M.Tech.(Computer Science) Dissertation Series

SAT Solver based Multi Cycle Droop Fault Testing

a dissertation submitted in partial fulfillment of the requirements for the M.Tech. (Computer Science) degree of the Indian Statistical Institute

By

Santanu Bhowmick

Roll No. : CS0702

July 16th 2009

under the supervision of

Prof. Susmita Sur-Kolay

Advanced Computing and Microelectronics Unit

Indian Statistical Institute

203, B.T. Road

Kolkata-700108

Indian Statistical Institute

Certificate of Approval

This is to certify that the thesis entitled **SAT Solver based Multi Cycle Droop Fault Testing** submitted by Santanu Bhowmick towards partial fulfillment for the degree of M. Tech. in Computer Science at Indian Statistical Institute, Kolkata, embodies the work done under my supervision.

Prof. Susmita Sur-Kolay
ACMU, ISI, Kolkata
Date :

(Countersigned)
External Examiner
Date :

Acknowledgment

Working on this topic has been a constant source of inspiration and a matter of immense joy for me. I present this topic with a hope that future works in this field will get a glimpse of problems in this area. Doing this project has been the most fulfilling work of my academic carrier as it has not only taught me about the finer details of this topic but also to work in a cooperative environment full of helping hands. I take pride in admitting that there are some people who have guided me and helped me through my entire course of work and without whose help I may not have been able to present this work before you. I express my gratitude to **Prof. Susmita Sur-Kolay**, Advanced Computing and Microelectronics Unit, ISI, Kolkata for her constant guidance and inspiration throughout this project. I sincerely extend my gratitude to **Prof. Bhargab B. Bhattacharya** and all faculty of ACMU for their motivation and cooperation to complete this work. Finally I would like to thank all of my colleagues, class mates, friends, family members and staff of ISI, Kolkata for their support and motivation to complete this project.

Santanu Bhowmick

Abstract

Driven by Moore's Law for more than four decades, the complexity and scale of VLSI integration has reached unforeseeable heights today. The increased density of switching devices and rising frequency has led to large power consumptions per unit area. Due to high frequency of operation and inductive effects of the power grid lines, a noticeable power drop occurs when logic gates within close physical proximity of each other switch simultaneously. This drop in power, known as droop, propagates along the power supply lines, decaying exponentially with spatial and temporal distance from its origin. This is manifest a few clock cycles later, in the form of a reduced power drop at a neighboring via, giving rise to the possibility of timing faults at some gates in that via. Such faults are known as *multi cycle droop faults (MDF)*. In this dissertation, a new approach is taken towards modeling of these faults in combinational circuits, using the concepts of Boolean Satisfiability to provide more flexibility and efficiency in test generation for detection of these faults. Finally, a prototype algorithm to generate test vectors for multi-cycle droop faults in full-scan circuits is presented and discussed.

Contents

1 Introduction **1**

1.1 Power Supply Droop 2

1.1.1 Low frequency Power Droop (LFPD) 3

1.1.2 Mid frequency Power Droop (MFPD) 4

1.1.3 High frequency Power Droop (HFPD) 7

1.2 Previous Works 8

1.3 Scope of this thesis 9

1.4 Organization of this thesis 9

2 Basic Concepts **10**

2.1 Multi-Droop Fault Model 10

2.2 An Overview of the Boolean Satisfiability Method 11

2.2.1 Formula Extraction 12

2.2.2 Formula for Fault Detection 13

3 MDF Detection in Combinational Circuits Using Boolean Satisfiability **15**

3.1 Requirements of an ATPG for MDF 15

3.2 Data Structures 16

3.3 Algorithm for Test Generation for detecting MDF 18

3.4 A working example: c17 23

3.5 Experimental Results 25

3.5.1 Experiment #1 (Logical Neighbors method) 26

3.5.2 Experiment #2 (Random Selection method) 27

3.5.3 Comparison of SAT Solver based approach with Conventional ATPG . . 28

4 Prototype Algorithm for Test Generation In Sequential Circuits 30

4.1 Preliminaries 30

4.2 Algorithm for MDF detection 32

4.3 A working example : s27 37

4.4 Observations on our proposed method 39

5 Implementation Details 40

5.1 Description of Important Classes 40

5.2 Summary of Important Methods 42

5.3 How to use Hydra - our MultiDroopTPG 43

6 Conclusion 44

6.1 Scope for Future Works 44

List of Figures

1.1 A typical 4 layer power supply grid with V_{DD} rails (shaded) and ground rails (white). Vias are the interconnect between two rails in consecutive layers. : courtesy [5] 2

1.2 Circuit under test (CUT) connected to a voltage regulator model, including capacitance C and the interconnect's parasitic inductance L. : courtesy [5] . . . 3

1.3 Voltage on the circuit under test (CUT) after a dI/dt event, without capacitor (dashed curve) and with capacitor (solid curve). : courtesy [5] 3

1.4 A fast current spike *(a)* injected into a power grid can excite localized inductive effects *(b)*, which quickly dissipates in time and space and becomes RC-only effects. (V_{cc} : nominal supply voltage). : courtesy [3] 5

1.5 Transient current response of the nonuniform decaps. Initially, there are various frequencies greater than the global resonant frequency, but eventually all currents respond at the main resonant global frequency. : courtesy [3] 6

1.6 Locality of mid frequency (a) and low frequency (b) currents present in the first and second dips, respectively of the current amplitude curves in Fig. 1.5 : courtesy [3]. 7

2.1 A circuit and its associated DAG for SAT Solver: courtesy [1] 12

2.2 *(a)* A fault-free circuit *(b)* XOR of the fault-free crcuit and the faulty circuit with D *s-a-1* 13

3.1 Properties of a test sequence corresponding to an $M_Aggression$. Figure shows the flow of time in which the test sequence should be applied. 17

3.2 The ISCAS85 benchmark circuit c17 23
3.3 Comparison between MDFTG and Hydra for Experiment # 1 27
3.4 Comparison between MDFTG and Hydra for Experiment # 2 28

4.1 The Huffman model for sequential circuits 31
4.2 Time-frame expansion of a Sequential Circuit 32
4.3 The ISCAS89 benchmark circuit s27 37

Chapter 1

Introduction

Contents

1.1 Power Supply Droop **2**

1.1.1 Low frequency Power Droop (LFPD) 3

1.1.2 Mid frequency Power Droop (MFPD) 4

1.1.3 High frequency Power Droop (HFPD) 7

1.2 Previous Works **8**

1.3 Scope of this thesis **9**

1.4 Organization of this thesis **9**

The scale of integration in *very large scale integration (VLSI)* chips is soaring due to the progress in technology. Today more than a billion transistors can be fabricated on a VLSI chip with operating frequency nearly 10 GHz (and even this scale is increasing day by day) thereby the power consumption per unit area and also per unit length of power supply lines is increasing phenomenally. The increase in power consumption density in VLSI chips along with decreasing threshold and power supply voltages has resulted in greater power supply noise and therefore different types of faults in the operation of these chips , one of them being the Droop Fault. Droop faults originate due to power supply noise termed as *power supply droop* or simply *droop*. This dissertation report has addressed a more specific kind of droop called *mid frequency power droop (MFPD)* and *multi cycle droop fault (MDF)* which originates from MFPD.

1.1 Power Supply Droop

Today's VLSI chips, having very large number of components and high frequency of operation, tend to draw considerable amount of power during their operation. Therefore large transients in supply current can happen while the chip is in operation. These transients specially happen when there is a sudden activity in the chip. For example when chip is going from idle mode to active mode. This flow of large transient current causes a temporal and spatial voltage variation in the power supply grid (Fig. 1.1). Such a variation in voltage at some particular location of power supply grid is termed as *power supply droop* or simply *droop*.

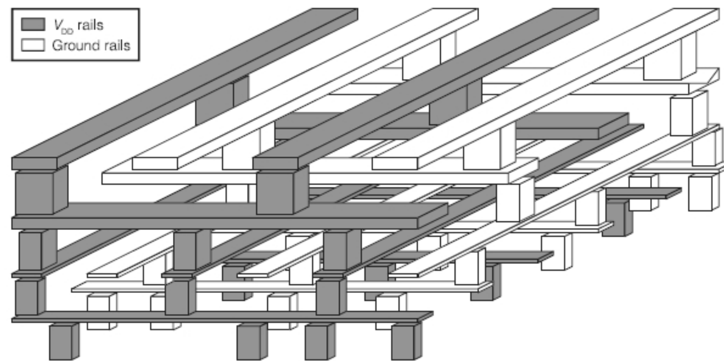

Figure 1.1: A typical 4 layer power supply grid with V_{DD} rails (shaded) and ground rails (white). Vias are the interconnect between two rails in consecutive layers. : courtesy [5]

Droop is broadly classified into three types according to their locality in both space and time.

1. *Low Frequency Power Droop (LFPD)*: This affects the whole grid after a few clock cycles.
2. *Mid Frequency Power Droop (MFPD)*: Localized to a small portion of power supply grid and effective for a few clock cycles.
3. *High Frequency Power Droop (HFPD)*: Highly localized and effective in the same cycle in which it is originated.

These types are discussed below.

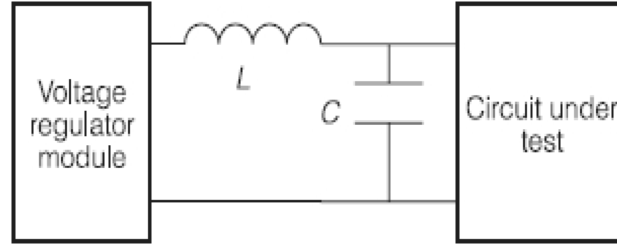

Figure 1.2: Circuit under test (CUT) connected to a voltage regulator model, including capacitance C and the interconnect's parasitic inductance L. : courtesy [5]

1.1.1 Low frequency Power Droop (LFPD)

A schematic of power supply to the VLSI chips has been shown in Fig. 1.2. In deep sub-micron regime large currents from *voltage regulator module (VRM)* flow through the power supply grid of chips delivering current to the transistors which are connected to the grid through vias. Fig. 1.1 shows a typical 4 layer power supply grid in these chips with vias at the cross points of lines in two consecutive layers.

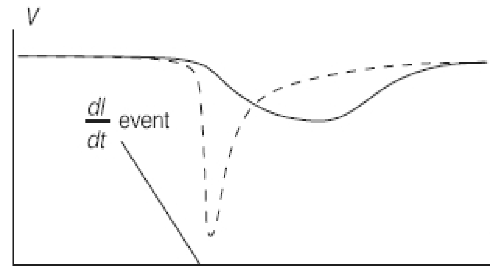

Figure 1.3: Voltage on the circuit under test (CUT) after a dI/dt event, without capacitor (dashed curve) and with capacitor (solid curve). : courtesy [5]

Fig. 1.2 illustrates the cause of LFPD. The circuit under test (CUT) is connected to the VRM. L denotes the parasitic inductance of the interconnect. Let the CUT demands a high current due to sudden increase in activity (as discussed before). Let us call this a dI/dt event. dI/dt event denotes an increased demand of dI amount of current in dt amount of time. Thus after a dI/dt event the power supply voltage V_{DD} to CUT will decrease by an amount $L(dI/dt)$. This amount of decrease in voltage is very high even for moderate values of current demanded ($\approx 10^2 A$ with $V_{DD} \approx 1V$) in a few clock cycles (1-10) in a high frequency ($\approx GHz$) range of

operation of a circuit with the value of L well below 1nH.

The effect of such a dI/dt event on voltage supplied by VRM is shown in Fig. 1.3. Dashed curve shows the voltage difference between the connecting points of VRM and CUT in absence of the capacitor C. We can see a sharp decrease in the voltage when dI/dt event happens. This takes some time for VRM to recuperate and come to the original voltage level. The solid curve shows the effect on the voltage in presence of C. This is the real situation. The capacitor is charged during the power on of VRM. When a dI/dt event happens, the fully charged capacitor C makes for the deficiency of current and in the process discharges itself. After several clock periods there is a situation when capacitor is discharged but VRM is not yet ready to deliver the full supply. This situation is shown as a dip albeit smaller in magnitude on the solid curve. We can see that this dip occurs after several clock cycles from the dI/dt event. This reduction in V_{DD}, which is felt by entire device (CUT), can slow the switching time of the gates in CUT. Such a transient reduction in power supply is known as LFPD.

When LFPD occurring in a circuit is sufficiently large in magnitude then it may be the case that certain gates have large enough delay causing them to fail to switch their state properly within one clock period. This gets revealed as a *stuck-at-fault (STF)*. Such a fault is known as *low frequency droop fault (LDF)*.

1.1.2 Mid frequency Power Droop (MFPD)

In contrast to LFPD, which affects V_{DD} over the entire package, MFPD is a localized phenomenon. MFPD affects only a small area of the chip and its effect last a few clock cycles only. We again see Fig. 1.1 in which a 4 layer power supply grid is shown with vias. Vias are junctions which deliver power to gates fabricated on the chip. In deep sub-micron regime typically 5-6 gates are connected to each via. We concentrate on a small segment of a power supply line typically containing 1-3 vias. When gates drawing power from this segment (i.e connected to the vias present in that segment) switch simultaneously in a considerable number within a clock period, a sudden increase in demand of power happens. This situation produces a negative spike in the supply current at that segment. Deficiency in current produced at that segment quickly tries to dissipate in time drawing current from the nearby region of the

power supply grid. However in the today's VLSI chips where the transistor density is very high along with the high frequency of operation the on-die inductive effects get excited with a quick change in the current flowing through some segment of power supply grid as in this case of a sudden decrease in current supply. On-die inductive effects are comparatively smaller and highly localized in the deep sub-micron regime and decay very quickly. Such transient inductive effects along with a background of RC (with resistance R and decoupling capacitance (decap) C) behavior shown by the chip plays an important role in the decay process of the current spike.

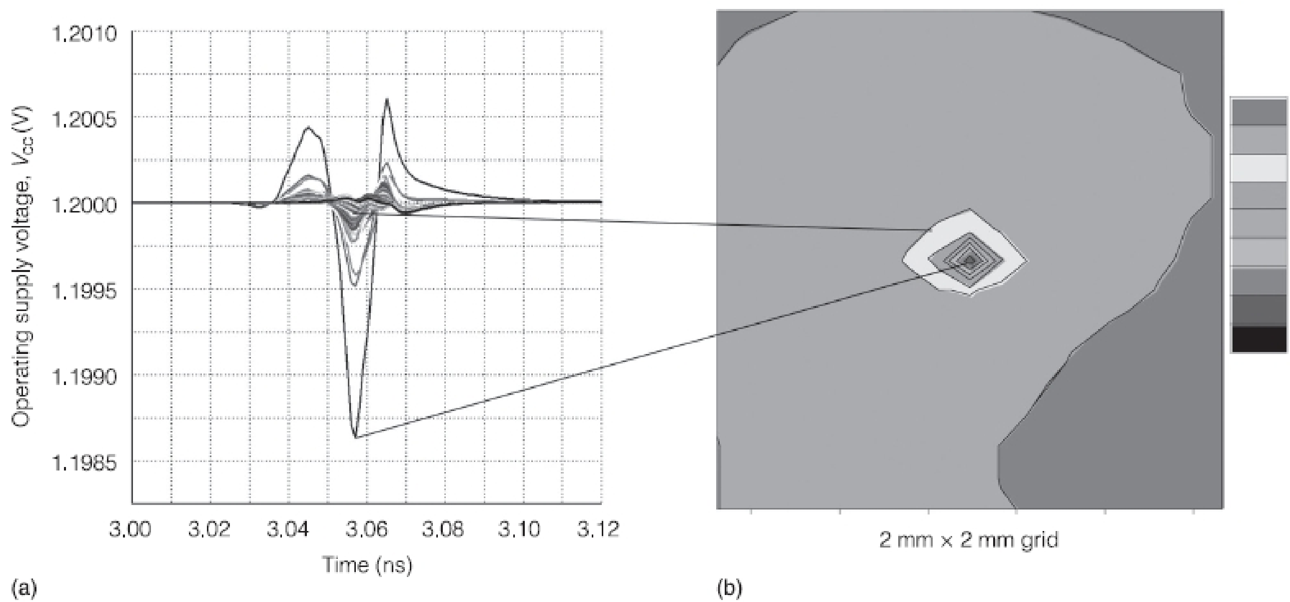

Figure 1.4: A fast current spike *(a)* injected into a power grid can excite localized inductive effects *(b)*, which quickly dissipates in time and space and becomes RC-only effects. (V_{cc} : nominal supply voltage). : courtesy [3]

Fig. 1.4 shows dissipation and the locality of such a current spike injected into a power grid segment. We can see an exponentially decreasing wave like phenomenon while the deficiency in voltage dissipates. This dissipation of deficiency takes a few clock cycles and also travels outwards from the point at which the spike was introduced. This is because the current deficiency is being quickly filled by the nearby power segments.

Transient inductive effects as discussed above are added by the action of decaps present in the circuit. Decaps act both globally and locally. Globally they correspond to the main reso-

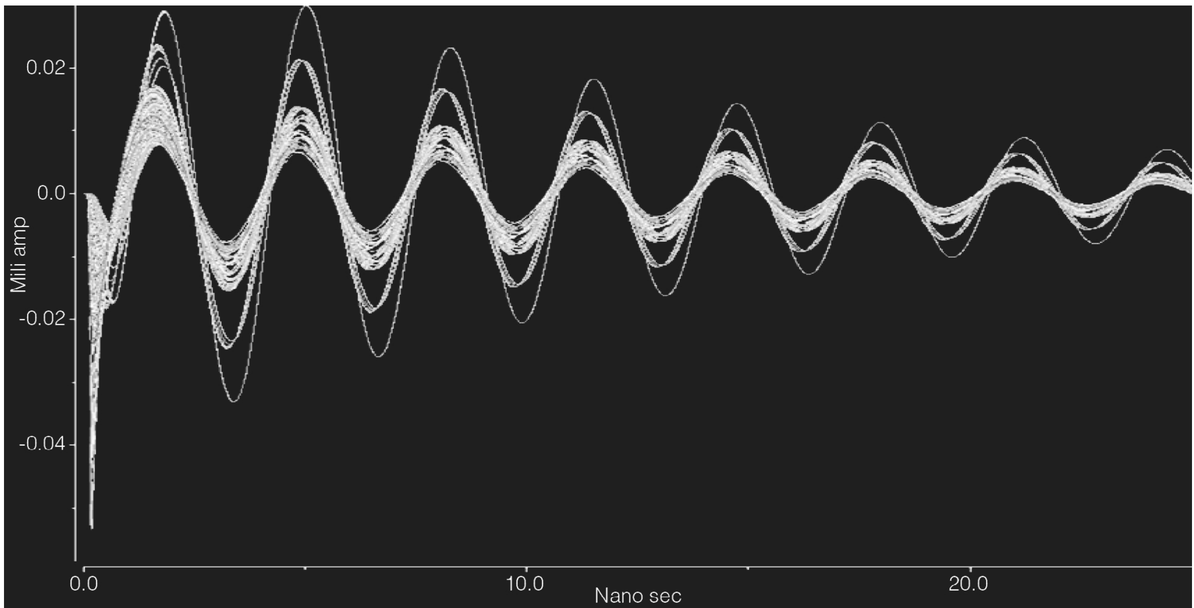

Figure 1.5: Transient current response of the nonuniform decaps. Initially, there are various frequencies greater than the global resonant frequency, but eventually all currents respond at the main resonant global frequency. : courtesy [3]

nance frequency of excitation currents while at higher frequencies they produce more localized effects. Fig. 1.5 shows the effect of such decaps. When a current spike (high frequency) is introduced the local effects of decaps are more effective. This effect along with the on-die inductive effect tries to slow down the recovery of current supply to the global resonant frequency current for a few clock cycle. Thus a deficient current supply can be felt at the same segment even after some clock cycles.

Fig. 1.6 shows the locality of mid frequency and low frequency currents on the chip when a current spike is introduced to a power segment. This shows that the deficiency of current supply is also felt by the nearby power segments though less in magnitude.

The above discussion shows that the droop produced by simultaneous switching of gates in physical proximity travels along the power grid and can be felt at some nearby positions after a few clock cycles but with less magnitude. Also the droop is felt at the same position for a few clock cycles. Such a droop is termed as *mid frequency power droop (MFPD))*. Again, as in the case of LFPD, MFPD also slows down the switching of the gates drawing power from the via where it is present. If the delay in switching is large enough for a clock period, which

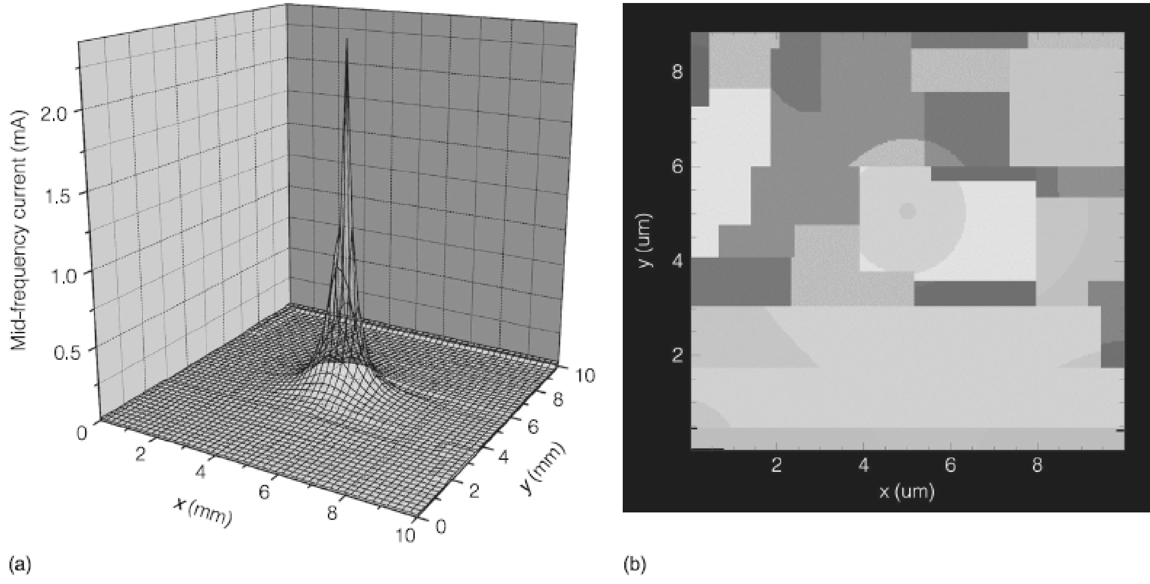

Figure 1.6: Locality of mid frequency (a) and low frequency (b) currents present in the first and second dips, respectively of the current amplitude curves in Fig. 1.5 : courtesy [3].

depends on the magnitude of the MFPD present at that via, then the gates may fail to switch within that clock period. This results in the faulty operation of the circuit. This is known as *mid frequency droop fault* or *multi cycle droop fault (MDF)*. It can be noted that LDF is also a multi cycle droop fault but for our purpose we denote *mid frequency droop fault* as *multi cycle droop fault (MDF)*.

Apart from this behavior there is also another difference between LFPD and MFPD. LFPD has highest magnitude after some time from the time of sudden increase in current demand (dI/dt event) i.e it increases and then decreases in time after the event as shown in Fig. 1.3. While in case of MFPD the highest magnitude of droop happens just when the sudden increase in demand of current happens. It decreases both spatially and temporally and eventually disappears. This is also one of the reasons why we termed *mid frequency droop fault* as *multi cycle droop fault (MDF)*.

1.1.3 High frequency Power Droop (HFPD)

HFPD can be considered as a special case of MFPD. The cause of HFPD is same as MFPD. This classification is done only to capture the effect of droop and fault caused by it within

only one clock cycle and at the same via where the sudden activity happens.The magnitude of MFPD within the first clock cycle and at the same location where it originates is highest. Therefore in VLSI chips with moderate transistor density often MFPD is of only one clock cycle length affecting the same locality where it originates. Thus notion of HFPD helps in formulating the faults due to MFPD within the same clock cycle.

Fig. 1.4 also depicts how a current spike injected in a circuit typically dissipates in space and time. If the injected current spike consists of only very high frequency components then it dissipates very quickly (within one clock period) as depicted in Fig. 1.5. In this figure we can see that the high frequency components present in the first dip quickly dissipate within a few nano seconds. For VLSI chips with operating frequency of the order of GHz dissipation often happens within one clock cycle. Also such a droop is highly localized. Such droop is termed as *High frequency Power Droop (HFPD)*.

The faulty operation of the circuit occurring due to HFPD is termed *high frequency droop fault (HDF)* or *single cycle droop fault (SDF)*. The reason for the origin of the fault is same as that of fault due to MFPD or LFPD.

1.2 Previous Works

This field is still nascent, and this is largely due to the fact that droop faults are being seen in chips with very high density of transistors with a very high frequency of operation. Advances in fabrication techniques promise production of such chips in large scale in the near future. But still we are on the verge of entering the domain where these chips will come into large scale use. Present day chips are experiencing a little effect due to droop.

The problem of *single cycle droop fault (SDF)* in combinational circuits was addressed in the work of Mitra et. al. in [4]. There SDF is modeled as *slow to rise (STR)* fault taking into account the switching of gates 0→1. The other way transitions have a similar modeling. The SDF gets revealed as *stuck-at-0 (s-a-0)* fault at the victim gates. A test vector pair is found which excites the HFPD on some victim gates and then propagates the effect of *s-a-0* fault on every victim gate at some primary output.

SDF in full scan circuits also has been covered in [4] by Mitra et.al.. A similar modeling

for full scan circuits as with combinatorial circuits has been done for it. A methodology for test pattern generation in *launch off capture (LOC)* environment in full scan circuits has been proposed and an *automatic test pattern generator (ATPG)* has been implemented for it.

While in [5] Ilia Polian et. al. have addressed the faults due to LFPD and HFPD both but not MFPD per se. They propose a method for ATPG for exciting the worst case droop by combining the effects of LFPD and HFPD both and detecting these faults.

The model for Mid-Frequency Power Droops in combinational chips was formulated by [2] and solved using conventional ATPG based approach.

1.3 Scope of this thesis

This thesis concentrates on MFPD and presents a model for it in terms of Boolean Satisfiability, for exciting and detecting the MDF. Using a public SAT solver, an efficient and robust ATPG is created which generates MDF test vectors. A prototype algorithm has been presented for detecting MDF in full-scan circuits .

1.4 Organization of this thesis

Chapter 2 builds the theoretical ground to understand MDF and the basic concepts of Boolean Satisfiability. This chapter summarizes the cumulative model for MDF and how to convert a circuit to its equivalent CNF , upon which additional constraints be imposed in terms of clauses to solve for various desired features of the circuit. Chapter 3 presents the algorithm to formulate clauses to model MFPD . The requirements of a test sequence to detect MDF has been presented , and it has been demonstrated how to formulate clauses such that a successful assignment represents a MDF test vector. Chapter 3 also presents the experiments done using an implementation of the algorithm (Hydra) and a comparison between the ATPG based approach and the SAT Solver based approach. Chapter 4 contains a prototype algorithm to detect MDF vectors for full-scan circuits , explained with an example. Chapter 5 contains the implementation details of the algorithm. Chapter 6 wraps up the thesis by concluding remarks and discusses scope for future work.

Chapter 2

Basic Concepts

Contents

2.1 Multi-Droop Fault Model **10**

2.2 An Overview of the Boolean Satisfiability Method **11**

2.2.1 Formula Extraction 12

2.2.2 Formula for Fault Detection 13

2.1 Multi-Droop Fault Model

The Cumulative Model for Mid Frequency Power Droop was developed in [2] . We present a brief summary of the model and its associated definitions for our understanding.

Target Via T: A via , on which effect of MFPD is to be observed.

Aggression : An event when a group of gates switch simultaneously $0 \rightarrow 1$, causing power droop to travel along the power grid in all directions ever decreasing in amplitude.

$\alpha_i(T)$**:** A gate in the vicinity of T which causes a droop at via T after i clock cycles , called an *aggressor*.

D **:** The least amount of droop at the target via T , that will cause switching delay in gates connected to T.

$\delta_i(g,T)$**:** The amount of droop causes at T by an aggressor $\alpha_i(T)$, the latter being g.

$\xi_i(g,T)$**:** It is the amount of droop causes at T by an aggressor $\alpha_i(T)$, divided by D. It is known as the *effectiveness* of aggressor $\alpha_i(T)$.

$\lambda_i(T)$**:** It is a set of gates , all of which cause a droop at T after i clock cycles. Such a list of gates is known as an *aggressor list*.

$\xi(\lambda_i(T))$**:** It is the worst case droop caused by simultaneous transition of all gates belonging to $\lambda_i(T)$ i.e. it is the sum of individual effectiveness $\xi_i(g,T)$ for all gates g belonging to $\lambda_i(T)$.

$M_Ag(T)$**:** An *M_Aggression* for a target via T consists of *aggressor lists* which have switched during any of the previous M clock cycles. Thus , it is defined as
$M_Ag(T) = \{\lambda_i(T) \mid 0 \leq i \leq M \text{ and } \lambda_M(T) \text{ is not empty}\}$

$\xi(M_Ag(T))$**:** It is the sum of effectiveness of all the aggressor lists belonging to the *M_Aggression* $M_Ag(T)$ i.e. it is the worst case droop at the target via T produced by the aggression(excitation) of each aggressor list $\lambda_i(T)$ belonging to it in their respective previous clock cycle.

From the above model , we can deduce the following :
If an M_Aggression for target via T has a total effectiveness greater than or equal to 1 , only then it is a likely candidate to produce an MDF at any victim gate of T

2.2 An Overview of the Boolean Satisfiability Method

The Boolean Satisfiability is a method for modeling faults in combinational circuits , which is neither a purely structural method (like D-algorithm , PODEM , FAN) nor an algebraic one (e.g. the Boolean Difference method). It generates a formula in *Conjunctive Normal Form (CNF)* from the given circuit that defines the set of test patterns that detect the fault and then use a Boolean Satisfiability algorithm to find a satisfiable assignment to the formula. We first show how to extract the formula for detecting single stuck-at fault in combinational circuits ,

which will be extended later to determining the formula for MDF faults in both combinational and full-scan circuits.

2.2.1 Formula Extraction

The topological description of the circuit is represented using a *Directed Acyclic Graph (DAG)* . Circuit inputs , outputs , gates and fan-out points constitute the vertices , while the edges are circuit lines(wires) . Each circuit output is a source , and each circuit input is a sink . An example of a circuit and its corresponding DAG is shown in 2.1.

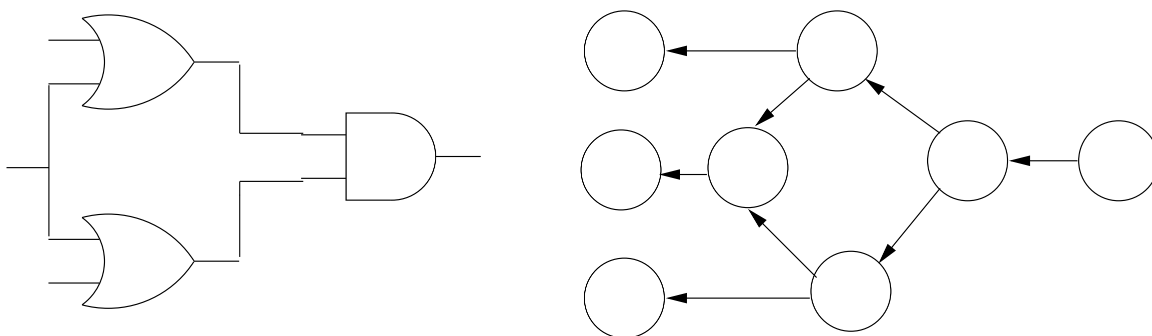

Figure 2.1: A circuit and its associated DAG for SAT Solver: courtesy [1]

For each edge , there is an unique variable assigned to it. For each node , there is a corresponding *formula* , in terms of the variables for its incoming and outgoing edges , which expresses the functionality of that node(gate or fan-out) in the circuit. E.g. an inverter with input X and output Y will have the formula $Y = \overline{X}$. Similarly , an OR gate with inputs X and Y and output Z will be represented by $Z = X + Y$.

In the next step , these formula would be converted into their respective CNF forms , as most SAT Solvers accept the input as a CNF expression . Take , for example , a 2-input OR gate. the formula assigned to it is

$$Z = X + Y$$

It is equivalent to

$$(Z \rightarrow (X + Y)).((X + Y) \rightarrow Z)$$

Using the fact that $P \rightarrow Q$ is equivalent to $\overline{P} + Q$, we convert all implications into disjunctions

Logic Gate	*Boolean Expression*	*CNF Formulae*
AND	$Z = X.Y$	$(\overline{Z} + X).(\overline{Z} + Y).(Z + \overline{X} + \overline{Y})$
OR	$Z = X + Y$	$(Z + \overline{X}).(Z + \overline{Y}).(\overline{Z} + X + Y)$
NOT	$Y = \overline{X}$	$(\overline{X} + Y).(X + \overline{Y})$
NOR	$Z = \overline{X + Y}$	$(\overline{Z} + \overline{X}).(\overline{Z} + \overline{Y}).(Z + X + Y)$
NAND	$Z = \overline{X.Y}$	$(Z + X).(Z + Y).(\overline{Z} + \overline{X} + \overline{Y})$

Table 2.1: Formulae for basic gates

to get

$$(\overline{Z} + X + Y).(Z + \overline{X}).(Z + \overline{Y})$$

The above CNF expression is true iff the values of X , Y and Z are in accordance to the truth-table of an OR gate. CNF expressions for all basic 2-input gates are given in table 2.1. Note that the formulae for gates having more than 2 inputs is produced in the same manner .

2.2.2 Formula for Fault Detection

Once we have extracted the formula for each of the nodes , we can obtain the formula for the entire circuit simply by concatenating the CNFs for all the nodes . As the formula for each node is independently satisfiable , the concatenated formula must also be satisfiable if the variables for the input nodes and output nodes match any entry of the circuit's truth table.

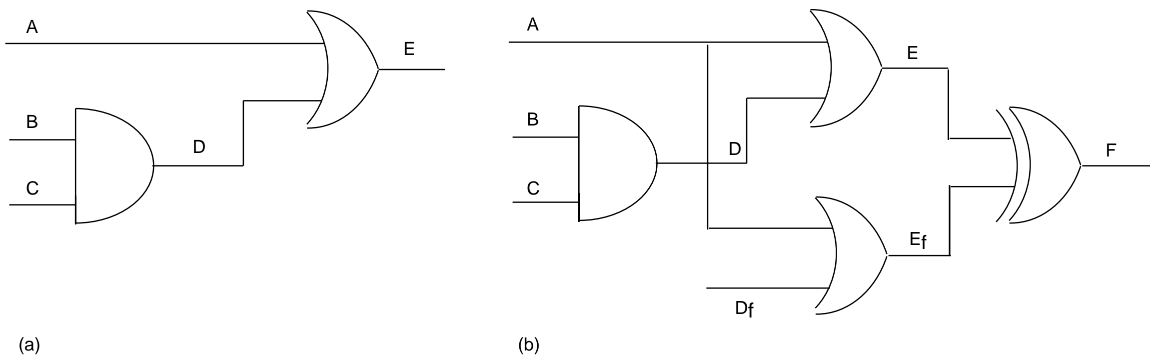

Figure 2.2: *(a)* A fault-free circuit *(b)* XOR of the fault-free crcuit and the faulty circuit with *D s-a-1*

The formula for the circuit in figure 2.2(a) is as follows:

$$(B+\overline{D}).(C+\overline{D}).(\overline{B}+\overline{C}+D)(\overline{A}+E).(\overline{D}+E).(A+D+\overline{E})$$

The first three clauses are due to the AND gate and the next three are due to the OR gate. Suppose , now we want to find a test vector (TV) that would detect a s-a-1 fault in D. This can be done by making a copy of the circuit , renaming the the variables associated with wires that lie on a path between the fault site and a circuit output , and inserting new nodes in the DAG to denote the implied discontinuity caused by the stuck-at fault. This is to represent the fact that in a faulty circuit , one value is generated at the fault site but another value is propagated on to the rest of the circuit. Unary clauses are tagged to the added nodes to model the above behavior i.e. we add the clause D_f to the formula for the faulty circuit as the wire D is stuck-at 1 . The formula for the faulty circuit is derived the same way by walking the DAG starting from the faulted output , and taking conjunction of the formulae of all encountered nodes . The formula for the faulted circuit in figure 2.2 is

$$(\overline{A}+E_f).(\overline{D_f}+E_f).(A+D_f+\overline{E_f})(D_f)$$

Any input combination that causes the faulted output to differ from the unfaulted output , constitutes a test for the given fault . In order to find such a test , we take the conjunction of the two extracted formula and add a clause depicting the XOR of the corresponding circuit outputs of the faulted and unfaulted circuit. Thus the clauses in the formula for when solved gives a test for D *s-a-1* is :

$(B+\overline{D}).(C+\overline{D}).(\overline{B}+\overline{C}+D)$ *for the AND gate,*
$(\overline{A}+E).(\overline{D}+E).(A+D+\overline{E})$ *for the OR gate,*
$(\overline{A}+E_f).(\overline{D_f}+E_f).(A+D_f+\overline{E_f})(D_f)$ *for the OR gate in the faulted circuit,*
$(E+\overline{E_f}+F).(\overline{E}+E_f+F)$ *and*
$(E+E_f+\overline{F}).(\overline{E}+\overline{E_f}+\overline{F}).(F)$ *for the XOR operation.*

In the next chapter, this model has been elaborated upon to formally define multi-cycle droop fault and the problem of detecting such faults.

Chapter 3

MDF Detection in Combinational Circuits Using Boolean Satisfiability

Contents

3.1 Requirements of an ATPG for MDF 15

3.2 Data Structures 16

3.3 Algorithm for Test Generation for detecting MDF 18

3.4 A working example: c17 23

3.5 Experimental Results 25

3.5.1 Experiment #1 (Logical Neighbors method) 26

3.5.2 Experiment #2 (Random Selection method) 27

3.5.3 Comparison of SAT Solver based approach with Conventional ATPG . 28

3.1 Requirements of an ATPG for MDF

In order to detect MDF in the target via T , we need to generate vectors that would produce excitation in the preceding cycles (corresponding to the given $M_Aggression$) so that the worst case droop is produced on via T. The MDF thus generated is then to be propagated to at least one Primary Output for fault detection. For a target via T along with its victim list

V, an *M_Aggression* $M_Ag(T)$ with a suitable value of M, a sequence of $M + 2$ test vectors (TV) is required to detect an MDF at T. Fig. 3.1 shows the properties of such a test sequence. Such a sequence should have the following properties:

Mid Frequency Power Droop (MFPD) Excitation : The gates belonging to the aggressor list must switch at the appropriate time to generate a cumulative power droop at the target via. Specifically , the gates in the list $\lambda_i(T) \in M_Ag(T)$, for each $0 \leq i \leq M$, undergoes a 0→1 transition within clock period $t - i$.

Multi Cycle Droop Fault (MDF) excitation : For producing MDF , we need to make the gates in victim list V undergo a transition at the instant the cumulative power droop takes place in the target via. So, the last vector in TV must ensure that the gates in victim list V are set to '0' within clock period $t - 1$ and are set to '1' in clock period t. If the droop produced in T is sufficient , then the gates in victim list will fail to switch and thus MDF will occur.

MDF propagation: The test vector set TV also needs to ensure that if MDF is indeed produced in the target via , then it must also be propagated to a circuit output for the fault to be observed . Therefore , at clock period t , the test vector applied at the inputs must set all gates in victim list to '1' (for MDF excitation) and simultaneously is a test vector for detecting multiple *s-a-0* fault in all gates in victim list V.

We can sum up all the above properties of a test sequence as follows:

An MDF generated by the aggression of a given M_Aggression ($M_Ag(T)$) is said to be testable (excitable) if and only if there exists a test sequence as above corresponding to $M_Ag(T)$.

3.2 Data Structures

For representing the problem in Boolean Satisfiability domain , we use the following ADT's :

- *Signal:* Every connecting line in the circuit is represented by a *Signal* data type. Each *Signal* type has 2 unique variables associated with it , one representing its assignment in

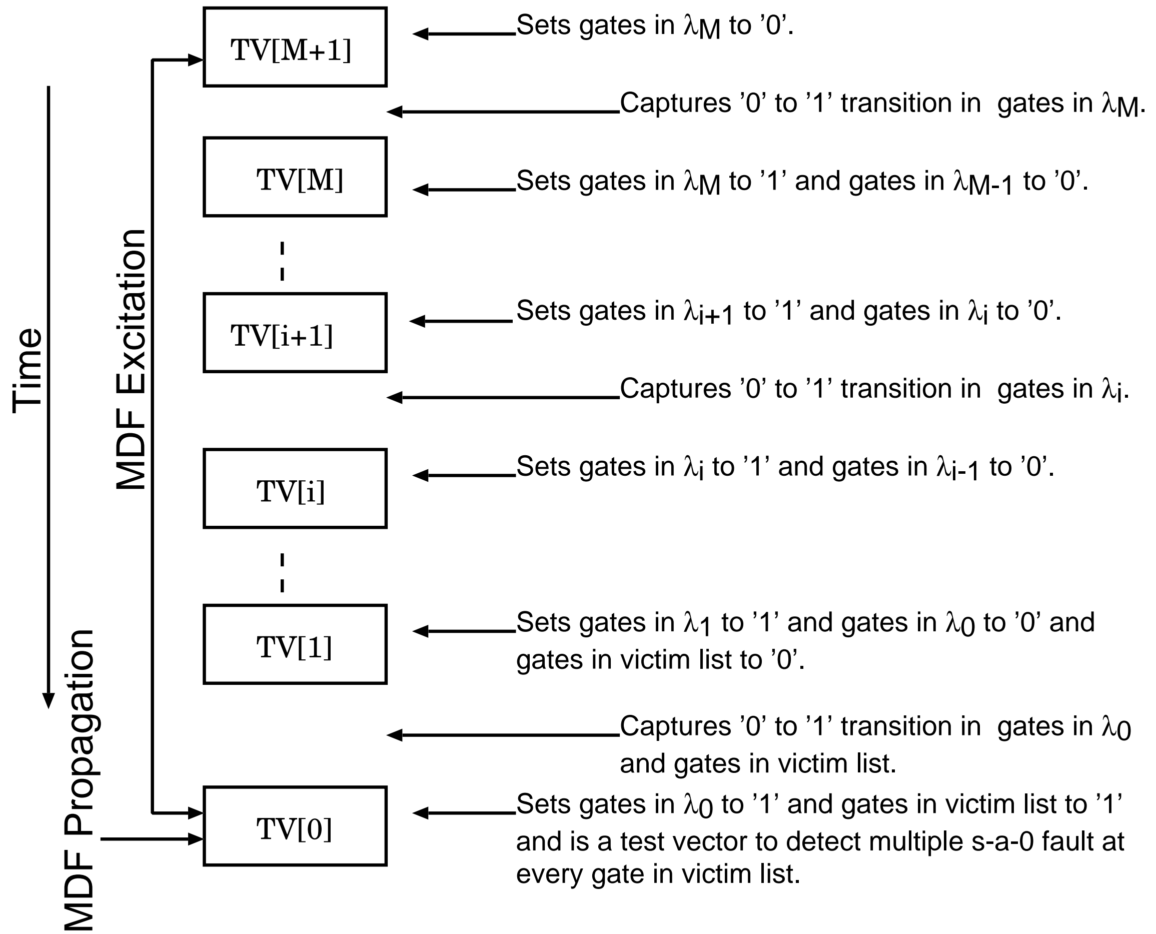

Figure 3.1: Properties of a test sequence corresponding to an *M_Aggression*. Figure shows the flow of time in which the test sequence should be applied.

the fault-free circuit and the other , its assignment in a faulty circuit. Given a *Signal* , we can determine the following in constant time:

1. Whether it a Primary Input (PI) or Primary Output (PO)

2. Its parent gate (if any) and child gates (if any)

- *LogicGate:* Every gate in the circuit is modeled as of type *LogicGate* . Given a *LogicGate* , we can determine the following in constant time :

  1. Its FanIn *Signal*s

  2. Its FanOut *Signal*

3. *normal_clauses* , the CNF clauses representing the functionality of the gate in the normal circuit , computed using the variables associated with the fault-free circuit for each FanIn *Signal* and the FanOut *Signal*.

4. *faulty_clauses* , the CNF clauses representing the functionality of the gate when it is on a fault path , computed using the variables associated with the faulty circuit for each FanIn *Signal* and the FanOut *Signal*.

- *Circuit:* The circuit ADT stores the list of *LogicGate* objects and *Signal* objects , together making up the entire circuit as given in input to the main algorithm.

3.3 Algorithm for Test Generation for detecting MDF

To generate a test sequence for a target via T, first make some tentative $M_Aggressions$ for T where M is the threshold clock cycles and using vias only nearby distance from T, also make a victim list V. These lists are made using external data coming from the layout information of the chip using static timing analysis and RLC network simulation of the chip [4]. Before we can move on to the main algorithm , we need to define certain procedures which are used by it.

Algorithm $get_Multiple_Fault_Detection_Clauses(S, v)$

Input: A set of gates S , and a logic value v which is either '0' or '1'

Output: CNF Clauses whose satisfiable assignment gives a test vector for multiple $s-a-v$ fault in S

Begin:

$fault_path_gates = \phi$

$fault_cone_primary_outputs = \phi$

$fault_detection_clauses = \phi$

$fault_activation_clause = \phi$

for Each gate g in S **do**

Get gates on the fault cone for a $s-a-v$ fault at output of g and add them to $fault_path_gates$.

Add Primary Outputs which are on the fault cone to $fault_cone_primary_outputs$.

Get X , the faulty variable associated with the FanOut of g and add the fault activation clause as follows:

if v is 1 **then**

Add the literal X to the clause $fault_activation_clause$

else

Add the literal $\overline{X}$ to the clause $fault_activation_clause$

end if

end for

Add $fault_activation_clause$ to $fault_detection_clauses$.

for Each gate g in $fault_path_gates$ **do**

Add the $faulty_clauses$ for g to $fault_detection_clauses$

end for

for Each PO $Signal$ in $fault_cone_primary_outputs$ **do**

Add the CNF Clauses representing the condition that the value assigned to at least 1 PO $Signal$ in the faulty circuit to differ from its corresponding value in the normal circuit.

end for

return $fault_detection_clauses$

End

Algorithm $get_Maximal_Excitable_Subset(S)$

Input: A set of gates

Output: A subset of S , such that they can be simultaneously set to '0' by at least 1 vector and be simultaneously set to '1' by at least 1 (other) vector

Begin:

$W \leftarrow$ the set of subsets of S , arranged in decreasing order of cardinality

while W is not empty **do**

Pick the first set w from W

if All gates in w can be simultaneously excited to '0' **then**

if All gates in w can be simultaneously excited to '1' **then**

return w

end if

else

Delete w from W

end if

end while

return Null set , signifying no such subset exists

End

Algorithm *MDF_Combinational_TPG* **(**C **,** V **,** M_Ag **)**

Input : A circuit C, a victim list V, an *M_Aggression* $M_Ag(T)$.

Output : If successful then a test sequence TV and number of test vectors in it else failure.

Begin

1. Initialize the 3 counters hope H , effectiveness covered E , effectiveness at stake S, which are used to keep track of the algorithm's progress , as follows :

$$E = 0, H = \sum_{\lambda_i(T) \in M_Ag(T)} \xi(\lambda_i(T)), S = 0$$

   If $H < 1$, terminate the algorithm and output failure.

2. Generate TV[0] as follows :

   - Compute C_1^0 , the CNF clauses for detecting multiple $s-a-0$ fault at gates in victim list , by calling procedure $get_Multiple_Fault_Detection_Clauses(V, 0)$.
   - Identify the maximal simultaneously excitable subset in $\lambda_0(T)$ by calling procedure $get_Maximal_Excitable_Subset(\lambda_0(T))$. Refine the given *M_Aggression* by replacing $\lambda_0(T)$ by this subset .Compute C_2^0 , the CNF clauses which ensure that all the gates in this subset are set to '1' .
   - Ideally , TV[0] should both set the gates in $\lambda_0(T)$ to '1' and also be a test vector for detecting multiple $s-a-0$ fault in the victim list V. Therefore , if we have an

assignment to the primary inputs which satisfy the conjunction of C_1^0 and C_2^0 , we have found TV[0] . If such an assignment is not possible , solve C_1^0 to get TV[0] , which just detects multiple $s-a-0$ fault. If C_1^0 is also unsatisfiable , then the MDF fault is redundant and the algorithm terminates.

Update the counters as follows :

$$H = H - \xi(\lambda_0(T))$$

If TV[0] satisfies both C_1^0 and C_2^0 , then

$$S = \xi(\lambda_0(T))$$

3. Generate TV[1] as follows :

   - Compute C_1^1 , the CNF clauses which set the gates in $\lambda_0(T)$ to '0'.
   - Refine the $M_{Aggression}$ by identifying the maximally excitable subset in $\lambda_1(T)$. Compute C_2^1 , the CNF clauses which set the gates in the refined $\lambda_1(T)$ to '1' .
   - Compute C_3^1 , the CNF clauses which sets the gates in V to '0' .
   - If the conjunction of all the above 3 sets of clauses have a satisfiable assignment , then we have found TV[1] . If not , let $C_4^1 = C_1^1.C_3^1$ and find an assignment for C_4^1 to get TV[1]. If this does not have a satisfiable assignment , then solve $C_5^1 = C_2^1.C_3^1$ to get TV[1] . If none of the above 3 combinations of clauses have a satisfiable assignment , then stop and output failure .

$$H = H - \xi(\lambda_1(T))$$

If TV[1] sets the gates in refined $\lambda_0(T)$ and $S > 0$, implying that TV[0] sets gates in $\lambda_0(T)$ to '1' and the transition of $\lambda_0(T)$ is captured , then

$$E = E + \xi(\lambda_0(T))$$

If $E \geq 1$, then output the test sequence . If TV[1] sets gates in refined $\lambda_1(T)$ to '1' , then

$$S = \xi(\lambda_1(T))$$

If $(E + S + H) < 1$, terminate and output failure.

4. Generate TV[i] , $2 \leq i \leq M$, as follows :

for $2 \leq i \leq M$ **do**

if $\lambda_i(T)$ and $\lambda_{i-1}(T)$ are both empty **then**

TV[i] is a random test vector

else

$C_1^i \leftarrow$ CNF clauses to set gates in λ_{i-1} to 0

$C_2^i \leftarrow$ CNF clauses to set gates in λ_i to 1

if Both C_1^i and C_2^i are simultaneously satisfiable **then**

TV[i] $\leftarrow$ is found

if $S > 0$ **then**

{*This means TV[i-1] sets* $\lambda_{i-1}(T)$ *to 1 and transition of* $\lambda_{i-1}(T)$ *has been captured*}

$E = E + \xi(\lambda_{i-1}(T))$

if $E > 1$ ***then***

Stop and output the test vectors

end if

end if

else

if $\lambda_i(T)$ is not empty AND $S > 0$ **then**

if C_1^i is satisfiable **then**

TV[i] found.

S = 0

else if C_2^i is satisfiable **then**

TV[i] found

$S = \xi(\lambda_i(T))$

else

TV[i] is a random vector

end if

end if

end if

end if

end for

5. (for $TV[M+1]$) If $TV[M]$ sets gates in $\lambda_M(T)$ to '1', then find a test vector which sets $\lambda_M(T)$ to '0'. If this test vector is possible, then

$$E = E + \xi(\lambda_M(T)).$$

Else if test vector above is not possible (empty), then output failure.

End.

3.4 A working example: c17

We take a small example to illustrate the working of the algorithm .

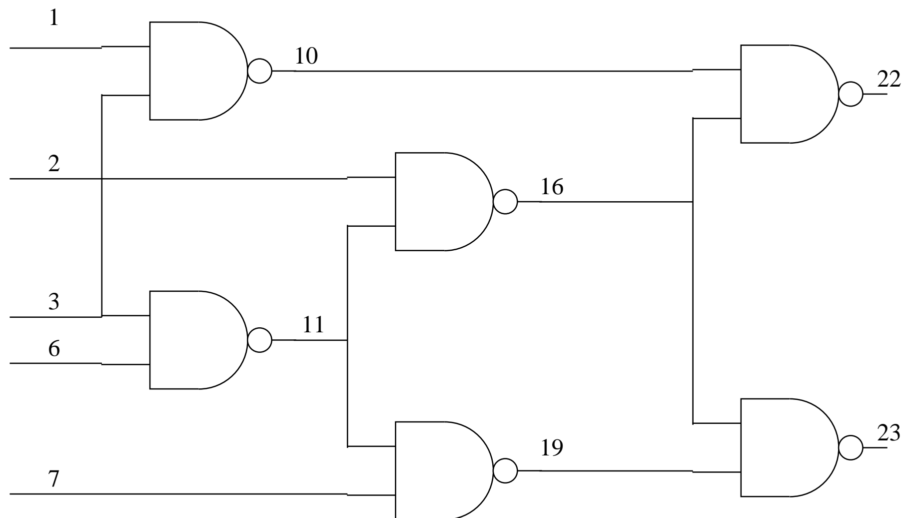

Figure 3.2: The ISCAS85 benchmark circuit c17

The inputs are as follows :

Circuit C **:** c17.bench

Victim list V**:** Gates 16 , 19

$M_Ag(T)$**:** It is as follows :

0 : 4 (16,19,10,11) 1

1 : 2 (22,23) 0.395

2 : 0

3 : 0

The circuit c17.bench is chosen due to its simplicity and ease of verification. Each of the 13 signal lines present in the circuit have 2 unique variables , one for the normal circuit and one for the faulty circuit. For ease of reference , each signal has its corresponding name as its normal circuit variable , and the same name having subscript f for the faulty circuit variable. Thus , signal 23 has variable 23 as its fault-free circuit variable and 23_f as the faulty circuit variable. Primed variables indicate complementation . Also , independent variables (as those required for fault-detection in Primary Outputs) are denoted by alphabets.

- $H = 1.395$, $E = 0$, $S = 0$
- C^{Normal} i.e. the normal circuit CNF clauses which are appended with other CNF clauses when solving :
  $(1+10).(3+10).(10'+1'+3').(3+11).(6+11).(11'+3'+6').(2+16).(11+16).(16'+2'+11').(11+19).(7+19).(11'+7'+19').(10+22).(16+22).(10'+16'+22').(16+23).(19+23).(16'+19'+23')$
- $C_1^0 =$
  $(16'_f+19'_f)(10+22_f).(16_f+22_f).(22'_f+16'_f+10').(16+23_f).(19_f+23_f).(23'_f+19'_f+16')(22'+22_f+X).(22+22'_f+X).(22+22_f+X').(22'+22'_f+X')(23'+23_f+Y).(23+23'_f+Y).(23+23_f+Y').(23'+23'_f+Y')(X+Y)$
- Maximally excitable subset in $\lambda_0(T)$ = 16 , 19 , 10. Clauses to ensure that all gates in this subset are set to '1' : $C_2^0 = (16).(19).(10)$

- Input Vector that satisfies $C_1^0.C_2^0$:

$$TV[0] = 00000$$

The value of the counters are as follows : $H = 0.395$, $S = 0.75$

- C_1^1 .i.e. CNF clauses to set gates in refined $\lambda_0(T)$ to '0' are $(16').(19').(10')$

- C_2^1 i.e. CNF clauses to set gates in refined $\lambda_1(T)$ to '1' are (22).(23)

- C_3^1 i.e. CNF clauses to set gates in victim list V to '0' are $(16').(19')$ The following vector satisfies all of the 3 above criteria , apart from satisfying the CNF formula for the normal circuit :

$$TV[1] = 11101$$

The value of the counters are as follows : $H = 0$, $S = 0.395$, $E = 0.75$

- C_1^2 i.e. CNF Clause to set gates in $\lambda_1(T)$ to '0' are $(22').(23')$

- $C_2^2$is empty

- Input vector that satisfies C_1^2 is

$$TV[2] = 00000$$

The value of the counters are as follows :

$H = 0$, $S = 0$, $E = 1.145$

As $E > 1$, MDF detection test sequence found which are as follows:

$TV[2] = 00000$

$TV[1] = 11101$

$TV[0] = 00000$

3.5 Experimental Results

A working ATPG for combinatorial circuits is implemented in C++ using the algorithm presented above. This ATPG is called Hydra . In absence of actual chip data the vias are made

using the ISCAS85 bench mark circuits. The *M_Aggressions* and victim lists are chosen from those vias. Two different methods have been employed to generate *M_Aggressions*. These are *(a) logical neighbors of seed* and *(b) random selection*, where seed is any gate randomly chosen from the circuit. The generated *M_Aggressions* are then fed to MDFTG along with circuit and victim lists and the response of MDFTG is observed. The experiments are discussed as follows :

3.5.1 Experiment #1 (Logical Neighbors method)

As the name suggests this method makes the *M_Aggression* using the logically connected gates nearest to a seed gate. Thus the formed *M_Aggressions* are strongly logically correlated. In this method we form an *M_Aggressions* using following steps:

$G \leftarrow$ Gate list from the given circuit C

$M \leftarrow$ A random number between 1 to 4

while G is not empty **do**

  $g \leftarrow$ A random gate chosen from G (Seed gate)

  $T' \leftarrow$ Logical neighbors of g in both directions, upto 2 levels

  if T' has more than 4 gates **then**

    $T \leftarrow$ Set of 5-6 gates chosen randomly from T' , which form the gates in the target via.

    $V \leftarrow$ Victim list , having 1-4 gates , formed from T randomly

    $Neighbors \leftarrow$ The set of gates which are neighbors of g starting from the 3rd level.upto $M+2$ levels.

    Form an *M_Aggression* from $Neighbors$ such that the gates further from g belong to $\lambda_i(T)$ with larger value of i

    Remove the gates in T from the list G.

  else

    Remove gates in T' from G

  end if

end while

In this manner a list of $M_Aggressions$ is formed. These $M_Ag(T)'s$ are fed to MDFTG and the number of testable $M_Ag(T)'s$ are recorded as shown in the table below.

Algorithm	Circuit	#M_Ag(T)	# Testable M_Ag(T)	# Redundant M_Ag(T)	# Unexcitable M_Ag(T)	Time taken (s)
MDFTG (Conventional ATPG based)	c432.bench	6	2	2	2	1
	c499.bench	7	2	2	3	1
	c880.bench	27	1	21	5	1
	c1355.bench	48	0	34	14	3
	c1908.bench	74	0	55	19	4
	c3540.bench	111	2	76	33	20
	c5315.bench	184	0	130	54	76
	c6288.bench	210	0	170	40	100
	c7552.bench	269	1	193	75	241
Hydra (SAT Solver based)	c432.bench	11	7	0	4	1
	c499.bench	8	4	0	4	4
	c880.bench	22	7	0	15	1
	c1355.bench	32	2	0	30	4
	c1908.bench	63	5	0	58	7
	c3540.bench	92	15	0	77	14
	c5315.bench	137	20	0	117	29
	c6288.bench	225	36	1	188	160
	c7552.bench	248	50	0	198	105

Figure 3.3: Comparison between MDFTG and Hydra for Experiment # 1

3.5.2 Experiment #2 (Random Selection method)

In this method the $M_Aggressions$ are formed choosing random gates from the circuit. Thus the formed $M_Aggressions$ are uncorrelated. In this method we form an $M_Aggression$ using following steps:

$G \leftarrow$ Gate list from the given circuit C

$M \leftarrow$ A random number between 1 to 4

while G has more than 4 gates **do**

$T \leftarrow$ Set of 5-6 gates chosen randomly from G , which form the gates in the target via.

$V \leftarrow$ Victim list of 1-4 gates chosen randomly from the set T

Remove the gates in T from G

Form the $M_Aggression$ choosing gates randomly from list G

end while

In this manner a list of $M_Aggressions$ is formed and fed to the two algorithms . A summary of the results follow :

Algorithm	Circuit	#M_Ag(T)	# Testable M_Ag(T)	# Redundant M_Ag(T)	# Unexcitable M_Ag(T)	Time taken (s)
MDFTG (Conventional ATPG based)	c432.bench	26	4	4	18	2
	c499.bench	33	3	8	22	2
	c880.bench	63	34	11	18	7
	c1355.bench	90	10	18	62	67
	c1908.bench	146	51	22	73	301
	c3540.bench	278	50	77	151	2096
	c5315.bench	384	35	27	322	1835
	c6288.bench	402	27	112	263	13164
	c7552.bench	585	27	37	521	5522
Hydra (SAT Solver based)	c432.bench	25	10	0	15	1
	c499.bench	32	14	0	18	2
	c880.bench	62	24	0	38	1
	c1355.bench	90	19	0	71	2
	c1908.bench	145	51	0	94	4
	c3540.bench	277	91	0	186	16
	c5315.bench	383	204	0	179	24
	c6288.bench	401	227	2	172	42
	c7552.bench	584	286	0	298	74

Figure 3.4: Comparison between MDFTG and Hydra for Experiment # 2

3.5.3 Comparison of SAT Solver based approach with Conventional ATPG

There are a number of advantages when using the SAT Solver based approach . These are :

1. It is easy to formulate the CNF clauses to restrict the solution space to have the desired features .i.e a set of gates being set to '1' or a line being s-a-0. In contrast , the ATPG based approach has to modify the circuit(by adding gates to ensure a set of gates switch together) to find the required set of vectors.

2. ATPG based method can only work with a reduced victim list V^R , such that no two gates in the reduced list have a path between them. This is essential for the algorithm

to be able to modify the circuit. But the SAT solver based approach can work with the entire victim list V

3. As circuit modification is time consuming , ATPG based procedure tries to minimize the time involved by eliminating entire aggressor lists , instead of refining the list to find a maximal excitable set which the SAT solver based approach does. This results in an improved accuracy of test generation.

4. Finally , due to the provisions of iterative solutions to a group of CNF clauses , the SAT solver can remember the implications it had made while processing the normal circuit CNF , and thus needs very little time when additional sets of clauses are added to the same group.

The results obtained by the implementation of our algorithm shows a clear superiority over the ATPG based approach , both in terms of successful test generation as well as the time taken for completion . This difference is magnified in case of larger and more complicated circuits like $c7552$ and $c6288$.

Chapter 4

Prototype Algorithm for Test Generation In Sequential Circuits

Contents

4.1 Preliminaries **30**

4.2 Algorithm for MDF detection **32**

4.3 A working example : s27 . . . . **37**

4.4 Observations on our proposed method **39**

In this section , a prototype algorithm is developed which would identify test-sequences for MDF detection in full-scan sequential circuits. There are a couple of basic advantages of using Boolean Satisfiability as the underlying combinational test generation method :

- It requires only a few minor adjustments to extend the algorithm for a sequential ATPG.
- The Boolean Satisfiability method can be easily extended to handle various fault models.

4.1 Preliminaries

A sequential circuit (as represented in the Huffman model in figure 4.1) with n inputs , m outputs and d D flip-flops can be unrolled into multiple time-frames , each frame with a

purely combinational circuit with n inputs , m outputs , d "Pseudo-Primary Inputs"(PPI) and d "Pseudo-Primary Outputs"(PPO) .

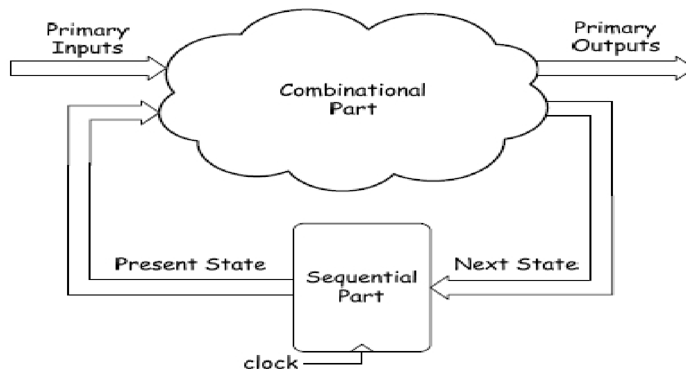

Figure 4.1: The Huffman model for sequential circuits

This has the advantage of bringing the circuit in consideration to the combinational domain , with the added constraint that the values at the PPO's of time-frame $(i-1)$ must match the values at the PPI's of the ith time frame. For convenience of reference , the following conventions would be used:

- PPI_j^i will denote the jth Pseudo-Primary Input of ith time-frame.
- PPO_j^i will denote the jth Pseudo-Primary Output of ith time-frame.
- The logic value of PPO_j^{i-1} is the value given to PPI_j^i
- The vector V returned by the SAT Solver which satisfies a given set of CNF clauses in the "unrolled" sequential circuit as shown in figure 4.2 , has input assignments for both the Primary Inputs(PIs) and the Pseudo-Primary Inputs(PPIs). Hence , for clarity of reference , $V_{PI}[i]$ denotes the value assigned to the ith PI , while $V_{PPI}[i]$ denotes the assignment to the ith PPI.

Thus we see MDF detection for sequential circuits involve only an additional constraint to be imposed while determining the test sequence , the corresponding clauses for which are easily generated . However , to be successful in finding a test sequence , we may have to consider more than 1 possible assignment for the first vector $TV[0]$ and work backwards from it. Hence , an iterative procedure is involved , which tries to find the test sequence till either all possible

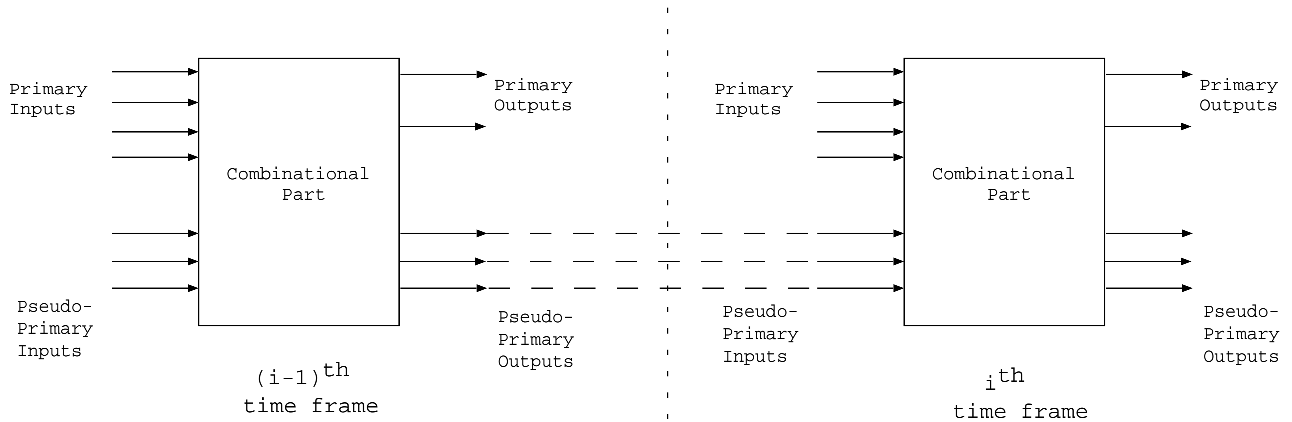

Figure 4.2: Time-frame expansion of a Sequential Circuit

assignments for $TV[0]$ are exhausted , or a test sequence is found. This approach has been formalized in the next section , where the complete algorithm for test detection is presented.

4.2 Algorithm for MDF detection

Algorithm *get_All_Assignments* (*Clauses* , **C**)

Input : A set of CNF clauses *Clauses* defined on the circuit C with n PI's and d PPI's

Output : All possible assignments of PI's and PPI's which satisfy *Clauses*

Begin:

1. Check if *Clauses* is satisfiable by feeding it into the SAT Solver. If not , return an empty vector , signifying that the condition specified by *Clauses* was not satisfied.

2. Let $Seed_Vector$ be the vector that satisfies *Clauses*. Put $Seed_Vector$ in a queue $Generated_TV$.

3. Denote the array of all assignments to be returned as $All_Possible_Vectors$ and add $Seed_Vector$ to it.

   while $Generated_TV$ is not empty **do**

   Pop the first vector from $Generated_TV$. Denote it by V.

   Initialize $PPI_Constraints = \phi$, which is used to hold CNF clauses denoting constraints to be placed on PPI's in order to generate other vectors which also satisfy

$Clauses$.

for $1 \leq i \leq d$ **do**

Let X be the variable associated with the ith PPI and $v = V^{i}_{PPI}$ be the value assigned to it in vector V .

if v is 1 **then**

$Opposing_Clause = \overline{X}$

$Supporting_Clause = X$

else

$Opposing_Clause = X$

$Supporting_Clause = \overline{X}$

end if

Add $Opposing_Clause$ to the list of constraints $PPI_Constraints$.

Solve the conjunction of $Clauses$ and $Opposing_Clause$. If any assignment V' exists , it is guaranteed to be different from V as it has at least one PPI assignment different from it.

Check if V' is present in $All_Possible_Vectors$. Add V' to $All_Possible_Vectors$ and $Generated_TV$ if absent.

Remove $Opposing_Clause$ from $PPI_Constraints$ and add $Supporting_Clause$ to it

end for

end while

return $All_Possible_Vectors$

End.

Algorithm $MDF_Sequential_TPG$**(** **C** **,** V **,** $M_Ag(T)$ **)**

Input : A circuit C, a victim list V, an $M_Aggression$ $M_Ag(T)$.

Output : If successful then a test sequence TV and number of test vectors in it else failure.

Begin:

1. Initialize the 3 counters hope H , effectiveness covered E , effectiveness at stake S, which

are used to keep track of the algorithm's progress , as follows :

$$E = 0, H = \sum_{\lambda_i(T) \in M_Ag(T)} \xi(\lambda_i(T)), S = 0$$

If $H < 1$, terminate the algorithm and output failure.

2. Generate TV[0] as follows:

   (a) Compute C_1^0 , the CNF clauses for detecting multiple $s-a-0$ fault at gates in victim list , by calling procedure $get_Multiple_Fault_Detection_Clauses(V, 0)$.

   (b) Identify the maximal simultaneously excitable subset in $\lambda_0(T)$ by calling procedure $get_Maximal_Excitable_Subset(\lambda_0(T))$. Refine the given $M_Aggression$ by replacing $\lambda_0(T)$ by this subset .Compute C_2^0 , the CNF clauses which ensure that all the gates in this subset are set to '1' .

   (c) If C_1^0 is unsatisfiable , then MDF is *redundant* and the algorithm terminates .

   (d) if C_1^0 and C_2^0 are jointly satisfiable , then $C^0 = C_1^0.C_2^0$ else $C^0 = C_1^0$.

   (e) Call procedure *get_All_Assignments* (C^0 , C) to get all candidate input assignments (for both PI's and PPI's) that satisfy C^0 , and store it in $Candidate_TVs$. The desired TV[0] is one of the input assignments in $Candidate_TVs$.

   Update the counters as follows :

$$H = H - \xi(\lambda_0(T))$$

   If $C^0 = C_1^0.C_2^0$ is the clause from which all candidate vectors for TV[0] are generated , then

$$S = \xi(\lambda_0(T))$$

3. Find the rest of the test sequence as follows :

   for Each input assignment $TrialVector$ in $Candidate_TVs$ **do**

   Let $C_1^1 \leftarrow$ CNF clauses which set the gates in $\lambda_0(T)$ to '0'

   Refine $\lambda_1(T)$ by identifying the maximally excitable subset in it .

Let $C_2^1 \leftarrow$ CNF clauses which set the gates in the refined $\lambda_1(T)$ to '1'

Let $C_3^1 \leftarrow$ CNF clauses which sets the gates in V to '0'.

Let $C_4^1 \leftarrow$ CNF clauses which constrain the PPO's of circuit C to be equal to their respective PPI assignments as given in $TrialVector$.

if $C_1^1.C_2^1.C_3^1.C_4^1$ is satisfiable **then**

Store this input vector as TV[1]

else if $C_1^1.C_1^3.C_4^1$ is satisfiable **then**

Store the assignment as TV[1]

else if $C_1^2.C_1^3.C_4^1$ is satisfiable **then**

Store the input assignment as TV[1]

else

Since none of the combinations of clauses could be satisfied , no transition is possible. Hence stop and output failure.

end if

$$H = H - \xi(\lambda_1(T))$$

If TV[1] sets the gates in refined $\lambda_0(T)$ and $S > 0$, implying that TV[0] sets gates in $\lambda_0(T)$ to '1' and the transition of $\lambda_0(T)$ is captured , then

$$E = E + \xi(\lambda_0(T))$$

If $E \geq 1$, then output the test sequence . If TV[1] sets gates in refined $\lambda_1(T)$ to '1' , then

$$S = \xi(\lambda_1(T))$$

If $(E + S + H) < 1$, terminate and output failure.

Generate TV[i] , $2 \leq i \leq M$, as follows :

for $2 \leq i \leq M$ **do**

$C_3^i \leftarrow$ CNF clauses which ensure that the PPO's of circuit C are equal to their respective PPI assignments as given in TV[i-1] .

if $\lambda_i(T)$ and $\lambda_{i-1}(T)$ are both empty **then**

TV[i] is a test vector satisfying C_3^i.

else

$C_1^i \leftarrow$ CNF clauses to set gates in λ_{i-1} to 0

$C_2^i \leftarrow$ CNF clauses to set gates in λ_i to 1

if $C_1^i.C_2^i.C_3^i$ is satisfiable **then**

TV[i] is found

if $S > 0$ **then**

{*This means TV[i-1] sets* $\lambda_{i-1}(T)$ *to 1 and transition of* $\lambda_{i-1}(T)$ *has been captured*}

$E = E + \xi(\lambda_{i-1}(T))$

if $E > 1$ ***then***

Stop and output the test vectors

end if

end if

else

if $\lambda_i(T)$ is not empty AND $S > 0$ **then**

if $C_1^i.C_3^i$ is satisfiable **then**

TV[i] found.

S = 0

else if $C_2^i.C_3^i$ is satisfiable **then**

TV[i] found

$S = \xi(\lambda_i(T))$

else

TV[i] is a random vector

end if

end if

end if

end if

end for

end for

(for $TV[M+1]$)

If $TV[M]$ sets gates in $\lambda_M(T)$ to '1', then find a test vector which sets $\lambda_M(T)$ to '0' and whose PPO values are correspondingly equal to the respective PPI values of $TV[M-1]$. If this test vector is possible, then

$$E = E + \xi(\lambda_M(T)).$$

Else if test vector above is not possible (empty), then output failure.

End.

4.3 A working example : s27

Once again , we take a small example to illustrate the working of the algorithm . We have chosen the circuit s27 for ease of comprehension of the result and the ability to verify the result by inspection . The nomenclature scheme for the variables representing lines in the circuit is the same as in 3.4 . The format of writing the input vectors is as follows : Primary Input (PI) assignments in order followed by the Pseudo-Primary Input (PPI) assignments in order , separated by a space. The ordering of the PI vectors is : G0 G1 G2 G3 , while the order of the PPI's is : G5 G6 G7. The PPI assignments of the first vector TV[M] is applied via Scan-In and for all subsequent vectors , the PPI assignments are done by functional output of the combinational circuit due to the previous vector.

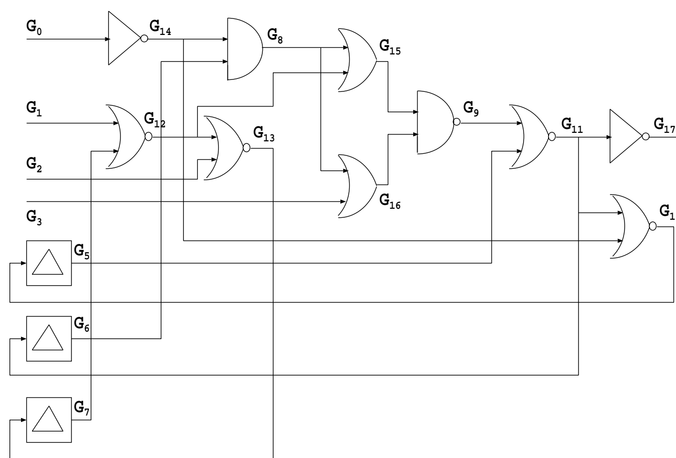

Figure 4.3: The ISCAS89 benchmark circuit s27

The inputs are as follows :

Circuit C : s27.bench

Victim list V: Gates G11 , G12

$M_Ag(T)$**:** It is as follows :

0 : 4 (G11 , G12 , G15) 0.9

1 : 2 (G14 , G16) 0.26

2 : 2 (G8 , G9) 0.1

3 : 0

- $H = 1.26$, $S = 0$, $E = 0$

- C^{Normal} i.e. the normal circuit CNF clauses which are appended with other CNF clauses when solving :
$(G14' + G10').(G11' + G10').(G10 + G14 + G11).(G5' + G11').(G9' + G11').(G11 + G5 + G9).(G1' + G12').(G7' + G12').(G12 + G1 + G7).(G2' + G13').(G12' + G13').(G13 + G2 + G12).(G0' + G14').(G14 + G0).(G12' + G15).(G8' + G15).(G15' + G12 + G8).(G3' + G16).(G8' + G16).(G16' + G3 + G8).(G11' + G17').(G17 + G11).(G14 + G8').(G6 + G8').(G8 + G14' + G6').(G16 + G9).(G15 + G9).(G9' + G16' + G15')$

- $C_1^0 =$
$(G11'_f + G12'_f).(G14' + G10'_f).(G11'_f + G10'_f).(G10_f + G14 + G11_f).(G5' + G11'_f).(G9'_f + G11'_f).(G11_f + G5 + G9_f).(G2' + G13'_f).(G12'_f + G13'_f).(G13_f + G2 + G12_f).(G12'_f + G15_f).(G8' + G15_f).(G15'_f + G12_f + G8).(G11'_f + G17'_f).(G17_f + G11_f).(G16 + G9_f).(G15_f + G9_f).(G9'_f + G16' + G15'_f).(G17' + G17_f + X).(G17 + G17'_f + X).(G17 + G17_f + X').(G17' + G17'_f + X').X$

- Maximal excitable subset in $\lambda_0(T)$ is G11 , G12 , G15 . Therefore , C_2^0 , the CNF clauses to set this set of gates to '1' simultaneously is $(G11).(G12).(G13)$

- $C^0 = C_1^0.C_2^0$ and the set of all possible candidates for $TV[0]$ that satisfy C^0 are $\{0011\,000, 1011\,010\}$. From this set , we choose the first trial solution as $TV[0]$ and try to find a test sequence.

$$TV[0] = 0011\ 000$$

- C_4^1 , i.e. clauses to ensure $TV[0]$ PPI's are a result of functional justification of $TV[1]$ are $(G10').(G11').(G13')$
- Set of gates in refined $\lambda_1(T)$ is G14,G16 .
- Test vector which simultaneously sets V to '0' , $\lambda_0(T)$ to 0 , $\lambda_1(T)$ to '1' and ensures that $TV[0]$ PPIs are a functional justification of its PPO's, is

$$TV[1] = 0111\ 101$$

- $H = 0.36$, $S = 0.9$, $E = 0$
- Maximal excitable subset in $\lambda_2(T)$ is G8 , G9 .
- Test vector that sets $\lambda_1(T)$ to '0' and $\lambda_2(T)$ to '1' , and whose PPO assignments match the corresponding PPI values as given in $TV[1]$ is

$$TV[2] = 1100\ 101$$

- $H = 0$, $S = 0.1$, $E = 1.16$.
- As $E > 1$, MDF vector sequence found which are as follows :
  $TV[2] = 1100$, Scan-In $= 101$
  $TV[1] = 0111$
  $TV[0] = 0011$

4.4 Observations on our proposed method

Empirical results show that in larger ISCAS89 circuits, the number of DFFs (and hence PPI-PPO pairs) dominate the number of Primary Inputs by a wide margin. This leads to a large number of satisfiable instances for $TV[0]$, and since the algorithm is an iterative one, it takes quite a lot of time to weed out the instances for which there are no MDF test vector sequences. So, care must be taken during implementation to ensure optimality in the operations inside the loop, to spend lesser time in test generation.

Chapter 5

Implementation Details

The entire project has been implemented in C++ . For solving the SAT instances , we have used a widely documented SAT Solver Zchaff [7] , which was chosen due to its flexible API and its reputation as being one of the fastest known SAT solvers.

5.1 Description of Important Classes

Class ***LogicGate*** :

This is an abstract class representing a generic Logic Gate. The explicit nature of the gate is set by deriving into a specific type like AND , OR , NOT etc. In such a derived class , all methods which depend on the nature of the gate have been made *virtual* , enabling the derived class(es) to represent each specific type of gate and their functionality by overriding the virtual base method defined in this class .

Some class members -

1. Pointer to the the fan-out *Signal* of the gate .
2. Array of pointers to the fan-in *Signal*s .
3. Array to store the corresponding CNF expressions for the gate .

Class *Signal* :

Used to encapsulate a generic signal line of a logic circuit , which can assume the values '0', '1' or 'X' . Each such object has pointers to the *LogicGate* objects to which it is connected in the circuit.

Some class members -

1. Name of the signal line
2. Pointer to its Parent *LogicGate*
3. Array of pointers to *LogicGate*s to which it is a fan-in line.
4. Two unique integers , one representing the signal in fault-free circuit and the other , in the faulty circuit,

Class *Circuit*:

Each instance encapsulates a complete circuit definition ,which it constructs from its description given in ISCAS89 benchmark format. The data structure is designed keeping in mind the inherent hierarchy of the circuit , so that one can query the status of an internal signal line in minimal time.

Some class members -

1. A LookUp Table , with names of the *Signal*s as key and the pointer to that object as the lookup value. This has been stored in the form of a Balanced Binary Search Tree , in order to optimise the look-up time.
2. Array of *Signal*s, storing the pointers to the Primary Inputs .
3. Array of *Signal*s, storing the pointers to the Primary Outputs .

Class *MultiDroopTPG* :

Inherited from class *Circuit* , this class uses the underlying data structures to model Multi-Cycle Droop Fault (MDF) and generate test sequences for MDF detection given a list of *M_Aggressions*

5.2 Summary of Important Methods

Circuit::buildCircuitFromNetlist

This method takes the input circuit in ISCAS89 .bench format and builds the underlying data structures , keeping in mind the hierarchical nature of the circuit to ensure that the most frequent operations (such as querying the value of a signal , setting a fault in a line etc) are done in minimal time.

MultiDroopTPG::generateLogicalNeighbourAggressions

Generates *M_Aggressions* using the method of logical neighbors for a given circuit.

MultiDroopTPG::generateRandomAggressions

Generates *M_Aggressions* using the method of random selection for a given circuit.

MultiDroopTPG::getMDFTestVectors

Writes the test vectors for the *M_Aggressions* input by the user , or generated by the above two methods , to a given file.

5.3 How to use Hydra - our MultiDroopTPG

The online help for the tool Hydra is given below , along with a detailed explanation of each of its options.

```
Hydra 1.0

Usage: ./Hydra -c <Netlist filename> [-M <Number of Cycles>]
(-a <Aggressorlist filename> | -r | -n) [-t <TestVector filename>]
[ -l <Log filename> ]

Arguments in [] are optional...

Arguments in () indicate one & only one option is required
to be specified ...

Use -l to see log on stdout
```

-c It is followed by the name of the file from which the circuit is to be constructed. Compulsory.

-M Indicates the number of cycles over which MDF occurence is to be tested . Optional , default value is 6.

-a It is an option to supply a user-defined list of $M_Aggressions$. Optional .

-n Denotes generation of $M_Aggressions$ by logical neighbours method. Optional.

-r Denotes generation of $M_Aggressions$ by random selection method. Optional.

-t Specifies the file-name where the test vectors for each $M_Ag(T)$ is to be written . Optional , default output is a file named "testvector.txt" .

-l Specifies the name of the logfile for facilitating debugging . Used without any argument , produces the log output on the terminal itself. Optional .

Chapter 6

Conclusion

Experiments show that even if we chose $M_Ag(T)$'s from logically nearer gates (Experiment # 1) there can be some MDF which are non redundant and hence testable. Also random selection (Experiment # 2) has many testable $M_Ag(T)$'s. These methods are two extreme cases to choose an aggression. In practice an $M_Ag(T)$ made from a real chip would be less constrained than that in Experiment # 1 and less random than that in Experiment # 2. So there is a considerable chance of MDF occurring in a real high performance chip specially in future chips.

6.1 Scope for Future Works

For complex sequential circuits , an iterative approach to finding out the MDF vectors may take a considerably long time. So, it would perhaps be better to formulate the entire sequence in to a single set of CNF clauses to be solved. Of course, the implementation needs to be done carefully since for each time-frame , a new set of variables would be required to represent the states of the signals in that frame , and so the potential state-space of satisfiable assignments may be too large.

Bibliography

[1] Tracy Larrabee, *Test Pattern Generation Using Boolean Satisfiability*, IEEE Transactions On Computer-Aided Design , Vol 11 , No. 1 , January 1992

[2] Onkar N. Tiwari, *Multi Cycle Droop Fault in Combinational Circuits*, M. Tech. (Computer Science) Dissertation Series, Indian Statistical Institute, 2008.

[3] S. Pant, E. Chiprout, D. Blaauw, *Power Grid Physics and implications for CAD*, Proc. of the 43rd Annual Conference on Design Automation, ACM IEEE Automation Conference, Session 13, pp. 199-204, 2006.

[4] D. Mitra, S. Bhattacharjee, S. Sur-Kolay, B. B. Bhattacharya, S. T. Zachariah and S. Kundu, *Test Pattern Generation for Droop Faults*, Proc. of the 19th International Conference on VLSI Design, pp. 343-348, 2006.

[5] I. Polian, A. Czutro, B. Becker, S. Kundu, *Power Droop Testing*, IEEE Design & Test, Volume 24, ISSUE 3, pp. 276-284, May 2007.

[6] C. Tirumurti, S. Sur-Kolay, S. Kundu, Y. S. Chang, *A Modeling Approach for Addressing Power Supply Switching Noise Related Failures of Integrated Circuits*, Proc. Design, Automation and Test in Europe Conference and Exhibition Volume II (DATE'04), pp 21078, 2004.

[7] *http://www.princeton.edu/ chaff/zchaff.html.*