The Design Of Digital Test Simulator

MOHAMED BIN OTHMAN
BAMBANG SUNARYO BIN SUPARJO
Department of Computer Science,
Universiti Pertanian Malaysia,
UPM Serdang 43400
Selangor Darul Ehsan

Abstract

Digital Test Simulator is a Computer Aided Design (CAD) tools written in Turbo Pascal language ver. 5.0. It is a gate level simulator to measures the testability values of a combinational logic circuit. It was developed based on the testability analysis system called Computer-Aided MEsure for LOgic Testability (CAMELOT).

The measurement of the testability, controllability and observability for every nodes are based on the topologyical description of the circuit. The final results produced by simulator can be expressed in the form of table and histogram. Comparison of the various nodal testability values allows the areas of poor testability to be readily identified and the inprovement can be done to the circuits to make it more testable.

Key Words: Logic circuit; Computer-Aided Design; CAMELOT approach; Digital Test Simulator

INTRODUCTION

Testing digital electronic system at chip board or system level is expensive. The increasing complexity of the design primitives has created a situation where it is now extremely easy for a digital circuit designer to produce a circuit design which is virtually untestable in real time. Testing digital circuit at any level is to detect the presence of hardware failures induced by fault in the manufacturing process or by operating stress or wearout mechanism.

It is a process of measuring the testability of the logic circuit at gate level on a digital computer. A model of circuit consists of a number of a node description. The testability value of a circuit is a mean of the testability value of each node in the circuit.

CAMELOT approach is used to develop the simulator and its called Digital Test Simulator (DTS). It is used to measure the testability values of the combinational logic circuit, beside designing process and developing test generation strategies. It is based on the algorithmic method where the testability of each node is a function of its controllability and abservability values. There testability of the circuit is a mean of the node's testability values.

OBJECTIVE

The objective of this project is produce a CAD tool that can be used to measure the testability of the combinational logic circuit at the gate level. It will help engineer to reduce the following problems:

1. Labour reduction
2. Timescale reduction
3. Error reduction and design intergrity

CONTROLLABITY

The controllability denoted as CY is the ability to control the fault-free value at the node from the primary inputs. Its values are constrained to be in range 0 and 1. The maximum value of 1 represents an input or output node, where it is easy to establish a logic 1 as it is a logic 0.

In general, the input to the device will not be 100% controllable. The output CY s of the device must

[1]Current address: Department of Electronics and Computer Science, University of Southampthon, S09 5NH, United Kingdom.

be therefore must be therefore take into account both the ease of transfer accross the device and the CY values on the inputs. The expression used to calculate each output CY is:

$$CY \text{ (output node)} = CTF \times f(CY \text{ s (input node)})$$

where CTF is controllability transfer factor of device for the output desired, and function f combine with CYs for all inputs are able to control the particular output. CTF is only dependent on the logic function of the device and not the position of the device in the circuit. For quantifying the CTF is given by:

$$CTF = 1 - \frac{N(0) - N(1)}{N(0) + N(1)}$$

where $N(0)$ is total number of ways that a 0 can be produced on the device output, and $N(1)$ is total number of ways of producing a 1. If $N(0)$ and $N(1)$ are equal then the CTF is 1. Generally between $0 < CTF < 1$. The combinational logic gates, the values of $N(0)$ and $N(1)$ can be obtained from the truth table. For AND gate, the value of CTF is 0.5.

The output OY value for a logic gate is a function of its CTF and some of the inputs CY values that control its value. The CY of a particular output is given by:

$$CY(\text{z}) = CTF(\text{z}) \times f(CY \text{ (input nodes controlling z)})$$

OBSERVABILITY

The second factor in testability is observability denoted as OY. It is the ability to cause a change from the fault-free value at a primary output to result form a change from the fault-free at the node level.

The observability transfer function (OTF) from input I of a gate to its output 0 denoted as $OTF(I - O)$, is 0 if no way of propagating fault-effect data between two points, and I if propagation always takes place. It will lie between these extremes.

A formal way of defining sensitivity transfer is contained the concept of the D-algorithm [1,5,6]. It is the concepts of propagating and non-propagating D-cubes (PDCs and $NPDC$s respectively). Each PDC identifies the sensitive path input, the fixed value combinations that support the path and the sensitive path output. On the other hand, each $NPDC$ identifies the sensitive path input, the fixed value combinations that block the path and the insensitive output. The total number of an input-output pair $(I - O)$ quantifies the number of possible ways of propagating fault-effect data and denote this number by $N(PDC : I - O)$. The total number of distinct but unpolarised $NPDC$s form I to O, denoted as $N(NPDC : I - O)$, indicate the number of ways of blocking the transfer of sensitivity across the device.

The device's OTF can be measured by the ratio:

$$OTF(I - O) = \frac{N(PDC : I - O)}{N(PDC : I - O) + N(NPDC : I - O)}$$

For instance, for the two input NAND gate, the OTF value is 0.5.

An alternative way of OT F calculation is derived from the equation:

$$OTF(I - O) = \frac{N(SP : I - O)}{N(SP : I - O) + N(IP : I - O)}$$

where $N(SP : I - O)$ is the number of different sensitive paths from input I to output O, and $N(IP : I - O)$ is the total number of insensitive path.

The procedure to calculate the OY transfer is follows. Start at the gate output node where OY is 1, and transfer this value to the circuit primary output to obtain the values for the OYs of device at the primary outputs. This process should be repeated for each circuit node and it is time consuming. The alternative methods is as follows. Start at the circuit's primary outputs and workback through the circuit calculating each nodal OY value as the node is encountered. Its requires a single pass per primary output and then its can be used in order to calculate OY transfers across two gates:

$$OY(A-C) = OY(C-C) \times OY(B-C) \times OY(A-B)$$

where $OY(A-A)$ is OY of node A at node A (defined as having the value 1), $OY(A-B)$ is OY of node A at node B, $OY(B-C)$ is OY of node B at node C, $OY(C-C)$ is OY of node C at node C, and $OY(A-C)$ is OY of node A at node C (defined as having the value 1).

The multiplicative property of CYs can be more easily understood by considering the sensitive inputs and outputs of gate on the sensitive path to be connected by a relay contact. If the contact is closed, then the path propagates and on the other hand if it open then the path is blocked. The chance of the operation of each contact is determined by CYs of the support inputs to the gate concerned and its transfer factor.

TESTABILITY

Both the controllability and observability measures relate to node in the circuit, rather than to the devices, although the devices are instrumental in dictating the values produced for each node. Testability must be a composit function of both controllability.

A measure of testability denoted as TY can be obtained from the product of CY and OY values for a node and the relationship employed in CAMELOT is:

$$TY \text{ node} = CY \text{ node} \times OY \text{ node}$$

which satisfies the requirement that:

$$TY = \begin{cases} 0 & \text{if either } CY \text{ or } OY \text{ is 0,} \\ 1 & \text{if } CY \text{ and } OY \text{ are 1,} \\ (0,1) & \text{if } CY \in (0,1) \text{ and } OY \in (0,1). \end{cases}$$

The testability value for the circuit should be a measure of the average difficulty of producing a test for a node in the circuit. The values used is the arithematic mean of the individual nodal TYs given by:

$$TY\,(\text{circuit}) = \frac{(TY:\ \text{nodes})}{\text{No. of nodes}}$$

Computation Algorithm for Testability

Briefly, the algorithm for calculating testability values is as follow:

Start	Initialize all the variables and constant.
Step 1	Prepare, read in and check a description of the circuit connectivity.
Step 2	Calculate nodal CY values, starting at the circuit's primary input and progressing through the circuit. This requires both the connectivity description and a library containing gate CTF values.
Step 3	Calculate nodal OY values, starting at the circuit's primary output and working back towards its primary inputs. Here, the library is interrogated to obtain OTF values for the gates in the circuit.
Step 4	Calculate nodal TY values from the nodal CY and OY values.
Step 5	Calculate and present the circuit's average testability and interpret the result.
End	Stop.

MODELLING

The modelling of digital test simulator is given below and only valid for combinational logic circuit.

Gate Level

The simulation elements are logic gates such as AND, OR, NAND, NOR, INVERTER, BUFFER, XOR and XNOR. Each element has only one output terminal and the maximum input terminal for any gates.

Circuit Topological

The circuit topological consists of three parts such as element coding, external input and observe node. A combination a logic circuit consists of a number of logic gates. In element coding each gate represented by the following syntax:

$$\langle G_i \text{ Total Gatetype Input}_1 \text{ Input}_2 \text{ Output} \rangle$$

It consists of five items such as gate, total input, gate name, input node and output node. The gate denoted as G_i is represents gate indentifier i in the circuit. The second item is total input represents the number of input per gate. The minimum value of total input is one for example INVERTER and BUFFER. The third item is gate name. There are eight logic gates that valid in simulation such as AND, OR, NAND, NOR, INVERTER denoted as INV, BUFFER denoted as BUFF, XOR, XNOR. The fourth and fifth item are input nodes while the last item is output node. Both of them are represented by any integer numbers that indicates the input and output node respectively.

The external input consists of a list of input node for each gate which needs an external input and the observe node consists of a output node where the testability values to be observed.

FUNCTIONAL REQUIREMENT OF DTS

The digital test simulator should be able to:

1. Simulate the combinational logic circuit consists of gate such as AND, OR, NAND, NOR, INV, BUFF, XOR and XNOR.
2. Calculate the value of controllability and observability for each input and output gates.
3. Calculate the testability value for each node in the circuit and determines the average value of testability for a circuit.
4. Check all user inputs for validity and generate appropriate message if invalid input is detected.
5. Keep the user's input data into a text file.
6. Allow user to modified the input data.
7. Keep the calculation value of *TY*, *CY* and *OY* into a output text file.
8. Allow the user to print out the result.
9. Should be able to display the result in the forms of histogram and table for *TY*, *CY* and *OY* values.

DATA STRUCTURE METHOD

As mentioned before that the modelling consists of three parts, these are element coding, external input and observe node. Each part has it own data structure. The method of the data structure used is linked list.

Element Coding

The number of element coding depends on the number of gates in circuit where each of gate has it's own node description. In pascal its represented by a record as shown below.

```
element     = ^ nodeelement ;
nodeelement = RECORD
              gatenode : string [4];
              totinput : integer ;
              gatename : string [4];
              inputnode: array [1..0] of string [4];
              outputnode : string [4];
              tottextinpgate, totintinpgate : integer;
              CYinput, OYinput: array [1...10] of real:
              CYoutput, OYoutput, TYoutput: real;
              CYsuccess, OYsuccess: boolean;
              path : array [1..10] of pathgate;
```

```
        gatefanout : string [4] ;
        gatecrossed, totpath : integer ;
        next : element;
    end;
```

The gatenode, totinput, gatename, inputnode, and outputnode are taken from a node description. The *CY* input, *OY* input, *CY* output, *OY* output, *TY* output, tottextinpgate, totintinpgate, path, gatefanout, gatecrossed and totpath are obtained from calculation. The *OY* success and *CY* success are also obtained from calculation process. The field of next is used build a linked list with other record.

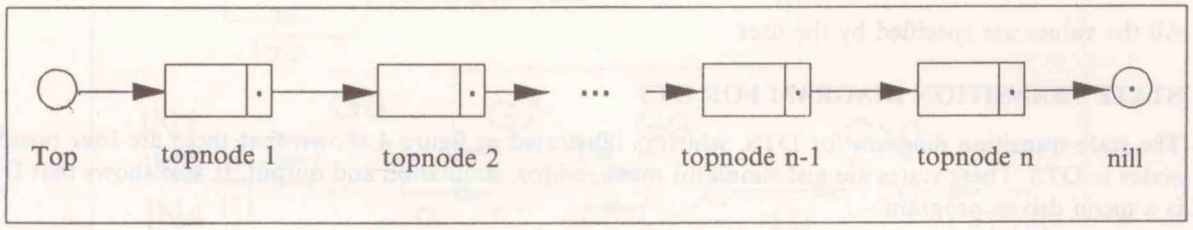

Figure 1: Linked List of Element coding

External Input

It is a list of circuit's primary input node which need an external input. In pascal, it is represented by a record as shown below.

```
subelement  = ^ nodeitem;
nodeitem    = record
                item : string [4];
                next : subelement;
              end;
```

The linked list of external input node is show in figure 2 below.

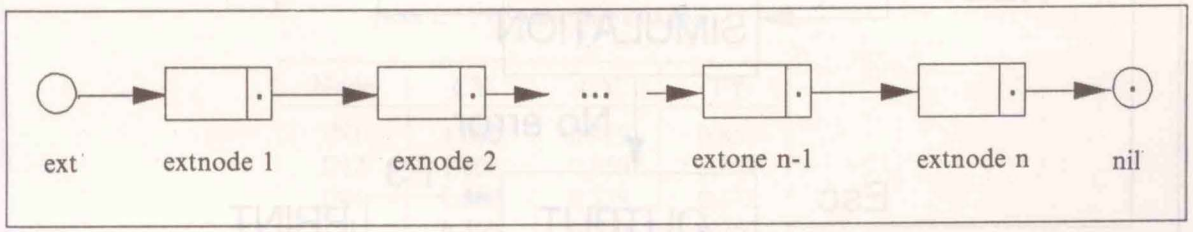

Figure 2: A Linked List of External Input

All the values are spesified by the user.

Observe Element

It is a list of circuit's primary output node where the outputs to be observed. In Pascal it is represented by a record similar as a record in external input node above.

The linked list of observe output node is show in figure 3 below.

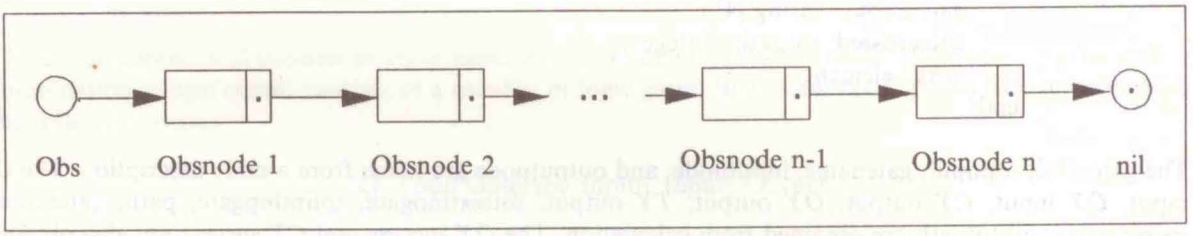

Figure 3: A Linked List of Observe Output

All the values are specified by the user.

STATE TRANSITION DIAGRAM FOR DTS

The state transition diagram for DTS, which is illustrated in figure 4 shown that there are four possible states in DTS. These states are test simulator menu, editor, simulation and output. It also shows that DTS is a menu driven program.

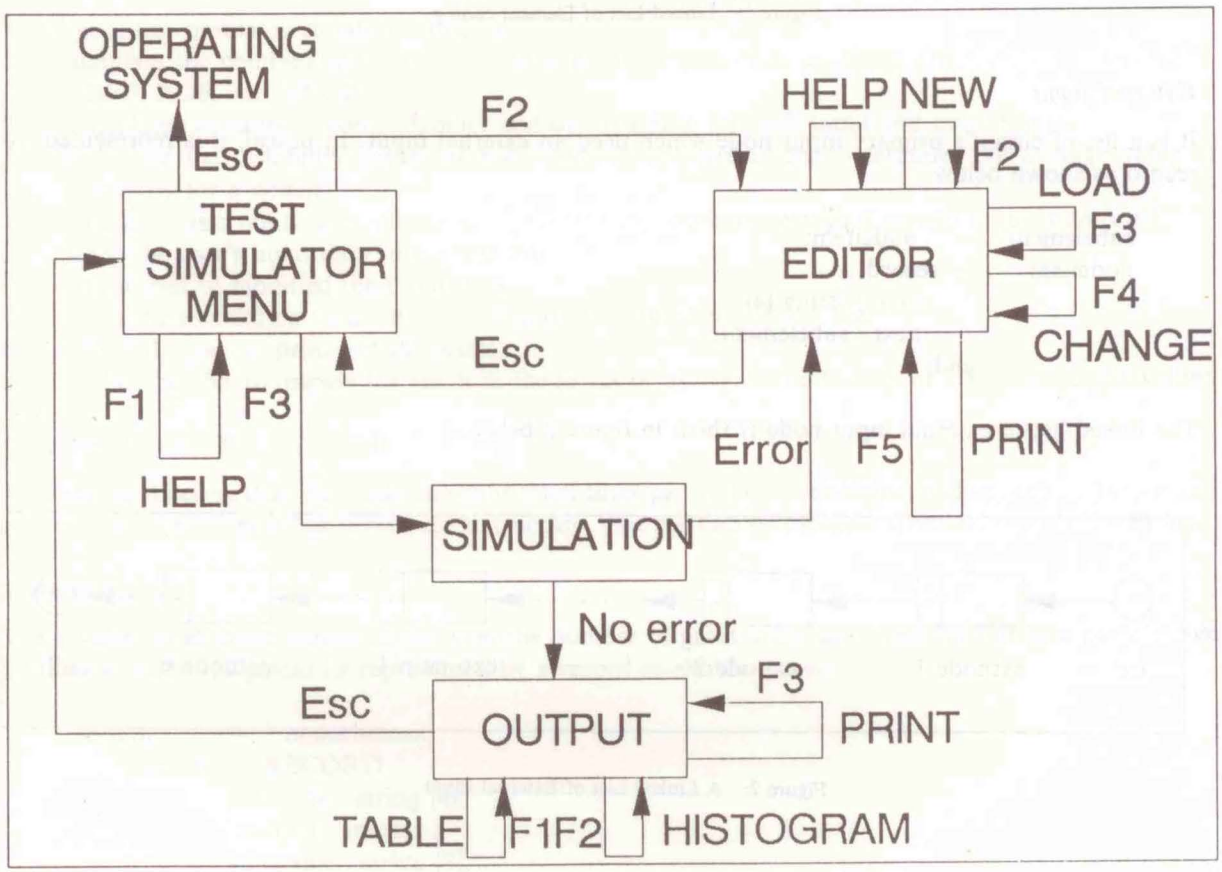

Figure 4: State Transition Diagram for DTS

RESULTS

The simulator has been tested with a few combinational logic curcuits from a simple circuit to more complex circuits. The following sections covers an example of testing simulation for a one bit full adder.

One Bit Full Adder

The one bit full adder circuit consists a few basic logic gates as shown in figure 5 below.

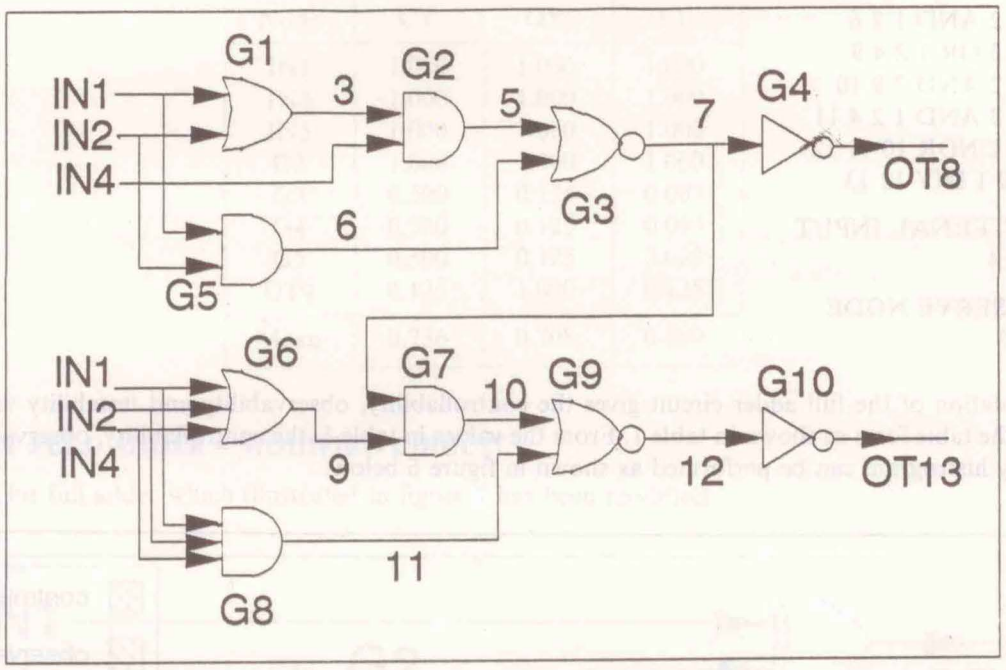

Figure 5: One Bit Full Adder Circuit

To simulate the full adder circuit, is is necessary to check the circuit connectivity. The node description as the input to the simulators are listed below.

TITLE: Full Adder

ELEMENT CODING

G1 2 OR 1 2 3

Table 1: TY Values for Full Adder

Node	CY	OY	TY
IN1	1.000	0.859	0.859
IN2	1.000	0.859	0.859
IN4	1.000	0.578	0.578
G1	0.500	0.500	0.250
G2	0.375	0.250	0.094
G3	0.219	0.250	0.055
OT8	0.219	1.000	0.219
G5	0.500	0.188	0.094
G6	0.250	0.109	0.027
G7	0.117	0.125	0.015
G8	0.250	0.059	0.015
G9	0.092	0.000	0.000
OT13	0.092	1.000	0.092
Mean	0.432	0.444	0.243

G2 2 AND 3 4 5
G3 2 NOR 5 6 7
G4 1 INV 7 8
G5 2 AND 1 2 6
G6 3 OR 1 2 4 9
G7 2 AND 7 9 10
G8 3 AND 1 2 4 11
G9 2 NOR 10 11 12
G10 1 INV 12 13

EXTERNAL INPUT
1 2 4

OBSERVE NODE
8 13

The simulation of the full adder circuit gives the controllability, observability and testability values and listed in the table form as shown in table 1. From the values in table 1, the controllability, observability and testability histrogram can be performed as shown in figure 6 below.

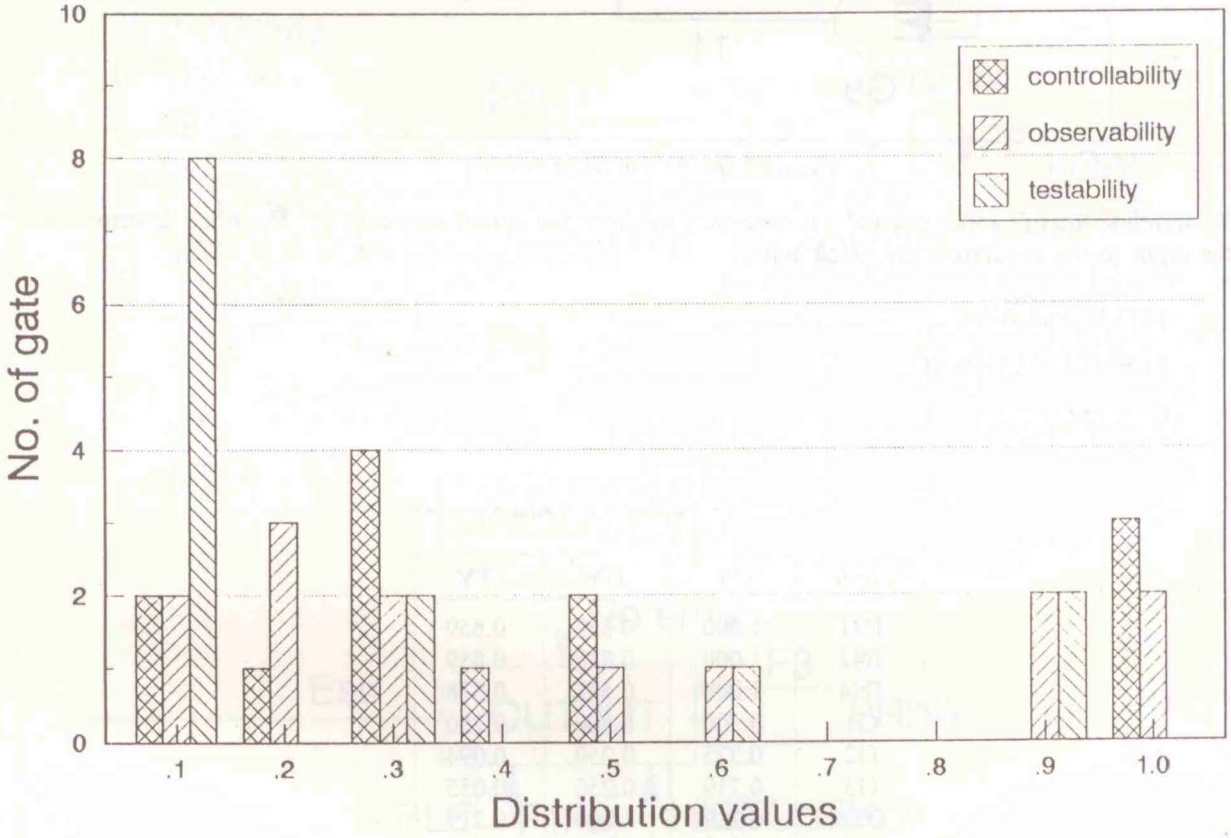

Figure 6: One Bit Full Adder Circuit

As can be seen in the results in the forms of table and histogram above, the CY, OY and TY have the small mean values so, the circuit is not testable. It means, it is not ease to control and to observe the node in the full adder circuit. This full adder should be modified to make it more testable. The modification can be done by replacing a few logic gates with the XOR logic gate. The simulation of the modified circuit will discuss in the following section.

Table 2: Testability Values for Full Adder Modified Circuit

Node	CY	OY	TY
IN1	1.000	1.000	1.000
IN4	1.000	1.000	1.000
IN5	1.000	1.000	1.000
G2	1.000	1.000	1.000
G3	0.500	0.125	0.063
G4	0.500	0.125	0.063
G5	0.500	0.125	0.063
OT9	0.125	1.000	0.125
Mean	0.736	0.708	0.590

ONE BIT FULL ADDER – MODIFIED CIRCUIT

The one bit full adder which illustrated in figure 7 has been modified.

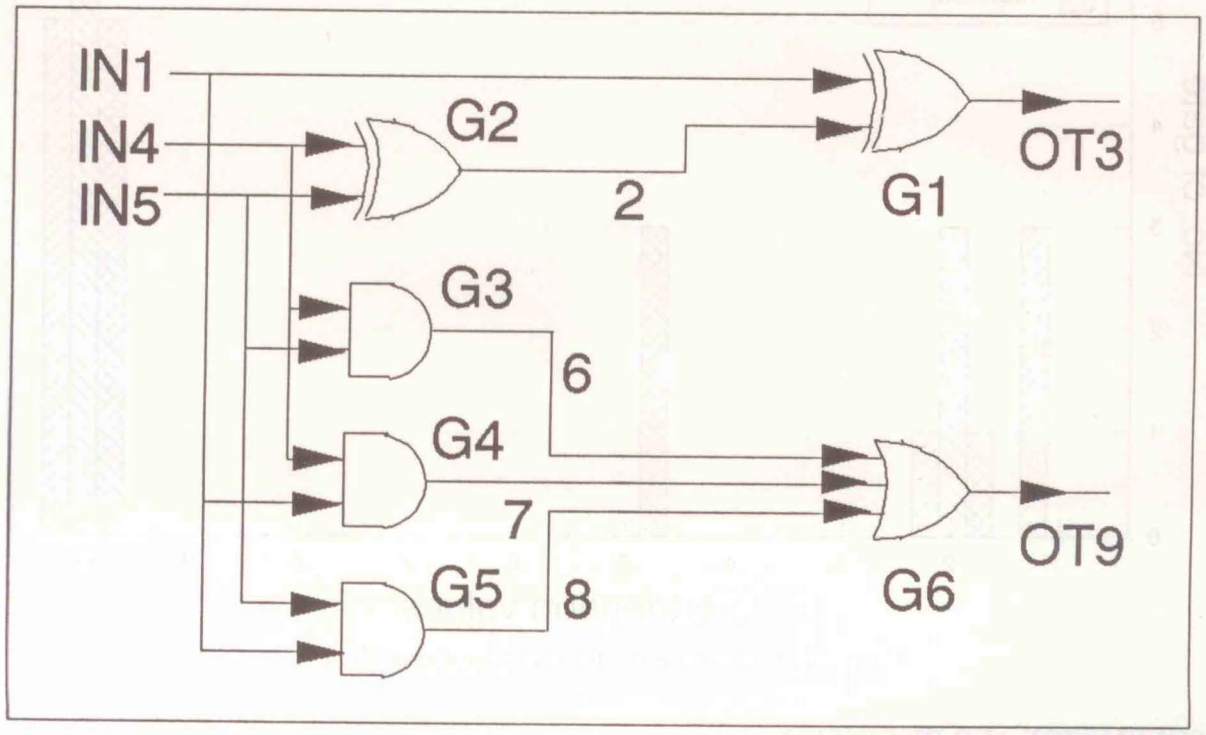

Figure 7: One Bit Full Adder - Modified Circuit

The node description can be listed as follow.

TITLE : Full Adder

ELEMENT CODING

G1 2 XOR 1 2 3
G2 2 XOR 4 5 2

G3 2 AND 4 5 6
G4 2 AND 4 1 7
G5 2 AND 5 1 8
G6 3 OR 6 7 8 9

EXTERNAL INPUT
1 4 5

OBSERVE NODE
3 9

The values of CY, OY and TY for the full adder circuit produced by the simulator are listed in table 2. The histograms for the full adder modified circuits are shown in figure 8 below.

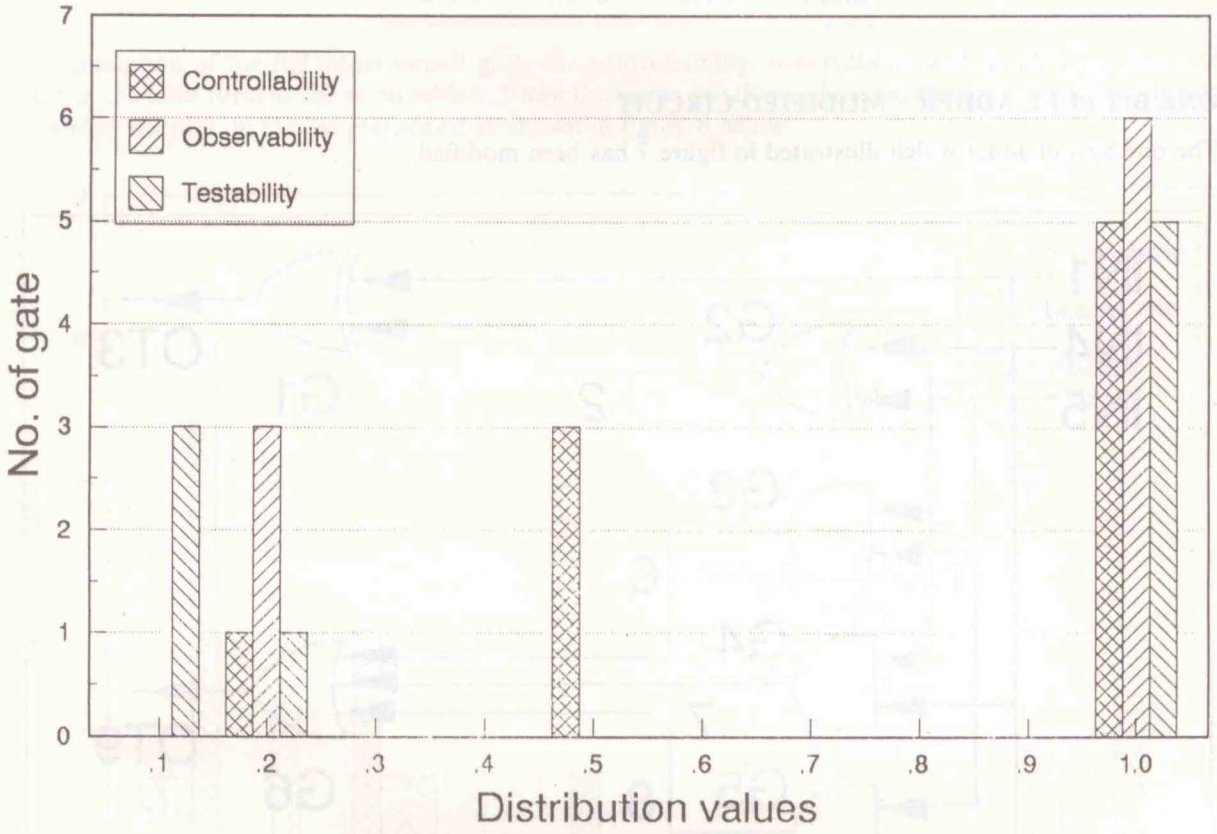

Figure 8: One Bit Full Adder - Modified Circuit

CONCLUSION AND DISCUSSION

The DTS is used to measure the values of the CY, OY and TY, following that the table and histrogram are build from its values. From the two examples of circuit simulation, it is cleared that the nodal with low CY, OY and TY can be identify and the user needs to do any improvement to the circuit to make it more testable.

The DTS has several capabilities, one of them is that all the element codings and other data input will automatically save into a file. It also save results into files and allows a user to print the files. The DTS will check the user inputs from file for validity. An appropriate message will be generated if invalid input is detected, and it allows the user to change the data inputs. It also can display the distribution histrogram on screen and print it.

As a conclusion, this project has been successful in achiving its objectives. It is hoped that by using this simulator the circuit designing process at the gate level can be done in a short time. This simulator can be used in both, designing process and developing test generation strategies. The DTS program produced is for gate level combinational logic circuit simulation.

REFERENCES

1. Bennetts R. G., *Design of Testable Logic Circuit*, Addison-Wesley Pub. Co., 1984.
2. Borland International, *Turbo Pascal Version 5.0: Reference Guide*, Borland International, USA, 1988.
3. Borland International, *Turbo Pascal 5.0: User's Guide*, Borland International, USA, 1988.
4. Miczo Alexander, *Digital Logic Testing and Simulation*, Harper and Row Publisher, 1986.
5. Mohamed bin Othman and Bambang Sunaryo bin Suparjo, Logic Simulator: Sequential Logic Minimization, *Journal of Technology*, 1991.
6. Wilkins B. R., *Testing Digital Circuit : An Introduction*, Van Nastrand Reinhold Co. Ltd, 1986.