DEVELOPMENT OF ATPG USING BASICALGORITHM PROCESSES

[1]Mr. Parth Thobhani, [2]Dr. Usha Mehta
Institute of Technology, Nirma University, Ahmedabad, Gujarat[1,2]
parth.thobhani@yahoo.com

ABSTRACT

To test the functioning of the fabricated circuit (IC) Automatic Test Pattern Generator (ATPG) is becoming increasingly important as designs become more complicated. Over the years ATPG tool development process is the research interest for testing the VLSI circuits. ATPG tool development mainly comprises of the three processes if we consider the Deterministic Test Pattern Generation part, which are Fault Equivalence, Line Justification, Fault Propagation. In this paper, all processes are implemented for the combinational circuit with some limitations. Considering the single stuck-at faults, the fault list becomes lengthy as the circuit under test has a large number of nets. To reduce the total number of faults to be tested Fault Equivalence method is used. Testability measure means finding out Controllability and Observability of a circuit. Controllability helps us to find out the easily controllable net during the line justification process. The actual test vector generation of every fault has the two most important processes that are Line justification and Fault propagation. In this paper codes written for fault equivalence, controllability, line justification, and fault propagation in Perl language for 2 fanin-fanout gates. All developed programs are generic and can be used for any combinational circuit which has non reconvergent fanouts. For all processes, the input required is the circuit netlist file (.txt file). All program works dynamically i.e. it also gives correct results if the circuit netlist is not in the order of input to output direction. For reconvergent fanout, it has limitation i.e. to generate the test vector for the fault which require backtracking. For the rest of the faults, it works fine.

Index Terms—Fault Equivalence, Automatic Test Pattern Generation, Test Generation Process

I. INTRODUCTION

Testing history reveals that Structural testing with fault models is a feasible way to test a chip. It reduces the time required for testing a chip drastically. It can automate the test pattern generation. To form initial stage ATPG using basic concepts of fault reduced set using fault equivalence method, testability measures, line justification, and fault propagation. These processes are developed using Perl programming. The commercial ATPG tools are available in the market like Tessent FastScan by Mentor Graphics, TetraMAx ATPG by Synopsys, and open-source ATPG by Atlanta. The cost of these ATPGs is high and they are complex. My aim to develop an open-source ATPG that is freely available and it should be less complex. Further, this type of open-source ATPG tool will be helpful for academic learning purposes. This tool will extract the circuit netlist which was written manually. The tool will do the Fault Equivalence. It will find Controllability & Observability. It will find the Test pattern for individual fault as well as the complete test set for the given circuit. The main objective is to understand the fundamentals of ATPG, Perl scripting language, process and development of the EDA tool. Most of the ATPG tool accepts circuit design in the form of a netlist which is generally text file (.txt) and by processing on them it generates the test vector set for a particular fault. We observed that in many IEEE transactions people are using ISCAS benchmark circuit format. Here, we adopted the ISCAS 85 benchmark circuit format for all processes. Also, the C17 combinational circuit is being used widely as it has reconvergent fanout at several nodes and at primary inputs. Figure 1 shows the ISCAS C17 combinational circuit. Figure 2 shows the ISCAS C17 combinational circuit netlist file [2]. The first entry of each line is the address. The address is the unique number that differentiates each node/net in the circuit. The second entry is nothing but the more meaningful name of the node. Third entry indicates the function performed by the gate driving this node. It

includes input, from, and any gate. Input means primary input net, from means branch of fanout stem and gate type, which means it is the output net of the mentioned gate. Fourth and fifth entry indicates no. of fanout and fanin i.e. the number of gates driven by the node and number of nodes driving the gate that drives this node [2]. No. of fanout and fanin entry is not required for type "from". In the original format at each line, 6th and 7th entry is also there, which is available stuck at fault in reduced fault set for that node. But, in this paper, we identified it by fault equivalence and then we use it for further processes. The last two entries are fanin nets (except for type "from", as it has stem net no. as last entry). From figure 2 we found that net 23 is an output net as it has 0 fanout, it is of type NAND, it has 2 fanin 21 and 19. Hence, we can extract the whole connectivity of the circuit by using this type of standard netlist format.

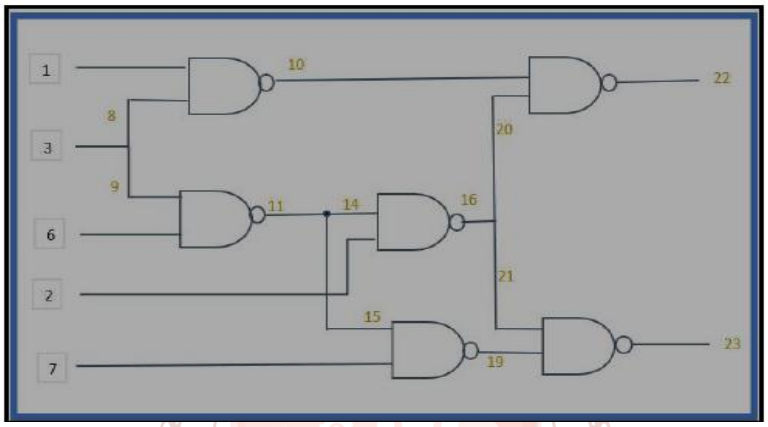

Fig. 1. ISCAS'85 C17 Circuit

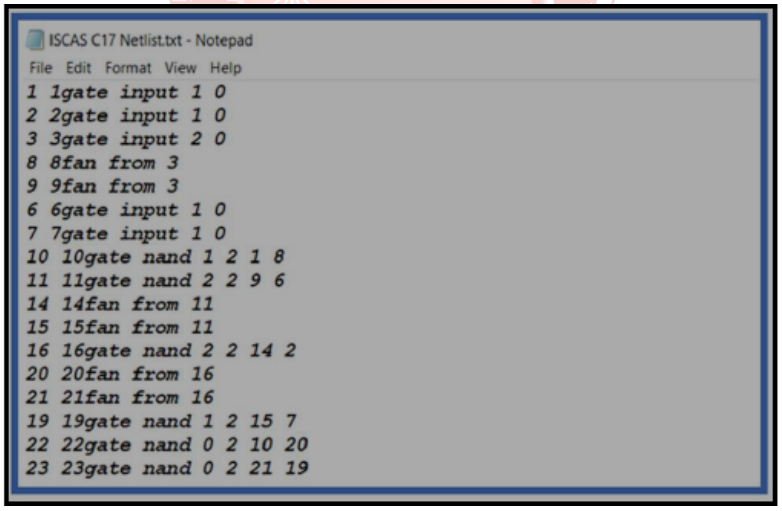

Fig. 2. ISCAS'85 C17 circuit Netlist

II. FAULT EQUIVALENCE AND CONTROLLABILITY

CONTROLLING AND INVERSION VALUES FOR GATES

The controlling value of the gate is the value by which we can control the output of the gate regardless of the other inputs. Like, for the AND gate and NAND gate it is 0. Inversion value indicates the nature of the gate. Like, for the NOR gate inversion value is 1 as it has inverting nature and for the OR gate, the inversion value is 0. Table I shows the controlling and inversion values of all the basic gates.

TABLE I

CONTROLLING AND INVERSION VALUES

Gate Type	Controlling value (c)	Inversion value (i)
AND	0	0
NAND	0	1
OR	1	0
NOR	1	1
NOT	0 and 1	1

FAULT EQUIVALENCE

Two single stuck-at fault f and g of a combinational circuit are said to be equivalent if they have the same output function under faulty condition [1]. We are not able to distinguish equivalent faults as they have the exact same set of test vector. Figure 3 shows the reduction in the fault of a particular gate using the fault equivalence method. Like, for AND gate stuck at 0 (sa0) fault at the output is equivalent to stuck at 0 faults at inputs individually. So, only output stuck at 0 faults is retained and input stuck at 0 faults eliminated, as it gives the same test vector set. For the stem, wire and primary output (PO) both the sa0 and sa1 faults are present. Figure 4 shows an example of fault reduction using fault equivalence for a combinational circuit. The fault collapsing ratio is defined as the total no. of faults after fault reduction to the total no. of initial faults [1]. Initially, the total no. of faults for the given combinational circuit is 32. After fault equivalence total no. of faults to be tested is 20. So, the collapse ratio is 20/32 that is 0.625.

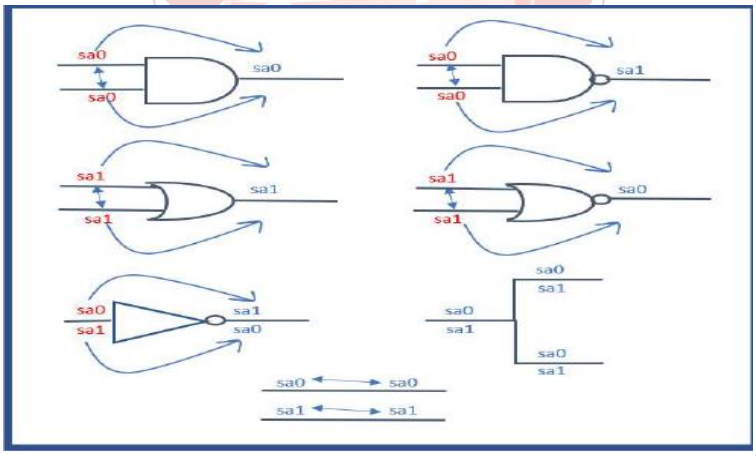

Fig. 3. Logically equivalent faults

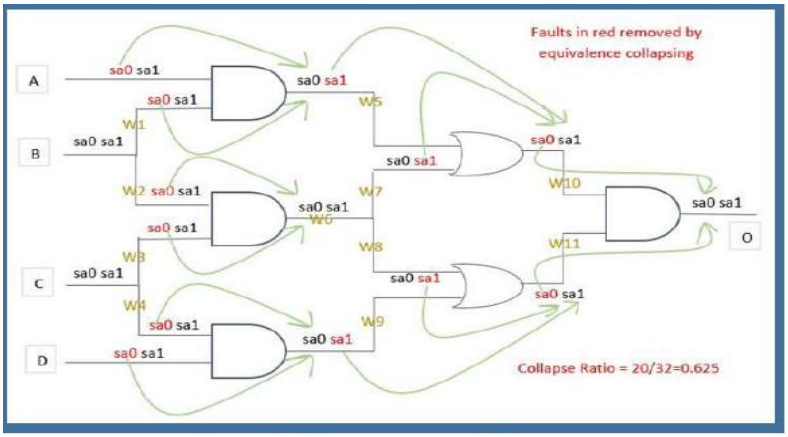

Fig. 4. Fault equivalence example circuit

CONTROLLABILITY

Testability measures like controllability 0, controllability 1, and observability of combinational circuits are discussed here. Controllability for a digital circuit is defined as the difficulty of setting up a particular net to logic value 0 or 1. Controllability 0 and 1 calculation of a particular net in the combinational circuit helps to tackle line justification problems where we need to select any one of the inputs of a gate to justify its output value. It selects the net which is easily controllable. The observability of a digital circuit is defined as the difficulty of observing the logic value of a particular net [5]. Observability calculation of a particular net in the combinational circuit helps to select the fanout branch of the stem which is easily observable [5]. In this paper, testability measures controllability 0 and controllability 1 value of every net is calculated using the Perl scripting language and used those values for the test pattern generation process. For the fanout, the controllability values remain the same as the net from where it is emerging. Controllability of primary inputs is set to (1,1) and the calculation flow of controllability values of all nets is from primary input to output. The flow chart of the controllability function is shown in Figure 5 whereas Figure 6 shows the controllability calculations of the ISCAS C17 combinational circuit.

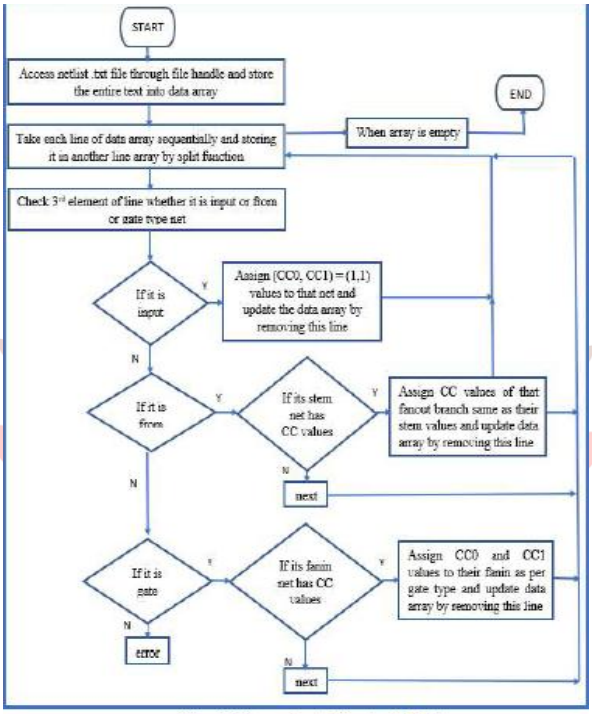

Fig. 5. Flow chart of controllability

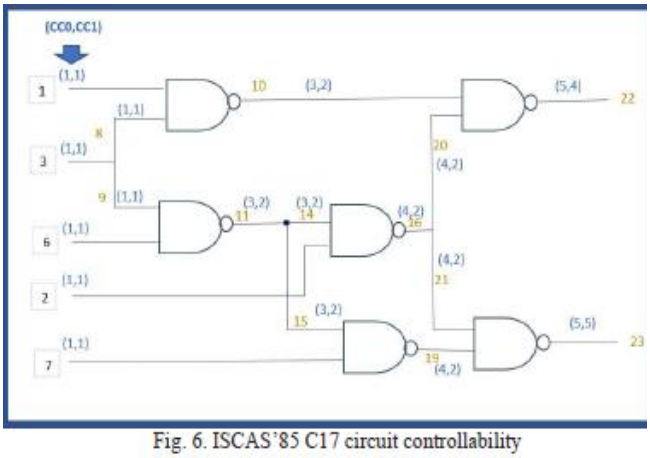

Fig. 6. ISCAS'85 C17 circuit controllability

III IMPLEMENTED ALGORITHM AND EXPERIMENTAL RESULTS

In this chapter, the basic process of test pattern generation using line justification and fault propagation is discussed.

AUTOMATIC TEST PATTERN GENERATION

ATPG stands for Automatic Test Pattern Generation or Automatic Test Pattern Generator. It is an automatic method/tool which generates the test vector set that is required to test the digital circuit or IC after its fabrication to check whether it has a defect or not. A defect is nothing but an error introduced into a device during the manufacturing/fabrication process. A defect changes device behavior, which can be identified easily with the fault model [1]. A fault is detectable if generated test vector for that fault gives the different logic value at any one of the primary outputs. Although few faults are undetectable if it gives the same logic value under faulty and fault-free circuit condition. Those faults are known as undetectable faults. Any ATPG algorithm from basic to advanced has three main processes that are Fault Sensitization, Line Justification, and Fault Propagation. In reconvergent fanout, circuit Backtracking is also an important process. Without Backtracking many faults are undetectable in this complex digital circuit's era.

FAULT SENSITIZATION AND LINE JUSTIFICATION

Fault Sensitization means assigning the opposite value than the stuck-at value of a particular net to test that single stuck-at fault [1]. To enable that value, we need to assign specific values to their primary inputs, which is line justification. Figure 7 shows the line justification algorithm which is implemented in this paper.

```
LineJ (net, assigned_value)
{
    set net to assigned_value;
    if (net is a PI)
        then return;
    elsif (net is a logic gate)
        c = controlling value of net;
        i = inversion value of net;
        invalue = (assigned_value xor i);
        if (invalue != c)
            foreach input j of net
                LineJ (j, invalue);
        else
            if
                (select one input j of net, which is easily controllable);
                LineJ (j, invalue);
            else
    else
        LineJ (stem_net, assigned_value);
}
```

Fig. 7. Line Justification Algorithm

FAULT PROPAGATION

Fault propagation is the process of transferring fault effect to any primary output to make it observable. To make this happen we need to assign the non-controlling value at the other input of the gate in the fault propagation path [1]. Here, Line justification again comes into the picture to activate the non-controlling value. Figure 8 shows the fault propagation algorithm which is being used in this paper.

```
Propagate (net, stuck_value)
{
        set net to stuck_value;
        if (net is a PO)
                then return;
        elsif (net is stem)
                if
                        (select one branch j of stem, which is easily observable);
                        Propagate (j, stuck_value);
                else
        else
                k = gate output of net (i.e. fanout);
                c = controlling value of k;
                i = inversion value of k;
                Foreach input j of k other than net
                        LineJ (j, !c);
                out_value = (stuck_value xor i);
                Propagate (k, out_value);
}
```

Fig. 8. Fault Propagation Algorithm

IV. RESULTS AND DISCUSSION

The combinational circuit shown in Figure 9 is chosen to carry out simulation of Fault Equivalence, Controllability, and Test vector generation Perl code. In this circuit, all basic gates AND, NAND, OR, NOR and NOT are selected with one stem which has two fanout branches. Figure 10 shows the input circuit netlist file which is a .txt file. The netlist file is as per standard ISCAS format [2]. Here, the modification is reduced fault set is not written directly in netlist file rather it is generated using fault equivalence code

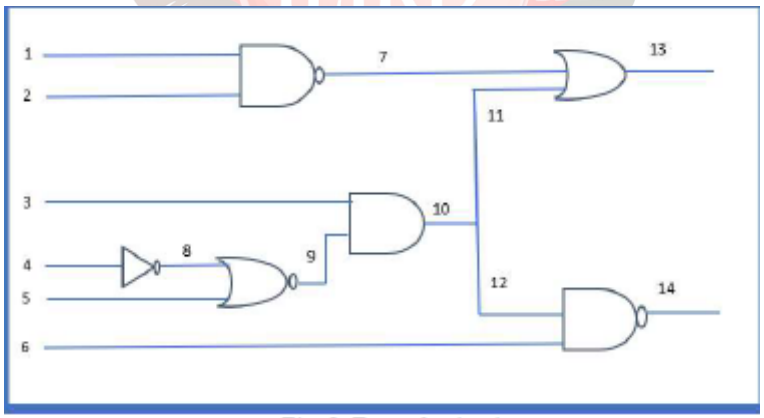

Fig. 9. Example circuit

Netlist.txt - Notepad

File Edit Format View Help

```
1 1gat inpt 1 0
2 2gat inpt 1 0
3 3gat inpt 1 0
4 4gat inpt 1 0
5 5gat inpt 1 0
6 6gat inpt 1 0
7 7gat nand 1 2 1 2
8 8gat not 1 1 4
9 9gat nor 1 2 8 5
10 10gat and 2 2 3 9
11 11fan from 10
12 12fan from 10
13 13gat or 0 2 7 11
14 14gat nand 0 2 12 6
```

Fig. 10. Input Netlist File

Figure 11 shows the fault equivalence result with the fault collapse ratio. So, total no. of fault to be tested is reduced from 28 to 20. Figure 12 shows the controllability result which shows the controllability values of all fourteen nets. It is used to find a particular net that is easily controllable.

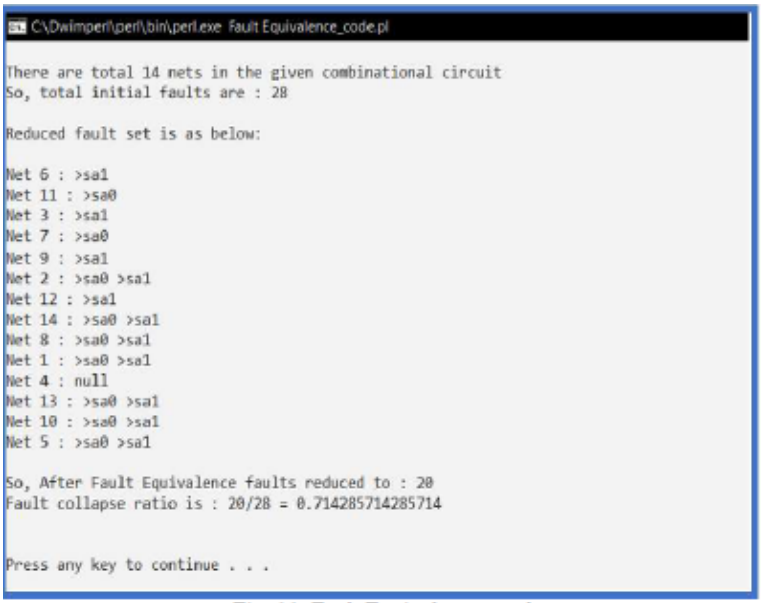

Fig. 11. Fault Equivalence result

Fig. 12. Controllability result

Table II indicates the fault collapsed ratio and percentage reduction in faults to be tested. The fault collapsed ratio is the ratio of no. of faults remained to the total no. of faults before fault equivalence.

TABLE II

TEST PATTERNS FOR ALL FAULTS FROM FAULT EQUIVALENCE DIRECTORY

Net no.	Faults to be tested	Test vector	Fault free / Faulty value
1	sa0	110XXX	0/1
	sa1	010XXX	1/0
2	sa0	110XXX	0/1
	sa1	100XXX	1/0
3	sa1	11010X	0/1
4	----	----	----
5	sa0	11111X	0/1
	sa1	11110X	1/0
6	sa1	XX1100	1/0
7	sa0	0X0XXX	1/0
8	sa0	11100X	0/1
	sa1	11110X	1/0
9	sa1	111X1X	0/1
10	sa0	11110X	1/0
	sa1	110XXX	0/1
11	sa0	11110X	1/0
12	sa1	XX0XX1	1/0
13	sa0	0XXXXX	1/0
	sa1	110XXX	0/1
14	sa0	XXXXX0	1/0
	sa1	XX1101	0/1

V. CONCLUSION AND FUTURE SCOPE

The generic Perl code is developed for common processes of Combinational ATPG tool like Fault Equivalence, Controllability, Line Justification, and Fault Propagation. Using all these processes test vectors are generated for nonreconvergent fanout circuits. The developed code is generic for a maximum 2 fanout fanin circuit. The major challenges faced in this work is the reading of each element of the input netlist file and the searching of the net other than the fault site, during the propagation of fault effect. As the standard ISCAS netlist format has no fanout details on the same line.

Further, we can develop backtracking function which is useful for test vector generation of combinational circuits which has reconvergent fanout. The above carried out processes with Backtracking function and Observability function addition will make this work a complete Combinational ATPG tool for deterministic test pattern generation. Then, it can be used as an open-source tool for academic learning purposes.

REFERENCES

[1]Michael Bushnell and Vishwani Agrawal, "Essentials of Electronic Testing". Springer Publication,2000, pp. 57-176.

[2] David Bryan, Iscas'85 benchmark circuits and netlistformat, North Carolina State University.

[3] Vaishali Dhare, Usha Mehta, "Implementation of Compaction Algorithm for ATPG Generated Partially Specified Test Data", International Journal of VLSI design & Communication Systems, Pg.no.93-103, Vol.4, No.1, February 2013.

[4] Vaishali Dhare, Usha Mehta, "Development of Controllability Observability Aided Combinational ATPG with Fault Reduction", Lecture Notes in Computer Science, Recent Trends in Networks and Communications, Springer, Pg. No 682-692, July 2010.

[5] L.H. Goldstein, "Controllability/Observability Analysis of digital circuits", IEEE Transactions on Circuits and Systems, Vol. 26, September 1979, 1984, pp. 685-693.