Journal of Electronic Testing (2023) 39:323–346
https://doi.org/10.1007/s10836-023-06066-7

A Flexible Concurrent Testing Scheme for Non-Feedback and Feedback Bridging Faults in Integrated Circuits

Pradeep Kumar Biswal[1]

Received: 1 November 2022 / Accepted: 12 April 2023 / Published online: 24 May 2023

Abstract
This paper presents a novel flexible concurrent testing scheme for non-feedback and feedback bridging faults in integrated circuits. All the existing concurrent testing schemes for non-feedback and feedback bridging faults available in the literature are designed with very straightforward manner and these schemes do not provide any idea regarding flexibility in design of the tester circuit. However, providing flexibility in design of the tester circuit is one of the important criteria to be included in concurrent testing of modern integrated circuits. To the best of my knowledge, the proposed scheme will be treated as first concurrent testing scheme that provides flexibility in design of the tester circuit for both non-feedback and feedback bridging faults. This work aims to provide flexibility in tester circuit design by dropping some of wires that are tapped from Circuit Under Test (CUT) to the tester circuit. Dropping some of tap wires from CUT to the tester circuit reduces load on CUT and this helps in reducing the area overhead of the tester circuit. Thus, flexibility in concurrent testing is achieved by dropping some of tap wires during design of tester circuit. Further, dropping some of tap wires (flexibility) in tester circuit design provides a trades-off analysis between area overhead, fault coverage and fault detection latency. The proposed scheme is verified using different ISCAS89 benchmark circuits and results illustrate that flexibility in design of tester circuit is achieved through dropping some of tap wires from CUT to the tester circuit. Further, it reduces area overhead greatly with minimal compromise in fault coverage.

Keywords Concurrent testing · Bridging fault model · Reduced ordered binary decision diagram · Area overhead · Fault coverage · Fault detection latency

Abbreviations

CUT	Circuit Under Test
BIST	Built In Self Test
ATE	Automatic Test Equipment
ROBDD	Reduced Ordered Binary Decision Diagram
FA	Finite Automata
FI-transition	Fault Identification transition
NSF	Next State Function
OF	Output Function
FF	Flip-Flops
BF	Bridging Fault
AO	Area Overhead
FC	Fault Coverage
RTL	Register Transfer Level

Responsible Editor: R. A. Parekhji

✉ Pradeep Kumar Biswal
biswalpradeep28@gmail.com

[1] Department of CSE, IIIT Bhagalpur, Bhagalpur, India

1 Introduction

The density of the devices of present day integrated circuits, that uses deep sub-micron technology, have been increasing rapidly by reducing feature size of the transistors and intercommunication wires. This reduction of feature size of the transistors and intercommunication wires increases the operation speed of the circuits to several GHz and raises the probability of occurrence of intermittent faults at the time of operation of the circuit. The standard off-line testing techniques like Automatic Test Equipment (ATE) based testing and Built In Self Test (BIST) cannot detect such intermittent faults. Therefore, concurrent testing plays an essential part in testing of integrated circuits [1, 10, 23]. The concurrent testing can be defined as a mechanism which allows the circuit to verify the correctness of its functionality during normal operation by constantly checking whether the responses of the circuit match with the golden responses (responses of the fault free circuit). There are four different type of techniques that are used for concurrent testing of integrated circuits [18, 23, 24, 30]. These are –

(a) Signature monitoring
(b) Self-checking design
(c) On-line Built In Self Test
(d) Partial replication

Concurrent testing techniques have been emphasized to keep the schemes as non-intrusive as possible (i.e., least changes to the original structure of Circuit Under Test (CUT)), totally self-checking, low area and power overheads, low fault detection latency, high fault coverage, provides flexibility, scalability, etc. It has been found that most of the concurrent testing schemes work on classical single stuck at fault model [5, 6] and the single stuck at fault model cannot capture a large fraction of physical defects. In order to capture more number of physical defects, advanced fault models, such as bridging fault model, delay fault model, etc., are developed [17, 28]. In this work, logical AND-OR bridging fault model have been considered where the shorts between any two wires can be modeled using logical AND or logical OR operations. The bridging faults are categorized into two main classes. These are non-feedback bridging faults and feedback bridging faults. A short between two wires results a non-feedback bridging fault when these wires do not depend on one another, i.e., there does not exist any path between the shorted wires. On the other hand, feedback bridging fault occurs when the shorted wires depend on one another, i.e., there exists at least one path between the shorted wires. The presence of a feedback bridging fault in a combinational circuit transforms it into a sequential circuit and the circuit may oscillate along the feedback loop. Thus, it is comparatively difficult to test a feedback bridging fault than an non-feedback bridging fault.

The concurrent testing scheme for bridging faults reported in papers [8] can detect only non-feedback bridging faults. They have not mentioned any methodology to detect feedback bridging faults. In [3], the authors have proposed an concurrent testing scheme for bridging faults. This scheme can detect the non-feedback bridging faults and non-oscillating feedback bridging faults successfully. It has been shown that the procedure used for detection of non-oscillating feedback bridging faults is almost same as the non-feedback bridging faults with some minor modification in the proposed algorithms. In concurrent testing, an on-chip tester circuit is designed using the set of test patterns for targeted faults in the CUT and the tester circuit runs parallel with the CUT and detects the occurrence of faults in the CUT during normal operation. Providing flexibility in design of tester circuit is one of the important criteria to be included in concurrent testing of present-day integrated circuits. This work aims at designing a flexible concurrent testing scheme for feedback and non-feedback bridging faults in integrated circuits. Since the tester circuit runs in parallel with the CUT by means of tapping all wires of the CUT, thus dropping some of tap wires reduces load on the CUT which in turn minimizes the number of additional buffers required for driving gates with high fanouts. So dropping some of tap wires reduces the area overhead of the tester circuit. However, dropping some of tap wires to the tester circuit causes minor changes in fault coverage and fault detection latency. Thus, flexibility in tester circuit design can be provided by the concept of dropping of some tap wires to the tester circuit. Further, the concept of dropping of some tap wires to the tester circuit (flexibility) enables the scheme to perform trade-off analysis between area overhead, fault coverage and fault detection latency. The proposed scheme follows the design principle of *partial replication* technique to perform concurrent testing. The foremost feature of the proposed scheme is exclusive use of Reduced Ordered Binary Decision Diagrams (ROBDDs) for generation of test sets, which improves the scalability to handle relatively large size circuits.

The rest of the paper is organized as follows. Literature review on concurrent testing of integrated circuits followed by motivation of the work is discussed in Sect. 2. The Finite Automate (FA) based modeling of circuit under normal and faulty conditions, generation of test patterns (called Fault Identification transitions ($FI-transitions$)) is explained in Sect. 3. Partitioning the CUT into a number of sub circuits based on the principle of cones of influences is discussed in Sect. 4. Procedure of generation of $FI-transitions$ for non-feedback and feedback bridging faults using ROBDD is illustrated in Sect. 5. Flexible tester circuit design using set of $FI-transitions$ is discussed in Sect. 6. Experimental results regarding area overhead, fault coverage and fault detection latency under dropping of some of the tap wires is illustrated in Sect. 7. Finally, conclusion and future scope of the research is discussed in Sect. 8.

2 Literature Review and Motivation of the Work

This section starts with discussion on different types of concurrent testing techniques for integrated circuits followed by motivation of the present work is built up.

The *signature monitoring* technique for concurrent testing fundamentally works on circuit modeling using finite state based model and analyses the signature invariant property during normal operation of the circuit [15, 25, 27]. The signature is represented as the state sequences traversed in the finite state model during execution of the circuit. The basic idea of this technique is runtime signature of fault free circuit is not similar to that of faulty circuit. It has been seen that signature invariant property may not hold for some circuits. In such cases the circuit structure is altered in order to satisfy the signature invariant property. Hence, the

signature monitoring technique has come under the intrusive methodology. Since the technique requires to alter and re-synthesis of the circuit which is not always acceptable in concurrent testing, so it has limited applicability. Further, the state explosion problem in finite sate based model makes the technique limited to circuits having typically about one hundred states.

The *self checking design* technique for concurrent testing makes use of error detection codes to perform concurrent testing. The working principle of this technique is the output of the circuit is encoded using one of the error detection codes and the tester circuit checks the coded output. If the coded output is a valid code word of the chosen error detecting code, then the circuit is fault free, otherwise the circuit is faulty [9, 11, 12]. In most of the cases, parity codes, berger codes, m-out-of-n codes, etc., are used as error detecting codes. The main disadvantages of self checking design based concurrent testing schemes are intrusiveness and high hardware overhead as output is always encoded with an error detecting code.

The *on-line BIST* technique for concurrent testing utilizes the on-chip resources of off-line testing (BIST) to perform concurrent testing. In off-line BIST, circuit is designed with additional on-chip circuitry that is used to test the CUT every time before it is started-up for normal operation. In on-line BIST, idle time of different components of the circuit is determined during normal operation and concurrent testing is performed during this time interval [2, 21, 29]. There are a number of issues in on-line BIST schemes such as availability of idle time of different components of the circuit, fitting test time within idle time available, test schedule, etc. Further, the present day circuits target to achieve pipelining and parallelism, which reduce the idle times of their components (i.e, high utilization of their components). Therefore, this technique cannot be considered as an efficient technique for concurrent testing.

The *partial replication* technique for concurrent testing designs a partial replicated circuit (i.e., minimized version of the original circuit) and cross checks the output responses of original circuit and partial replicated circuit during normal operation. If the output responses are not identical then the circuit is faulty, otherwise the circuit is fault free [4, 5, 13, 14]. In this technique, the exhaustive set of test patterns for all targeted faults of the CUT are generated using one of the ATPG algorithms, then a subset of these test patterns are taken to design the partial replication circuit, which is executed in parallel with the CUT. The advantages of partial replication based concurrent testing scheme are non-intrusiveness, i.e., minimal changes in the original structure of the circuit for concurrent testing, low area overhead, high fault coverage, low fault detection latency, etc. Rayudu et al. [26] have proposed an off-line testing scheme to detect toggling faults, bridge faults and stuck at faults in both combinational and sequential circuits. In this work, they have used ROBDD based designs of the circuit to detect such faults.

The concurrent testing schemes reported in papers [3, 5, 8] are based on partial replication technique where they have targeted to achieve low area overhead, high fault coverage, low fault detection latency, etc. The work reported in [5] has used classical stuck at fault model. Although the scheme has achieved more than 95% fault coverage, the use of single stuck-at-fault model cannot capture a large fraction of physical defects. For example, the shorts between two wires cannot be modeled using single stuck at fault model. For this reason, advanced fault models like bridging fault model, delay fault model, etc., have been developed to capture a large number of physical defects in the present-day integrated circuits. The bridging fault model is one of the advanced fault models which can capture the faults that are occurred due to short of any two wires of the circuit. For a circuit having k number of wires, the total number of single stuck-at-faults is $2k$ ($O(k)$) since each wire can have stuck-at-0 and stuck-at-1 faults. In case of bridging fault model, if the circuit having k number of wires, then the total number of bridging faults (taking short between any two wires at a time) is $O({}^{k}C_{2}) = O(k^2)$. Thus, adopting bridging fault model one can ensure that more number of faults or defects can be detected than that of single stuck-at fault model [7, 16, 19]. The concurrent testing schemes reported in papers [3, 8] use bridging fault model to detect faults during normal operation of the circuit. The scheme reported in [8] can only detect the occurrence of any non-feedback bridging faults in the CUT, whereas they have not considered the feedback bridging faults. Thus, this scheme has low fault coverage. The scheme reported in [3] can detect the non-feedback and non-oscillating feedback bridging faults successfully. This scheme has improved fault coverage to some extend by considering non-oscillating feedback bridging faults. The above schemes [3, 8] have designed the tester circuits by tapping all the wires of the circuit. These schemes don't have any concern regarding flexibility in design of the tester circuits. However, providing flexibility in design of tester circuit is one of the important criteria to be included in concurrent testing of integrated circuits designed using deep sub-micron technology. In order to provide flexibility in design of tester circuit, some of the tap wires to the tester circuit can be dropped which reduces the number of fanout of the gates of the circuit, thus area overhead is reduced. Further, dropping of some tap wires to the tester circuit makes minor changes on fault coverage and fault detection latency. Thus, through flexibility (dropping some of tap wires) in tester circuit design, trades-off analysis between area overhead, fault coverage and fault detection latency can be explored.

From the above discussion, this work aims at designing a flexible concurrent testing scheme for feedback and

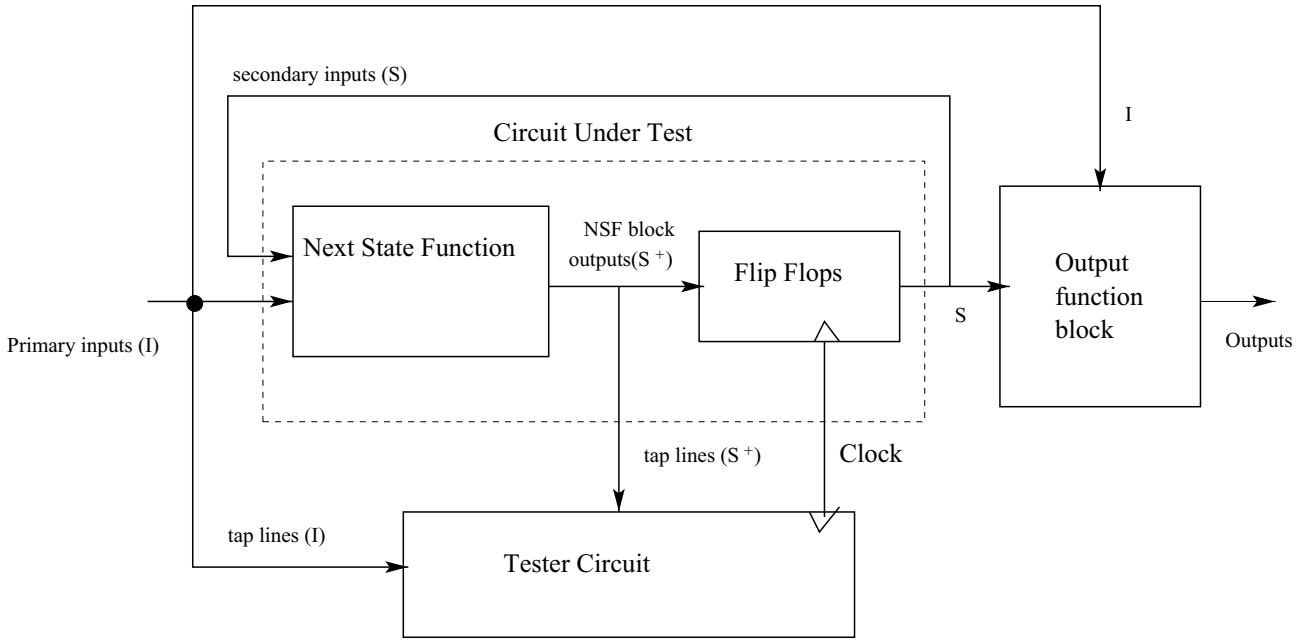


Fig. 1 Architecture of concurrent testing of a sequential circuit

non-feedback bridging faults in digital integrated circuits. The proposed scheme follows the principle of *partial replication* technique for concurrent testing in order to attain the advantages such as non-intrusiveness, low area overhead, high fault coverage, low detection latency, etc. The scheme applies flexibility in the tester circuit design by dropping some of the tap wires to the tester circuit and studies the trades-off between area overhead verses fault coverage and fault detection latency. Further, the scheme uses ROBDDs for generation of test patterns, which directly improves the scalability of the scheme and handles large size circuits.

3 Finite Automata (FA) Based Circuit Modeling Under Normal and Faulty Conditions and Generation of Fault Identification Transitions

This section includes 2 subsections; (a) Sequential circuit modeling under normal and faulty conditions using finite automata, and (b) generation of fault identification transitions. A digital sequential circuit consists of Next State Function (NSF) block, set of Flip-Flops (FFs), and Output Function (OF) block. In this work, concurrent testing is performed in NSF block and set of FFs of the sequential circuit and the OF block is not included because it is a combinational circuit and the same mechanism can be easily applied to it. The fundamental architecture of concurrent testing of sequential circuit is shown in Fig. 1. Circuit Under Test (CUT) is made up of combination of NSF block and set of FFs of the sequential circuit. The CUT and the tester circuit are driven by same clock and the tester circuit is devised by means of tapping of wires; NSF outputs (S^+) and primary inputs (I), as depicted in Fig. 1.

The CUT can be modeled using a finite automata (FA) M as follows:

$$M = \langle Q, \Sigma, q_0, \delta, V \rangle \tag{1}$$

where, Q is set of states (finite), Σ is set of input symbols (finite), $q_0 \in Q$ is the initial state, $\delta : Q \times \Sigma \rightarrow Q$ is the transition function, and $V = \{v_1, v_2, ..., v_m\}$ is set of Boolean variables. The set V is partitioned into two (2) subsets; set of state variables ($S = \{v_1, v_2, ...v_k\}$) and set of input variables ($I = \{v_{k+1}, v_{k+2}, ...v_m\}$). A state $q \in Q$ is a mapping $q : S \rightarrow \{0, 1\}$, i.e., state encoding is performed using a binary k-tuples. An input symbol $\sigma \in \Sigma$ is a mapping $\sigma : I \rightarrow \{0, 1\}$. A transition $\delta(q, \sigma) = q^+$ is represented as $\delta = \langle q, \sigma, q^+ \rangle$, where q, q^+, and σ are the initial state (denoted as $begin(\delta)$), final state (denoted as $end(\delta)$), and input symbol (denoted as $input(\delta)$) of the transition, respectively. Modeling digital circuits using FA framework, the set of final states doesn't have any significant role, so set of final states in FA model is not defined here.

3.1 CUT Modeling Under Normal and Bridging Fault Conditions

This subsection illustrates the process of modeling of CUT under normal and bridging fault conditions using Finite Automata (FA) framework. In bridging faults, two or more

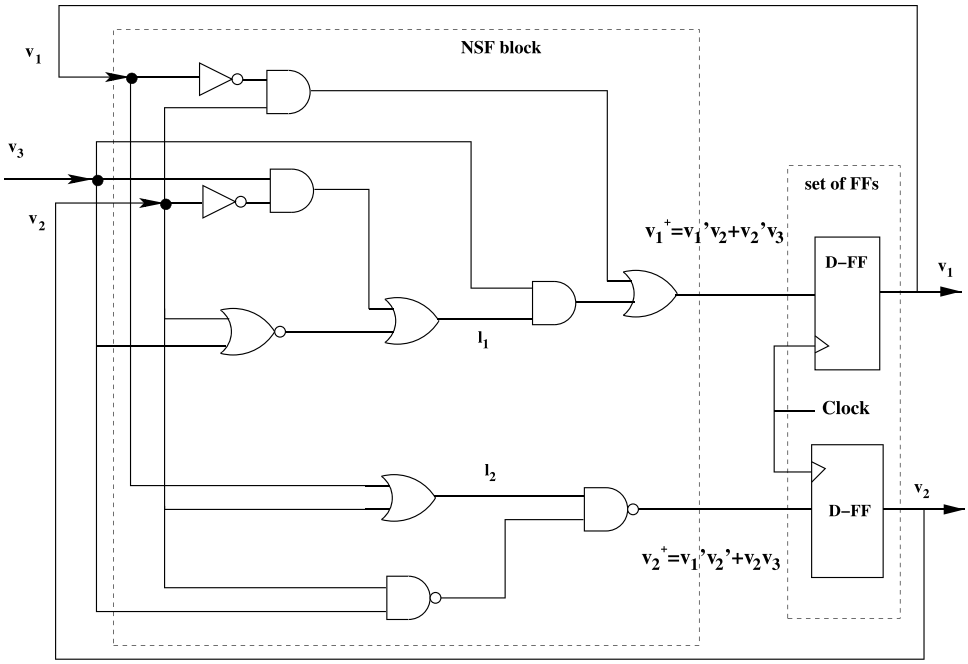


Fig. 2 Example: A simple sequential circuit (CUT)

wires of the circuit are shorted together. For simplicity, bridging faults involve shorting of any two wires of the circuit is considered in this work. There are two kinds of bridging faults; OR-bridging faults and AND-bridging faults. The OR-bridging fault (AND-bridging fault) between two wires implies they are shorted together to form logical OR (AND) operation. The bridging faults are categorized into two different types; (a) Non-feedback bridging faults and (b) Feedback bridging faults. In case of non-feedback bridging fault, there doesn't exist any path between two shorted wires. On the other hand, in case of feedback bridging fault there must be exist at least one path between two shorted wires.

In order to demonstrate the procedure of modeling of CUT under normal and bridging fault conditions, a simple sequential circuit (CUT) as shown in Fig. 2 is taken. Consider the OR-Bridging fault between the wires l_1 and l_2 (say F_1) and the faulty circuit is depicted in Fig. 3. Here, the CUT has 3 inputs where v_1 and v_2 are secondary inputs and v_3 is primary input. Under normal condition, the Boolean expressions for NSF block outputs are $v_1^+ = v_1^{'}v_2 + v_2^{'}v_3$ and $v_2^+ = v_1^{'}v_2^{'} + v_2v_3$ (shown in Fig. 2). Under faulty condition, the Boolean expressions for NSF block outputs are changed as $v_1^+ = v_1^{'}v_2 + v_3$ and $v_2^+ = v_2v_3$ (shown in Fig. 3). FA is considered as a well accepted model for modeling digital circuits because of it's simplicity [22]. Figure 4 shows the FA model for representing the behavior of the CUT under normal and faulty (F_1) conditions. It consists of two sub-models; normal sub-model (left hand side of the figure) and faulty sub-model (right hand side of the figure). For each fault there is a corresponding FA sub-model to represent the faulty behavior of the CUT. For simplicity, two sub-models have been shown; one for normal condition and another for faulty (F_1) condition. In normal sub-model (F_1 sub-model), the states are named as q_{0r} (q_{1r}), where $r >= 1$. In similar way, transitions are named in normal (as $\delta_{0k}, k >= 1$) and F_1 (as $\delta_{1k}, k >= 1$) sub-models, which can be found in Fig. 4. In order to distinguish between states under normal and faulty sub-models, an unmeasurable set of status variables $C = \{Nr, F_1, F_2, ..., F_h\}$ have been added, where Nr represents normal status, $F_i, 1 <= i <= h$, represents i^{th} fault status, and h is the total number of targeted faults in the CUT. It can be noted that the status variables are dummy (unmeasurable) variables and they are used for modeling purposes only. If they are measurable, then fault detection process becomes straightforward. The occurrence of fault (F_1) during normal operation of the circuit is modeled by moving the control of execution from normal sub-model to F_1 sub-model by the dotted transition shown in Fig. 4. The dotted transition is in the form of $\langle q_{0r}, T, q_{1r} \rangle$, where $C(q_{0r}) = Nr$, $C(q_{1r}) = F_1$, and T indicates *TRUE*. That means, these transitions are occurred at any time asynchronously without triggering edge of the clock.

Now, some definitions that are related to FA modeling of CUT under normal and faulty conditions are defined. Consider the followings:- M is the FA-model of CUT, and F_i is the bridging fault in the CUT.

Definition 1 Normal (Nr)**-state and Faulty** (F_i)**-state:** A state in M is called a *Nr-state* if it is denoted as q_{0r}, where $r >= 1$ and $C(q_{0r}) = Nr$.

A state in M is called a F_i *-state* if it is denoted as q_{ir}, where $r >= 1$ and $C(q_{ir}) = F_i$.

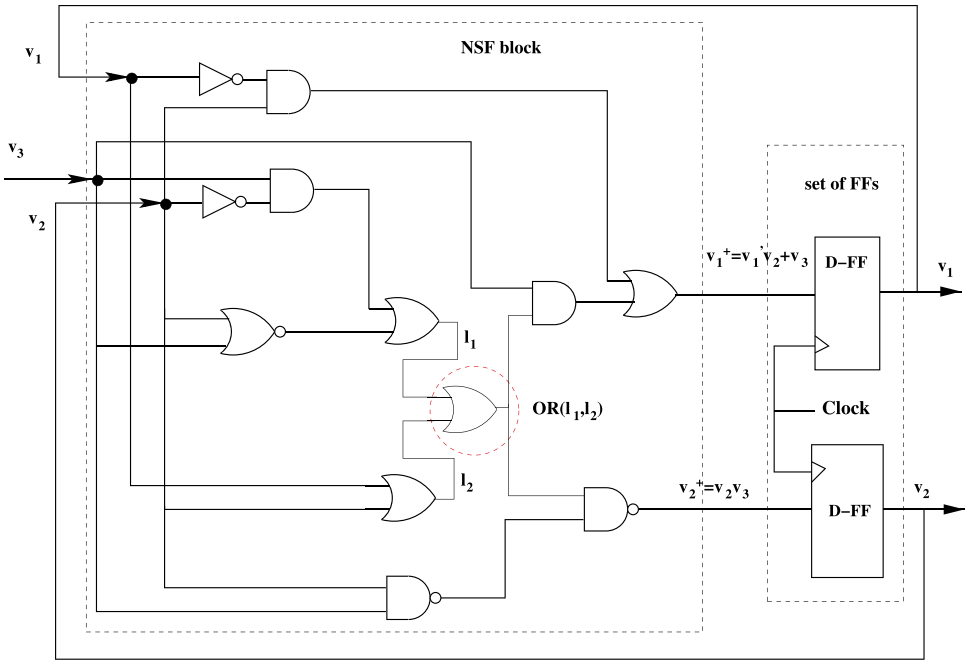


Fig. 3 CUT with OR-Bridging (l_1, l_2)

The set of all *Nr*-states (F_i-states) is denoted as Q_{Nr} (Q_{F_i}).

In this example (Fig. 4), the states q_{01}, q_{02}, q_{03}, and q_{04} are normal states and the states q_{11}, q_{12}, q_{13}, and q_{14} are faulty (F_1) states.

Definition 2 Normal(*Nr*)-*M*-**transition and Faulty** (F_i)-*M*-**transition:** A *M*-transition $\langle q, \sigma, q^+ \rangle$ is called a *Nr-M -transition*, if $\{q, q^+\} \in Q_{Nr}$.

A *M*-transition $\langle q, \sigma, q^+ \rangle$ is called a F_i-*M-transition*, if $\{q, q^+\} \in Q_{F_i}$.

In this example (Fig. 4), the transitions $\delta_{01}, \delta_{02}, ...\delta_{08}$ are normal (*Nr*)-*M*-transitions and the transitions $\delta_{11}, \delta_{12}, ...\delta_{18}$ are faulty (F_1)-*M*-transitions.

Definition 3 Symmetrical States: Two states q_i and q_j are said to be *symmetrical*, denoted as $q_i \approx q_j$, if $q_i|_S = q_j|_S$, where $q_i|_S$ represents the values of the state variable at state q_i.

It is very clear that any two sates of a given FA sub-model never be symmetrical. However, two states from different FA sub-models may be symmetrical. In this example (Fig. 4), the states q_{01} (in normal FA sub-model) and q_{11} (in faulty (F_1) FA sub-model) are symmetrical because of $q_{01}|_S = q_{11}|_S (= 00)$.

Definition 4 Symmetrical Transitions: Two transitions $\delta_i(\langle q_1, \sigma_1, q_1^+ \rangle)$ and $\delta_j(\langle q_2, \sigma_2, q_2^+ \rangle)$ are *symmetrical*, denoted as $\delta_i \approx \delta_j$, if $q_1|_S = q_2|_S$ (i.e., $q_1 \approx q_2$), $q_1^+|_S = q_2^+|_S$ (i.e., $q_1^+ \approx q_2^+$) and $\sigma_1|_I = \sigma_2|_I$.

In this example (Fig. 4), the transitions $\delta_{03}(\langle q_{02}, 0, q_{03} \rangle)$ and $\delta_{13}(\langle q_{12}, 0, q_{13} \rangle)$ are symmetrical ($\delta_{03} \approx \delta_{13}$) because of $q_{02} \approx q_{12}, q_{12} \approx q_{13}$, and $\sigma_1|_I = \sigma_2|_I (= 0)$.

3.2 Generation of Fault Identification Transitions (*FI − transitions*)

This subsection presents the process of generation of test patterns to detect the occurrence of a bridging fault in the CUT. Consider the CUT and OR-bridging fault between the wires l_1 and $l2$ (F_1) shown in the Fig. 3. The FA model of CUT under normal and F_1 conditions is shown in the Fig. 4. When one compares the transitions of normal FA sub-model with the transitions of faulty FA sub-model, then three transitions (marked in red color) are found in faulty sub-model that are differed from normal behavior. These transitions are– $\delta_{11} : \langle q_{11}, 0, q_{11} \rangle$, $\delta_{12} : \langle q_{11}, 1, q_{13} \rangle$, and $\delta_{18} : \langle q_{14}, 1, q_{14} \rangle$. For these transitions there is no *symmetrical transitions* in the normal FA sub-model. All other transitions in faulty sub-model there is a *symmetrical transition* in the normal sub-model. For example, in case of transition $\delta_{11} : \langle q_{11}, 0, q_{11} \rangle$, the corresponding transition in normal sub-model is $\delta_{01} : \langle q_{01}, 0, q_{02} \rangle$, where $q_{01}|_S = q_{11}|_S (= 00)$, $\sigma_{\delta_{01}} = \sigma_{\delta_{11}} (= 0)$ but $q_{02}|_S (= 01) \neq q_{11}|_S (= 00)$. That means, for a given state and input combination, the next states reached are different under normal and faulty conditions. Such type of transitions detect the occurrence of fault F_1 in the CUT and known as Fault Identification transition (*FI − transition*). The main significance of symmetrical transitions is to determine *FI − transitions* for a fault. In simple words, *FI − transitions* (in faulty FA sub-model) do not

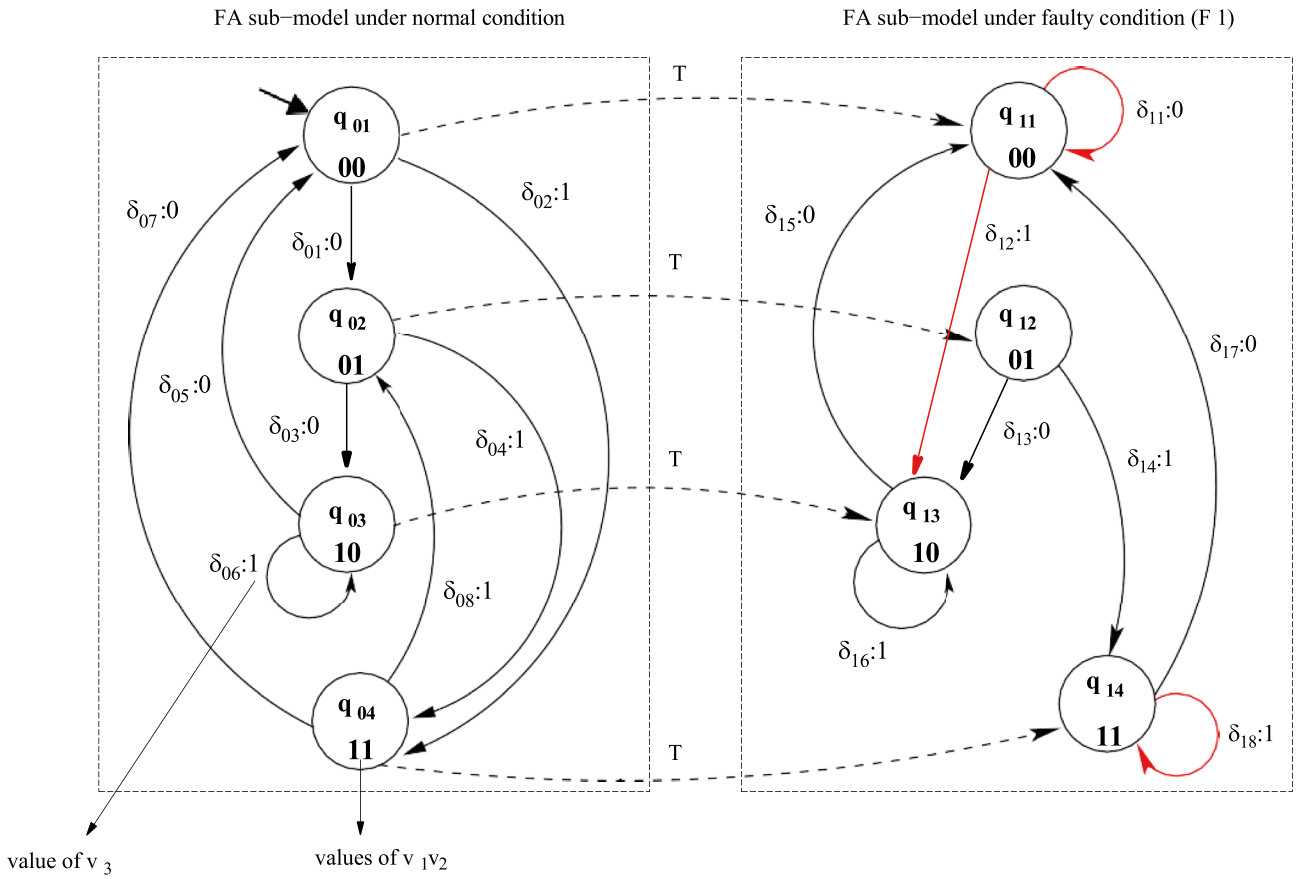


Fig. 4 FA-model of CUT under normal and faulty (F_1) conditions

have corresponding symmetrical transitions in normal FA sub-model. Based on above discussion, the $FI-transition$ can be defined as follows:

Definition 5 Fault Identification Transition ($FI-transition$): A faulty (F_i)-M-transition $\delta_{ir} = \langle q_{ir}, \sigma_{ir}, q_{ir}^{+} \rangle$ is a $FI-transition$ for fault F_i, if there is a normal (Nr)-M-transition $\delta_{0k} = \langle q_{0k}, \sigma_{0k}, q_{0k}^{+} \rangle$ such that $q_{ir} \approx q_{0k}, \sigma_{ir} = \sigma_{ok}$ and $q_{ir}^{+} \not\approx q_{ok}^{+}$.

The above procedure of generation fault identification transitions for a fault from finite automata based modeling framework is quite complex for large size circuits. In case of large size circuits the number of sates in FA model increases exponentially with the increase of the number of flip-flops in the circuit. Further, the number of state variables needed to encode the states in the FA model is also increased exponentially. This phenomenon is called *state explosion problem* in FA model, which restricts the technique to handle small sized circuits. In this paper, several techniques have been adopted in order handle comparatively large size circuits. First, partition the CUT into a number of sub-circuits based on the principle of *cones of influence* with respect to the NSF block outputs. Second, generation $FI-transitions$ for bridging fault using ROBDD representation of the cone outputs, instead of using explicit FA model of the CUT.

4 Partitioning the CUT Into a Number of Sub-Circuits

This section explains the procedure of partition of CUT into a number of sub-circuits based on the principle of cones of influences. Consider the CUT shown Fig. 2, where v_3 is the primary input (I) and v_1 and v_2 are the secondary inputs (S), which are feedback from the flip-flip outputs. The outputs of the NSF block (S^{+}) are v_1^{+} and v_2^{+}. $v_1 v_2$ and $v_1^{+} v_2^{+}$ represent the present state and next state of the CUT, respectively. Now, the NSF block can be formally defined as $S \times I \rightarrow S^{+}$, where $S = \{v_1, v_2, ...v_k\}$ be the set of state variables, $I = \{v_{k+1}, v_{k+2}, ...v_m\}$ be the set of input variables, and $S^{+} = \{v_1^{+}, v_2^{+}, ...v_k^{+}\}$ be the set of NSF block outputs. Let $S_0^{+} = \{v_{01}^{+}, v_{02}^{+}, ...v_{0k}^{+}\}$ and $S_i^{+} = \{v_{i1}^{+}, v_{i2}^{+}, ...v_{ik}^{+}\}$ represent the outputs of the NSF block under normal and faulty (F_i) conditions, respectively. Figure 5 shows the cones of influences with respect to the NSF block outputs of the CUT. It involves two cones; one is with respect to wire v_1^{+} and another is with respect to wire v_2^{+}. In this way, the circuit is

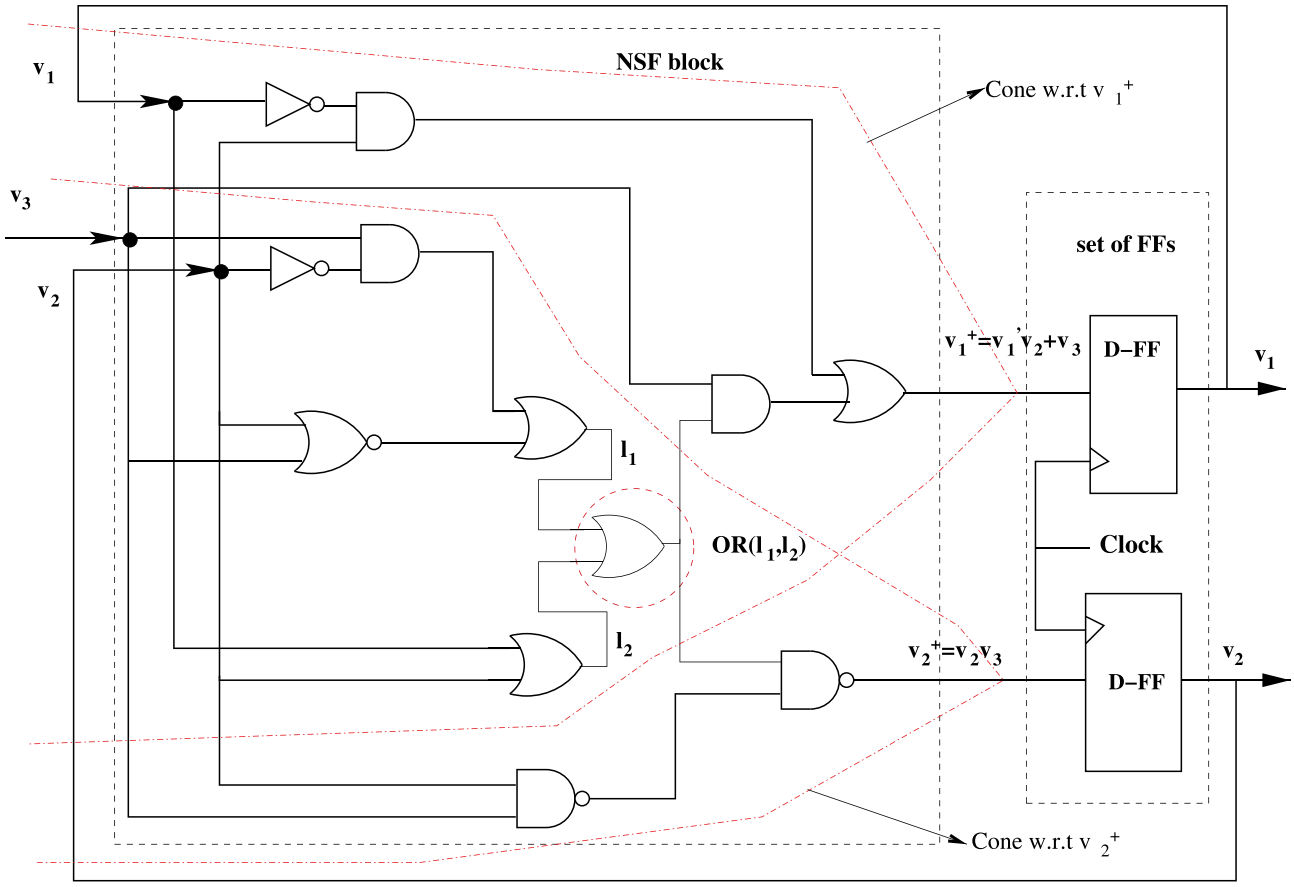


Fig. 5 OR-Bridging (l_1, l_2)

partitioned into a number of small sized sub-circuits using cones of influences, thus the scheme can handle comparatively large size circuits.

It can be observed in the Fig. 5 that the OR-Bridging fault between the wires l_1 and l_2 is active only when these two wires have different logic values, i.e.,when the values of $l_1 l_2$ is either 10 or 01. When both the values of l_1 and l_2 are logic 0 (logic 1), the output of the OR-gate is 0 (1), so the fault has no effect. The effect of OR-Bridging fault between the wires l_1 and l_2 is shown in Table 1. When $l_1 = 1$ and $l_2 = 0$, then l_2 becomes logic 1 due to OR-Bridging fault between l_1 and l_2. In case of OR-Bridging fault, the wire with logic value 1 (i.e., l_1) is called *dominating wire* and the wire with logic value 0 (i.e., l_2) is called *dominated wire*. In OR-Bridging fault, when the logic value at *dominating wire* becomes 1, then it overrides the value at the *dominated wire* by pulling it from logic 0 to logic 1. Thus, the OR-bridging fault results in *stuck-at-1* (s-a-1) fault at the *dominated wire* when the value at *dominating wire* is logic 1. Figures 6 and 7 show the effect of OR-Bridging fault between the wires l_1 and l_2, where l_1 dominates l_2 and l_2 dominates l_1, respectively. In order to detect OR-Bridging fault between the wires l_1 and l_2, the below mechanisms are followed.

1. l_1 dominates l_2: Find the test patterns that drives logic 1 to wire l_1 and detects *stuck-at-1* fault at wire l_2. Figure 6 illustrates the same.
2. l_2 **dominates** l_1: Find the test patterns that drives logic 1 to wire l_2 and detects *stuck-at-1* fault at wire l_1. Figure 7 illustrates the same.

It may be noted that the mechanism to detect OR-Bridging fault between two wires l_1 and l_2 can be directly used to detect AND-Bridging between l_1 and l_2 by applying the principle

Table 1 Effect of OR-Bridging between l_1 and l_2

Values of $l_1 l_2$	OR-Bridging (l_1, l_2)	Remarks
00	0	No difference
01	1	l_1 becomes 1
10	1	l_2 becomes 1
11	1	No difference

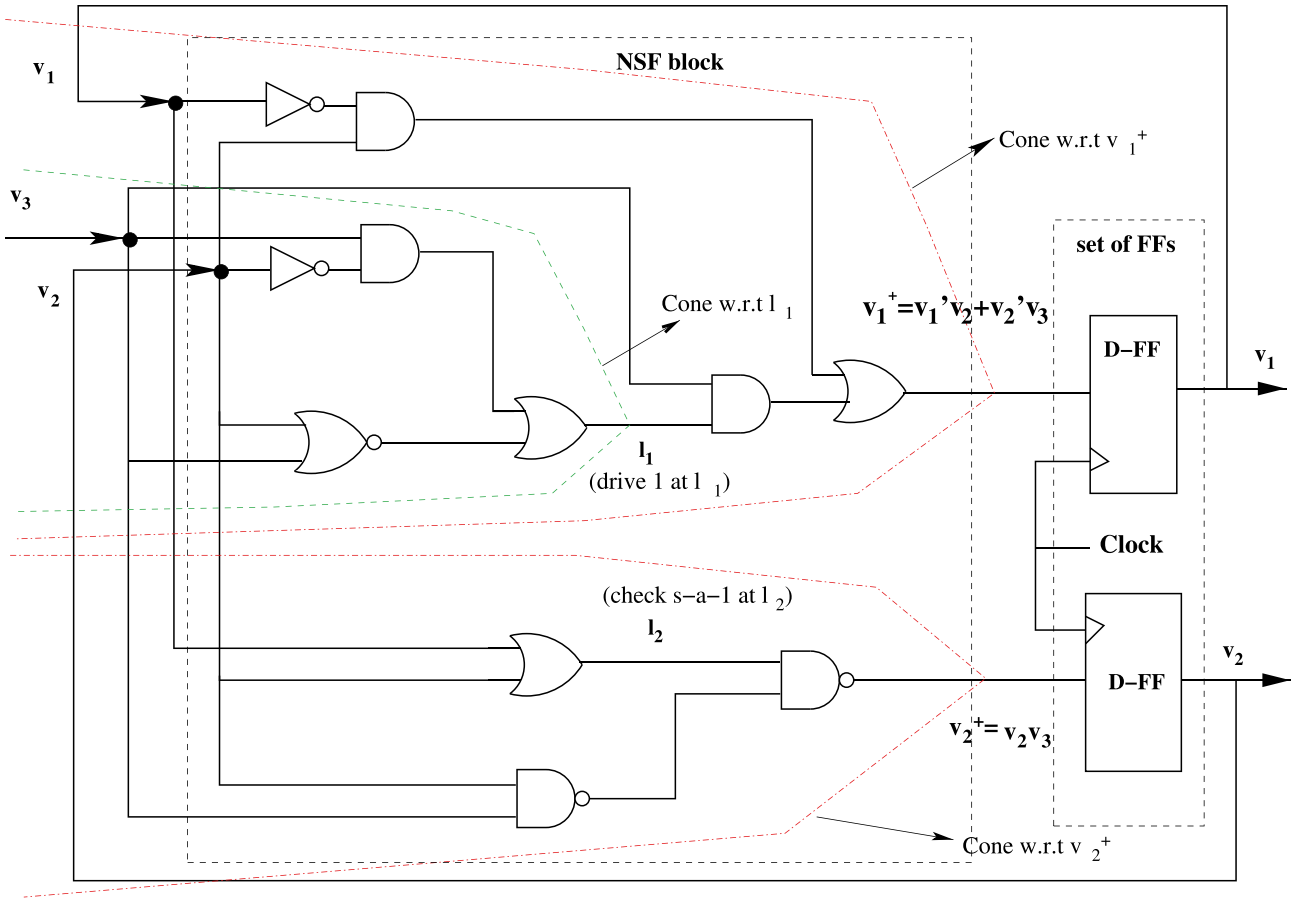


Fig. 6 OR-Bridging (l_1, l_2): $l1$ dominates $l2$

of duality. For example, in case of AND-Bridging the logic value at *dominating wire* is 0 and it makes the *dominated wire* wire from logic 1 to logic 0, i.e., to check stuck-at-0 fault at the *dominated wire* . In the next section, the procedures of generation of $FI-transitions$ for non-feedback and feedback bridging faults using ROBDD will be discussed.

5 Generation of $FI-transitions$ for Non-feedback and Feedback Bridging Faults Using Reduced Ordered Binary Decision Diagram (ROBDD)

5.1 Generation of $FI-transitions$ for Non-feedback Bridging Faults Using ROBDD

In this subsection, the procedure of generation of complete set of $FI-transitions$ for the non-feedback bridging faults using ROBDD will be discussed. For demonstrating the procedure, consider the example of CUT and OR-Bridging fault between the wires l_1 and l_2 (say, F_i), shown in Fig. 3. Since there is no path between the wires l_1 and l_2, so the bridging fault between the wires l_1 and l_2 is a non-feedback bridging fault. As per discussion in the last section the OR-Bridging fault between wires l_1 and l_2 can be detected using two mechanisms; (1) l_1 dominates l_2 and (2) l_2 dominates l_1.

5.1.1 Generation of $FI-transitions$ for Non-feedback Bridging Fault When l_1 Dominates l_2

At first, the NSF block is partitioned into a number of cones of influences with respect to the NSF block outputs, i.e., v_i^+, where $1 \leq i \leq k$. Consider non-feedback OR-Bridging fault between the wires l_1 and l_2 (F_i) and the mechanism where l_1 dominates l_2. In this case, fault can be detected by means of test patterns which drives logic 1 at wire l_1 and check *stuck-at-1* fault at wire l_2 (shown in Fig. 6). Since it is required to manifest the *stuck-at-1* fault at wire l_2 through the NSF block outputs, so the cones of influences with respect to NSF outputs which include the wire l_2 are found out first. Then, ROBDD based Algorithm 1 is used to generate $FI-transitions$ for non-feedback OR-bridging fault in the CUT.

Algorithm 1 Generation of $FI-transitions$ for non-feedback OR-bridging fault between wires l_1 and l_2 (F_i) when l_1 dominates l_2

1. Find out the cones of influences with respect to NSF outputs which include the wire l_2. Let $\mathbb{C}^+ \subseteq S^+$ be the set of such cones.

2. Generate the cone of influence with respect to *dominating wire* l_1 under normal condition. Let v_{l_1} be the Boolean expression at l_1 under normal condition. Build a ROBDD which represents the Boolean expression of v_{l_1}. Say it $robdd_{dominating}$.

3. Let TP be the exhaustive set of test patterns to detect F_i.

Initially $TP \leftarrow \phi$

4. for *each cone w.r.t* $v_j^+ \in \mathbb{C}^+$ **do**

4.1 Build a ROBDD which represents the Boolean expression of v_j^+ under normal condition. Say it $robdd_{normal}$.

4.2 Build a ROBDD which represents the Boolean expression of v_j^+ under *stuck-at-1* fault at wire l_2. Say it $robdd_{faulty}$.

4.3 Perform XOR operation between normal and faulty ROBDDs, i.e., $robdd_{xor} = robdd_{normal} \oplus robdd_{faulty}$.

4.4 Apply *satisfy-all-1* operation on $robdd_{xor}$, which generates set of input patterns that results different values of v_j^+ under normal and faulty (*stuck-at-1* fault at l_2) conditions. Let $IP_{l_2,s-a-1}$ be the such set of input patterns.

4.5 Apply *satisfy-all-1* operation on $robdd_{dominating}$, which generates set of input patterns that drive logic 1 at dominating wire l_1. Let $IP_{l_1,1}$ be the such set of input patterns.

4.6 The set of test patterns to detect F_i and fault is manifested at v_j^+ is obtained as $TP_{v_j^+} = IP_{l_1,1} \cap IP_{l_2,s-a-1}$, i.e., these test patterns drives logic 1 at l_1 and detects *stuck-at-1* fault at l_2.

4.7 Required complete set of test patterns to detect F_i is obtained as $TP = TP + TP_{v_j^+}$.

end

5 for *each test pattern* $tp \in TP$ **do**

5.1 Assume the test pattern tp is generated through the cone v_j^+.

5.2 Generate $FI-transition$ from the tp. Let tp be the values of the set of variables $\{v_1, v_2, ...v_k, v_{k+1}, v_{k+2}, ...v_m\}$ and $\delta_{ir} = \langle q_{ir}, \sigma_{ir}, q_{ir}^+ \rangle$ be it's corresponding $FI-Transistion$, then q_{ir} is the values of the set of variables $\{v_1, v_2, ...v_k\}$ and σ_{ir} is the values of the set of variables $\{v_{k+1}, v_{k+2}, ...v_m\}$ and q_{ir}^+ contains value of v_j^+ obtained by applying tp in faulty ROBDD ($robdd_{faulty}$) and rest of state variables have don't care ($\times$) values.

end

Now, the Algorithm 1 is applied to generate *FI–Transistions* for non-feedback bridging fault between wires l_1 and l_2 (F_i) shown in the Fig. 6. Follow the mechanism of l_1 dominates l_2, i.e., test patterns are generated by driving logic 1 to wire l_1 and checking *stuck-at-1* fault at wire l_2. The wire l_2 is only present in the cone w.r.t v_2^+. Figure 8(a), (b) show the ROBDD representations of cone w.r.t v_2^+ under normal ($robdd_{normal}$) and faulty ($robdd_{faulty}$) conditions, respectively. The Boolean expressions 4 $v_2^+ = v_1'v_2' + v_2v_3$ and $v_2^+ = v_2v_3$, respectively. The logical XOR operation is performed between the normal and faulty ROBDDs and the XORed ROBDD ($robdd_{xor}$)is shown in Fig. 8(c). The *satisfy-all-1* operation is applied on $robdd_{xor}$ to generate input patterns ($\langle v_1v_2v_3 \rangle$) as $IP_{l_2,s-a-1} = \{00\times\} = \{000, 001\}$. The ROBDD representation of cone w.r.t wire l_1 (Boolean expression $v_{l_1} = v_2'$) is shown in Fig. 8(d) ($robdd_{dominating}$). Further, the *satisfy-all-1* operation is applied on $robdd_{dominating}$ in order to generate test patterns ($\langle v_1v_2v_3 \rangle$) that drives logic 1 at wire l_1, i.e., $IP_{l_1,1} = \{\times0\times\} = \{000, 001, 100, 101\}$. Now, the intersection operation between $IP_{l_1,1}$ and $IP_{l_2,s-a-1}$ is performed to generate set of test patterns as $TP_{v_2^+} = \{000, 001\}$, i.e., drives logic 1 at wire l_1 and detects *stuck-at-1* at wire l_2

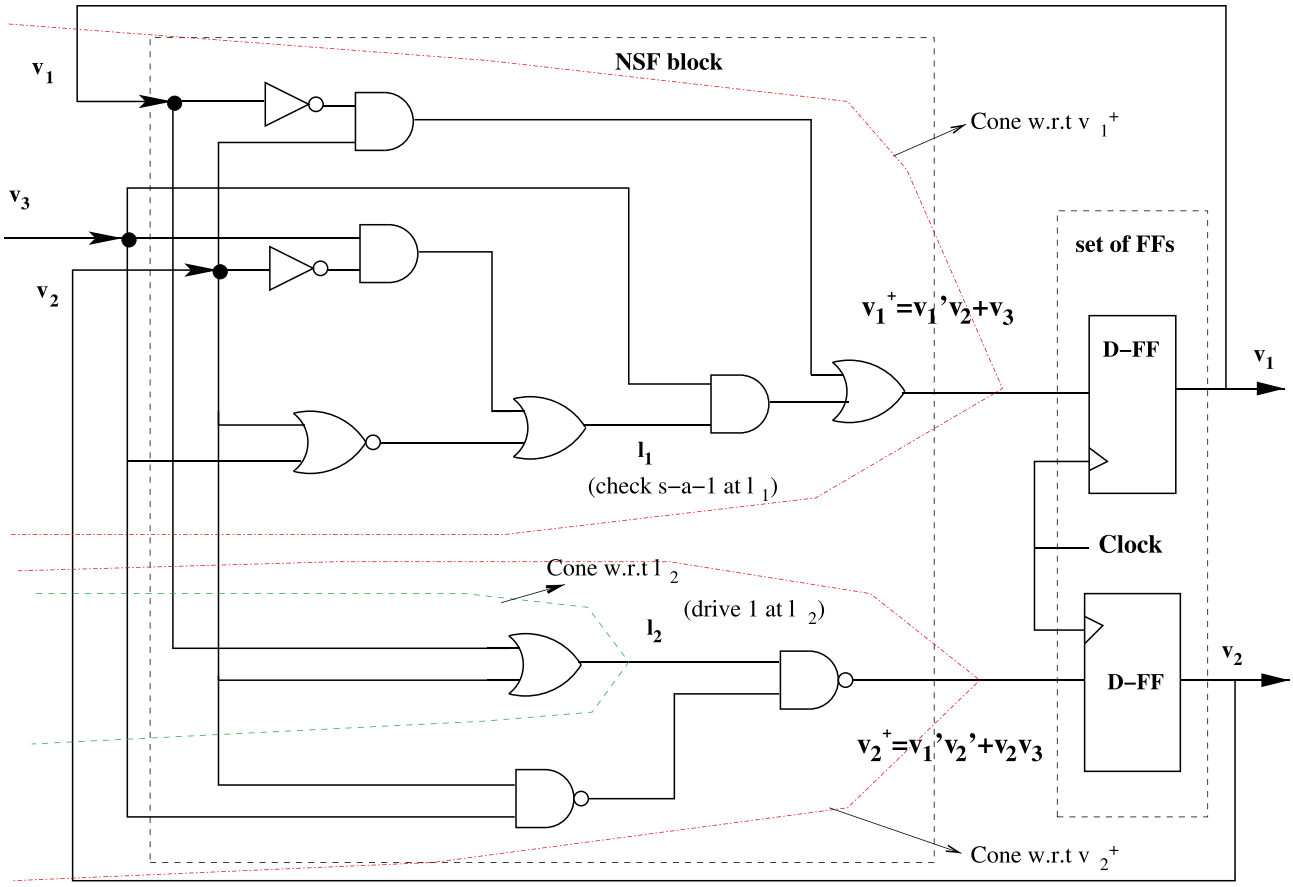


Fig. 7 OR-Bridging (l_1, l_2): $l2$ dominates $l1$

and fault is manifested through v_2^+. For test pattern $\langle 000 \rangle$, the corresponding $FI - transition$ is $\langle 00, 0, \times 0 \rangle$ where the faulty response of v_2^+ is obtained by applying test pattern $\langle 000 \rangle$ in $ROBDD_{faulty}$, i.e., logic 0 and the value of v_1^+ is $\times$ because fault is manifested through v_2^+. This is corresponding to the transition δ_{11} shown in Fig. 4. Similarly, for test pattern $\langle 001 \rangle$, the corresponding $FI - transition$ is $\langle 00, 1, \times 0 \rangle$, which corresponds to transition δ_{12} shown in Fig. 4.

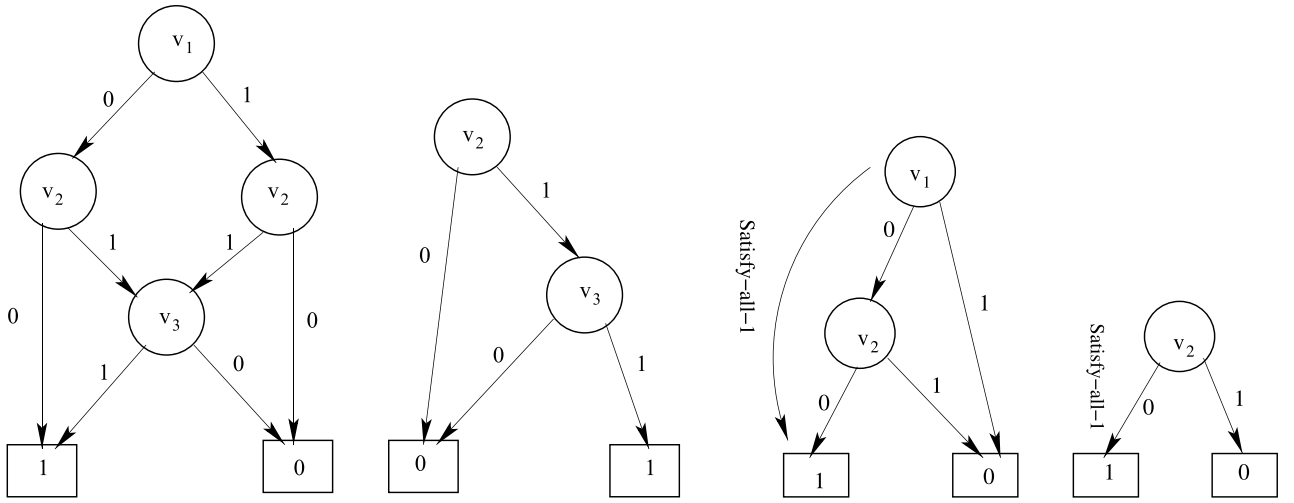
5.1.2 Generation of *FI − transitions* for Non-feedback Bridging Fault When l_2 Dominates l_1

The procedure discussed in Subsection 5.1.1 is repeated with reversing the roles of l_1 and l_2. Here, l_2 becomes dominating wire and l_1 becomes dominated wire. In this case, bridging between the wires l_1 and l_2 is detected by driving logic 1 to wire l_2 and detect *stuck-at-1* fault at wire l_1, which is shown in Fig. 7. The ROBDDs required to generate test patterns is shown in Fig. 9. In similar way, the set of test patterns are generated as $TP_{v_1^+} = \{111\}$ and it's corresponding $FI - transition$ is $\langle 11, 1, 1\times \rangle$, which corresponds to transition δ_{18} shown in Fig. 4.

Finally, the complete set of $FI - transitions$ generated for fault F_i is $\{\langle 00, 0, \times 0 \rangle, \langle 00, 1, \times 0 \rangle, \langle 11, 1, 1\times \rangle\}$.

5.2 Generation of *FI − transitions* for Feedback Bridging Fault Using ROBDD

This subsection presents the procedure of generation of complete set of $FI - transitions$ for feedback bridging faults using ROBDD. As discussed before, a bridging fault between the wires l_1 and l_2 is said to be feedback bridging fault when there exists at least one path between the wires l_1 and l_2. The most important thing in feedback bridging fault is *oscillation*. It has been seen that some of the feedback bridging faults cause oscillation in the CUT and termed as *oscillating feedback bridging fault*. Detection of such faults is quite impossible using standard logical fault models. Consider the feedback bridging fault between the wires l_1 and l_2 and the wire l_1 is closer towards the primary inputs and wire l_2 is closer towards the primary outputs. Since the two mechanisms have been used for generation of $FI - transitions$ for non-feedback bridging faults; (a) l_1 dominates l_2 and (b) l_2



(a) Normal ROBDD of cone w.r.t v_2^+.

(b) Faulty ROBDD of cone w.r.t v_2^+ (s-a-1 at l_2).

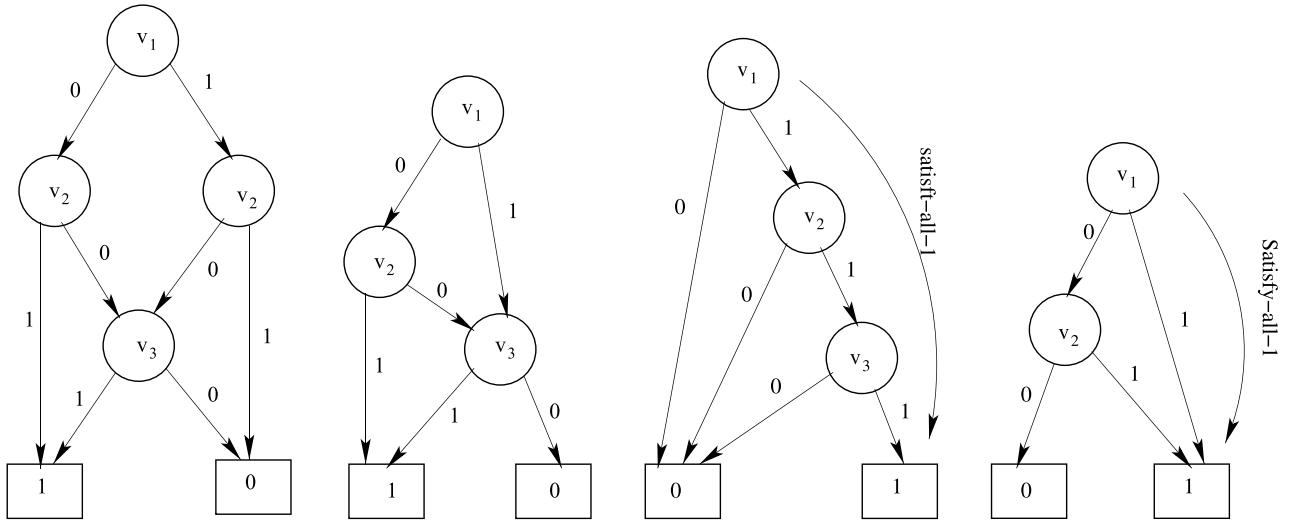
(c) XORed ROBDD

(d) ROBDD of cone w.r.t wire l_1.

Fig. 8 ROBDDS for l_1 dominate l_2

dominates l_1. The mechanism of l_1 **dominates** l_2 discussed in Subsection 5.1.1 is directly applied in order to generate $FI-transitions$ for feedback bridging faults. However, in case of l_2 **dominates** l_1 there is required to execute some additional steps for generation of $FI-transitions$ for feedback bridging faults. The execution of additional steps are essential in order to isolate test patterns that cause oscillations. Now, the mechanism of l_2 **dominates** l_1 for generation of $FI-transitions$ for feedback bridging faults will be discussed using the example of CUT shown in Fig. 10.

Consider the CUT and feedback bridging fault between the wires l_1 and l_2 as shown in Fig. 10. Since there exists a path between l_1 and l_2, so they must lie in the same cone of influence, i.e., cone w.r.t v_1^+. Apply Algorithm 1 with the mechanism l_2 *dominates* l_1 to generate $FI-transitions$ for this feedback bridging fault. ROBDD representations of cone w.r.t v_1^+ under normal and faulty (*stuck-at-1*) at wire l_1 are shown in Fig. 11(a) ($robdd_{normal}$) and (b) ($robdd_{faulty}$), respectively. The Boolean expressions of v_1^+ under normal and faulty conditions are $v_1^+ = v_1v_2'v_3 + v_1'v_2v_3$ and



(a) Normal ROBDD of cone w.r.t v_1^+.

(b) Faulty ROBDD of cone w.r.t v_1^+ (s-a-1 at l_1).

(c) XORed ROBDD

(d) ROBDD of cone w.r.t wire l_2.

Fig. 9 ROBDDS for l_2 dominate l_1

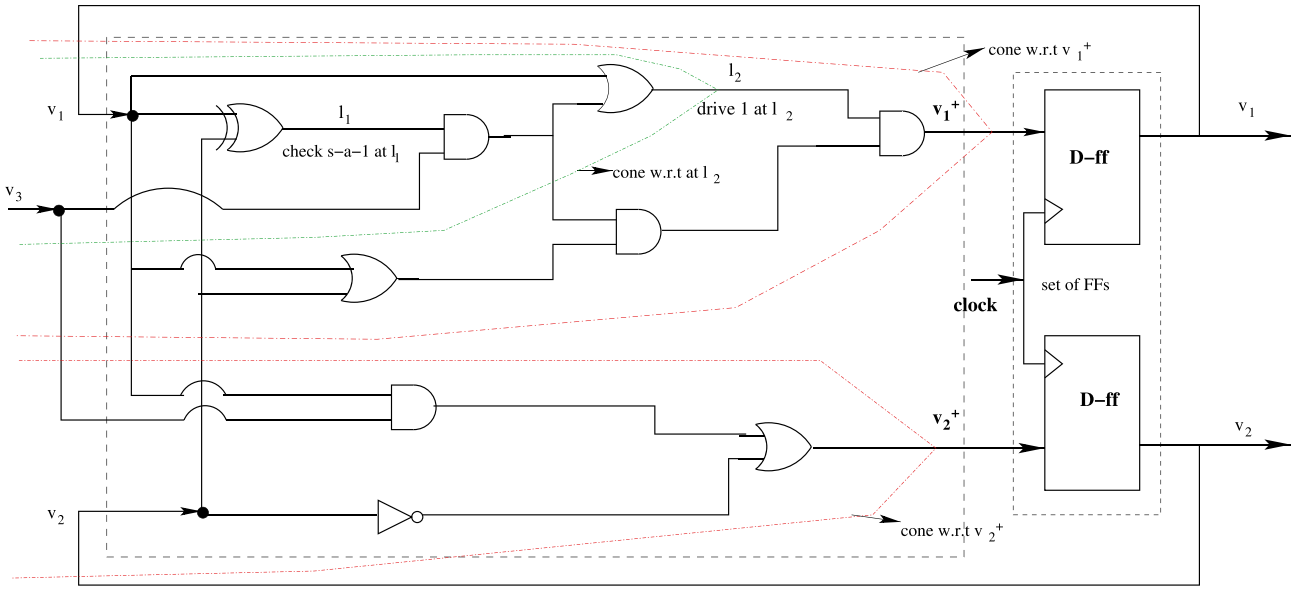


Fig. 10 Feedback bridging (l_1, l_2): $l2$ dominates $l1$

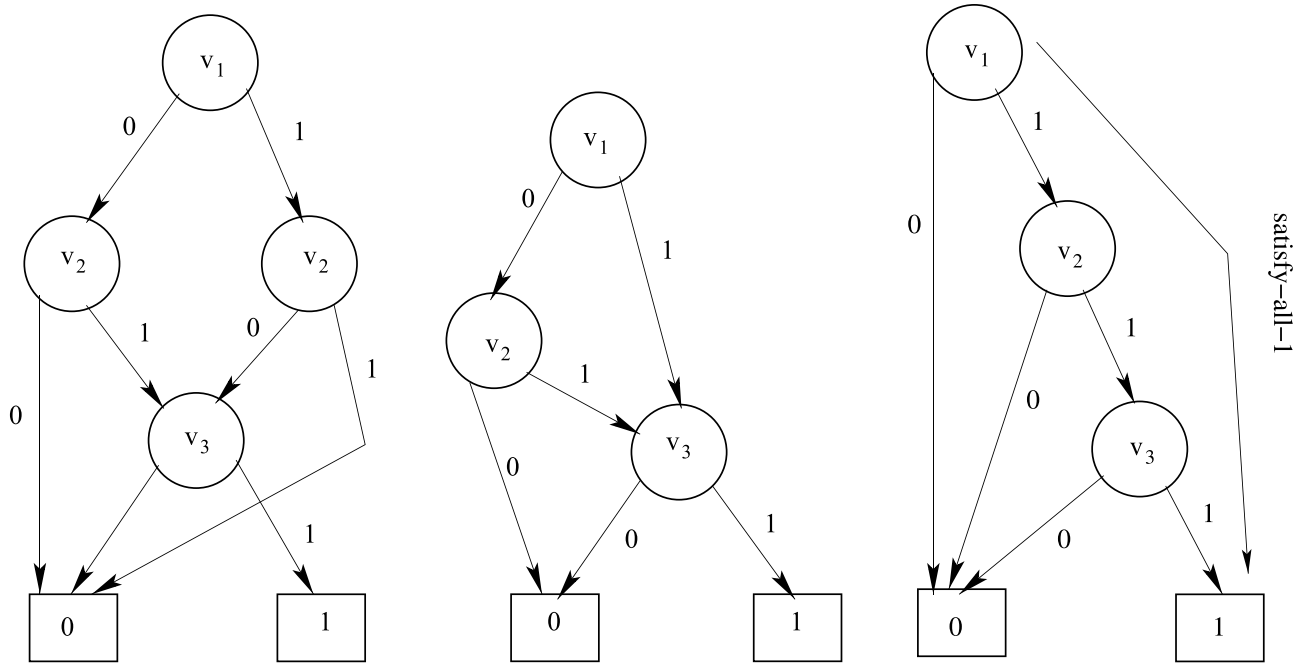
$v_1^+ = v_1v_3 + v_2v_3$, respectively. Perform logical XOR operation between $robdd_{normal}$ and $robdd_{faulty}$ to build XORed ROBDD ($robdd_{xor}$), which is depicted in Fig. 11(c). Apply *satisfy-all-1* on $robdd_{xor}$ to find input patterns ($\langle v_1v_2v_3\rangle$) to detect *stuck-at-1* fault at l_1. The obtained set of input patterns is $IP_{l_1,s-a-1} = \{111\}$. The ROBDD representation of cone w.r.t l_2 under normal condition (Boolean expression is $v_1 + v_2v_3$) is shown in Fig. 11(d) ($robdd_{dominating}$). *Satisfy-all-1* operation is applied to generate input patterns that drives logic 1 at wire l_2 under normal condition. The obtained set of input patterns is $IP_{l_2,1} = \{011, 1\times\times\} = \{011, 100, 101, 110, 111\}$. The intersection between $IP_{l_2,1}$ and $IP_{l_1,s-a-1}$ is performed to produce test patterns to detect the feedback bridging fault between l_1 and l_2. The obtained set of test patterns is $TP = \{111\}$. Now, the following additional steps are executed to find out whether this test pattern causes oscillation or not.

- Build a ROBDD which represents the cone w.r.t wire l_2 under *stuck-at-1* fault at wire l_1. Say it $robdd_{l_2,l_1(s-a-1)}$. In this example, the Boolean expression of l_2 under *stuck-at-1* at l_1 is $v_1 + v_3$ and $robdd_{l_2,l_1(s-a-1)}$ is shown in Fig. 11(e).
- Apply *satisfy-all-0* operation on $robdd_{l_2,l_1(s-a-1)}$ in order to generate input patterns that makes logic 0 at wire l_2 during propagation of *stuck-at-1* fault at l_1. Say it $IP_{l_2=0,l_1(s-a-1)}$. In this example, $IP_{l_2=0,l_1(s-a-1)} = \{0\times 0\} = \{000, 010\}$.
- For each test pattern $tp \in TP$ do the followings:
  - If $tp \in IP_{l_2=0,l_1(s-a-1)}$, then eliminate tp from TP, because it causes oscillation.
  - If $tp \notin IP_{l_2=0,l_1(s-a-1)}$, then it remains as a test pattern and does not cause oscillation. In this case tp is mapped to corresponding $FI - transttion$.

In this example, $TP = \{111\}$, $IP_{l_2=0,l_1(s-a-1)} = \{000, 010\}$ and $\langle 111\rangle \notin IP_{l_2=0,l_1(s-a-1)} = \{000, 010\}$. So, the test pattern $\langle 111\rangle$ remains as a test pattern and it is mapped to $FI - transition$ as $\langle 11, 1, 1\times\rangle$.

5.3 Illustration of Detection of Oscillating Feedback Bridging Fault

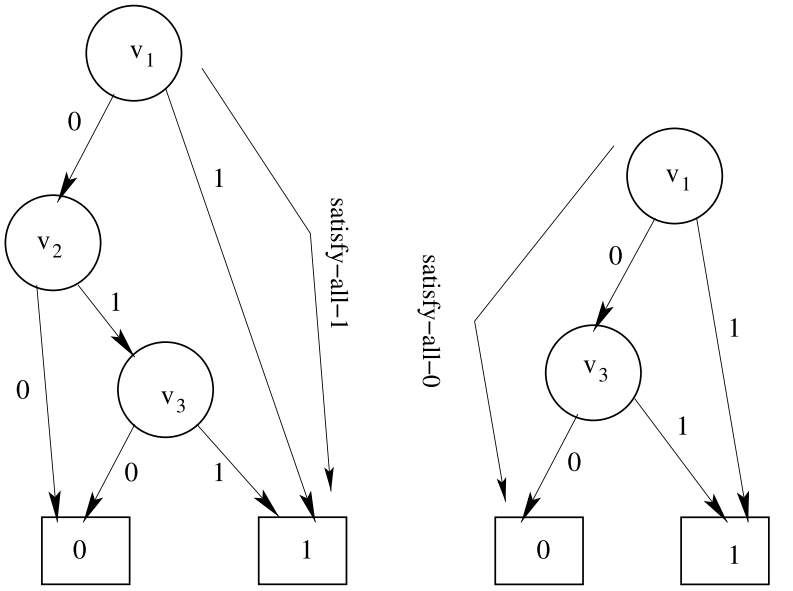
This subsection illustrates the process of detection of an oscillating feedback bridging fault. Consider the CUT and feedback bridging fault between wires l_1 and l_2 as shown in Fig. 12. Apply Algorithm 1 with the mechanism l_2 *dominates* l_1 to generate $FI - transitions$ for this feedback bridging fault. The Boolean expressions of v_1^+ under normal and faulty ($s - a - 1$ *at* l_1) conditions are $v_1^+ = v_1'v_2' + v_1'v_3' + v_2v_3$ and $v_1^+ = v_1'v_3' + v_2v_3$, respectively. ROBDD representation of cone w.r.t v_1^+ under normal and faulty conditions are shown in Fig. 13(a) ($robdd_{normal}$) and (b) ($robdd_{faulty}$), respectively. The XORed ROBDD ($robdd_{xor}$) is shown in Fig. 13(c). *satisfy-all-1* operation is performed on $robdd_{xor}$ to find input patterns ($\langle v_1v_2v_3\rangle$) to detect *stuck-at-1* fault at l_1. The obtained set of input patterns is $IP_{l_1,s-a-1} = \{001\}$. The ROBDD representation of cone w.r.t l_2 ($robdd_{dominating}$) under normal condition (Boolean expression is $v_1'v_2' + v_1'v_3'$) is shown in Fig. 13(d). Apply *Satisfy-all-1* operation to $robdd_{dominating}$ in order to find input patterns that drives logic 1 at wire l_2 under normal condition. The obtained set of input



(a) Normal ROBDD of cone w.r.t v_1^+.

(b) Faulty ROBDD of cone w.r.t v_1^+ (s-a-1 at l_1).

(c) XORed ROBDD



(d) ROBDD of cone w.r.t wire l_2.

(e) Faulty ROBDD of cone w.r.t wire l_2 *(s-a-1) at* l_1

Fig. 11 ROBDDS for feedback bridging fault: l_2 dominate l_1

patterns is $IP_{l_2,1} = \{00\times, 010\} = \{000, 001, 010\}$. The set of test patterns is obtained as $TP = IP_{l_1,s-a-1} \cap IP_{l_2,1} = \{001\}$. Now, the following steps are executed to check whether this test pattern causes oscillation or not.

- Build a ROBDD ($robdd_{l_2,l_1(s-a-1)}$) which represents the cone w.r.t wire l_2 under *stuck-at-1* at wire l_1. The boolean expression of l_2 under *stuck-at-1* at l_1 is $v_1'v_3'$ and $robdd_{l_2,l_1(s-a-1)}$ is shown in Fig. 13(e).

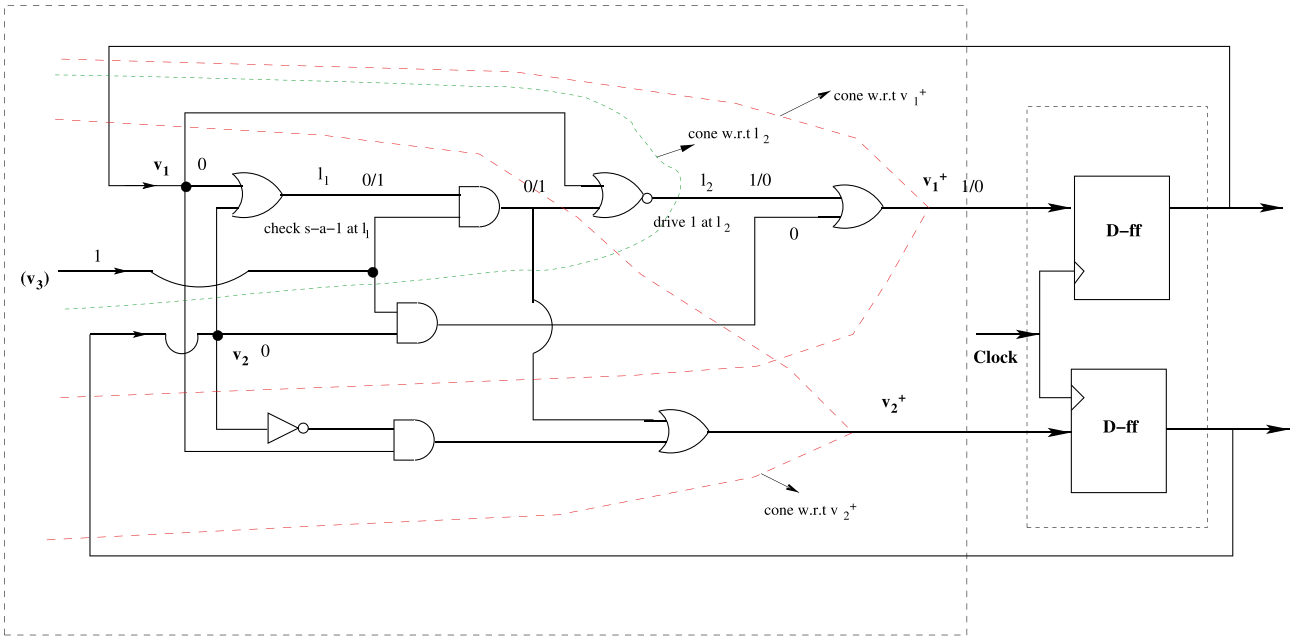


Fig. 12 Oscillating feedback bridging (l_1, l_2): $l2$ dominates $l1$

- Apply *satisfy-all-0* operation on $robdd_{l_2,l_1(s-a-1)}$ in order to generate input patterns ($IP_{l_2=0,l_1(s-a-1)}$) that makes logic 0 at wire l_2 during propagation of *stuck-at-1* fault at l_1 and the $IP_{l_2=0,l_1(s-a-1)} = \{1\times\times, 0\times1\} = \{100, 101, 110, 111, 001, 011\}$.
- For each test pattern $tp \in TP$ do the followings:
  - If $tp \in IP_{l_2=0,l_1(s-a-1)}$, then eliminate tp from TP, because it causes oscillation.
  - If $tp \notin IP_{l_2=0,l_1(s-a-1)}$, then it remains as a test pattern and does not cause oscillation. In this case tp is mapped to corresponding $FI-transtion$.

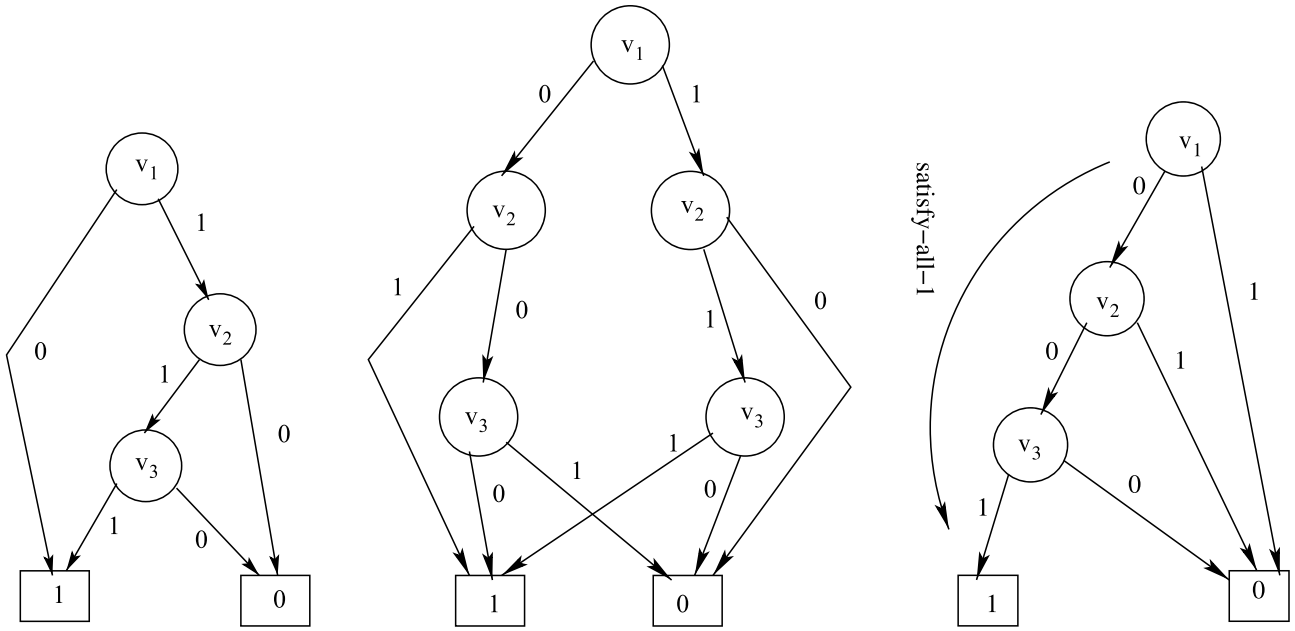
In this example, $TP = \{001\}$, $IP_{l_2=0,l_1(s-a-1)} = \{100, 110, 111, 001, 011\}$ and $\langle 001\rangle \in IP_{l_2=0,l_1(s-a-1)} = \{100, 101, 110, 111, 001, 011\}$. So, the test pattern $\langle 001\rangle$ causes oscillation. That means during propagation of *stuck-at-1* fault at l_1, the logic value at l_2 is changed from 1 to 0, thus causes oscillation. Such type of feedback bridging faults are known as *oscillating feedback bridging faults*. In this way, *oscillating feedback bridging faults* are detected.

6 A Flexible Tester Circuit Design Using Set of *FI − transtions*

In concurrent testing, a tester circuit is designed using the set of $FI-transitions$ and it is placed on-chip along with the CUT. The tester circuit executes parallel with circuit under test and detects the occurrence of any fault in the CUT on the fly during the normal operation. During design of tester circuit, all the wires of the CUT are tapped to the tester circuit. Therefore, buffer requirements for tapping all the wires to the tester circuit is very high, which results high area overhead. This work aims at providing flexibility in tester circuit design by dropping some of tap wires to the tester circuit. Dropping some of the tap wires reduces the buffer requirements for fanout the gates of the CUT, which directly decreases the area overhead. However, dropping some of tap wires to the tester circuit makes some of the $FI-transitions$ as $non-FI-transistions$, i.e., they don't remain as $FI-transitions$. Thus, it (dropping of some tap wires) results minor changes on fault coverage and fault detection latency. Thus, flexibility in tester circuit design is provided in terms of trades-off between dropping of tap wires versus area overhead, fault coverage and fault detection latency. The existence of $FI-transitions$ under dropping of some tap wires will be discussed in the next subsection.

6.1 Checking the Existence of *FI − transitions* Under Dropping of Some Tap Wires

In order to illustrate the process of checking of existence of $FI-transitions$ under dropping of some tap wires, consider the fault F_i (shown in Fig. 2) and the set of $FI-transitions$ generated to detect F_1 using the procedure discussed in the Sect. 5. The set of $FI-transitions$ is $\{\langle 00,0,\times0\rangle, \langle 00,1,\times0\rangle, \langle 11,1,1\times\rangle\}$. Consider the



(a) Normal ROBDD of cone w.r.t v_1^+.

(b) Faulty ROBDD of cone w.r.t v_1^+ (s-a-1 at l_1).

(c) XORed ROBDD

(d) ROBDD of cone w.r.t wire l_2.

(e) Faulty ROBDD of cone w.r.t wire l_2 *(s-a-1) at* l_1

Fig. 13 ROBDDS for feedback bridging fault: l_2 dominate l_1

$FI - transition$, $\langle 00, 0, \times 0\rangle$ and check the existence of it under dropping of some tap wires. There are 3 tap wires of the CUT shown in Fig. 2 and these are v_1, v_2, and v_3. For simplicity, drop any one of the tap wires and check the existence of $FI - transition$ under that dropped tap wire. Consider two cases; case (i) drop tap wire v_1 and case (ii) drop tap wire v_3.

- **Case i**: *Drop tap wire* v_1. The $FI - transition$, $\langle 00, 0, \times 0\rangle$ manifests F_1 through v_2^+. The faulty response for this $FI - transition$ at output v_2^+ is 0. This is obtained by applying input pattern $\langle 000\rangle$ in the faulty ROBDD shown in Fig. 8(b). Since the wire v_1 is dropped, the input pattern $\langle 000\rangle$ becomes $\times 00 = \{000, 100\}$, i.e., v_1 is treated as don't care ($\times$). For input pattern $\langle 000\rangle$, out-

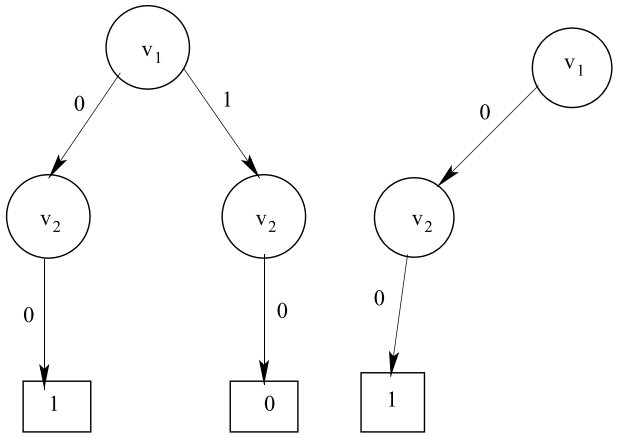
put response (v_2^+) under normal condition is 1, whereas for input pattern ⟨100⟩, output response (v_2^+) under normal condition is 0. This is obtained by applying input patterns ⟨000⟩ and ⟨100⟩ in the normal ROBDD shown in Fig. 8(a). It is seen that for input pattern ⟨000⟩, the normal and faulty responses are same and that is 0. So the $FI-transition : \langle 00,0,\times 0\rangle$ does not remain as an $FI-transition$ under dropping of tap wire v_1.

- **Case ii**: *Drop tap wire* v_3. In similar way, the input patterns ⟨000⟩ becomes $00\times = \{000, 001\}$, since the wire v_3 is dropped. For both the input patterns ⟨000⟩ and ⟨001⟩, the output responses (v_2^+) under normal condition is 1. The faulty response for the $FI-transition : \langle 00,0,\times 0\rangle$ manifests through v_2^+ is 0. Since the normal and faulty responses are different for $FI-transition : \langle 00,0,\times 0\rangle$, so it remains as an $FI-transition$ under dropping of tap wire v_3.

In this way, the existence of $FI-transitions$ under dropping of some tap points is checked. The checking of existence of $FI-transitions$ under dropping of some tap wires can be better accomplished using ROBDD. Consider the $FI-transitions : \delta_{ir} = \langle q_{ir}, \sigma_{ir}, q_{ir}^+\rangle$ for the fault F_i, which is manifested through output v_j^+. Let the faulty response for $FI-transition$ at output v_2^+ is $response_{faulty}$. The following steps are executed to check the existence of $FI-transitions$ under dropping of some tap points using ROBDD.

1. Let $robdd_{normal}$ be the ROBDD representation of output v_j^+ under normal condition.
2. If v_i is not a dropped wire and value of v_i in $FI-transition$ is 0(1), then keep the edge with label 0(1) and eliminate the edge with label 1(0) from the node corresponding to v_i in $robdd_{normal}$.
3. Remove the nodes and edges in the $robdd_{normal}$, which are not reachable from root after eliminating edges.
4. Repeat Steps 2 and 3 for each wire v_i which is not being dropped.
5. In the resultant ROBDD if there is a path from root to leaf whose value is $response_{faulty}$, then the $FI-transition$ does not remain as an $FI-transition$, otherwise it remains an $FI-transition$.

The existence of $FI-transition$ under dropping of some tape wires can be easily checked using ROBDD. Consider case (i), where the wire v_1 is dropped, the corresponding modified ROBDD to check the existence of $FI-transition : \langle 00,0,\times 0\rangle$ is shown in Fig. 14(a). The modified ROBDD is constructed from normal ROBDD (Fig. 8(a)), where all the outgoing edges of node corresponding to v_1 are kept since it is dropped in the design of tester circuit. For nodes corresponding to v_2, keep the outgoing edges with label 0 and eliminate the outgoing edges



(a) Modified ROBDD under dropping of wire v_1.

(b) Modified ROBDD under dropping of wire v_3.

Fig. 14 Modified ROBDDs for checking existence of $FI-transition$ under dropping of tap wires

with label 1, since the wire v_2 is not a dropped wire and the value of v_2 in $FI-transition$ is 0. Remove the node corresponding v_3 since it is not reachable from root. The modified ROBDD is shown in Fig. 14(a). Since the faulty response value at v_2^+ is 0 (i.e., $response_{faulty} = 0$) and there exists a path in the modified ROBDD from root to $response_{faulty}(= 0)$. So, the $FI-transition : \langle 00,0,\times 0\rangle$ does not remain as an $FI-transition$ under dropping of wire v_1. In similar way, it can be checked that the $FI-transition : \langle 00,0,\times 0\rangle$ remains as an $FI-transition$ under dropping of wire v_3. The modified ROBDD for checking the same is shown in Fig. 14(b).

6.2 Tester Circuit Design Using Existing Set of $FI-transitions$ Under Dropping of Some Tap Wires

This subsection illustrates the procedure of design of tester circuit using set of $FI-transitions$. First a complete set of $FI-transitions$ has been generated for the targeted bridging faults of the CUT. Then, the existence of each $FI-transitions$ has been checked under dropping of some tap wires. Finally, the tester circuit is designed using the existing set of $FI-transitions$. Since the CUT is a synchronous circuit, so the tester circuit must be designed as a synchronous circuit. The tester circuit runs parallel with the CUT and detects occurrence of a fault by means of executing any $FI-transitions$ during the normal operation of the CUT.

Figure 15 shows the state transition diagram of the tester circuit designed using $FI-transition : \langle 00,0,\times 0\rangle$ where the wire v_3 is dropped. The tester circuit starts

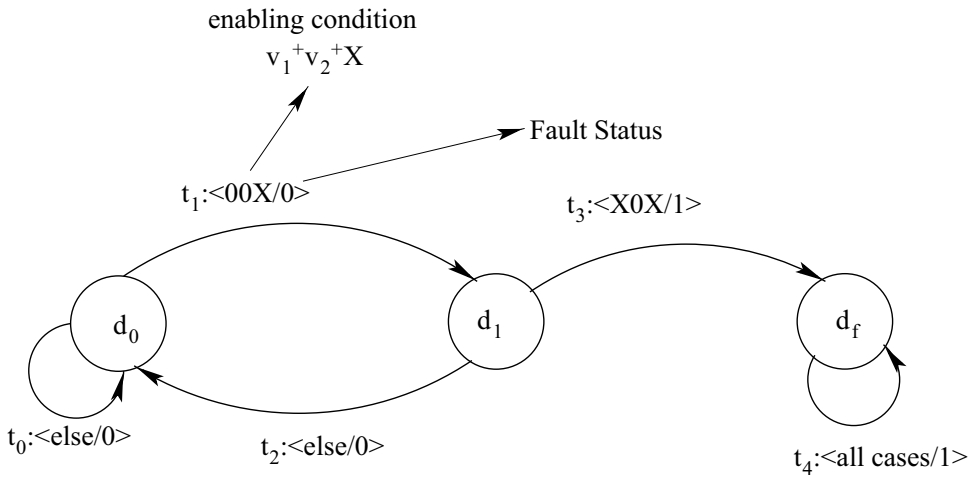


Fig. 15 State transition diagram of tester circuit for $FI-transition : \langle 00, 0, \times 0\rangle$ where v_3 is dropped

from initial state d_0, moves to state d_1 with the transition $t_1 : \langle 00 \times /0\rangle$. The transition t_1 is associated with enabling condition values 00× and fault status value 0. The enabling condition values (00) imply the values of next state variables $(v_1^+ v_2^+)$ and × implies v_3 has don't care value. That means when the CUT moves to the state 00, the tester moves to state d_1 with fault status 0 indicates fault has not been detected yet. At state d_1, the tester checks the occurrence of $FI-transition : \langle 00, 0, \times 0\rangle$ in the CUT (next clock edge). This checking is accomplished by transition $t_3 : \langle \times 0 \times /1\rangle$. The enabling condition of t_3 is ×0×, which implies the values of next state variables $v_1^+ v_2^+ = \times 0$ and the value of v_3 is don't care. Further, the fault status 1 indicates fault has occurred in the CUT. Once the tester has reached at state d_f by the transition t_3, it remains in that state (d_f) forever with fault status 1. This is accomplished by transition $t_5 : \langle all\ cases/1\rangle$. At state d_1, if the enabling condition of transition t_3 is not satisfied, then tester moves back to state d_0 by transition $t_2 : \langle else/0\rangle$. Similarly, at state d_0, if the enabling condition of transition t_1 is not satisfied, then tester remains in state d_0 by the transition $t_0 : \langle else/0\rangle$.

In this way, state transition diagram of tester circuit for each $FI-transitions$ is designed. In the diagram, it has common initial (d_0) and final (d_f) states for each $FI-transitions$. The number of intermediate states depends on the number of $FI-transitions$ taken in the design of the tester circuit. Further, it has been seen that in some cases two or more $FI-transitions$ have common intermediate state. Thus, the tester circuit can be defined as FA as follows:

$$T = \langle D, \Sigma_D, d_0, \delta_d, O_D, d_f\rangle, \tag{2}$$

where D is the finite set of states, Σ_D is the input alphabet, d_0 is the initial state, $\delta_d : D \times \Sigma_D \rightarrow D$ is the transition function, $O_D : D \times \Sigma_D \rightarrow \{0, 1\}$ is the output function (indicate fault status), and d_f is the final state. The state transition diagram is constructed using the following steps.

1. Create two states; d_0 is initial state and d_f is the final state.
2. Let $V_{not\ dropped}$ be the set of wires that are not dropped in design of tester circuit. Repeat this step for each $FI-transition : \delta_{ir} = \langle q_{ir}, \sigma_{ir}, q_{ir}^+\rangle$
   (a) Create an intermediary state d_j, which is reached from d_0 using transition t_j. The enabling condition of t_j is values of $v_1^+, v_2^+, ..., v_k^+ (\in V_{not\ dropped}) = begin(\delta_{ir})$ and values of inputs $v_{k+1}, ..., v_m (\in V_{not\ dropped}) = \times (don't\ care)$. Fault status of t_j is 0.
   (b) There is a transition t_k from state d_j to final state d_f. The enabling condition of t_k is values of $v_1^+, v_2^+, ..., v_k^+ (\in V_{not\ dropped}) = end(\delta_{ir})$ and values of inputs $v_{k+1}, ..., v_m (\in V_{not\ dropped}) = input(\delta_{ir})$. Fault status of t_k is 1.
3. For any two intermediary states d_j (for $FI-Transiton : \delta_{ir} = \langle q_{ir}, \sigma_{ir}, q_{ir}^+\rangle$) and d_k (for $FI-Transiton : \delta_{is} = \langle q_{is}, \sigma_{is}, q_{is}^+\rangle$), if $q_{ir} = q_{is}$ then d_j and d_k are merged into a single state.
4. Add a transition from each intermediary state d_j to initial state d_0, whose enabling condition is any values of $v_1^+, v_2^+, ..., v_k^+, v_{k+1}, ..., v_m$, other than the enabling condition of transition from state d_j to final state d_f. Fault status of this transition is 0.
5. Add a self transition at the state d_0, whose enabling condition is any values of $v_1^+, v_2^+, ..., v_k^+, v_{k+1}, ..., v_m$, other than the enabling condition of transition from state d_0 to any intermediary state d_j. Fault status of this self transition is 0.
6. Add a self transition at the state d_f, whose enabling condition is any values of $v_1^+, v_2^+, ..., v_k^+, v_{k+1}, ..., v_m$ and fault status of this self transition is 1.

7 Experimental Results

In order to validate the proposed methodologies discussed in Sect. 5 (generation of $FI-transitions$) and Sect. 6 (flexible design of tester circuit), a set of ISCAS89 sequential benchmark circuits [20] have been taken. Bridging faults are applied between all possible pairs of wires of each benchmark circuits. A set of $FI-transitions$ are generated for each faults of the circuit and checked the existence of them under dropping of some tap wires using proposed ROBDD based technique. Then tester circuit is design using these $FI-transitions$. The generic flow of the method that is applied to the circuits is follows as:

1. Extract the part of the wirelist/ netlist that corresponds to the NSF block of the circuit.
2. Eliminate FFs and partition the wirelist (into sub-circuits) according to cones of influence corresponding to each of the FFs. The output of each of the sub-circuits is the corresponding input to the flip-flop.
3. Repeat the following steps for all the sub-circuits generated in Step 2.
   (a) Insert bridging faults (l_1, l_2) in all possible locations.
   (b) Repeat the following steps to generate $FI-transitions$ for each fault using the mechanism of l_1 dominates l_2.
   i Generate ROBDD for the sub-circuit under normal condition.
   ii Generate ROBDD for the sub-circuit under faulty ($s\text{-}a\text{-}1$ *at* l_2) condition.
   iii XOR the normal and faulty ROBDDs. The variable values corresponding to the paths to leaf node "1", are the input patterns to detect $s\text{-}a\text{-}1$ *at* l_2.
   iv Generate ROBDD for the sub-circuit w.r.t wire l_1 under normal condition. The variable values corresponding to the paths to leaf node "1", are the input patterns to drive *logic 1 at* l_2.
   v Intersect the input patterns obtained from Steps (3.b.iii) and (3.b.iv) to generate the test patterns to detect bridging fault between l_1 and l_2. Map these test patterns to $FI-transitions$ for the fault.[1]
   vi Finally, determine the $FI-transitions$ which remain so under given dropping of tap wires and drop the remaining ones.
4. Generate the FA model of the tester circuit with the $FI-transitions$ and translate the FA to Verilog code.

[1] Details of these steps can be found in Algorithm 1.

The performance of the proposed scheme is analyzed in terms of area overhead, fault coverage and detection latency. Further, flexibility of the scheme is studied by dropping some of tap wires. Now, performance parameters are defined as follows:

Fault Coverage (FC): A fault (say F_i) is supposed to be covered, if at least one of the $FI-transition$ is taken in design of the tester circuit. Thus, fault coverage is defined as:

$$Fault\ Covergae = \frac{Number\ of\ faults\ covered}{Total\ Number\ of\ faults\ in\ the\ CUT} \times 100\%$$

Area Overhead (AO): Area overhead can be defined as:

$$Area\ Overhead = \frac{Area\ of\ the\ tester\ circuit}{Area\ of\ the\ CUT}$$

Fault Detection Latency: In this work, fault detection latency is calculated based on the number of $FI-transitions$ are taken in the design of tester circuit. A fault may have a set of $FI-transitions$ to detect it. Instead of taking all $FI-transitions$ of the fault in design of tester circuit, a subset of $FI-transitions$ is taken in design of tester circuit in order to reduce area overhead without affecting fault coverage. This is accomplished with the help of fault detection latency. A fault (F_i) is detected with 0 detection latency, when all of it's $FI-transitions$ are taken in design of the tester circuit. Suppose for F_i, there are total n number of $FI-transitions$ and out of them $m(\leq n)$ number of $FI-transitions$ are taken in design of tester circuit. Then fault detection latency of F_i is defined as:

$$Fault\ detection\ latency\ of\ F_i = \lceil \frac{n}{m} \rceil - 1$$

For example, if there are total 10 number of $FI-transitions$ to detect fault F_i and out of them 5 number of $FI-transitions$ are taken in design of the tester circuit. Then, detection latency for F_i is 1. This implies, if some of the $FI-transitions$ for a fault are dropped during design of the tester circuit then detection latency of that fault is increased. However, it does not affect the fault coverage ratio and it reduces the area overhead.

7.1 Analysis of Fault Coverage

In this subsection, the fault coverage of the proposed scheme is discussed and it is compared with the existing scheme with the help of different benchmark circuits. Table 2 shows the detail description of different ISCAS89 benchmark circuits. Column 1 indicates the circuit's name with number of flip flops and number of gates. Column 2 shows the total number of bridging faults of the circuit. Columns 3 and 4 display the percentage of non-feedback and feedback bridging faults of the circuit, respectively. Columns 5 and 6 show the percentage of fault coverage of the proposed and existing schemes, respectively. In this work, the different combinations (1 or 2 or 3) of tap wires are dropped and the effect of dropping of tap wires is studied with respect to fault coverage, area overhead and detection latency.

Table 2 Description of ISCAS89 Benchmark Circuits, Fault Coverage of Proposed Scheme and comparison with [3]

Circuits (FFs, GATEs)	Total number of bridging faults (BFs)	Number of non-feedback BFs (%)	Number of feedback BFs(%)	Average FC (%)	Existing FC (%)[3]
s298-(14,119)	9180	90.94	9.06	97.5	99
s344-(15,160)	16836	85.96	14.04	97	99
s382-(21, 158)	16471	90.11	9.89	97.5	99
s838-(32,446)	130816	91.15	8.85	97.5	99
s1238-(18,508)	145530	89.25	10.75	98	99
s1423-(74,657)	279378	83.07	16.93	97.5	99
s5378-(179,2779)	4477528	79.86	20.14	97	99
s9234-(228,5597)	17073246	78.72	21.28	98	99
s13207-(669,7951)	37415575	78.12	21.88	97	99
s15850-(597,9772)	53898153	77.13	22.87	97.5	99
s35932-(1728,16065)	158909878	75.28	24.72	97	99
s38417-(1636,22179)	284232403	74.14	25.86	97.5	99
s38584-(1452,19253)	214586686	74.27	25.73	97.5	99

Figures 16 and 17 depict the fault coverage for the benchmark circuits s344 and s1432, respectively. In each case, it has shown the fault coverage with 15 different combinations (1 or 2 or 3) of dropping of tap wires. The following points can be noted regarding fault coverage of proposed scheme.

- For each combinations (1 or 2 or 3) of dropping of tap wires for all benchmark circuits fairly good fault coverage is achieved.
- Table 2 shows the comparison of fault coverage between the proposed and existing schemes. It has been found the average fault coverage of the proposed scheme for all benchmark circuits with different combinations (1 or 2 or 3) of dropping of tap wires is approximately 97.5%. However, the average fault coverage of the existing scheme ([3]) for all benchmark circuits is 99%. There is a minor difference of fault coverage between the proposed and existing schemes. This is because of dropping of tap wires makes few of $FI-transitions$ as $non-FI-transitions$ and cannot detect the fault. Further, it is found that in some rare cases all the $FI-transitions$ for a fault become $non-FI-transitions$ under dropping of tap wires, in such cases fault cannot be covered.
- The average fault coverage of proposed scheme for all benchmark circuits is (≈)97.5%. This is achieved because of for each fault at least one $FI-transition$ is taken in design of tester circuit. The remaining (≈)2.5% faults are hard to cover (detect) because these are redundant faults and oscillating feedback bridging faults. In both the cases generation of $FI-transitions$ is quite impossible.

7.2 Analysis of Area Overhead

In this subsection, the area overhead of the proposed scheme is discussed and it is compared with the existing scheme with the help of different benchmark circuits.

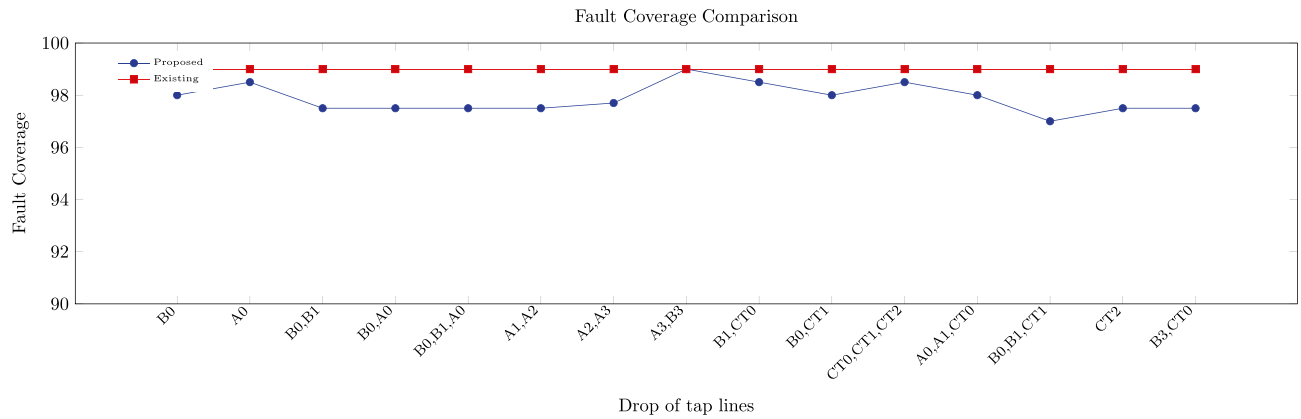


Fig. 16 Fault Coverage for benchmark circuit s344 with 15 different combinations (1 or 2 or 3) of dropping of tap wires and comparison with [3]

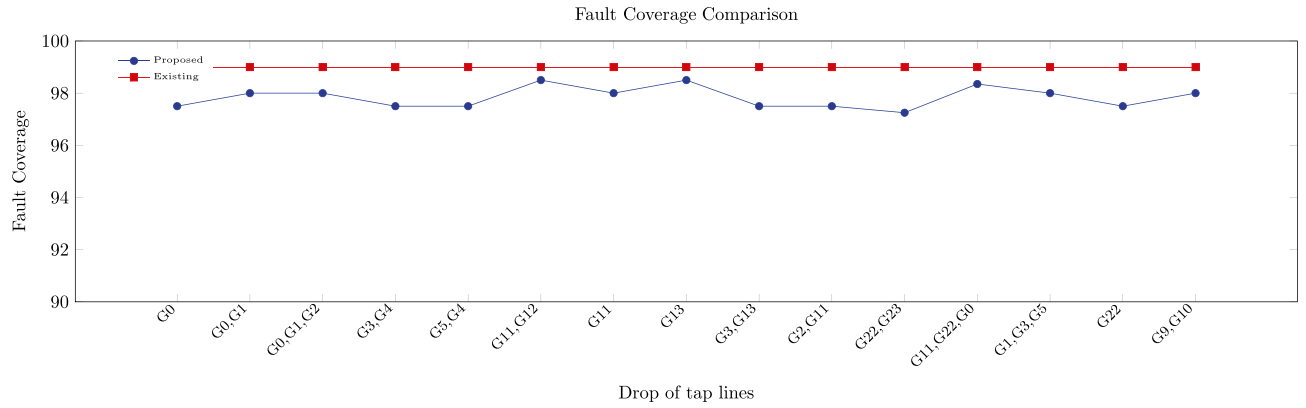


Fig. 17 Fault Coverage for benchmark circuit s1423 with 15 different combinations (1 or 2 or 3) of dropping of tap wires and comparison with [3]

Table 3 shows the area overhead ratio of the proposed scheme for different combinations (1 or 2 or 3) of dropping of tap wires and comparison with the area overhead reported in [3]. Column 1 shows the name of the circuit with number of flip flops and number of gates. Columns 2, 3, and 4 indicate the area overhead of the proposed scheme for dropping of one, two, and three tap wires, respectively. Column 5 represents the average area overhead (dropping of 1, 2, and 3 tap wires) of the proposed scheme and Column 6 shows area overhead of the existing scheme reported in paper [3]. The following points can be noted regarding area overhead of the proposed scheme.

- It is observed from Columns 2, 3, and 4 of Table 3 that the area overhead of proposed scheme is decreased with increase of number of dropping of tap wires. Reduction of area overhead is one of the primary objectives in concurrent testing of modern integrated circuits.
- It can be seen from Table 3 that for a given dropping of tap wires, the area overhead ratio is decreased with increase of the size of the circuit. The reason is the existence of common $FI - transitions$ in large sized circuits is more than that of small sized circuits. A common $FI - transition$ is one which can detect more than one faults. In this work, first the common $FI - transitions$ are taken in the design of tester circuit. Then rest of faults are covered (based on the given value of detection latency) by taking non-common $FI - transitions$ in design of the tester circuit. In this case, the value of detection latency is 0, i.e., all $FI - transitions$ for a fault are taken in design of tester circuit.
- In all cases (dropping of 1 or 2 or 3 tap wires), the area overhead of proposed scheme is decreased with increase of the number of gates (size) of the circuit. This follows the area overhead ratio approximation of any partial replication based concurrent testing scheme reported in [14].

Table 3 Area Overhead of proposed scheme for different combinations (1 or 2 or 3) of dropping of tap wires and comparison with

Circuits (FFs, Gates)	AO for different combinations (1 or 2 or 3) of dropping of tap wires and comparison with [3]				
	AO for dropping of 1 tap wire	AO for dropping of 2 tap wires	AO for dropping of 3 tap wires	Average AO	existing AO [3]
s298-(14,119)	2.45	1.98	1.78	2.07	4.03
s344-(15,160)	2.35	1.86	1.69	1.97	4.04
s382-(21, 158)	2.12	1.63	1.58	1.78	4.12
s838-(32,446)	1.54	1.40	1.31	1.42	2.64
s1238-(18,508)	1.44	1.32	1.23	1.33	2.54
s1423-(74,657)	1.41	1.28	1.11	1.27	2.51
s5378-(179,2779)	1.32	1.15	1.08	1.18	2.32
s9234-(228,5597)	1.29	1.01	0.96	1.09	2.29
s13207-(669,7951)	1.10	0.95	0.91	0.98	2.27
s15850-(597,9772)	1.08	0.93	0.88	0.96	2.22
s35932-(1728,16065)	0.95	0.91	0.85	0.90	2.13
s38417-(1636,22179)	0.93	0.88	0.82	0.87	2.11
s38584-(1452,19253)	0.91	0.85	0.80	0.85	2.12

Table 4 Trade-off analysis between Area Overhead, detection latency and dropping of tap wires. Comparison of Area Overhead with existing scheme [3]

Circuits (FFs, Gates)	AO for different combinations (1 or 2 or 3) of dropping of tap wires and comparison with [3]									
	Detection latency = 2					Detection latency = 4				
	AO for dropping of			Average	existing	AO for dropping of			Average	existing
	1 tap wire	2 tap wires	3 tap wires	AO	AO [3]	1 tap wire	2 tap wires	3 tap wires	AO	AO [3]
s298-(14,119)	1.56	1.54	1.43	1.51	3.90	1.54	1.52	1.35	1.47	3.86
s344-(15,160)	1.43	1.40	1.37	1.40	3.80	1.41	1.38	1.33	1.37	3.74
s382-(21, 158)	1.24	1.21	1.20	1.22	3.75	1.23	1.21	1.17	1.20	3.90
s838-(32,446)	1.20	1.12	1.10	1.14	2.75	1.18	1.10	1.07	1.11	2.52
s1238-(18,508)	1.16	1.11	1.08	1.11	2.50	1.15	1.08	1.05	1.09	2.31
s1423-(74,657)	1.10	1.02	0.97	1.03	2.42	1.08	1.00	0.96	1.01	2.21
s5378-(179,2779)	1.08	0.97	0.95	1.00	1.80	1.06	0.96	0.94	0.98	1.61
s9234-(228,5597)	1.05	0.95	0.93	0.97	1.68	1.03	0.93	0.92	0.96	1.57
s13207-(669,7951)	0.97	0.93	0.91	0.94	1.60	0.96	0.90	0.87	0.91	1.53
s15850-(597,9772)	0.92	0.87	0.85	0.88	1.52	0.91	0.86	0.84	0.87	1.45
s35932-(1728,16065)	0.90	0.85	0.83	0.86	1.35	0.88	0.83	0.82	0.84	1.215
s38417-(1636,22179)	0.88	0.83	0.81	0.84	1.31	0.86	0.82	0.81	0.83	1.192
s38584-(1452,19253)	0.87	0.82	0.80	0.83	1.24	0.85	0.81	0.80	0.82	1.183

The approximation states that the area overhead ratio is approximately $a + 1/n$, where a fraction test patterns (here, $FI - transitions$) taken in design of tester circuit and n is the number of state bits (directly proportional to size of the circuit).

- It is seen that the average area overhead obtained from dropping of 1, 2 and 3 tap wires is much less than that of the area overhead of the existing scheme reported in [3]. This is because of dropping of tap wires reduces the number of buffer requirements to drive the tester circuit, which directly reduces the area overhead. Thus, it is seen from the table that dropping of tap wires have a significant effect (reduction) in the area overhead.

7.3 Trade-off Analysis Between Area Overhead, Detection Latency and Dropping of Tap Wires

As discussed the proposed scheme provides flexibility in design of tester circuit by dropping some of tap wires to the tester circuit. The flexibility of the scheme is illustrated in terms of trade-off analysis between area overhead, detection latency and dropping of tap wires. Table 4 shows the area overhead of the proposed scheme for a given detection latency and dropping of tap wires. Column 1 shows the details regarding benchmark circuits, Columns from 2 to 6 are meant for detection latency 2, where Columns 2, 3, and 4 represent the area overhead of the proposed scheme for dropping of one, two, and three tap wires, respectively. Columns 5 represents the average area overhead of proposed scheme obtained from dropping of one, two, and three tap wires and Column 6 represents the area overhead of existing scheme. Similarly, Columns from 7 to 8 are meant for detection latency 4, where Columns 7, 8, and 9 represent the area overhead of the proposed scheme for dropping of one, two, and three tap wires, respectively. Columns 10 represents the average area overhead of proposed scheme obtained from dropping of one, two, and three tap wires and Column 11 represents the area overhead of existing scheme. The following points are observed from the trade-off analysis between area overhead, detection latency and dropping of tap wires.

- For a given value of detection latency, the area overhead of the proposed scheme is decreased with increase of number of dropping of tap wires. It is seen in the table that for both the cases (detection latency=2 and detection latency =4), the area overhead is gradually decreased when the number of dropping of tap wires to the tester circuit is increased from one to three.
- For example, the average area overhead ratios for the benchmark circuit $s13207$ are 0.94 and 0.91 when detection latency is 2 and 4, respectively. The reason for decrease of area overhead ratio is that for each fault the number of $FI - transitions$ taken in design of tester circuit with detection latency = 2 is less than that of with detection latency = 4.
- It can be seen in the Table 4 that dropping of tap wires facilitates trade-off analysis between area overhead and

detection latency. Hence, flexibility in design of the tester circuit can be achieved through dropping of tap wires. Further, it reduces area overhead greatly with minimal compromise in fault coverage.

8 Conclusion

This paper presents a flexible concurrent testing scheme for non-feedback and feedback bridging faults in the integrated circuit. The proposed scheme provides flexibility in design of the tester circuit by dropping of some tap wires, which facilitates trade-off analysis between area overhead, fault coverage and detection latency. The proposed scheme follows the working principle of partial replication technique and achieves low area overhead with minor changes in fault coverage. Experimentally, it is seen that dropping of tap wires is worked as trade-off parameter between area overhead and detection latency, thus, flexibility in design of tester circuit is achieved through it. Also, it is seen that for a given value of detection latency the area overhead is reduced with increase of number of dropping of tap wires. The reason is that dropping of some of tap wires makes less number of wires to be tapped by the tester circuit from the CUT, which decreases the number of driving buffers, thus area overhead is reduced. The proposed scheme utilizes ROBDD based algorithms to generate test patterns ($FI - transitions$) for non-feedback and feedback bridging faults as well. Further, checking the existence of an $FI - transitions$ under dropping of tap wires is carried out with the help of ROBDD based algorithms. Use of ROBDD based algorithms to generate and check of existence of $FI - transitions$ under dropping of tap wires make the proposed scheme more robust and improve scalability, thus large sized circuit can be handled successfully.

Using the proposed methodology flexible concurrent tester circuit can be designed for any digital circuit, however the design complexity of the scheme is increased with increase of the size of the circuit and may become impractical for circuits typically having more than tens of thousands of inputs and state bits. The reason is that the proposed scheme uses ROBDD based operations and in such cases generation of ROBDDs itself becomes too complex and may suffer state explosion problem. This issue can be addressed by improving the scalability of the proposed scheme. Some of the possible techniques to improve the scalability of the proposed scheme are as follows:- (1) Modeling of the circuits at higher abstraction level compared to gate level, e.g., Register Transfer Level (RTL), (2) Fault modeling at higher level of abstractions and correlation with accepted fault models like stuck-at, bridging, etc. (3) Trade-off analysis at higher abstraction level. Further, in this work three different types of dropping of tap wires have been taken. These are dropping of a single tap wire, two tap wires, and three tap wires, respectively. The decision of dropping of tap wires has been done randomly and there is no use of any algorithm in identification of dropping of tap wires. However, the identification of dropping of tap wires can be performed by solving an optimization problem where area overhead of the tester circuit, fault coverage, fault detection latency, etc., as optimization parameters. Clearly, further research is essential to resolve these problems.

Data Availability The manuscript uses ISCAS89 benchmark circuits and is cited in the reference list.

Declarations

Conflict of Interest/Conflict of Interest The author declare that there is no conflict of interest.

References

1. Acharya N, Urbanek M, De Jong WA, Saeed SM (2021) Test points for online monitoring of quantum circuits. ACM J Emerg Technol Comput Syst (JETC) 18(1):1–19
2. Balasubrahamanyam Y, Chowdary GL, Subrahmanyam TJVS (2012) A novel low power pattern generation technique for concurrent BIST architecture. Int J Comput Technol Appl 3(2):561–565
3. Biswal PK, Biswas S (2015) A binary decision diagram based on-line testing of digital VLSI circuits for feedback bridging faults. Microelectron J, Elsevier 46(7):598–616
4. Biswal PK, Biswas S (2019) A binary decision diagram approach to on-line testing of asynchronous circuits with dynamic and static C-elements. J Electron Test, Springer 35(5):715–727
5. Biswas S, Mukhopadhyay S, Patra A (2005) A formal approach to on-line monitoring of digital VLSI circuits: theory, design and implementation. J Electron Test, Springer 21(5):503–537
6. Biswas S, Mukhopadhyay S, Patra A, Sarkar D (2008) Unified technique for on-line testing of digital circuits: delay and stuck-at fault models. J Circuits, Syst Comput, World Scientific 17(06):1069–1089
7. Biswas S, Mukhopadhyay S, Patra A, Sarkar D (2006) Concurrent testing of digital circuits for advanced fault models. In: Proc. of Design and Diagnostics of Electronic Circuits and systems, IEEE, pp 202–207
8. Biswas S, Srikanth P, Jha R, Mukhopadhyay S, Patra A, Sarkar D (2005) On-line testing of digital circuits for n-detect and bridging fault models. In: Proc. of Asian Test Symposium (ATS'05), pp 88–93
9. Bolchini C, Montandon R, Salice F, Sciuto D (2000) Design of VHDL based totally self-checking finite state machine and data path descriptions. IEEE Trans VLSI Syst 8(1):98–103
10. Bushnell M, Agrawal V (2006) Essentials of electronic testing for digital, memory and mixed-signal VLSI Circuits, vol 17. Springer Science and Business Media
11. Chang WF, Wu CW (1999) Low-cost modular totally self-checking checker design for m-out-of-n code. IEEE Trans Comput 48(8):815–826
12. Dhawan S, Vries CD (1988) Design of self-checking sequential machines. IEEE Trans Comput 37(10):1280–1284

13. Drineas P, Makris Y (2003) SPaRre: Selective partial replication for concurrent fault-detection in fsms. IEEE Trans Instrum Meas 52(6):1729–1737
14. Drineas P, Makris Y (2002) Non-intrusive design of concurrently self-testable FSMs. In: Proc. of Asian Test Symposium, pp 33–38
15. El-Mahlawy MH (2015) Signature multi-mode hardware-based self-test architecture for digital integrated circuits In: Proc. of International Conference on Electronics, Circuits, and Systems (ICECS), pp 437–441
16. Emmert JM, Stroud CE, Bailey JR (2000) A new bridging fault model for more accurate fault behavior. In: Proc. of Systems Readiness Technology Conference. Future Sustainment for Military Aerospace (Cat. No. 00CH37057), IEEE, pp 481–485
17. Favalli M, Dalpasso M (2016) Boolean and pseudo-boolean test generation for feedback bridging faults. IEEE Trans Comput 65(3):706–715
18. Favalli M, Metra C (2002) On-line testing approach for very deep-submicron ICs. IEEE Des Test Comput 19(2):16–23
19. Gaur H, Sasamal T, Singh A, Mohan A (2020) Fault models and test approaches in reversible logic circuits. Design and testing of reversible logic. Springer, pp 153–167
20. ISCAS89 sequential benchmark circuits. https://filebox.ece.vt.edu/~mhsiao/iscas89.html
21. Maier J, Steininger A (2014) Online test vector insertion: a concurrent built-in self-testing (CBIST) approach for asynchronous logic. In: Proc. of 17th International Symposium on Design and Diagnostics of Electronic Circuits & Systems, pp 33–38
22. Mano MM (2017) Digital logic and computer design. Pearson Education, India
23. Nicolaidis M, Anghel L (1999) Concurrent checking for VLSI. Microelectron Eng, Elsevier 49(1):139–156
24. Nicolaidis M, Zorian Y (1998) On-line testing for VLSI- A compendium of approaches. J Electron Test, Springer 12(1–2):7–20
25. Nicolescu B, Gorse N, Savaria Y, Aboulhamid E, Velazco R (2005) On the use of model checking for the verification of a dynamic signature monitoring approach. IEEE Trans Nucl Sci 52(5):1555–1561
26. Rayudu K, Jahagirdar J, Rao P (2020) Modern design approach of faults (toggling faults, bridge faults and SAT) of reduced ordered binary decision diagram based on combo & sequential blocks. Int J Reconfigurable & Embedded Syst ISSN 2089 (4864)4864
27. Robinson SH, Shen JP (1992) Direct methods for synthesis of self-monitoring state machines. In: Proc. of International Symposium on Fault-Tolerant Computing, pp 306–315
28. Tahoori MB (2004) Application-specific bridging fault testing of FPGAs. J Electron Test, Springer 20(3):279–289
29. Voyiatzis I, Paschalis A, Gizopoulos D, Halatsis C, Makri FS, Hatzimihail M (2008) An input vector monitoring concurrent BIST architecture based on a precomputed test set. IEEE Trans Comput 57(8):1012–1022
30. Zhao Y, Khursheed S, Al-Hashimi BM (2015) Online fault tolerance technique for TSV-based 3D-IC. IEEE Trans Very Large Scale Integr VLSI Syst 23(8):1567–1571

Pradeep Kumar Biswal has received the M.Tech. degree from the Department of Computer Science & Engineering, IIT Guwahati, India, in the year of 2011. He completed his Ph.D degree in the Department of CSE, IIT Guwahati, India, in the year of 2017. Now, he is working as Assistant Professor in the Department of CSE, IIIT Bhagalpur, Bihar, India, since October, 2017. His research interests include VLSI Testing, Design for Testability, Applications of Decision Diagrams, Discrete Event System Modeling, etc. He has published 12 research papers. The papers are published in the reputed journals like Journal of Electronics Testing (Springer), Microelectronics Journal (Elsevier), Engineering Science and Technology, an International Journal (Elsevier), VLSI Design (Hindawi), etc. His conference papers are published in the proceedings of IEEE VLSI Design (VLSID), IEEE Region 10 Symposium (TENSYMP), IEEE Mediterranean Conference on Control and Automation, etc.