Test Pattern Generation Algorithm Using Structurally Synthesized BDD

Mousumi Saha, Naveen Singh Bisht, Shrinivas Yadav, Praveen Kumar K

***Abstract*— Structurally Synthesized Binary Decision Diagrams (SSBDDs) have an important characteristic property of keeping information about circuit's structure. Boolean difference of a circuit is used to find test pattern for stuck at fault in combinational circuit but the algebraic manipulation involved in solving Boolean difference is a tedious job. In this paper an efficient algorithm is proposed to compute Boolean difference and test patterns simply using searching the paths of SSBDD. This model reduces algebraic manipulations and takes less time to compute the test pattern.**

***Index Terms*— Automatic Test Pattern generation, Binary decision diagrams, Boolean difference, Stuck at Fault, Structurally Synthesized Binary decision diagram.**

I. INTRODUCTION

In a combinational circuit the presence of a single stuck at fault can be tested easily by a set of inputs. These set of inputs are applied to faulty circuit which generate different output. The variation in output will show the circuit is faulty or fault free. Automatic test pattern generation is a method by which we can test the fault of a circuit. Under this method a set of binary digits are produced. Using these sets as input to circuit, we can easily detect the every single stuck at fault of the circuit. During last decade a large number of test pattern generation algorithms have been proposed. There are two types: structural method and algebraic method. The most notable structural methods are D algorithm [1] [2], PODEM, FAN, SOCRATES etc. Boolean difference is the most famous algebraic method to find test pattern [3].

Binary Decision Diagram and Reduced Ordered Binary Decision Diagram are used for representation and manipulation of Boolean Functions [4, 5, 6, and 7]. But this model have some problem like it suffers from the memory explosion, which limits its usability on large designs. So we cannot use as a model for such methods that require a certain degree of structural information about the design. To overcome this drawback we are moving a new concept called Structurally Synthesized Binary Decision Diagram (SSBDD). SSBDD is one of the popular model to represent the Boolean function of a Boolean circuit. It keeps the information of circuit structure of digital circuit [8].

Manuscript received 26th November, 2010.

Mousumi Saha is with the National Institute of Technology, Durgapur, India (Phone: 91-9474487495; Fax: 343-2547375; Email: msaha.nitd@gmail.com.)

Naveen Singh Bisht is with the Tata Consultancy Services, Pune, Maharastra. India (Email:naveen.bisht@tcs.com.)

Shrinivas Yadav is with the Huawei Technologies Co. Ltd, Bangalore, India (Email: shrinivasy@huawei.com.)

Praveen Kumar K is with the National Institute of Technology, Durgapur, India (Email: praveen.kumar432@gmail.com.)

In this paper an efficient and effective algorithm is proposed to find the test pattern of a combinational circuit using searching the paths of corresponding SSBDD. It is clear that the efficiency of any algorithm is said to be increased, if the simulation time was reduced. In this approach a small amount of algebraic manipulation is required. So this approach presents the easy way to find the test pattern and very helpful to find the test pattern for large circuits, fan and fan-free circuits also.

The paper is organized as follows. In section 2 we describe the preliminaries of SSBDD. In section 3 and section 4 are dedicated to explain proposed approach and algorithm for generating test pattern successively. In section 5 include some experimental result on appropriate examples. Section 6 highlights directions for future work. Finally section 7 gives conclusion.

II. PRELIMINARIES

A. *Basic concept of Boolean difference*

Boolean difference is an algebraic method to find the test pattern for a combinational circuit. It can detect errors at any position in a circuit. The Boolean difference of a function $f(x_1, x_2, x_3 \dots \dots \dots x_d \dots \dots \dots \dots x_n)$ with respect the variable x_d is

$$\frac{\partial f}{\partial x_d} = f_{x_d} \oplus f_{\overline{x_d}}$$

Here f_{x_d} and $f_{\overline{x_d}}$ are obtained from y=f(x) by replacing variable x_d by value 1 and 0 correspondingly. The Boolean difference with respect to the variable x_d indicates whether f is sensitive to changes in input x_d. If the function does not depends on x_d. In that case, x_d is said to be unobservable. This method needs huge manipulation of mathematical expression for a particular Boolean function. But once we know the Boolean difference, then the test patterns can easily determine for a circuit.

B. *Basic concept of SSBDD*

The SSBDDs are based upon the equivalent parenthesis form (EPF), that is, they describe a digital circuit structurally [9] [10]. The SSBDD models are generated by a superposition procedure that extracts information about both, function and data of structural paths of the circuit. For example the equivalent SSBDD of a combinational circuit in figure 1 is as shown in figure 2.

SSBDD is a very powerful model for representation and manipulation of Boolean function. This model has different flavor compared to other commonly used mathematical model. First time it was introduced as Structural Alternative Graphs [3] and generalized as multiple-valued decision diagrams in [11].

SSBDD model reduces the no of nodes to represent the circuit compared to the ROBDD. One more is the size of the SSBDD model is linear in respect to the circuit size while it is exponential for ROBDD [12].Due to that simulation time [13, 14] can be reduced remarkably to generate the test pattern.

DOI: 10.5176_2010-2283_1.2.31

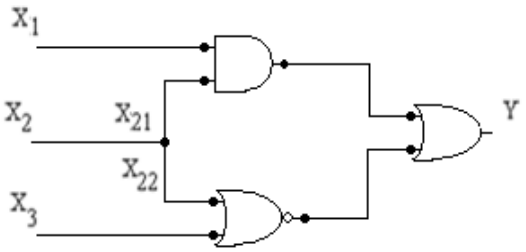


Fig. 1. Circuit for $y=x_1x_{21}+\bar{x}_{22}\bar{x}_3$

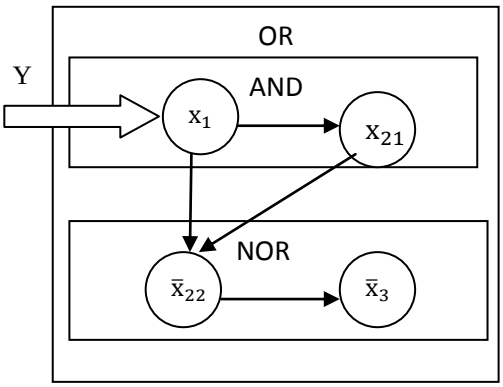


Fig. 2. Illustration of superposition principle for the Function $y=\mathbf{x_1x_{21}}+\bar{\mathbf{x}}_{\mathbf{22}}\bar{\mathbf{x}}_{\mathbf{3}}$.

III. PROPOSED APPROACH

We follow the below to calculate the test pattern.

1. SSBDD should be drawn starting from suspicious (faulty) literal. It can be represented by γ.
   If A is stuck at fault 0 then we draw SSBDD having root node as A. (where γ=A).
   If A is stuck at fault 1 then we draw SSBDD having root node as A. (where $\gamma=\bar{A}$).
2. We are using tracking direction R for RIGHT and D for DOWN.
3. We define a term TRACK [A][B] where A,BЄ{R,D} means it is set of recursively all traced literals starting from next to root node, first traced at A side and end with all having no final B side.
4. Example for finding the TRACK [A][B]:
   In finding TRACK [A][B] ,we follow recursive approach to find all paths and will stop when there is no more path. Let's take an arbitrary SSBDD, which is as s hown in figure 3. TRACK[R] [R] is found as follows. TRACK[R] [R] means we are tracing the paths next to root node in the RIGHT direction until there is no path in the same direction. If any intermediate node having downward path we have to trace that path and reach that downward node, from there again move in the right direction only in a recursive manner.
   In figure 3 the paths for TRACK[R] [R] are v2 to v3, v2 to v5 via v4, and v2 to v8 via v4 and v5. Similarly TRACK[D][D],TRACK[R][D],TRACK[D][R] will follow the same procedure.
5. We define another term COMPATIBLE (A, B) where A, B are literals or product of literals. This returns the term Cartesian product of (A, B).which are not contradicting (not having literal and complement of literal in same term) each other.
   Example:
   a) COMPATIABLE $(\bar{A}BC, ABC) = \bar{A}BC * ABC$
   $= \Phi$.
   b) COMPATIABLE (AC, AD) $= AC * AD$
   $= ACD$.
   c) Let $A=\{AB, \bar{A}C\}$, $B = \{A, A\bar{C}\}$ than
   COMPATIBLE (A, B) $= [\{AB, \bar{A}C\} * \{A, A\bar{C}\}]$
   $= \{AB, AB\bar{C}\}$.
6. α= COMPATIABLE (TRACK[D][D] * TRACK [R][R])
   β= COMPATIABLE (TRACK[R] [D] * TRACK [D][R])
7. μ = α ∪ β.

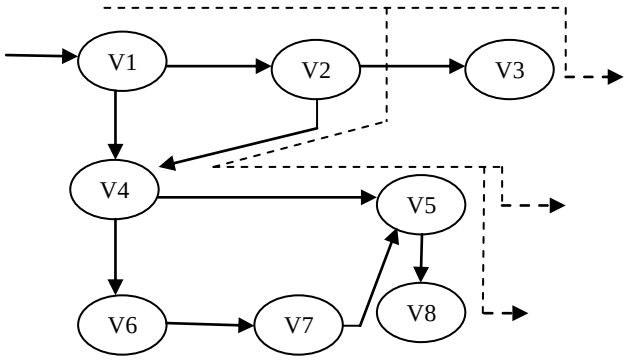


Fig. 3. Finding TRACK[R][R]

IV. PROPOSED ALGORITHM

In figure 4, the flow chart is used to explain the algorithm to find test pattern. The following steps are considered.

1. For given input Boolean function or circuit we draw SSBDD with faulty literal as root node γ.
2. Initially we keep solution set α, β, μ as empty.
3. We have used term
   TRACK[D][D]=Track recursively starting from root node , first going DOWN till find a node having no DOWN and stop when no such path is exists.
   TRACK[D][R]=Track recursively starting from root node , first going DOWN till find a node having no RIGHT and stop when no such path is exists.
   TRACK[R][D] =Track recursively starting from root node , first going RIGHT till find a node having no DOWN and stop when no such path is exists.
   TRACK[R][R] = Track recursively starting from root node, first going RIGHT till find a node having no RIGHT and stop when no such path is exists.
4. If for root node one of the tracks i.e. TRACK [D] or TRACK [R] does not exists then μ will be other track.
   Where TRACK [D] means the path from root node to DOWN word does not exists and TRACK [R] means the path from root node to RIGHT word does not exists.
5. If both TRACKS exists form root node then we find TRACK [D] [D] & TRACK[R] [R] and calculate
   α=
   COMPATIBLE (TRACK [D] [D] * TRACK [R] [R])
   And then find TRACK [D] [R] & TRACK[R] [D] and calculate
   β=
   COMPATIBLE (TRACK [D] [R] * TRACK[R] [D])
6. Find union of α and β as μ.

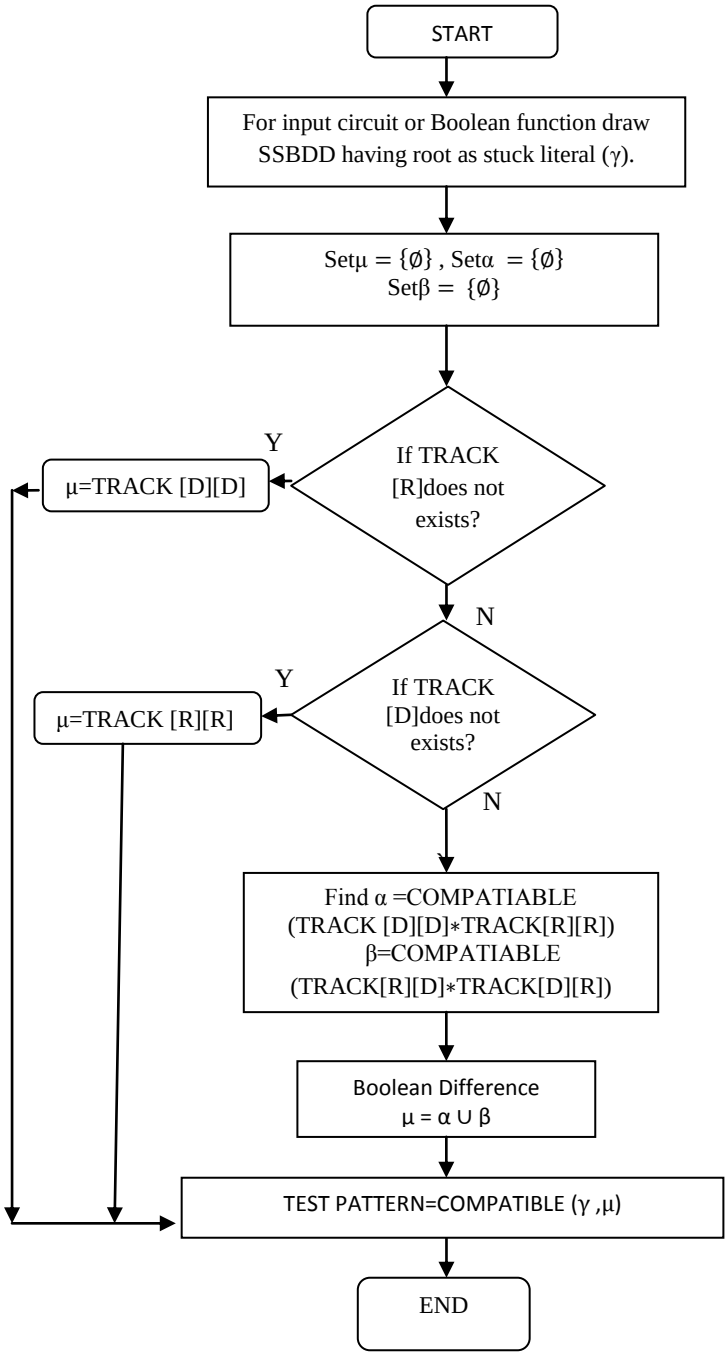


Fig. 4. Algorithm for finding test pattern

7. Now finally getting test pattern just calculates COMPATIBLE (γ, μ).

V. EXAMPLES

This section explained generation of test pattern with the help of different types of models, like "track missing" and "no track missing". The track missing means that all combinations of TRACKs which are not existed in that SSBDD. All combinations means TRACK [R][R], TRACK [D][D],TRACK [R][D] and TRACK[D] [R].No track missing means all combinations of tracks present in a given SSBDD. We explain one more case called undetectable fault, for which the Boolean difference is ∅ , which means that even if the fault present in the Boolean circuit, we cannot detect that fault.

A. Track is missing

Let the function F (A, B, C) = A (B+C) and A as stuck at fault 0 and 1.

The following steps are considered to find test pattern using algorithm.

Step1. SSBDD for given function is shown in the figure 5. Here fault is assumed to be at A. So root node is taken as A.

Step2. Here A is stuck literal, so stuck at 0 can be taken as γ = A and stuck at 1 can be taken as $\gamma=\overline{A}$.

Step3. Here TRACK[R][D],TRACK[D][R],TRACK[D][D] are missing.

Step4. Boolean difference μ is TRACK[R] [R]
= {B, $\overline{B}$C}
= B+$\overline{B}$C
= B+C
= {B, C}.

Step5. Test pattern for A stuck at 0
= COMPATIBLE (γ, μ)
= A {B, C}
= {AB, AC}
= {11Φ, 1Φ1}.

Test pattern for A stuck at 1
= COMPATIBLE (γ, μ)
= $\overline{A}$ {B, C}
= {$\overline{A}$B, $\overline{A}$C}
= {01Φ, 0Φ1}.

B. No track is missing

Let function F (A, B, C) = AB+$\overline{B}$C and B as stuck at fault 0 and 1.

The following steps are considered to find test pattern using algorithm.

Step1. SSBDD for given function is shown in figure 6. Here fault is assumed to be at B. So root node is taken as B.

Step2. No TRACKS are missing in a given function F = AB+$\overline{B}$C.

Step3. All the TRACKs can be calculated as
TRACK [D][D]={ $\overline{C}$ }
TRACK [R][R]={ A }
TRACK [R][D]={ $\overline{A}$ }
TRACK [D][R]={ C }

Step4. α =
COMPATIABLE (TRACK [D] [D] TRACK [R] [R])
= COMPATIABLE { $\overline{C}$ ∗ A }
= {A $\overline{C}$}.

β =
COMPATIABLE (TRACK[R] [D] TRACK [D] [R])
= COMPATIABLE { $\overline{A}$ ∗ C}
= { $\overline{A}$ C }.

Step5. Boolean difference
μ = α ∪ β
= { A $\overline{C}$} ∪ {$\overline{A}$ C }
= { A $\overline{C}$, $\overline{A}$ C }.

Step6. Test pattern for B stuck at 0
= COMPATIBLE (γ, μ)

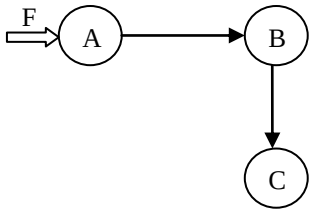


Fig. 5. SSBDD diagram for F=A (B + C).

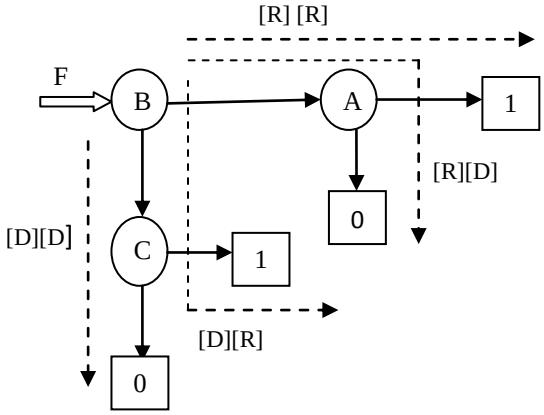


Fig. 6. SSBDD diagram for F= AB+$\bar{B}$C

= Bμ
= B { A $\bar{C}$, $\bar{A}$ C }
= {AB$\bar{C}$, $\bar{A}$ BC }
= {110,011}.

Test pattern for B stuck at 1
= COMPATIBLE (γ, μ)
= $\bar{B}$μ
= $\bar{B}$ { A $\bar{C}$, $\bar{A}$ C }
= {A$\bar{B}\bar{C}$, $\bar{A}$ $\bar{B}$C }
= {100,001}.

Let one more function F (A, B, C, D) = $\bar{A}\bar{B}$ +A$\bar{B}$+ C$\bar{D}$ + $\bar{C}$D and A as faulty literal.

The following steps are considered to find test pattern using algorithm.

Step1. SSBDD for given function is shown in figure 7. Here fault is assumed to be at A. So root node is taken as A.

Step2. No TRACKs are missing in a given function F.

Step3. All the TRACKS can be calculated as
TRACK [D][D]={ B$\bar{C}\bar{D}$, BCD }
TRACK [R][R]={ $\bar{B}$, BC$\bar{D}$ }
TRACK [R][D]={ B$\bar{C}\bar{D}$, BCD }
TRACK [D][R]={ $\bar{B}$C$\bar{D}$, $\bar{B}$C$\bar{D}$ }

Step4. α =
COMPATIABLE (TRACK [D][D] TRACK[R][R])
= COMPATIABLE { B$\bar{C}\bar{D}$, BCD ∗ $\bar{B}$, BC$\bar{D}$ }
= {Φ}.
β =
COMPATIABLE (TRACK [R][D] TRACK[D][R])
= COMPATIABLE {B$\bar{C}\bar{D}$, BCD ∗ $\bar{B}$C$\bar{D}$, $\bar{B}$C$\bar{D}$}
= {Φ}.

Step5. Boolean difference
μ = α ∪ β
= Φ ∪ Φ
= Φ.

Step6. Test pattern for stuck at 0 for A
= COMPATIBLE (γ, μ)
= Aμ
= Φ.

Test pattern for stuck at 1 for A
= COMPATIBLE (γ, μ)
= $\bar{A}$ μ
= Φ.

Step7. It means the function F=$\bar{A}$ $\bar{B}$ +A$\bar{B}$+ C$\bar{D}$ + $\bar{C}$D is not sensitized with respect to the variable A. So the fault at A is undetectable and the given Boolean function of the circuit is not in optimized form. This Boolean function can be redundant. Now we can represent the function F=$\bar{A}\bar{B}$ +A$\bar{B}$+ C$\bar{D}$ + $\bar{C}$D by the function F=$\bar{B}$ + C$\bar{D}$ + $\bar{C}$D and easily understand that this minimized function is totally independent of A. Due to that if any error occurs at A the fault from the primary output line is not observable and detectable.

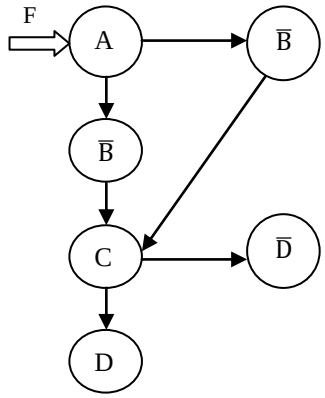
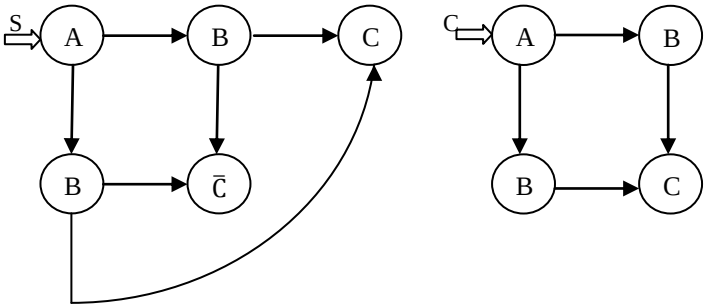


Fig. 7. SSBDD diagram for F=$\bar{A}\bar{B}$ +A$\bar{B}$+ C$\bar{D}$ + $\bar{C}$D



(A) (B)

Fig. 8. SSBDD diagram for (A) Sum S = A⊕ B ⊕ C
(B) Carry C = AB+BC+CA.

C. *Analyzing Multi-output Function*

The proposed algorithm is also efficient for multi-output function ABC. Consider the full adder circuit and corresponding equations are

Sum S = A⊕ B ⊕ C
Carry C = AB+BC+CA

We can easily draw the corresponding SSBDD of Sum and Carry functions as shown in the figure 8. If apply the proposed

algorithm on the SSBDD Sum, then the terms of Boolean difference will be like {BC, $\overline{B}$C, B$\overline{C}$, $\overline{B}\overline{C}$}. Now without computation of Boolean difference for function of carry, we can easily determine the terms of its Boolean difference. Using analyzing the SSBDD of Carry, The term $\overline{B}$C (Considering from one of the term of sum) is also one of the term of Boolean difference of carry with respect to A. Because, if we start from A (Considering figure 8 (B)) and goes to RIGHT and check the result searching the path $\overline{B}$C and again start from A and goes DOWN to check the result using same path $\overline{B}$C. Using this way different values come and then that can be considered as one of the term of Boolean difference for Carry with respect to the root node A. Same procedure will be used to determine all other terms of Boolean difference for second function Carry.

VI. FUTURE WORK

Here we showed that it is easy to find test pattern for single stuck at fault using SSBDD for any Boolean function. We are going to use SSBDD for finding test patterns for multiple stuck at fault model or other fault model like stuck-open fault, bridging-fault or delay-fault.

VII. CONCLUSION

Here we showed that it is easy to find test pattern for single stuck at fault using SSBDD for any Boolean function. The Boolean difference is determined using tracing the paths of SSBDD of the circuit. Proposed algorithm reduces the simulation time due to two reasons, one is that algorithm uses the SSBDD model instead of the other BDD model and another is that it reduces mathematical manipulation due to searching the path of SSBDD to find the Boolean difference instead of using classical Boolean difference method.

REFERENCES

[1] Sunil K. Jain, Vishwani D. Agrawal "TEST GENERATION FOR MOS CIRCUITS USING D-ALGORITHM", 20th Design Automation Conference, IEEE published on 1983,

[2] Michael H. Schulz, Erwin Trischler, and Thomas M. Sarfert "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System", IEEE Transactions On Computer-Aided design, VOL. 7, NO. 1, January 1988.

[3] F.F Sellers, m. y. Hsiao And L. W. Beannson, "Analyzing Errors With The Boolean Difference", IEEE Transaction on Computers, Vol.C-17, July1968, 678-683.

[4] S.Minato, Binary Decision Diagrams and Application for VL SICAD. Kluwer Academic Publishers, 1996, 141p.

[5] R.Drechsler, B.Becker. Binary Decision Diagrams, Theory, Implementation Kluwer Academic Publishrs, 1998, 200p.

[6] R. Ubar, "Test Generation for Digital Circuits Using Alternative Graphs (in Russian)", in Proc. Tallinn Technical University, 1976, No. 409, Tallinn Technical University Tallinn Estonia, pp.75-81.

[7] S.Akers, "Binary Decision Diagrams," IEEE Trans. O Comp. ,Vol. 27, 1978, pp.509-516.

[8] Jutman,R.Ubar,"Design Error Diagnosis In Digital Circuits with Stack-at-Fault Model ", Journal of Microelectronics Reliability. Pengamonpresu, VOL 40, NO 2, 2002,pp307-320.

[9] H.-T.Liaw, C.-S.Lin. "On the OBDD representation of General Boolean functions". IEEE Trans.on Comp., Vol.C-41, No.6, pp.61-64, June 1992.

[10] A. Jutman, J. Raik, R. Ubar, "On Efficient Logic Level Simulation Of Digital Circuit Represented By The SSBDD Model" , PROC 23rd International Conference On Micro-electronics (MIFL 2002) VOL 2, pp621-624.

[11] R.Ubar, "Multi-Valued Simulation of Digital Circuits with Structurally Synthesized Binary Decision Diagrams," OPA, Gordon and Breach Publishers, Multiple Valued Logic, 1998 Vol. 4, pp141-157

[12] R. Ubar, A. Jutman, Z. Peng, "Timing Simulation of Digital Circuits with Binary Decision Diagrams", in Proc.of date 2001 Conference, München, Germany,2001,pp.460-466.

[13] R.Ubar, "Parallel Critical Path Tracing Fault Simulation," in Proc.of the 39.Int.Wiss. Kolloquium, Ilmenau, Germany, 19 94, Band 1, pp. 399-404.

[14] A. Jutman, J. Raik, R. Ubar. SSBDDs: "Advantageous Model and Efficient Algorithms for Digital Circuit Modeling Simulation" & Test.5th Int. Workshop on Boolean Problems. Freiberg, Germany, September 19-20, pp.157-166.

[15] H.-T.Liaw, C.-S.Lin. "On the OBDD Representation Of General Boolean functions. IEEE Trans. on Comp., Vol.C-41, No.6, pp. 61-64, June 1992.

Mousumi Saha was born at Durgapur in West Bengal State, India on 21st December 1974. She received her B.E degree (C.S.E) from the Regional Engineering College (Now NIT), Durgapur in 1997 and M.Tech degree in C.S.E from Calcutta University, West Bengal, India in 2001. She is currently working at National Institute of Technology, Durgapur as an Assistant Professor in the Department of Computer Application. Her primary areas of research include Switching Theory and VLSI Testing.

Naveen Singh Bisht was born at Senu Village in Uttrakhand State, India on 17th June,1987. He received his B.Sc degree from Chaudhary Charan Singh University Meerut, Uttrakhand in 2006 and Master of Computer Application degree from National Institute Of Technology Durgapur ,West Bengal in 2010. He is currently working in Tata Consultancy Services as an Assistant System Engineer in Pune, Maharastra.

Shrinivas Yadav was born at Harpur Village in Uttar Pradesh State, India on 10th March 1986. He received his B.Sc degree from University of Allahabad, Uttar Pradesh in 2005 and Master Of Computer Application degree from National Institute Of Technology Durgapur ,West Bengal, India in 2010. He is currently working in Huawei Technologies Co. Ltd , as a Software Engineer in Bangalore, Karnataka.

Praveen Kumar K was born at Panyam Village in the Andhra Pradesh State, India on 2nd July, 1987.He received his B.Tech degree (E.C.E) from Jawaharlal Nehru Technological University, Hyderabad, India in 2009 and is currently pursuing his M.Tech degree in Micro Electronics and VLSI at National Institute Of Technology, Durgapur. His primary research includes VLSI Testing.