A Comprehensive Approach to the Partial Scan Problem using Implicit State Enumeration

Priyank Kalla and Maciej Ciesielski
Department of Electrical and Computer Engineering
University of Massachusetts Amherst, MA-01003
Email: {pkalla, ciesiel}@ecs.umass.edu

Submitted to IEEE Transactions on CAD

Designated contact author:
Priyank Kalla, *pkalla@ecs.umass.edu*

ABSTRACT

This paper presents a novel technique to evaluate the non-controllability measures of state registers for partial scan design. Our model uses implicit techniques for FSM traversal to identify non-controllable state registers. By implicitly enumerating the states of a machine, we accurately evaluate the non-controllability of flip-flops by determining exactly what values can or cannot be stored, or are difficult to store in the state registers. By doing so, we not only target the untestable faults due to state unreachability of the machine, but also the difficult-to-test faults caused by difficult-to-control flip-flops. The values observed in the flip-flops during the implicit FSM traversal are used to evaluate flip-flop controllability measures to support the testability analysis. This technique is programmed as an algorithm called SIMPSON and we analyze its effectiveness by carrying out extensive experiments over a large set of MCNC and ISCAS benchmarks.

For large circuits, implicit state enumeration becomes infeasible because of computer memory and time limitations. To overcome these limitations, we propose the use of approximate reachability analysis of the circuit to estimate the non-controllability of state registers. By partitioning a large FSM into smaller sub-FSMs, and implicitly traversing the individual submachines, the reachable state set can be over-approximated as a product of smaller subsets. The values observed in the flip-flops of the submachines during the approximate FSM traversal facilitates the estimation of their non-controllability measures. An algorithm called SAMSON is proposed for this purpose and its effectiveness is illustrated over some of the larger circuits in the ISCAS benchmark suite. The results demonstrate the superiority of our method over conventional state-of-the-art scan register selection techniques in terms of higher fault coverage achieved by selecting fewer, or an equal number, of partial scan registers.

This work has been supported in part by a grant from NSF under contract No. MIP-9613864

I. Introduction

Over the years, attempts to automate test generation for sequential circuits have been pursued extensively. Though these attempts have met with varying levels of success, automatic test generation (ATG) has generally had difficulties with large sequential circuits, because of which, various *design for testability* techniques have become common practice in industry. The *full scan* technique has been developed to simplify the problem of testing a sequential circuit by converting it into a combinational one. This enables the application of combinational test generation algorithms, such as the D-algorithm [1], PODEM [2] and FAN [3], on such circuits. Although the testing problem is simplified, the area and performance of the circuit are adversely affected due to the necessary circuit modifications required to accommodate the complete scan chain. This also results in unacceptable lengths of the resulting tests due to extensive serial shifting of test patterns and responses.

Partial scan, on the other hand, provides a trade-off between the ease of testing and the costs associated with scan design. However, the key problem in partial scan design is the selection of scan registers. A lot of research has been devoted to define the criteria to guide the selection of the scan memory elements. These techniques can be categorized according to i) testability analysis [4], ii) test pattern generation [5] [6], iii) structural analysis [7] [8] [9], iv) fault oriented [10] and cost analysis based approach [11], and v) other miscellaneous methods [12] [13] [14], etc. All of the above mentioned techniques have met with some measure of success but have their respective limitations.

Partial scan methods based on *testability analysis*, use controllability, observability [15], and sequential depth as measures of circuit testability. Flip-flops with poor controllability measures are selected for scan. *Trishler* [4] describes a method whereby flip-flops which are not easily controllable are included in the incomplete scan path. **Limitations**: the effectiveness of this method, evaluated in terms of scan overhead, fault coverage, etc., depends entirely on testability analysis which, depending on the heuristics, may not accurately model the problems faced during test generation [16]. The correlations between testability measures and test generation costs have not been well established.

In partial scan methods based on *structural analysis* [7] [17], the sequential circuit is transformed into a directed graph, whose vertices represent flip-flops, primary inputs and outputs, and whose arcs represent the combinational paths. Heuristics are used to select a minimal set of flip-flops that eliminate the cycles in the graph. The premise behind this approach is the assessment that flip-flops in a loop are hard to control and observe. **Limitations**: such techniques operate solely on the network topology and do not explicitly analyze the behaviour of the sequential circuit. Thus, there is no guarantee that the selected scan elements are the most non-controllable, which may lead to the selection of scan registers which do not provide sufficiently high fault coverage [18]. Recently, *Xiang et al.* [19] suggested that breaking all the cycles in the graph may not be necessary, since some of the cycles do not influence the complexity of test generation. They presented algorithms that exploit the circuit state information to identify those cycles in the graph that complicate test generation significantly. Corresponding flip-flops that break only these cycles were selected for partial scan.

In the partial scan methods based on *test generation* [5] [6], tests are first generated for a large number of faults. Then, for each undetectable (or aborted) fault, a set of flip-flops is found, such that making those flip-flops observable and controllable makes the fault detectable. The incomplete scan path then utilizes a minimal subset of memory elements which influenced the easy detection of as many faults as possible. Recently, *Sharma et al.* [20] proposed a technique that uses a test generator to perform multi-hop reachability in order to identify the hard-to-reach states of the circuit. Targeting these hard-to-reach states using scan allows the detection of hard-to-detect faults. **Limitations:** such techniques incorporate the cost of test generation as well as the cost of calculating minimal sets of registers to scan, and are thus time and compute intensive. Also, these techniques rely heavily on test generators. Use of an unsophisticated test generator that aborts too many faults may result in some unnecessary scan registers. The *fault oriented* partial scan design approach [10] is also test generator dependent. Structural analysis of the circuit is enhanced by focusing on the untestable and aborted faults. Thus, it also suffers from the above drawbacks.

Considering the fact that the above techniques do not incorporate the cost of scan design in selecting scan flip-flops, an optimization based approach was presented by *Chikermane et al.* [9] that formulates the partial scan register selection technique as an *optimization problem*. As the use of scan flip-flops results in layout and delay overheads, it is important to choose a set of flip-flops which give the best improvement in testability, while keeping the cost of scan design bounded. Based on this idea, a tool called OPUS [9] was developed which is actively used in both academia and industry. However, the testability criteria for selection of scan flip-flops is based on testability heuristics such as the SCOAP controllability/observability measures [15] or on structural parameters of the circuit such as the number of cycles/loops in the circuit, the length of directed cycles, etc. Thus, this approach also suffers from the limitations outlined above.

There are a few other miscellaneous partial scan approaches based on empirical models [13] or some other heuristic estimates of flip-flop non-controllability [12], etc., which also suffer from one or more drawbacks outlined above.

In this paper, we present a new approach to the partial scan problem that thoroughly analyzes the behaviour of the sequential circuit and its state encoding to evaluate the non-controllability measures of the state registers. To analyze the behaviour of the underlying finite state machine (FSM) of the sequential circuit over the complete state space, we use *implicit techniques* for FSM traversal. Using implicit state enumeration, we implicitly exercise all the state transitions and visit all the states in the reachable state set of the machine. State information thus obtained is used to identify the non-controllable state registers. We present algorithms to select non-controllable state registers for scan using implicit state enumeration and present the results which illustrate the effectiveness of our technique over a large set of benchmarks.

The paper is organized as follows. The next section highlights the contribution of this paper and indicates how and why our approach is different from other partial scan approaches. Section III reviews basic terms and definitions related to implicit state enumeration techniques and sequential circuit testing. Section IV describes the motivation behind this work. Section V describes techniques to identify non-controllable registers for partial scan using implicit FSM traversal. In Section VI an algorithm, SIMPSON, is proposed and the results are presented and analyzed. Limitations of the SIMPSON algorithm are discussed and subsequently, extensions to the algorithm are presented. In Section VII we review state space decomposition and approximate FSM traversal techniques. Section VIII describes how we can exploit approximate implicit FSM traversal techniques to evaluate the non-controllability measures of the flip-flops for partial scan design. An algorithm, SAMSON, is presented and its effectiveness is analyzed over the larger circuits of the ISCAS'89 benchmark suite. Section IX points out possible future research directions and concludes the paper.

II. Contribution of this Research

In this paper, we present a comprehensive approach to analyze the sequential behaviour of a circuit to accurately evaluate the non-controllability of flip-flops in order to make a judicious choice of scan registers. It is well understood that testability of a sequential circuit is inherently captured by its state transition behaviour and its encoding [21][19][18]. In order to accurately assess the non-controllability measures of the flip-flops, we need to thoroughly analyze the behaviour and the encoding of the underlying FSM of the circuit. Implicit state enumeration is a technique that can be exploited to analyze the behaviour of the sequential circuit for testing purposes.

Our model to evaluate the controllability measures of flip-flops is based on a systematic behavioural analysis of the underlying FSM of a sequential circuit. By implicitly enumerating the states of a machine, we accurately evaluate the non-controllability of flip-flops by determining *exactly* what values can or cannot be stored, or are difficult to store in the state registers. By doing so, we not only target the untestable faults due to state unreachability of the machine, but also the difficult-to-test faults caused by difficult-to-control flip-flops. The values observed in the flip-flops during the implicit FSM traversal (*i.e.,* the states of the flip-flops) are used to evaluate flip-flop controllability measures to support the testability analysis. This technique is programmed as an algorithm called SIMPSON and we analyze its effectiveness by carrying out extensive experiments. The experimental results clearly reflect the accuracy of the proposed flip-flop controllability measures and demonstrate the superiority of our approach over conventional state-of-the-art partial scan design approaches. Specifically, as compared to the techniques that use structural parameters of the circuit [7][17] and/or SCOAP controllability/observability heuristics [15] to select scan registers, our technique results in fewer scan registers and provides higher fault coverage.

However, implicit state enumeration of sequential circuits with a large number of state registers is often infeasible. The underlying state space of a sequential circuit is potentially exponential in the number of state registers. For circuits with a large number of flip-flops, not only does it take unacceptable amount of time to traverse the entire reachable state space, but storage and processing of the set of reachable states also becomes infeasible. *Approximate reachability analysis* techniques have been proposed in literature [22] [23] to overcome the space and time overheads associated with exact implicit FSM traversal methods. These techniques partition a large FSM into smaller sub-FSMs and perform reachability analysis on these smaller sub-machines. As a result, the reachable states are approximated by an upper-bound; the over-estimate of the reachable states is computed as a product of smaller subsets.

We exploit the power of approximate reachability analysis techniques to analyze the state-space of circuits that contain a large number of flip-flops. Using intelligent techniques to partition a large machine into smaller ones, and then performing approximate implicit FSM traversal, we are able to *estimate* the non-controllability metrics of flip-flops for partial scan design. We present an algorithm SAMSON for this purpose and demonstrate its effectiveness on some of the larger circuits in the ISCAS'89 benchmark suite. Even though approximate FSM traversal results in some "loss of information" of state machine reachability, we demonstrate by experiments that this loss of information does not significantly affect the proposed flip-flop testability analysis criteria. Using the SAMSON algorithm, we were able to select partial scan flip-flops for large sequential machines and provide high fault coverage. Specifically, as compared to the scan techniques based on structural analysis [9] [10] and/or SCOAP testability heuristics [15], the same number of scan flip-flops selected by SAMSON provides higher fault coverage.

III. PRELIMINARIES

In this section, we review basic terms and definitions related to Boolean functions, finite state machines, sequential circuits and sequential circuit testing, and summarize the breadth-first traversal techniques for implicit state enumeration as used in this paper.

A. Boolean Functions and Boolean Function Vectors

An n-input and m-output Boolean function F is a mapping from an n-dimensional Boolean space to an m-dimensional Boolean space, $F : B^n \to B^m$, where $B = \{0, 1\}$. B^n is the domain and B^m the co-domain of F. If $m > 1$, then F is a multiple output function; it can be represented as a vector of single output Boolean functions called *Boolean Function Vector* (BFV).

The *support* of a Boolean function is the set of variables it depends upon. A *literal* represents a variable or its complement. A conjunction of a set of literals is called a *cube*, and it represents a point, or a set of points, in the Boolean space. If a cube has literals of all variables in the support of the function, the cube is a *minterm* and it represents a point in the domain of the Boolean function.

B. Sequential Circuits and Finite State Machines

We consider synchronous sequential circuits composed of combinational logic gates and flip-flops, where all the flip-flops are synchronized by the same clock. We assume that the circuit has a known initial state, and can always be driven to that initial state either by explicit reset circuitry, or by application of a synchronizing sequence.

Associated with a sequential circuit is its underlying finite state machine (FSM) which inherently captures the behaviour of the circuit. In mathematical terms, a completely specified, deterministic finite state machine of Mealy type is a 6-tuple $\langle \sum, O, S, S^0, \Delta, \Lambda \rangle$, where

- $\sum$ is the input alphabet, *i.e.*, a finite, non-empty set of input values,
- O is the output alphabet,
- S is the finite set of states,
- $S^0 \subseteq S$ is the set of initial states,
- $\Delta : S \times \sum \rightarrow S$ is the next state transition function,
- $\Lambda : S \times \sum \rightarrow O$ is the output function.

The behaviour of an FSM can be represented by a *State Transition Graph* (STG) that depicts the transitions that the machine can make between its states under the application of some input, and the output that it generates. The STG of an FSM is a directed graph $G = (V, E)$, where V is the set of vertices and E is the set of directed edges such that $V = S$, $(s_1, s_2) \in E, s_1 \in S, s_2 \in S$, if $\Delta(s_1 \in S, x \in \sum) = s_2$. Associated with every edge of an STG is a label i/o, where $i = x$ and $o = \Lambda(s, x) \in O$.

Note that Δ and Λ are multi-output Boolean functions and are represented by BFVs. These BFVs implicitly define all the state transitions of the given FSM. The states of the machine that can be reached from the initial state by application of any input sequence are termed as *reachable* or *valid* states. A state that cannot be reached from a reset state is called an *unreachable* or *invalid* state.

C. Cofactors and Quantification

Given an m-variable Boolean function $f(x_1, \ldots, x_m)$, the *positive cofactor* of f with respect to x_i is $f_{x_i} = f(x_1, x_2, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_m)$. Similarly, $f_{\overline{x}_i} = f(x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_m)$ stands for the corresponding *negative cofactor*. Given an m-variable Boolean function $f(x_1, x_2, \ldots, x_m)$, the *existential abstraction* of f with respect to x_i is:

$$\exists_{x_i} f = f_{x_i} + f_{\overline{x}_i} \quad (1)$$

Given $f(s,x) = (s_1, \ldots, s_n, x_1, \ldots, x_m)$ the existential abstraction with respect to a set of variables is defined as:

$$\exists_x f(s,x) = \exists_{x_1}(\cdots(\exists_{x_{m-1}}(\exists_{x_m} f(s,x)))) \tag{2}$$

D. Sets and Characteristic Functions

Given a Boolean space B^l, a set of minterms in B^l can be represented by a *characteristic function* of S, $\chi_S(s)$ which satisfies the property: $s \in S \iff \chi_S(s) = \mathbf{1}$, for all $s \in B^l$. In other words, a minterm of B^l which evaluates χ_S to **1** (i.e., an on-set minterm) is an element of S. The characteristic function of the universe is tautology and that of a null set is **0**. ROBDDs [28] [29] are often exploited as implicit set representations in order to represent their characteristic functions. In the context of implicit state enumeration, the sets of reachable and unreachable states of a machine are represented by their respective characteristic functions. Henceforth we will use the terms sets and their characteristic functions interchangeably, without diluting their meaning.

E. Symbolic Image Computation

Definition III.1: Given a BFV $F : B^n \rightarrow B^m$, and a domain subset $C \subseteq B^n$, the **image** of C under F is defined by:

$$IMAGE(F,C) = \{F(x) | \forall x \in C\} \tag{3}$$

If $C = B^n$, the image of C under F is also called the range of F.

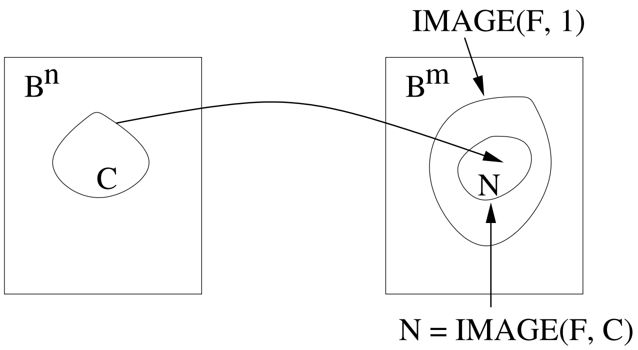


Fig. 1. Symbolic IMAGE computation.

Let $t_i = \Delta_i(s,x)$, $1 \leq i \leq n$, be the i^{th} encoded next state transition function of a given encoded FSM. Let s and x be the coding vectors for the states and inputs, respectively. Let n be the total number of state encoding bits (number of registers). A symbolic state set $C(s)$ is mapped by $\Delta(s,x)$ into a state set N in the range of the functional vector Δ. The set of such co-domain points represents the *image* of C under the transition function Δ. Fig. 1 depicts the image of the domain subset C under the transition function F. In the symbolic approach, the image is computed using *transition relations*.

Definition III.2: Given a deterministic transition function $t = \Delta(s, x)$, where s represents the present state variables, x represents the input variables, and t the next state variables, the corresponding **transition relation** $T(s, x, t)$ is defined by:

$$T(s, x, t) = \prod_{i=1}^{i=n} (t_i \equiv \Delta_i(s, x)) \tag{4}$$

Note that t_i takes the same values that are evaluated by the i^{th} encoded transition function Δ_i. In the binary case, the symbol "$\equiv$" stands for the XNOR operation.

Given the above definitions, we can easily compute the image of a domain sub-set $C(s)$ (say, initial state of the FSM) under F (say, the transition function of the FSM) as:

$$N(t) = IMAGE(T, C) = \exists_x \exists_s T(s, x, t) \cdot C(s) \tag{5}$$

In other words, the image of a set of initial states under the transition function of an FSM proceeds as follows. First, we compute the transition relation $T(s, x, t)$ from the circuit equations. Then we compute its conjunction with the characteristic function of the set of initial states $C(s)$. Then, we existentially abstract all the input (x) and the present state (s) variables to obtain the image $N(t)$. This image, $N(t)$, is the set of next states that are directly reachable from the set of initial states in one transition.

F. Implicit State Enumeration using Symbolic Image Computations

Traversing an FSM means executing symbolically all its transitions. If a state transition diagram is available, an explicit traversal means following all directed paths whose tail is the initial state, thus detecting all reachable states. If the FSM is described by a synchronous logic network, a traversal means determining all possible value assignments to state variables that can be achieved, starting from the assignment corresponding to the reset state. In this case, reachable and unreachable state sets are represented *implicitly* by functions over the state variables.

The technique to implicitly compute the set of reachable and unreachable states of an FSM, as originally presented in [26], has been improved over the years [27] [22] [23] which has dramatically extended the realm of problems for which reachability analysis can be carried out. These approaches are based on a breadth first traversal (BFS) of the entire state machine. The key ideas of the method are the use of symbolic image computations to perform the BFS traversal, and the use of BDDs as implicit set representations to store and process the set of reachable and unreachable states. In what follows, we briefly describe how states are implicitly enumerated during BFS traversal using symbolic image computations.

Algorithm 1 describes the BFS traversal procedure to enumerate the reachable states of an FSM using symbolic image computations. Initially, $from^0 = S^0$, is the characteristic function of the initial state set. to^i represents the states reached in the i^{th} iteration of BFS traversal. It is evaluated by computing the symbolic

Inputs: Transition function BFVs (Δ), Initial State (S^0).
$from^0 = reached = S^0$;
$i = 0$;
while TRUE **do**
$i++$;
to^i = IMAGE($\Delta, from^{i-1}$);
$new^i = to^i \cdot \overline{reached}$;
if $new^i == 0$ **then**
return ($reached$);
end if
$reached = reached + new^i$;
$from^i = new^i$;
end while

Algorithm 1: Procedure BFS_TRAVERSAL: Implicit state enumeration using image computations.

image of the domain subset $from^{i-1}$ under the transition function Δ of the FSM. The characteristic function $reached$ represents the set of states that have been reached so far from the initial states. Some states in to^i may have been reached in previous iterations, so a set difference operation with $reached$ is required to compute new^i, the new states reached in this iteration for the first time. If no *new* states are reached in any iteration, the procedure terminates and $reached$ is guaranteed to contain all reachable states of the machine[1].

G. Sequential Circuit Testing Terminology

A gate has an input/output *stuck-at-1 (stuck-at-0)* fault if the logical value associated with the input/output is 1(0), independent of the values at the primary inputs. A fault f is said to be *detectable* if there exists an input sequence Y such that for every pair of initial states S and S^f of the fault-free and faulty circuits, respectively, the response Z of the fault-free circuit to Y is different from the response Z^f of the faulty circuit at some time unit [24]; otherwise it is *undetectable*.

In general, the problem of sequential circuit test generation involves finding primary input sequences which can excite a fault and propagate its effect to the primary outputs. Thus, to detect a single stuck-at fault, first the *fault excitation state* has to determined, the circuit has to be driven to the fault excitation state, and finally the fault effect has to be propagated to the primary outputs [25]. The process of finding an input sequence which takes the machine from the reset state (or an unknown initial state) to the fault excitation state is called *state justification* and such a sequence is called a *justification sequence*. An assignment to primary input and present state lines that propagates the effect of a fault at either the primary outputs or next state lines is called the *excitation vector* for the fault. Note that the present state part of the excitation vector is the excitation state for the fault. For an excitation vector to be valid, the excitation state for the vector should contain at least one reachable state.

[1] Note that the BFS_TRAVERSAL procedure is essentially is least fixed point computation.

IV. Motivation

Let us now motivate the importance of analyzing the circuit state information in order to identify non-controllable state registers for partial scan design. Consider the circuit shown in Fig. 2. Using the sequential circuit test generator HITEC [30], the circuit was found to be 77% testable. The untestable faults are depicted in the figure (marked by $\otimes$ and $\times$). In order to improve upon the fault coverage for this circuit, we decided to use partial scan. The program OPUS [9] was used to select partial scan registers. OPUS selected register r_1 for partial scan, resulting in 84% fault coverage (refer to Table I). The faults marked by $\otimes$ became testable as a result of making r_1 controllable; however, the faults marked by $\times$ could still not be tested.

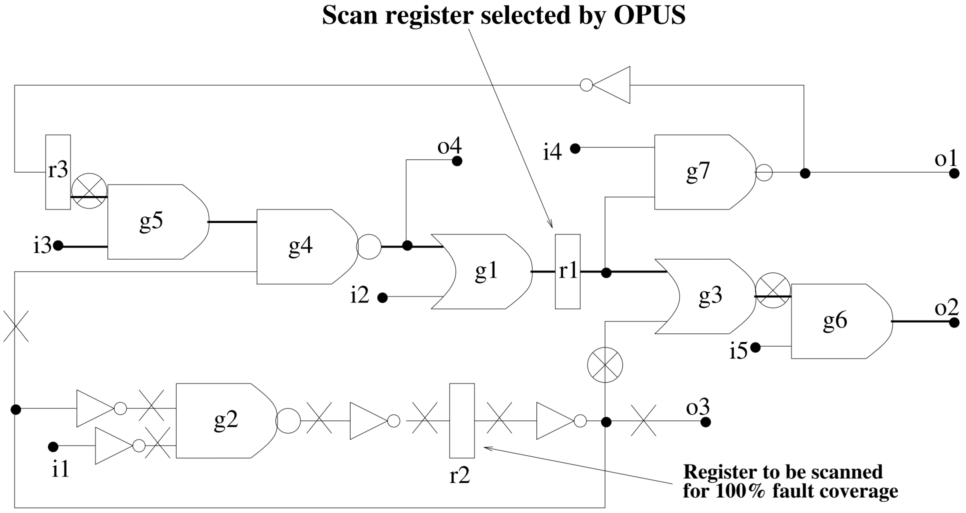


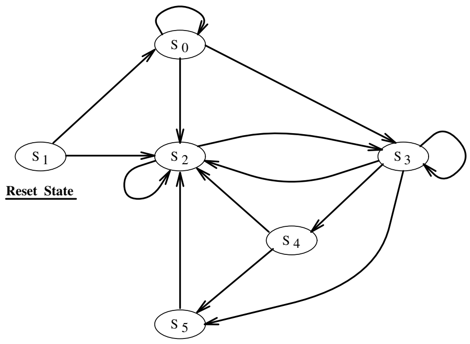
Fig. 2. Example circuit depicting the scanned registers.

TABLE I

Testability data for the example when r_1 and r_2 are scanned.

	OPUS Scan r_1	Scanning r_2
Total # of faults	44	44
# of detected faults	37	44
Percentage fault cov.	84%	100%
# of untestable faults	7	0
# of vectors	13	21

Scanning register r_3 gave the same results (84% fault coverage). However, by selecting r_2 for scan, we were able to achieve 100% fault coverage. The above observation leads us to the following question: How do we know that register r_2 is the best register to scan? To answer this question, let us analyze the state transition behaviour and the encoding of the underlying FSM of this circuit shown in Fig. 3. We can see from the STG that once the machine is in one of the states determined by the set $\alpha = (S_2, S_3, S_4, S_5)$, it cannot make a transition to any of the states in the set $\beta = (S_0, S_1)$. Once the machine enters the set α, it remains within α, and there is no path in the STG from α to β. Thus, for the encoding in Fig. 3(b), it is not possible to change the value of r_2 from 0 to 1. Clearly, this sort of a behaviour of the underlying FSM of this sequential circuit manifests itself in terms of the non-controllability of register r_2. This, in turn, causes the untestable faults in the circuit.



(a)

State	Code		
	r_1	r_2	r_3
S_0	1	1	0
S_1	0	1	0
S_2	1	0	0
S_3	1	0	1
S_4	0	0	1
S_5	0	0	0

(b)

Fig. 3. STG of the example circuit and the corresponding state encoding.

The reason that OPUS failed to identify r_2 as the best register to scan is that its algorithm is based predominantly on the *structural analysis* of the sequential circuit. No information about sequential behaviour of the circuit (state transitions and the encoding) is used for the selection of scan registers. It becomes clear, however, that the state transition information and the encoding of a sequential circuit are important factors in determining the non-controllability of state registers. Thus, in order to select partial scan registers, there is a need for techniques that analyze the behaviour of a sequential circuit over *multiple clock cycles*. Over and above, such techniques have to be computationally efficient in memory and time requirements. Implicit state enumeration is an efficient technique that can be exploited to analyze the behaviour of the sequential circuit for testing purposes.

Motivated by the above observation, we investigated how implicit state enumeration could be used to assess flip-flop controllability measures. Specifically, we can use this analysis to: i) identify non-controllable flip-flops, ii) identify difficult-to-control flip-flops, and iii) exploit the information about the unreachable (or illegal) states, in order to select the best possible set of scan flip-flops in a systematic and non-greedy fashion.

V. Identifying Partial Scan Registers Using Implicit State Enumeration

There have been a few attempts to use implicit state enumeration to analyze the circuit behaviour and exploit it for testing purposes. *Cho et. al.* [31] used implicit state enumeration for test generation and redundancy identification. They used implicit state enumeration to perform reachability analysis and used this information during the *state justification* and *state differentiation* phases of test generation. *Long et.*

al. [32] also proposed a BDD-based method to enumerate the unreachable states and used this information to identify the sequentially untestable faults. However, none of the above works targeted the partial scan problem.

In this section, we present a new testability analysis framework that uses implicit state enumeration to analyze the circuit behaviour in order to evaluate non-controllability of flip-flops. By carrying out reachability analysis on the circuit, not only do we manage to enumerate the reachable and unreachable states, but also pin-point the non-controllable and difficult-to-control flip-flops. We use this flip-flop controllability information to correctly target the registers to be scanned for partial scan design.

A. Non-controllable Registers: Missing Transitions

We shall now explain, by means of an example, how non-controllable flip-flops can be identified by implicitly traversing the underlying state machine of a circuit. Consider again the example circuit and its STG shown in Fig. 2 and Fig. 3, respectively. Following are the next state equations (or next state transition functions) of the example circuit.

$$r_1[t] = i_2[t] + \overline{i_3[t]\ r_3[t-1]} + r_2[t-1] \tag{6}$$

$$r_2[t] = \overline{i_1[t]}r_2[t-1] \tag{7}$$

$$r_3[t] = i_4[t]r_1[t-1] \tag{8}$$

From these encoded next state transition functions, the corresponding transition relations can be readily derived as follows:

$$\begin{aligned} T(s,x,t) &= \prod_{i=1}^{i=3}(t_i \equiv \Delta_i(s,x)) \\ &= (t_1 \equiv i_2[t] + \overline{i_3[t]\ r_3[t-1]} + r_2[t-1])(t_2 \equiv \overline{i_1[t]}r_2[t-1])(t_3 \equiv i_4[t]r_1[t-1]) \end{aligned} \tag{9}$$

The given initial state of this circuit is $S_1 = \langle r_1 = 0, r_2 = 1, r_3 = 0 \rangle$. The characteristic function of the set representing the initial state can is represented as:

$$C(s) = \overline{r_1[t-1]} \cdot r_2[t-1] \cdot \overline{r_3[t-1]} \tag{10}$$

The set of states directly reachable from the initial state $C(s)$ under the transition function Δ can be computed as

$$N(t) = IMAGE(T,C) = \exists_x \exists_s T(s,x,t) \cdot C(s).$$

Substituting the value of $T(s,x,t)$ obtained from (9) and the value of $C(s)$ obtained from (10), and quantifying *w.r.t.* input ($x = \{i_1[t], i_2[t], i_3[t], i_4[t], i_5[t]\}$) and present state ($s = \{r_1[t-1], r_2[t-1], r_3[t-$

$1]\})$ variables, we get:

$$N(t) = t_1 \cdot t_2 \cdot \overline{t_3} + t_1 \cdot \overline{t_2} \cdot \overline{t_3} \tag{11}$$

This implies that the forward image of the state $S_1 = \langle 010 \rangle$ under the given transition function of the FSM (or equivalently, the set of states reachable in one step from the initial state S_1) is $\langle S_0 = (110), S_2 = (100) \rangle$. This can be verified by the STG and the encoding of the circuit shown in Fig. 3. The above reachability computations when performed iteratively lead us to the following state traversal:

$$(S_1 : 010) \rightarrow (S_0 : 110, S_2 : 100) \rightarrow (S_3 : 101) \rightarrow (S_4 : 001, S_5 : 000).$$

That is, from S_1, the directly reachable states in one step are S_0 and S_2. From S_0 and S_2, the directly reachable state in one step is S_3, and finally S_4 and S_5, at which point the entire reachable state space has been explored. Let us now examine the FSM traversal trace for register r_2.

$$(S_1 : r_2 = 1) \rightarrow (S_0 : r_2 = 1, S_2 : r_2 = 0) \rightarrow (S_3 : r_2 = 0) \rightarrow (S_4 : r_2 = 0, S_5 : r_2 = 0).$$

Notice that register r_2 can change its value from 1 to 0; *i.e.* it is possible to get a falling transition $(1 \rightarrow 0)$ at the output of r_2. However, a rising transition $(0 \rightarrow 1)$ is missing. In other words, once r_2 gets the value 0, it can never obtain the value 1. Thus, register r_2 is *unsettable* to logic value 1 from logic value 0. If certain registers cannot make some transition, then it may not be possible for a test generator to justify the values in the registers during its state justification phase. This, in turn, may render some faults sequentially untestable. Such registers are surely good candidates for scan (verify from Table I that scanning register r_2 leads to a fully testable circuit). Thus, while implicitly enumerating all the states in the set of reachable states of a machine, by observing the values in all the flip-flops we can identify those registers that do not make some transition and select them for partial scan.

B. *Difficult-to-Set Flip-Flops*

Hartanto et. al. [33] had suggested that identifying the states that are difficult to traverse by the test generation tools can significantly speed up test generation for sequential circuits. They proposed a method to identify those flip-flops which were difficult to control. They defined the difficult-to-set flip-flops as follows:

Definition V.1: A state element s_d in a sequential machine M is **difficult-to-set** to a value v if a test generator, under a specified time and backtrace limit, does not find an input sequence that can bring the machine M from its fully unspecified initial state (consisting of all unknown values in the flip-flops and corresponding to the entire state space) to a state where the value of s_d is v.

It was indicated that the method to identify difficult-to-set flip-flops was dependent on the test generator used. To identify these difficult-to-set flip-flops they had to modify the circuit by creating a primary output at each flip-flop. A deterministic test pattern generator was then used to test for stuck-at-0 and stuck-at-1 faults at these lines transformed into primary outputs. The difficult-to-set flip-flops were identified by observing the values at the output of each flip-flop.

We present a method to identify such difficult-to-set flip-flops that does not require any circuit modifications. Also, our model to identify difficult-to-set flip-flops does not depend on any test generator. While implicitly enumerating the reachable states of a circuit, for each register, we evaluate the largest number of states successively traversed in a sequence, for which the flip-flops do not change their values. In other words, for each register, we record the length of the longest sequence of 0s and 1s (whichever is greater), which indicates the difficulty in setting a flip-flop to a particular value. To find such difficult-to-set flip-flops, we define a term, *degree of unsettability* of a flip-flop.

Definition V.2: The **degree of unsettability** of a flip-flop is defined as the length of the longest sequence of states in the implicit traversal trace of an FSM, for which a flip-flop does not change its value.

Scanning such difficult-to-set flip-flops, identified by their degree of unsettability, would help in detecting the difficult-to-test faults. The reason for this can be explained as follows: If a test generator has to justify a value in a register, say a value 1, and it encounters a backtrace path of a long sequence of 0s, then it may have to backtrace many time frames in search for a value 1. In doing so, it may abort such faults and classify them as difficult-to-test, which may lead to reduced fault coverage.

C. Sequentially Untestable Faults: Targeting the Illegal States

Knowledge of state space is known to be quite useful in causing early backtraces in test generation. Test generators often spend a significant amount of time on undetectable faults as they eventually have to backtrace a large subset of the state space in order to prove that the values in the registers cannot be justified due to the unreachable states. A powerful technique for proving the undetectability of the faults is the identification of illegal states. Formal methods [26] [27] [31], and other recent approaches based on BDDs [32] are widely used to identify illegal states. After computing the reachability information using implicit state enumeration on an FSM, all the reachable states are stored implicitly in a BDD. Complementing this BDD results in the set of all the unreachable states.

We use the information on the unreachable states to target the selection of scan flip-flops. Fig. 4 enumerates all the unreachable states of an MCNC benchmark example. All these states are stored implicitly using a BDD which represents the characteristic function of this set of unreachable states.

It is clear from the list of all the unreachable states that register R_1 would be a good candidate for scan. This is because in all the unreachable states, R_1 takes the value 1. Hence, in the reachable state set, it would be difficult to set R_1 to 1. Let us denote the characteristic function of the set of unreachable states by $F_{illegal}$. In this case, $F_{illegal}$ is *unate* in variable R_1. Now the problem of identifying the non-controllable registers

	R1	R2	R3	R4	R5
	1	0	0	0	0
	1	0	0	1	0
	1	0	0	1	1
	1	0	1	0	0
	1	0	1	0	1
	1	0	1	1	0
	1	0	1	1	1
	1	1	0	1	0
	1	1	1	0	0
	1	1	1	0	1
	1	1	1	1	0
Degree of unateness:	11	3	3	1	3

Fig. 4. Illegal states of a benchmark circuit.

from the illegal state set could be transformed into one of identifying that state variable over which $F_{illegal}$ is unate.

However, no claims can be made about the unateness of the characteristic functions of the unreachable state set of an FSM in general. Some functions may not be unate in any of the variables in their support set, whereas some may be unate in all the variables in their support. Hence, it is necessary to define (with some abuse of terminology) the *degree of unateness* for each state variable in order to measure the non-controllability of the registers.

Definition V.3: Let $F_{illegal}$ represent the characteristic function of the set of all the illegal states of an FSM. Let R_i be a variable in the support of $F_{illegal}$. The absolute value of the difference between the number of zeros and ones that a variable R_i can take in the domain of $F_{illegal}$ is defined as the **degree of unateness** of variable R_i.

Using the above measure of non-controllability, we can select the register for partial scan that has the highest degree of unateness. However, two or more registers may have the same measure of the degree of unateness (as is the case with registers R_2 and R_3 in Fig. 4). For this reason, we need to simultaneously take into account their degree of unsettability in order to differentiate between their relative non-controllability measures.

D. Overall Non-Controllability Measures of Flip-Flops

From the above mentioned techniques, we can now define the overall non-controllability measure of each flip-flop R_i of the circuit as follows:

$$Non-controllability(R_i) = degree\_of\_unsettability(R_i) + degree\_of\_unateness(R_i) \tag{12}$$

In other words, the overall non-controllability measure of flip-flops is the sum of their respective degrees of unsettability and unateness. The addition (sum) of the unsettability and the unateness measures properly addresses the issues of both state machine reachability and unreachability. If all the states of the circuit are reachable, *i.e.* the unreachable state set is empty, the degree of unateness is zero, and the degree of unsettability dictates the non-controllability measure. In contrast, for a machine that has a much higher number of illegal states than legal states, chances are that the degree of unateness would bias the non-controllability measure in its favour. The "middle-ground" would be achieved when the machine has comparable reachable and unreachable state set size, in which case both unsettability and unateness measures would evenly contribute to the non-controllability measure of the flip-flops.

VI. The SIMPSON algorithm

Inputs: Sequential circuit, number of registers to scan.
Outputs: Scan registers listed in decreasing order of their non-controllability.

```
from^0 = reached = S^0;
i = 0;
while TRUE do
  i++;
  to^i = IMAGE(Δ, from^(i-1));
  new^i = to^i · NOT(reached);
  for each state variable r_j do
    record_if_transitions_present_or_missing(r_j, new^i);
    compute_degree_of_unsettability(r_j, new^i);
  end for
  if new^i == 0 then
    break;
  end if
  reached = reached + new^i;
  from^i = new^i;
end while
/* FSM traversal completed */
for each state variable r_j do
  if missing transition for r_j then
    scan state variable r_j;
  end if
end for
illegal_states = bdd_complement(reached);
for each state variable r_j do
  compute_degree_of_unateness(r_j, illegal_states);
  non-controllability(r_j) = degree_of_unsettability(r_j) + degree_of_unateness(r_j);
end for
order state variables in terms of their non-controllabilities; /* Sorting */
output the required scan registers;
```

Algorithm 2: SIMPSON: Scan Register Selection using Implicit State Enumeration.

Based on the flip-flop controllability analysis techniques outlined above, we present an algorithm, SIMPSON (Scan register selection using IMPlicit State enumeratiON), that uses implicit state enumeration to analyze the behaviour of the sequential circuit in order to select the non-controllable state registers for partial scan.

The algorithm SIMPSON, shown in Algorithm 2 proceeds as follows. Using symbolic image computation, the FSM is traversed implicitly. During each reachability step of the FSM traversal, both rising and falling transitions on each register are recorded. Also, during the FSM traversal the *degree of unsettability* (length of the longest sequence of 0s or 1s, whichever is greater) for each memory element is recorded. After the completion of FSM traversal, if a register is found to be missing either a rising or falling transition, it is selected for scan. Furthermore, from the implicitly enumerated reachable states, the unreachable states of the circuit are computed. From this set of unreachable states, the *degree of unateness* of each state variable is computed. Subsequently, the overall non-controllability measure for each flip-flop (sum of their unateness and unsettability measures) is computed. The state variables are then sorted in terms of the decreasing order of their degree of non-controllability. The algorithm lists the memory elements of a sequential circuit in decreasing order of the degree of their non-controllability with the most non-controllable memory element at the top of the list.

The above algorithm was programmed within the VIS [34] tool-set. VIS provides a robust platform for performing reachability analysis using symbolic image computations. The sets of states and transition relations of the FSM are stored in memory using ROBDDs. The CUDD [35] package was used for storage and manipulation of sets of states, Boolean functions, and relations. Using SIMPSON, extensive experiments on a set of MCNC and ISCAS'89 benchmarks were carried out on a Sun UltraSparc5 workstation with 320MBytes of RAM.

TABLE II

NON-CONTROLLABILITY MEASURES AND FAULT COVERAGE STATISTICS OBTAINED BY SCANNING EACH LATCH, INDIVIDUALLY, FOR BENCHMARK $s832$.

Latch order	Non-controllability measure	Fault cov. (scan)	# Vectors
G38	28	98.39% (856/870)	881
G41	19	97.47% (848/870)	552
G40	18	97.35% (847/850)	679
G42	17	97.24% (846/870)	692
G39	16	95.28% (829/870)	639

Let us first demonstrate how the flip-flop non-controllability measures computed by SIMPSON distinguish between the relative non-controllabilities of the flip-flops of a circuit. Consider the following testability statistics corresponding to the ISCAS'89 benchmark circuit $s832$. This circuit has a total number of 5 flip-flops. Its non-scan fault coverage is 93.4% (813 detected faults/870 total faults). Its full-scan fault coverage is 98.39% (856 detected faults/870 total faults). Table II lists the flip-flops ordered by SIMPSON according to the degree of their non-controllabilities. The respective ATPG statistics obtained by scanning each of these latches individually are also reported. Flip-flop G38 has the highest non-controllability measure. By scanning this latch and generating tests using the ATPG tool HITEC [30], 98.39% fault coverage was achieved. The flip-flop G41 is next most non-controllable flip-flop. Scanning this flip-flop allows us to achieve 97.47% fault coverage. This fault coverage is less than that achieved by scanning the most non-controllable flip-flop G38, but more than that achieved by scanning the ones down the order. From the table it is clear that the estimated non-controllabilities are in accordance with their observed test generation and fault coverage statistics.

Let us now observe the experimental results carried out using SIMPSON over a large number of the ISCAS'89 benchmark circuits and analyze how they compare with the ones obtained by using contemporary partial scan approaches. The results depicted in Table III are compared with both the structural analysis (cycle breaking) and the SCOAP testability analysis approaches used by OPUS.

TABLE III

EXPERIMENTAL RESULTS: APPLYING THE SIMPSON ALGORITHM.

Circuit	Total FFs	OPUS: Cycle Breaking				OPUS: SCOAP Testability Option				SIMPSON			
		Scan FFs	Fault Cov	CPU Time	# of Vects	Scan FFs	Fault Cov	CPU Time	# of Vects	Scan FFs	Fault Cov	CPU Time	# of Vects
bbsse	6	3	99.66%	< 1 ms	69	-	–	–	–	1	99.66%	10 ms	131
false	3	1	85%	< 1 ms	10	-	–	–	–	1	100%	10 ms	21
ex1	5	4	100%	< 1 ms	152	-	–	–	–	3	100%	10 ms	304
rie	5	3	100%	< 1 ms	171	-	–	–	–	2	100%	4 ms	89
planet	7	5	100%	< 1 ms	252	-	–	–	–	4	100%	10ms	64
imec10	8	6	100%	< 1 ms	350	-	–	–	–	5	100%	40 ms	716
s386	4	3	100%	< 1 ms	82	3	98.18%	< 1 ms	194	1	100%	10 ms	157
s820	5	1	97%	< 1 ms	542	1	97.06%	< 1 ms	700	1	100%	40 ms	894
s832	5	4	98.39%	< 1 ms	440	4	98.39%	< 1 ms	327	1	98.39%	20 ms	581
s1488	6	3	96.5%	< 1 ms	1033	3	100%	< 1 ms	493	3	100%	20 ms	455
s1494	6	2	99.88%	< 1 ms	472	2	99.2%	< 1 ms	776	2	99.8%	20 ms	472
s510	6	5	100%	< 1 ms	392	5	100%	< 1 ms	122	3	100%	60 ms	480
s344	15	1	94.83%	< 1 ms	85	1	97.66%	< 1 ms	70	1	97.66%	120 ms	70
s349	15	5	98.86%	< 1 ms	97	5	99.14%	< 1ms	80	3	99.57%	20ms	91
s420	16	0	98.3%	< 1 ms	36	–	–	–	–	0	98.3%	10 ms	36
s641	19	7	100%	< 1 ms	172	7	99.36%	< 1 ms	199	3	99.37%	420 ms	236
s713	19	5	91%	< 1 ms	203	5	92.94%	< 1 ms	241	3	93.1%	430 ms	257
s444	21	9	95.99%	< 1 ms	184	9	94.94%	< 1 ms	115	5	98.1%	530 ms	1321
s400	21	9	97.4%	< 1 ms	186	9	96.23%	< 1 ms	151	6	97.42%	590ms	723
s382	21	9	98.75%	< 1 ms	166	9	97.49%	< 1 ms	131	6	98.75%	620 ms	693
s953	29	5	100%	< 1 ms	309	5	100%	< 1 ms	309	3	100%	200ms	426
s838	32	-	–	–	–	-	–	–	–	–	–	–	–
s1423	74	22	92.15%	2 sec	203	22	86.27%	2 sec	329	–	–	–	–

A. *Analyzing the Results*

Results depicted in Table III require some explanation. Three different techniques were used to select partial scan registers. First, using the structural analysis/cycle breaking option of OPUS, partial scan registers were selected. The structural analysis option of OPUS automatically selects the minimum number of scan flip-flops required to break all cycles/loops in the design. After scanning the selected registers, tests for the circuits were generated using the sequential circuit test generator HITEC [30], and the fault coverage achieved was recorded. Next, the SCOAP testability analysis option of OPUS was used to select partial scan registers. The SCOAP testability option of OPUS does not automatically select the number of registers to scan for high fault coverage. It lists all the flip-flops of the circuit sorted according to their SCOAP controllability/observability measures. For our experiments with the SCOAP testability option of OPUS, we selected the same number of registers to scan as suggested by its cycle breaking option. The selected registers were scanned, the test vectors generated and the fault coverage was recorded. Finally, SIMPSON was used to select partial scan registers. The testability statistics presented for SIMPSON in Table III show the *minimum* number of scan registers required to achieve *higher or equally high* fault coverage as that obtained by using OPUS. The results are very encouraging. For almost all examples, SIMPSON suggests a better set of registers to scan than OPUS; by selecting fewer registers for scan, higher fault coverage is achieved.

Let us first compare the results obtained by using the "cycle breaking" option of OPUS with those obtained by using SIMPSON. For benchmarks *false, s1488, s344* and *s820*, OPUS and SIMPSON select the same number of registers for scan. However, SIMPSON selects different registers than OPUS and provides higher fault coverage. For benchmark *s1494*, OPUS and SIMPSON select the same registers for scan and hence their testability statistics are identical. For benchmark *s838*, VIS was unable to completely traverse the FSM in acceptable time. The BFS traversal did not converge because the reachable state set of this machine is very large. Also, since the size of the image computed at each iteration is relatively small, it requires too many iterations of image computation to converge to a fix point. For benchmark $s1423$, VIS could not complete the traversal because of memory limitations. The set of reachable states was too large to be compactly represented by a monolithic BDD. In all other cases SIMPSON selects fewer registers for scan than OPUS and provides higher fault coverage. For none of these benchmarks does SIMPSON produce worse results than OPUS.

Let us now compare the results obtained by using the "SCOAP testability analysis" option of OPUS with those obtained by using SIMPSON. For benchmarks *s1488* and *s344*, OPUS and SIMPSON select the same registers for scan and hence their testability statistics are identical. For benchmark *s820 and s1494*, OPUS and SIMPSON select the same number of registers for scan. However, SIMPSON selects a different register than OPUS and provides higher fault coverage. For all other benchmarks, SIMPSON selects fewer registers for scan than OPUS and still provides higher fault coverage[2]. Thus, it can be concluded from the results that the testability analysis techniques used by SIMPSON to select non-controllable state registers

[2]Data not available for the first few examples, bbsse-imec10, because of circuit net-list format incompatibilities

for partial scan are not only very accurate but also superior to the conventional state-of-the-art techniques used by OPUS.

Notice that the SIMPSON algorithm is not designed to answer the question: "How many flip-flops should be selected for partial scan?" It only provides an ordered list of flip-flops based on their non-controllability measures. Selecting flip-flops for scan affects the circuit adversely with respect to area and timing characteristics. Test engineers often have to provide scan-based design-for-test solutions within the area/timing constraints imposed on the designs. With the above issue in mind, the decision on the number of flip-flops to select for partial scan is left as a prerogative of the designer. With the above experiments, we only wish to demonstrate the accuracy of the non-controllability estimates used by SIMPSON to differentiate between the flip-flops for scan design.

B. Extension to SIMPSON: Incorporating Latch Correlations

The approach used by SIMPSON is still quite greedy. It does not take into account inter-dependencies and correlations among state registers. By scanning a register, it may become possible to indirectly control other registers. Scanning such indirectly controllable registers would be unnecessary. Thus, a straightforward extension to SIMPSON would be the incorporation of a technique that analyzes latch dependencies and correlations. *Cho et. al.* [22] suggested a model to evaluate latch dependencies, latch affinities and latch correlations. They used it to decompose the complete state space of a huge FSM into interacting FSMs so that implicit state enumeration could now be carried out on decomposed FSMs, each of a relatively smaller size. Such a model could be readily incorporated within SIMPSON. While selecting a set of scan registers, correlations of all the registers with respect to a pre-selected scan element could be analyzed to select the next best candidate for partial scan.

However, the model to evaluate latch dependencies and correlations proposed by *Cho et. al.* in [22] is based predominantly on the structure and topology of the circuit. Such structural models for analysis of latch correlations have a drawback as they cannot take into account *false dependencies* among the state registers (*e.g.* function $f = a + ab$ does not depend on b). The issue of combinational false dependencies can still be resolved efficiently; however, the presence of sequential false dependencies among registers, such as register-to-register multi-cycle false paths, significantly complicate the analysis [18]. Techniques that could efficiently analyze "functional" dependencies and correlations among the state registers of a circuit need to be developed.

Fortunately, by using SIMPSON iteratively, we can indirectly take into account correlations among state registers as follows. By scanning a register, some of the previously unreachable states become reachable and the size of the unreachable state set should shrink. As the state space of the underlying FSM of a circuit changes, so do the controllability measures of the flip-flops. Thus, after selecting a register for scan (modifying the circuit by converting the flip-flop into a primary input and output) we could recompute the set of reachable and unreachable states and apply the algorithm again on the modified circuit. In this way, we can use SIMPSON iteratively while selecting the single most non-controllable memory element for scan during each iteration. This would help us in avoiding those registers that can be indirectly controlled by scanning other registers in a more systematic and non-greedy fashion. Table IV presents the results of using the above extension to SIMPSON.

TABLE IV

EXPERIMENTAL RESULTS: ITERATIVE APPLICATION OF SIMPSON.

Circuit	Total # Regs.	SIMPSON - One Pass			SIMPSON -Iterative		
		# of Scan Regs.	Fault Cov.	# of Vects.	# of Scan Regs.	Fault Cov.	# of Vects.
s1488	6	3	100%	455	2	100%	625
s1494	6	2	98.8%	472	2	99.2%	776
s510	6	3	100%	480	3	100%	480
s344	15	2	98.54%	93	2	98.54%	93
s349	15	3	99.57%	91	3	99.57%	91
s641	19	3	99.37%	236	3	99.37%	236
s713	19	3	93.1%	257	3	93.1%	257
s444	21	5	98.1%	1321	5	99.58%	1627
s400	21	6	97.42%	723	6	99.06%	1729
s382	21	6	98.75%	693	6	98.75	693

Interestingly, for most benchmarks (*s510, s344, s349, s641, s713, and s382*) iterative application of SIMPSON results in selection of the same registers for scan as selected by its one-pass application; hence the identical fault coverage and testability overhead statistics. As no knowledge about the degree of correlation among the registers for these benchmarks is available, it is difficult to explain precisely why the greedy one-pass approach of SIMPSON performs just as good as the iterative approach. One possible reason could be that the selected scan registers do not have a high degree of correlation among them. For the benchmark *s1488*, iterative application of SIMPSON results in fewer scan registers with equally high fault coverage. For all other benchmarks (*s400, s444 and s1494*), higher fault coverage is achieved with the same number of scan registers by using SIMPSON iteratively. Iterative application of SIMPSON never produces worse results than those obtained by its one-pass counterpart.

C. Limitations of SIMPSON

SIMPSON, however, suffers from all the drawbacks associated with implicit state enumeration techniques using BDDs. For very large circuits, BDDs are known to suffer from the state explosion problem since they have worst-case memory requirements exponential in the size of the support set. This makes them impractical to represent large sets of reachable and unreachable states. Also, the CPU time used by SIMPSON can be a limiting factor in evaluating the scan flip-flops for very large circuits. As it can be observed from Table III, the processing time used by OPUS is a small fraction of that used by SIMPSON. The larger CPU time requirements for SIMPSON can be attributed to the iterative reachability computations performed while state enumeration. Use of dynamic variable ordering methods to limit the increase in the size of BDDs can further enhance the computation time.

As it becomes infeasible to represent and manipulate the set of all the reachable and unreachable states for very large circuits using BDDs, approximate reachability analysis [23] [22] can be carried out to analyze the behaviour of the underlying sequential machine. Approximate reachability analysis avoids the BDD explosion problem because it allows us to represent a super-set of reachable states of the machine as the product of smaller subsets, such that BDDs of their characteristic functions can be built and processed easily. Approximate reachability analysis trades off accuracy for space efficiency. The remainder of the paper analyzes how we can use approximate FSM traversal in order to estimate the non-controllability of flip-flops and target them for partial scan design.

VII. Approximate Reachability Analysis

In the previous section, we observed the practical limitations of the exact FSM traversal techniques to implicitly enumerate the states of a sequential circuit. We now review approximate reachability analysis techniques to estimate the upper-bound on the set of reachable states of an FSM. *Cho et al.* [23] presented techniques to decompose the state space of a large machine into smaller sub-machines and then perform reachability analysis on the resulting component machines to over-approximate the overall reachable state set. In what follows, we first describe the basic theory behind state space decomposition and then describe the approximate FSM traversal technique used in the context of this paper. For a detailed analysis of state space decomposition and approximate FSM traversal, the reader is referred to [23] [22].

A. State Space Decomposition

Let V be the set of state variables of an FSM, M, associated with a circuit C. A state variable partition of M is a partition of V. A state variable partition induces a *state space decomposition* (SSD). Let $\pi = \{\pi_1, \ldots, \pi_n\}$ be a state variable partition that partitions V into n components. Each component, π_i, represents a Boolean subspace consisting of the coordinate vectors corresponding to the state variables in π_i.

Let S be a set of states of the circuit C. Given a state variable partition $\pi = \{\pi_1, \ldots, \pi_n\}$, the *projection* of S on π_i is defined by:

$$P_i(S) = \exists_{(V-\pi_i)} S \qquad (13)$$

Let Δ be the transition function BFV for the original machine. The state variable partition $\pi = \{\pi_1, \ldots, \pi_n\}$ further induces a partition on the following:

1. A partition on Δ as $\{\Delta_1, \ldots, \Delta_n\}$; this is called a BFV partition. In other words, each block, π_i, of the state variable partition identifies a set of next state function components Δ_i. Each component Δ_i can be seen as the next state BFV of a sub-FSM of the original machine.

2. A partition on the original machine M as $\{M_1, \ldots, M_n\}$, where each M_i is a sub-FSM given by $M_i = \langle \sum \times B^{l-q}, O, S^*, P_i(S^0), \Delta_i, \Lambda_i \rangle$, where B= $\{0, 1\}$, l is the total number number of state variables, q is the number of state variables in partition π_i, $S^* \subseteq B^q$, Δ_i is the transition function BFV of M_i and Λ_i is the identity function from $P_i(S)$ to itself.

The set $\{M_1, \ldots, M_n\}$ constitutes the SSD FSM network obtained from partitioning the original machine. The primary inputs of M are shared by the M_i's and the communication between M_i's is through the state variables in V. The state variables local to a sub-FSM M_i, *i.e.* those that appear in partition π_i, are called *local state variables*. State variables not in partition π_i, but appearing in the support of Δ_i, are termed as *pseudo-primary inputs* of the sub-machine M_i. Note that state space decomposition is intended solely for reachability analysis purposes and, therefore, the corresponding partitioning of the output function BFV, Λ, of the original FSM, M, is intentionally left out [23].

B. Machine Partitioning for State Space Decomposition

The accuracy and efficiency of the approximate traversal approach strongly depends upon the latch set partitioned into subsets to obtain state space decomposition. Using intelligent ways to partition the machine the accuracy of approximate reachability analysis can be significantly improved. *Cho et al.* [22] showed that this can be achieved by taking into account the circuit structure while performing state space decomposition. The information provided by evaluating the mutual relationships among groups of latches can be efficiently exploited for this purpose. They proposed to evaluate latch *connectivity, correlation* and *affinity* measures to identify latches that can be grouped together to create machine partitions.

Connectivity, ω_{ij}, measures the mutual dependency between two state registers r_i, r_j. In other words, it indicates whether or not the next state function of state variable r_i has state variable r_j in its support. While connectivity accurately models the degree of dependency among state variables, it does not consider the overall relationship between pairs of state variables induced by shared primary inputs, structure of the next state logic, and so on. To this end, they proposed another measure called latch *correlation*, ρ_{ij}, that facilitates the systematic evaluation of the functional relationship between two latches r_i, r_j. State variables with a high degree of correlation need to be grouped together so that the resulting transition function BFVs have a small image, or at least a small image when constrained. This results in sub-FSMs whose approximate traversal provides a smaller, more accurate, reachable state set.

Connectivity and correlation are distinct measures of structural relationship between state variables. It is important to take into account both connectivity and correlation simultaneously. For this reason they defined latch *affinity*, σ_{ij}, as a weighted average of latch connectivity and correlation measures:

$$\sigma_{ij} = \alpha \cdot \omega_{ij} + (1 - \alpha) \cdot \rho_{ij} \tag{14}$$

where $0 \leq \alpha \leq 1$. From the computed affinities among all pairs of state variables of the circuit, a state variable affinity graph is created. An initial partition is created by grouping together the elements of the strongly connected components (SCCs) of the state variable affinity graph. Larger components are then broken and smaller ones are aggregated using seed/clustering techniques to partition the overall machine [22].

Fig. 5(a) depicts how the identification of SCCs of a hypothetical state variable affinity graph corresponds to machine partitioning. In the figure, the nodes labeled $r_1, \ldots, r_9$ represent the flip-flops of the circuit. By grouping together the flip-flops of an SCC, a state variable partition $\pi = \{\pi_1, \ldots, \pi_4\}$ is created. The state variable partition further induces a corresponding partition on the original transition function BFVs Δ as $\{\Delta_1, \ldots, \Delta_4\}$. Sub-FSM's M_i's can now be associated with each partition. FSM traversal can then be performed on each of these sub-FSMs to approximate the reachable state set.

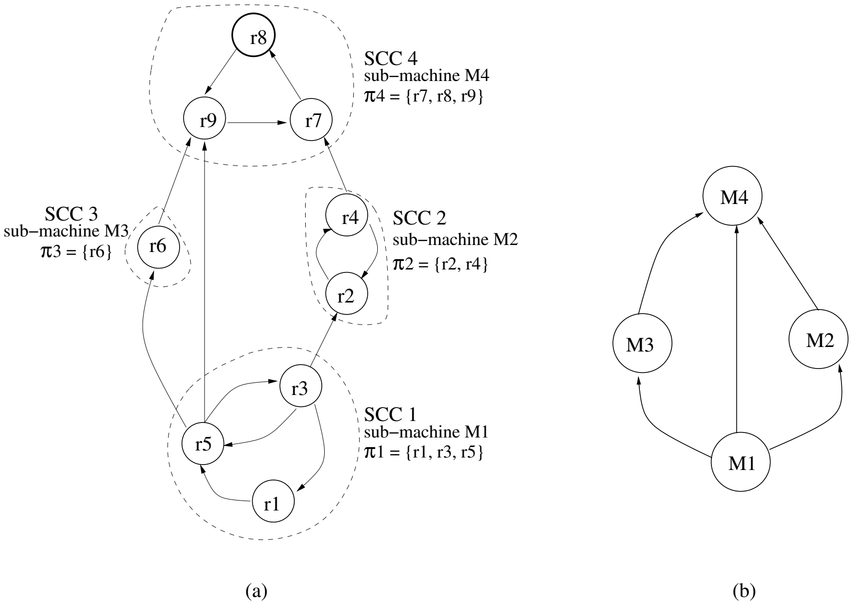


Fig. 5. Partitioning of machine M into submachines $\{M_1, M_2, M_3, M_4\}$ depicted on a hypothetical latch affinity graph. (a) Latches contained in an SCC grouped together correspond to a partition. (b) The FSM dependency graph derived from the state variable affinity graph.

Fig. 5 (b) depicts the FSM dependency graph derived by collapsing together all the nodes of each SCC of the affinity graph. Formally, an *FSM dependency graph* is a directed graph, $\bar{G} = (\bar{M}, \bar{E})$, where $\bar{M} = \{M_1, \ldots, M_n\}$ is the set of sub-FSMs and $\bar{E}$ is a set of directed edges such that if a local state variable of M_i is a pseudo-primary input of M_j then, $(M_i, M_j) \in \bar{E}$. This graph is used to compute a topological order of the sub-FSMs which, in turn, defines the order in which these submachines are traversed during approximate FSM traversal. $M_1 \prec M_2 \prec M_3 \prec M_4$ is one such order.

C. Approximate FSM Traversal

Once the state variable set of a large FSM is partitioned, it creates multiple Boolean subspaces of the original state space. Each subspace implicitly defines a sub-FSM which is a part of the original machine. Each of these sub-FSMs can be traversed implicitly using the procedure BFS_TRAVERSAL as described in Algorithm 1 to compute the set of reachable states in its corresponding subspace. The state variables belonging to other sub-machines, but appearing in the support of the submachine being traversed, may be treated as pseudo-primary inputs. Once all the sub-FSMs have been traversed, the overall reachable states of the original FSM can be approximated by computing the product of the reachable states of the sub-FSMs. This is an over-approximation (upper bound) of the reachable states of the original FSM. The overall problem is thus reduced to multiple application of the exact traversal procedure to subproblems of affordable size, whose solutions can be found efficiently.

However, the adverse effect of partitioning/decomposition is the loss of precision in the computation of reachable and unreachable states. This is because, when a submachine is separated from the other components of the system, it acquires additional degrees of freedom in its behaviour. An important aspect of approximate FSM traversal is to how to minimize the loss of precision in the computation of reachable states while completing the approximate traversal in acceptable time and memory limits.

The approximation of the reachable states can be considerably improved if the interaction between the sub-FSMs is properly modeled. This is a crucial issue as it helps reduce the degrees of freedom introduced by state space decomposition. In [23], two generic approximate FSM traversal methods were presented that differ in ways the interaction between FSMs is modeled during BFS traversal. These methods are listed below:

- *Machine-by-Machine Traversal:* Machine by Machine (MBM) traversal, as the name suggests, processes the sub-FSMs one at a time during one least fixed point computation. The sub-FSMs are processed serially and iteratively, untill a fixed point in the computation of over-approximated reached state set is obtained.
- *Frame-by-Frame Traversal:* Frame by Frame (FBF) traversal handles the sub-FSMs in parallel, and the overall traversal is a *one-sweep* operation. Each sub-FSM is started in its initial state and the image of that set is computed. As a result, all machines move one time frame ahead. And then another coordinated image computation is performed, one per each sub-FSM, and so on. The algorithm terminates when the computed reached state set converges. Two variants of the FBF traversal methods were proposed: *reached frame by frame* (RFBF) and *to frame by frame* (TFBF). The two variants differ in the way the convergence check is

performed.

For the purpose of our work, we use the *reached frame by frame* (RFBF) technique to estimate the non-controllability measures of the flip-flops of the circuit. The reasons for choosing the RFBF approximate reachability technique over the others are as follows. While the MBM technique converges faster than the other two, the over-approximated reachable state set computed is farther away in precision from the exact reached state set as compared with RFBF and TFBF traversal. This results in a significant loss of information of the reachable states. On the other hand, TFBF computes the upper-bound on the reachable state set closest to the exact reachable set. However, as a more accurate upper-bound is estimated, tighter constraints are required to model the interaction between sub-FSMs. Because of a tighter constraint set, TFBF requires more image computations to converge; this results in longer CPU times to complete the traversal. The RFBF traversal technique is an accuracy-time tradeoff between MBM and TFBF and is hence used for the purpose of evaluating non-controllability measures of flip-flops.

```
Inputs: n sub-FSMs {M_i}, corresponding projected initial state {S_i^0}.
  for i = 1...n do
    reached_i^0 = constraint_i^0 = S_i^0;/* initialize */
  end for
  j = 0;
  converged = FALSE;
  while converged == FALSE do
    for i = 1...n do
      Δ_c = SSD_CONSTRAIN(Δ_i, {constraint_i^j}); /* constrain transition relations */
      reached_i^{j+1} = IMAGE(Δ_c, constraint_i^j); /* 1-step reachability */
      constraint_i^{j+1} = reached_i^{j+1} /* update constraints */
    end for
    if reached_i^{j+1} == reached_i^j for all i then
      converged = TRUE;
    end if
    j = j + 1;
  end while
  /* Traversal completed: Return the set of reached states for all sub-FSMs */
  return ({reached_i^j})
```

Algorithm 3: Procedure RFBF_TRAVERSAL: Approximate state enumeration using image computations.

The RFBF traversal algorithm, taken from [23], is reproduced in Algorithm 3. The subscripts denote decomposed machines and superscripts denote the current reachability iteration. The algorithm proceeds as follows. It takes as input the SSD FSM network, *i.e.* the sub-FSMs M_i's, and the corresponding decomposed set of initial states $\{S_i^0\}$, where $\{S_i^0\} = P_i(S^0)$, the *projection* of the initial states for the given partition π_i. The current $reached$ state set is used to impose constraints on the pseudo-primary inputs of the sub-FSMs being traversed (the SSD_CONSTRAIN operation in the algorithm). This is performed in order to reduce the information loss due to machine decomposition. Suppose, M_i is the sub-FSM being traversed and M_j be its fanin FSM as computed by the FSM dependency graph. The local state variables of M_j

that are not pseudo-primary inputs of M_i are existentially quantified from the $reached_i$ set. This quantified $reached_i$ set is imposed as a constraint on Δ_i of M_i. The procedure is applied for all fanin sub-FSMs of M_i. The *bdd_constrain* operator, as defined by *Coudert et al.* [26], is used for this purpose. RFBF converges when $reached_i$ for all M_i's converges. It has been shown [23] that the algorithm is guaranteed to converge to a fix point and the set of reachable states thus computed is an over-approximation of the exact reachable states.

VIII. The SAMSON Algorithm

Inputs: A large FSM M, its initial state S^0.
Outputs: Scan registers listed in decreasing order of their non-controllability.

```
Decompose M into n sub-FSMs {M_i | 1 ≤ i ≤ n}.
Compute corresponding projected initial state set {S_i^0}.
for i = 1...n do
  reached_i^0 = constraint_i^0 = S_i^0; /* initialize */
end for
j = 0;
converged = FALSE;
while converged == FALSE do
  for i = 1...n do
    Δ_c = SSD_CONSTRAIN(Δ_i, {constraint_i^j}); /* constrain transition relations */
    reached_i^{j+1} = IMAGE(Δ_c, constraint_i^j); /* 1-step reachability */
    for each local state variable r_k ∈ M_i do
      compute_degree_of_unsettability(r_k);
    end for
    constraint_i^{j+1} = reached_i^{j+1}; /* update constraints */
  end for
  if reached_i^{j+1} == reached_i^j for all i then
    converged = TRUE;
  end if
  j = j + 1;
end while
/* Traversal completed: Compute the product of reachable states of sub-FSMs */
over_approx_reached = ∏_{i=1}^{i=n} reached_i^j;
unreached_states = bdd_complement(over_approx_reached);
for each state variable r_k of the entire circuit do
  compute_degree_of_unateness(r_k, unreached_states);
  non-controllability(r_k) = degree_of_unsettability(r_k) + degree_of_unateness(r_k);
end for
order the state variables of the entire circuit in decreasing order of their non-controllabilities
```

Algorithm 4: Procedure SAMSON: Scan using approximate state enumeration

Based on the approximate FSM traversal algorithm RFBF described in the previous section, and the flip-flop non-controllability measures presented in Section V, we present an algorithm SAMSON (Scan register selection using ApproxiMate State enumeratiON) to evaluate the non-controllability measures of flip-flops of a circuit. SAMSON, described in Algorithm 4, takes as input a large sequential circuit, partitions it into sub-FSMs, and performs approximate reachability analysis using symbolic image computations on individual sub-FSMs (RFBF traversal procedure). During each image computation step for a sub-FSM, we record the *degree of unsettability* for each *local* state variable. Once the traversal terminates for all submachines, $\{reached_i\}$'s contain the reachable states of sub-FSMs $\{M_i\}$'s. The upper-bound on the overall reached state set is then computed as a cartesian product of all $\{reached_i\}$; this is stored in *over_approx_reached*. Complement of this over-approximated reached state set results in an under-approximation of the overall unreachable state set. From this under-approximated set of unreachable states, we compute the *degree of unateness* for each state variable of the original machine. The overall non-controllability of all flip-flops can then be evaluated by adding their respective degrees of unsettability and unateness. The flip-flops of the entire circuit are then sorted in decreasing order of their non-controllabilities.

Note that unlike SIMPSON, SAMSON does not examine missing rising and falling transitions for any flip-flop. Since the traversal is performed on partitioned sub-FSMs individually, a missing transition on a local state variable does not guarantee the same behaviour in the overall machine. We have noted experimentally that by scanning such flip-flops, which correspond to missing rising or falling transitions in their respective sub-FSM traversal trace, does not necessarily contribute to the improvement in the overall fault coverage.

SAMSON was programmed within the VIS tool-set and experiments were carried out using the ISCAS'89 benchmark circuits that contained a larger number of flip-flops (# of FFs $>$ 30). The results depicted in Table V are compared with OPUS using both the structural analysis/cycle breaking and the SCOAP testability analysis options.

TABLE V

EXPERIMENTAL RESULTS: APPLYING THE SAMSON ALGORITHM.

Circuit	Total FFs	OPUS: Cycle Breaking				OPUS: SCOAP Testability Option				SAMSON			
		Scan FFs	Fault Cov	CPU Time	# of Vects	Scan FFs	Fault Cov	CPU Time	# of Vects	Scan FFs	Fault Cov	CPU Time	# of Vects
s1423	74	22	92.1%	2 sec	203	22	86.2%	2 sec	329	22	95.2%	23 sec	495
s5378	164	30	93.4%	3 sec	1294	30	90.4%	8 sec	1538	30	88.6%	107 sec	1324
s9234	211	65	81.1%	9 sec	7026	65	36.9%	12 sec	1217	65	85.3%	352 sec	7122
s15850	534	91	82.3%	8 sec	12003	91	76.2%	41 sec	5417	91	86.0%	675 sec	12689
s13207	638	58	77%	6 sec	2856	58	62.9%	26 sec	3958	58	84.2%	47 min	6870
s38584	1426	313	92.1%	23 sec	12879	313	89.7%	22 min	9560	313	93.4%	197min	19321

A. Analyzing the Results

Our experimental setup requires some explanation. First, the structural analysis/cycle breaking option of OPUS was used to select the partial scan registers. OPUS automatically selects the minimum number of scan registers required to break all the loops in the design. After scanning the required registers, the ATPG tool HITEC was used to generate tests and the observed fault coverage was recorded. Next, the SCOAP testability option was used to select partial scan registers. SCOAP testability option does not automatically select the number of registers to be scanned. It only evaluates and orders the state registers in terms of their SOAP controllability/observability measures. For our experiment with the SCOAP option of OPUS, we used the same number of scan registers as suggested by the cycle breaking option of OPUS. After scanning the required registers, tests were generated and the testability statistics were recorded. Finally, we used the SAMSON algorithm to order the registers in terms of their non-controllability measures. The number of registers scanned was again kept the same as suggested by the cycle-breaking option of OPUS in order to make a fair comparison. The requisite number of registers were scanned and test generation was carried out.

Machine partitioning is an important issue with SAMSON, as the nature and size of the partitioned machine affects the precision of the approximation of reachable states. While it is true that by creating smaller partitions the approximate FSM traversal would converge quickly, it would, however, result in a significant loss of precision on reachable states. The larger the size (in the number of state variables) of partitioned sub-FSMs, the fewer the number of partitions, and more accurate the approximation. However, image computations on large partitions would require excessive CPU time and memory and would make the entire process infeasible. From our previous experiments with exact reachability analysis, we had observed that exact FSM traversal techniques cannot efficiently handle circuits that contain more than 30 flip-flops (approximately). For this reason, we imposed a restriction on the machine partitioning algorithms so as to include *no more than 30 flip-flops* in each partitioned sub-FSM.

The results obtained by SAMSON compare favourably with those obtained by OPUS. For all benchmark circuits other than $s5378$, the scan registers selected by SAMSON provide higher fault coverage than those selected by both options of OPUS. The CPU times for SAMSON, while longer than those for OPUS, are certainly not impractical. The CPU times for SAMSON can be attributed to the time required for i) machine partitioning, ii) iterative image computations, iii) the constraining of partitioned transition relations and iv) computing the non-controllability measures for all the flip-flops. By investing some time in performing the above computations, SAMSON selects a better set of registers to scan than OPUS and provides higher fault coverage.

For the benchmark $s5378$ the cycle breaking option of OPUS selects 30 scan registers and provides 93.4% fault coverage, while the same number of registers scanned using SAMSON provide only 88.6% fault coverage. We analyzed this circuit to find the reasons for the poor performance of SAMSON *vis-a-vis* OPUS. We made the following observation. The affinity graph of the circuit has only one strongly connected component (SCC) consisting of 124 flip-flops. Since we limit our partition size to 30 flip-flops, this SCC gets broken and we are unable to create a sub-FSM that encompasses this SCC. Intuitively, if a sub-FSM entirely contains all the flip-flops of an SCC of the circuit, the interaction between the flip-flops can be modeled more efficiently for FSM traversal. This may potentially lead to a better approximation of overall machine reachability. By breaking this SCC into different sub-FSMs, we allow additional degrees of freedom on the state variables and are unable to model their interaction efficiently. We conjecture that it is perhaps for this reason that the non-controllability measures evaluated by SAMSON for the flip-flops of $s5378$ are inaccurate. Unfortunately, since it is computationally infeasible to perform implicit state enumeration on a partition that would contain this SCC (124 flip-flops), there is no way to verify whether the above is really the case.

However, similar observations on other benchmarks support our intuition. For example, the benchmark $s13207$ has 17 SCCs. Only 1 out of the 17 SCCs contains 252 latches and the rest 16 SCCs contain fewer than 30 latches. These SCCs are contained entirely within a partition. Note that for this benchmark, SAMSON provides higher fault coverage (84.2%) as compared with OPUS (77%). Perhaps by encompassing the SCCs within partitions we were able to model their interaction more effectively.

B. Iterative Application of SAMSON

In Section VI-B we had discussed that by scanning a register it may become possible to indirectly control other registers of the circuit. Scanning such indirectly controllable registers should then be unnecessary. Subsequently, we had argued that by using SIMPSON iteratively, *i.e.* by selecting the most non-controllable flip-flop for scan in each iteration of SIMPSON and modifying the circuit by transforming the flip-flop into primary input and output, we could avoid selecting such indirectly controllable flip-flops for scan. Analogous to the iterative application of SIMPSON, we analyze the effect of using SAMSON iteratively on the above larger circuits of the ISCAS'89 benchmark suite.

Since the circuits experimented with SAMSON contain a large number of flip-flops, selecting only one (the most non-controllable) flip-flop for scan in each iteration would make the entire scan selection process too time consuming. For this reason, we experimented with two different criteria for selecting a set of scan registers in each iteration. The results are depicted in Table VI in columns "Iterative SAMSON - 1" and "Iterative SAMSON - 2" and are compared with the results obtained by using the one-pass application of SAMSON. The total number of flip-flops selected for scan in each experiment is kept the same as that for the one-pass application of SAMSON.

Iterative SAMSON - 1: Recall that SAMSON decomposes a large FSM into smaller ones and then performs reachability analysis on the individual sub-FSMs. Therefore, for the first set of iterative experiments with SAMSON, we select the most non-controllable flip-flop from *each partitioned submachine*, in every iteration. In other words, we select one (the most non-controllable) flip-flop for scan per sub-FSM per it-

TABLE VI

EXPERIMENTAL RESULTS: APPLYING THE SAMSON ALGORITHM ITERATIVELY.

Circuit	Total FFs	SAMSON: One Pass				Iterative SAMSON - 1				Iterative SAMSON - 2			
		Scan FFs	Fault Cov	CPU Time	# of Vects	Scan FFs	Fault Cov	CPU Time	# of Vects	Scan FFs	Fault Cov	CPU Time	# of Vects
s1423	74	22	95.2%	23 sec	495	22	78.2%	115 sec	341	22	95.1%	74 sec	487
s5378	164	30	88.6%	107 sec	1324	30	75.52%	489 sec	394	30	89.2%	389 sec	1444
s9234	211	65	85.3%	352 sec	7122	65	80.8%	34 min	5076	65	86.2%	21 min	7519
s15850	534	91	86.0%	675 sec	12689	91	74.8%	171 min	2964	91	86.9%	39 min	12781
s13207	638	58	84.2%	47 min	6870	58	77.5%	134 min	4926	58	84.9%	148 min	7003

eration. After the flip-flops are selected for scan, one from each sub-FSM of the SSD FSM network, their present state and next state lines are transformed into primary inputs and outputs of the circuit. SAMSON is again applied on the modified circuit and another set of flip-flops is selected for scan, one from each sub-FSM. The process terminates when the required number of flip-flops have been selected for scan.

By modifying the scan flip-flops into primary inputs and outputs, not only does the state space of the original machine change, but the structure of the circuit (and hence the latch connectivity, correlation and affinity measures) also undergoes a change. This means that when the modified circuit is again partitioned for approximate FSM traversal, the new SSD FSM network may potentially be vastly different from the ones obtained in previous iterations.

The results obtained by experimenting with the above technique are not encouraging. For almost all the circuits, the fault coverage achieved by selecting scan flip-flops one per submachine per iteration is worse than that achieved by using the one-pass approach of SAMSON. We analyzed the submachines and the testability measures of the corresponding flip-flops to find the reasons behind the apparent failure of this technique. We found that some submachines contained flip-flops whose non-controllability measures were low, suggesting that the corresponding sub-FSM was an easily testable machine. Easily testable machines may not require scan to provide high fault coverage of the area corresponding to their portion of the circuit. On the other hand, some submachines contained a significant number of flip-flops whose non-controllability measures were very high, suggesting that the corresponding sub-FSM was a difficult to test machine. Such sub-FSMs may require a high degree of scan to cover the faults corresponding to their portion of the circuit. Selecting the easily controllable flip-flops for scan, while omitting the ones that belong to a difficult to test submachine, is the main cause of the reduced fault coverage. Clearly, selecting one flip-flop per sub-FSM per iteration is not an effective strategy to obtain a good set of scan flip-flops for high fault coverage.

<u>*Iterative SAMSON - 2:*</u> We have argued that selecting one flip-flop for scan in each iteration of SAMSON would be too time consuming. Therefore, in our second set of iterative experiments, instead of selecting one

scan register per iteration, we propose to select a predetermined number of scan registers in each iteration. For our experiments, in each iteration of SAMSON, we selected 25% of the required number of scan registers; limiting the number of iterations of SAMSON to four. For example, for the benchmark $s5378$, the required number of scan registers is 30. Since 25% of 30 is 7.5, we select eight (rounding up 7.5 to 8) scan registers in each of the first three iterations and the remaining six in the last one. After each iteration of SAMSON, the selected scan registers are transformed into primary inputs and outputs and the algorithm is applied again, and so on. The results, when compared with "Iterative SAMSON - 1" are better in terms of fault coverage. When compared with the one-pass approach of SAMSON, the fault coverage achieved is marginally better though it comes at the cost of higher CPU times. For example, for the benchmark $s13207$, the improvement in fault coverage is only 0.7%, and it comes at a cost of more than triple the investment in CPU time. As far as the achieved fault-coverage/CPU-time trade-off is concerned, from the experiments one can conclude that the one-pass approach of SAMSON is a better option.

IX. Conclusions and Future Work

In this paper, we presented a new approach to the partial scan problem that analyzes the circuit state information in order to evaluate the non-controllability of flip-flops. Implicit state enumeration is used as a tool to analyze the reachable and unreachable state space of the underlying FSM of the sequential circuit in order to evaluate its testability measures. An algorithm, SIMPSON, has been proposed and the results have been presented which demonstrate the importance of using the circuit state information in order to evaluate the non-controllability of the flip-flops. As compared to conventional state-of-the-art partial scan register selection techniques, our approach performs better; by selecting fewer scan registers, higher or equally high fault coverage is achieved.

However, SIMPSON suffers from BDD size explosion problems and is computationally expensive and slow for very large circuits. The excessive CPU time requirements arise from the iterative reachability computations performed while state enumeration. As it becomes infeasible to represent and manipulate the set of all the reachable and unreachable states for very large circuits using BDDs, the use of approximate reachability analysis is proposed to analyze the behaviour of the underlying sequential machine. Approximate reachability analysis avoids the BDD explosion problem because it allows us to represent a super-set of reachable states of the machine as the product of smaller subsets, such that BDDs of their characteristic functions can be built and processed easily. We presented an algorithm, SAMSON, that exploits the power of approximate implicit FSM traversal techniques to estimate the non-controllabilities of flip-flops. These most non-controllable flip-flops are targeted for partial scan design. Approximate FSM traversal results in some "loss of information" of the reachable and unreachable state sets of the machine. However, we demonstrated by experiments that such a loss does not significantly affect the proposed non-controllability measures evaluated on the approximated state sets. Using our techniques, we are able to select a better set of scan registers that provide higher fault coverage than that achieved by using contemporary scan register selection techniques, though at the cost of higher computation times.

We also discussed how the scanning of a register changes the state space of the underlying machine and, as a result, affects the ranking of the flip-flops according to their non-controllability measures. We analyzed the effect of iterative application of the proposed algorithms while selecting only one flip-flop (in the case of SIMPSON) or a set of flip-flops (in the case of SAMSON) in each iteration. While iterative application of the presented algorithms does provide higher fault coverage, the improvement in fault coverage is often marginal, and it is achieved at the expense of significant amount of CPU times.

The testability measures of flip-flops proposed in this paper address not only the non-controllable and difficult-to-control registers, but also take into account the unreachable states of the machine that contribute to the sequentially untestable faults. While we have shown how to model the non-controllability of flip-flops using state machine transitions, it is not clear how to address the issue of *observability* using implicit FSM traversal. This topic is worthy of future research. Further, the techniques presented in the paper are well suited for finite state machines, *i.e.* sequential circuits with feedback. It is not clear how the proposed techniques can be modified to identify pipeline registers for partial scan. Also, if the designs contain RAMs, it would be interesting to research whether or not the states of the RAM should be analyzed for determining scan flip-flops in the design.

REFERENCES

[1] J. P. Roth, "Diagnosis of Automata Failures: A Calculus and a Method", *IBM Journal of Res. Dev.*, pp. 278–291, July 1966.

[2] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits", *IEEE Transactions on Computers*, pp. 215–222, March 1981.

[3] H. Fujiwura and T. Shimono, "On Acceleration of Test Generation Algorithms", *IEEE Transactions on Computers*, pp. 1137–1144, December 1983.

[4] E. Trishler, "Incomplete Scan Path with an Automatic Test Generation Methodology", *Proceedings of the International Test Conference*, pp. 153–162, 1980.

[5] V. D. Agarwal, K. T. Cheng, D. D. Johnson, and T. Lin, "A Complete Solution to the Partial Scan Problem", *Proceedings of the International Test Conference*, pp. 44–51, 1987.

[6] H. K. T. Ma, S. Devadas, A. R. Newton, and A. Sangiovanni-Vincentelli, "An Incomplete Scan Design Approach to Test Generation of Sequential Machines", *Proceedings of the International Test Conference*, pp. 730–734, 1988.

[7] K. T. Cheng and V. D. Agarwal, "An Economical Scan Design for Sequential Logic Test Generation", *Proceedings of the 19th International Symposium on Fault-Tolerant Computing*, pp. 28–35, 89.

[8] D. H. Lee and S. M. Reddy, "On Determining Scan Flip-Flops in Partial Scan Design", *Proceedings of International Conference on Computer Aided Design*, pp. 322–325, 1990.

[9] V. Chikermane and J. H. Patel, "An Optimization Based Approach to the Partial Scan Design Problem", *Proceedings of the International Test Conference*, pp. 377–386, 1990.

[10] V. Chikermane and J. H. Patel, "A Fault Oriented Partial Scan Design Approach", *Proceedings of the International Test Conference*, pp. 400–404, 91.

[11] P. S. Parikh and M. Abramovici, "A Cost Based Approach to Partial Scan", *Proceedings of the Design Automation Conference*, pp. 255–259, 1993.

[12] Y. Bertrand, F. Bancel, and M. Renovell, "Multiconfiguration Technique to Reduce Test Duration for Sequential Circuits", *Proceedings of the International Test Conference*, pp. 989–997, 1993.

[13] K. S. Kim and C. R. Kime, "Partial Scan by use of Empirical Testability", *Proceedings of the International Conference on Computer Aided Design*, pp. 314–317, 1990.

[14] M. Abramovici, J. J. Kulikowski, and R. K. Roy, "The Best Flip-Flops to Scan", *Proceedings of the International Test Conference*, pp. 166–173, 1991.

[15] L. H. Goldstien, "Controllability/Observability Analysis of Digital Circuits", *IEEE Transactions on Circuits and Systems*, pp. 685–693, 1979.

[16] V. D. Agarwal and M. R. Mercer, "Testability Measures - What do they tell us?", *Proceedings of the International Test Conference*, pp. 391–396, 1982.

[17] H. J. Wunderlich, "The Design of Random-Testable Sequential Circuits", *Proceedings of the 19th International Symposium on Fault Tolerant Computing*, pp. 110–117, 1989.

[18] P. Kalla and M. J. Ciesielski, "Testability of Sequential Circuits with Multi-Cycle False Paths", *Proceedings of the 15th IEEE VLSI Test Sympsium*, pp. 322–328, 1997.

[19] D. Xiang, S. Venkataraman, W. K. Fuchs, and J. H. Patel, "Partial Scan Design Based on Circuit State Information", *Proceedings of the 33rd Design Automation Conference*, pp. 807–812, 1996.

[20] S. Sharma and M. Hsiao, "Partial Scan using Multi-Hop State Reachability Analysis", *in VLSI Test Symposium*, pp. 121–126, 1999.

[21] T. E. Marchok, A. El-Maleh, W. Maly, and J. Rajski, "Complexity of Sequential ATPG", *Proceedings of the Design Automation Conference*, pp. 252–261, 1995.

[22] H. Cho, G. Hachtel, E. Macii, M. Poncino, and F. Somenzi, "Automatic State Space Decomposition for Approximate FSM Traversal Based on Circuit Analysis", *IEEE Transactions on Computer Aided Design*, pp. 1451–1464, December 1996.

[23] H. Cho, G. Hachtel, E. Macii, B. Plessier, and F. Somenzi, "Algorithms for Approximate FSM Traversal Based on State Space Decomposition", *IEEE Transactions on Computer Aided Design*, pp. 1465–1478, December 1996.

[24] I. Pomeranz and S. M. Reddy, "Classification of Faults in Synchronous Sequential Circuits", *IEEE Transactions on Computers*, pp. 1066–1077, September 1993.

[25] S. Devadas, H. K. T. Ma, A. R. Newton, and A. Sangiovanni-Vincentelli, "Irredundant Sequential Machines via Optimal Logic Synthesis", *IEEE Transactions on Computer Aided Design*, pp. 8–17, January 1990.

[26] O. Coudert and J. C. Madre, "A Unified Framework for the Formal Verification of Sequential Circuits", *Proceedings of the International Conference on Computer Aided Design*, pp. 126–129, 1990.

[27] H. Touati, H. Savoj, B. Lin., R. K. Brayton, and A. Sangiovanni-Vincentelli, "Implicit State Enumeration of Finite State Machines using BDDs", *Proceedings of the International Conference on Computer Aided Design*, pp. 130–133, 1990.

[28] K. S. Brace, R. Rudell, and R. E. Bryant, "Efficient Implementation of the BDD Package", *Proceedings of the Design*

Automation Conference, pp. 40–45, 1990.

[29] R. E. Bryant, "Graph Based Algorithm for Boolean Function Manipulation", *IEEE Transactions on Computers*, pp. 677–691, August 1986.

[30] T. M. Niermann and J. H. Patel, "HITEC: A Test Generation Package for Sequential Circuits", *Proceedings of the European Conference on Design Automation*, pp. 214–218, February 1991.

[31] H. Cho, G. Hachtel, and F. Somenzi, "Redundancy Identification/Removal and Test Generation for Sequential Circuits using Implicit State Enumeration", *IEEE Transactions on Computer Aided Design*, pp. 935–945, July 93.

[32] D. E. Long, M. A. Iyer, and M. Abramovici, "Identifying Sequentially Untestable Faults using Illegal States", *Proceedings of the VLSI Test Symposium*, pp. 4–11, 1995.

[33] I. Hartanto, V. Boppana, and W. K. Fuchs, "Identification of Unsettable Flip-Flops for Partial Scan and Faster ATPG", *Proceedings of the International Conference on Computer Aided Design*, 1996.

[34] R. K. Brayton, G. D. Hachtel, A. Sangiovanni-Vencentelli, F. Somenzi, A. Aziz, S-T. Cheng, S. Edwards, S. Khatri, Y. Kukimoto, A. Pardo, S. Qadeer, R. Ranjan, S. Sarwary, Swamy G. Shiple, S., and T. Villa, "VIS: A System for Verification and Synthesis", *Proceedings of the Computer Aided Verification Conference*, 1996.

[35] F. Somenzi, "Colorado Decision Diagram Package", *Computer Programme*, 1997.

A note to the editor and reviewers

A preliminary version of this paper was submitted to IEEE-TCAD in Sept. 1998. The paper was rejected on the grounds that we had experimented only with exact reachability analysis - which could not handle circuits with a large number of flip-flops. The associate editor and all the reviewers had suggested that we research, further, the use of approximate reachability analysis so as to handle large designs. They had recommended us to complete the experiments with approximate reachability analysis and resubmit it as a new paper. We have concluded the research in this area, and this submission is complete in that respect. Few other issues were raised by the reviewers of the older version. Their concerns and a short response to those comments are listed below:

Reviewer's concern: I'd find a better example than the one used in Section "Motivation" (Fig. 2) of the weakness of structural techniques. The problem seems to be a bit unrealistic, though easy to show. If you cannot find an example of reasonable size it's okay.

Author's response: The main aim of providing a motivating example was to show the importance of analyzing state space information for evaluating the non-controllability of flip-flops. Further, we used this example to show how (and why) implicit state enumeration can be used for this purpose. The reason we used this example was not to show the failure of structural analysis techniques for scan, but that this example can easily be used, didactically, to demonstrate implicit FSM traversal and subsequent controllability analysis. That structural techniques fail to select a "good" scan register for this circuit is really a side issue.

Reviewer's concern: The authors describe the algorithm quite well however, they stop short when it comes to describing the criteria they used to select the FFs to scan from the ranked list. Because it is missing, I would like to know if they intentionally left that detail out because the criteria was not the same for every circuit?

Author's response: We have argued in this paper that the proposed techniques differentiate between the flip-flops according to their evaluated non-controllability measures, and list the flip-flops of the entire circuit in the decreasing order of their non-controllabilities. The question "how many flip-flops to select for scan?" is left open to the prerogative of the test/design engineers - based on the area/delay constraints within which they wish to provide a scan-based DFT solution.

Reviewer's concern: I believe the fact that selecting one FF for scan effects the ranking is not a minor issue to be addressed at the end of the paper. It was a major concern of mine until I saw it addressed in the Extensions to SIMPSON. Please complete the research in this area before creating a Transactions Paper on this subject.

Author's response: This issue has been analyzed in this version of the paper. We have performed numerous experiments, and have also drawn strong conclusions based on the experimental results.