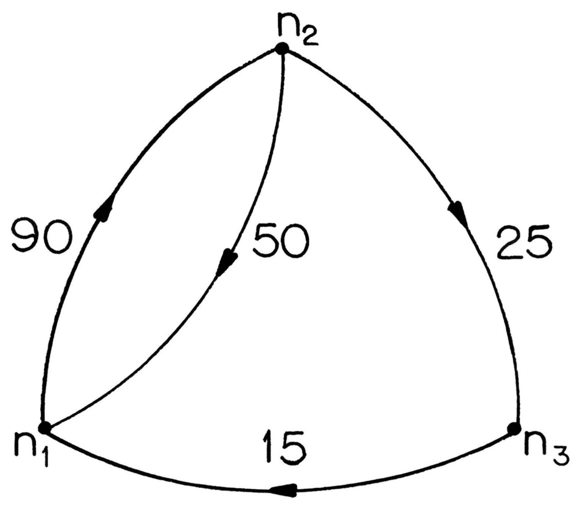
5. Description of the Basic Theorems

At this point, it is interesting to investigate whether the matrices discussed relate to each other in any manner. The matrix R is expected to be a function of the powers of A, because the elements of A^n indicate the possible number of paths of length n, and the reachability question asks whether any path exists between two given nodes. The procedure then must be to take powers of A until the longest path has been searched and to transform the total number of possibilities to a "one" to indicate reachability. The latter function is labeled U and defined as follows: $U(a) = 1$, where a is any



$$
P = \begin{array}{c|ccccc|}
 & n_1 & n_2 & n_3 & n_4 & n_5 \\
n_1 & 0 & .25 & .5 & .25 & 0 \\
n_2 & 0 & .4 & .6 & 0 & 0 \\
n_3 & 0 & 0 & 0 & 0 & 1 \\
n_4 & .5 & 0 & .5 & 0 & 0 \\
n_5 & 0 & 0 & 0 & 1 & 0
\end{array}
$$

Figure 9. A Probability Network and its Probability Matrix.



$$G = \begin{array}{c|ccc|} & n_1 & n_2 & n_3 \\ n_1 & 0 & 90 & \infty \\ n_2 & 50 & 0 & 25 \\ n_3 & 15 & \infty & 0 \end{array}$$

Figure 10. A Cost Network and its Cost Matrix.

number other than zero, and U(0) = 0.

After this stage, the reachability matrix will not change. The above descriptions can be summarized with a theorem:

Theorem 1: $R_n = U(I + A + A^2 + \ldots + A^n)$ in which I is the identity matrix and defines reachability over length n. If $R_n = R_{n+1}$, then $R = R_n$ [3].

The cost of going from one node to another over a specified length is of great importance. One certainly would try to find the minimum cost path, which is called cost geosdesic. When taking powers of G, the matrix multiplication is performed by using the modified multiplication "x" and the modified addition "+" operations defined as follows: a x b = a + b and c + d = min(c, d) [3].

Obviously, the arithmetic does not add all possible paths of a certain length but rather finds the cost of different paths of the same length and chooses the minimum value. Let us find g^2_{11} of G^2, i.e., the cost of going from node (1) to itself over a path of length two by referring to Figure 10.

$$\begin{aligned} g^2_{11} &= g_{11} \times g_{11} + g_{12} \times g_{21} + g_{13} \times g_{31} \\ &= \min(g_{11} + g_{11}, g_{12} + g_{21}, g_{13} + g_{31}) \\ &= \min(0, 140, \infty) \\ &= 0. \end{aligned}$$

Staying at node (1) is assumed to have no cost, hence that path is chosen.

Theorem 2: There is a positive integer n for which $G^n = G^{n+1}$ [3].

After reaching this condition, there is no need to take higher powers of G, and G^n is called the total cost matrix.

In general, if the edges of the network indicate only zeros and ones, a distance matrix is formed, and the modified arithmetic yields the minimum distance.

D. Fault Diagnosis Using Graph Theory

A fault is a malfunction in the system. It is of vital importance whether a fault exists or not. Testing the machine for a fault if it exists is called a diagnosis. Fault diagnosis in digital computers is one of the major areas where efficient methods have to be established. In recent years, applications of graph theory to this area have been promising. An approach called a test point method finds spots on the system where test points can be located [14]. Some points serve as inputs, and the results can be detected at other points. The blocking gate method is another approach in which the minimum number of edges to be blocked are found to diagnose the system [16]. The latter method is discussed in detail below.

The blocking gate method assumes that there exists only one fault at a time and that it is not self-correcting. Also no faults can cancel each other during diagnosis. One way to diagnose the system is to insert blocking gates on every edge in the graph. The idea can be extended to conduct the diagnosis more efficiently.

First, the system is modeled with a directed graph. Then the graph is reduced so that it does not contain any strongly connected components. Finally, the graph is transformed into a single input single output graph (SIOG) to enable the method to work with one input and one output. This is easily accomplished. If the graph has more than one input, an input node (i) is entered to the graph that fans out to all the necessary inputs. If there is more than one output, they will lead to an extra output node (o). One can start with the example of the SIOG shown in Figure 11 to illustrate the steps in the blocking gate method.

At this point, a reduced SIOG exists and it is ready for the introduction of the pattern for finding the locations of the minimum number of blocking gates. The range of node (k) is the set of nodes on the directed path from node (i) to node (o), when node (k) is deleted. The node range matrix, NR, of a SIOG is a square matrix with rows and columns corresponding to the nodes of the graph. The NR matrix of the graph SIOG1 is given in Figure 12. The k'th row of the matrix has elements of ones for the nodes in the range of node (k). Otherwise, the elements are zero. The set of nodes, whose columns are equal, constitutes the partition of maximum distinguishability, MD. Referring to Figure 12, the MD

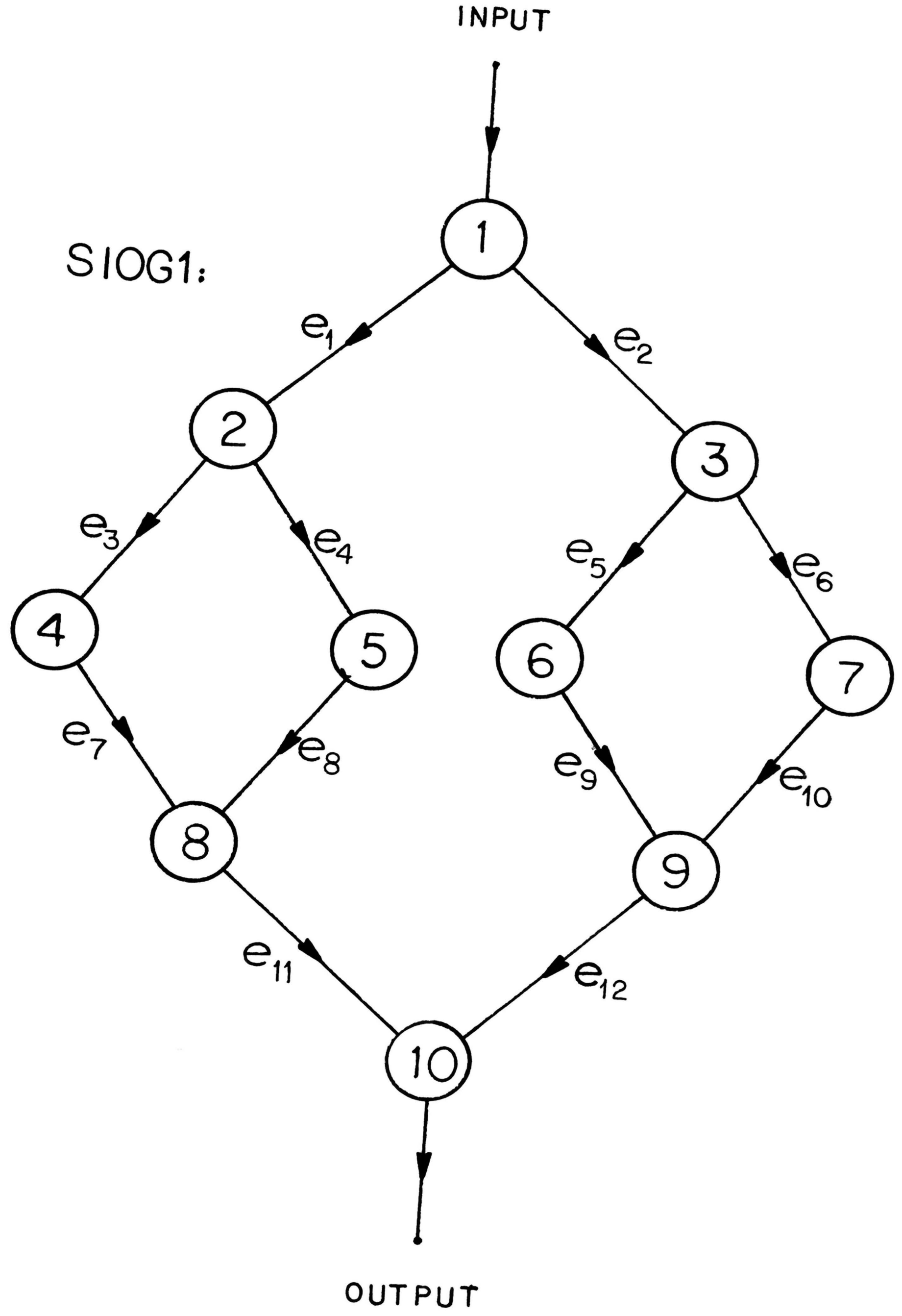


Figure 11. A Single Input Single Output Graph SIOG1.

$$
NR = \begin{array}{c|cccccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
3 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
4 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
5 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
6 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
7 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
8 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
9 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

Figure 12. The Node Range Matrix of the Graph SIOG1.

set for the graph SIOG1 is found as follows:

$$MD = \{\overline{1,10};\ \overline{2,8};\ \overline{3,9};\ \overline{4};\ \overline{5};\ \overline{6};\ \overline{7}\}.$$

Every edge of the graph is to be tested so that the set of minimum number of edges is found. The blocking gates have to be located on this set of edges to distinguish the particular distinguishability class. Another matrix is used to perform what is mentioned above. The matrix, ER, has columns corresponding to the partitions of maximum distinguishability; its rows correspond to the edges of the graph. If the edges are ordered with consideration for the cost of blocking gates to be located on them, then the resultant sets of edges obtained from the matrix, ER, can be compared and the set with the minimum cost can be chosen. Figure 13 illustrates the ER matrix of the graph SIOG1. It is assumed that the cost of building blocking gates on the edges increases as the output node is approached.

The edge range of an edge (k) is defined as the set of nodes on a directed path from node (i) to node (o), when edge (k) is blocked. The element er_{kl} on the k'th row of the matrix ER would be one if the partition l includes the nodes within the range of edge (k). Otherwise, the elements are zero. After constructing the matrix, equal rows are found. These rows form the sets of a new partition, named E. The equal rows of the ER matrix form the following set:

$$ER = \begin{array}{c|ccccccc|} & \overline{1\text{-}10} & \overline{2\text{-}8} & \overline{3\text{-}9} & \overline{4} & \overline{5} & \overline{6} & \overline{7} \\ e_1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ e_2 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ e_3 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ e_4 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ e_5 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ e_6 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ e_7 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ e_8 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ e_9 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ e_{10} & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ e_{11} & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ e_{12} & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{array}$$

Figure 13. The Edge Range Matrix of the Graph SIOG1.

$$E = \{\overline{1,11};\ \overline{2,12};\ \overline{3,7};\ \overline{4,8};\ \overline{5,9};\ \overline{6,10}\}.$$

The elements of E are to distinguish the partition of MD. Ramamoorthy and Mayeda [16] discussed the method theoretically and proved that the set E distinguishes the set MD.

The matrix ER is reduced to another matrix EQ when the equal rows are deleted. The small numbered edges name the rows of EQ, because they imply less cost. To complete the example if blocking gates are placed on edges 1, 2, 3, 4, 5, 6 during design stages, the system will be able to diagnose faults within nodes 4, 5, 6, 7 and sets of nodes {1,10}, {2,8}, {3,9}.

III. GRAPH MODEL FAULT DIAGNOSIS OF A MINICOMPUTER: SCC 650

A. Description of SCC 650 [18,19]

The Scientific Control Corporation (SCC) 650 is a high speed, general purpose, digital minicomputer. It has a 4K, random access, core memory. Its memory word length is 12 bits, and its memory cycle time is two microseconds. The computer contains a single interrupt channel and input-output equipment (teletype and paper tape device). Some machines have multiply and divide hardware, a digital-to-analog converter, an analog-to-digital converter, and multiplexer.

The SCC 650 is a synchronous machine. The memory cycle is divided into eight intervals. A three-bit counter controls the timing together with a clock. Because each interval provides ample time for most of the transfers and operations within the computer, as many of these as possible are performed in parallel.

The 12-bit, parallel, binary adder is a combinational circuit. It is capable of performing "addition", "subtraction", "and", and "exclusive or" operations. The main registers communicating with the memory are 12-bit registers.

Each instruction has three sequences: fetch, effective address calculation, and execution. The overlapping of these sequences is permissible. Figure 14 shows the data paths for the SCC 650.

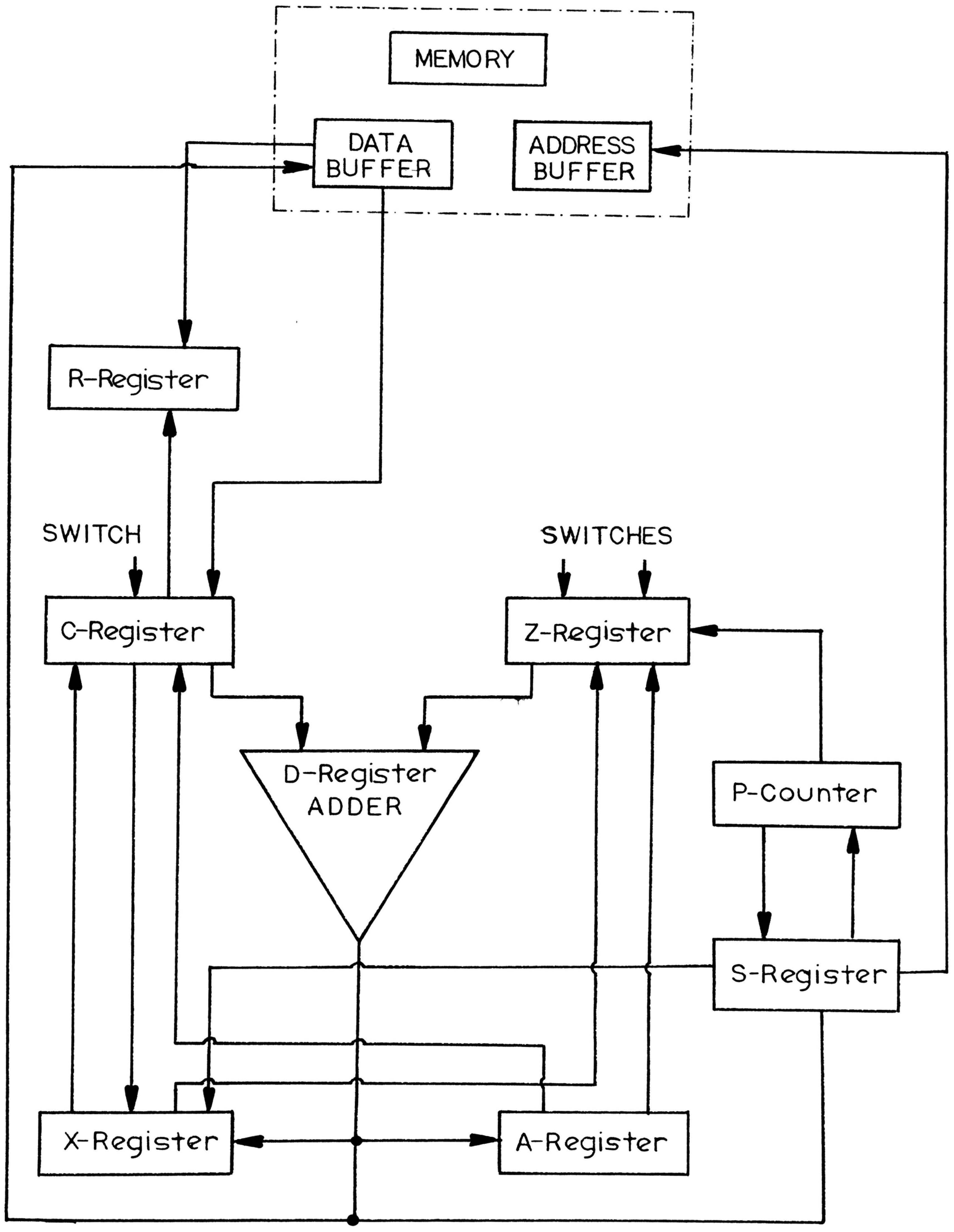


Figure 14. Data Paths for SCC 650.

B. Graph Model of SCC 650

There is a useful relation between discrete sequential systems and directed graphs. By relying on Ramamoorthy's work, any discrete sequential system can be shown to be isomorphic to a directed graph [14,16].

The SCC 650 is an appropriate minicomputer to be represented graphically. On the data and instruction level, information flows through the memory and the main registers of the computer so that the graph representation has a meaning. The necessary transfers for the instructions can be followed as a directed path on the graph. The graph model presented and analyzed in this paper is an attempt to search for information that can improve the computer. Although the conclusions are not exciting, it is felt that the SCC 650 provides an easily understandable example for a starting point. The graph model of SCC 650 is given in Figure 15. The circled letters indicate the nodes of the graph. They correspond to the memory and essential registers of Figure 14.

The adjacency and reachability matrices and the Q matrix of the graph G are given in Figure 16. It should be noted that all nodes except node R are strongly connected. This is to be expected, because node D, the adder, feeds the information back to the computer. The technique of reducing a group of strongly connected nodes into one node is not useful in this case, because the condensation will have only two nodes, R and the set of all

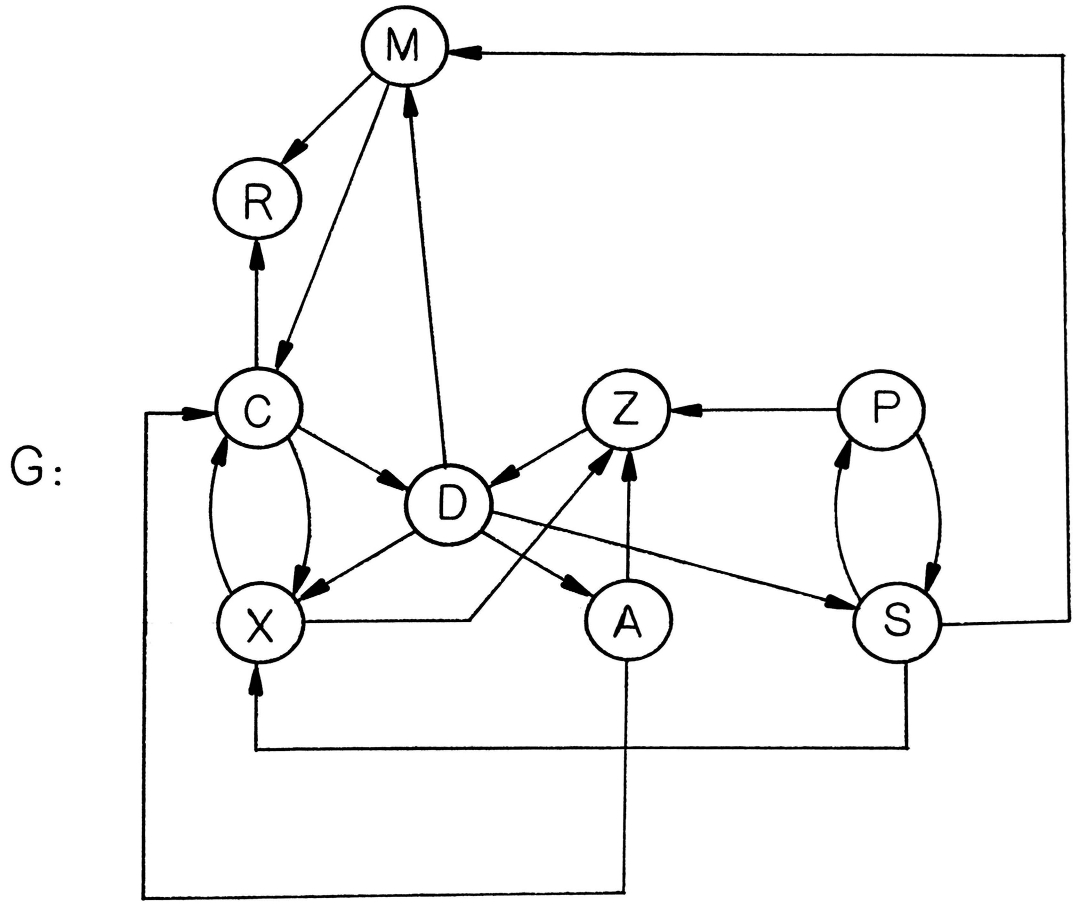


Figure 15. Graph Model of SCC 650.

$$A(G) = \begin{array}{c|ccccccccc} & A & C & D & M & P & R & S & X & Z \\ \hline A & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ C & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ D & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ M & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ P & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ R & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ S & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ X & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ Z & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

a. The Adjacency Matrix for G

Figure 16. The Matrices of the Graph G.

$$R(G) = \begin{array}{c|ccccccccc|} & A & C & D & M & P & R & S & X & Z \\ \hline A & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ C & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ D & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ M & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ P & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ R & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ S & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ X & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ Z & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}$$

b. The Reachability Matrix for G

Figure 16. The Matrices of the Graph G (cont.).

$$Q(G) = \begin{array}{c|ccccccccc}
 & A & C & D & M & P & R & S & X & Z \\
\hline
A & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
C & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
D & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
M & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
P & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
R & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
S & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
X & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
Z & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1
\end{array}$$

c. $Q = R \times R^T$ for G

Figure 16. The Matrices of the Graph G (cont.).

the other nodes.

In general, the graphs of computers face the problem of being strongly connected, because the information has to loop around the nodes. To analyze these graphs, other techniques have to be tried for reduction. One approach is to break a certain edge or edges. Some edges can be deleted from the graph [15], or nodes can be removed. One can also add new nodes to make the information on the graph more precise. The effect of removing edges and nodes from the graph is not easily determined, therefore, the reason to use this technique depends on the specifications of the computer rather than on a predetermined conclusion. Otherwise, it can be determined by observation.

C. Fault Diagnosis of SCC 650

The diagnosis of the SCC 650 can be performed with the blocking gate method [16]. To apply this method, the graph given in Figure 15 has to be modified to a reduced, single input, single output graph.

To reduce the graph without losing too much information, one has to break the outputs of some node. Practically, this can be done by blocking the signals with the logical gates. A program [20] has been written that finds the connectedness matrix of each graph, in which the outputs of one node are deleted in succession. The result of this program has been evaluated as follows: The desired graph should have a minimum number of strongly connected components, and these can be combined into a new node. Accordingly, the outputs of node D are blocked, and the strong components, C and X and P and S, are combined into two new nodes, CX and PS, respectively.

The minicomputer has on its front panel four switch registers consisting of 12 switches. These can input to the memory address register, C, memory buffer register, S, location counter, P, and accumulator, A. To form an SIOG, an input node has to be created which leads to the node PS and an output node connected to node D. Because the outputs of node D are blocked, it is convenient to monitor this point.

Another modification has to be made on the graph to distinguish several outputs of a certain node. For instance, the new node "m" separates the outputs of memory to nodes R and CX; nodes "c" and "s" function similarly. The modified graph, on which the blocking gate method can be performed, is given in Figure 17.

The artificial input and output nodes are not considered in the node set of the graph, they, therefore, need not be diagnosed.

The node range matrix is given in Figure 18. It can be observed that nodes M and m, R and A, CX and s, and D and PS have equal columns, correspondingly, so that they form the partitions of D in addition to c and Z. However, by applying the method further, the elements of MD will be fault diagnosed.

The partitions of maximum distinguishability are found as follows: MD = {M-m; R-A; CX-s; c; D-PS; Z} (Figure 19).

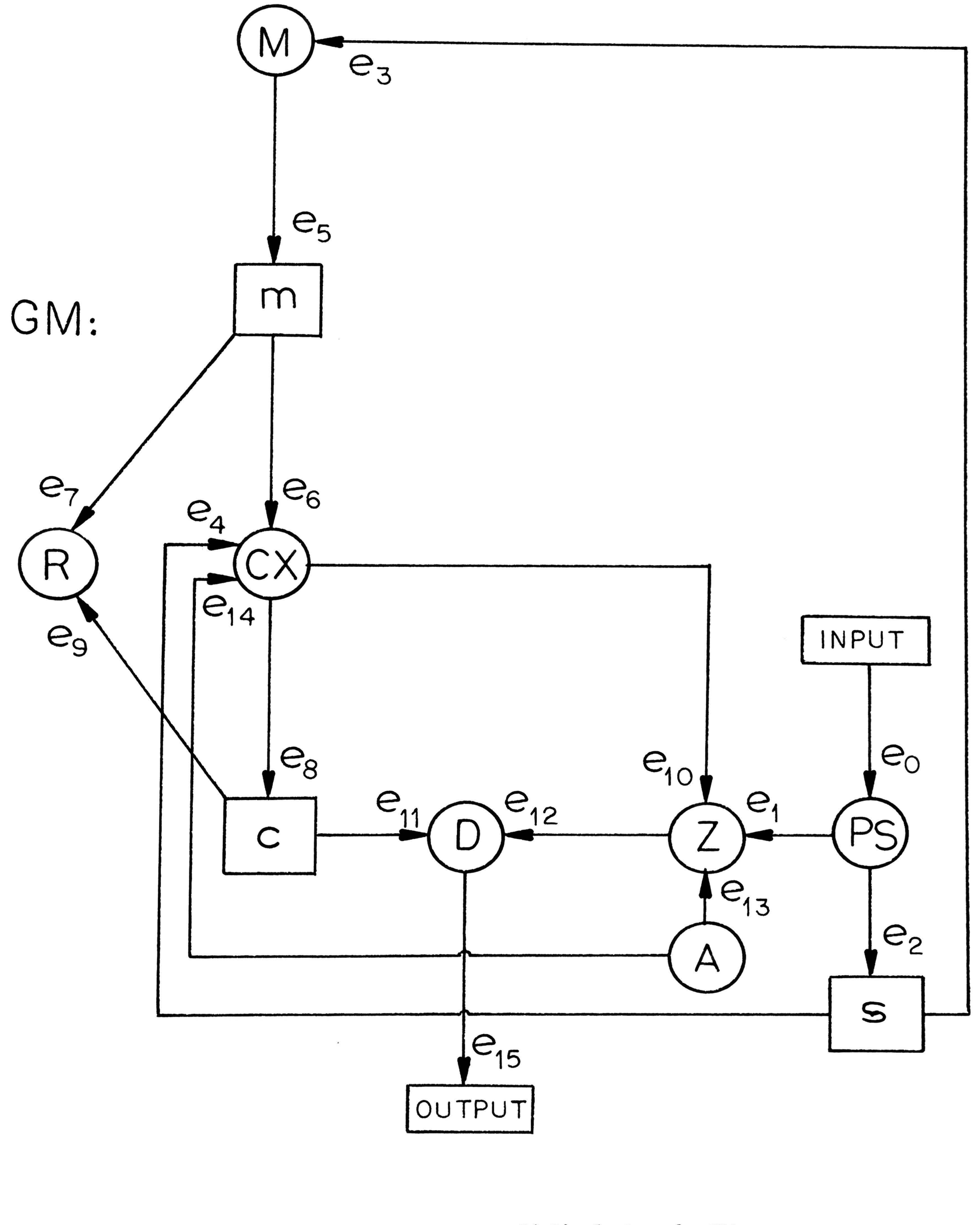


Figure 17. The Modified Graph GM.

$$NR = \begin{array}{c|cccccccccc|} & A & c & CX & D & M & m & PS & R & s & Z \\ A & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ c & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ CX & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ D & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ M & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ m & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ PS & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ R & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ s & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ Z & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \end{array}$$

Figure 18. The Node Range Matrix of the Graph GM.

$$ER = \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \\ e_8 \\ e_9 \\ e_{10} \\ e_{11} \\ e_{12} \\ e_{13} \\ e_{14} \end{array} \begin{array}{c} \begin{array}{cccccc} \overline{M\text{-}m} & \overline{R\text{-}A} & \overline{CX\text{-}s} & \overline{c} & \overline{D\text{-}PS} & \overline{Z} \end{array} \\ \left[\begin{array}{cccccc} 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \end{array}\right] \end{array}$$

Figure 19. The Edge Range Matrix of the Graph GM.

The equal rows of the matrix ER give the partitions of edges as follows: $E = \{e_1e_{12}; e_2; e_3e_5e_6; e_4e_7e_9e_{10}e_{13}e_{14}; e_8e_{11}\}$. The reduced edge range matrix is shown in Figure 20.

As the final step, the set $\{e_1, e_2, e_3, e_4, e_8\}$ is applied to locate the fault, if it exists. This procedure is shown in Figure 21. The end leaves of the tree structured graph are the partitions of MD.

If the fault is found in one of the strong components, the diagnosis is still not complete. Therefore, the procedure of blocking gate method is applied to a reduced graph in which the outputs of another node other than node D are blocked. After trying several nodes for this purpose, node S is found to be appropriate. All the necessary steps are given in Figures 22, 23. The MD of the graph GR is {R; T; P-S}, and E of the graph GR is $\{e_1e_{17}; e_5e_{16}\}$.

This approach enables the procedure to distinguish nodes R and A and D and PS, which were in the same partitions in the previous application of the blocking gate approach. A fortunate result is that the second application does not increase the number of blocking gates.

To conclude the diagnosis of SCC 650, it has to be mentioned that nodes C and X and nodes P and S are not distinguishable under the method applied here. The test point method can be applied in addition to the present method to overcome the problem [14].

$$EQ = \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_8 \end{array} \begin{array}{c} \begin{array}{cccccc} \overline{\text{M-m}} & \overline{\text{R-A}} & \overline{\text{CX-s}} & \overline{c} & \overline{\text{D-PS}} & \overline{Z} \end{array} \\ \left[\begin{array}{cccccc} 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{array} \right] \end{array}$$

Figure 20. The Reduced Edge Range Matrix of the Graph GM.

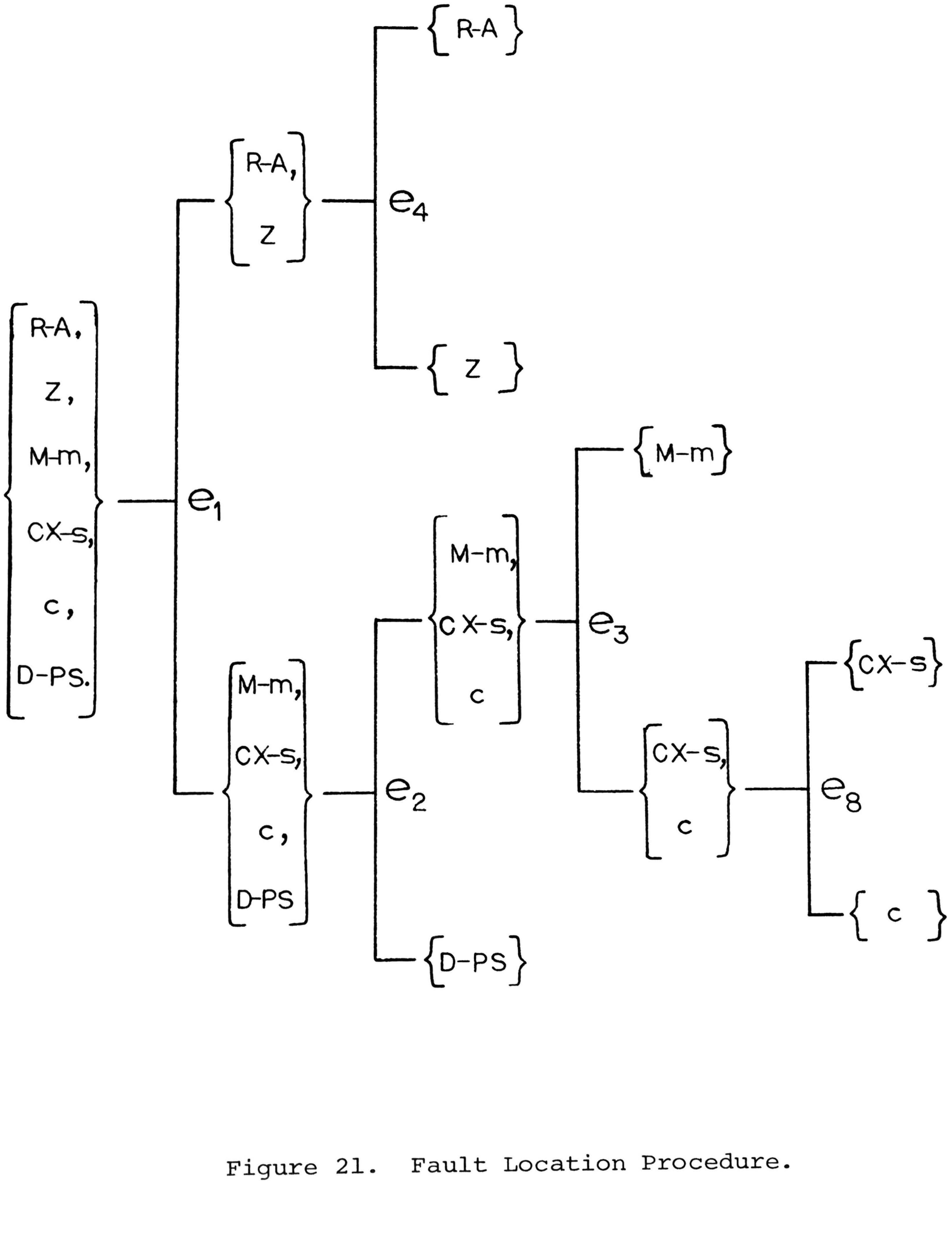
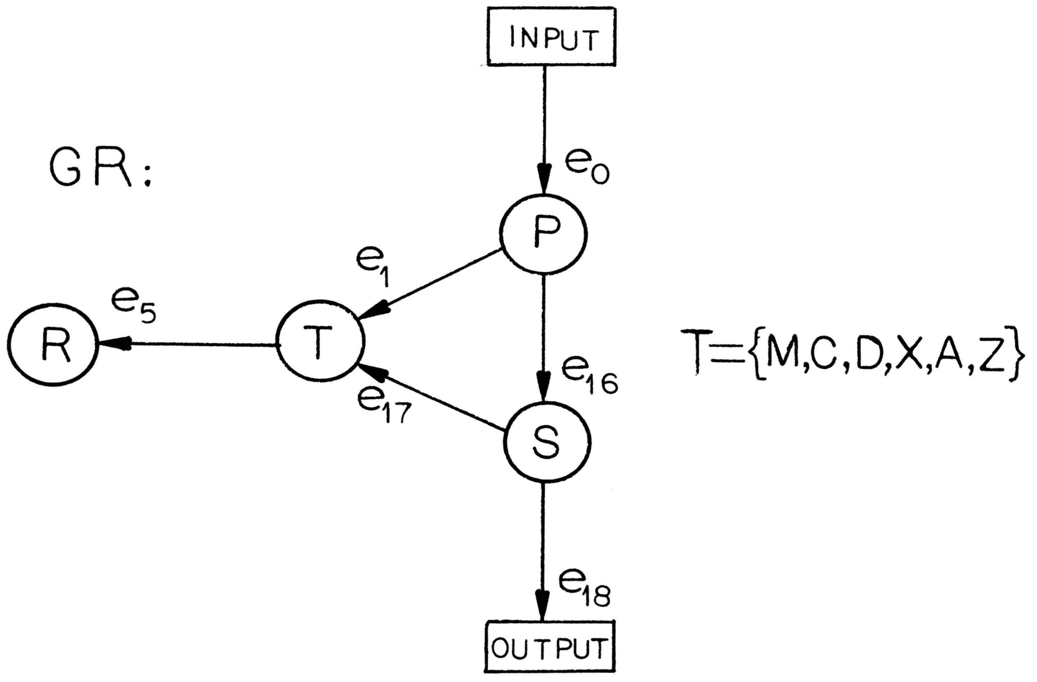


Figure 21. Fault Location Procedure.



a. The Reduced Graph GR, when Outputs of Node S are Blocked

$$NR = \begin{array}{c|cccc} & R & T & P & S \\ \hline R & 0 & 1 & 1 & 1 \\ T & 0 & 0 & 1 & 1 \\ P & 0 & 0 & 0 & 0 \\ S & 0 & 0 & 0 & 0 \end{array}$$

b. The Node Range Matrix of the Graph GR.

Figure 22. Steps in Finding Partitions of MD.

$$ER = \begin{array}{c|ccc} & R & T & P\text{-}S \\ \hline e_1 & 0 & 0 & 1 \\ e_5 & 0 & 1 & 1 \\ e_{16} & 0 & 1 & 1 \\ e_{17} & 0 & 0 & 1 \end{array}$$

a. The Edge Range Matrix of the Graph GR

$$EQ = \begin{array}{c|ccc} & R & T & P\text{-}S \\ \hline e_1 & 0 & 0 & 1 \\ e_5 & 0 & 1 & 1 \end{array}$$

b. The Reduced Edge Range Matrix of the Graph GR

Figure 23. Steps in Finding the Edges with Blocking Gates.

IV. ANALYTICAL STUDY OF CONVENTIONAL AND MICROPROGRAMMED COMPUTERS

A. Microprogramming

1. Description

Microprogramming was proposed by Dr. M. V. Wilkes in the early 1950's. The technology of the period did not suffice for an economically feasible development in practice; the main drawback arising from the lack of the fast storage elements. An informal description of microprogramming will clarify the reason [21-22].

A conventional computer can be organized by the basic components: I/O, storage, arithmetic, logic and control units. The control unit is a hardwired section to perform all the necessary gating operations of the entire computer. Its design philosophy has relied upon the available technology through the generations of computers. In 1960's the developments in nondestructive storage technology resulted in the production of the high speed read-only memories. Hence the replacement of the transistor logic control section with a stored logic or a microprogram control proved to be advantageous [23]. A stored program to control and select operations for the execution of several instructions is extensively utilized in computers such as some models of IBM 360/370 series, Burroughs B2500-B3500, B1700, Standard Computer 670-2700, the Honeywell H4200/H8200 and several minicomputers such as Hewlett Packard 2100S.

If the microprogram is stored in a read-write memory such that the user could access and modify the microinstructions to the advantage of his particular need, the machine is called _dynamically microprogrammed_. IBM 360/25 is an example of using read-write memory to store microprograms but does not utilize the ability to change the microinstruction to any extent [24].

2. Some Aspects of Implementation

Considering the amount of control within a computer one suspects that the microinstruction requires a large number of bits. Hence the type of the control memory used in the system affects the design of a microprogram. In case the number of bits/word is preferred to be small, schemes may be used to encode mutually exclusive control signals reducing the amount of necessary storage and the length of microinstructions. However, the additional circuitry for the decoders and encoders slows down the execution of the microinstructions.

The length of the microinstruction depends also on _polyphase_ and _monophase_ programming [22]. In the former the microinstruction initiates more than one microoperation, where monophase corresponds to the case of one microoperation being initiated. Polyphase microprogramming is more complex, but the execution time is reduced.

Horizontal programming and vertical programming are alternative methods in implementing microprograms. In horizontal programming, long words are used to set up controls for a number of register transfers. In vertical programming, words may be short, but the microinstructions are applied in a certain sequence.

Upon the choices made the microprogrammed computer characteristics will change, although all the different implementations may control the same instruction sequence. For example IBM 360 series have several microprogrammed computers over a wide range of performance with the number of bits/word in the control memory varying from 60 to 100 [24].

3. Advantages and Disadvantages of Microprogramming

Flexibility is an important advantage during design stages. The design of the microprogram and the implementation of the system is done in parallel. The system builders have the chance to modify the control section, if later in the design procedure they observe that the addition of some instructions would improve the performance. The read-only storage (ROS) containing the microprogram may even be replaced with another ROS programmed more efficiently. Taking economic aspects into account this may not be feasible, however, compared to a hardwired system the possibility of enhancing the system performance still exists.

If the computer is dynamically microprogrammed there is more flexibility. The microprogram on the random access memory (RAM) can be modified with no hardware cost.

Simplicity is another factor in favor of microprogramming. Conventional control design is complex and difficult to implement, maintain and understand. Any design mistake is undesirable, since it is costly to change the hardware. Fault detection and diagnosis of the hardwired system is very difficult. Additional circuitry to ease the problem is expensive. Microprogram on the other hand is easily fault detected and diagnosed [25].

It has been stated occasionally that a microprogrammed general purpose computer is less expensive to implement compared to a hardwired one, because the amount of logic required is less. However, the cost comparison over a period of time may prove the reverse. One should consider the performance factor while discussing economy. Microprogrammed computers are slow because they involve storage. Presently memories are the slowest components in computers excluding I/O devices. Also ROS control is sequential and no overlaps are allowed in processor and memory operations.

IBM 360/370 machines emulating earlier IBM machines (1400 and 7000 series) are economical and they improve the performance; microprograms are written for second generation computers. Hence speed and lower cost are gained over the emulated machine. Also because of improved hardware technology it is cheaper for the customer to increase the capability of his system rather than buying a new machine.

4. Applications of Microprogramming

In the former sections microprogramming is described as used in designing computers or emulating earlier computers. There are other areas of its application such as, support of earlier operating systems on newer systems, designing special purpose devices, improving system performance and maintenance [26]. A few examples will be discussed here to illustrate practical applications.

Standard Computer MLP-900 Processor [27] is designed for general purpose emulation, simulation and interpretation. The MLP processor has its own instruction set, control memory and interrupt system. This is referred to as the concept of "a computer within a computer". The read-write control store is extensively used in enhancing system microdiagnostics. The microprogram is implemented in vertical form. This choice was made to use the control memory more efficiently, in spite of some performance loss. The short length word configuration is easily microprogrammed and the language is very similar to common assembler language. Therefore, no special software is required for the development of microdiagnostics. These features enable the system to operate with an efficient microdiagnostic at low cost and short time.

Tucker-Flynn Processor [28] is dynamically microprogrammed. Its 4K read-write microstorage (64 bit/word; 50 nsec) is used for both data and subroutine storage and is about ten times faster than read-write main storage (64 bit/word; 500 nsec). The microinstructions are horizontally programmed. The decoding is minimized because 32 bits of each microinstruction can explicitly specify the operation of the processor to be performed, by setting the control gatings.

There are three basic operations referred to as addition, shiftmask, and sequence, which are executed in parallel. The set of microinstructions providing the processor with the control of a certain operation is called a macro. The rarely used macros may be located in the relatively slow main storage if micromemory space is limited. If a set of "problem oriented macros" are used, the microprogrammed processor can perform 20 times faster than the conventional processor. Algorithms must be written for these macros, that require careful study; however, the equivalent operation cannot be performed by a single conventional machine instruction. Hence the advantage of the dynamically microprogrammed processor over the conventional one lies in the appropriate design or selection of the macrosets, relevant to the particular application. Linkage of macros here is implemented by software which is also microprogrammed. More flexibility is offered to the rigid hardware design.

Burroughs B1700 [29] is a small-to-medium scale general purpose computer commercially available. Its primary objective is to interpret all programming languages in fast run time and low computation cost. Hence it is to emulate all existing and possibly future machines. No machine language is built into the hardware. The main memory is made of MOS/LSI circuitry. The 1024-bit chips have 180 nsec access time. The microcode is also in the main memory, but it may be buffered in a faster storage with 60 nsec access time. B1724 has both main memory and control memory. Its control memory is four times faster than the main memory. In B1700 series main memory is bit addressable and has variable operation lengths from zero to 24 bits in parallel. The processors use a 32-deep automatic stack. There are several compact microprograms (less than 4000 16-bit microinstructions) emulating COBOL, FORTRAN, BASIC and RPG language processors and, second and third generation machines. Hardware is implemented for parallel interpretation of many microprograms.

The so called B1700 Master Control Program (MCP) is an efficient operating system utilizing virtual memory, multiprogramming and multiprocessing concepts. The user is not limited in the amount of physical storage and the throughput is enhanced. Since there is no machine language, MCP is written in a high level language.

B. Analytical Approach to Study Flexibility of Computers

1. Representing Computers with Directed Graphs

A directed graph is constructed from a set of nodes and directed edges. In the representation of computers a set of nodes can be defined to correspond to the most vital components of the computer. A high level graphic description of a computer then would consist of a "storage unit node", an "arithmetic unit node" and edges corresponding to the interrelations among these nodes. Four types of nodes are defined to represent a computer structure adequately. These nodes are shown in Figure 24.

Memory Node: Each memory node M, represents a storage unit, the flip-flop, the flip-flop register, also the random access storage device. The memory node has one single directed input edge and a single directed output edge. Depending upon the storage device that node M represents, the edges may be weighted with number of bits.

Decision Node: Each decision node D, has a finite number of directed input edges and a single directed output edge. Through external control the node selects any single input edge and connects it to the output edge. The node then can select one of several-bit edges, depending upon the edge weight.

Fan-Out Node: Each fan-out node F, has a single directed

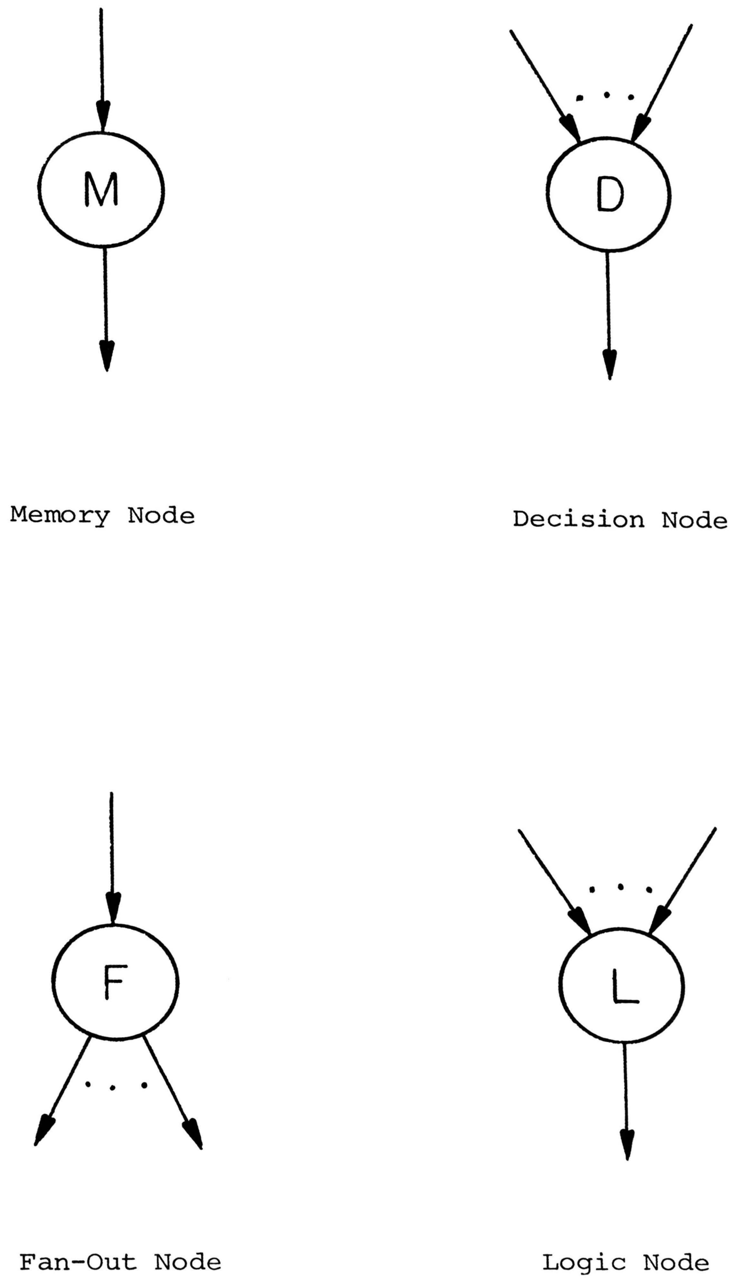
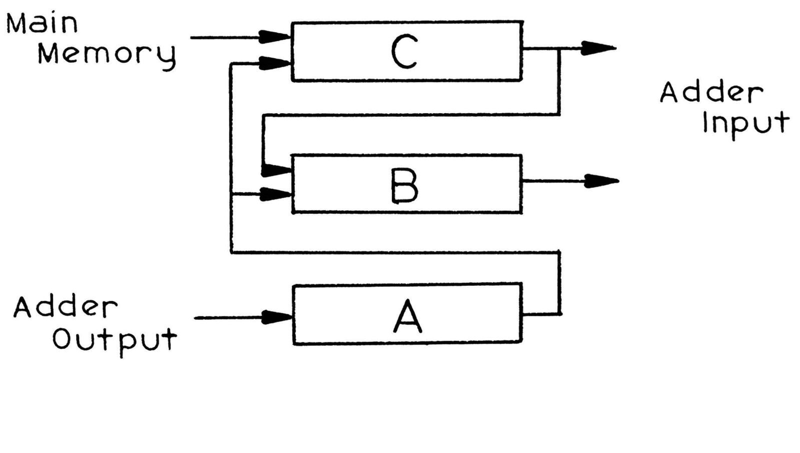


Figure 24. Node Behavioral Characteristics.

input edge and a finite number of directed output edges. The node connects the input edge to all output edges. The node then distributes the bits on the input edge to several output edges.

Logic Node: Each logic node L, has a finite number of directed input edges and a single directed output edge. The node produces an output edge, which is a logic function only of the present input edges. Different logic functions for the same set of input edges can be implemented by external control. The node can represent either a simple combinational switching network or a parallel adder. The number of bits in the parallel adder will depend upon the edge weight.

A few examples will be demonstrated to illustrate the use of the nodes. Figure 25 shows a bank of buffer registers and its directed graph. The three memory nodes represent the registers designated by A, B and C, while decision nodes and fan-out nodes are used for multiple register inputs and outputs, respectively. A multiple-port memory and its directed graph are shown in Figure 26. Using three main memories an interleaved system that gains memory access overlap by address-interleaving among memory units can be formed. Such a system and its directed graph are shown in Figure 27. The buffer registers of Figure 25 can be extended to a conventional memory-processor network shown in Figure 28. The memory nodes M1, M2, M3, M4 represent the main memory,



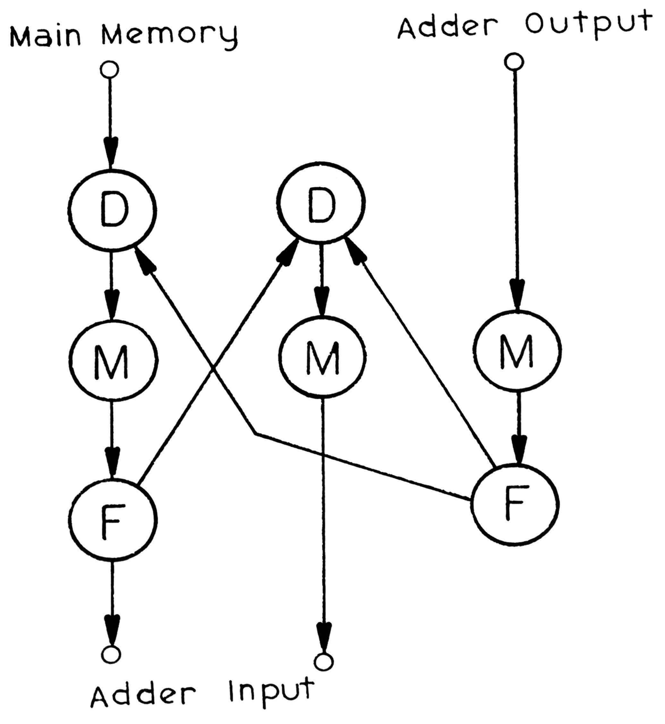
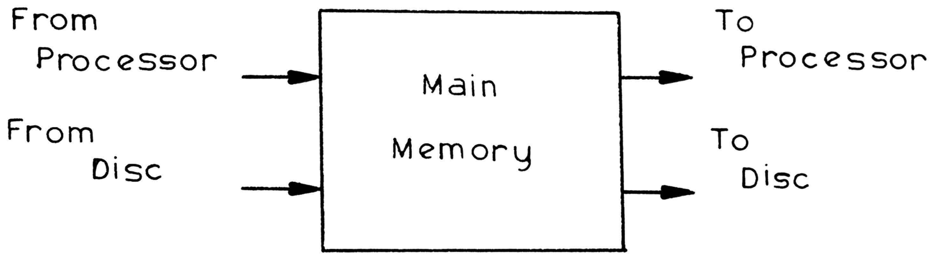


Figure 25. Buffer Registers and Their Directed Graph.



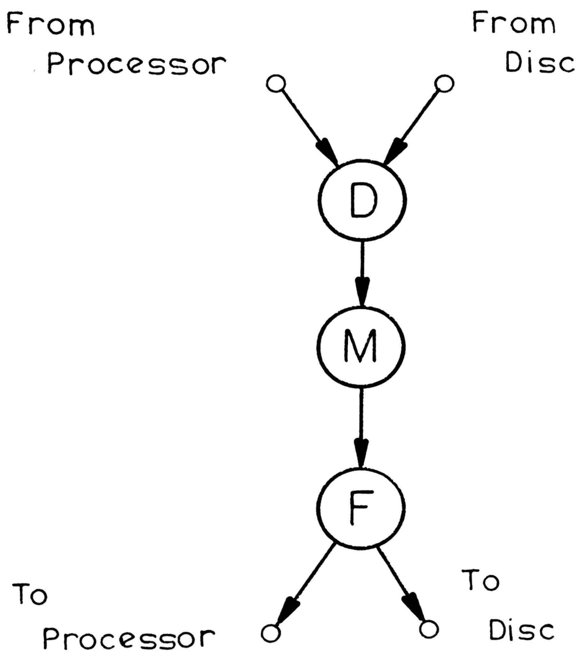
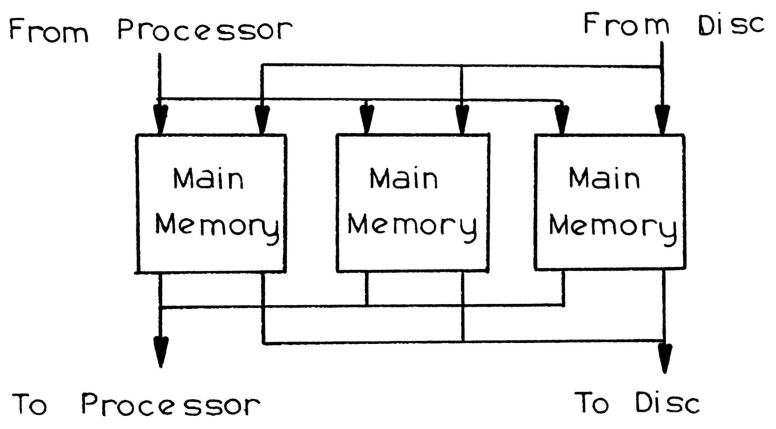


Figure 26. Multiple-Port Memory and its Directed Graph.



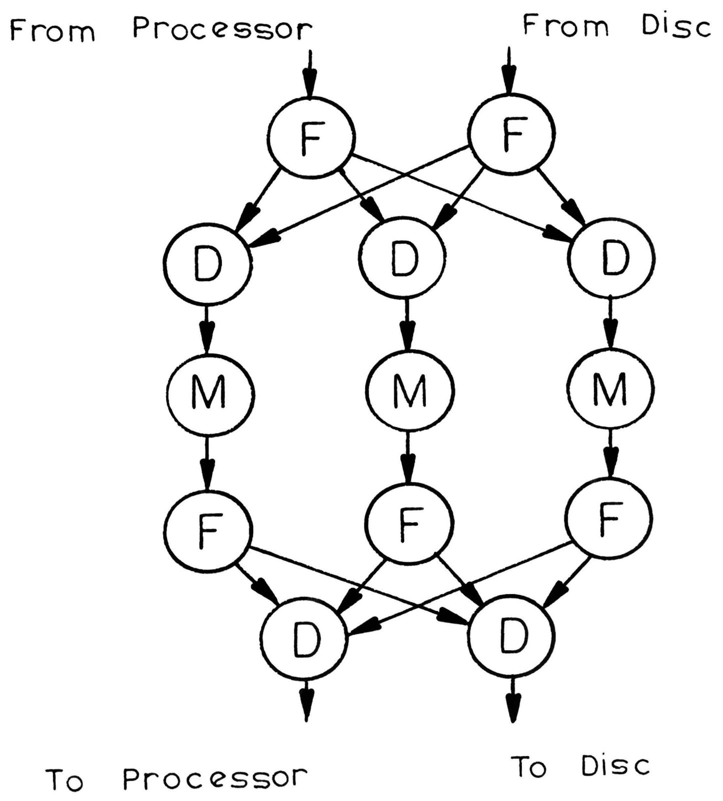


Figure 27. Interleaved Memory Units and Their Directed Graph.

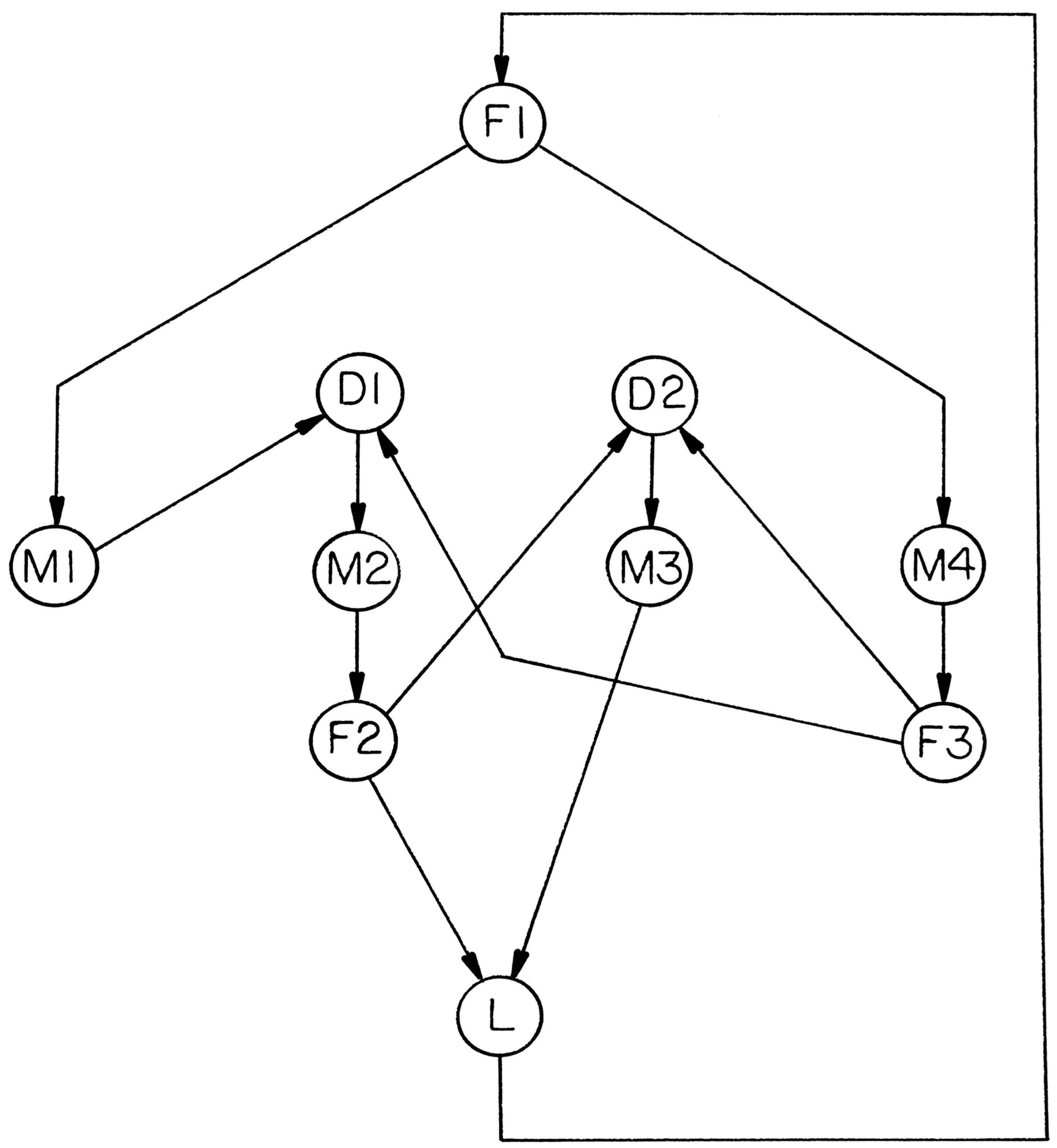


Figure 28. A Conventional Memory-Processor Network Directed Graph.

C-register, B-register and A-register, respectively. The adder is represented by the logic node L. The directed graph may adequately represent the minicomputer SCC 650 if the program counter and the index register are overlooked.

2. Measuring Flexibility on a Directed Graph

The word flexibility is a vague term and must be interpreted for the purposes of the discussion. In the following analysis flexibility will correspond to the ability of any major unit represented by the node (i) to access any other major unit represented by the node (j) directly (excluding intercommunications between logic elements). Hence the most flexible computer representation is a directed graph where every node is adjacent to every other node except logic node.

In the above section the decision node D and fan-out node F simply serve the creation of multiple inputs and outputs for the one input one output memory node and one output logic node. Therefore, in the considerations of flexibility, these nodes may be combined with memory and logic nodes to form equivalent nodes: storage node S and execution node E, respectively. These nodes are multiple input multiple output and are shown in Figure 29. Figure 30 indicates the equivalent directed graph of the memory-processor network of Figure 28.

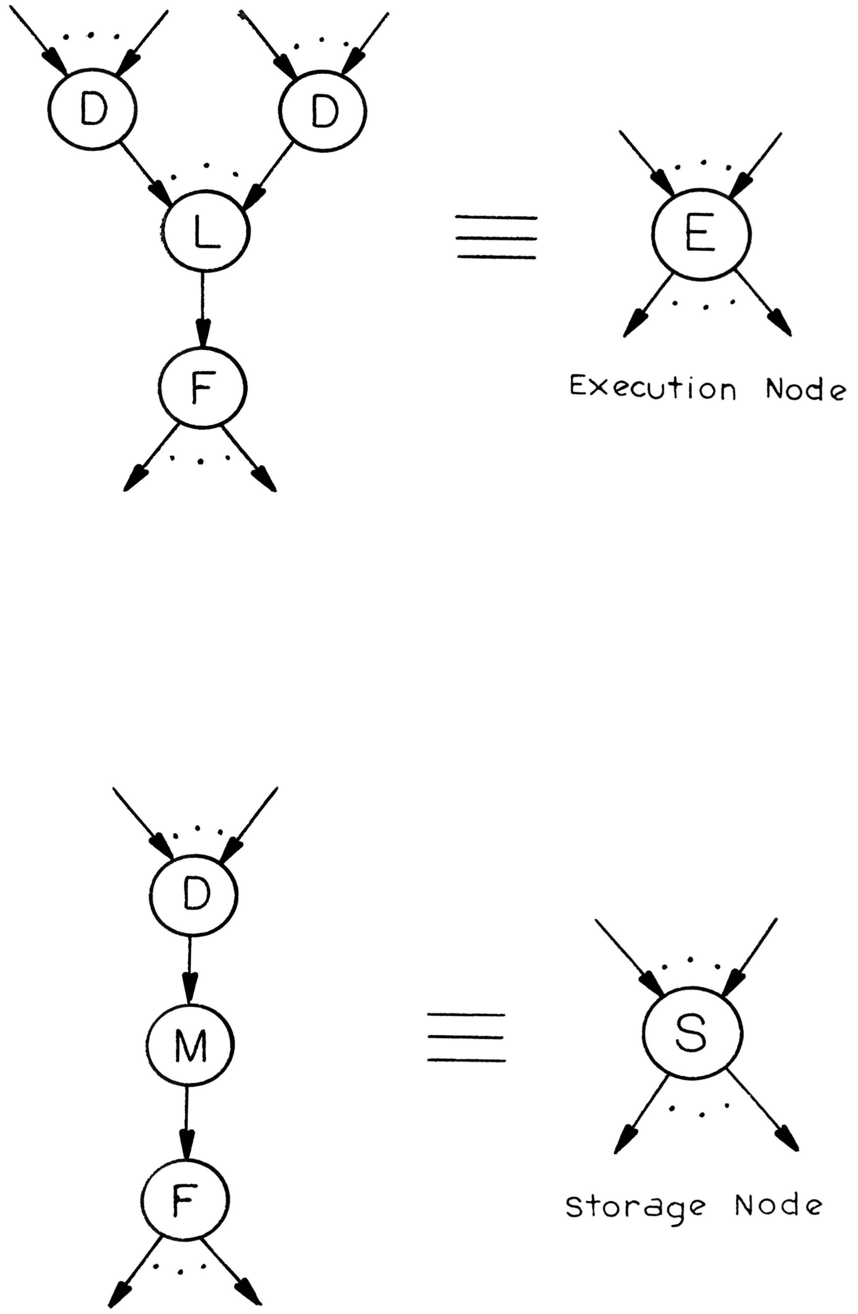


Figure 29. Execution and Storage Nodes.

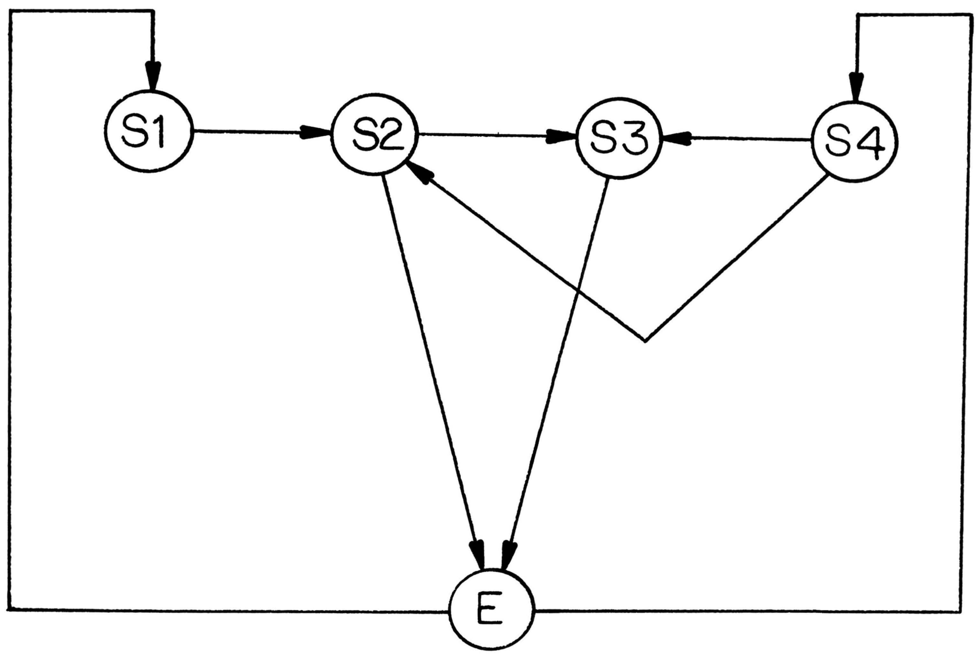


Figure 30. Equivalent Directed Graph of Figure 28.

Three values of crosscoupling are defined on a directed graph composed of equivalent nodes to measure flexibility:

CM1 is the ratio of the number (A) of edges from any storage node S, to any storage node S, to the number (B) of all possible edges between any storage node S. CM1 = A/B.

CM2 is the ratio of the number (C) of edges from any execution node E to any storage node S, to the number (D) of all possible edges from any execution node E to any storage node S. CM2 = C/D.

CM3 is the ratio of the number (E) of edges from any storage node S to any execution node E, to the number (F) of all possible edges from any storage node S to any execution node E. CM3 = E/F.

It can be seen that the sum (B + D + F) indicates the number of all possible edges in the most flexible directed graph, where the sum (A + C + E) indicates the number of existing edges in the directed graph. Therefore, a total measure of crosscoupling can be obtained as follows:

$$\text{CMT} = (A + C + E)/(B + D + F), \text{ where } 0 \leq \text{CMT} \leq 1.$$

CMT of a perfectly flexible system would be one.

The three crosscoupling measures can be found using blocks of the adjacency matrix. To find CM1 let A1 be the block of the adjacency matrix whose rows and columns correspond to the storage nodes of the directed graph.

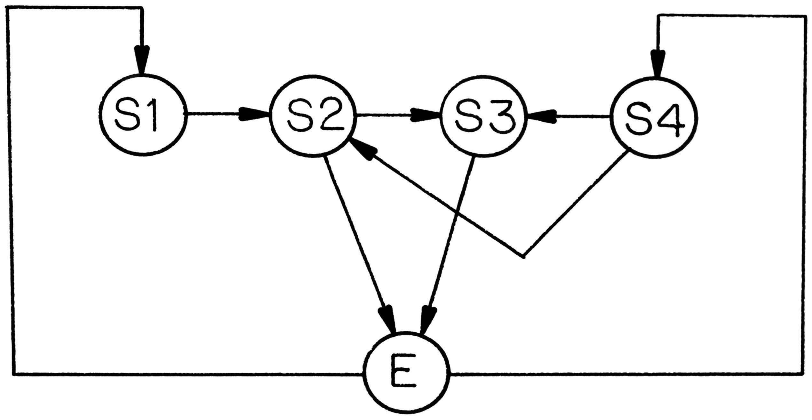
CM1 = A/B, where A is the number of 1's in the A1 matrix and B is the number of entries in the A1 matrix. To find CM2 let A2 be the block of the adjacency matrix, where rows correspond to the execution nodes and columns to the storage nodes. CM2 = C/D, where C is the number of 1's and D is the number of entries in the A2 matrix. To find CM3 let A3 be the block of the adjacency matrix whose rows correspond to the storage nodes and columns to the execution node. CM3 = E/F, where E is the number of 1's and F is n times the number of entries in the A3 matrix; n is the number of inputs of the execution node. If the directed graph has several execution nodes, find the A3 matrix for each and add the numerators and the denominators to find the ratio of CM3. A factor of n is introduced in finding F since any storage node can access any input of the execution node.

In Figures 31 and 32 directed graphs and adjacency matrices of a conventional and a microprogrammed computer are given. Table I compares the CMT values. Although the result indicates that the microprogrammed computer is more flexible, examples can be easily found to demonstrate the reverse.

Table I

Crosscoupling Measures

COMPUTER	CM1	CM2	CM3	CMT
Conventional	4/16	2/4	2/8	8/28(.286)
Microprogrammed	6/25	3/4	3/10	12/39(.308)



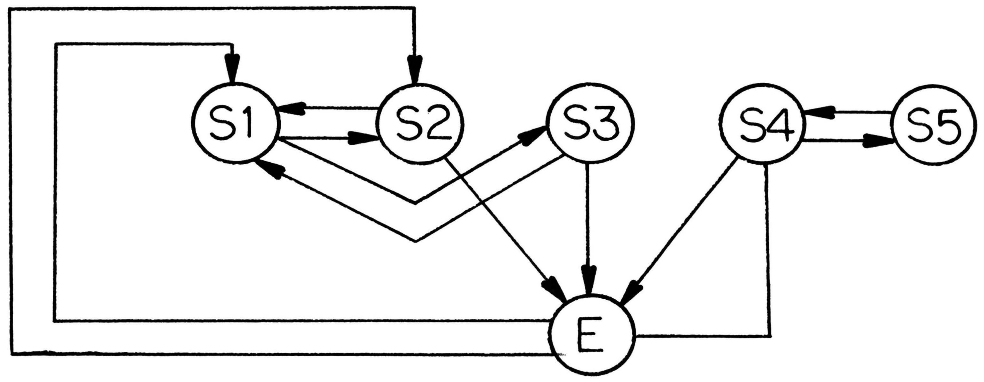
S1: Main Memory

S2–S4: Local Registers

E: Adder

$$A = \begin{array}{c|ccccc} & S1 & S2 & S3 & S4 & E \\ \hline S1 & 0 & 1 & 0 & 0 & 0 \\ S2 & 0 & 0 & 1 & 0 & 1 \\ S3 & 0 & 0 & 0 & 0 & 1 \\ S4 & 0 & 1 & 1 & 0 & 0 \\ E & 1 & 0 & 0 & 1 & 0 \end{array}$$

Figure 31. A Conventional Computer Directed Graph and its Adjacency Matrix A.



SI: Main Memory

S2-S4: Local Registers

S5: Micromemory

E: Logic Unit

$$A = \begin{array}{c|cccccc}
 & S1 & S2 & S3 & S4 & S5 & E \\
\hline
S1 & 0 & 1 & 1 & 0 & 0 & 0 \\
S2 & 1 & 0 & 0 & 0 & 0 & 1 \\
S3 & 1 & 0 & 0 & 0 & 0 & 1 \\
S4 & 0 & 0 & 0 & 0 & 1 & 1 \\
S5 & 0 & 0 & 0 & 1 & 0 & 0 \\
E & 0 & 1 & 1 & 1 & 0 & 0
\end{array}$$

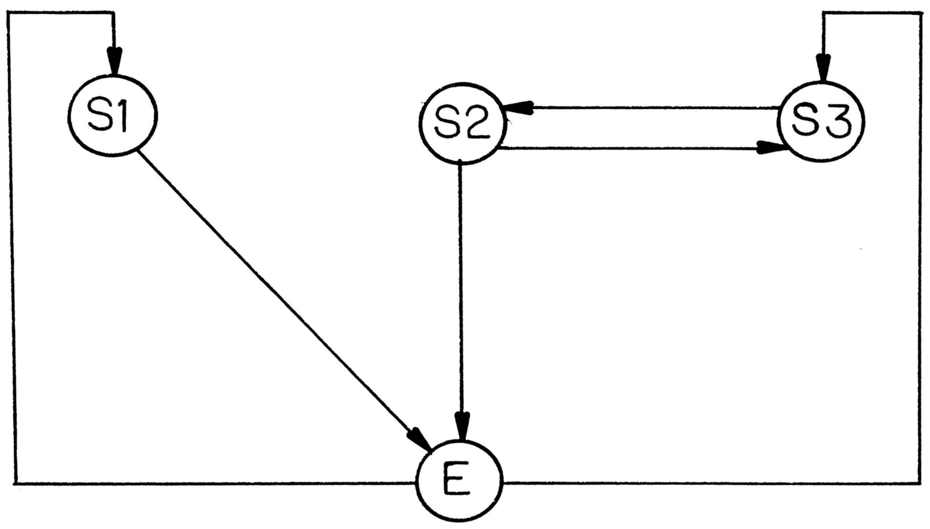
Figure 32. A Microprogrammed Computer Directed Graph and its Adjacency Matrix A.

3. Flexibility Improvements and its Effect on Cost

In the previous section total crosscoupling was defined to be the ratio of the number of the edges that the directed graph has, to the number of all possible edges. To simplify the problem the assumption was made so that each edge provided equal flexibility. If the desire is to improve the existing structure's flexibility by adding edges, we can order the edges to be added according to their importance. For example, for a certain computer structure, the addition of an edge from the main memory node to a buffer register node may serve more than the addition of an edge from a buffer register node to a logic unit node. Let us take the structure shown in Figure 33. For purposes of this discussion the edges to be added can be ordered as follows: the directed edge from S1 to S2, S1 to S3, S3 to E, S3 to S1, E to S2, S2 to S1.

Before adding these edges to the existing directed graph a simple cost analysis can be done. The cost parameter can be taken as a function of the number of pins and interconnections of components. Hence the number of edges of the graph is the determining factor for the cost parameter. Based upon this argument the cost on an edge directed to the

G1:



CMT = 6/12

Figure 33. The Structure G1.

node (i) is formulated as follows:

C = f(indegree of the node (i)).

For our purposes the function f will be assumed linear. If node (i) has one input edge (C=1), the second input edge with less flexibility indexing will have C=2. The execution node should be treated differently, since it may have input edges of C=1 as many as its logic function variables. The total cost CT is the sum of the costs on all edges.

The cost can be related to the indegree by an exponential or some other more complicated function f, that complies with the realistic figures more satisfactorily.

In Figure 34 the edges of the directed graph G1 are weighted with a linear cost function. The total-cost/total-crosscoupling ratio CT/CMT of this graph is then 7/6/12 = 14.

It was mentioned earlier that the first edge addition would be from S1 to S2, to improve flexibility. Since S2 has only one input, with the new edge its indegree will be 2. Therefore, the cost of the edge from S1 to S2 is 2. Then CT = 9, CMT = 7/12 and CT/CMT = 15.43. In Table II the effect of the addition of the edges are shown. Since a function is assumed that increases CT as CMT increases the result is reasonable. For a constant CMT value of 7/12 any edge addition except into node S3 would give a CT/CMT ratio of 15.43. The addition of an edge from S1 to S3 would have a

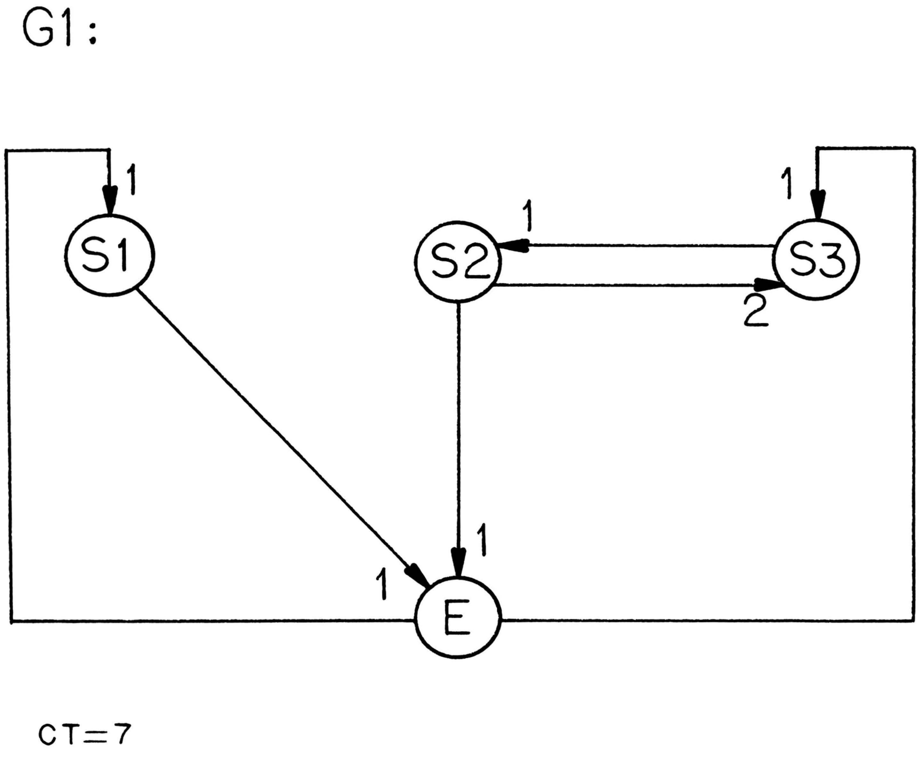


Figure 34. The Structure G1 with Cost Weighted Edges.

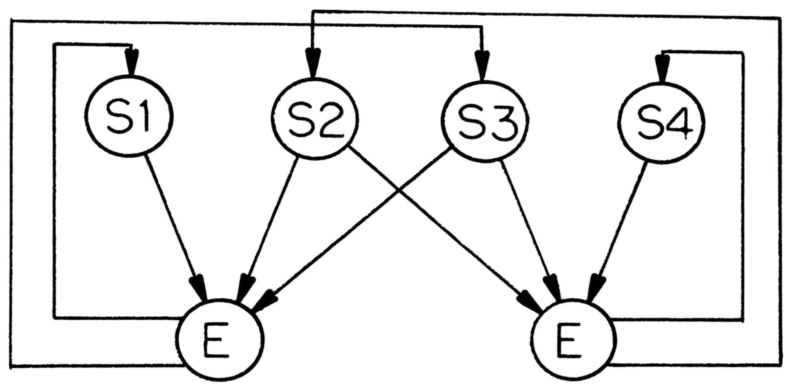
CT/CMT ratio of 17.14. Total costs can be calculated and compared for constant CMT's for all possible combinations of edges to be added.

Table II

Effect of Added Edges

ADDED EDGES	CT	CMT	CT/CMT
From S1 to S2	9	7/12	15.43
From S1 to S3	12	8/12	18
From S3 to E	14	9/12	18.67
From S3 to S1	16	10/12	19.2
From E to S2	19	11/12	20.73
From S2 to S1	22	12/12	22

If the tendency in the design philosophy favors cost savings rather than flexibility, then it is best to have CMT as small as possible. In Figure 35 the directed graph represents a dynamically microprogrammed computer. This network incorporates two execution nodes to take advantage of the parallel nature of the microinstruction and two sets of file registers shared between logic units. CMT value is .25 which is smaller than any CMT of the examples discussed. CT value is 12 and CT/CMT ratio is 48. The high CT/CMT value indicates that the structure is cheap but less flexible.



S1 : Main Memory	CM1 = 0/16
S2,S3 : Bank of File Registers	CM2 = 4/8
S4 : Micromemory	CM3 = 6/16
E1, E2 : Logic Units	CM4 = 10/40 (.25)

$$A = \begin{array}{c|cccccc} & S1 & S2 & S3 & S4 & E1 & E2 \\ \hline S1 & 0 & 0 & 0 & 0 & 1 & 0 \\ S2 & 0 & 0 & 0 & 0 & 1 & 1 \\ S3 & 0 & 0 & 0 & 0 & 1 & 1 \\ S4 & 0 & 0 & 0 & 0 & 0 & 1 \\ E1 & 1 & 0 & 1 & 0 & 0 & 0 \\ E2 & 0 & 1 & 0 & 1 & 0 & 0 \end{array}$$

Figure 35. A Dynamically Microprogrammed Structure with Small CMT and its Adjacency Matrix A.

V. CONCLUSION

In this research it has been demonstrated that a study on the fundamental level would enable graph theory to be easily used as a tool in the analysis of digital computers. The application of the blocking gate method on the graph model of the SCC 650 minicomputer for the purpose of its fault-diagnosis illustrates the simplicity of the application of some graph theory concepts. The strongly connected nature of computer structures seemed to cause problems because of the possibility of reducing the graph model down to one node, when the standard reduction techniques were used. However, this problem has been handled by breaking edges that caused the strongly connectedness of a group of nodes.

Computer structures including conventional and microprogrammed types has been represented by directed graphs and analyzed with respect to their structural behavior. This type of approach requires statistical values to be associated with nodes or edges in order to evaluate performance on the graph models. Rather than limiting the research on the numerical characteristics of a certain computer, a general scheme is provided to measure flexibility of the structures and a cost function is related to this parameter. Measurement of other parameters would follow this basic scheme possibly requiring weighted edges or nodes and computer programming to work on large graphs.

Graph theory applications to computers are not necessarily bound on the study of existing computers. Applications during the design stages would give more insight to the problem in hand, and also indicate possible improvements. Some applications have to be considered during the design stages, such as the blocking gate method. If the circuits are to be diagnosed with this method, additional circuitry must be accounted for within the design repertoire.

REFERENCES

1. E. J. Henley and R. A. Williams, Graph Theory in Modern Engineering, New York, Academic Press, 1973.

2. H. Frank and I. T. Frish, Communication, Transmission, and Transportation Networks, Reading, Massachusetts, Addison Wesley, 1971.

3. F. Harary, R. Z. Norman and D. Cartwright, Structural Models: An Introduction to the Theory of Directed Graphs, New York, John Wiley & Sons, Inc., 1965.

4. M. J. Gonzalez, Jr. and C. V. Ramamoorthy, "Program Suitability for Parallel Processing", IEEE Trans. Computers, vol. C-20, pp. 647-655, June 1971.

5. C. V. Ramamoorthy and K. M. Chandy and M. J. Gonzalez, Jr., "Optimal Scheduling Strategies in a Multiprocessor System", IEEE Trans. Computers, vol. C-21, pp. 137-147, February 1972.

6. J. Bruno and S. M. Altman, "A Theory of Asynchronous Control Networks", IEEE Trans. Computers, vol. C-20, pp. 629-639, June 1971.

7. H. C. Torng, Switching Circuits, Reading, Massachusetts, Addison Wesley, 1972.

8. R. E. Miller, "A Comparison of Some Theoretical Models of Parallel Computation", IEEE Trans. Computers, vol. C-22, pp. 710-717, August 1973.

9. J. L. Bear, "A Survey of Some Theoretical Aspects of Multiprocessing", ACM Computing Surveys, vol. 5, No. 1, March 1973.

10. T. C. Hu, Integer Programming and Network Flows, Reading, Massachusetts, Addison Wesley, 1969.

11. C. V. Ramamoorthy and L. C. Chang, "System Segmentation for the Parallel Diagnosis of Computers", IEEE Trans. Computers, vol. C-20, pp. 261-270, March 1971.

12. J. D. Russel and C. R. Kime, "Structural Factors in the Fault Diagnosis of Combinational Networks", IEEE Trans. Computers, vol. C-20, pp. 1276-1285, November 1971.

REFERENCES (cont.)

13. R. L. Kleir and C. V. Ramamoorthy, "Optimization Strategies for Microprograms", IEEE Trans. Computers, vol. C-20, pp. 783-794, July 1971.

14. C. V. Ramamoorthy, "A Structural Theory of Machine Diagnosis", in 1967 Spring Joint Comput. Conf., AFIPS Conf. Proc., vol. 30, Washington, D. C.: Thompson, 1967, pp. 743-756.

15. P. D. Stigall, "A Data-Flow Structure for Dynamic-Microprogrammed Computers", IEEE Computer Society Repository R 72-239.

16. C. V. Ramamoorthy and W. Mayeda, "Computer Diagnosis Using Blocking Gate Approach", IEEE Trans. Computers, vol. C-20, pp. 1294-1300, November 1971.

17. M. A. Breuer, "Testing for Intermittent Faults in Digital Circuits", IEEE Trans. Computers, vol. C-22, pp. 241-246, March 1973.

18. J. H. Tracey and H. T. Pottinger, "Formal Description of SCC 650 Computer", Technical Report CRL 68.1, January 1968.

19. J. H. Tracey and R. F. Crall, "Operation of the SCC 650 Digital Computer", Technical Bulletin CRL 68.1, August 1968.

20. P. D. Stigall and O. Tasar, "Graph Model Analysis of Computer Structures (Program Documentation)", Technical Report CRL 74.2, April 1974.

21. P. M. Davies, "Readings in Microprogramming", IBM System Journal, vol. 11, pp. 16-40, 1972.

22. T. L. Dollhoff, "Microprogrammed Control for Small Computers", Computer Design, vol. 12, pp. 91-97, May 1973.

23. S. S. Husson, Microprogramming: Principles and Practices, Englewood Cliffs, N. J., Prentice-Hall, Inc., 1970.

24. C. G. Bell and A. Newell, Computer Structures: Readings and Examples, New York, McGraw-Hill, Inc., 1971.

25. C. V. Ramamoorthy and L. C. Chang, "System Modeling and Testing Procedures for Microdiagnostics", IEEE Trans. Computers, vol. C-21, pp. 1169-1183, November 1972.

REFERENCES (cont.)

26. M. J. Flynn and R. F. Rosin, "Microprogramming: An Introduction and a Viewpoint", IEEE Trans. Computers, vol. C-20, pp. 727-731, July 1971.

27. R. M. Guffin, "Microdiagnostics for the Standard Computer MLP-900 Processor", IEEE Trans. Computers, vol. C-20, pp. 803-888, July 1971.

28. A. B. Tucker and M. J. Flynn, "Dynamic Microprogramming: Process Organization and Programming", Communications of ACM, vol. 14, pp. 240-250, April 1971.

29. W. T. Wilner, "Design of the Burroughs B1700", in 1972 Fall Joint Comput. Conf., AFIPS Conf. Proc., vol. 41, Part 1, Montvale, N. J., AFIPS Press, 1972, pp. 489-497.

VITA

Ömür Taşar was born on May 25, 1948 in Istanbul, Turkey. She received her Bachelor of Science degree with Honors in Electrical Engineering from Robert College, Istanbul, Turkey in June, 1971. She enrolled in the graduate school at the University of Missouri-Rolla in August, 1972 to begin work on her Master of Science degree in Electrical Engineering.