IMPLEMENTATION OF FAULT SIMULATION TECHNIQUES

A DISSERTATION

submitted in partial fulfilment of the
requirements for the award of the degree
of
MASTER OF TECHNOLOGY
in
ELECTRONICS & COMMUNICATION ENGINEERING
(COMPUTER SCIENCE & TECHNOLOGY)

By

RAHUL AGARWAL

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING
UNIVERSITY OF ROORKEE
ROORKEE-247 667 (INDIA)

May 1987

CANDIDATE'S DECLARATION

I hereby declare that the work which is being presented in the dissertation entitled, 'IMPLEMENTATION OF FAULT SIMULATION TECHNIQUES', in partial fulfilment for the award of the degree of MASTER OF TECHNOLOGY in Electronics and Communication Engineering with specialization in COMPUTER SCIENCE AND TECHNOLOGY, submitted in the Department of Electronics and Communication Engineering, University of Roorkee, Roorkee, is an authentic record of my own work carried out for a period of about ten months from August 1986 to May 1987 under the supervision of Dr. N.K.Nanda, Prof. and Head, Department of Electronics and Communication Engineering and Mr. Kuldip Singh, Reader Dept. of Cont. Education, University of Roorkee, Roorkee, India.

The matter embodied in this dissertation has not been submitted by me for the award of any other degree or diploma.

Rahul Agarwal

(RAHUL AGARWAL)

ROORKEE
MAY ,1987

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Kuldip Singh

(KULDIP SINGH)
READER
CONT. EDU.DEPARTMENT
UNIVERSITY OF ROORKEE
ROORKEE-247667, U.P., INDIA

Nanda

(N.K.NANDA)
PROFESSOR AND HEAD
ELECT. AND COMM.ENGG.DEPTT.
UNIVERSITY OF ROORKEE
ROORKEE-247667, U.P., INDIA.

ACKNOWLEDGEMENTS

I express my deep sense of gratitude to Dr. N.K.Nanda, Professor and Head, Electronics and Communication Engineering Department at University of Roorkee for his expert guidance at each step during the completion of this work. I am also deeply indebted to Mr. Kuldip Singh, Reader, Deptt. of Cont. Education, University of Roorkee whose timely guidance and co-operation has helped greatly in the completion of this work.

I am very thankful to my friend and colleague Shri Ajit Malaviya without whose help it would not have been possible to complete the work successfully.

I am also thankful to all my friand for providing an intellectually stimulating and enjoyable environment. Finally thanks are due to Mr. D.C.Bhardwaj for typing this dissertation.

RAHUL AGARWAL

ABSTRACT

Advances in digital technology has led to an increase of circuit complexity and its size in terms of the number of gates used. So it has become essential to use computer aided design and testing of digital circuits. One of the basic tools for this is fault simulation.

Various fault simulation techniques and their alternatives have been proposed in literature. In this work Parallel Simulation, Deductive Simulation, criticalpath tracing and statistical fault analysis have been implemented on DEC 2050.

The implementations are valid for combinational circuits. The single-stuck-at fault model has been used. The results have been presented in terms of the C.P.U. requirements of the different techniques. The results indicate that parallel simulation requires minimum C.P.U. time.

CONTENTS

CHAPTER - 1

INTRODUCTION

1.1 INTRODUCTION AND MOTIVATION

With the complexity of technology on the increase computer aided design and testing is becoming essential in many fields. One of the main application areas is the design and testing of digital circuits. This is specially true for the design of a large highly reliable system where it becomes very cumbersane to directly fabricate and test the digital circuits. This is because it is necessary to (i) check the logic before commitment to hardware (ii) Evaluate the logic design alternatives. (iii) Verify the fault test procedures (iv) Obtain detailed circuit operational reference data (v) consider initial dependence on timing [4].

Logic simulation enables digital system development by facilitating the above requirements. It complements actual equipment debugging and helps in reducing the time and cost of digital design [4]. It is the process of building and execcising a model of a digital circuit on a digital computer. It involves evaluation of signal values in the modelled circuit as a function of time for same applied input sequence.

Logic simulation has two principle application areas -

(i) design verification

(ii) fault analysis

Design verification is often called true value simulation and is used to determine the output sequence that will be produced by any specified input sequence.

Fault analysis is the study of faults activated by a particular input sequence. This is done by the process of fault simulation i.e. the simulation of the behaviour of the circuit under various faulty conditions. It gives information about -

(a) The faults detected by the proposed test sequence

(b) Operational characteristics of the circuit under specific fault conditions.

(c) The degree of fault resolution obtainable with a given test sequence [3]

One of the uses of fault simulation is to determine the fault coverage of a given test sequence. The fault coverage is defined as the ratio of the number of faults detected to the total number of faults simulated. It is a measure of test quality which can be used to improve the test sequence.

For larger and more complex circuits the C.P.U. time required for fault simulation increases. Various fault simulation algorithms and alternatives have emerged in response to this rapidly escalating C.P.U. time.

Most of the fault simulators that have been implemented till date can be classified into three categories.

(1) Serial simulation - This category employs the single fault injection technique which simulates either the fault free circuit or are of the faulty circuits.

(2) Parallel simulation - Here the true value and a small set of faults are simulated concurrently.

(3) Deductive simulation - In this category all fault symptoms are propogated from the site of failure to circuit outputs by use of fault list manipulations.

Some alternative simulation techniques have also been proposed. The main among them are

(i) Critical Path Tracing - It is a recursive path tracing algorithm in which detected faults are only considered.

(ii) Statistical fault Analysis - Here a statistical estimation of the fault coverage is done by true value simulation of a number of input sequences.

1.2 PROBLEM SPECIFICATION

A number of fault simulation techniques are available in literature. In this thesis a brief comparative analysis of these has been done. Further four of this techniques have been implemented on DEC-2050.

Here combinational circuits have been considered and the single stuck-at fault model has been used.

1.3 DISSERTATION OUTLINE

The thesis has been organised into four chapters.

Chapter 1 is the introduction.

Chapter 2 gives a review of the various fault simulation techniques and alternatives. It attempts a critical analysis of the various techniques.

Chapter 3 deals with the implementation of the various techniques considered. Each implementation has been explained in terms of the data structures used and the algorithms.

Chapter 4 gives the results of the various techniques implemented. It also gives the conclusion and the scope of future extensions.

CHAPTER - II

REVIEW OF VARIOUS FAULT SIMULATION TECHNIQUES

Fault simulation involves the determination of the fault-free value of every signal in the circuit, and also the values of these signals in the presence of faults from the prespecified set. The various methods of fault simulation differ in their complexity, speed, storage requirements, accuracy and the levels of models that can be simulated efficiently.

In this chapter the different methods and their relative advantages and disadvantages have been discussed. The various methods have been discussed and then they are analyzed relatively.

2.1 Serial Simulation :

This is one of the earliest and simplest method of fault simulation. This technique involves simulation of one fault at a time. The fault free circuit is simulated initially. Then the circuit model is modified to accomodate a single logical fault. Stuck type faults are inserted by simply setting the stuck signal to the approperiate constant value.

This type of simulator requires one complete pass through the circuit per fault insertion. A large circuit will have a very large number of faults and so the number of passes is bound to become prohibitive. As a result the method becomes quite inefficient. But the memory requirements for this method are less than that for other methods. So with a small host computer this method may become useful.

2.2 Parallel Simulation [3,4,11,12] :

This technique simulates the effects of a number of faults simulataneously by virtue of the fact that the operations in the host computer are word-oriented. The number of faults simulated during one pass of the simulator is usually determined by the world length of the computer. With multiple precision arithmetic a larger number of faults can be simulated simultaneously. Multiple passes become necessary when the number of faults comes large [11,12].

For two logic values and stuck-type faults a word of length n can accomodate the fault free value and (n-1) faulty values of every signal in the circuit. This is called a fult word [5]. The logical gates are simulated by performing the corresponding logical operations on the words representing its input values. A s-a-1/s-a-0

fault is inserted by ORing/ANDing with a fault mask having a 1/0 in the faulty bit position and 0's/1's elsewhere [3,12].

If the word size is 'n', then $n_f = n - 1$ different faults can be processed simulataneouely . Processing of n_f different faults simultaneously is called a pass. If N_F different faults are to be processed $[N_F/n_f]$ passes are required. Independent faults can be processed simultaneously to reduce the number of passes. Two faults are said to be independent if they cannot effect any common portion of the circuit. In a parallel fault simulator independent faults can be simulated in the same bit position. The least upper bound on the number of independent faults which can be processed simultaneously in the same bit position is the number of output lines 'o' in the circuit. So the minimum number of passes is $[N_F/n_f.p]$. The main disadvantage with this procedure is that of identifying independent faults. If this cannot be done efficiently, then the time saved in simulation is lost [4,5].

The efficiency of parallel simulation depends on the width W of a computer word as well as the instruction set and architecture of the host machine. Assuming a fixed memory cycle time and instruction execution time,

by doubling W twice as many faults can be processed in the same time. If the host machine has string operators, then even further efficiencies can be achieved [4].

The basic advantage of parallel simulation is that the gate's output fault word is computed by simple logical operations on input fault words and the memory requirements are meagre [5].

2.3 Deductive Simulation [2,3,4,5,11] :

Deductive simulation is based upon the concept, that if the inputs to logic elements and the affect of faults on the circuit for these input values are known the output of the element under all fault conditions can be deduced. Instead of computing the signal values for all faults being simulated (as in the case of parallel fault simulation), this method computes signal values for fault free circuit, and deduces the faults that will cause each signal to have a value different from its fault free value. This is accomplished by using the concept of fault list propogation [2,4,11].

The deductive simulation can process all faults via. a one pass simulation. This is done by simulating the fault-free circuit, and by associating with each line α in the circuit a set of list L_α which contains the name of every fault which produces an error on the

line 'α'. The deductive simulation method involves determining the output true values and fault lists of elements from their input true values and fault lists. The true value of an element output is computed whenever one or more of its input true values change. The output fault list is computed whenever any of its input true values or fault lists change. Set Algebra is used for fault list computations. The equation used for computing the output fault list depends not only on the type of the gate but also on its input values [3,11].

The deductive simulator is typically table-driven event directed. There are two types of events-logic events which occur when a signal line takes on a new logic value and list events which occur when a fault list L_α changes i.e. either gains or losses one or more entries [2,4].

The basic algorithm for the propogation of fault lists is given below :

1. If the value of ψ in the normal circuit is O(1), let $E(x_1, x_2 \ldots\ , x_n)$ be a sum of products or product of sums Boolean expression for ψ ($\bar{\psi}$) in terms of the inputs to the element $x_1, x_2, \ldots, x_n$. If the value of an

input variable x_i in the normal circuit is 0, replaced(all appearances of) x_i in E by L_{x_i}, and all appearances of $\bar{x}_i$ by $\bar{L}_{x_i}$; if the value of x_i in the normal circuit is 1, replace x_i by $\bar{L}_{x_i}$ and $\bar{x}_i$ by L_{x_i}. Replace, by $\cap$ and + by U.

(2) If d is the element output add $d_i(d_0)$ to the fault list obtained in (1) [4].

From the above algorithm it is clear that the computation required in determining the set of faults that cause the output of any gate in the circuit to be incorrect (for deductive simulation) is usually more complex than that required in determining the output of a gate for the normal and faulty circuits simultaneously (i.e. in parallel simulation). However, a single pass through the deductive simulation program is sufficient to determine the effects of a large number of faults. Thus, the computation time required for simulating a large number of faults is reduced, but at the expense of the memory requirements. It has been shown experimentally that if sufficient memory is available then deductive simulation is more cost effective than parallel simulation when a large circuit is to be simulated with a large number of faults [5].

2.4 Concurrent Simulation : A variation of Deductive Simulation [3-5,13] :

This technique has been proposed by Ulrich and Baker [13]. It is a variation of the deductive simulation technique. The variation is in the method of implementation rather than with the principle and has to do with the manner in which fault-list data are updated as successive test input vectors are evaluated. In the deductive technique each input vector is considered as a seperate entity. Although fault lists are passed on, any change in activity caused by a change in the input vector values usually means a complete re-assesment of each of the model fault lists irrespective of whether an update has caused the data to change. On the otherhand the concurrent simulator only updates lists if they are different. In other words, updates are resticted to nodes that change [4,5,13].

Concurrent simulation also has some of the features of one fault-at-time simulation since the evaluations of fault-free and faulty signal values are one explicitly one at a time [5].

However, only those elements whose inputs and/or states in the presence of any fault are different

from their fault free values, are evaluated. This is accomplished by maintaining fault state lists analogous to faults lists used in deductive simulation. A fault state list, consisting of fault states, is associated with each element of the circuit. Each fault state consists of a fault identifier (fault number) and the state of the element (inputs, outputs, and state variables) in the presence of the particular fault. The fault-free state of the element (usually identified by fault number 0) is also contained in the fault list of that element. Unlike fault lists in deductive simulation, fault state lists are associated with elements and not with signals [5,13].

Since faulty circuits and good circuits are processed concurrently, this technique is called concurrent simulation. Here fast simulation techniques such as table lookup can be used. Concurrent simulator only processes active circuits in contrast to deductive simulation. The difference in the processing performance of the two techniques becomes more apparent when a fault 'f' causes the circuit to oscillate. In deductive simulation it is necessary to repeatedly process each gate in a circuit loop. In concurrent simulator only a single fault is processed.

The main disadvantage of concurrent simulation procedure is that it requires more storage than the deductive procedure. When an event occurs due to a fault, the fault state list must be serially scanned to see if the fault is an entry in this list.

However, for multivalued logic concurrent simulation is ideal since each element in a super fault list is handled individually, hence all tools available for fault free simulation are available for fault simulation.

2.5 Critical Path Tracing [1,11] :

This method is based on the analysis of sunsitive traces. It is a reversal of the critical cube generation [11,15] and is essentially based on the observation that failures along a sensitive trace are detected at the circuit outputs along the sensitive paths.

This is a more efficient alternative to fault simulation. It involves fault free simulation of the circuit under consideration and then using the computed signal values for tracing paths. The distinctive features of this technique as compared to fault simulation are given below :

1. Significant saving is achieved in this technique as it directly identifies the

faults detected by a test without simulating all possible faults.

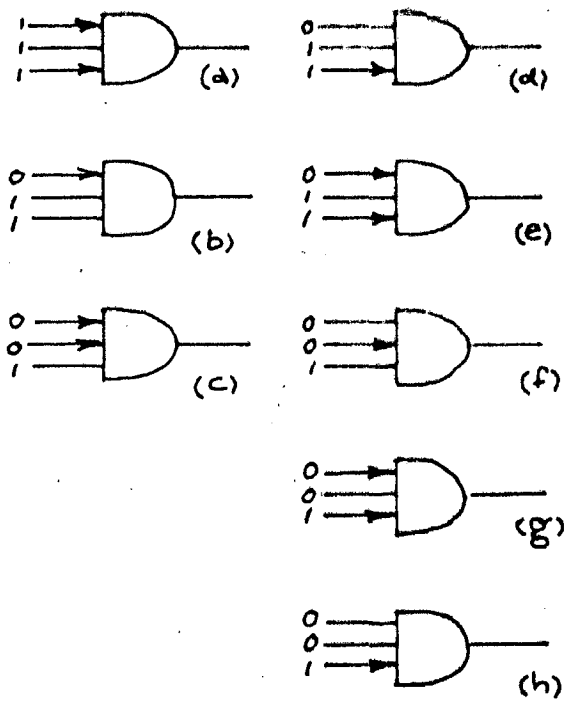
2. Since it deals with faults implicitly, so fault enumeration, fault collapsing, fault-partitioning etc. are not required.
3. Being a path tracing algorithm it does not require the computation of values in faulty circuits by gate evaluation or fault list propagation.
4. It is an approximate method in that it sometimes does not mark as detected some faults that are actually detected by the evaluated test.

The technique involves the marking of the sensitive inputs of gates during the fault free simulation run.

Next it marks the primary outputs as critical and then proceed on to a recursive tree traversal procedure so as to mark the sunsitive inputs of a gate as critical if its output is critical. For fanout free circuits this recursive procedure is enough to mark all the critical paths in the circuit.

For circuits with fanout a distinction between the stem and its fanout branches has to be made. With reconvergent fanout the phenomenon of self-masking asserts itself if the inversion parity of the different reconvergent paths is different. As a result a stem need not always be critical even if one or more of its fanout branches are critical. To compute whether the stem of a fanout branch is critical some extra operations need to be performed. Firstly the fanout free regions (ffR) of the circuit are marked out. These are the regions of the circuit for which the simple backtracing procedure explained above are applicable. A primary output is critical and this gives the starting point of the algorithm. Backtracing critical paths inside and ffR may identify some of its input as critical. An ffR input is either a primary input without fanout or a fanout branch. To determine whether the stem is critical, given that some of its fanout branches are critical, stem analysis is needed.

Stem analysis is done by using the concept of capture line. A line y is said to be sensitized to a fault on line x is the effect of that fault propogates from x to y. If a test t sensitizes line y to f, but does not sensitize any other line with the same level



(d-c) → propogation
(d-h) → non propogation

FIGURE 2.1

```
1.  Basic algorithm
      for every primary output z
      [stems to check = ϕ
       Extend Z
       while (stems to check = ϕ )
       [
       j = highest level stem
       remove j from stems to check
       if critical (j) then Extend (j)
       ]
      ]

2.  Extend(i)
    [ mark i as critical
      if i is a fanout branch then
      add stem (i) to stems to check
    else
      for every input j of i
         if sensitive (j) then extend (j)
    ]

3.  Critical (i)
    [ frontier = (fanouts of i]
      repeat
      [ i = lowest level gate in frontier
        remove i from frontier
        if frontier = ϕ then
           if propogates (i) then add fanouts to frontier
        else
           if propogates (i) then
              if i is critical then return TRUE
               return FALSE)
       ]
       ]
```

FIGURE 2.2

as y, then y is said to be a capture line of f in test t. A capture line, if one exists, belongs to all pathes on which the effect of f can propogate to the primary output in the test 't'. It can be shown that a test t detects a fault f if and only if all capture lines of f in t are critical in t. Thus, other analysis involves finding a capture line of the stem, whose status (critical or not critical) has already been determined. Then the stem is critical if the capture line is critical. This technique is referred to as partial fault effect propation.

The above technique of stem analysis involve implicit fsult effect propagation. This is the propogation of faults without explicitly computing the faulty values and is based on the marking of sensitive gate inputs done during the true value simulation. A gate G propogates a fault if either (1) fault effects arrive only on sensitive inputs of G or (2) G has no sensitive inputs and fault effects arrive on all inputs with the controlling value and only on these inputs. An illustration for the case of a three input AND gate is shown in the Figure. 2.1

The basic algorithm required for the implimentation of this technique is shown in Figure 2.2 It

assumes that the true value simulation including the marking of the sensitive inputs has been performed. The critical path tracing ~~ansi~~ an ffR is represented by the procedure EXTEND and the checking of a stem for criticality is done by the procedure CRITICAL.

As expected, the performance of this algorithm is strongly influenced by the fanout structure of the circuit and less by its size. It is claimed that an the average critical path tracing was 60% faster than the concurrent simulator. Only in the most case circuit are increase of about 30 % time was registered with critical path tracing.

The key factors contributing to the increased efficiency of critical path tracing compared to fault simulation are as follows :

1. It deals directly only with the detected faults rather than all possible faults.
2. It deals with faults only implicitly rather than explicitly.
3. It is an approximate rather than an exact technique.

An example has been shown in Figure 2.1 3 to illustrate certical path tracing.

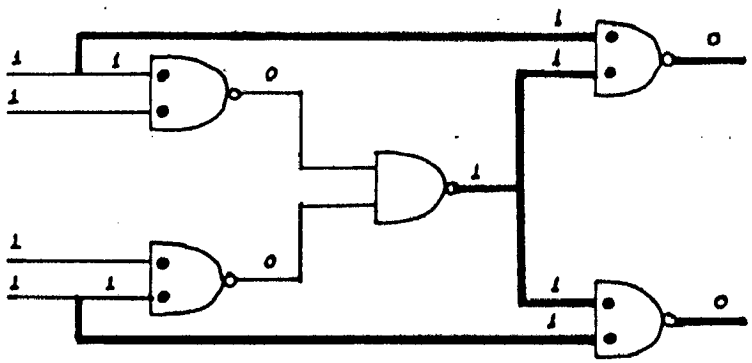


FIG. 2.3

2.6 Statistical Fault Analysis [8,9,12] :

Besides critical path tracing another alternative to fault simulation is statistical fault Analysis or STAFAN. As VLSI circuits grow in complexity, fault simulation costs get increasingly burden some. The fault simulation techniques consume CPU time and memory at a rate increasing at least as the square of the number of gates in the circuit. This is a serious limitation [8].

Stafan has been proposed to overcome these limitations. In this technique fault free simulation is used to calculate the detection probability of each stuck fault in the circuit from signal statistics. It produces an estimate of the fault coverage by the vector stimuli used in simulation. It makes use of the concepts of controllability and observability with one defined as follows :

(a) C1(ℓ) [C0(ℓ)] or one-controllability [zero-controllability] is the probability of the line ℓ having a value of one [zero[or a randamly selected vector.

(b) B1(ℓ) [B0(ℓ)] or one-observebility[zero-observability] of a line ℓ is the probability of observing or one [zero] on the line ℓ, at the primary output [8].

Gate type	Quantity	Expression
AND	B1(I/P)	B1(O/P).C1(O/P)/C1(IP).
	BO(I/P)	BO(O/P).(3(I/P)-C1(OIP))/CO(I/P).
OR	B1(I/P)	B1(O/P),(S(I/P)-CO(O/P)/C1(I/P)
	BO(I/P)	BO(O/P).CO(O/P)/CO(I/P)
NAND	B1(I/P)	B1(O/P).CO(O/P)/C1(I/P)
	BO(I/P)	BO(O/P).(S(I/P).(S(I/P)-CO(O/P))/ C1(I/P)
NOR	B1(I/P)	BO(O /P).(S(I/P)-C1(O/P))/C1(I/P)
	BO(I/P)	B1(O/P).C1(O/P)/CO(I/P)
Not/ XOR/ XNOR	B1(I/P)	B1(O/P)
	BO(I/P)	BO(O/P)
CONT	B1(I/P)	$\bigcup_{i=1}^{m} B_1(O/P)$;
	BO(I/P)	$\bigcup_{i=1}^{m} BO(D/P)$; where m = fanout [13].

Figure : 2.4

To evaluate the controllabilities we associate with each mode of the given circuit a one-canter and zero-canter. After simulating 'n' different test vectors the controllabilities are estimated as follows :

$$C1(\ell) = \frac{\text{one-cont}}{n} \text{ and } CO(\ell) = \frac{\text{zero cont}}{n}$$

An additional sensitization counter is associated with every node for the estimation of observabilities. This canter is incremented only when the path from the corresponding line to the gate output is sensitized. The sensitization propability is estimated as

$$S(\ell) = \frac{\text{sensitization-cont}}{n}$$

Com compute the observabilities B_1 and B_0 for the PO's one set to unity. Next they are propogated towards the primary inputs. The formulae for the evaluation of the observabilities of the inputs of various gate type are shown in Figure 2.4 [9].

For fanout points the obserability of the input is given by

$$BO(\ell) = (1 - \alpha)\max_{1 \le K \le m} [BO(i_k)] + \alpha \sum_{k=1}^{m} BO(i_k)$$

Here i_k are the fanout branches and α is an orbatrary constant.

After the observabilities of all the nodes have been computed, the detection probabilities are computed for all the nodes as shown below : The detection probability of ℓ,s-a-1 fault is

$$D1(\ell) = BO(\ell).CO(\ell)$$

similarly for ℓ-s-a-0 we have

$$DO(\ell) = B1(\ell).C1(\ell).$$

From the detection probabilities the fault coverages can be estimated after normalization.

Two types of operation are required in this technique - (a) updating of counters and (b) computation of controllabilities and observabilities. The first operation accounts for stafan's main overhead which is evaluated as

$$\text{Overhead } \alpha\ (4 + n_j^2 + an_i + a)\ ;$$

where,

n_j = avg. No. of 1/pS per gate

n_i = avg. No. of gates in a loop

and $a < 1$

Being a statistical method is bound to show better results for larger circuits. It is claimed that the estimated fault coverage as predicted by STAFAN closely follows the fault simulation results.

2.7 Comparison of the Different Techniques [1,4,5,8,10,11] :

Since simulation is an essential step of computer-aided design process for digital systems, the efficiency of particular simulator implementation becomes a major concern. A comparison of techniques discussed can be made on the basis of their complexity, speed, storage requirements and accuracy [4].

The performance of fault simulation algorithms is of vital importance because it can consume many hours of C.P.U. time. A growing circuit size generally results in a growing fault list and a larger number of tests being required to test the circuit. In the worst case time may increase in proportion to the 3rd power of the circuit size [4,5,10].

A comparison of the parallel fault simulator with deductive fault simulator [5] indicated, that in general, the deductive fault simulator is faster but the parallel fault simulator has an advantage for small, highly sequential circuits of sizes upto about 600 gates [5,10].

In principle, deductive and parallel simulators assume identical timing for fault free and faulty circuits because of the strong coupling between fault-free and faulty signal changes in both the techniques [11].

In concurrent simulation the timing of each fault will be handled independently [11].

The parallel simulation requires a word for each node of the circuit, so the storage requirement is fixed for a given circuit and for the number of faults simulated in one pass [11]. For deductive and concurrent simulators the storage requirements will depend on the circuit activity [11].

In the case of deductive simulation more storage is required for sequential circuits due to larger test sequences and longer fault lits. The length of fault lists can vary widely requiring dynamic storage allocation. The performance of deductive simulation can be adversely effected by excessive paging in the host computer.

The fault list in the deductive simulator stores fault numbers which in fact are positions in the parallel simulators fault word of those bits which complement the true value. If the fault words are sparsely populated the deductive technique provides a saving in the C.P.U. time and fault storage. As the fault word becomes densely populated the parallel simulator requires less C.P.U. time and fault storage [5].

Concurrent simulator is expected to require more storage since in this method each fault is represented by a fault identifier and a fault state and the concurrent fault list contains a larger number of entries [11].

Another performance criteria is the capability to accomodate larger set of logic values. This is necessary to keep up with the modern technologies. In the context of 3-valued system it has been shown that deductive simulation is more pessimistic than both parallel and concurrent simulation [11]. In parallel and concurrent simulation each fault is treated independently of the fault free circuit. In the case of deductive simulation, no information about faulty values is maintained if its true value is known.

Among the alternatives to fault simulation we have critical path tracing. Although this technique is faster than traditional fault simulation, it has been shown that certain faults detected by multiple path sensitigation are not identified as detected by this technique [1]. The algorithm saves C.P.U. time since it deals only with the detected faults rather than with all possible fault . A serious disadvantage of the technique lies in the increased complexity of the algorithm. Besides this, the method has limited applicability to sequential circuits and its performance is limited by the amount of fanout in the circuit [1].

The other alternative is statistical fault analysis. It gives an estimate of the fault coverage by using true value simulation. It does not explicitly simulate the faults like all the previous methods. The estimation of fault coverage helps in analysing the effectiveness of test vectors. The fault coverage and undertected fault data obtained for the actual circuits are shown to agree within 5 % of fault simulation results, yet the C.P. time and memory requirements fall short of those required in fault simulation. A disadvantage of this method is that it does not give information on exactly which faults are deected by a given test sequence [8,10].

CHAPTER - 3

IMPLEMENTATION

This chapter deals with the design and implementation of the various fault simulation techniques. The chapter is divided into 5 parts. The first part deals with the preprocessing which is common to all the Techniques. Each of the remaining parts deals with a fault simulation technique. Each section is further divided into two parts. The first part deals with the data structures involved and the second part deals with the algorithmic flow of the particular implement. ation. All the methods implemented use stuck-at-fault modelling.

3.1 PREPROCESSING

The first step in the design of any of the simulation algorithms is to prepare, read and check the circuit connectivity description. This requires an input description for input data statement describing the logic of the circuit. The input description of the circuit is read from a file, and the connectivity information is obtained.

3.1.1 Data Structures for preprocessing

The convention adopted is to express all primary inputs as X1, X2,.....,Xn, all interconnection nodes as C1,C2,...,Cn and the primary outputs as Z1, Z2,...,Zn. The labelling of the interconnections is done by using Terminal Numbering Convention (TNC) where each device output is labelled

with an integer greater than the integer used to label the devices interconnection inputs.

A language has been defined to describe the logical circuit. This language consists of

(i) Device number

(ii) inputs to the device

(iii) outputs of that device

(iv) Type of the device.

A slash - '/' has been used as a delimiter character identify the data baundaries as described above.

The general format of an input record is shown below:

<device no.>/<inputs>/<outputs>/<device type>/

The various options allowed are given below

Device number = Gi

Input = Xi, Ci

Output = Ci, Zi

device type = AND, OR, NAND, NOR, XOR, XNOR, NOT, CONT

Here CONT represents a continuity block between is a fanout point in the circuit. All other devices are standard logic gates.

Only the basic elements mentioned above have been considered since only gate level simulation has been considered here.

The device input can either be a primary input Xi or some intermediate line Ci, similarly outputs of a device can be either an intermediate connection is or a primary output Zi.

The data structures created from the given circuit description are enumerated below.

1. TYPE : This gives the type of the gate interms of the device code.
2. ICNT : It is the number of inputs to the gate.
3. INPUT: Gives the input node number of the gate in terms of the numeric code generated by the program.
4. OCNT : It is the number of output to the gate.
5. OUT : gives the node number of the outputs of the gate.
6. PRINCN = number of primary inputs in circuit.
7. PROTCN = number of primary outputs in the circuit.
8. INTCNT = number of interconnection
9. NODNUM = total no. of nodes in the circuit.
10. NOGA = total no. of gates in the circuit.

Besides the above data structures a connectivity matrix is created for the ease of evaluation. The rows of the matrix correspond to the device inputs and columns correspond to the outputs of the devices. So the identifying numeric code of a device is placed in the appropriate position to give the connectivity information of that gate.

The data structures are clearly depicted in Fig. 3.1 for an example.

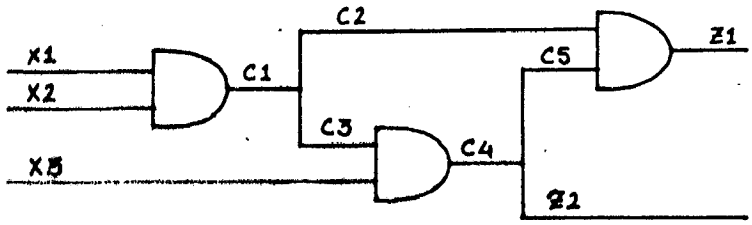


FIG. 3.1 (a)

G1/X1,X2/C1/AND/

G2/C1/C2,C3/CONT/

G3/X3,C3/CA/OR/

G4/C4/C5,Z2/CONT/

G5/C2,C5/Z1/AND/

(b) Ckt description

Symbol	X1	X2	X3	C1	C2	C3
Code	1	2	3	4	5	6

Symbol	C4	C5	Z1	Z2
Code	7	8	9	10

(c) Node Coding

Gateno.	type	icnt	inputs	ocnt	outputs
1	1	2	1,2	1	4
2	8	1	4	2	5,6
3	2	2	3,6	1	7
4	8	1	7	2	8,10
5	1	2	5,8	1	9

(d) Generated description.

	1	2	3	4	5	6	7	8	9	10
1				1						
2				1						
3							2			
4					8	8				
5									1	
6							2			
7.								8		8
8									1	
9										
10										

(e) Connectivity description.

3.1.2 Algorithms for preprocessing

The preprocessing can be divided into a number of steps for better understanding of the various algorithms. This routine could be combined with the routine SIMUlation in the various methods but has been implemented as a seperate entity/to differentiate the two modules. The various steps in the implementation of the preprocessing step are given below.

1. Initialize the variables INDEX, STYPE, CKTDES and CONECT.

2(a) Read the number of gates in NOGA

(b) Read the circuit description in CKTDES so that each row gives description of are gate in A1 format.

3. Change the numerals in CKTDES from A to I format as follows

```
Repeat
  Repeat
    if CKTDES(J,I)=INDEX(G) for any I then
          CKTDES(J,I) = G-1
          if CKTDES(J,I+1)=INDEX(G)then
             CKTDES(J,I)← CKTDES(J,I)* 10+(G-1)
  until all indexes are considered
until all gates are considered.
```

4. Initialize PRINCN, INTCNT and PROTCN to zeros and find the number of primary inputs, interconnections and primary outputs as follows:

```
  Repeat
  I ← 2
  Repeat
  if CKTDES(J,I)='X' then
     if PRINCN<CKTDES(J,I)then PRINCN - CKTDES (J,I+1)
  If CKTDES(J,I)='C' then
     if INTCNT<CKTDES(J,I+1)then INTCNT - CKTDES(J,I+1)
  if CKTDES (J,I)='Z' then
     if PROTCN<CKTDES(J,I+1) then PROTCN← CKTDES(J,I+1)
  until the present row is covered
  until all gates are considered.
```

5. Delineate the different fields in the description into CON (I,D,N) with I as pointer to gate number, D as pointer to the field number and N as pointer to the data in the field as follows

```
Repeat
D← 1
J← 0
N← 1
    Repeat
    if CKTDES (I,J) ≠ '/' then
       begin
       CON(I,D,N)← CKTDES (I,J)
       Increment N
       end;
    else
       begin
       CON(I,D,N)← CKTDES(I,J)
       N← 1
       Increment D
       end
    until D=5
Until all gates are coused by  I
```

6. Create the data field ICNT, INPUT, OCNT, OUT and TYPE

Gate type coding is dare as follows

```
if CON(I,4,J) = 'AND' then TYPE(I) ← 1
if CON(I,4,J) = 'OR' then TYPE(I) ← 2
if CON(I,4,J) = 'NAND' then TYPE(I) ← 3
if CON(I,4,J) = 'NOR' then TYPE (I) ← 4
if CON(I,4,J) = 'XOR' then TYPE (I) ← 5
if CON(I,4,J) = 'XNOR' then TYPE (I)← 6
if CON(I,4,J) = 'NOT' then TYPE (I) ← 7
if CON(I,4,J) = 'CONT' then TYPE (I) ← 8
Repeat
ICNT(I)← 0
OCNT(I)← 0
   Repeat
   N← 0
     Repeat
     increment N
     if CON(I,D,N) = 'X' then
        begin
        increment N
        increment ICNT(I)
        NODE (CON(I,D,N),1)← 'X'
        NODE (CON(I,D,N),2)← CON(I,D,N)
        INPUT(I, ICNT(I))← CON (I,D,N)
        end
     if CON (I,D,N) = 'C' then
```

```
        begin
        increment N
        NODE(CON(I,D,N)+PRINCN,1)='C'
        if d ≠ 3 then
            begin
            increment ICNT(I)
            INPUT(I,ICNT(I))←CON(I,D,N)+PRINCN
            NODE(CON(I,D,N)+PRINCN,2)=CON(I,D,N)
            end,
        else
            begin
            increment N
            increment OCNT(I)
            OUT(I,OCNT(I))← CON(I,D,N)+PRINCN
            MODE (CON(I,D,N)+PRINCN,2)←CON(I,D,N)
            end;
    if CONC(I,D,N)='Z' then
            begin
            increment N
            increment OCNT(I)
            OUT(I,OCNT(I)←CON(I,D,N)+PRINCN+INTCNT
            NODE(CON(I,D,N)+PRINCN+INTCNT,2)← CON(I,D,N)
            end;
    until CON (I,D,N) = '/'
until all gates are considered
```

7. Create connectivity matrix

```
Repeat
N ← F/10
CONECT(1,F+2) ← N
CONECT(2,F+2) ← F-N* 10
CONECT(F+2,1) ← N
CONECT(F+2,2) ← F-N*10
   Repeat
     Repeat
     if INPUT(g,h)=F then
        CONECT(F+2, OUT(G,1)+2) ← TYPE(G)
     until all inputs are considered
   until all gates are considered
until all nodes are considered
```

8. Output the device coding, node coding, connectivity matrix and fault coding.

3.2 PARALLEL SIMULATION

This technique involves logical operations at the bit level between the computer words. This is possible only in assembly language programming. So this program has been written in MACRO the DEC-2050 assembly language. The preprocessing of the previous section which has been implemented in FORTRAN has been used as the main program for whioh the MACRO subroutine for parallel simulation has been called. The data structures and algorithm of of routine PALSIM have been discussed here.

3.2.1 Data Structures

All the data structures created by the preprocessing program are used in this routine. Besides them it creates an array NODE which consists of a sets of 7 words, each such sets corresponding to a node in the circuit. These words carry the results of the simulation upto the corresponding node. To account for faults generated at the node the bit numbers (2* node no -1) and (2* node.no.) are set to 0 and 1 respectively. This is done by defining the logical length of a byte =2 bits and using the Adjust byte pointer instruction.

At the end of the simulation the program compares the different bits in the primary output word with the fault free bit (i.e. bit no.1). If faulty it evaluates the fault no. and stores it in FTDT which keeps track of the fault numbers detected.

3.2.2 Algorithm for Parallel simulation Program

The algorithm for the preprocessing program has already been given in section 3.1.1. Here the algorithm for the routine PALSIM has been given.

1. Save the Accumulators in SREG.

2. Phase 1 : Set up true values of the primary inputs.

```
X ← 0 , primary input number
while X < PRINCN Do
    begin
       Y ← True value (X)
       BITS(Y) ← Y
         A ← X
         M ← X *NNODES;index into the block of
                       memory used for nodes
         N ← 0;
         Repeat
             NODE(M) ← Y : managing multiple
             increment M ; words for a node
             increment N
         Until N=NNODES, number of words per node
             call Setftt; set faults for this node
         increment X
    end, (of while)
```

3. Phase 2, computation of intermediate and output Nodes

```
    while X<NOGA Do
      Begin
         W ← X * 10; Adjust index for 2D Array
         Z ← TYPE(x)
         R ← INPUT(W)-1;Inputnode
         V ← OUT(W)-1; outputnode
         N ← ∅
         N ← R * NNODES
```

```
    Repeat
      S ← NODE(M); Int inputnode
      Y ← 1
          Repeat
            Y ← Y+1
            W ← W+1
            R ← INPUT(W)-1
            R ← R * NNODES+N, remaining inputs
            S ← S (OP) NODE(R)
          Until Y ≥ ICNT(X)
       If Z=8 then ; continuityblock
          Break to LABLE 11
       Else
          Begin
            R ← V * NNODES (+N
            NODE(R) ← S
            Increment N
            Increment M
          end (If)
    Until N ≥ NNODES
    CALL SETFLT
LABLE 11; Increment X
  end (while)
```

4. Phase 3: Check for faults detected and list them.

```
X ← PRINCN+INTCNT;Starting count of output nodes
C ← ∅
  while X<NODNUM Do
    begin
      B ← (*25 ; index into fault list
      M ← X * NNODES;
      R ← NODECM
    ; If fault free output value is '1'
      complement all words of this output node
      If R < 0 then
         begin
           D ← M
               N ← 1
           while N<NNODES DO
               begin
                  increment D
                  S ← NODE(D)
                  S ← COMPLEMENT(S)
                  NODE(D) ← S
               end; (while)
            R ← COMPLEMENT(R)
          end(if)
Z ← NODNUM *2; total number of faults
A ← ∅
D ← ∅
Rotate R
W ← 17
```

```
LABEL Z1:
    increment D
    decrement W
    If R<0 then; fault detected
       begin
         Increment A
         FTDT(D) ← D; fault number
         Increment B
       end (if)
    If D ≥ Z, then; all faults checked
       begin
         increment X
         NFTDT(C) ← A; total number of faults on this output
         increment C
         (Break to while Loop)
       end;
    Rotate R
       If W>0 then; this word completed
           begin
             Increment M
             R ← NODE(M)
             W ← 18
           end; (if)
        Go TO LABLE Z1
    end;( while)
```

5. Restore saved Accumulators

   Return to Main.

3.3 DEDUCTIVE SIMULATION

The basic algorithm used is given in section 2.3. The preprocessing step is the same as in section 3.1. This section deals with the data structures and implementation philosophy of deductive simulation.

3.3.1 Data structures used:

Since this technique involves propogation of fault lists along with the true values during the fault free simulation seperate data structures need to be created for:

(a) propogation of true values

(b) faults generated at each node

(c) fault lists at each node

(d) evaluation of fault lists at the outputs of a gate, given the fault list at all the inputs.

For the true value simulation the data structures created in the preprocessing step are used. The new arrays created are X(I,J) and TVALUE(I). Here X(I,J) gives the Input vectors (or tests) to be simulated. TVALUE is the true value of a node which is obtained during the simulation of a particular test vector.

To account for faults generated at each node a variable FG is created for each node number. It stores the fault generated at that node in accordance with its true value.

The fault list at each node is stored in an array FT. It includes the faults propagated upto that node and also the generated fault.

For the evaluation of the fault lists at the gate outputs the Union, intersection and complementation of the fault lists of the faults lists at the inputs of the gate need to performed. To enable these operations without destroying the actual fault lists at the different the temperory arrays - JFT, UFT, IFT and CFT have been created to contain the intermediate results of these evaluations.

To output the information of faults detected the faults detected in the previous simulation run are stored in PFT. The difference between FT and PFT i.e. additional faults detected are stored in AFT.

3.3.2 Algorithms for Deductive Simulation

The MAIN program is same as the program for preprocessing of the circuit. The other routines in this program are SIMU, a routine for each gate type. UNION, INTER, COMPLE, COMPØ and COMP1. The Algorithmic flow of this routines is given below:

1. SIMU : This routine controls the whole simulation process and outputs the fault detection information for each test. It calls the routines for the different gates in accordance with the TYPE of the gate. The steps followed in this routine are given below.

(i) (a) Read the number of input tests in XIN.

(b) Read the tests in X(I,J) with a row giving one test vector. Repeat step (ii) to (v) XIN times for each test in X.

(ii) (a) Assign the true values of all the primary inputs into TVALUE(1) to TVALUE(PRINCN).

(b) Assigns the faults generated at the primary inputs as

if (TVALUE(J)=0) then FG(J)=2*J-1(s-a-0)

if (TVALUE(J)=1) then FG(J)=2*J (s-a-1)

(iii) Call the routines for the different gates according to the types until all the gates are simulated. The routines return true values of the outputs in TVALUE(output node number) and the faults transmitted upto and including the outputs in FT(I,J).

(iv) For the Ist test, copy NFT and FT for each primary output into NPFT and PFT respectively. For the subsequent tests compute the additional faults detected and store them in AFT. Update PFT each time so that additional faults detected for the next test can be computed.

(v) Output the faults detected/additional faults detected by each test at each primary output. Also output the CPU time required to simulate the given no. of tests.

2. Routines for the different gates

The routines for the different gates perform the following functions -

(i) Assign true value to the output of the gate

(ii) Call the routines UNION, INTER, COMPØ, COMP1 in accordance with the true value of the output to evaluate the fault list at the output. (Table 3.2).

For NOR and XNOR gates all the faults are propogated so only UNION is called.

In case of NOT gate and the CONTinuity block the fault list at the input is propogated to the output.

(iii) Compute the fault generated at the output and include this fault in the fault list of the output.

TABLE 3.1

INPUTS		OUTPUTS					
A	B	AND	OR	NAND	NOR	XOR	XNOR
0	0	0	0	1	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	1	0	0	0	1

EVALUATION OF TRUE VALUE OF OUTPUTS.

TABLE 3.2

TRUE VALUE OF OUTPUT	TYPE OF GATE				
	AND(NAND)	OR(NOR)	NAND(AND)	NOR(OR)	XOR/XNOR
	Call comp 1	Call comp 1	Call comp ∅	Call Comp ∅	Call Union
	Call Inter	Call Union	Call Union	Call Inter	

ROUTINES CALLED BY EACH GATE

3. Procedure Union : This procedure evaluates the union of the fault lists of the inputs.

```
NODE1 ← INPUT(I,1)
J ← 2
Repeat
NODE2 ← INPUT(I,J)
if NODE1=0 then
   begin
   UFT(K) ← JFT(NODE1,K) for all K
   UNFT ← NFTJ(NODE1)
   end;
   Repeat
   if JFT(NODE2,L)≠UFT(K) then
      begin
      if k>UNFT then
         begin
         increment UNFT
         UFT (UNFT) ← JFT(NODE2,L)
         end;
```

```
      else
         increment k
      end;
   else
      begin
      increment L
      K ← 1
      end;
   until L=NFTJ(NODE2)
   NODE1 ← 0
   increment J
Until J = ICNT(I)
Copy UFT(K) into FT(output,K) and
UNFT into NFT(output)
Return
```

4. Procedure Inter: This procedure evaluates the intersection of the fault lists of the inputs of the gates.

```
NODE1 ← INPUT(I,1)
J ← 2
Repeat
NODE2 ←INPUT(I,J)
JNFT ← 0
if NODE1 ≠ 0 then
   begin
   IFT(K) ← JFT (NODE1,K) for all K
   INFT ← NFTJ(NODE1)
   end;
   Repeat
   if IFT(K) ≠ JFT(NODE2,L) then
```

```
      begin
      if L > NFTJ (NODE2) then
         begin
         increment K
         L <- 1
         end;
      else
         increment L

   else
      begin
      increment JNFT
      IFT(JNFT) <- JFT (NODE2,L)
      increment K
      L <- 1
      end;
   until K = INFT
   INFT <- JNFT
   NODE1 <- 0
until J = ICNT(I)
Copy from IFT and INFT to FT and NFT respectively
Return
```

5. Procedure Comple -

This procedure evaluates the complement of a given fault list.

```
T ← 0
Repeat
if INPUT (I,G) > T then
      T ← INPUT(I,G)
      increment G
until G= ICNT(I)
Initialize CFT(NOE,K)to contain all faults upto 2*T
NCFT(NDE) ← 2*T
K ← 1
Repeat
if FT (NDE,L)=CFT(NDE,K)then
   begin
   decrement NCFT(NDE)
   M ← K
     Repeat
     CFT (NDE,M) ← CFT(NDE,M+1)
     increment M
     until M = NCFT(NDE)
   increment L
   K ← 1
   end;
else
   if K ≤ ICNT (NDE) then
   increment K
until L = NFT(NDE)
Return.
```

6. Procedure COMPØ/COMP1 -

This procedure creates the temporary fault list arrays JFT. for the input nodes of a gates. It call the procedure COMPLE

```
K <- 1
Repeat
NDE <- INPUT(I,K)
if TVALUE (NDE) = 0(1) then
   begin
   call COMPLE
   A <- 1
   Repeat
   JFT (NDE,A)<- CFT(NDE,A)
   increment A
   until A = NCFT(NDE)
   NFTJ(NDE)<- NCFT(NDE)
   end,
else
   begin
   A<- 1
   Repeat
   JFT(NDE,A)<- CFT(NDE,A)
   increment A
   until A = NFT(NDE)
   NFTJ(NDE)<- NFT (NDE)
   end;
   increment K
until K = ICNT(K)
Return.
```

3.4 CRITICAL PATH TRACING

This section deals with the implementation philosophy of critical path tracing. The basic algorithm followed is the same as shown in section 2.5.

3.4.1 Data structures

This method requires more information about the circuit from the preprocessing step than that provided by the preprocessing discussed in section 3.1. So additional information needs to be specified in the circuit description about the number of fanout free regions (cones) in the circuit and the inputs and outputs from each cone. This portion of the circuit is called cone description.

The cone description is given in a manner similar to the gate description given earlier. The general record format is <cone no.>/<inputs>/<outputs>

The inputs to a care can be either primary inputs or outputs of fanout blocks. The output of a cone is either a primary output of the circuit or the stem of a fanout block

ICON(I) has been used to give the number of inputs of the ith cone. CINPUT gives the inputs and COUT the output of the care.

Since critical path tracing is a recursive procedure so at each node a variable is needed to identify the gate to which the node is input/output and the care to which belongs. This is done by INODE, ONODE and CONE respectively.

As recursion is not a standard feature of FORTRAN so a 2nd order subscripted variable J(K(I)) has been used used as a pointer to the level of the recursion.

Three boolean variables have been used for each node in the circuit. These are STINPT, PROP and CRIT. STINPT gives information about the sensitivity of a node. PROP is set if the node propogate the effect of the fault under consideration. CRIT gives the criticality of a node. Another variable EXTD has been used to identify cones that have already been processed.

3.4.2 Algorithms for Critical path tracing

As in the case of deductive simulation the main program comprises the circuit description. In addition it also preprocesses the circuit to get information about the fanout free regions of the circuit. The flow of the program is given below.

1. Extension of preprocessing for cone description.

   The following segment is executed only if the circuit has fanout.

   i (a) Read the number of cones in NOCO

   (b) Read the cone description CONDES (I,J), each row giving description of one cone.

   ii. Create CONCON similar to CON in section 3.1.2.

   iii. Create ICON, CINPUT, COUT similarly as ICNT, INPUT and OUT we created in section 3.1.2.

(iv) In addition, INODE, ONODE, OCONE and CONE are oreated as follows.

```
I(- 1
Repeat
K(- 0
  Repeat
  increment K
  L(- 1
     Repeat
     if I = INPUT(K,L) then
        INODE(I)(- K
     until L = ICNT(K)
  L (- 1
     Repeat
     if I = OUT(K,L) then
         ONODE(I)(- K
     until L = OCNT(K)
  Until K=total no.of gates
  Repeat
     Repeat
     if I=CINPUT(K,L) then
         OCONE(I)(- K
     until all input to ith cone are considered
  until all cones are considered
Until all nodes are considered
CONO(- 1
Repeat
NDE(- COUT(CONO)
increment NC(NDE)
CONE(NDE,NC(NDE))(- CONO
I(- 1
HK(I)(- ONODE(NDE)
JJ(HK(I))(- 1
```

```
Repeat
NDE ← INPUT (HK(I),JJ(HK(I))).
increment NC(NDE)
CONE (NDE, NC(NDE)) ← CONO
if NDE ≠ CINPUT (CONO,L) then
      begin
      if L<ICON(CONO) then
        increment L
      else
        begin
        increment I
        L ← 1
        HK(I) ← ONODE(NDE)
        JJ(HK(I)) ←1
        end,
      end
   else
      begin
      if JJ(HK(I)) <ICNT(I) then
         increment JJ(HK(I))
      else
         begin
         if I > 1 then
           decrement I
           L ← 1
         end,
      until I=1 and L =1
until all cones are considered.
```

2. SIMU : This routine controls the fault free simulation of the circuit. On completion of fault free simulation it hands over the control to TRACE. In all other aspects the routine is similar to that in deductive simulation.

3. Routines for gates: The routines for the gates perform the following functions.

   i. true value evaluation of the output of the gate

   ii. Recognize sensitive inputs of the gate as shown in table and set STINPT for them.

   For XOR/XNOR gates all inputs are sensitive while for NOT/CONT the input is always sensitive.

   iii. Assign the level of the output node of the gate as maximum level of input +1.

NO OF INPUTS HAVING CONTROLLING VALUE	SENSITIVITY
0	all inputs are sensitive
1	the input having controlling value is sensitive
>1	no. input is sensitive

TABLE 3.3.

4. Procedure TRACE: This procedure controls the whole process of identification of critical paths in the circuit. It assumes that fault free simulation has already been performed along with assignment of sensitive inputs and levels of the nodes:

Repeat the following steps for all primary outputs.

1. Initialization

```
CRIT(I) <- 0
EXTD(I) <- 0        for all nodes
PROP(I) <- 0
```

2. Store the current node in NDE

```
CRIT (NDE) <- 1
CALL EXTEND
```

3. While the no. of stems to check ≠ 0, repeat the following

```
begin
NDE <- highest level stem to be checked
decrement no. of stems to be checked.
CALL critic
if CRIT(NDE)=1 then
   CALL EXTEND
end.
```

4. Repeat

```
if CRIT(I)=1 then
   increment NFT(output)
   FT(output, NFT) <- FG(output)
```

until all nodes are considered.

5. Return

5. Procedure EXTEND

This procedure is a recursive tree traversal procedure to identify critical nodes within a fanout free region starting with its output. The steps of the algorithm for this procedure are given below. Since FORTRAN does not have the recursion facility so it has been implemented by using a 3rd order variable J(K(I)).

1. initialization

I ← 0

CONO ← cone no. of the present fanout free region

EXTO (CONO) ← 1

2. Increment I

K(I) ← gate no. to which NDE is output

if K(I) = 0 then

go to step 5

3. If NDE = any input of the present cone then

begin

if NDE is output of a continuity block then

Add input of the continuity block to stems to check

go to step 5

end,

else

begin

J(K(I)) ← 0

go to step 4

end;

4. Increment J(K(I))

NDE<- input no. J(K(I)) of gate no. K(I)
if NDE is sensitive then
NDE<- critical
go to step 1
end;
else
begin
if I > 1 then
decrement I
end;

5. if J(K(I)) < no. of inputs to gate no. K(I) then
go to step 4
if I > 1 then
decrement I
repeat step 5

6. Return

6. Procedure Critio

This procedure is for the stem analysis of stems to be checked. Here the concept of capture time as explained in section 2.4. The steps for the algorithm are enumerated below.

1. Initialization
NFRONT<- 0
PROP(NDE)<- 1
CGTE<- gate no. of the gate to which node is input

2. Repeat this step for all output nodes of CGTE
begin
if any cone to which the node belongs is extended then
if the node is not already in the frontier then
include the node in frantier
end;

3. Find the lowest level gate in the frontier.

   Remove it from the frontier and store its o/p in FTYNOD
   decrement NFRONT
   FTY <- gate no. to which FTYNOD is input

4. if NFRONT = 0 then
   begin
   call PROPGT
   if PROP/output of FTY)=1 then
   CGTE = FTY

   go to step 3
   end;

5. Call PROPGT

   if PROP (output of FTY)=1 then
   if CRIT (FTYNOD)=1 then
   CRIT(NDE)=1

   Return

7. Procedure Propogate : This procedure is to check whether faults on the FTYNOD are propogated by FTY. If yes, then PROP(OUT(FTY,1)=1 else PROP (OUT(FTY,1))=0. This procedure uses the principle of implicit fault effect propogation discussed is section 2.4. The steps for the procedure are outlined below.

   1. if FTYNOD is sensitive then

      Repeat for all inputs of FTY
      begin
      if the input is not sensitive but propogates faults then
      go to step 4
      end;

      go to step 3.

2. if FTYNOD does not have the controlling value then
   go to step 4.

   begin
   if PROP(I)=1 only for those inputs of FTY which
   have controlling value of the gate then
   go to step 3.
   go to step 4.

3. PROP(I)<- 1 for all outputs of FTY
   Return

4. PROP(I)<- 0 for all outputs of FTY
   Return.

3.5 STATISTICAL FAULT ANALYSIS

This technique involves the calculation of probabilities at the various nodes of the circuit. For this initially fault free simulation is done from which additional data at the various nodes is collected. For the fault free simulation the date structures created in preprocessing are used.

ONECNT, ZERCNT and SENCNT contain the data collected from fault simulation. They give the one-count, zero-count and the sensitization count for each node in the circuit. From these CONE, CZERO and S are calculated for each node. There are one-controllability, zero- controllability and the sensitization probability. These in turn are used to calculate the zero (BZERO) and are observabilities (BONE).

The probability of detection of a s-a-$\emptyset$ fault is given by DONE and s-a-1 fault by DZERO. These are obtained by the product of the controllability and observability. TP,XX, and FF are intermediate structures for the evaluation of FC which gives the fault coverage.

3.5.2 Algorithms for Statistical Fault Analysis

This program uses the circuit modelling program of section 3.1. without any modifications. Besides the fault free simulation this method involves calculations which are performed by the routine SIMU which also controls the fault free simulation. The routine gives the results in the form of fault coverage and C.P.U. time required. The initial steps are same as in deductive simulation. The additions are given below.

```
Repeat
Initialization
BONE -1
BZERO-1        } for all primary outputs

    Repeat
    if device type =1 then
       Repeat
       BONE(INP)  <- BONE(OUT)*CONE(OUT)/CONE(INP)
       until all inputs are considered.
    if device type = 2 then
       Repeat
       BONE(INP) <- BONE(OUT)* (S(INP)-CZERO(OUT))/CONE(INP)
       BZERO(INP)<- BZERO(OUT)*CZERO(OUT)/CZERO(INP)
       Until all inputs are considered.
    if device type = 3 then
       Repeat
       BONE(INP) <- BZERO(OUT)*CZERO(OUT)/CONE(INP)
       BZERO(INP)<- BONE(OUT)*(S(INP)-CZERO(OUT))/CZERO(INP)
       until all inputs are considered
```

```
if device type = 4then
   Repeat
   BONE(INP)<- BZERO(OUT)*(S(SNP)-CONE(OUT))/CONE(INP)
   BZERO(INP)<- BONE(OUT) *CONE(OUT)/CZERO(INP)
   Until all inputs are considered.
if device type =5,6 or 7 then
   BONE(INP)<- BONE(OUT)
   BZERO(INP)<- BZERO(OUT)
if device type = 8 the
   B1CONT<- ø
   BøCONT<- ø
   Repeat
   B1CONT<- B1CONT + BONE(OUT)
   BøCONT<- BøCONT+ BZERO(OUT)
   until all outputs are considered
   BONE (INP)<- B1CONT/OCNT
   BZERO(INP)<- BøCONT/OCNT
until all gates are considered
   Repeat
   DONE(I)<- BZERO(I) * CZERO(I)
   DZERO(I)<- BONE(I) * CONE (I)
   WW(2*I)<- W(DONE(I),XIN)
   WW(2*I-1)<- W(DZERO(I), XIN)
   TP(2*I)<-((1.0 -DONE(I))** XIN)/WW(2*I)
   TP(2*I-1)<- ((1.0-DZERO(I))** XIN)/WW(2*I-1)
   XX(2*I, subset no.) <- 1.0 - TP(2*I)
   XX(2*I-1,subset no.)<- 1.0- TP(2*I-1)
   Until all nodes are considered
Until all subsets are considered.
```

```
Repeat
INTER(1)<- ((1.0-XX(KF,1))** XIN)/W(XX(KF,1),XIN)
FF (for ,1)<- 1.0 - INTER(1)
   Repeat
   INTER (Sn)<- INTER( Sn-1) * (1.0-XX(fn,Sn))**XIN
   FF(fn,Sn)<- 1.0 -INTER (Sn)
   Until all subsets are considered
Until all faults are oonsidered
Repeat
   Repeat
   IFC (Sn)<- IFC (Sn) +FF( fn,Sn)
   Until all faults are considered
FC(Sn)<- IFC(Sn)/No. of faults
Until all subsets are considered.
```

PROGRAM LISTING

The listings of all the programs developed has been given in the Appendix.

CHAPTER - 4

R E S U L T S

4.1 RESULTS AND CONCLUSIONS

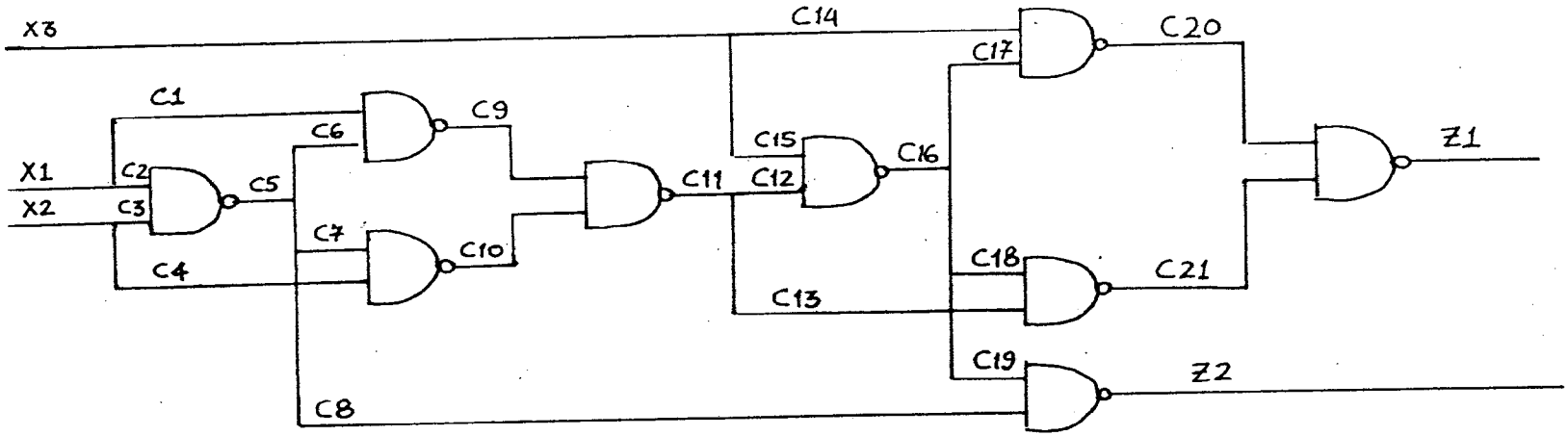
The techniques considered are - Parallel simulation, deductive simulation, critical path tracing and statistical fault Analysis. They have been implemented on the DECSYSTEM-2050. Five different examples have been considered for each of the methods implemented.

The results for all the examples are attached in this chapter. The performance of all these techniques with respect to C.P.U. time has been obtained for all the examples. Also the fault detection information in terms of the faults detected/ fault coverage has been obtained.

The circuits considered are given in Figures 4.1 to 4.5. For each of the circuits the preprocessing information which includes- circuit description, node coding connectivity matrix and fault coding has also been included.

The results in terms of the C.P.U. time requirements have been summarized in table 4.1. It is rather difficult to draw any discernable conclusions from this table, so it has been analyzed in terms of certain circuit parameters.

The parameters on which the results depend have been identified as- the circuit size i.e. the number of gates in the circuit, the faults simulated and the amount of fanout



circuit - 1

Figure - 4.1

NO. OF GATES =15
CIRCUIT DESCRIPTION

G1/X1/C1,C2/CONT/
G2/X2/C3,C4/CONT/
G3/C2,C3/C5/NAND/
G4/C5/C6,C7,C8/CONT/
G5/C1,C6/C9/NAND/
G6/C7,C4/C10/NAND/
G7/C9,C10/C11/NAND/
G8/C11/C12,C13/CONT/
G9/X3/C14,C15/CONT/
G10/C15,C12/C16/NAND/
G11/C16/C17,C18,C19/CONT/
G12/C14,C17/C20/NAND/
G13/C18,C13/C21/NAND/
G14/C20,C21/Z1/NAND/
G15/C19,C8/Z2/NAND/

[illegible] DESCRIPTION
==========
[illegible]. CONES = 5

[illegible]/C2,C3/C5/
[illegible]/C1,C6,C7,C4/C11/
[illegible]/C15,C14/C16/
[illegible]/C14,C17,C18,C19/Z1/
[illegible]/C1,C6,C7,C4,C8,C15/Z2/

NODE CODING

SYMBOLIC CODE	X_1	X_2	X_3	C_1	C_2	C_3	C_4	C_5	C_6	C_7
NUMERIC CODE	1	2	3	4	5	6	7	8	9	10
SYMBOLIC CODE	C_8	C_9	C10	C11	C12	C13	C14	C15	C16	C17
NUMERIC CODE	11	12	13	14	15	16	17	18	19	20
SYMBOLIC CODE	C18	C19	C20	C21	Z_1	Z_2				
NUMERIC CODE	21	22	23	24	25	26				

CONNECTIVITY MATRIX

```
   0000000011111111112222222
   1234567890123456789012345
01     8
02      8
03               8
04           3
05        3
06        3
07          3
08         8
09          3
10           3
11                        3
12            3
13            3
14             8
15                 3
16                    3
17                  3
18                 3
19                  8
20                  3
21                    3
22                      3
23                     3
24                     3
25
26
```

FAULT CODING

NODE NO.	1	2	3	4	5	6	7	8	9	10
S-A-0	1	3	5	7	9	11	13	15	17	19
S-A-1	2	4	6	8	10	12	14	16	18	20

NODE NO.	11	12	13	14	15	16	17	18	19	20
S-A-0	21	23	25	27	29	31	33	35	37	39
S-A-1	22	24	26	28	30	32	34	36	38	40

NODE NO.	21	22	23	24	25	26
S-A-0	41	43	45	47	49	51
S-A-1	42	44	46	48	50	52

RESULTS FOR DEDUCTIVE SIMULATION
****

TESTS SIMULATED
==========
T 1 : 1 1 1
T 2 : 0 0 0
T 3 : 1 0 1
T 4 : 0 1 0

FAULTS DETECTION INFORMATION
==========

FAULTS DETECTED FOR T 1

FAULT NO.S DETECTED AT Z 1 ARE = 5, 33, 1, 3, 9, 11, 16, 18, 20, 23, 25, 28, 30, 37, 39, 46, 49,

FAULT NO.S DETECTED AT Z 2 ARE = 22, 51,

FAULTS DETECTED FOR T 2

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 6, 34, 45, 2, 4, 8, 14, 32,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 37, 43, 15, 21, 52,

FAULTS DETECTED FOR T 3

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 7, 12, 15, 17, 24, 27, 29, 35, 42,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 5, 35, 1, 7, 17, 24, 27, 29,

FAULTS DETECTED FOR T 4

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 10, 13, 19, 26, 31, 36, 41, 48,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 6, 30, 2, 10,
AVERAGE FAULT LIST LENGTH : 4.6154
==========

TIME TAKEN TO SIMULATE 4 TESTS = 171 MS
==========

```
RESULTS FOR PARALLEL SIMULATION
*******************************

FAULTS DETECTED FOR T 1
-------------------------------

FAULT NO.S DETECTED AT Z  1 ARE =  1,  3,  5,  9, 11, 16, 18, 20, 23, 25,
 28, 30, 33, 37, 39, 46, 49,

FAULT NO.S DETECTED AT Z  2 ARE = 22, 51,

FAULTS DETECTED FOR T 2
-------------------------------

ADDITIONAL FAULTS DETECTED AT Z  1 ARE =  2,  4,  6,  8, 14, 32, 34, 45,

ADDITIONAL FAULTS DETECTED AT Z  2 ARE = 15, 21, 37, 43, 52,

FAULTS DETECTED FOR T 3
-------------------------------

ADDITIONAL FAULTS DETECTED AT Z  1 ARE =  7, 12, 15, 17, 24, 27, 29, 35,
 42,

ADDITIONAL FAULTS DETECTED AT Z  2 ARE =  1,  5,  7, 17, 24, 27, 29, 35,

FAULTS DETECTED FOR T 4
-------------------------------

ADDITIONAL FAULTS DETECTED AT Z  1 ARE = 10, 13, 19, 26, 31, 36, 41, 48,

ADDITIONAL FAULTS DETECTED AT Z  2 ARE =  2,  6, 10, 36,

TIME TAKEN TO SIMULATE  4 TESTS =   41 MS
```

RESULTS FOR CRITICAL PATH TRACING
****

FAULTS DETECTED FOR T 1

FAULT NO.S DETECTED AT Z 1 ARE = 1, 3, 5, 9, 11, 16, 18, 20, 23, 25, 28, 30, 33, 37, 39, 46, 49,

FAULT NO.S DETECTED AT Z 2 ARE = 22, 51,

FAULTS DETECTED FOR T 2

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 2, 4, 6, 8, 14, 32, 34, 45,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 15, 21, 37, 43, 52,

FAULTS DETECTED FOR T 3

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 7, 12, 15, 17, 24, 27, 29, 35, 42,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 1, 5, 7, 17, 24, 27, 29, 35,

FAULTS DETECTED FOR T 4

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 10, 13, 19, 26, 31, 36, 41, 48,

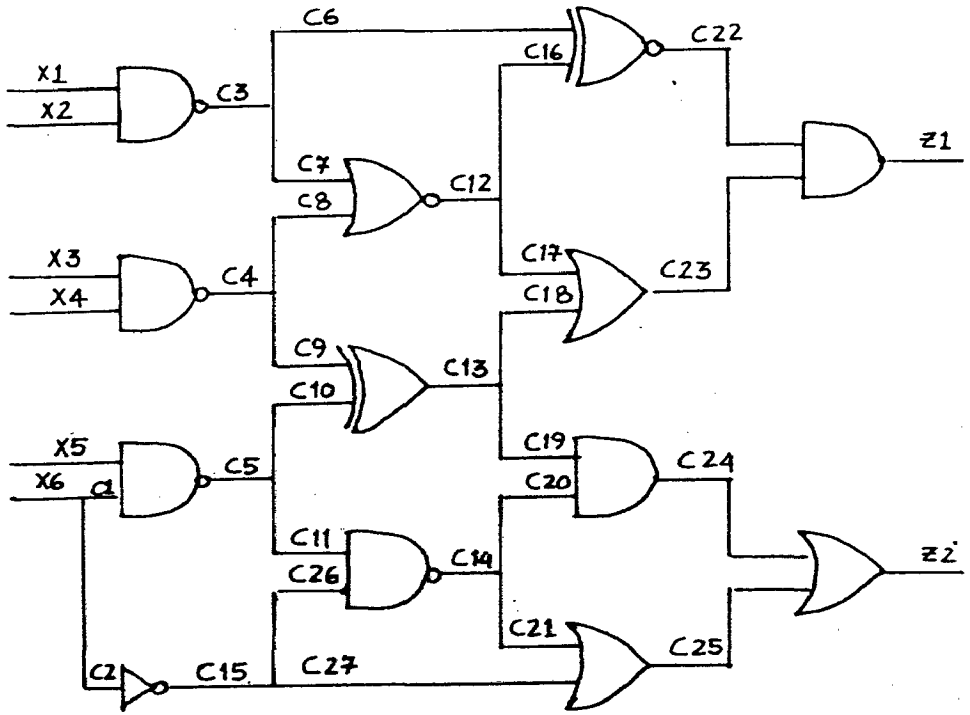
ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 2, 6, 10, 36,

TIME TAKEN TO SIMULATE 4 TESTS = 100 MS

RESULTS FOR STAFAN
****

NODE	C0	C1	B1	B0	D1	D0
1	.5000	.5000	.2438	.2284	.1142	.1219
2	.5000	.5000	.2438	.2284	.1142	.1219
3	.5000	.5000	.4444	.4444	.2222	.2222
4	.5000	.5000	.2222	.2963	.1481	.1111
5	.5000	.5000	.2654	.1605	.0802	.1327
6	.5000	.5000	.2654	.1605	.0802	.1327
7	.5000	.5000	.2222	.2963	.1481	.1111
8	.2500	.7500	.3210	.5309	.1327	.2407
9	.2500	.7500	.1481	.2963	.0741	.1111
10	.2500	.7500	.1481	.2963	.0741	.1111
11	.2500	.7500	.6667	1.0000	.2500	.5000
12	.2500	.7500	.2963	.4444	.1111	.2222
13	.2500	.7500	.2963	.4444	.1111	.2222
14	.5000	.5000	.4444	.4444	.2222	.2222
15	.5000	.5000	.3889	.2222	.1111	.1944
16	.5000	.5000	.5000	.6667	.3333	.2500
17	.5000	.5000	.5000	.6667	.3333	.2500
18	.5000	.5000	.3889	.2222	.1111	.1944
19	.2500	.7500	.4444	.7778	.1944	.3333
20	.2500	.7500	.3333	.6667	.1667	.2500
21	.2500	.7500	.3333	.6667	.1667	.2500
22	.2500	.7500	.6667	1.0000	.2500	.5000
23	.2500	.7500	.6667	1.0000	.2500	.5000
24	.2500	.7500	.6667	1.0000	.2500	.5000
25	.5000	.5000	1.0000	1.0000	.5000	.5000
26	.5000	.5000	1.0000	1.0000	.5000	.5000

RACTION OF FAULTS DETECTED = 0.9283
IME TAKEN TO ESTIMATE FAULT COVERAGE = 82



circuit-2
Figure-4.2

NO. OF GATES =21
CIRCUIT DESCRIPTION

```
G1/X1,X2/C3/AND/
G2/X3,X4/C4/AND/
G3/X6/C1,C2/CONT/
G4/X5,C1/C5/AND/
G5/C3/C6,C7/CONT/
G6/C4/C8,C9/CONT/
G7/C5/C10,C11/CONT/
G8/C2/C15/NOT/
G9/C15/C26,C27/CONT/
G10/C7,C8/C12/NOR/
G11/C9,C10/C13/XOR/
G12/C26,C11/C14/NAND/
G13/C12/C16,C17/CONT/
G14/C13/C18,C19/CONT/
G15/C14/C20,C21/CONT/
G15/C6,C16/C22/XNOR/
G17/C17,C18/C23/OR/
G18/C19,C20/C24/AND/
G19/C21,C27/C25/OR/
G20/C22,C23/Z1/AND/
G21/C24,C25/Z2/OR/
```

CONE DESCRIPTION

NO. OF CONES = 9

```
T1/X1,X2/C3/
T2/X3,X4/C4/
T3/X5,X6/C5/
T4/C2/C15/
T5/C7,C8/C12/
T6/C9,C10/C13/
T7/C11,C26/C14/
T8/C6,C16,C17,C18/Z1/
T9/C19,C20,C21,C27/Z2/
```

NODE CODING

SYMBOLIC CODE	X_1	X_2	X_3	X_4	X_5	X_6	C_1	C_2	C_3	C_4
NUMERIC CODE	1	2	3	4	5	6	7	8	9	10
SYMBOLIC CODE	C_5	C_6	C_7	C_8	C_9	C10	C11	C12	C13	C14
NUMERIC CODE	11	12	13	14	15	16	17	18	19	20
SYMBOLIC CODE	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24
NUMERIC CODE	21	22	23	24	25	26	27	28	29	30
SYMBOLIC CODE	C25	C26	C27	Z_1	Z_2					
NUMERIC CODE	31	32	33	34	35					

CONNECTIVITY MATRIX

```
   0000000001111111112222222222333333
   1234567890123456789012345678901234 5
01          1
02          1
03           1
04           1
05            1
06       8
07            1
08                    7
09              8
10                 8
11                    8
12                             6
13                       4
14                       4
15                        5
16                        5
17                         3
18                            8
19                               8
20                                  8
21                                     8
22                                  6
23                                   2
24                                   2
25                                    1
26                                    1
27                                     2
28                                        1
29                                        1
30                                          2
31                                          2
32                         3
33                                     2
34
35
```

FAULT CODING

NODE NO.	1	2	3	4	5	6	7	8	9	10
S-A-0	1	3	5	7	9	11	13	15	17	19
S-A-1	2	4	6	8	10	12	14	16	18	20
NODE NO.	11	12	13	14	15	16	17	18	19	20
S-A-0	21	23	25	27	29	31	33	35	37	39
S-A-1	22	24	26	28	30	32	34	36	38	40
NODE NO.	21	22	23	24	25	26	27	28	29	30
S-A-0	41	43	45	47	49	51	53	55	57	59
S-A-1	42	44	46	48	50	52	54	56	58	60
NODE NO.	31	32	33	34	35					
S-A-0	61	63	65	67	69					
S-A-1	62	64	66	68	70					

RESULTS FOR DEDUCTIVE SIMULATION
****

TESTS SIMULATED
==========

```
T 1 :  1 1 1 1 1 1
T 2 :  0 0 0 0 0 0
T 3 :  0 1 0 1 0 1
T 4 :  1 0 1 0 1 0
T 5 :  1 1 1 1 0 0
```

FAULTS DETECTION INFORMATION
==========

FAULTS DETECTED FOR T 1

FAULT NO.S DETECTED AT Z 1 ARE = 24, 26, 28, 35, 43, 56, 68,

FAULT NO.S DETECTED AT Z 2 ARE = 64, 39, 69,

FAULTS DETECTED FOR T 2

NO ADDITIONAL FAULTS DETECTED AT Z 1

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 61,

FAULTS DETECTED FOR T 3

NO ADDITIONAL FAULTS DETECTED AT Z 1

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 53,

FAULTS DETECTED FOR T 4

NO ADDITIONAL FAULTS DETECTED AT Z 1

NO ADDITIONAL FAULTS DETECTED AT Z 2

FAULTS DETECTED FOR T 5

NO ADDITIONAL FAULTS DETECTED AT Z 1

NO ADDITIONAL FAULTS DETECTED AT Z 2

AVERAGE FAULT LIST LENGTH : 2.5429
==========

TIME TAKEN TO SIMULATE 5 TESTS = 212 MS
==========

RESULTS FOR PARALLEL SIMULATION
****

FAULTS DETECTED FOR T 1

FAULT NO.S DETECTED AT Z 1 ARE = 24, 26, 28, 36, 43, 56, 68,

FAULT NO.S DETECTED AT Z 2 ARE = 39, 64, 69,

FAULTS DETECTED FOR T 2

NO ADDITIONAL FAULTS DETECTED AT Z 1

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 61,

FAULTS DETECTED FOR T 3

NO ADDITIONAL FAULTS DETECTED AT Z 1

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 53,

FAULTS DETECTED FOR T 4

NO ADDITIONAL FAULTS DETECTED AT Z 1

NO ADDITIONAL FAULTS DETECTED AT Z 2

FAULTS DETECTED FOR T 5

NO ADDITIONAL FAULTS DETECTED AT Z 1

NO ADDITIONAL FAULTS DETECTED AT Z 2

TIME TAKEN TO SIMULATE 5 TESTS = 53 MS

RESULTS FOR CRITICAL PATH TRACING
****

FAULTS DETECTED FOR T 1

FAULT NO.S DETECTED AT Z 1 ARE = 24, 26, 28, 36, 43, 56, 68,

FAULT NO.S DETECTED AT Z 2 ARE = 39, 64, 69,

FAULTS DETECTED FOR T 2

NO ADDITIONAL FAULTS DETECTED AT Z 1

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 61,

FAULTS DETECTED FOR T 3

NO ADDITIONAL FAULTS DETECTED AT Z 1

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 53,

FAULTS DETECTED FOR T 4

NO ADDITIONAL FAULTS DETECTED AT Z 1

NO ADDITIONAL FAULTS DETECTED AT Z 2

FAULTS DETECTED FOR T 5

NO ADDITIONAL FAULTS DETECTED AT Z 1

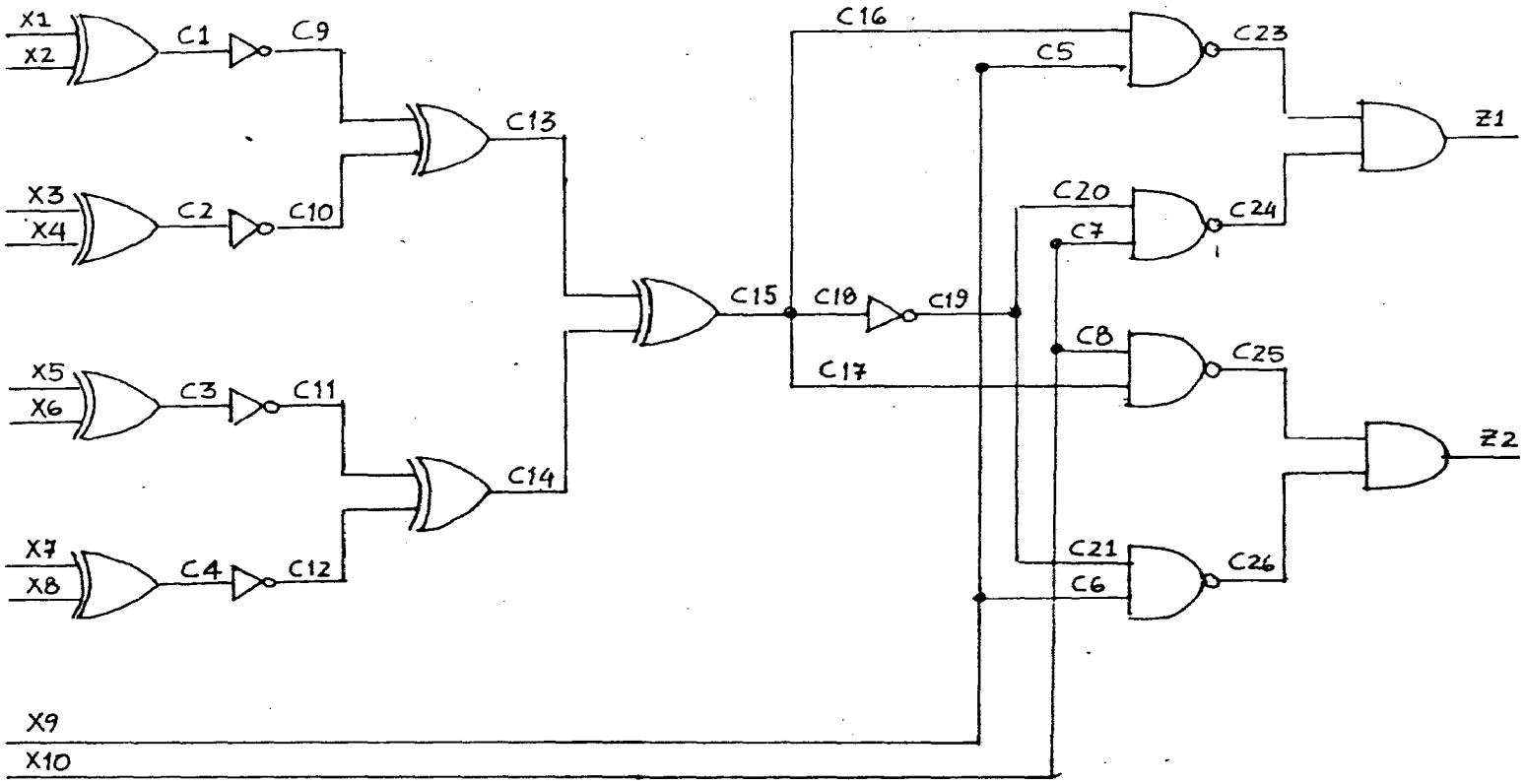
NO ADDITIONAL FAULTS DETECTED AT Z 2

TIME TAKEN TO SIMULATE 5 TESTS = 114 MS

RESULTS FOR STAFAN
****

NODE	C0	C1	B1	B0	D1	D0
1	.5000	.5000	.0000	.1875	.0938	.0000
2	.5000	.5000	.0000	.1875	.0938	.0000
3	.5000	.5000	.0000	.0000	.0000	.0000
4	.5000	.5000	.0000	.0000	.0000	.0000
5	.5000	.5000	.0000	.0417	.0208	.0000
6	.5000	.5000	.0000	.0521	.0260	.0000
7	.5000	.5000	.0000	.0417	.0208	.0000
8	.5000	.5000	.0000	.0625	.0313	.0000
9	.7500	.2500	.0000	.3750	.2813	.0000
10	.7500	.2500	.0000	.0000	.0000	.0000
11	.7500	.2500	.0000	.0833	.0625	.0000
12	.7500	.2500	.0000	.7500	.5625	.0000
13	.7500	.2500	.0000	.0000	.0000	.0000
14	.7500	.2500	.0000	.0000	.0000	.0000
15	.7500	.2500	.0000	.0000	.0000	.0000
16	.7500	.2500	.0000	.0000	.0000	.0000
17	.7500	.2500	.0000	.1667	.1250	.0000
18	.2500	.7500	.0000	.0000	.0000	.0000
19	1.0000	.0000	.0000	.0000	.0000	.0000
20	.0000	1.0000	.2500	.0000	.0000	.2500
21	.5000	.5000	.0000	.0625	.0313	.0000
22	.2500	.7500	.0000	.0000	.0000	.0000
23	.2500	.7500	.0000	.0000	.0000	.0000
24	1.0000	.0000	.0000	.0000	.0000	.0000
25	1.0000	.0000	.0000	.0000	.0000	.0000
26	.0000	1.0000	.0000	.0000	.0000	.0000
27	.5000	1.0000	.5000	.0000	.0000	.5000
28	1.0000	.0000	.0000	.7500	.7500	.0000
29	.2500	.7500	.0000	.0000	.0000	.0000
30	1.0000	.0000	.0000	.0000	.0000	.0000
31	.0000	1.0000	1.0000	.0000	.0000	1.0000
32	.5000	.5000	.0000	.1250	.0625	.0000
33	.5000	.5000	.0000	.0000	.0000	.0000
34	1.0000	.0000	1.0000	1.0000	1.0000	.0000
35	.0000	1.0000	1.0000	1.0000	.0000	1.0000

FRACTION OF FAULTS DETECTED = 0.1922
TIME TAKEN TO ESTIMATE FAULT COVERAGE = 93



Circuit - 3
Figure-43

NO. OF GATES =22
CIRCUIT DESCRIPTION

```
G1/X1,X2/C1/XOR/
G2/X3,X4/C2/XOR/
G3/X5,X6/C3/XOR/
G4/X7,X8/C4/XOR/
G5/X9/C5,C6/CONT/
G6/X10/C7,C8/CONT/
G7/C1/C9/NOT/
G8/C2/C10/NOT/
G9/C3/C11/NOT/
G10/C4/C12/NOT/
G11/C9,C10/C13/XOR/
G12/C11,C12/C14/XOR/
G13/C14,C13/C15/XOR/
G14/C15/C16,C17,C18/CONT/
G15/C18/C19/NOT/
G16/C19/C20,C21/CONT/
G17/C16,C5/C22/NAND/
G18/C17,C8/C24/NAND/
G19/C20,C7/C23/NAND/
G20/C6,C21/C25/NAND/
G21/C22,C23/Z1/AND/
G22/C24,C25/Z2/AND/
```

CON. DESCRIPTION
==========
NO. OF CONES = 4

```
T1/X1,X2,X3,X4,X5,X6,X7,X8/C15/
T2/C18/C19/
T3/C16,C5,C2 ,C7/Z1/
T4/C8,C17,C21,C6/Z2/
```

NODE CODING

SYMBOLIC CODE	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X1
NUMERIC CODE	1	2	3	4	5	6	7	8	9	1(
SYMBOLIC CODE	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C1
NUMERIC CODE	11	12	13	14	15	16	17	18	19	2(
SYMBOLIC CODE	C11	C12	C13	C14	C15	C16	C17	C18	C19	C2
NUMERIC CODE	21	22	23	24	25	26	27	28	29	3(
SYMBOLIC CODE	C21	C22	C23	C24	C25	Z_1	Z_2			
NUMERIC CODE	31	32	33	34	35	36	37			

CONNECTIVITY MATRIX

```
   0000000001111111111222222222233333333
   1234567890123456789012345678901234567
01             5
02             5
03              5
04              5
05               5
06               5
07                5
08                5
09                 8
10                   8
11                     7
12                      7
13                       7
14                        7
15                                  3
16                                   3
17                                  3
18                                   3
19                          5
20                          5
21                           5
22                           5
23                            5
24                            5
25                             8
26                                  3
27                                   3
28                                7
29                                 8
30                                  3
31                                   3
32                                    1
33                                    1
34                                     1
35                                     1
36
37
```

FAULT CODING

NODE NO.	1	2	3	4	5	6	7	8	9	10
S-A-0	1	3	5	7	9	11	13	15	17	19
S-A-1	2	4	6	8	10	12	14	16	18	20

NODE NO.	11	12	13	14	15	16	17	18	19	20
S-A-0	21	23	25	27	29	31	33	35	37	39
S-A-1	22	24	26	28	30	32	34	36	38	40

NODE NO.	21	22	23	24	25	26	27	28	29	30
S-A-0	41	43	45	47	49	51	53	55	57	59
S-A-1	42	44	46	48	50	52	54	56	58	60

NODE NO.	31	32	33	34	35	36	37
S-A-0	61	63	65	67	69	71	73
S-A-1	62	64	66	68	70	72	74

RESULTS FOR DEDUCTIVE SIMULATION
****

TESTS SIMULATED
==========
T 1 : 1 1 1 1 1 1 1 1 1 1
T 2 : 0 0 0 0 0 0 0 0 0 0
T 3 : 1 0 1 0 1 0 1 0 1 0
T 4 : 0 1 0 1 0 1 0 1 0 1

FAULTS DETECTION INFORMATION
==========

FAULTS DETECTED FOR T 1

FAULT NO.S DETECTED AT Z 1 ARE = 19, 33, 56, 57, 59, 66, 72,

FAULT NO.S DETECTED AT Z 2 ARE = 17, 31, 56, 57, 61, 70, 74,

FAULTS DETECTED FOR T 2

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 63, 20, 34, 65, 71,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 67, 18, 32, 69, 73,

FAULTS DETECTED FOR T 3

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 9, 12, 25, 42, 13, 16, 27, 44, 48, 1
  1, 21, 38, 5, 8, 23, 40, 46, 50, 52,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 1, 4, 5, 8, 9, 12, 13, 16, 21, 23
 25, 27, 38, 40, 42, 44, 46, 48, 50,

FAULTS DETECTED FOR T 4

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 2, 3, 6, 7, 10, 11, 14, 15,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 10, 11, 14, 15, 2, 3, 6, 7, 54,
AVERAGE FAULT LIST LENGTH : 8.3514
==========

TIME TAKEN TO SIMULATE 4 TESTS = 270 MS
==========

RESULTS FOR PARALLEL SIMULATION
****

FAULTS DETECTED FOR T 1

FAULT NO.S DETECTED AT Z 1 ARE = 19, 33, 56, 57, 59, 66, 72,

FAULT NO.S DETECTED AT Z 2 ARE = 17, 31, 56, 57, 61, 70, 74,

FAULTS DETECTED FOR T 2

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 20, 34, 63, 65, 71,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 18, 32, 67, 69, 73,

FAULTS DETECTED FOR T 3

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 1, 4, 5, 8, 9, 12, 13, 16, 21, 2
25, 27, 38, 40, 42, 44, 46, 48, 50, 52,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 1, 4, 5, 8, 9, 12, 13, 16, 21, 2
25, 27, 38, 40, 42, 44, 46, 48, 50,

FAULTS DETECTED FOR T 4

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 2, 3, 6, 7, 10, 11, 14, 15,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 2, 3, 6, 7, 10, 11, 14, 15, 54,

TIME TAKEN TO SIMULATE 4 TESTS = 95 MS

RESULTS FOR CRITICAL PATH TRACING
****

FAULTS DETECTED FOR T 1

FAULT NO.S DETECTED AT Z 1 ARE = 19, 33, 56, 57, 59, 66, 72,

FAULT NO.S DETECTED AT Z 2 ARE = 17, 31, 56, 57, 61, 70, 74,

FAULTS DETECTED FOR T 2

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 20, 34, 63, 65, 71,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 18, 32, 67, 69, 73,

FAULTS DETECTED FOR T 3

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 1, 4, 5, 8, 9, 12, 13, 16, 21, 2
25, 27, 38, 40, 42, 44, 46, 48, 50, 52,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 1, 4, 5, 8, 9, 12, 13, 16, 21, 2
25, 27, 38, 40, 42, 44, 46, 48, 50,

FAULTS DETECTED FOR T 4

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 2, 3, 6, 7, 10, 11, 14, 15,

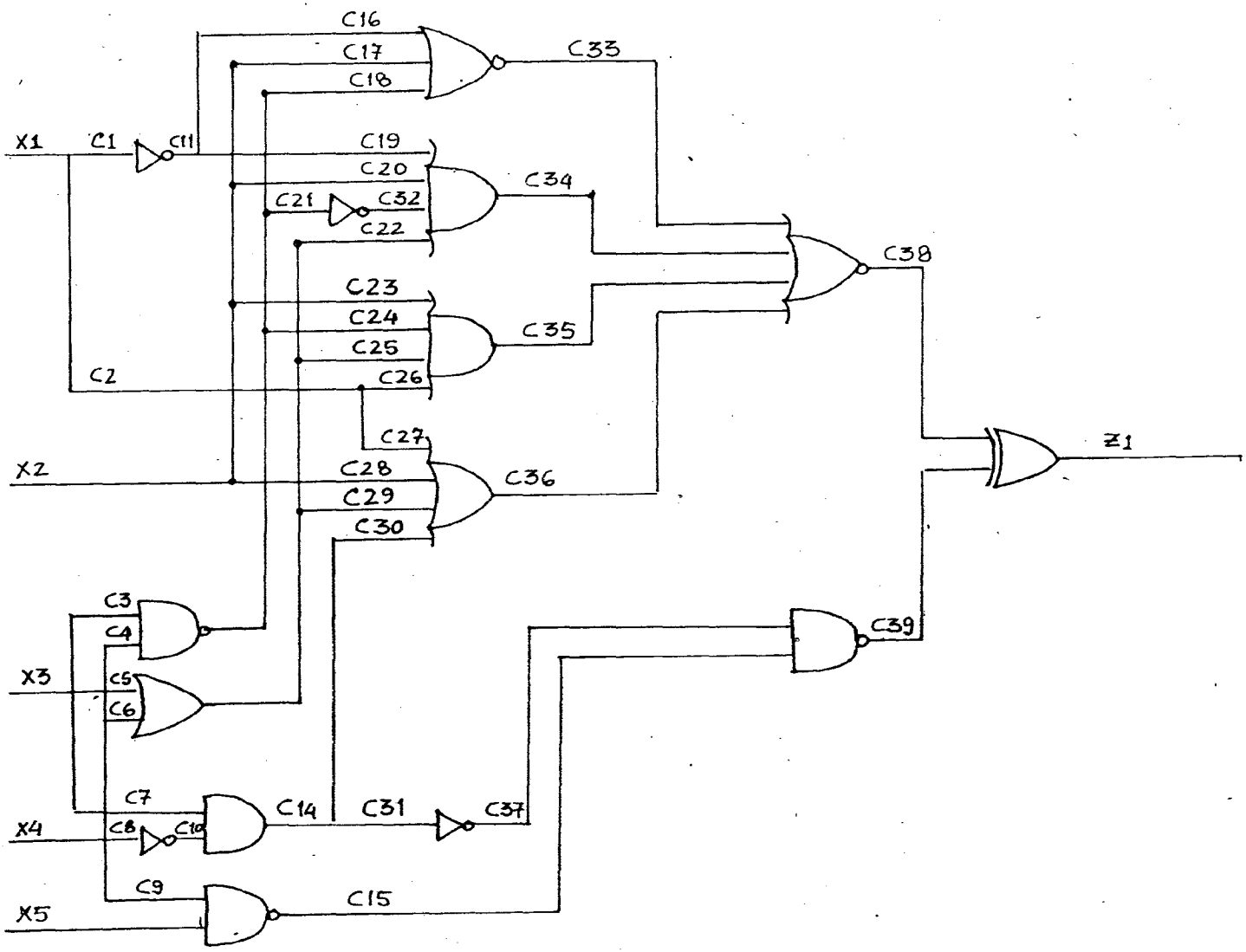
ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 2, 3, 6, 7, 10, 11, 14, 15, 54,

TIME TAKEN TO SIMULATE 4 TESTS = 112 MS

RESULTS FOR STAFAN
****

NODE	C0	C1	B1	B0	D1	D0
1	.5000	.5000	.1667	.1667	.0833	.0833
2	.5000	.5000	.1667	.1667	.0833	.0833
3	.5000	.5000	.1667	.1667	.0833	.0833
4	.5000	.5000	.1667	.1667	.0833	.0833
5	.5000	.5000	.1667	.1667	.0833	.0833
6	.5000	.5000	.1667	.1667	.0833	.0833
7	.5000	.5000	.1667	.1667	.0833	.0833
8	.5000	.5000	.1667	.1667	.0833	.0833
9	.5000	.5000	.5000	.5000	.2500	.2500
10	.5000	.5000	.5000	.5000	.2500	.2500
11	.5000	.5000	.1667	.1667	.0833	.0833
12	.5000	.5000	.1667	.1667	.0833	.0833
13	.5000	.5000	.1667	.1667	.0833	.0833
14	.5000	.5000	.1667	.1667	.0833	.0833
15	.5000	.5000	.0000	.0000	.0000	.0000
16	.5000	.5000	1.0000	1.0000	.5000	.5000
17	.5000	.5000	1.0000	1.0000	.5000	.5000
18	.5000	.5000	.0000	.0000	.0000	.0000
19	.5000	.5000	.1667	.1667	.0833	.0833
20	.5000	.5000	.1667	.1667	.0833	.0833
21	.5000	.5000	.1667	.1667	.0833	.0833
22	.5000	.5000	.1667	.1667	.0833	.0833
23	1.0000	.0000	.1667	.1667	.1667	.0000
24	1.0000	.0000	.1667	.1667	.1667	.0000
25	1.0000	.0000	.1667	.1667	.1667	.0000
26	1.0000	.0000	.0000	.2500	.2500	.0000
27	1.0000	.0000	.0000	.2500	.2500	.0000
28	1.0000	.0000	.5000	.0000	.0000	.0000
29	.0000	1.0000	.5000	.0000	.0000	.5000
30	.0000	1.0000	.5000	.0000	.0000	.5000
31	.0000	1.0000	.5000	.0000	.0000	.5000
32	.0000	1.0000	.5000	.0000	.0000	.5000
33	.5000	.5000	1.0000	1.0000	.5000	.5000
34	.0000	1.0000	.5000	.0000	.0000	.5000
35	.5000	.5000	1.0000	1.0000	.5000	.5000
36	.5000	.5000	1.0000	1.0000	.5000	.5000
37	.5000	.5000	1.0000	1.0000	.5000	.5000

FRACTION OF FAULTS DETECTED = 0.6733
TIME TAKEN TO ESTIMATE FAULT COVERAGE = 103



Circuit-4
Figuve-4.4

```
NO. OF GATES =24
CIRCUIT DESCRIPTION
-------------------
G1/X1/C1,C2/CONT/
G2/X3/C3,C5,C7/CONT/
G3/X4/C4,C6,C8,C9/CONT/
G4/C1/C11/NOT/
G5/C3,C4/C12/NAND/
G6/C5,C6/C13/OR/
G7/C8/C10/NOT/
G8/C9,X5/C15/NAND/
G9/C11/C16,C19/CONT/
G10/C2/C26,C27/CONT/
G11/X2/C17,C20,C23,C28/CONT/
G12/C12/C18,C21,C24/CONT/
G13/C13/C22,C25,C29/CONT/
G14/C7,C10/C14/AND/
G15/C14/C30,C31/CONT/
G16/C21/C32/NOT/
G17/C31/C37/NOT/
G18/C16,C17,C18/C33/NOR/
G19/C19,C20,C32,C22/C34/AND/
G20/C23,C24,C25,C26/C35/AND/
G21/C27,C28,C29,C30/C36/NOR/
G22/C33,C34,C35,C36/C38/NOR/
G23/C37,C15/C39/NAND/
G24/C38,C39/Z1/XOR/
```

```
CONE DESCRIPTION
================
NO. OF CONES = 5

T1/C1/C11/
T2/C3,C4/C12/
T3/C5,C6/C13/
T4/C7,C8/C14/
T5/C16,C17,C18,C19,C20,C32,C22,C23,C24,C25,C26,C27,
C28,C29,C30,C31,C9,X5/Z1/
```

NODE CODING

SYMBOLIC CODE	X_1	X_2	X_3	X_4	X_5	C_1	C_2	C_3	C_4	C_5
NUMERIC CODE	1	2	3	4	5	6	7	8	9	10
SYMBOLIC CODE	C_6	C_7	C_8	C_9	C10	C11	C12	C13	C14	C15
NUMERIC CODE	11	12	13	14	15	16	17	18	19	20
SYMBOLIC CODE	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25
NUMERIC CODE	21	22	23	24	25	26	27	28	29	30
SYMBOLIC CODE	C26	C27	C28	C29	C30	C31	C32	C33	C34	C35
NUMERIC CODE	31	32	33	34	35	36	37	38	39	40
SYMBOLIC CODE	C36	C37	C38	C39	Z_1					
NUMERIC CODE	41	42	43	44	45					

CONNECTIVITY MATRIX

```
  0000000001111111112222222222333333333344444
  1234567890123456789012345678901234567890123
01      8
02                    8
03       8
04        8
05                  3
06                7
07                        8
08                3
09                3
10                 2
11                 2
12                  1
13                7
14                  3
15                 1
16                   8
17                     8
18                        8
19                              8
20                                    3
21                                4
22                                4
23                                4
24                                 1
25                                 1
26                              7
27                                 1
28                                  1
29                                  1
30                                  1
31                                  1
32                                   4
33                                   4
34                                   4
35                                   4
36                                    7
37                                 1
38                                     4
39                                     4
40                                     4
41                                     4
42                                      3
43                                       5
44                                       5
45
```

FAULT CODING

NODE NO.	1	2	3	4	5	6	7	8	9	10
S-A-0	1	3	5	7	9	11	13	15	17	19
S-A-1	2	4	6	8	10	12	14	16	18	20
NODE NO.	11	12	13	14	15	16	17	18	19	20
S-A-0	21	23	25	27	29	31	33	35	37	39
S-A-1	22	24	26	28	30	32	34	36	38	40
NODE NO.	21	22	23	24	25	26	27	28	29	30
S-A-0	41	43	45	47	49	51	53	55	57	59
S-A-1	42	44	46	48	50	52	54	56	58	60
NODE NO.	31	32	33	34	35	36	37	38	39	40
S-A-0	61	63	65	67	69	71	73	75	77	79
S-A-1	62	64	66	68	70	72	74	76	78	80
NODE NO.	41	42	43	44	45					
S-A-0	81	83	85	87	89					
S-A-1	82	84	86	88	90					

RESULTS FOR DEDUCTIVE SIMULATION
****

TESTS SIMULATED
==========
T 1 : 1 1 1 1 1
T 2 : 0 0 0 0 0
T 3 : 1 0 1 0 1
T 4 : 0 1 0 1 0

FAULTS DETECTION INFORMATION
==========

FAULTS DETECTED FOR T 1

FAULT NO.S DETECTED AT Z 1 ARE = 3, 43, 76, 1, 11, 32, 48, 78, 5, 7,
 15, 17, 34, 58, 80, 82, 85, 9, 27, 40, 87, 90,

FAULTS DETECTED FOR T 2

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 2, 4, 6, 8, 14, 20, 22, 24, 36,
 64, 66, 68, 70, 81, 86, 72, 83, 39, 88,

FAULTS DETECTED FOR T 3

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 18, 33, 45, 56, 23, 26, 29, 37, 71,

FAULTS DETECTED FOR T 4

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 16, 51, 74, 62, 10, 89,
AVERAGE FAULT LIST LENGTH : 3.3333
==========

TIME TAKEN TO SIMULATE 4 TESTS = 266 MS
==========

RESULTS FOR PARALLEL SIMULATION
****

FAULTS DETECTED FOR T 1

FAULT NO.S DETECTED AT Z 1 ARE = 1, 3, 5, 7, 9, 11, 15, 17, 27, 32, 34, 40, 43, 48, 58, 76, 78, 80, 82, 85, 87, 90,

FAULTS DETECTED FOR T 2

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 2, 4, 8, 14, 20, 22, 36, 38, 64, 68, 70, 72, 81, 83, 86, 88,

FAULTS DETECTED FOR T 3

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 18, 23, 26, 29, 33, 37, 45, 56, 71,

FAULTS DETECTED FOR T 4

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 10, 16, 51, 62, 74, 89,

TIME TAKEN TO SIMULATE 4 TESTS = 83 MS

RESULTS FOR CRITICAL PATH TRACING
****

FAULTS DETECTED FOR T 1

FAULT NO.S DETECTED AT Z 1 ARE = 1, 3, 5, 7, 9, 11, 15, 17, 27, 32, 34, 41, 43, 48, 59, 76, 78, 80, 82, 85, 87, 90,

FAULTS DETECTED FOR T 2

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 2, 4, 8, 14, 20, 22, 36, 38, 64, 68, 70, 72, 81, 83, 86, 88,

FAULTS DETECTED FOR T 3

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 18, 23, 26, 29, 33, 37, 45, 56, 71,

FAULTS DETECTED FOR T 4

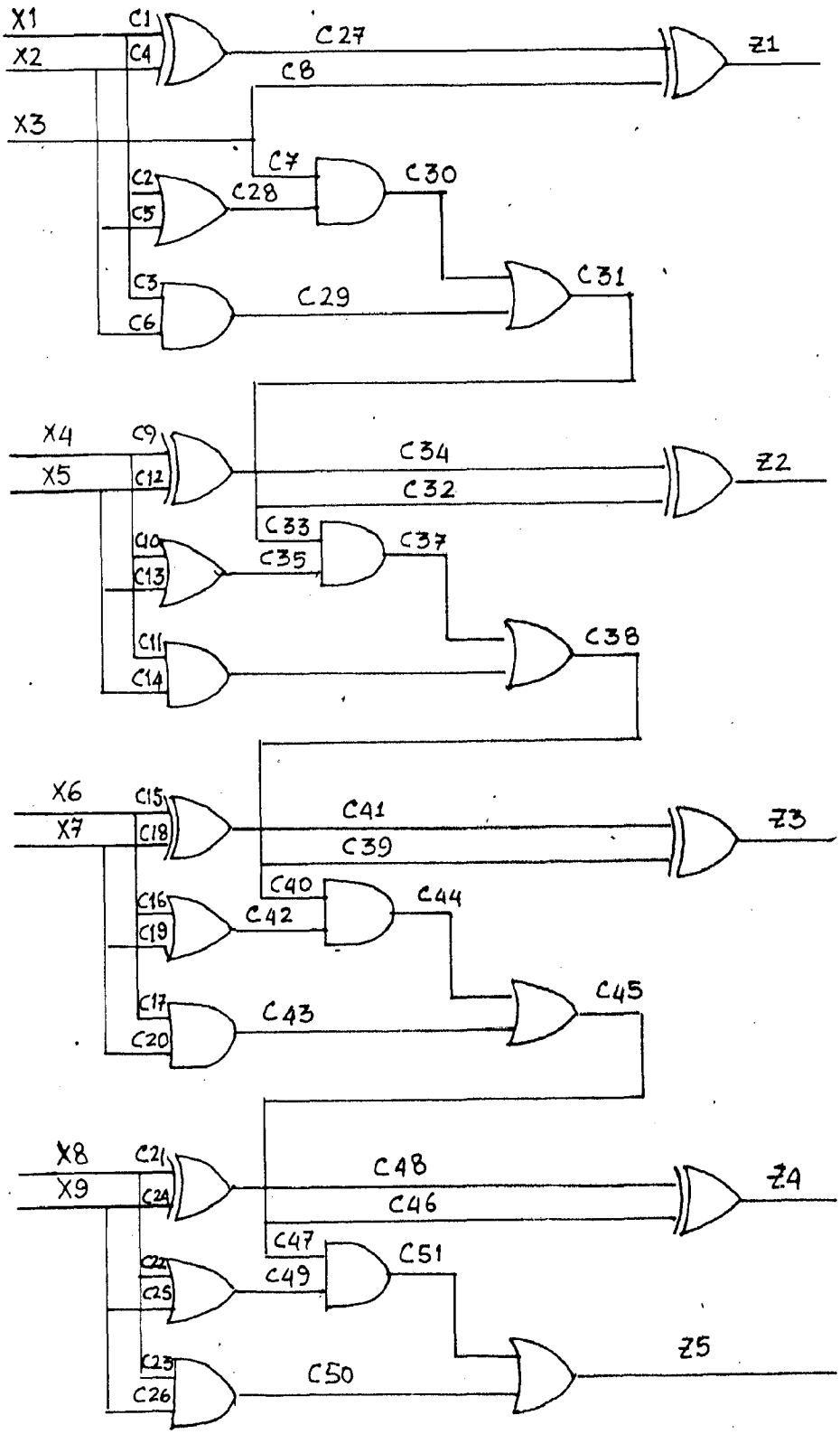
ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 10, 16, 51, 62, 74, 89,

TIME TAKEN TO SIMULATE 4 TESTS = 121 MS

RESULTS FOR STAFAN
****

NODE	C0	C1	B1	B0	D1	D0
1	.5000	.5000	.0000	.3125	.1563	.0000
2	.5000	.5000	.0938	.2188	.1094	.0469
3	.5000	.5000	.1111	.1806	.0903	.0556
4	.5000	.5000	.2083	.2188	.1094	.1042
5	.5000	.5000	.5000	.3333	.1667	.2500
6	.5000	.5000	.0000	.1875	.0938	.0000
7	.5000	.5000	.0000	.4375	.2188	.0000
8	.5000	.5000	.1667	.0417	.0208	.0833
9	.5000	.5000	.1667	.0417	.0208	.0833
10	.5000	.5000	.0000	.1667	.0833	.0000
11	.5000	.5000	.0000	.1667	.0833	.0000
12	.5000	.5000	.1667	.3333	.1667	.0833
13	.5000	.5000	.1667	.3333	.1667	.0833
14	.5000	.5000	.5000	.3333	.1667	.2500
15	.5000	.5000	.1667	.3333	.1667	.0833
16	.5000	.5000	.0000	.1875	.0938	.0000
17	.2500	.7500	.0833	.3333	.0833	.0625
18	.2500	.7500	.0000	.3333	.0833	.0000
19	.7500	.2500	.3333	.6667	.5000	.0833
20	.2500	.7500	.6667	1.0000	.2500	.5000
21	.5000	.5000	.0000	.0000	.0000	.0000
22	.5000	.5000	.3750	.0000	.0000	.1875
23	.2500	.7500	.2500	.0000	.0000	.1875
24	.5000	.5000	.0000	.3750	.1875	.0000
25	.5000	.5000	.0000	.0000	.0000	.0000
26	.2500	.7500	.0000	.2500	.0625	.0000
27	.2500	.7500	.0000	.0000	.0000	.0000
28	.5000	.5000	.0000	.3750	.1875	.0000
29	.2500	.7500	.0000	.7500	.1875	.0000
30	.2500	.7500	.0000	.0000	.0000	.0000
31	.5000	.5000	.0000	.3750	.1875	.0000
32	.5000	.5000	.0000	.5000	.2500	.0000
33	.5000	.5000	.0000	.5000	.2500	.0000
34	.2500	.7500	.0000	1.0000	.2500	.0000
35	.7500	.2500	.0000	.3333	.2500	.0000
36	.7500	.2500	.6667	1.0000	.7500	.1667
37	.7500	.2500	.0000	.2500	.1875	.0000
38	1.0000	.0000	.0000	.7500	.7500	.0000
39	1.0000	.0000	.0000	.7500	.7500	.0000
40	1.0000	.0000	.0000	.7500	.7500	.0000
41	.7500	.2500	1.0000	1.0000	.7500	.2500
42	.2500	.7500	.6667	1.0000	.2500	.5000
43	.2500	.7500	1.0000	1.0000	.2500	.7500
44	.5000	.5000	1.0000	1.0000	.5000	.5000
45	.7500	.2500	1.0000	1.0000	.7500	.2500

FRACTION OF FAULTS DETECTED = 0.5846
TIME TAKEN TO ESTIMATE FAULT COVERAGE = 119



Circuit -5

Figure 4.5

NO. OF GATES =36
CIRCUIT DESCRIPTION

```
G1/X1/C1,C2,C3/CONT/
G2/X2/C4,C5,C6/CONT/
G3/X3/C7,C8/CONT/
G4/X4/C9,C10,C11/CONT/
G5/X5/C12,C13,C14/CONT/
G6/X6/C15,C16,C17/CONT/
G7/X7/C18,C19,C20/CONT/
G8/X8/C21,C22,C23/CONT/
G9/X9/C24,C25,C26/CONT/
G10/C1,C4/C27/XOR/
G11/C2,C5/C28/OR/
G12/C3,C6/C29/AND/
G13/C7,C28/C30/AND/
G14/C29,C30/C31/OR/
G15/C31/C32,C33/CONT/
G16/C8,C27/Z1/XOR/
G17/C32,C34/Z2/XOR/
G18/C9,C12/C34/XOR/
G19/C13,C10/C35/OR/
G20/C14,C11/C36/AND/
G21/C35,C33/C37/AND/
G22/C36,C37/C38/OR/
G23/C38/C39,C40/CONT/
G24/C15,C18/C41/XOR/
G25/C39,C41/Z3/XOR/
G26/C19,C16/C42/OR/
G27/C42,C40/C44/AND/
G28/C17,C20/C43/AND/
G29/C43,C44/C45/OR/
G30/C45/C46,C47/CONT/
G31/C21,C24/C48/XOR/
G32/C48,C46/Z4/XOR/
G33/C25,C22/C49/OR/
G34/C26,C23/C50/AND/
G35/C49,C47/C51/AND/
G36/C50,C51/Z5/OR/
```

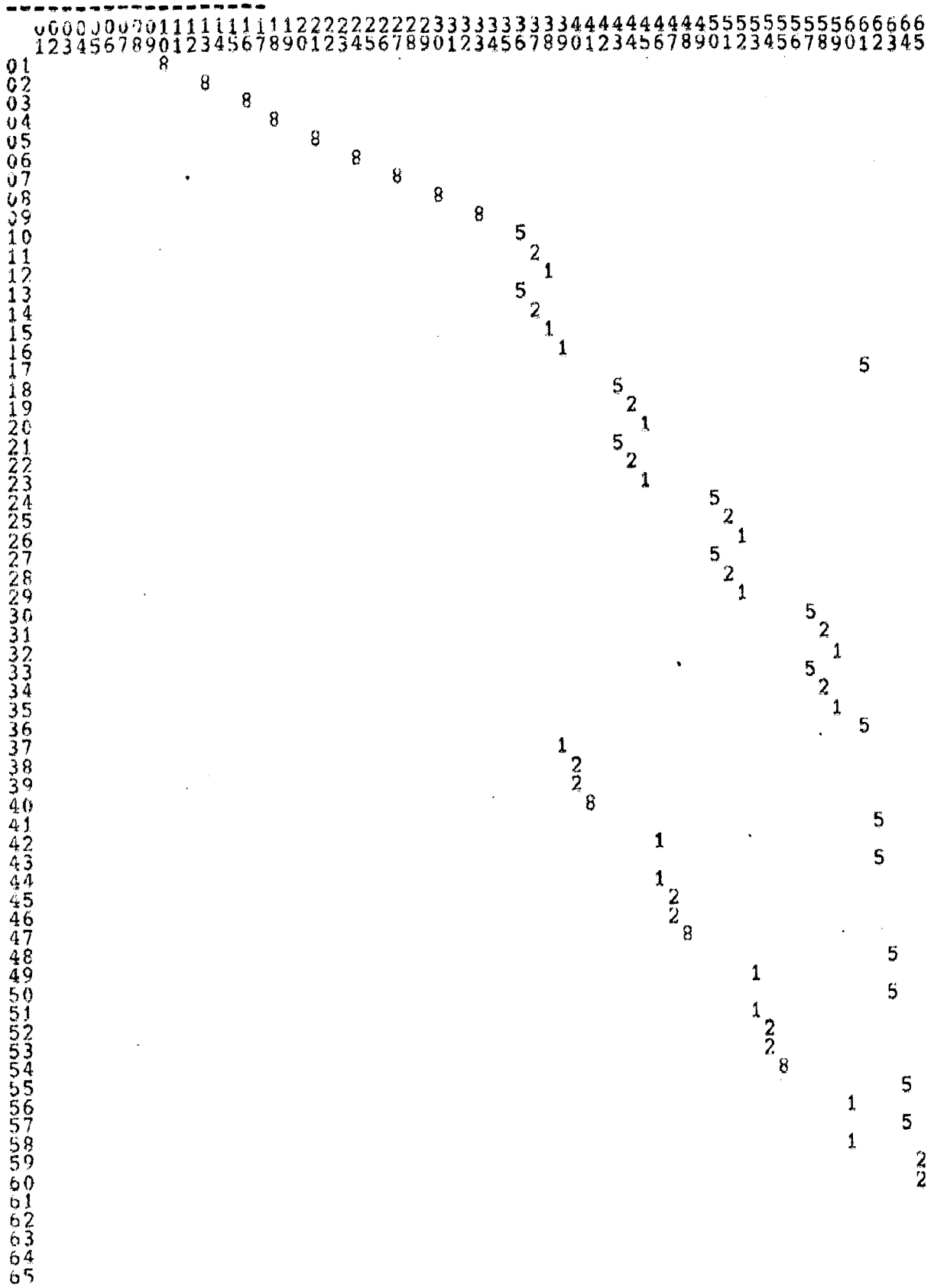
CONT DESCRIPTION
==========
NO. OF CONTS = 8

```
T1/C1,C4,C8/Z1/
T2/C2,C5,C3,C6/C31/
T3/C9,C12,C32/Z2/
T4/C10,C13,C11,C14/C38/
T5/C15,C18,C41/Z3/
T6/C16,C19,C17,C20/C45/
T7/C21,C24,C48/Z4/
T8/C22,C25,C23,C26/Z5/
```

NODE CODING

SYMBOLIC CODE	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	C_1
NUMERIC CODE	1	2	3	4	5	6	7	8	9	10
SYMBOLIC CODE	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C10	C11
NUMERIC CODE	11	12	13	14	15	16	17	18	19	20
SYMBOLIC CODE	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21
NUMERIC CODE	21	22	23	24	25	26	27	28	29	30
SYMBOLIC CODE	C22	C23	C24	C25	C26	C27	C28	C29	C30	C31
NUMERIC CODE	31	32	33	34	35	36	37	38	39	40
SYMBOLIC CODE	C32	C33	C34	C35	C36	C37	C38	C39	C40	C41
NUMERIC CODE	41	42	43	44	45	46	47	48	49	50
SYMBOLIC CODE	C42	C43	C44	C45	C46	C47	C48	C49	C50	C51
NUMERIC CODE	51	52	53	54	55	56	57	58	59	60
SYMBOLIC CODE	Z_1	Z_2	Z_3	Z_4	Z_5					
NUMERIC CODE	61	62	63	64	65					

CONNECTIVITY MATRIX



FAULT CODING

NODE NO.	1	2	3	4	5	6	7	8	9	10
S-A-0	1	3	5	7	9	11	13	15	17	19
S-A-1	2	4	6	8	10	12	14	16	18	20
NODE NO.	11	12	13	14	15	16	17	18	19	20
S-A-0	21	23	25	27	29	31	33	35	37	39
S-A-1	22	24	26	28	30	32	34	36	38	40
NODE NO.	21	22	23	24	25	26	27	28	29	30
S-A-0	41	43	45	47	49	51	53	55	57	59
S-A-1	42	44	46	48	50	52	54	56	58	60
NODE NO.	31	32	33	34	35	36	37	38	39	40
S-A-0	61	63	65	67	69	71	73	75	77	79
S-A-1	62	64	66	68	70	72	74	76	78	80
NODE NO.	41	42	43	44	45	46	47	48	49	50
S-A-0	81	83	85	87	89	91	93	95	97	99
S-A-1	82	84	86	88	90	92	94	96	98	100
NODE NO.	51	52	53	54	55	56	57	58	59	60
S-A-0	101	103	105	107	109	111	113	115	117	119
S-A-1	102	104	106	108	110	112	114	116	118	120
NODE NO.	61	62	63	64	65					
S-A-0	121	123	125	127	129					
S-A-1	122	124	126	128	130					

RESULTS FOR DEDUCTIVE SIMULATION
****

TESTS SIMULATED
==========
T 1 : 1 1 1 1 1 1 1 1
T 2 : 0 0 0 0 0 0 0 0
T 3 : 1 0 1 0 1 0 1 0 1
T 4 : 0 1 0 1 0 1 0 1 0

FAULTS DETECTION INFORMATION
==========

FAULTS DETECTED FOR T 1

FAULT NO.S DETECTED AT Z 1 ARE = 5, 33, 1, 19, 3, 25, 72,121,

FAULT NO.S DETECTED AT Z 2 ARE = 79, 81, 0,123,

FAULT NO.S DETECTED AT Z 3 ARE = 93, 95, 11, 47, 13, 53,100,125,

FAULT NO.S DETECTED AT Z 4 ARE = 15, 59, 17, 65,114,107,109,127,

FAULT NO.S DETECTED AT Z 5 ARE =129,

FAULTS DETECTED FOR T 2

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 6, 34, 2, 20, 4, 26,122,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 76, 78, 80, 82, 7, 35, 9, 41, 86,12

ADDITIONAL FAULTS DETECTED AT Z 3 ARE = 90, 92, 94, 96, 12, 48, 14, 54,126,

ADDITIONAL FAULTS DETECTED AT Z 4 ARE = 16, 60, 18, 66,104,106,108,110,128,

ADDITIONAL FAULTS DETECTED AT Z 5 ARE =118,120,130,

FAULTS DETECTED FOR T 3

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 71,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 1, 5, 21, 31, 73, 77, 8, 36, 10, 4

ADDITIONAL FAULTS DETECTED AT Z 3 ARE = 1, 5, 9, 21, 31, 43, 73, 77, 79, 8
87, 91, 99,

ADDITIONAL FAULTS DETECTED AT Z 4 ARE =113, 1, 5, 9, 13, 21, 31, 43, 55, 7
77, 79, 83, 87, 91, 93, 97,101,105,

ADDITIONAL FAULTS DETECTED AT Z 5 ARE = 1, 5, 9, 13, 17, 21, 31, 43, 55, 6
73, 77, 79, 83, 87, 91, 93, 97,101,105,107,111,115,119,

FAULTS DETECTED FOR T 4

NO ADDITIONAL FAULTS DETECTED AT Z 1

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 2, 24, 6, 32, 85,

ADDITIONAL FAULTS DETECTED AT Z 3 ARE = 10, 46, 2, 6, 24, 32, 76, 78, 80, 8

ADDITIONAL FAULTS DETECTED AT Z 4 ARE = 14, 58, 2, 6, 10, 24, 32, 46, 76, 7
80, 84, 90, 92, 94, 98,

ADDITIONAL FAULTS DETECTED AT Z 5 ARE = 18, 70, 2, 6, 10, 14, 24, 32, 46, 5
76, 78, 80, 84, 90, 92, 94, 98,104,106,108,112,
AVERAGE FAULT LIST LENGTH : 5.6769
==========

TAKEN TO SIMULATE 4 TESTS = 397 MS
==========

RESULTS FOR PARALLEL SIMULATION
****

FAULTS DETECTED FOR T 1

FAULT NO.S DETECTED AT Z 1 ARE = 1, 3, 5, 19, 25, 33, 72,121,

FAULT NO.S DETECTED AT Z 2 ARE = 79, 81,123,

FAULT NO.S DETECTED AT Z 3 ARE = 11, 13, 47, 53, 93, 95,100,125,

FAULT NO.S DETECTED AT Z 4 ARE = 15, 17, 59, 65,107,109,114,127,

FAULT NO.S DETECTED AT Z 5 ARE =129,

FAULTS DETECTED FOR T 2

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 2, 4, 6, 20, 26, 34,122,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 7, 9, 35, 41, 76, 78, 80, 82, 86,12

ADDITIONAL FAULTS DETECTED AT Z 3 ARE = 12, 14, 48, 54, 90, 92, 94, 96,126,

ADDITIONAL FAULTS DETECTED AT Z 4 ARE = 16, 18, 60, 66,104,106,108,110,128,

ADDITIONAL FAULTS DETECTED AT Z 5 ARE =118,120,130,

FAULTS DETECTED FOR T 3

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 71,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 1, 5, 8, 10, 21, 31, 37, 42, 73, 7

ADDITIONAL FAULTS DETECTED AT Z 3 ARE = 1, 5, 9, 21, 31, 43, 73, 77, 79, 8
87, 91, 99,

ADDITIONAL FAULTS DETECTED AT Z 4 ARE = 1, 5, 9, 13, 21, 31, 43, 55, 73, 7
79, 83, 87, 91, 93, 97,101,105,113, 1,

ADDITIONAL FAULTS DETECTED AT Z 5 ARE = 1, 5, 9, 13, 17, 21, 31, 43, 55, 8
73, 77, 79, 83, 87, 91, 93, 97,101,105,107,111,115,119,

FAULTS DETECTED FOR T 4

NO ADDITIONAL FAULTS DETECTED AT Z 1

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 2, 6, 24, 32, 85,

ADDITIONAL FAULTS DETECTED AT Z 3 ARE = 2, 6, 10, 24, 32, 46, 76, 78, 80, 8

ADDITIONAL FAULTS DETECTED AT Z 4 ARE = 2, 6, 10, 14, 24, 32, 46, 58, 76, 7
80, 84, 90, 92, 94, 98, 2,

ADDITIONAL FAULTS DETECTED AT Z 5 ARE = 2, 6, 10, 14, 18, 24, 32, 46, 58, 7
76, 78, 80, 84, 90, 92, 94, 98,104,106,108,112,

TIME TAKEN TO SIMULATE 4 TESTS = 156 MS

RESULTS FOR CRITICAL PATH TRACING
****

FAULTS DETECTED FOR T 1

FAULT NO.S DETECTED AT Z 1 ARE = 1, 3, 5, 19, 25, 33, 72,121,

FAULT NO.S DETECTED AT Z 2 ARE = 79, 81,123,

FAULT NO.S DETECTED AT Z 3 ARE = 11, 13, 47, 53, 93, 95,100,125,

FAULT NO.S DETECTED AT Z 4 ARE = 15, 17, 59, 65,107,109,114,127,

FAULT NO.S DETECTED AT Z 5 ARE =129,

FAULTS DETECTED FOR T 2

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 2, 4, 6, 20, 26, 34,122,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 7, 9, 35, 41, 76, 78, 80, 82, 86,12

ADDITIONAL FAULTS DETECTED AT Z 3 ARE = 12, 14, 48, 54, 90, 92, 94, 96,126,

ADDITIONAL FAULTS DETECTED AT Z 4 ARE = 16, 18, 60, 66,104,106,108,110,128,

ADDITIONAL FAULTS DETECTED AT Z 5 ARE =118,120,130,

FAULTS DETECTED FOR T 3

ADDITIONAL FAULTS DETECTED AT Z 1 ARE = 71,

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 1, 5, 8, 10, 21, 31, 37, 42, 73, 7

ADDITIONAL FAULTS DETECTED AT Z 3 ARE = 1, 5, 9, 21, 31, 43, 73, 77, 79, 8
87, 91, 99,

ADDITIONAL FAULTS DETECTED AT Z 4 ARE = 1, 5, 9, 13, 21, 31, 43, 55, 73, 7
79, 83, 87, 91, 93, 97,101,105,113, 1,

ADDITIONAL FAULTS DETECTED AT Z 5 ARE = 1, 5, 9, 13, 17, 21, 31, 43, 55, 6
73, 77, 79, 83, 87, 91, 93, 97,101,105,107,111,115,119,

FAULTS DETECTED FOR T 4

NO ADDITIONAL FAULTS DETECTED AT Z 1

ADDITIONAL FAULTS DETECTED AT Z 2 ARE = 2, 6, 24, 32, 85,

ADDITIONAL FAULTS DETECTED AT Z 3 ARE = 2, 6, 10, 24, 32, 46, 76, 78, 80, 8

ADDITIONAL FAULTS DETECTED AT Z 4 ARE = 2, 6, 10, 14, 24, 32, 46, 58, 76, 7
80, 84, 90, 92, 94, 98, 2,

ADDITIONAL FAULTS DETECTED AT Z 5 ARE = 2, 6, 10, 14, 18, 24, 32, 46, 58, 7
76, 78, 80, 84, 90, 92, 94, 98,104,106,108,112,

TIME TAKEN TO SIMULATE 4 TESTS = 216 MS

RESULTS FOR STAFAN
****

NODE	C0	C1	B1	B0	D1	D0
1	.5000	.5000	.3707	.4080	.2040	.1853
2	.5000	.5000	.3707	.4080	.2040	.1853
3	.5000	.5000	.6680	.6680	.3340	.3340
4	.5000	.5000	.0382	.0764	.0382	.0191
5	.5000	.5000	.0382	.0764	.0382	.0191
6	.5000	.5000	.3750	.4167	.2083	.1875
7	.5000	.5000	.3750	.4167	.2083	.1875
8	.5000	.5000	.3889	.4444	.2222	.1944
9	.5000	.5000	.3889	.4444	.2222	.1944
10	.5000	.5000	1.0000	1.0000	.5000	.5000
11	.5000	.5000	.1120	.0000	.0000	.0560
12	.5000	.5000	.0000	.2240	.1120	.0000
13	.5000	.5000	1.0000	1.0000	.5000	.5000
14	.5000	.5000	.1120	.0000	.0000	.0560
15	.5000	.5000	.0000	.2240	.1120	.0000
16	.5000	.5000	.3359	.3359	.1680	.1680
17	.5000	.5000	1.0000	1.0000	.5000	.5000
18	.5000	.5000	.0000	.0000	.0000	.0000
19	.5000	.5000	.1146	.0000	.0000	.0573
20	.5000	.5000	.0000	.2292	.1146	.0000
21	.5000	.5000	.0000	.0000	.0000	.0000
22	.5000	.5000	.1146	.0000	.0000	.0573
23	.5000	.5000	.0000	.2292	.1146	.0000
24	.5000	.5000	1.0000	1.0000	.5000	.5000
25	.5000	.5000	.1250	.0000	.0000	.0625
26	.5000	.5000	.0000	.2500	.1250	.0000
27	.5000	.5000	1.0000	1.0000	.5000	.5000
28	.5000	.5000	.1250	.0000	.0000	.0625
29	.5000	.5000	.0000	.2500	.1250	.0000
30	.5000	.5000	1.0000	1.0000	.5000	.5000
31	.5000	.5000	.1667	.0000	.0000	.0833
32	.5000	.5000	.0000	.3333	.1667	.0000
33	.5000	.5000	1.0000	1.0000	.5000	.5000
34	.5000	.5000	.1667	.0000	.0000	.0833
35	.5000	.5000	.0000	.3333	.1667	.0000
36	.5000	.5000	1.0000	1.0000	.5000	.5000
37	.2500	.7500	.2240	.0000	.0000	.1680
38	.7500	.2500	.0000	.4479	.3359	.0000
39	.5000	.5000	.3359	.6719	.3359	.1680
40	.5000	.5000	.6719	.6719	.3359	.3359
41	.5000	.5000	1.0000	1.0000	.5000	.5000
42	.5000	.5000	.3438	.3438	.1719	.1719
43	.5000	.5000	1.0000	1.0000	.5000	.5000
44	.2500	.7500	.2292	.0000	.0000	.1719
45	.7500	.2500	.0000	.4583	.3438	.0000
46	.5000	.5000	.3438	.6875	.3438	.1719
47	.5000	.5000	.6875	.6875	.3438	.3438
48	.5000	.5000	1.0000	1.0000	.5000	.5000
49	.5000	.5000	.3750	.3750	.1875	.1875
50	.5000	.5000	1.0000	1.0000	.5000	.5000
51	.2500	.7500	.2500	.0000	.0000	.1875
52	.7500	.2500	.0000	.5000	.3750	.0000
53	.5000	.5000	.3750	.7500	.3750	.1875
54	.5000	.5000	.7500	.7500	.3750	.3750
55	.5000	.5000	1.0000	1.0000	.5000	.5000
56	.5000	.5000	.5000	.5000	.2500	.2500
57	.5000	.5000	1.0000	1.0000	.5000	.5000
58	.2500	.7500	.3333	.0000	.0000	.2500
59	.7500	.2500	.0000	.6667	.5000	.0000
60	.5000	.5000	.5000	1.0000	.5000	.2500
61	.5000	.5000	1.0000	1.0000	.5000	.5000
62	.2500	.7500	1.0000	1.0000	.2500	.7500
63	.5000	.5000	1.0000	1.0000	.5000	.5000
64	.5000	.5000	1.0000	1.0000	.5000	.5000
65	.5000	.5000	1.0000	1.0000	.5000	.5000

FRACTION OF FAULTS DETECTED = 0.7291
TIME TAKEN TO ESTIMATE FAULT COVERAGE = 171

Circuit No.	Parallel	Deductive	Critical Path	STAFAN
1.	41	171	100	82
2.	53	212	114	93
3.	95	270	112	103
4.	83	266	121	119
5.	156	397	216	171

C.P.U.TIME IN MILLISECONDS

TABLE

CIRCUIT NO	NO. OF GATES	NO.OF FAULTS	FANOUT
1.	15	52	6
2.	21	70	8
3.	22	74	4
4.	24	90	9
5.	36	130	12

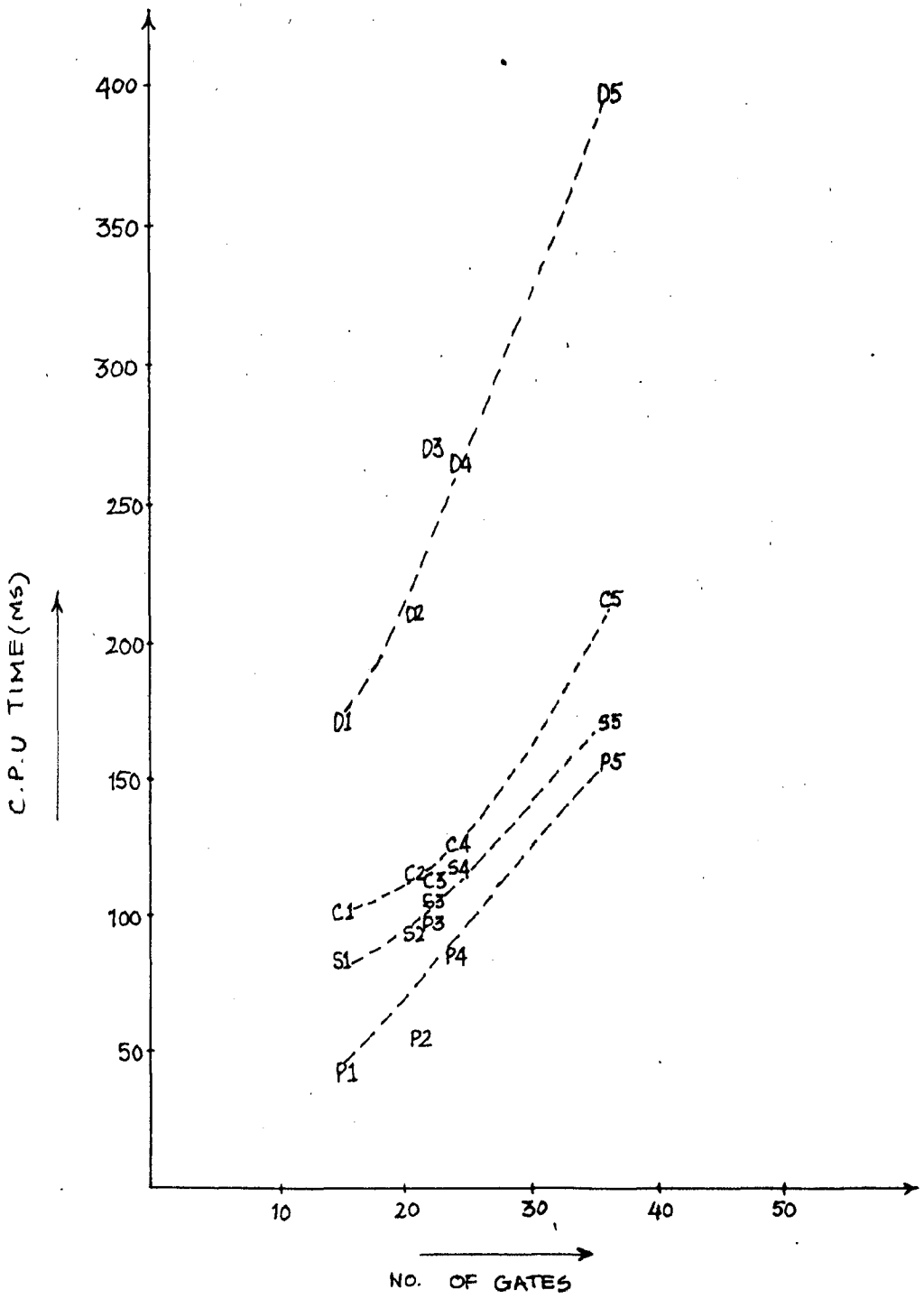
CIRCUIT PARAMETERS

TABLE 4.2

in circuit. These parameters for each of the circuits considered have been shown in Table 4.2.

The relationship between the C.P.U. time requirements of the various techniques and the circuit size have been considered in Figure 4.6. From these curves it is apparent that in general the C.P.U. time requirements increase with circuit size for all the methods. The relative requirements of the C.P.U. time of the various methods can also be observed. It is seen that parallel simulation is the most efficient. The marked difference between the performances of parallel simulation and deductive simulation can be assigned to the fact that parallel simulation has been implemented in assembly language while the other techniques have been implemented in FORTRAN. As a result the logic gates are simulated as simple logic operation between the computer words while for other techniques subroutine calls specialized routines have to be made. Critical path tracing and STAFAN show higher performance efficiency than deductive simulation. This is because in critical path tracing only the detected faults are processed, while STAFAN estimates the fault coverage information directly from the true value simulation data without exclusively simulating each of the faults.

The average C.P.U. time/gate requirement for parallel simulation has been calculated in the Table 4.3. The information obtained from this table is plotted against the number of faults



C.P.U TIME AS A FUNCTION OF SIZE

FIGURE 4.6

simulated in Figure 4.7. This figure gives information about the C.P.U. time requirements as the number of passes through the parallel simulation program increase. This is because in one pass only 36 faults can be simulated.

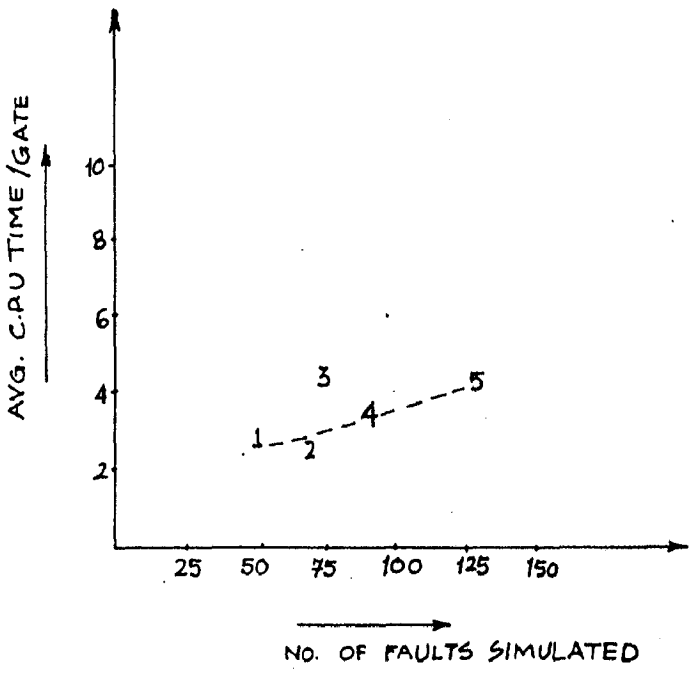
Ckt. No.	C.P.U.Time	No. of Gates	An. time per gate
1.	41	15	2.73
2.	53	21	2.52
3.	95	22	4.31
4.	83	24	3.45
5.	156	36	4.33

CALCULATION OF AVG TIME PER GATE
FOR PARALLEL SIMULATION

TABLE 4.3.

Ckt.No.	C.P.U.Time	No. of gates	C.P.V. time/gate	Avg. fault list length.
1.	171	15	11.45	4.6154
2.	212	21	10.1	2.5429
3.	270	22	12.23	8.3514
4.	266	24	11.08	3.333
5.	397	36	11.04	5.6769

TABLE 4.4

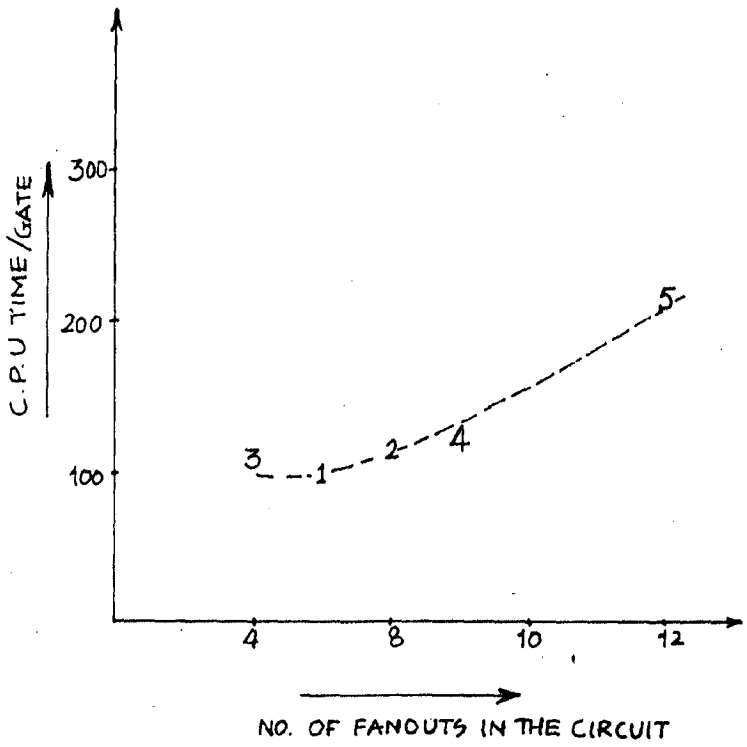


PARALLEL SIMULATION:
AVG. C.P.U TIME/GATE VS. FAULTS SIMULATED
FIGURE 4.7

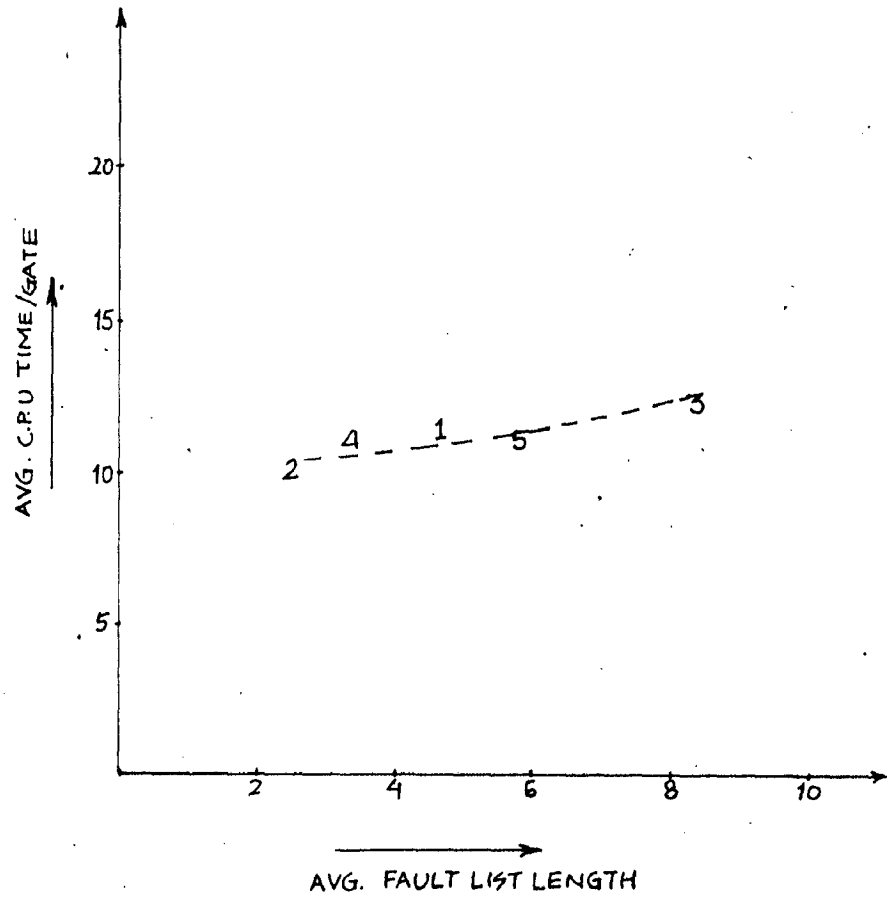
The efficiency of critical path tracing depends upon the amount of fanout in the circuit. A sketch between the C.P.U. time requirements and the number of fanouts in each circuit is shown in Figure 4.8 for this technique. It is apparent from this sketch that the time requirements of this technique increase with the amount of fanout.

For deductive simulation the average fault list length has been chosen as the parameter for performance efficiency. This is because as the average fault list length increases the amount of processing required increases thus increasing C.P.U. time required. This is shown in Table 4.4 and Figure 4.9, where average C.P.U. time/gate has been plotted against the average fault list length.

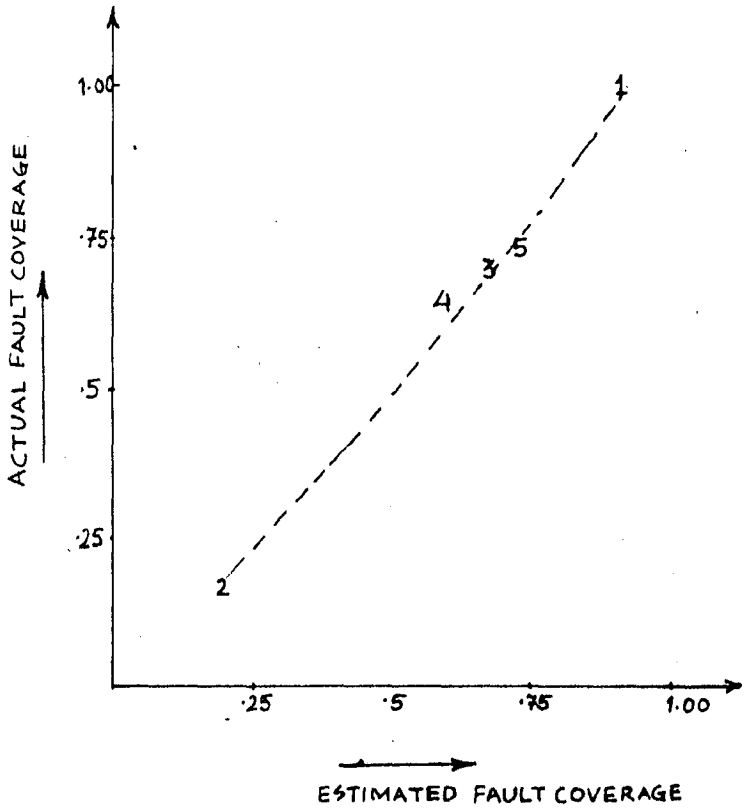
To evaluate the performance of STAFAN it is necessary to check the accuracy of the fault coverage estimate produced. This can be done by calculating the actual fault coverage from the fault detection information of the fault simulation techniques and then comparing it with the estimated fault coverage. This has been done in table 4.5 and Figure 4.10 respectively. From Figure 4.10 it is evident that the curve is very close to a straight line with slope unity. It can be concluded that a fairly good estimate of fault coverage is obtained from STFAN. The accuracy of the estimation will increase with the increase in circuit size due to the statistical nature of the technique.



CRITICAL PATH TRACING:
C.P.U TIME AS A FUNCTION OF FANOUT NUMBER
FIGURE 4.8



DEDECTIVE SIMULATION:
AVG. C.P.U TIME/GATE VS. AVG. FAULT LIST LENGTH
FIGURE 4.9



FAULT COVERAGE :
COMPARISON OF ACTUAL & ESTIMATED
FIGURE 4·10

From the results it is clear that the basic advantage of fault simulation techniques over STAFAN are that they give a complete fault detection information for the different tests. However as the trend of VLSI circuits is increasing statistical techniques like STAFAN become advantageous because of the enormous increase in the number of gates/ circuit and an explosion in the number of faults to be considered.

Circuit No.	No. of faults detected	Total no. of faults	Actual Fault coverage	Estimated Fault coverage
1.	52	52	1.00	0.9283
2.	12	70	0.1714	0.1922
3.	51	74	0.689	0.6733
4.	56	90	0.644	0.5846
5.	95	130	0.730	0.7291

TABLE 4.5

4.2 SUGGESTIONS FOR FUTURE WORK

In this work the four main techniques have been considered for combinational circuits using the single stuck-at fault model. Suggestion for future extension of this work are given below.

1. The sequential circuits can be included using the block array model [4].
2. Here only two logic values -0/1 have been considered. The work can be extended to accomodate multiple level logic.
3. The deductive algorithm implemented in this work is the one suggested by Armstrong. This can be extended to concurrent simulation by considering fault states as suggested by **Wrich** and Baker.
4. Multiple stuck -at faults may also be simulated.
5. The present work represents the problem of fault simulation at the gate level. It would be very useful to extend this to include fault simulation at functional level. This will facilatete application to much complicated and larger circuits.

REFERENCES

1. ABROMOVICI. M, P.R.Menon and D.T.Miller, 'Critical Path Tracing: An Alternative to fault simulation'. IEEE Design and Test. Feb. 1984.
2. Armstrong D.B., 'A Deductive method of simulating faults in logic circuits', IEEE Trans. Comput. Vol. C-21, May 1972.
3. Bennets, R.G., 'Introduction to Digital Board Testing', Crave Russak, New York, 1982.
4. Brever M.A. and A.D.Friedman, 'Diagnosis and Reliable Design of Digital Systems', Computer Sc. Press Inc.,1976.
5. Chang, H.Y.,S.G.Chappell, C.H.Elureudorf and L.D.Schmidt, 'Comparison of Parallel and Deductive fault simulation Methods', IEEE Trans. Comput, Vol. -23, No.11, 1974, Nov.
6. Friedman, A.D. and P.R.Manon, 'Fault detection in digital circuits', Prentice Hall, Englewood Cliff, N.J.,1971.
7. GORIN, R.E., 'Introduction to DECSYSTEM-20 Assembly language programming', Digital Equipment Corp., Bedford Masachussets, 1981.
8. Jain S.K. and V.D.Agrawal, 'Statistical fault Analysis', IEEE Design and Test Feb. 1985.
9. Jain, S.K. and V.D.Agarwal, 'SCAFAN: An Alternative to fault simulation', Proc. 21st Design Automation Conf., June, 1984.

10. MICZO, Alexander, 'Digital logic Testing and simulation', Harper and Ran Publishers, New York, 1986.

11. Pradhan, D.K., 'Fault tolerant computing: Theory and techniques', Vol.1, Prentice Hall, Englewood Cliffs, N.J., 1986.

12. Seshu, S., 'On an improved Diagnosis Pattern', IEEE Trans. Electron Computer, Vol. EC-14, 1965.

13. Trivedi, K.S.,'Probability and Statistics with Reliability, Queueing and Computer Sc. Applications', Prentice Hall, Englewoods Cliffs, N.J., 1982.

14. Ulrich, E.G. and T.Baker, 'Concurrent simulation of nearly identical digital networks', IEEE Computer, April, 1974.

15. Wang, D.T.'An Algorithm for the generation of Test sets for combinational logic networks', IEEE Trans. Computers, Vol. C-24, No.7, July 1975.

16. HILO Logic simulator Manual: Technical Memorandeun, Prumel University, Deptt. of Electrical Engg. and Electronics.

17. DECSYSTEM-20-Maoro Assembler Manual, Digital Equipment Corp., Bedford, Masachussets, 1981.

18. DECSYSTEM-20-Link Reference Manual, Digital Equipment Corp., Bedford, Masachussets, 1981.

19. DECSYSTEM-20, FORTRAN Reference Manual, Digital Equipment Corp., Bedford, Masachussets, 1981.

APPENDIX

```
00000       IMPLICIT INTEGER(A-Z)
00100       DIMENSION CKTDES(100,100),INDEX(10),NODE(100,2),ICNT(100)
00200       1,OCNT(100),CON(100,4,100),TYPE(100),INPUT(100,10),OUT(100,10)
00300       2,STYPE(8,4),CONECT(100,100),FLTNUM(100,2),CO(100),CI(100)
00400       COMMON INPUT,OUT,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN,NOGA,
00500       1NODNUM
00550       OPEN(UNIT=21,DIALOG)
00575       OPEN(UNIT=23,DIALOG)
00600       DATA INDEX/'0','1','2','3','4','5','6','7','8','9'/
00700       DATA STYPE/'A','D','N','N','X','X','N','C','N','R','A','D',
00800       1'O','N','O','O','D','/','N','R','R','O','T','N','/',
00900       20,'D','/','/','R','/','T'/
01000       DO 8 J=1,100
01100       DO 6 I=1,100
01200       CKTDES(J,I)=' '
01250       CONECT(J,I)=' '
01300 6     CONTINUE
01400 8     CONTINUE
01700       READ(21,*),NOGA
01750       WRITE(23,10),NOGA
01800 10    FORMAT(1X,'NO. OF GATES =',I2)
01900       WRITE(23,30)
02000 30    FORMAT(1X,'CIRCUIT DESCRIPTION'/1X,'----------------------')
02100       READ(21,40)((CKTDES(J,I),I=1,100),J=1,NOGA)
02200 40    FORMAT(100A1)
02300       WRITE(23,35)((CKTDES(J,I),I=1,50),J=1,NOGA)
02400 35    FORMAT(1X,50A1)
02500       DO 43 J=1,NOGA
02600       DO 42 I=1,100
02700       DO 41 G=1,10
02800       IF(CKTDES(J,1).EQ.INDEX(G))GO TO 38
02900       GO TO 41
03000 38    CKTDES(J,I)=G-1
03100       DO 39 A=1,10
03200       IF(CKTDES(J,I+1).EQ.INDEX(A))CKTDES(J,I)=CKTDES(J,I)*10+(A-1)
03300 39    CONTINUE
03400 41    CONTINUE
```

PREPROCESSING

```
03560 42      CONTINUE
03600 43      CONTINUE
03700         PRINCN=0
03800         INTCNT=0
03900         PROTCN=0
04000         DO 90 J=1,NOGA
04100         DO 80 I=2,100
04200         IF(CKTDES(J,I).EQ.'X')GO TO 50
04300         IF(CKTDES(J,I).EQ.'C')GO TO 60
04400         IF (CKTDES(J,I).EQ.'Z')GO TO 70
04500         GO TO 80
04600 50      IF(PRINCN.LT.CKTDES(J,I+1))PRINCN=CKTDES(J,I+1)
04700         GO TO 80
04800 60      IF(INTCNT.LT.CKTDES(J,I+1))INTCNT=CKTDES(J,I+1)
04900         GO TO 80
05000 70      IF(PROTCN.LT.CKTDES(J,I+1))PROTCN=CKTDES(J,I+1)
05100 80      CONTINUE
05200 90      CONTINUE
05300         DO 120 I=1,NOGA
05400         D=1
05500         J=0
05600         N=1
05700 100     J=J+1
05800         IF(CKTDES(I,J).NE.'/')GO TO 110
05900         CON(I,D,N)=CKTDES(I,J)
06000         D=D+1
06100         IF(D.EQ.5)GO TO 120
06200         N=1
06300         GO TO 100
06400 110     CON(I,D,N)=CKTDES(I,J)
06500         N=N+1
06600         GO TO 100
06700 120     CONTINUE
06800         DO 190 I=1,NOGA
06900         ICNT(I)=0
07000         OCNT(I)=0
07100         DO 180 D=2,3
```

```
07200          N=0
07300 130      N=N+1
07400          IF(CON(I,D,N).EQ.'X')GO TO 140
07500          IF (CON(I,D,N).EQ.'C')GO TO 150
07600          IF(CON(I,D,N).EQ.'Z')GO TO 170
07700          IF(CON(I,D,N).EQ.'/')GO TO 180
07800          GO TO 130
07850 140      IF(D.EQ.3)GO TO 700
07900          N=N+1
08000          ICNT(I)=ICNT(I)+1
08100          NODE(CON(I,D,N),1)='X'
08200          NODE(CON(I,D,N),2)=CON(I,D,N)
08300          INPUT(I,ICNT(I))=CON(I,D,N)
08400          GO TO 130
08500 150      N=N+1
08600          NODE((CON(I,D,N)+PRINCN),1)='C'
08700          NODE((CON(I,D,N)+PRINCN),2)=CON(I,D,N)
08800          IF(D.EQ.3)GO TO 160
08850          CI(CON(I,D,N))=1
08900          ICNT(I)=ICNT(I)+1
09000          INPUT(I,ICNT(I))=CON(I,D,N)+PRINCN
09100          GO TO 130
09150 160      CO(CON(I,D,N))=1
09200          OCNT(I)=OCNT(I)+1
09300          OUT(I,OCNT(I))=CON(I,D,N)+PRINCN
09400          GO TO 130
09450 170      IF(D.EQ.2)GO TO 700
09500          N=N+1
09600          OCNT(I)=OCNT(I)+1
09700          OUT(I,OCNT(I))=CON(I,D,N)+PRINCN+INTCNT
09800          NODE((CON(I,D,N)+PRINCN+INTCNT),1)='Z'
09900          NODE((CON(I,D,N)+PRINCN+INTCNT),2)=CON(I,D,N)
10000          GO TO 130
10100 180      CONTINUE
10200 190      CONTINUE
10202          DO 701 K=1,INTCNT
10204          IF(CI(K).NE.CO(K))GO TO 700
```

```
      CONTINUE
      NODNUM=PRINCN+INTCNT+PROTCN
      WRITE(23,195)
195   FORMAT(1H1,1X,'NODE CODING'/1X,'-----------')
      N=NODNUM
      NN=N/10
      DO 210 P=1,NN
      PP=(P-1)*10+1
      LL=PP+9
      WRITE(23,200)((NODE(I,J),J=1,2),I=PP,LL),(I,I=PP,LL)
200   FORMAT(/1X,'SYMBOLIC CODE ',10(A1,I2,3X)/1X,'NUMERIC CODE '
     1,10(I3,3X))
      N=N-10
      IF(N.EQ.0)GO TO 211
210   CONTINUE
      WRITE(23,205)((NODE(I,J),J=1,2),I=LL+1,NODNUM)
205   FORMAT(/1X,'SYMBOLIC CODE ',10(A1,I2,3X))
      WRITE(23,206)(I,I=LL+1,NODNUM)
206   FORMAT(1X,'NUMERIC CODE ',10(I3,3X))
211   DO 215 I=1,NOGA
      DO 214 K=1,8
      DO 212 J=1,4
      IF(COD(I,4,J).EQ.STYPE(K,J))GO TO 212
      GO TO 214
212   CONTINUE
      TYPE(I)=K
      GO TO 215
214   CONTINUE
215   CONTINUE
      WRITE(23,515)
515   FORMAT(1H1,1X,'CONNECTIVITY MATRIX'/1X,'------------------')
      DO 411 F=1,NODNUM
      N=F/10
      CONECT(1,F+2)=INDEX(N+1)
      CONECT(2,F+2)=INDEX(F-N*10+1)
      CONECT(F+2,1)=INDEX(N+1)
      CONECT(F+2,2)=INDEX(F-N*10+1)
```

PREPROCESSING

```
10813 411      CONTINUE
10814          DO 421 F=1,NODNUM
10815          DO 413 G=1,NOCA
10816          DO 415 H=1,ICNT(G)
10818          IF(INPUT(G,H).NE.F)GO TO 415
10820          CONECT(F+2,OUT(G,1)+2)=INDEX(TYPE(G)+1)
10822          GO TO 421
10824 415      CONTINUE
10826 413      CONTINUE
10828 421      CONTINUE
10829          DO 419 I=1,NODNUM+2
10830          WRITE(23,417)(CONECT(I,J),J=1,NODNUM+2)
10832 417      FORMAT(1X,70(A1))
10932 419      CONTINUE
11800          WRITE(23,217)
11900 217      FORMAT(1H1,1X,'FAULT CODING'/1X,'------------')
12000          DO 245 I=1,NODNUM
12100          DO 240 K=1,2
12200          Z=K-1
12300          IF(Z.EQ.0)GO TO 230
12400          FLTNUM(I,2)=2*I
12500          GO TO 240
12600 230      FLTNUM(I,1)=2*I-1
12900 240      CONTINUE
13000 245      CONTINUE
13001          N=NODNUM
13002          NN=N/10
13003          DO 542 Y=1,NN
13004          YY=(Y-1)*10+1
13006          MM=YY+9
13053          WRITE(23,302)(I,I=YY,MM),((FLTNUM(I,J),I=YY,MM),J=1,2)
13076 302      FORMAT(/1X,'NODE NO.',10(I3,3X)/1X,'S-A-0   ',10(I3,3X)/1X
13077          1,'S-A-1   ',10(I3,3X))
13078          N=N-10
13080          IF(N.EQ.0)GO TO 909
13082 542      CONTINUE
13084          WRITE(23,445)(I,I=MM+1,NODNUM)
```

```
13086 445     FORMAT(/1X,'NODE NO.',10(I3,3X))
13088         WRITE(23,455)(FLTNUM(I,1),I=MM+1,NODNUM)
13090 455     FORMAT(1X,'S-A-0   ',10(I3,3X))
13100         WRITE(23,555)(FLTNUM(I,2),I=MM+1,NODNUM)
13110 555     FORMAT(1X,'S-A-1   ',10(I3,3X))
13120 999     GO TO 767
13150 700     WRITE(23,702)
13175 702     FORMAT(1X,'ERROR IN INPUT DESCRIPTION')
13200 767     STOP
13300         END
```

```
;================================================================
;          ROUTINE FOR PARALLEL SIMULATION
;================================================================
          TITLE PARALLEL
          SEARCH MONSYM
INTERN PALSIM
ENTRY PALSIM
RADIX 10
.COMMON AREA1[2300]
.COMMON APEA2[5]
.COMMON AREA3[2700]
INPUT=AREA1
OUT=INPUT+^D1000
ICNT=OUT+^D1000
OCNT=ICNT+^D100
TYPE=OCNT+^D100
PRINCN=AREA2
INTCNT=PRINCN+1
PROTCN=INTCNT+1
NOGA=PROTCN+1
MODNUM=NOGA+1
FTDT=AREA3
NFTDT=FTDT+2500
TVAL=NFTDT+100
A=1
B=2
C=3
D=4
R=5
S=6
W=7
X=8
Y=9
Z=10
V=11
P=12
N=13
```

```
038?0 N=14
03900
04?00 SREG:   BLOCK   25
041?0 NODE:   BLOCK   1000
042?0 NNODES: 0
04300
04400
04500 TABLE:  0
04600         AND     S,NODE(R)
04700         IOR     S,NODE(R)
04800         AND     S,NODE(R)
049?0         IOR     S,NODE(R)
05000         XOR     S,NODE(R)
05100         XOR     S,NODE(R)
05200         SETCM   S,S
05300         CALL      CONT
05400
05500
05600
05710 OPDEF CALL [PUSHJ P,]
05800 OPDEF RET  [POPJ  P,]
05900 STKLEN==10
06?00 STACK:  BLOCK STKLEN
06100
06200 PALSIM: MOVEM   15,SREG+15
06300         MOVEI   15,SREG
06400         BLT
06500         MOVE    P,[IOWD STKLEN,STACK]
06600 PHASE1: SETZ    X,
06700         MOVEI   A,7
06800         MOVEM   A,NNODES
06900         CAML    X,PRINCN
07?00         JRST      PHASE2
07100         MOVE    Y,IVAL(X)
07200         CAIE    Y,0
07300         SETO    Y,
07400         MOVE    A,X
```

PARALLEL SIMULATION

```
07500           MOVE    M,NNODES
07600           IMUL    M,X
07700           SETZ    N,
07800 PH12:     MOVEM   Y,NODE(M)
07900           AOJ     M,
08000           AOJ     N,
08100           CAME    N,NNODES
08200           JRST      PH12
08300           CALL      SETFLT
08400           AOJA    X,PHASE1+1
08500 PHASE2:   SETZ    X,
08600
08700           CAML    X,NOGA
08800           JRST      PHASE3
08900           MOVE    W,X
09000           MOVE    Z,TYPE(X)
09100           IMULI   W,10
09200           MOVE    R,INPUT(W)
09300           MOVE    V,OUT(W)
09400           SOJ     V,
09500           SOJ     R,
09600           SETZ    N,
09700           MOVE    M,NNODES
09800           IMUL    M,R
09900           ADD     M,N
10000 PHA2:     MOVE    S,NODE(M)
10100           MOVE    W,X
10200           IMULI   W,10
10300           MOVEI   Y,1
10400 PH21:     AOJ     Y,
10500           AOJ     W,
10600           MOVE    R,INPUT(W)
10700           SOJ     R,
10800           IMUL    R,NNODES
10900           ADD     R,N
11000           XCT       TABLE(Z)
11100
```

PARALLEL SIMULATION

```
11200           CAIN    Z,8
11300           JRST      PHB2
11400           CAMGE   Y,ICNT(X)
11500           JRST      PH21
11600 PH22:     MOVE    R,V
11700           IMUL    R,NNODES
11800           ADD     R,N
11900           CAIN    Z,3
12000           SETCM   S,S
12100           CAIN    Z,4
12200           SETCM   S,S
12300           CAIN    Z,6
12400           SETCM   S,S
12500           MOVEM   S,NODE(R)
12600           AOJ     N,
12700           AOJ     M,
12800           CAME    N,NNODES
12900           JRST      PHA2
13000           MOVE    A,V
13100           CALL      SETFLT
13200 PHB2:     AOJA    X,PHASE2+1
13300
13400
13500 PHASE3:   MOVE    X,PRINCN
13600           ADD     X,INTCNT
13700           SETZ    C,
13800 PH33:     CAML    X,NODNUM
13900           JRST      PH34
14000           MOVE    B,C
14100           IMULI   B,25
14200           MOVE    M,NNODES
14300           IMUL    M,X
14400           MOVE    R,NODE(M)
14500           JUMPGE  R,PH3B+1
14600           MOVE    D,M
14700           SETZ    N,
14800 PH3A:     AOJ     N,
```

```
14900           CAML    N,NNODES
15000           JRST      PH3B
15100           AOJ     D,
15200           MOVE    S,NODE(D)
15300           SETCM   S,S
15400           MOVEM   S,NODE(D)
15500           JRST      PH3A
15600 PH3B:     SETCM   R,R
15700           MOVE    Z,NODNUM
15800           IMULI   Z,2
15900           SETZ    A,
16000           SETZ    D,
16100           ROT     R,1
16200
16300           MOVEI   W,34
16400           SKIPA
16500           MOVEI   W,36
16600 AA:       SOJ     W,
16700           AOJ     D,
16800           JUMPGE  R,AA1
16900           AOJ     A,
17000           MOVEM   D,FTDT(B)
17100           AOJ     B,
17200 AA1:      CAML    D,Z
17300           JRST      BB
17400           ROT     R,1
17500           JUMPG   W,AA
17600           AOJ     M,
17700           MOVE    R,NODE(M)
17800           JRST      AA-1
17900 BB:       MOVEM   A,NFTDT(C)
18000           AOJ     C,
18100           AOJA    X,PH33
18200 PH34:     MOVSI   15,SREG
18300           BLT     15,15
18400           POPJ    15,
18500 SETFLT:   CAILE   A,17
```

```
18600          JRST     STF2
18700          MOVE     B,[POINT 2,NODE,2]
18800          MOVE     C,A
18900          IMUL     C,NNODES
19000 STF1:    ADD      B,C
19100          ADJBP    A,B
19200          MOVEI    C,1
19300          DPB      C,A
19400          RET
19500 STF2:    MOVE     C,A
19600          IMUL     C,NNODES
19700          AOJ      C,
19800          MOVE     B,[POINT 2,NODE,1]
19900          SUBI     A,17
20000          JRST     STF1
20100
20200
20300 CONT:    SETZ N,
20400          MOVE S,NODE(M)
20500          MOVE V,OCNT(X)
20600          MOVE W,X
20700          IMULI W,10
20800 CNT2:    MOVE R,OUT(W)
20900          SOJ      R,
21000          IMUL R,NNODES
21100          ADD R,N
21200          MOVEM S,NODE(R)
21300          AOJ W,
21400          SOJGE V,CNT2
21500          AOJ M,
21600          AOJ N,
21700          CAME N,NNODES
21800          JRST CONT+1
21900          MOVE V,OCNT(X)
22000          MOVE W,X
22100          IMULI W,10
22200 CNT3:    MOVE A,OUT(W)
```

```
22300          SOJ     A,
22400          CALL SETFLT
22500          AOJ W,
22600          SOJGE V,CNT3
22700          RET
22800
22900
23000 END
```

```
00100 C=======================================================================
00200 C       FAULT FREE SIMULATION
00300 C=======================================================================
00400         SUBROUTINE FFSIMU
00500 C       XIN=NO. OF INPUT SETS TO BE SIMULATED
00600 C       X(I,J)=ITH INPUT SET'S JTH PRIMARY INPUT
00700         IMPLICIT INTEGER(A-Z)
00800         REAL FRACT,TUX
00900         DIMENSION ICNT(100),OCNT(100),TYPE(100),INPUT(100,10)
01000        1,TVALUE(100),FT(100,200),FG(100),NFT(100),X(1024,10)
01100        2,JFT(100,200),NFTJ(100),CFT(100,200),NCFT(100),FC(200)
01200        3,OUT(100,10),PFT(25,100),NPFT(25),AFT(25,100),NAFT(25)
01300         COMMON INPUT,OUT,FT,ICNT,OCNT,NFT,FG/AREA1/TYPE,NUGA,PRINCN
01400        1,INTCNT,PROTCN,NODNUM/AREA2/TVALUE,I/AREA3/JFT,NFTJ/AREA4/
01500        2CFT,NCFT/AREA5/NDE
01600         READ(21,*),XIN
01700         READ(21,*)((X(I,J),J=1,PRINCN),I=1,XIN)
01800         WRITE(22,80)
01900 80      FORMAT(5X,'RESULTS FOR DEDUCTIVE SIMULATION'/5X,'*
02000        1*')
02100         WRITE(22,90)
02200 90      FORMAT(//5X,'TESTS SIMULATED'/5X,'================')
02300         DO 110 S=1,XIN
02400         WRITE(22,100),S,(X(S,J),J=1,PRINCN)
02500 100     FORMAT(5X,'T',I2,' : ',15(I2)/)
02600 110     CONTINUE
02700         WRITE(22,98)
02800 98      FORMAT(//5X,'FAULTS DETECTION INFORMATION'/5X,'==========
02900        1====================='/)
03000         PUT=PRINCN+INTCNT
03100         CALL TIMER(TIME1)
03200         DO 1000 Y=1,XIN
03300         DO 280 J=1,PRINCN
03400         TVALUE(J)=X(Y,J)
03500         NFT(J)=1
03600         IF(TVALUE(J).EQ.0)GO TO 260
03700         FG(J)=2*J-1
```

```
03800          GO TO 263
03900 260      FG(J)=2*J
04000 263      FT(J,1)=FG(J)
04100 280      CONTINUE
04200          I=1
04300 300      GO TO (310,312,314,316,318,320,322,324),TYPE(I)
04400 310      CALL DAND
04500          GO TO 340
04600 312      CALL DOR
04700          GO TO 340
04800 314      CALL DNAND
04900          GO TO 340
05000 316      CALL DNOR
05100          GO TO 340
05200 318      CALL DXOR
05300          GO TO 340
05400 320      CALL DXNOR
05500          GO TO 340
05600 322      CALL DNOT
05700          GO TO 340
05800 324      CALL DCONT
05900 340      I=I+1
06000          IF(I.LE.NOGA)GO TO 300
06100          WRITE(22,499),Y
06200 499      FORMAT(/5X,'FAULTS DETECTED FOR T',I2/5X,'----------
06300         1----------')
06400          DO 550 P=1,PROTCN
06500          B=POT+P
06600          IF(Y.GT.1)GO TO 388
06700          DO 488 H=1,NFT(B)
06800          PFT(P,H)=FT(B,H)
06900          FC(H)=FT(B,H)
07000 488      CONTINUE
07100          NPFT(P)=NFT(B)
07200          GO TO 288
07300 388      NAFT(P)=0
07400          DO 358 GG=1,NFT(B)
```

```
07500          DO 348 HH=1,NPFT(P)
07600          IF(PFT(P,HH).EQ.FT(B,GG))GO TO 358
07700 348      CONTINUE
07800          NAFT(P)=NAFT(P)+1
07900          AFT(P,NAFT(P))=FT(B,GG)
08000 358      CONTINUE
08100          IF(NAFT(P).EQ.0)GO TO 721
08200          DO 689 E=1,NAFT(P)
08300          PFT(P,NPFT(P)+E)=AFT(P,E)
08400 689      CONTINUE
08500          NPFT(P)=NPFT(P)+NAFT(P)
08600          WRITE(22,699),P,(AFT(P,C),C=1,NAFT(P))
08700 699      FORMAT(/5X,'ADDITIONAL FAULTS DETECTED AT Z',I3,' ARE ='
08800          1,25(I3,','))
08900          GO TO 550
09000 721      WRITE(22,722),P
09100          GO TO 550
09200 722      FORMAT(/5X,'NO ADDITIONAL FAULTS DETECTED AT Z',I2)
09300 288      WRITE(22,500),P,(FT(B,C),C=1,NFT(B))
09400 500      FORMAT(/5X,'FAULT NO.S DETECTED AT Z',I3,' ARE =',25(I3,','))
09500 550      CONTINUE
09600 1000     CONTINUE
09700          CALL TIMER(TIME2)
09800          TIME=TIME2-TIME1
09900          WRITE(22,1001),XIN,TIME
10000 1001     FORMAT(//5X,'TIME TAKEN TO SIMULATE ',I2,' TESTS =',I5,' MS'/
10100          15X,'==============================')
10200          RETURN
10300          END
10400 C=========================================================
10500 C        ROUTINE FOR AND GATE
10600 C=Z=======================================================
10700          SUBROUTINE DAND
10800          IMPLICIT INTEGER(A-Z)
10900          DIMENSION INPUT(100,10),OUT(100,10),FT(100,200),ICNT(100),
11000          1OCNT(100),NFT(100),FG(100),TVALUE(100),CFT(100,200),
11100          2NCFT(100),JFT(100,200),NFTJ(100)
```

```
11200          COMMON INPUT,OUT,FT,ICNT,OCNT,NFT,FG/AREA2/TVALUE,I/AREA3/JFT,
11300         1NFTJ/AREA4/CFT,NCFT/AREA5/NDE
11400          DO 10 J=1,ICNT(I)
11500          IF(TVALUE(INPUT(I,J)).EQ.0)GO TO 50
11600 10       CONTINUE
11700          TVALUE(OUT(I,1))=1
11800          CALL COMPO
11900          CALL UNION
12000          FG(OUT(I,1))=2*OUT(I,1)-1
12100          GO TO 90
12200 50       TVALUE(OUT(I,1))=0
12300          CALL COMP1
12400          CALL INTER
12500          FG(OUT(I,1))=2*OUT(I,1)
12600 90       NFT(OUT(I,1))=NFT(OUT(I,1))+1
12700          FT(OUT(I,1),NFT(OUT(I,1)))=FG(OUT(I,1))
12800          RETURN
12900          END
13000 C=========================================================
13100
13200 C        ROUTINE TO SIMULATE OR GATE
13300 C=========================================================
13400          SUBROUTINE OOR
13500          IMPLICIT INTEGER(A-Z)
13600          DIMENSION INPUT(100,10),OUT(100,10),FT(100,200),ICNT(100),
13700         1OCNT(100),NFT(100),FG(100),TVALUE(100),CFT(100,200),
13800         2NCFT(100),JFT(100,200),NFTJ(100)
13900          COMMON INPUT,OUT,FT,ICNT,OCNT,NFT,FG/AREA2/TVALUE,I/AREA3/JFT,
14000         1NFTJ/AREA4/CFT,NCFT/AREA5/NDE
14100          DO 10 J=1,ICNT(I)
14200          IF(TVALUE(INPUT(I,J)).EQ.1)GO TO 50
14300 10       CONTINUE
14400          TVALUE(OUT(I,1))=0
14500          CALL COMP1
14600          CALL UNION
14700          FG(OUT(I,1))=2*OUT(I,1)
14800          GO TO 90
```

```
14900 50      TVALUE(OUT(I,1))=1
15000         CALL COMPO
15100         CALL INTER
15200         FG(OUT(I,1))=2*OUT(I,1)-1
15300 90      NFT(OUT(I,1))=NFT(OUT(I,1))+1
15400         FT(OUT(I,1),NFT(OUT(I,1)))=FG(OUT(I,1))
15500         RETURN
15600         END
15700 C=======================================================
15800 C       ROUTINE TO SIMULATE NAND GATE
15900 C=======================================================
16000         SUBROUTINE DNAND
16100         IMPLICIT INTEGER(A-Z)
16200         DIMENSION INPUT(100,10),OUT(100,10),FT(100,200),ICNT(100),
16300        1OCNT(100),NFT(100),FG(100),TVALUE(100),CFT(100,200),
16400        2NCFT(100),JFT(100,200),NFTJ(100)
16500         COMMON INPUT,OUT,FT,ICNT,OCNT,NFT,FG/AREA2/TVALUE,I/AREA3/JFT,
16600        1NFTJ/AREA4/CFT,NCFT/AREA5/NDE
16700         DO 10 J=1,ICNT(I)
16800         IF(TVALUE(INPUT(I,J)).EQ.0)GO TO 50
16900 10      CONTINUE
17000         TVALUE(OUT(I,1))=0
17100         CALL COMPO
17200         CALL UNION
17300         FG(OUT(I,1))=2*OUT(I,1)
17400         GO TO 90
17500 50      TVALUE(OUT(I,1))=1
17600         CALL COMP1
17700         CALL INTER
17800         FG(OUT(I,1))=2*OUT(I,1)-1
17900 90      NFT(OUT(I,1))=NFT(OUT(I,1))+1
18000         FT(OUT(I,1),NFT(OUT(I,1)))=FG(OUT(I,1))
18100         RETURN
18200         END
      C=======================================================
      C       ROUTINE TO SIMULATE NOR GATE
      C=======================================================
```

```
18600          SUBROUTINE DNOR
18700          IMPLICIT INTEGER(A-Z)
18800          DIMENSION INPUT(100,10),OUT(100,10),FT(100,200),ICNT(100),
18900          1OCNT(100),NFT(100),FG(100),TVALUE(100),CFT(100,200),
19000          2NCFT(100),JFT(100,200),NFTJ(100)
19100          COMMON INPUT,OUT,FT,ICNT,OCNT,NFT,FG/AREA2/TVALUE,I/AREA3/JFT,
19200          1NFTJ/AREA4/CFT,NCFT/AREA5/NDE
19300          DO 10 J=1,ICNT(1)
19400          IF(TVALUE(INPUT(I,J)).EQ.1)GO TO 50
19500 10       CONTINUE
19600          TVALUE(OUT(1,1))=1
19700          CALL COMP1
19800          CALL UNION
19900          FG(OUT(1,1))=2*OUT(1,1)-1
20000          GO TO 90
20100 50       TVALUE(OUT(1,1))=0
20200          CALL COMP0
20300          CALL INTER
20400          FG(OUT(1,1))=2*OUT(1,1)
20500 90       NFT(OUT(I,1))=NFT(OUT(I,1))+1
20600          FT(OUT(1,1),NFT(OUT(I,1)))=FG(OUT(I,1))
20700          RETURN
20800          END
20900 C=========================================================
21000 C        ROUTINE TO SIMULATE CONTINUITY BLOCK
21100 C=========================================================
21200          SUBROUTINE DCONT
21300          IMPLICIT INTEGER(A-Z)
21400          DIMENSION INPUT(100,10),OUT(100,10),FT(100,200),ICNT(100),
21500          1OCNT(100),NFT(100),FG(100),TVALUE(100),CFT(100,200),
21600          2NCFT(100),JFT(100,200),NFTJ(100)
21700          COMMON INPUT,OUT,FT,ICNT,OCNT,NFT,FG/AREA2/TVALUE,I/AREA3/JFT,
21800          1NFTJ/AREA4/CFT,NCFT/AREA5/NDE
21900          DO 25 J1=1,OCNT(1)
22000          NFT(OUT(I,J1))=NFT(INPUT(1,1))
22100          DO 15 J2=1,NFT(INPUT(I,1))
22200          FT(OUT(1,J1),J2)=FT(INPUT(I,1),J2)
```

```
22300 15      CONTINUE
22400 25      CONTINUE
22500         IF(TVALUE(INPUT(I,1)).EQ.0)GO TO 50
22600         DO 10 J=1,OCNT(I)
22700         TVALUE(OUT(I,J))=TVALUE(INPUT(I,1))
22800         FG(OUT(I,J))=2*OUT(I,J)-1
22900         NFT(OUT(I,J))=NFT(OUT(I,J))+1
23000         FT(OUT(I,J),NFT(OUT(I,J)))=FG(OUT(I,J))
23100 10      CONTINUE
23200         GO TO 60
23300 50      DO 20 J=1,OCNT(I)
23400         TVALUE(OUT(I,J))=TVALUE(INPUT(I,1))
23500         FG(OUT(I,J))=2*OUT(I,J)
23600         NFT(OUT(I,J))=NFT(OUT(I,J))+1
23700         FT(OUT(I,J),NFT(OUT(I,J)))=FG(OUT(I,J))
23800 20      CONTINUE
23900 60      RETURN
24000         END
24100 C=======================================================================
24200         SUBROUTINE DNOT
24300 C=======================================================================
24400         IMPLICIT INTEGER(A-Z)
24500         DIMENSION INPUT(100,10),OUT(100,10),FT(100,200),ICNT(100),
24600        1OCNT(100),NFT(100),FG(100),TVALUE(100),CFT(100,200),
24700        2NCFT(100),JFT(100,200),NFTJ(100)
24800         COMMON INPUT,OUT,FT,ICNT,OCNT,NFT,FG/AREA2/TVALUE,I/AREA3/JFT,
24900        1NFTJ/AREA4/CFT,NCFT/AREA5/NDE
25000         NFT(OUT(1,1))=NFT(INPUT(1,1))
25100         IF(TVALUE(INPUT(I,1)).EQ.0)GO TO 10
25200         TVALUE(OUT(1,1))=0
25300         FG(OUT(1,1))=2*OUT(1,1)
25400         GO TO 20
25500 10      TVALUE(OUT(1,1))=1
25600         FG(OUT(1,1))=2*OUT(1,1)-1
25700 20      DO 40 K=1,NFT(OUT(1,1))
25800         FT(OUT(1,1),K)=FT(INPUT(1,1),K)
25900 40      CONTINUE
```

```
26000          NFT(OUT(I,1))=NFT(OUT(I,1))+1
26100          FT(OUT(1,1),NFT(OUT(I,1)))=FG(OUT(I,1))
26200          RETURN
26300          END
26400 C=======================================================================
26500          SUBROUTINE DXOR
26600 C=======================================================================
26700          IMPLICIT INTEGER(A-Z)
26800          DIMENSION INPUT(100,10),OUT(100,10),FT(100,200),ICNT(100),
26900          1OCNT(100),NFT(100),FG(100),TVALUE(100),CFT(100,200),
27000          2NCFT(100),JFT(100,200),NFTJ(100)
27100          COMMON INPUT,OUT,FT,ICNT,OCNT,NFT,FG/AREA2/TVALUE,I/AREA3/JFT,
27200          1NFTJ/AREA4/CFT,NCFT/AREA5/NDE
27300          N=0
27400          DO 20 K=1,ICNT(I)
27500          NDE=INPUT(I,K)
27600          DO 5 A=1,NFT(NDE)
27700          JFT(NDE,A)=FT(NDE,A)
27800 5        CONTINUE
27900          NFTJ(NDE)=NFT(NDE)
28000          IF(TVALUE(NDE).EQ.0)N=N+1
28100 20       CONTINUE
28200          IF((N.EQ.0).OR.(N.EQ.2))TVALUE(OUT(1,1))=0
28300          IF(N.EQ.1)TVALUE(OUT(I,1))=1
28400          CALL UNION
28500          IF(TVALUE(OUT(I,1)).EQ.1)GO TO 50
28600          FG(OUT(1,1))=2*OUT(1,1)
28700          GO TO 60
28800 50       FG(OUT(1,1))=2*OUT(I,1)-1
28900 60       NFT(OUT(I,1))=NFT(OUT(I,1))+1
29000          FT(OUT(1,1),NFT(OUT(I,1)))=FG(OUT(1,1))
29100          RETURN
29200          END
29300 C=======================================================================
29400          SUBROUTINE DXNOR
29500 C=======================================================================
29600          IMPLICIT INTEGER(A-Z)
```

```
29700          DIMENSION INPUT(100,10),OUT(100,10),FT(100,200),ICNT(100),
29800          1OCNT(100),NFT(100),FG(100),TVALUE(100),CFT(100,200),
29900          2NCFT(100),JFT(100,200),NFTJ(100)
30000          COMMON INPUT,OUT,FT,ICNT,OCNT,NFT,FG/AREA2/TVALUE,I/AREA3/JFT,
30100          1NFTJ/AREA4/CFT,NCFT/AREA5/NDE
30200          N=0
30300          DO 20 K=1,ICNT(1)
30400          NDE=INPUT(1,K)
30500          DO 5 A=1,NFT(NDE)
30600          JFT(NDE,A)=FT(NDE,A)
30700 5        CONTINUE
30800          NFTJ(NDE)=NFT(NDE)
30900          IF(TVALUE(NDE).EQ.0)N=N+1
31000 20       CONTINUE
31100          IF((N.EQ.0).OR.(N.EQ.2))TVALUE(OUT(1,1))=1
31200          IF(N.EQ.1)TVALUE(OUT(1,1))=0
31300          CALL UNION
31400          IF(TVALUE(OUT(I,1)).EQ.1)GO TO 50
31500          FG(OUT(1,1))=2*OUT(1,1)
31600          GO TO 60
31700 50       FG(OUT(1,1))=2*OUT(1,1)-1
31800 60       NFT(OUT(1,1))=NFT(OUT(I,1))+1
31900          FT(OUT(1,1),NFT(OUT(I,1)))=FG(OUT(I,1))
32000          RETURN
32100          END
32200 C=========================================================
32300 C        ROUTINE FOR UNION OF FAULT LISTS
32400 C=========================================================
32500          SUBROUTINE UNION
32600          IMPLICIT INTEGER(A-Z)
32700          DIMENSION INPUT(100,10),OUT(100,10),ICNT(100),OCNT(100),
32800          1FT(100,200),FG(100),NFT(100),UFT(200),JFT(100,200),NFTJ(100),
32900          2TVALUE(100)
33000          COMMON INPUT,OUT,FT,ICNT,OCNT,NFT,FG/AREA2/TVALUE,I/AREA3/JFT,NF
33100          NODE1=INPUT(1,1)
33200          DO 140 J=2,ICNT(I)
33300          NODE2=INPUT(1,J)
```

```
33400           IF(NODE1.EQ.0)GO TO 20
33500           DO 10 K=1,NFTJ(NODE1)
33600           UFT(K)=JFT(NODE1,K)
33700 10        CONTINUE
33800           UNFT=NFTJ(NODE1)
33900 20        DO 40 L=1,NFTJ(NODE2)
34000           DO 30 K=1,UNFT
34100           IF(JFT(NODE2,L).EQ.UFT(K))GO TO 40
34200 30        CONTINUE
34300           UNFT=UNFT+1
34400           UFT(UNFT)=JFT(NODE2,L)
34500 40        CONTINUE
34600           NODE1=0
34700 140       CONTINUE
34800           NFT(OUT(I,1))=UNFT
34900           DO 150 K=1,NFT(OUT(I,1))
35000           FT(OUT(I,1),K)=UFT(K)
35100 150       CONTINUE
35200           RETURN
35300           END
35400 C=======================================================================
35500 C         ROUTINE FOR INTERSECTION OF FAULT LISTS
35600 C=======================================================================
35700           SUBROUTINE INTER
35800           IMPLICIT INTEGER(A-Z)
35900           DIMENSION INPUT(100,10),OUT(100,10),ICNT(100),OCNT(100),
36000           1FT(100,200),FG(100),NFT(100),IFT(200),JFT(100,200),NFTJ(100),
36100           2TVALUE(100)
36200           COMMON INPUT,OUT,FT,ICNT,OCNT,NFT,FG/AREA2/TVALUE,I/AREA3/JFT,NFT
36300           NODE1=INPUT(I,1)
36400           DO 140 J=2,ICNT(I)
36500           NODE2=INPUT(I,J)
36600           JNFT=0
                IF(NODE1.EQ.0)GO TO 10
                DO 5 K=1,NFTJ(NODE1)
                IFT(K)=JFT(NODE1,K)
                CONTINUE
```

```
37100           INFT=NFTJ(NODE1)
37200 10        DO 120 K=1,INFT
37300           DO 20 L=1,NFTJ(NODE2)
37400           IF(IFT(K).EQ.JFT(NODE2,L))GO TO 110
37500 20        CONTINUE
37600           GO TO 120
37700 110       JNFT=JNFT+1
37800           IFT(JNFT)=JFT(NODE2,L)
37900 120       CONTINUE
38000           INFT=JNFT
38100           NODE1=0
38200 140       CONTINUE
38300           NFT(OUT(I,1))=INFT
38400           DO 150 K=1,NFT(OUT(I,1))
38500           FT(OUT(I,1),K)=IFT(K)
38600 150       CONTINUE
38700           RETURN
38800           END
38900 C=======================================================================
39000 C         ROUTINE FOR COMPLIMENT OF FAULT LIST
39100 C=======================================================================
39200           SUBROUTINE COMPLE
39300           IMPLICIT INTEGER(A-Z)
39400           DIMENSION INPUT(100,10),OUT(100,10),FT(100,200),ICNT(100),
39500           1OCNT(100),NFT(100),FG(100),TVALUE(100),CFT(100,200),
39600           2NCFT(100),JFT(100,200),NFTJ(100)
39700           COMMON INPUT,OUT,FT,ICNT,OCNT,NFT,FG/AREA2/TVALUE,I/AREA3/JFT,
39800           1NFTJ/AREA4/CFT,NCFT/AREA5/NDE
39900           T=0
40000           DO 282 G=1,ICNT(I)
40100           IF(INPUT(I,G).GT.T)T=INPUT(I,G)
40200 282       CONTINUE
40300           DO 5 K=1,2*T
40400           CFT(NDE,K)=K
40500 5         CONTINUE
40600           NCFT(NDE)=2*T
40700           DO 30 L=1,NFT(NDE)
```

```
40800           DO 20 K=1,NCFT(NDE)
40900           IF(FT(NDE,L).EQ.CFT(NDE,K))GO TO 10
41000           GO TO 20
41100 10        NCFT(NDE)=NCFT(NDE)-1
41200           DO 15 M=K,NCFT(NDE)
41300           CFT(NDE,M)=CFT(NDE,M+1)
41400 15        CONTINUE
41500           GO TO 30
41600 20        CONTINUE
41700 30        CONTINUE
41800           RETURN
41900           END
42000 C=========================================================
42100 C         ROUTINE COMPO
42200 C=========================================================
42300           SUBROUTINE COMPO
42400           IMPLICIT INTEGER(A-Z)
42500           DIMENSION INPUT(100,10),OUT(100,10),FT(100,200),ICNT(100),
42600           1OCNT(100),NFT(100),FG(100),TVALUE(100),CFT(100,200),
42700           2NCFT(100),JFT(100,200),NFTJ(100)
42800           COMMON INPUT,OUT,FT,ICNT,OCNT,NFT,FG/AREA2/TVALUE,I/AREA3/JFT,
42900           1NFTJ/AREA4/CFT,NCFT/AREA5/NDE
43000           DO 20 K=1,ICNT(I)
43100           NDE=INPUT(I,K)
43200           IF(TVALUE(INPUT(I,K)).EQ.0)GO TO 50
43300           DO 5 A=1,NFT(NDE)
43400           JFT(NDE,A)=FT(NDE,A)
43500 5         CONTINUE
43600           NFTJ(NDE)=NFT(NDE)
43700           GO TO 20
43800 50        CALL COMPLE
43900           DO 15 A=1,NCFT(NDE)
44000           JFT(NDE,A)=CFT(NDE,A)
44100 15        CONTINUE
44200           NFTJ(NDE)=NCFT(NDE)
44300 20        CONTINUE
44400           RETURN
```

```
44500           END
44600 C=========================================================
44700 C         ROUTINE COMP1
44800 C=========================================================
44900           SUBROUTINE COMP1
45000           IMPLICIT INTEGER(A-Z)
45100           DIMENSION INPUT(100,10),OUT(100,10),FT(100,200),ICNT(100),
45200           1OCNT(100),NFT(100),FG(100),TVALUE(100),CFT(100,200),
45300           2NCFT(100),JFT(100,200),NFTJ(100)
45400           COMMON INPUT,OUT,FT,ICNT,OCNT,NFT,FG/AREA2/TVALUE,I/AREA3/JFT,
45500           1NFTJ/AREA4/CFT,NCFT/AREA5/NDE
45600           DO 20 K=1,ICNT(I)
45700           NDE=INPUT(I,K)
45800           IF(TVALUE(INPUT(I,K)).EQ.1)GO TO 50
45900           DO 5 A=1,NFT(NDE)
46000           JFT(NDE,A)=FT(NDE,A)
46100 5         CONTINUE
46200           NFTJ(NDE)=NFT(NDE)
46300           GO TO 20
46400 50        CALL COMPLE
46500 10        DO 15 A=1,NCFT(NDE)
46600           JFT(NDE,A)=CFT(NDE,A)
46700 15        CONTINUE
46800           NFTJ(NDE)=NCFT(NDE)
46900 20        CONTINUE
47000           RETURN
47100           END
```

CRITICAL PATH

```
C=====================================================================
      CONE MODELLING
C=====================================================================
37    READ(23,*),NOCO
      READ(23,56)((CONDES(J,I),I=1,50),J=1,NOCO)
56    FORMAT(50A1)
      DO 143 J=1,NOCO
      DO 142 I=1,50
      DO 141 G=1,10
      IF(CONDES(J,I).EQ.INDEX(G))GO TO 138
      GO TO 141
138   CONDES(J,I)=G-1
      DO 139 A=1,10
      IF(CONDES(J,I+1).EQ.INDEX(A))CONDES(J,I)=CONDES(J,I)*10+(A-1)
139   CONTINUE
141   CONTINUE
142   CONTINUE
143   CONTINUE
      DO 1200 I=1,NOCO
      D=1
      J=0
      N=1
1100  J=J+1
      IF(CONDES(I,J).NE.'/')GO TO 1010
      CONCON(I,D,N)=CONDES(I,J)
      D=D+1
      IF(D.EQ.4)GO TO 1200
      N=1
      GO TO 1100
1010  CONCON(I,D,N)=CONDES(I,J)
      N=N+1
      GO TO 1100
1200  CONTINUE
      DO 1900 I=1,NOCO
      ICON(1)=0
      DO 1800 D=2,3
      N=0
```

```
      N=N+1
      IF(CONCON(I,D,N).EQ.'X')GO TO 1400
      IF (CONCON(I,D,N).EQ.'C')GO TO 1500
      IF(CONCON(I,D,N).EQ.'Z')GO TO 1700
      IF(CONCON(I,D,N).EQ.'/')GO TO 1800
      GO TO 1300
1400  N=N+1
      ICON(I)=ICON(I)+1
      CINPUT(I,ICON(I))=CONCON(I,D,N)
      GO TO 1300
1500  N=N+1
      IF(D.EQ.3)GO TO 1600
      ICON(I)=ICON(I)+1
      CINPUT(I,ICON(I))=CONCON(I,D,N)+PRINCN
      GO TO 1300
1600  COUT(I)=CONCON(I,D,N)+PRINCN
      GO TO 1300
1700  N=N+1
      COUT(I)=CONCON(I,D,N)+PRINCN+INTCNT
      GO TO 1300
1800  CONTINUE
1900  CONTINUE
1901  NODNUM=PRINCN+INTCNT+PROTCN
      DO 300 I=1,NODNUM
      INODE(I)=0
      ICONE(I)=0
      OCONE(I)=0
      ONODE(I)=0
      DO 295 K=1,NOGA
      DO 265 L=1,ICNT(K)
      IF(I.EQ.INPUT(K,L))INODE(I)=K
265   CONTINUE
      DO 275 L=1,OCNT(K)
      IF(I.EQ.OUT(K,L))ONODE(I)=K
275   CONTINUE
      CONTINUE
      DO 298 KK=1,NOCO
```

```
C7500          DO 297 LL=1,ICON(KK)
C7600          IF(I.EQ.CINPUT(KK,LL))ICONE(I)=KK
07700 297      CONTINUE
07800          IF(I.EQ.COUT(KK))OCONE(I)=KK
07900 298      CONTINUE
08000 300      CONTINUE
08100          DO 302 I=1,NODNUM
08200          NC(I)=0
08300 302      CONTINUE
08400          DO 350 CONO=1,NOCO
08500          NDE=COUT(CONO)
08600          NC(NDE)=NC(NDE)+1
08700          CONE(NDE,NC(NDE))=CONO
08800          I=0
08900 315      I=I+1
09000          HK(I)=ONODE(NDE)
09100          JJ(HK(I))=0
09200 320      JJ(HK(I))=JJ(HK(I))+1
09300          NDE=INPUT(HK(I),JJ(HK(I)))
09400          NC(NDE)=NC(NDE)+1
09500          CONE(NDE,NC(NDE))=CONO
09600          DO 330 L=1,ICON(CONO)
09700          IF(NDE.NE.CINPUT(CONO,L))GO TO 330
09800          GO TO 340
09900 330      CONTINUE
10000          GO TO 315
10100 335      I=I-1
10200 340      IF(JJ(HK(I)).LT.ICNT(HK(I)))GO TO 320
10300 345      IF(I.GT.1)GO TO 335
10400 350      CONTINUE
10500          CALL FFSIMU
10600          STOP
10700          END
10750
10775
10787
10793
```

```
C=============================================================
      SUBROUTINE FFSIMU
C=============================================================
      IMPLICIT INTEGER(A-Z)
      DIMENSION TYPE(100),INPUT(100,10),LEVEL(100),OUT(100,10)
     1,TVALUE(100),X(1024,10),AFT(25,100),NAFT(25),PFT(25,100)
     2,NPFT(25),NFT(25),FT(25,100),FG(100),CINPUT(25,25),COUT(25)
     3,ICON(25),INODE(100),ONODE(100),ICNT(100),FC(200)
     4,NC(100),OCNT(100),STINPT(100),ICONE(100),OCONE(100),CONE(100,1
     5,CRIT(100),PROP(100),EXTD(25),STEMCK(25)
      COMMON INPUT,OUT,CINPUT,COUT,ICON,INODE,ONODE,
     1ICONE,OCONE,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN
     2,NODNUM,NOCO,NOGA/AREA2/TVALUE,I/AREA3/LEVEL,STINPT,EXTD,STEMCK
     3,CRIT,PROP,NSTMCK,NDE,FTYNOD/AREA4/FT,FG,NFT
      READ(21,*),XIN
      READ(21,*)((X(I,J),J=1,PRINCN),I=1,XIN)
      WRITE(22,80)
 80   FORMAT(5X,'RESULTS FOR CRITICAL TRACING'/5X,'************
     1*****************')
      WRITE(22,90)
 90   FORMAT(5X,'TESTS SIMULATED'/5X,'================'//)
      DO 110 S=1,XIN
      WRITE(22,100),S,(X(S,J),J=1,PRINCN)
 100  FORMAT(5X,'T',I2,' : ',15(I2)/)
 110  CONTINUE
      WRITE(22,98)
 98   FORMAT(//5X,'FAULT DETECTION INFORMATION'/5X,'============
     1================')
      POT=PRINCN+INTCNT
      CALL TIMER(TIME1)
      DO 1000 Y=1,XIN
      DO 254 K=1,NODNUM
      STINPT(K)=0
 254  CONTINUE
      DO 280 J=1,PRINCN
      LEVEL(J)=1
      TVALUE(J)=X(Y,J)
```

CRITICAL PATH

```
14700           IF(TVALUE(J).EQ.0)GO TO 260
14800           FG(J)=2*J-1
14900           GO TO 280
15000 260       FG(J)=2*J
15100 280       CONTINUE
15200           I=1
15300 300       GO TO (310,312,314,316,318,320,322,324),TYPE(I)
15400 310       CALL DAND
15500           GO TO 340
15600 312       CALL DOR
15700           GO TO 340
15800 314       CALL DNAND
15900           GO TO 340
16000 316       CALL DNOR
16100           GO TO 340
16200 318       CALL DXOR
16300           GO TO 340
16400 320       CALL DXNOR
16500           GO TO 340
16600 322       CALL DNOT
16700           GO TO 340
16800 324       CALL CONT
16900 340       I=I+1
17000           IF(I.LE.NOGA)GO TO 300
17100           CALL TRACE
17200           WRITE(22,499),Y
17300 499       FORMAT(/5X,'FAULTS DETECTED FOR T',I2/1X,'----------
17400          1----------')
17500           DO 550 P=1,PROTCN
17600           IF(Y.GT.1)GO TO 388
17700           DO 488 H=1,NFT(P)
17800           PFT(P,H)=FT(P,H)
17900           FC(H)=FT(P,H)
18000 488       CONTINUE
18100           NPFT(P)=NFT(P)
18200           GO TO 288
18300 388       NAFT(P)=0
```

```
18400           DO 358 GG=1,NFT(P)
18500           DO 348 HH=1,NPFT(P)
18600           IF(PFT(P,HH).EQ.FT(P,GG))GO TO 358
18700 348       CONTINUE
18800           NAFT(P)=NAFT(P)+1
18900           AFT(P,NAFT(P))=FT(P,GG)
19000 358       CONTINUE
19100           IF(NAFT(P).EQ.0)GO TO 721
19200           DO 689 E=1,NAFT(P)
19300           PFT(P,NPFT(P)+E)=AFT(P,E)
19400 689       CONTINUE
19500           NPFT(P)=NPFT(P)+NAFT(P)
19600           WRITE(22,699),P,(AFT(P,C),C=1,NAFT(P))
19700 699       FORMAT(5X,'ADDITIONAL FAULTS DETECTED AT Z',I2,' ARE ='
19800           1,25(I3,','))
19900           GO TO 550
20000 721       WRITE(22,722),P
20100           GO TO 550
20200 722       FORMAT(5X,'NO ADDITIONAL FAULTS DETECTED AT Z',I2)
20300 288       WRITE(22,500),P,(FT(P,C),C=1,NFT(P))
20400 500       FORMAT(5X,'FAULT NO.S DETECTED AT Z',I2,' ARE =',25(I3,','))
20500 550       CONTINUE
20600 1000      CONTINUE
20700           CALL TIMER(TIME2)
20800           TIME=TIME2-TIME1
20900           WRITE(22,914),XIN,TIME
21000 914       FORMAT(5X,'TIME TAKEN TO SIMULATE ',I2,' TESTS =',I5,' MS'/5X,'
21100           1=============================')
21200           RETURN
21300           END
21400 C=========================================================
21500 C  **    ROUTINE TO SIMULATE AND GATE
21600 C=========================================================
21700           SUBROUTINE DAND
21800           IMPLICIT INTEGER(A-Z)
21900           DIMENSION TYPE(100),INPUT(100,10),LEVEL(100),OUT(100,10)
22000           1,TVALUE(100)
```

```
      2,NFT(25),FT(25,100),FG(100),CINPUT(25,25),COUT(25)
      3,ICON(25),INODE(100),ONODE(100),ICNT(100)
      4,NC(100),OCNT(100),STINPT(100),ICONE(100),OCONE(100),CONE(100,1
      5,CRIT(100),PROP(100),EXTD(25),STEMCK(25)
      COMMON INPUT,OUT,CINPUT,COUT,ICON,INODE,ONODE,
     1ICONE,OCONE,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PRUTCN
     2,NODNUM,NOCO,NOGA/AREA2/TVALUE,I/AREA3/LEVEL,STINPT,EXTD,STEMCK
     3,CRIT,PROP,NSTMCK,NDE,FTYNOD/AREA4/FT,FG,NFT
      DO 10 J=1,ICNT(I)
      IF(MAXLEV.LT.LEVEL(INPUT(I,J)))MAXLEV=LEVEL(INPUT(I,J))
      IF(TVALUE(INPUT(I,J)).EQ.1)GO TO 10
      K=J
      N=N+1
10    CONTINUE
      IF(N.EQ.0)GO TO 110
      IF(N.GT.1)GO TO 105
      STINPT(INPUT(I,K))=1
105   TVALUE(OUT(I,1))=0
      FG(OUT(I,1))=2*OUT(I,1)
      GO TO 130
110   TVALUE(OUT(I,1))=1
      FG(OUT(I,1))=2*OUT(I,1)-1
      DO 120 J=1,ICNT(I)
      STINPT(INPUT(I,J))=1
120   CONTINUE
130   LEVEL(OUT(I,1))=MAXLEV+1
      RETURN
      END
C=====================================================================
C  **   ROUTINE TO SIMULATE NAND GATE
C=====================================================================
      SUBROUTINE DNAND
      IMPLICIT INTEGER(A-Z)
      DIMENSION TYPE(100),INPUT(100,10),LEVEL(100),OUT(100,10)
     1,TVALUE(100)
      2,NFT(25),FT(25,100),FG(100),CINPUT(25,25),COUT(25)
      3,ICON(25),INODE(100),ONODE(100),ICNT(100)
```

```
25800          4,NC(100),OCNT(100),STINPT(100),ICONE(100),OCONE(100),CONE(100,10
25900          5,CRIT(100),PROP(100),EXTD(25),STEMCK(25)
26000          COMMON INPUT,OUT,CINPUT,COUT,ICON,INODE,ONODE,
26100          1ICONE,OCONE,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN
26200          2,NODNUM,NOGO,NOGA/AREA2/TVALUE,I/AREA3/LEVEL,STINPT,EXTD,STEMCK
26300          3,CRIT,PROP,NSTMCK,NDE,FTYNOD/AREA4/FT,FG,NFT
26400          N=0
26500          MAXLEV=0
26600          DO 10 J=1,ICNT(I)
26700          IF(MAXLEV.LT.LEVEL(INPUT(I,J)))MAXLEV=LEVEL(INPUT(I,J))
26800          IF(TVALUE(INPUT(I,J)).EQ.1)GO TO 10
26900          K=J
27000          N=N+1
27100 10       CONTINUE
27200          IF(N.EQ.0)GO TO 110
27300          IF(N.GT.1)GO TO 105
27400          STINPT(INPUT(I,K))=1
27500 105      TVALUE(OUT(I,1))=1
27600          FG(OUT(I,1))=2*OUT(I,1)-1
27700          GO TO 130
27800 110      TVALUE(OUT(I,1))=0
27900          FG(OUT(I,1))=2*OUT(I,1)
28000          DO 120 J=1,ICNT(I)
28100          STINPT(INPUT(I,J))=1
28200 120      CONTINUE
28300 130      LEVEL(OUT(I,1))=MAXLEV+1
28400          RETURN
28500          END
28600 C=======================================================
28700 C  **    ROUTINE TO SIMULATE OR GATE
28800 C=======================================================
28900          SUBROUTINE OOR
29000          IMPLICIT INTEGER(A-Z)
29100          DIMENSION TYPE(100),INPUT(100,10),LEVEL(100),OUT(100,10)
29200          1,TVALUE(100)
29300          2,NFT(25),FT(25,100),FG(100),CINPUT(25,25),COUT(25)
29400          3,ICON(25),INODE(100),ONODE(100),ICNT(100)
```

```
      4,NC(100),OCNT(100),STINPT(100),ICONE(100),OCONE(100),CONE(100,
      5,CRIT(100),PROP(100),EXTD(25),STEMCK(25)
      COMMON INPUT,OUT,CINPUT,COUT,ICON,INODE,ONODE,
      1ICONE,OCONE,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN
      2,NODNUM,NOCO,NOGA/AREA2/TVALUE,I/AREA3/LEVEL,STINPT,EXTD,STEMCK
      3,CRIT,PROP,NSTMCK,NDE,FTYNOD/AREA4/FT,FG,NFT
      N=0
      MAXLEV=0
      DO 10 J=1,ICNT(I)
      IF(MAXLEV.LT.LEVEL(INPUT(I,J)))MAXLEV=LEVEL(INPUT(I,J))
      IF(TVALUE(INPUT(I,J)).EQ.0)GO TO 10
      K=J
      N=N+1
10    CONTINUE
      IF(N.EQ.0)GO TO 110
      IF(N.GT.1)GO TO 105
      STINPT(INPUT(I,K))=1
105   TVALUE(OUT(I,1))=1
      FG(OUT(I,1))=2*OUT(I,1)-1
      GO TO 130
110   TVALUE(OUT(I,1))=0
      FG(OUT(I,1))=2*OUT(I,1)
      DO 120 J=1,ICNT(I)
      STINPT(INPUT(I,J))=1
120   CONTINUE
130   LEVEL(OUT(I,1))=MAXLEV+1
      RETURN
      END
C=======================================================
C  **   ROUTINE TO SIMULATE NOR GATE
C=======================================================
      SUBROUTINE DNOR
      IMPLICIT INTEGER(A-Z)
      DIMENSION TYPE(100),INPUT(100,10),LEVEL(100),OUT(100,10)
      1,TVALUE(100)
      2,NFT(25),FT(25,100),FG(100),CINPUT(25,25),COUT(25)
      3,ICON(25),INODE(100),ONODE(100),ICNT(100)
```

```
33200          4,NC(100),OCNT(100),STINPT(100),ICONE(100),OCONE(100),CONE(100,1
33300          5,CRIT(100),PROP(100),EXTD(25),STEMCK(25)
33400          COMMON INPUT,OUT,CINPUT,COUT,ICON,INODE,ONODE,
33500          1ICONE,OCONE,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN
33600          2,NODNUM,NOCO,NOGA/AREA2/TVALUE,I/AREA3/LEVEL,STINPT,EXTD,STEMCK
33700          3,CRIT,PROP,NSTMCK,NDE,FTYNOD/AREA4/FT,FG,NFT
33800          N=0
33900          MAXLEV=0
34000          DO 10 J=1,ICNT(I)
34100          IF(MAXLEV.LT.LEVEL(INPUT(I,J)))MAXLEV=LEVEL(INPUT(I,J))
34200          IF(TVALUE(INPUT(I,J)).EQ.0)GO TO 10
34300          K=J
34400          N=N+1
34500 10       CONTINUE
34600          IF(N.EQ.0)GO TO 110
34700          IF(N.GT.1)GO TO 105
34800          STINPT(INPUT(I,K))=1
34900 105      TVALUE(OUT(I,1))=0
35000          FG(OUT(I,1))=2*OUT(I,1)
35100          GO TO 130
35200 110      TVALUE(OUT(I,1))=1
35300          FG(OUT(I,1))=2*OUT(I,1)-1
35400          DO 120 J=1,ICNT(I)
35500          STINPT(INPUT(I,J))=1
35600 120      CONTINUE
35700 130      LEVEL(OUT(I,1))=MAXLEV+1
35800          RETURN
35900          END
36000 C=======================================================================
36100          SUBROUTINE DNOT
36200 C=======================================================================
36300          IMPLICIT INTEGER(A-Z)
36400          DIMENSION TYPE(100),INPUT(100,10),LEVEL(100),OUT(100,10)
36500          1,TVALUE(100)
36600          2,NFT(25),FT(25,100),FG(100),CINPUT(25,25),COUT(25)
36700          3,ICON(25),INODE(100),ONODE(100),ICNT(100)
36800          4,NC(100),OCNT(100),STINPT(100),ICONE(100),OCONE(100),CONE(100,1
```

```
33200          4,NC(100),OCNT(100),STINPT(100),ICONE(100),OCONE(100),CONE(100,
33300          5,CRIT(100),PROP(100),EXTD(25),STEMCK(25)
33400          COMMON INPUT,OUT,CINPUT,COUT,ICON,INODE,ONODE,
33500          1ICONE,OCONE,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN
33600          2,NODNUM,NOCO,NOGA/AREA2/TVALUE,I/AREA3/LEVEL,STINPT,EXTD,STEMCK
33700          3,CRIT,PROP,NSTMCK,NDE,FTYNOD/AREA4/FT,FG,NFT
33800          N=0
33900          MAXLEV=0
34000          DO 10 J=1,ICNT(I)
34100          IF(MAXLEV.LT.LEVEL(INPUT(I,J)))MAXLEV=LEVEL(INPUT(I,J))
34200          IF(TVALUE(INPUT(I,J)).EQ.0)GO TO 10
34300          K=J
34400          N=N+1
34500    10    CONTINUE
34600          IF(N.EQ.0)GO TO 110
34700          IF(N.GT.1)GO TO 105
34800          STINPT(INPUT(I,K))=1
34900    105   TVALUE(OUT(I,1))=0
35000          FG(OUT(I,1))=2*OUT(I,1)
35100          GO TO 130
35200    110   TVALUE(OUT(I,1))=1
35300          FG(OUT(I,1))=2*OUT(I,1)-1
35400          DO 120 J=1,ICNT(I)
35500          STINPT(INPUT(I,J))=1
35600    120   CONTINUE
35700    130   LEVEL(OUT(I,1))=MAXLEV+1
35800          RETURN
35900          END
36000 C=======================================================================
36100          SUBROUTINE DNOT
36200 C=======================================================================
36300          IMPLICIT INTEGER(A-Z)
36400          DIMENSION TYPE(100),INPUT(100,10),LEVEL(100),OUT(100,10)
36500          1,TVALUE(100)
36600          2,NFT(25),FT(25,100),FG(100),CINPUT(25,25),COUT(25)
36700          3,ICON(25),INODE(100),ONODE(100),ICNT(100)
36800          4,NC(100),OCNT(100),STINPT(100),ICONE(100),OCONE(100),CONE(100,1
```

```
369c0           5,CRIT(100),PROP(100),EXTD(25),STEMCK(25)
37000           COMMON INPUT,OUT,CINPUT,COUT,ICON,INODE,ONODE,
371c0           1ICONE,OCONE,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN
372c0           2,NODNUM,NOCO,NOGA/AREA2/TVALUE,I/AREA3/LEVEL,STINPT,EXTD,STEMCK
373c0           3,CRIT,PROP,NSTMCK,NDE,FTYNOD/AREA4/FT,FG,NFT
374c0           LEVEL(OUT(I,1))=LEVEL(INPUT(I,1))+1
375c0           IF(TVALUE(INPUT(I,1)).EQ.0)GO TO 20
376c0           TVALUE(OUT(I,1))=0
377c0           FG(OUT(I,1))=2*OUT(I,1)
378c0           GO TO 30
379c0  20       TVALUE(OUT(I,1))=1
38000           FG(OUT(I,1))=2*OUT(I,1)-1
381c0  30       STINPT(INPUT(I,1))=1
382c0           RETURN
383c0           END
384c0  C=======================================================
385c0           SUBROUTINE DXOR
386c0  C=======================================================
387c0           IMPLICIT INTEGER(A-Z)
388c0           DIMENSION TYPE(100),INPUT(100,10),LEVEL(100),OUT(100,10)
389c0           1,TVALUE(100)
390c0           2,NFT(25);FT(25,100),FG(100),CINPUT(25,25),COUT(25)
391c0           3,ICON(25),INODE(100),ONODE(100),ICNT(100)
392c0           4,NC(100),OCNT(100),STINPT(100),ICONE(100),OCONE(100),CONE(100,1
393c0           5,CRIT(100),PROP(100),EXTD(25),STEMCK(25)
394c0           COMMON INPUT,OUT,CINPUT,COUT,ICON,INODE,ONODE,
395c0           1ICONE,OCONE,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN
396c0           2,NODNUM,NOCO,NOGA/AREA2/TVALUE,I/AREA3/LEVEL,STINPT,EXTD,STEMCK
397c0           3,CRIT,PROP,NSTMCK,NDE,FTYNOD/AREA4/FT,FG,NFT
398c0           N=0
399c0           MAXLEV=0
400c0           DO 20 K=1,ICNT(I)
401c0           IF(MAXLEV.LT.LEVEL(INPUT(I,K)))MAXLEV=LEVEL(INPUT(I,K))
402c0           STINPT(INPUT(I,K))=1
403c0           IF(TVALUE(NDE).EQ.0)N=N+1
404c0  20       CONTINUE
405c0           IF((N.EQ.0).OR.(N.EQ.2))TVALUE(OUT(I,1))=0
```

```
40600          IF(N.EQ.1)TVALUE(OUT(I,1))=1
40700          IF(TVALUE(OUT(I,1)).EQ.1)GO TO 50
40800          FG(OUT(I,1))=2*OUT(I,1)
40900          GO TO 60
41000 50       FG(OUT(I,1))=2*OUT(I,1)-1
41100 60       LEVEL(OUT(I,1))=MAXLEV+1
41200          RETURN
41300          END
41400 C=======================================================================
41500          SUBROUTINE DXNOR
41600 C=======================================================================
41700          IMPLICIT INTEGER(A-Z)
41800          DIMENSION TYPE(100),INPUT(100,10),LEVEL(100),OUT(100,10)
41900          1,TVALUE(100),X(1024,10),AFT(25,100),NAFT(25),PFT(25,100)
42000          2,NFT(25),FT(25,100),FG(100),CINPUT(25,25),COUT(25)
42100          3,ICON(25),INODE(100),ONODE(100),ICNT(100)
42200          4,NC(100),OCNT(100),STINPT(100),ICONE(100),OCONE(100),CONE(100,10
42300          5,CRIT(100),PROP(100),EXTD(25),STEMCK(25)
42400          COMMON INPUT,OUT,CINPUT,COUT,ICON,INODE,ONODE,
42500          1ICONE,OCONE,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN
42600          2,NODNUM,NOCO,NOGA/AREA2/TVALUE,I/AREA3/LEVEL,STINPT,EXTD,STEMCK
42700          3,CRIT,PROP,NSTMCK,NDE,FTYNUD/AREA4/FT,FG,NFT
42800          N=0
42900          MAXLEV=0
43000          DO 20 K=1,ICNT(I)
43100          IF(MAXLEV.LT.LEVEL(INPUT(I,K)))MAXLEV=LEVEL(INPUT(I,K))
43200          IF(TVALUE(NDE).EQ.0)N=N+1
43300          STINPT(INPUT(I,K))=1
43400 20       CONTINUE
43500          IF((N.EQ.0).OR.(N.EQ.2))TVALUE(OUT(I,1))=1
43600          IF(N.EQ.1)TVALUE(OUT(I,1))=0
43700          IF(TVALUE(OUT(I,1)).EQ.1)GO TO 50
43800          FG(OUT(I,1))=2*OUT(I,1)
43900          GO TO 60
44000 50       FG(OUT(I,1))=2*OUT(I,1)-1
44100 60       LEVEL(OUT(I,1))=MAXLEV+1
44200          RETURN
```

CRITICAL PATH

```
44300          END
44400 C=======================================================
44500 C  **    ROUTINE TO SIMULATE CONTINUITY BLOCK
44600 C=======================================================
44700          SUBROUTINE CONT
44800          IMPLICIT INTEGER(A-Z)
44900          DIMENSION TYPE(100),INPUT(100,10),LEVEL(100),OUT(100,10)
45000          1,TVALUE(100),X(1024,10),AFT(25,100),NAFT(25),PFT(25,100)
45100          2,NFT(25),FT(25,100),FG(100),CINPUT(25,25),COUT(25)
45200          3,ICON(25),INODE(100),ONODE(100),ICNT(100)
45300          4,NC(100),OCNT(100),STINPT(100),ICONE(100),OCONE(100),CONE(100,10
45400          5,CRIT(100),PROP(100),EXTD(25),STEMCK(25)
45500          COMMON INPUT,OUT,CINPUT,COUT,ICON,INODE,ONODE,
45600          1ICONE,OCONE,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN
45700          2,NODNUM,NOCO,NOGA/AREA2/TVALUE,I/AREA3/LEVEL,STINPT,EXTD,STEMCK
45800          3,CRIT,PROP,NSTMCK,NDE,FTYNOD/AREA4/FT,FG,NFT
45900          STINPT(INPUT(I,1))=1
46000          IF(TVALUE(INPUT(I,1)).EQ.0)GO TO 50
46100          DO 10 J=1,OCNT(I)
46200          TVALUE(OUT(I,J))=TVALUE(INPUT(I,1))
46300          FG(OUT(I,J))=2*OUT(I,J)-1
46400          LEVEL(OUT(I,J))=LEVEL(INPUT(I,1))+1
46500 10       CONTINUE
46600          GO TO 60
46700 50       DO 20 J=1,OCNT(I)
46800          TVALUE(OUT(I,J))=TVALUE(INPUT(I,1))
46900          FG(OUT(I,J))=2*OUT(I,J)
47000          LEVEL(OUT(I,J))=LEVEL(INPUT(I,1))+1
47100 20       CONTINUE
47200 60       RETURN
47300          END
47400 C=======================================================
47500          SUBROUTINE TRACE
47600 C=======================================================
47700          IMPLICIT INTEGER(A-Z)
47800          DIMENSION EXTD(25),CRIT(100),LEVEL(100),STEMCK(25),PROP(100)
47900          1,FG(100),NFT(25),FT(25,100),STINPT(100)
```

CRITICAL PATH

```
      COMMON /AREA1/PRINCN,INTCNT,PROTCN
     2,NODNUM,NOCO,NOGA/AREA3/LEVEL,STINPT,EXTD,STEMCK
     3,CRIT,PROP,NSTMCK,NDE,FTYNOD/AREA4/FT,FG,NFT
      POT=PRINCN+INTCNT
      DO 100 K=1,PROTCN
      DO 8 V=1,NODNUM
      CRIT(V)=0
    8 CONTINUE
      DO 3 T=1,NOCO
      EXTD(T)=0
    3 CONTINUE
      OUTPUT=K
      NDE=OUTPUT+POT
      NSTMCK=0
      CRIT(NDE)=1
      CALL EXTEND
    5 IF(NSTMCK.EQ.0)GO TO 104
      HIGH=1
      IF(NSTMCK.EQ.1)GO TO 12
      DO 10 P=2,NSTMCK
      IF(LEVEL(STEMCK(P)).GT.LEVEL(STEMCK(HIGH)))HIGH=P
   10 CONTINUE
   12 NDE=STEMCK(HIGH)
      DO 20 P=HIGH,NSTMCK
      STEMCK(P)=STEMCK(P+1)
   20 CONTINUE
      NSTMCK=NSTMCK-1
      CALL CRITIC
      IF(CRIT(NDE).EQ.1)CALL EXTEND
      DO 90 F=1,NODNUM
      PROP(F)=0
   90 CONTINUE
      GO TO 5
  104 DO 110 AA=1,NODNUM
      IF(CRIT(AA).EQ.0)GO TO 110
      NFT(OUTPUT)=NFT(OUTPUT)+1
      FT(OUTPUT,NFT(OUTPUT))=FG(AA)
```

```
      110   CONTINUE
      100   CONTINUE
      120   RETURN
            END
C=====================================================================
            SUBROUTINE EXTEND
C=====================================================================
            IMPLICIT INTEGER(A-Z)
            DIMENSION EXTD(25),CRIT(100),STEMCK(25),OUT(100,10),COUT(25)
           1,K(50),J(100),TYPE(100),ICON(25),ONODE(100),INODE(100)
           2,STINPT(100),CINPUT(25,25),OCNT(100)
           3,ICNT(100),INPUT(100,10),LEVEL(100)
           4,NC(100),PROP(100),ICONE(100),OCONE(100),CONE(100,10)
            COMMON INPUT,OUT,CINPUT,COUT,ICON,INODE,ONODE,
           1ICONE,OCONE,ICNT,OCNT,TYPE/AREA3/LEVEL,STINPT,EXTD,STEMCK
           3,CRIT,PROP,NSTMCK,NDE,FTYNOD
            I=0
            CONO=OCONE(NDE)
            IF(CONO.EQ.0)GO TO 50
            EXTD(CONO)=1
   10       I=I+1
            K(I)=ONODE(NDE)
            IF(K(I).EQ.0)GO TO 40
            DO 20 PP=1,ICON(CONO)
            IF(NDE.NE.CINPUT(CONO,PP))GO TO 20
            IF(TYPE(K(I)).NE.8)GO TO 15
            DO 13 F=1,NSTMCK
            IF(STEMCK(F).EQ.INPUT(K(I),1))GO TO 15
   13       CONTINUE
            NSTMCK=NSTMCK+1
            STEMCK(NSTMCK)=INPUT(K(I),1)
   15       GO TO 40
   20       CONTINUE
            J(K(I))=0
   25       J(K(I))=J(K(I))+1
            NDE=INPUT(K(I),J(K(I)))
            IF(STINPT(NDE).EQ.0)GO TO 35
```

CRITICAL PATH

```
      CRIT(NDE)=1
      GO TO 10
   30 I=I-1
   35 IF(J(K(I)).LT.ICNT(K(I)))GO TO 25
   40 IF(I.GT.1)GO TO 30
   50 RETURN
      END
C=======================================================================
      SUBROUTINE PROPGT
C=======================================================================
      IMPLICIT INTEGER(A-Z)
      DIMENSION PROP(100),CINPUT(25,25),COUT(25),ICON(25),ONODE(100)
     1,TYPE(100),OCNT(100),LEVEL(100),EXTD(25),CONE(100,10),
     2INODE(100),STINPT(100),TVALUE(100),STEMCK(25),CRIT(100)
     3,ICNT(100),INPUT(100,10),OUT(100,10),ICONE(100),OCONE(100)
      COMMON INPUT,OUT,CINPUT,COUT,ICON,INODE,ONODE,
     1ICONE,OCONE,ICNT,OCNT,TYPE/AREA2/TVALUE,1/AREA3/LEVEL
     2,STINPT,EXTD,STEMCK,CRIT,PROP,NSTMCK,NDE,FTYNOD/AREA8/CONE,NC
     D=INODE(FTYNOD)
      IF(STINPT(FTYNOD).NE.1)GO TO 100
      DO 102 G=1,ICNT(D)
      IF(STINPT(INPUT(D,G)).EQ.1)GO TO 102
      IF(PROP(INPUT(D,G)).NE.1)GO TO 102
      GO TO 210
  102 CONTINUE
      GO TO 200
  100 IF((TYPE(D).EQ.1).OR.(TYPE(D).EQ.3))GO TO 400
      IF((TYPE(D).EQ.2).OR.(TYPE(D).EQ.4))GO TO 600
  400 IF(TVALUE(FTYNOD).NE.0)GO TO 210
      DO 10 K=1,ICNT(D)
      R=INPUT(D,K)
      IF((TVALUE(R).EQ.0).AND.(PROP(R).EQ.1))GO TO 10
      IF((.NOT.(TVALUE(R).EQ.0)).AND.(.NOT.(PROP(R).EQ.1)))GO TO 10
      GO TO 210
   10 CONTINUE
      GO TO 200
  600 IF(TVALUE(FTYNOD).NE.1)GO TO 210
```

```
      CRITICAL PATH
591 0         DO 20 K=1,ICNT(D)
592 0         R=INPUT(D,K)
593 0         IF((TVALUE(R).EQ.1).AND.(PROP(R).EQ.1))GO TO 10
594 0         IF((.NOT.(TVALUE(R).EQ.1)).AND.(.NOT.(PROP(R).EQ.1)))GO TO 10
595 0         GO TO 210
596 0  20     CONTINUE
597 0  200    DO 201 K=1,OCNT(D)
598 0         PROP(OUT(D,K))=1
599 0  201    CONTINUE
600 0         GO TO 240
601 0  210    PROP(OUT(D,1))=0
602 0  240    RETURN
603 0         END
604 0 C=======================================================================
605 0         SUBROUTINE CRITIC
606 0 C=======================================================================
607 0         IMPLICIT INTEGER(A-Z)
608 0         DIMENSION CRIT(100),LEVEL(100),PROP(100),EXTD(25),INPUT(100,10)
609 0        2,INODE(100),CINPUT(25,25),COUT(25),ICON(25),ONODE(100)
610 0        3,OCNT(100),OUT(100,10),FRONT(20),ICNT(100),TYPE(100),
611 0        4NC(100),STINPT(100),STEMCK(25),ICONE(100),OCONE(100),CONE(100,10)
612 0         COMMON INPUT,OUT,CINPUT,COUT,ICON,INODE,ONODE,
613 0        1ICONE,OCONE,ICNT,OCNT,TYPE/AREA3/LEVEL,STINPT,EXTD,STEMCK
614 0        3,CRIT,PROP,NSTMCK,NDE,FTYNOD/AREA8/CONE,NC
615 0         NFRONT=0
616 0         PROP(NDE)=1
617 0         CGTE=INODE(NDE)
618 0         DO 10 K=1,OCNT(CGTE)
619 0         DO 8 Y=1,NC(OUT(CGTE,K))
620 0         V=CONE(OUT(CGTE,K),Y)
621 0         IF(EXTD(V).NE.1)GO TO 8
622 0         GO TO 9
623 0  8      CONTINUE
624 0         GO TO 10
625 0  9      NFRONT=NFRONT+1
626 0         FRONT(NFRONT)=OUT(CGTE,K)
627 0         PROP(FRONT(NFRONT))=1
```

```
      CONTINUE
   15 LOW=1
      IF(NFRONT.EQ.1)GO TO 22
      DO 20 Q=2,NFRONT
      W=INODE(FRONT(Q))
      X=INODE(FRONT(LOW))
      IF(LEVEL(OUT(W,1)).LT.LEVEL(OUT(X,1)))LOW=Q
   20 CONTINUE
   22 FTYNOD=FRONT(LOW)
      FTY=INODE(FTYNOD)
      DO 30 P=LOW,NFRONT
      FRONT(P)=FRONT(P+1)
   30 CONTINUE
      NFRONT=NFRONT-1
      IF(NFRONT.EQ.0)GO TO 100
      CALL PROPGT
      IF(PROP(OUT(FTY,1)).NE.1)GO TO 15
      DO 40 F=1,OCNT(FTY)
      G=OUT(FTY,F)
      DO 38 E=1,NC(G)
      V=CONE(G,E)
      IF(EXTD(V).NE.1)GO TO 38
      GO TO 34
   38 CONTINUE
      GO TO 40
   34 DO 37 T=1,NFRONT
      IF(FRONT(T).EQ.OUT(FTY,F))GO TO 40
   37 CONTINUE
      NFRONT=NFRONT+1
      FRONT(NFRONT)=OUT(FTY,F)
   40 CONTINUE
      GO TO 15
  100 CALL PROPGT
      IF(PROP(OUT(FTY,1)).NE.1)GO TO 120
      IF(CRIT(FTYNOD).EQ.1)CRIT(NDE)=1
  120 RETURN
      END
```

```
00100 C=======================================================================
00200 C       FAULT-FREE SIMULATION
00300 C=======================================================================
00400         SUBROUTINE FFSIMU
00500 C       XIN=NO. OF INPUT SETS TO BE SIMULATED
00600 C       X(I,J)=ITH INPUT SET'S JTH PRIMARY INPUT
00700         IMPLICIT INTEGER(A-Z)
00800         REAL W,XV,CONE,CZERO,BONE,BZERO,DONE,DZERO,S,TIME,TIME1,TIME2
00900         1,ONECNT,ZERCNT,SENCNT,INTER,IFC,FF,FC,B1CONT,B0CONT,WW,TP,XX
01000 C       DOUBLE PRECISION TP,XX,INTER,IFC,FC,FF
01100         DIMENSION ICNT(100),OCNT(100),TYPE(100),INPUT(100,10)
01200         1,TVALUE(100),X(1024,10),ONECNT(100),ZERCNT(100),SENCNT(100)
01300         2,OUT(100,10),CONE(100),CZERO(100),BONE(100),BZERO(100),S(100)
01400         3,DONE(100),DZERO(100),XX(200,25),FF(200,25),INTER(25),IFC(25)
01500         4,FC(25),WW(100),TP(100)
01600         COMMON INPUT,OUT,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN,NOGA
01700         1,NODNUM/AREA2/TVALUE,I/AREA9/ONECNT,ZERCNT,SENCNT/AREA10/X,TOTA
01800         W(XV,XIN)=1-(XIN-1)*5.0*XV/(1-XV)
01900         READ(21,*),XIN
02000         READ(21,*)((X(I,J),J=1,PRINCN),I=1,XIN)
02100         NV=1
02200         POT=PRINCN+INTCNT
02300         CALL TIMER(TIME1)
02400         DO 567 B1=1,PROTCN
02500         B2=B1+POT
02600         BONE(B2)=1
02700         BZERO(B2)=1
02800 567     CONTINUE
02900         INITIL=1
03000         YY=0
03100 250     YY=YY+1
03200         DO 1000 Y=INITIL,XIN+INITIL-1
03300         DO 280 J=1,PRINCN
03400         TVALUE(J)=X(Y,J)
03500         IF(TVALUE(J).EQ.0)GO TO 279
03600         ONECNT(J)=ONECNT(J)+1
03700         GO TO 280
```

STAFAN

```
03800 279      ZERCNT(J)=ZERCNT(J)+1
03900 280      CONTINUE
04000          I=1
04100 300      GO TO (310,312,314,316,318,320,322,324),TYPE(I)
04200 310      CALL AND
04300          GO TO 340
04400 312      CALL OR
04500          GO TO 340
04600 314      CALL NAND
04700          GO TO 340
04800 316      CALL NOR
04900          GO TO 340
05000 318      CALL XOR
05100          GO TO 340
05200 320      CALL XNOR
05300          GO TO 340
05400 322      CALL NOT
05500          GO TO 340
05600 324      CALL CONT
05700 340      I=I+1
05800          IF(I.LE.NOGA)GO TO 300
05900 1000     CONTINUE
06000          DO 342 B=1,NODNUM
06100          CONE(B)=ONECNT(B)/XIN
06200          CZERO(B)=ZERCNT(B)/XIN
06300          S(B)=SENCNT(B)/XIN
06400 342      CONTINUE
06500          DO 378 Z=1,NOGA
06600          P=NOGA-Z+1
06700          M=ICNT(P)
06800          GO TO (367,369,371,373,375,375,375,376),TYPE(P)
06900 367      DO 368 L=1,M
07000          BONE(INPUT(P,L))=BONE(OUT(P,1))*CONE(OUT(P,1))/CONE(INPUT(P,L))
07100          BZERO(INPUT(P,L))=BZERO(OUT(P,1))*(S(INPUT(P,L))-CONE(OUT(P,1)))
07200          1/CZERO(INPUT(P,L))
07300 368      CONTINUE
07400          GO TO 378
```

STAFAN

```
07500 369      DO 370 L=1,M
07600          BZERO(INPUT(P,L))=BZERO(OUT(P,1))*CZERO(OUT(P,1))/CZERO(
07700          1INPUT(P,L))
07800          BONE(INPUT(P,L))=BONE(OUT(P,1))*(S(INPUT(P,L))-CZERO(OUT(P,1)))
07900          1/CONE(INPUT(P,L))
08000 370      CONTINUE
08100          GO TO 378
08200 371      DO 372 L=1,M
08300          BONE(INPUT(P,L))=BZERO(OUT(P,1))*CZERO(OUT(P,1))/CONE(INPUT(P,L))
08400          BZERO(INPUT(P,L))=BONE(OUT(P,1))*(S(INPUT(P,L))-CZERO(OUT(P,1)))
08500          1/CZERO(INPUT(P,L))
08600 372      CONTINUE
08700          GO TO 378
08800 373      DO 374 L=1,M
08900          BZERO(INPUT(P,L))=BONE(OUT(P,1))*CONE(OUT(P,1))/CZERO(INPUT(P,L))
09000          BONE(INPUT(P,L))=BZERO(OUT(P,1))*(S(INPUT(P,L))-CONE(OUT(P,1)))
09100          1/CONE(INPUT(P,L))
09200 374      CONTINUE
09300          GO TO 378
09400 375      DO 475 L=1,M
09500          BONE(INPUT(P,M))=BONE(OUT(P,1))
09600          BZERO(INPUT(P,M))=BONE(OUT(P,1))
09700 475      CONTINUE
09800 376      B1CONT=0
09900          BOCONT=0
10000          DO 377 Q=1,OCNT(P)
10100          B1CONT=B1CONT+BONE(OUT(P,Q))
10200          BOCONT=BOCONT+BZERO(OUT(P,Q))
10300 377      CONTINUE
10400          BONE(INPUT(P,1))=B1CONT/OCNT(P)
10500          BZERO(INPUT(P,1))=BOCONT/OCNT(P)
10600 378      CONTINUE
10700          WRITE(22,897)
10800 897      FORMAT(//5X,'NODE',3X,'  C0  ',3X,'  C1  ',3X,'  B1  ',3X,'  B0
10900          1,3X,'  D1  ',3X,'  D0  ')
11000          DO 385 F=1,NODNUM
11100          DONE(F)=BZERO(F)*CZERO(F)
```

```
11200           DZERO(F)=BONE(F)*CONE(F)
11300 C         WW(2*F)=W(DONE(F),XIN)
11400 C         WW(2*F-1)=W(DZERO(F),XIN)
11500           TP(2*F)=(1.0-DONE(F))XIN
11600           TP(2*F-1)=(1.0-DZERO(F))XIN
11700           XX(2*F,YY)=1.0-TP(2*F)/WW(2*F)
11800           XX(2*F-1,YY)=1.0-IP(2*F-1)/WW(2*F-1)
11900 385       CONTINUE
12000           DO 997 KK=1,NODNUM
12100           WRITE(22,998),KK,CZERO(KK),CONE(KK),BONE(KK),BZERO(KK),DONE(KK)
12200           1,DZERO(KK)
12300 998       FORMAT(5X,I4,3X,F6.4,3X,F6.4,3X,F6.4,3X,F6.4,3X,F6.4,3X,F6.4,3X,F6.4)
12400           INITIL=INITIL+XIN
12500           IF(YY.LT.NV)GO TO 250
12600           DO 91 KF=1,2*NODNUM
12700           INTER(1)=(1.0-XX(KF,1))XIN
12800           FF(KF,1)=1.0-INTER(1)/WW(XX(KF,1),XIN)
12900           IF(NV.EQ.1)GO TO 92
13000           DO 92 JF=2,NV
13100           INTER(JF)=INTER(JF-1)*(1.0-XX(KF,JF))XIN
13200           FF(KF,JF)=1.0-INTER(JF)/WW(XX(KF,JF),XIN)
13300 92        CONTINUE
13400 91        CONTINUE
13500           DO 98 JL=1,NV
13600           DO 96 KL=1,2*NODNUM
13700           IFC(JL)=IFC(JL)+FF(KL,JL)
13800 96        CONTINUE
13900           FC(JL)=IFC(JL)/(2*NODNUM)
14000           WRITE(22,97),FC(JL)
14100 97        FORMAT(///5X,'FRACTION OF FAULTS DETECTED = ',F7.4)
14200 98        CONTINUE
14300           CALL TIMER(TIME2)
14400           TIME=TIME2-TIME1
14500           WRITE(22,587),TIME
14600 587       FORMAT(///5X,'TIME TAKEN TO ESTIMATE FAULT COVERAGE =',I5)
14700           RETURN
14800           END
```

```
14900 C=======================================================================
15000 C       ROUTINE TO SIMULATE AND GATE
15100 C=======================================================================
15200         SUBROUTINE AND
15300         IMPLICIT INTEGER(A-Z)
15400         REAL ONECNT,ZERCNT,SENCNT
15500         DIMENSION ICNT(100),OCNT(100),TYPE(100),INPUT(100,10)
15600         1,TVALUE(100),X(1024,10),ONECNT(100),ZERCNT(100),SENCNT(100)
15700         3,OUT(100,10)
15800         COMMON INPUT,OUT,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN,NOGA
15900         1,NODNUM/AREA2/TVALUE,I/AREA9/ONECNT,ZERCNT,SENCNT
16000         N=0
16100         DO 10 J=1,ICNT(I)
16200         IF(TVALUE(INPUT(I,J)).EQ.1)GO TO 10
16300         K=J
16400         N=N+1
16500 10      CONTINUE
16600         IF(N.EQ.0)GO TO 110
16700         IF(N.GT.1)GO TO 105
16800         SENCNT(INPUT(I,K))=SENCNT(INPUT(I,K))+1
16900 105     TVALUE(OUT(I,1))=0
17000         ZERCNT(OUT(I,1))=ZERCNT(OUT(I,1))+1
17100         GO TO 130
17200 110     TVALUE(OUT(I,1))=1
17300         ONECNT(OUT(I,1))=ONECNT(OUT(I,1))+1
17400         DO 120 J=1,ICNT(I)
17500         SENCNT(INPUT(I,J))=SENCNT(INPUT(I,J))+1
17600 120     CONTINUE
17700 130     RETURN
17800         END
17900 C=======================================================================
18000 C       ROUTINE TO SIMULATE NAND GATE
18100 C=======================================================================
18200         SUBROUTINE NAND
18300         IMPLICIT INTEGER(A-Z)
18400         REAL ONECNT,ZERCNT,SENCNT
18500         DIMENSION ICNT(100),OCNT(100),TYPE(100),INPUT(100,10)
```

```
18600          1,TVALUE(100),X(1024,10),ONECNT(100),ZERCNT(100),SENCNT(100)
18700          3,OUT(100,10)
18800          COMMON INPUT,OUT,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN,NOGA
18900          1,NODNUM/AREA2/TVALUE,I/AREA9/ONECNT,ZERCNT,SENCNT
19000          N=0
19100          DO 10 J=1,ICNT(I)
19200          IF(TVALUE(INPUT(I,J)).EQ.1)GO TO 10
19300          K=J
19400          N=N+1
19500 10       CONTINUE
19600          IF(N.EQ.0)GO TO 110
19700          IF(N.GT.1)GO TO 105
19800          SENCNT(INPUT(I,K))=SENCNT(INPUT(I,K))+1
19900 105      TVALUE(OUT(I,1))=1
20000          ONECNT(OUT(I,1))=ONECNT(OUT(I,1))+1
20100          GO TO 130
20200 110      TVALUE(OUT(I,1))=0
20300          ZERCNT(OUT(I,1))=ZERCNT(OUT(I,1))+1
20400          DO 120 J=1,ICNT(I)
20500          SENCNT(INPUT(I,J))=SENCNT(INPUT(I,J))+1
20600 120      CONTINUE
20700 130      RETURN
20800          END
20900 C=========================================================================
21000 C        ROUTINE TO SIMULATE OR GATE
21100 C=========================================================================
21200          SUBROUTINE OR
21300          IMPLICIT INTEGER(A-Z)
21400          REAL ONECNT,ZERCNT,SENCNT
21500          DIMENSION ICNT(100),OCNT(100),TYPE(100),INPUT(100,10)
21600          1,TVALUE(100),X(1024,10),ONECNT(100),ZERCNT(100),SENCNT(100)
21700          3,OUT(100,10)
21800          COMMON INPUT,OUT,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN,NOGA
21900          1,NODNUM/AREA2/TVALUE,I/AREA9/ONECNT,ZERCNT,SENCNT
22000          N=0
22100          DO 10 J=1,ICNT(I)
22200          IF(TVALUE(INPUT(I,J)).EQ.0)GO TO 10
```

```
22300           K=J
22400           N=N+1
22500 10        CONTINUE
22600           IF(N.EQ.0)GO TO 110
22700           IF(N.GT.1)GO TO 105
22800           SENCNT(INPUT(I,K))=SENCNT(INPUT(I,K))+1
22900 105       TVALUE(OUT(I,1))=1
23000           ONECNT(OUT(I,1))=ONECNT(OUT(I,1))+1
23100           GO TO 130
23200 110       TVALUE(OUT(I,1))=0
23300           ZERCNT(OUT(I,1))=ZERCNT(OUT(I,1))+1
23400           DO 120 J=1,ICNT(I)
23500           SENCNT(INPUT(I,J))=SENCNT(INPUT(I,J))+1
23600 120       CONTINUE
23700 130       RETURN
23800           END
23900 C=======================================================================
24000 C         ROUTINE TO SIMULATE NOR GATE
24100 C=======================================================================
24200           SUBROUTINE NOR
24300           IMPLICIT INTEGER(A-Z)
24400           REAL ONECNT,ZERCNT,SENCNT
24500           DIMENSION ICNT(100),OCNT(100),TYPE(100),INPUT(100,10)
24600           1,TVALUE(100),X(1024,10),ONECNT(100),ZERCNT(100),SENCNT(100)
24700           3,OUT(100,10)
24800           COMMON INPUT,OUT,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN,NOGA
24900           1,NODNUM/AREA2/TVALUE,I/AREA9/ONECNT,ZERCNT,SENCNT
25000           N=0
25100           DO 10 J=1,ICNT(I)
25200           IF(TVALUE(INPUT(I,J)).EQ.0)GO TO 10
25300           K=J
25400           N=N+1
25500 10        CONTINUE
25600           IF(N.EQ.0)GO TO 110
25700           IF(N.GT.1)GO TO 105
25800           SENCNT(INPUT(I,K))=SENCNT(INPUT(I,K))+1
25900 105       TVALUE(OUT(1,1))=0
```

```
26000           ZERCNT(OUT(I,1))=ZERCNT(OUT(I,1))+1
26100           GO TO 130
26200 110       TVALUE(OUT(I,1))=1
26300           ONECNT(OUT(I,1))=ONECNT(OUT(I,1))+1
26400           DO 120 J=1,ICNT(I)
26500           SENCNT(INPUT(I,J))=SENCNT(INPUT(I,J))+1
26600 120       CONTINUE
26700 130       RETURN
26800           END
26900 C=========================================================================
27000 C         ROUTINE FOR XOR GATE
27100 C=========================================================================
27200           SUBROUTINE XOR
27300           IMPLICIT INTEGER(A-Z)
27400           REAL ONECNT,ZERCNT,SENCNT
27500           DIMENSION ICNT(100),OCNT(100),TYPE(100),INPUT(100,10)
27600          1,TVALUE(100),X(1024,10),ONECNT(100),ZERCNT(100),SENCNT(100)
27700          3,OUT(100,10)
27800           COMMON INPUT,OUT,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN,NOGA
27900          1,NODNUM/AREA2/TVALUE,I/AREA9/ONECNT,ZERCNT,SENCNT
28000           N=0
28100           DO 10 J=1,ICNT(I)
28200           SENCNT(INPUT(I,J))=SENCNT(INPUT(I,J))+1
28300           IF(TVALUE(INPUT(I,J)).EQ.0)N=N+1
28400 10        CONTINUE
28500           IF((N.EQ.0).OR.(N.EQ.2))GO TO 50
28600           TVALUE(OUT(I,1))=1
28700           ONECNT(OUT(I,1))=ONECNT(OUT(I,1))+1
28800           GO TO 60
28900 50        TVALUE(OUT(I,1))=0
29000           ZERCNT(OUT(I,1))=ZERCNT(OUT(I,1))+1
29100 60        RETURN
29200           END
29300 C=========================================================================
29400 C         ROUTINE FOR XNOR GATE
29500 C=========================================================================
29600           SUBROUTINE XNOR
```

STAFAN

```
29700          IMPLICIT INTEGER(A-Z)
29800          REAL ONECNT,ZERCNT,SENCNT
29900          DIMENSION ICNT(100),OCNT(100),TYPE(100),INPUT(100,10)
30000          1,TVALUE(100),X(1024,10),ONECNT(100),ZERCNT(100),SENCNT(100)
30100          3,OUT(100,10)
30200          COMMON INPUT,OUT,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN,NOGA
30300          1,NODNUM/AREA2/TVALUE,I/AREA9/ONECNT,ZERCNT,SENCNT
30400          N=0
30500          DO 10 J=1,ICNT(I)
30600          SENCNT(INPUT(I,J))=SENCNT(INPUT(I,J))+1
30700          IF(TVALUE(INPUT(I,J)).EQ.0)N=N+1
30800 10       CONTINUE
30900          IF((N.EQ.0).OR.(N.EQ.2))GO TO 50
31000          TVALUE(OUT(I,1))=0
31100          ZERCNT(OUT(I,1))=ZERCNT(OUT(I,1))+1
31200          GO TO 60
31300 50       TVALUE(OUT(I,1))=1
31400          ONECNT(OUT(I,1))=ONECNT(OUT(I,1))+1
31500 60       RETURN
31600          END
31700 C=========================================================================
31800 C        ROUTINE TO SIMULATE NOT GATE
31900 C=========================================================================
32000          SUBROUTINE NOT
32100          IMPLICIT INTEGER(A-Z)
32200          REAL ONECNT,ZERCNT,SENCNT
32300          DIMENSION ICNT(100),OCNT(100),TYPE(100),INPUT(100,10)
32400          1,TVALUE(100),X(1024,10),ONECNT(100),ZERCNT(100),SENCNT(100)
32500          3,OUT(100,10)
32600          COMMON INPUT,OUT,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN,NOGA
32700          1,NODNUM/AREA2/TVALUE,I/AREA9/ONECNT,ZERCNT,SENCNT
32800          SENCNT(INPUT(I,1))=SENCNT(INPUT(I,1))+1
32900          IF(TVALUE(INPUT(I,1)).EQ.0)GO TO 50
33000          TVALUE(OUT(I,1))=0
33100          ZERCNT(OUT(I,1))=ZERCNT(OUT(I,1))+1
33200          GO TO 60
33300 50       TVALUE(OUT(I,1))=1
```

```
33400          ONECNT(OUT(I,1))=ONECNT(OUT(I,1))+1
33500 60       RETURN
33600          END
33700 C=======================================================================
33800 C        ROUTINE FOR CONTINUITY BLOCK
33900 C=======================================================================
34000          SUBROUTINE CONT
34100          IMPLICIT INTEGER(A-Z)
34200          REAL ONECNT,ZERCNT,SENCNT
34300          DIMENSION ICNT(100),OCNT(100),TYPE(100),INPUT(100,10)
34400          1,TVALUE(100),X(1024,10),ONECNT(100),ZERCNT(100),SENCNT(100)
34500          3,OUT(100,10)
34600          COMMON INPUT,OUT,ICNT,OCNT,TYPE/AREA1/PRINCN,INTCNT,PROTCN,NOGA
34700          1,NODNUM/AREA2/TVALUE,I/AREA9/ONECNT,ZERCNT,SENCNT
34800          SENCNT(INPUT(I,1))=SENCNT(INPUT(I,1))+1
34900          IF(TVALUE(INPUT(I,1)).EQ.0)GO TO 15
35000          DO 10 K=1,OCNT(I)
35100          TVALUE(OUT(I,K))=TVALUE(INPUT(I,1))
35200          ONECNT(OUT(I,K))=ONECNT(OUT(I,K))+1
35300 10       CONTINUE
35400          GO TO 60
35500 15       DO 20 K=1,OCNT(I)
35600          TVALUE(OUT(I,K))=TVALUE(INPUT(I,1))
35700          ZERCNT(OUT(I,K))=ZERCNT(OUT(I,K))+1
35800 20       CONTINUE
35900 60       RETURN
36000          END
```