Degree-Of-Freedom Analysis for Sequential Machines Targeting BIST Quality and Gate Area

Samir Roy[1] Biplab K Sikdar[2] Monalisa Mukherjee[2] Debesh K Das[3]

[1] Department of Computer Science & Technology, Kalyani Govt. Engineering College, Kalyani, India 741235 samir@kucse.wb.nic.in
[2]Department of Computer Science & Technology, Bengal Engineering College (D U), Howrah, India 711103 {biplab,mona}@becs.ac.in
[3]Department of Computer Science & Engineering, Jadavpur University, Calcutta, India 700032 debeshd@hotmail.com

Abstract— **This paper reports the design of $BIST$ structures for sequential machines. Testability of an FSM is limited due to the fact that some machine states remain unreachable and some act as a sink under any input sequence. The proposed scheme provides uniform mobility, referred to as degree of freedom, among the machine states in test mode by enhancing the reachability and emitability of the states. Uniform mobility of states ensures higher fault efficiency in a $BIST$ structure. A graph based approach is introduced for state code assignment to minimize gate area. Experimental results on benchmark circuits establish that the proposed scheme does improve the $BIST$ quality simultaneously reducing the gate area of the synthesized machine.**

I. Introduction

Built-in self test ($BIST$) for sequential circuits is rather a complicated issue due to poor fault efficiencies achieved through pseudo random pattern generators. Test research community have emphasized on evolving innovative $BIST$ structures [1]. Several attempts were made to improve the fault efficiency in a $BIST$ structure for sequential circuits. The modification of flip-flops to make the machine states reachable [2], [3], improving the quality of test pattern generators [4], [5] are the significant among them. A comprehensive study on $BIST$ for sequential circuits is discussed in [6] that proposes $BIST$ with minimum overhead.

It has been observed that, some states in a sequential circuit (FSM) are hard to reach and some others are hard to exit. Existence of hard-to-reach and hard-to-exit states results in poor fault efficiency with the pseudo random test vectors of a BIST test pattern generator.

In this paper a metric, called the *Degree of Freedom* (DOF) in FSM states, has been introduced which quantifies the merit of an FSM state with respect to the $BIST$ structure of the machine. The DOF in a state expresses the openness of the state in terms of its reachability and exitability (henceforth referred to as *emitability*) to other states. Based on the DOF-analysis of a sequential machine, an efficient state code assignment scheme is proposed to enhance fault efficiency and to reduce gate area of the $BIST$ed FSM.

The proposed design analyzes an FSM to extract the relative degree of freedom of it's states. Also, the cost of state encoding in terms of gate area of the synthesized machine is computed. A complete weighted graph, called the *intimacy graph*, is constructed to search for the state codes that minimizes the gate area while maintaining the enhanced $BIST$ quality obtained from DOF analysis. The reduction in gate area is achieved through a genetic algorithm (GA) [7] based state code assignment scheme.

The rest of the paper is organized as follows. *Section III* provides the overview of the state code assignment scheme based on the DOF-analysis preceded by the preliminaries on sequential machine $BIST$ structure noted in Section II. The method to reduce gate area of the synthesized machine is presented in *Section IV*. Experimental results on benchmark circuits, reported in *Section V*, confirms that the proposed scheme enhances the testability to a large extent simultaneously reducing the gate area of the synthesized machine.

II. The Preliminaries

A sequential machine consists of a combinational circuit (CC) and the system register (SR). The outputs $y_1, y_2, \cdots, y_k$ from the k memory elements of SR define the present state (PS) of the machine.

A. BIST Structure for FSM

Built-in self test techniques for synchronous sequential circuits are proposed in [2], [3], [4]. The $BIST$ structure developed for fully or partially scanned sequential circuits reconfigures the circuit flip-flops into scan registers and make them as a part of the $BIST$ test pattern generator (TPG) and signature analyzer (SA), in test mode. If the combinational logic block (CC) of the sequential machine is not tested separately, then there is no need to configure the circuit flip-flops. This structure requires two test registers - the PIs of the circuit are fed from the output of a pattern generator and the POs are fed into the signature analyzer. Moreover, as the circuit flip-flops remain unmodified this allows at-speed testing of the circuit under its normal operation condition and saves the hardware overhead of reconfiguring the flip-flops. For the current design we have considered a cellular automata (CA) based pseudo random pattern generator (PRPG) as the $BIST$ test pattern generator (TPG).

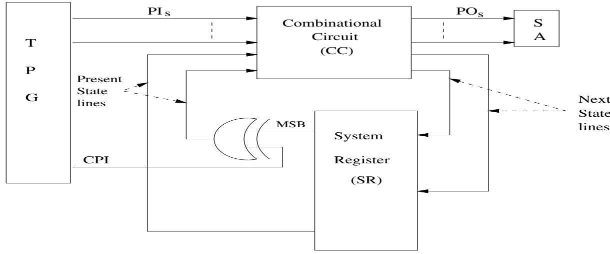

Fig. 1. FSM with control point at present state line

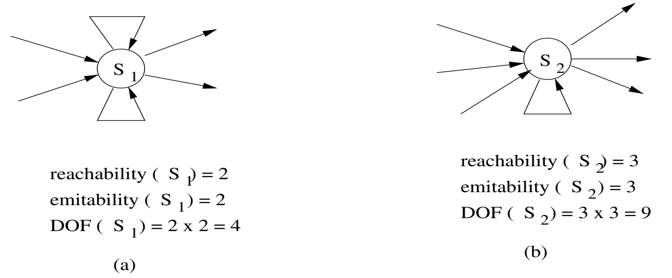

Fig. 2. Degree of Freedom in FSM States

B. Degree of Freedom in FSM States

For an FSM with n states, at least k flip-flops are required to encode the states, where $2^{k-1} < n \leq 2^k$. Out of 2^k state codes, n of them are assigned and the rest $(2^k - n)$ codes are kept unused. These unused state codes are referred to as the *unreachable* states of FSM. Since the FSM never attains an *unreachable* state for any input sequence, the corresponding state code never appear at the PS lines. This fact restricts the testability of the sequential machine. Such a code may be required to test some faults of the combinational logic block CC (*Fig.1*).

Moreover, an FSM may contain a large number of states with few incoming edge on them, which makes the states hard to reach during circuit functioning. The bit patterns corresponding to those *hard-to-reach* state code will rarely appear on the PS lines. The presence of *hard-to-reach* states reduces the fault coverage in CC. Again, states with a large number of self loops act like a *sink* in the sense that once the FSM reaches that state it tends to remain there indefinitely. Getting stuck at a specific bit pattern in PS lines will hinder the detection of faults in CC. The following parameters are defined to characterize the $BIST$ quality of FSM states.

Definition 1: For each state S of an FSM, *reachability(S)* is the number of edges incident on S from other states. The self loops are not counted in reachability(S) analysis.

Definition 2: For each state S of an FSM, *emitability(S)* is the number of edges exit from S. The self loops are not counted as they fall on itself.

Definition 3: The *degree of freedom* in a state S of the FSM, denoted as DOF(S), is defined as the product of its *reachability* and *emitability*, i.e., DOF(S) = reachability(S) $\times$ emitability(S).

Fig.2 illustrates the computation of the degree of freedom in an FSM having two PIs.

$DOF(S)$ represents the ease with which state S may be reached and exited during circuit functioning. The next *section* provides means to enhance the fault efficiency in a $BIST$ structure through DOF analysis.

III. State Encoding Based on DOF

In the present scheme out of 2^k state codes, 0 to $(n-1)$ are assigned to the reachable states of the FSM so that the MSB of the unreachable state codes is 1. For example, consider an FSM with $n = 5$ states. The number of unreachable states is 3. In the present encoding scheme, the 5 reachable states are encoded as {000,001,010,011,100}. The unreachable state codes are {101,110,111} with MSB as 1.

A. Reaching Unreachable States

The above encoding ensures that for every unreachable state code there exists a reachable state with MSB as 0. To input the unreachable codes to CC in test mode we propose to add a control point at the MSB of PS lines. The control point is implemented with an XOR gate as shown in *Fig.1*. One input to XOR is the MSB of PS lines and the other one is the control primary input (CPI). Since the reachable state codes are likely to appear on the PS lines, unreachable state codes also can be fed to CC by keeping CPI as 1. In test mode, the CPI is fed (*logic 0 or 1*) from the test pattern generator, whereas, for normal functioning of the FSM the CPI is fixed to *logic 0*.

B. Reaching Low Reachable States

We now present a state encoding scheme that considers the DOF values in FSM states and ensures easy mobility between the set of states {LDF} with low degree of freedom and the set of states {HDF} with high degree of freedom.

Suppose (S_l, S_h) is a pair of states, where $S_l \in \{LDF\}$ and $S_h \in \{HDF\}$. To ensure mobility between S_l & S_h, the codes C_l and C_h are assigned to S_l and S_h respectively, where C_l and C_h differ only in a single bit position (p). In this work the LSB is taken as the pth bit position, which ensures maximum number of (C_l, C_h) pairs (in fact, $\lfloor n/2 \rfloor$) that differ in exactly one bit. If a control point with XOR is added at the LSB of present state (PS) lines, then the FSM can easily switch from the state S_h to S_l and vice versa.

Therefore, by adding two control points, at the MSB and LSB, one can ensure mobility between 2^{k-2}

pairs of states. The scheme for state encoding is next summarized.

Algorithm 1: Encode_state_from_DOF_analysis
Input: State transition table (STT) of sequential machine
Output: A state encoding with enhanced $BIST$ quality
Step 1: Find the number of states n and k, $2^{k-1} < n \le 2^k$
Step 2: Compute DOF for all the states $S_0, S_1, \cdots, S_{n-1}$
Step 3: Sort the states $S_0, S_1, \cdots, S_{n-1}$ in ascending order of DOF. Let the sorted list be $S_0^{'}, S_1^{'}, ..., S_{n-1}^{'}$
Step 4: Take the pair of states $(S_i^{'}, S_{n-1-i}^{'})$ $\forall i = 0, 1, \cdots, \lfloor n/2 \rfloor$. The resulting pairs $(S_0^{'}, S_{n-1}^{'})$, $(S_1^{'}, S_{n-2}^{'})$, $\cdots$, $(S_i^{'}, S_{n-i-1}^{'})$, $\cdots$ constitute the set of *valid pairs* $\{VP\}$
Step 5: Let $\{CS\}$ contains the codes $\{0, 1, \cdots, (n-1)\}$. Decide the bit position p (say, LSB) where second control point is to be inserted
Step 6: Take a pair of code C_i & C_j from $\{CS\}$, where C_i and C_j differ only in the pth bit position.
Assign codes C_i & C_j to a valid pair of states $\in \{VP\}$.
Repeat *Step 6* for all the valid pairs $\in \{VP\}$

Example 1: Consider an FSM with 5 states S_0, ..., S_4 with DOF values as 5, 3, 10, 2 and 6 respectively. The sorted list of states in ascending order of DOF is $\{S_3, S_1, S_0, S_4, S_2\}$. Therefore, the valid pairs are (S_3, S_2) and (S_1, S_4). The pool of codes $CS = \{000, 001, 010, 011, 100\}$. Obviously, 101,110 and 111 are the unreachable states. A possible state code assignment where the codes of a valid pair differ at the LSB is given below.
State-code-assignment 1: $(000) \rightarrow S_3$, $(001) \rightarrow S_2$, $(010) \rightarrow S_1$, $(011) \rightarrow S_4$, $(100) \rightarrow S_0$.

Note that the *state-code-assignment1* mentioned above is not unique. For example, the following code assignment is also a valid assignment for this FSM.
State-code-assignment 2: $(010) \rightarrow S_3$, $(011) \rightarrow S_2$, $(000) \rightarrow S_1$, $(001) \rightarrow S_4$, $(100) \rightarrow S_0$.

Among all the states from 0 to $n-1$, there can be $\lfloor n/2 \rfloor$ code pairs where the codes within a pair differ only in the LSB. Selection of a code pair $(2i, 2i+1)$, for a valid pair (S_k, S_l) may affect the gate area of the synthesized FSM. Moreover, the mapping of the codes $2i$ and $2i+1$ to S_k and S_l also demands due attention while optimizing the gate area. These problems are addressed in *Section IV*.

C. Computation of DOF

To compute the DOF in FSM-states, the STT of the FSM is scanned and as each transition is being read, the reachability and the emitability values of the states involved in that transition are updated. The algorithm to find DOF values is noted below.

Algorithm 2: Find_DOF
Input: STT of the FSM with n states $S_0, S_1, ..., S_{n-1}$
Output: $DOF(S_i)$ $\forall i = 0, 1, 2, 3, \cdots, (n-1)$
Step 1: Initialize
reachability $(S_i) \leftarrow 0$, emitability $(S_i) \leftarrow 0$, $\forall i = 0, 1, ..., n-1$
Step 2: Do Step 3 for successive entries of the STT
Step 3: Let the current entry is
$< inputpattern > \quad S_i \quad S_j \quad < outputpattern >$
if $(S_i \ne S_j)$ then do
reachability $(S_j) \leftarrow reachability(S_j) + 1$
emitability $(S_i) \leftarrow emitability(S_i) + 1$
else the transition is a self loop, ignore it
Step 4: Compute $DOF(S_i) = reachability(S_i) \times emitability(S_i) \quad \forall i = 0, 1, ..., n-1$

In an FSM with N primary inputs (PIs) there are 2^N transitions for each state -that is, the STT may consists of $n \times 2^N$ entries, where n is the number of FSM states. Therefore, the worst case complexity of *Algorithm 2* is $O(n \times 2^N)$. However in the STT, there exists a large number of don't cares in $< inputpattern >$, and thus the average complexity reduces to $O(n \times 2^{N-d})$, where on average d number of don't cares exist in an entry of the STT. Still for a large value of N, the complexity is too high and, therefore, we propose a heuristic to compute the DOF in states.

The heuristic to compute DOF: Instead of scanning the entire STT, we apply a sequence of test patterns on the FSM and trace the behavior as it jumps from one state to another. At each step, the *reachability* and *emitability* values of the relevant states are updated. A number of such test pattern sequences are to be applied in succession and the average *reachability* and *emitability* values may be computed from them. Finally, the DOF is computed from the average *reachability* and *emitability.* If t_s is the number of test sequences and l is the average length of the pattern sequences, then complexity reduces to $O(t_s l)$.

Once the DOF values for all the FSM states are computed, the set {VP} of valid pairs can be obtained by executing *Step 3* and *Step 4* of *Algorithm 1.*

IV. State Encoding Targeting Gate Area

This section describes the mapping between a code pair (C_l, C_h) and a valid pair (S_k, S_l) to optimize the gate area of the synthesized FSM.

Definition 4: Given two states S_i and S_j the *intimacy* between S_i and S_j, denoted as $c(S_i, S_j)$, is a non-negative integer that expresses the desired degree of closeness between the codes of S_i and S_j. The closeness is measured in terms of hamming distance.

The pairs of $\{VP\}$ are to be mapped to the code pairs of $\{CP\} = \{(0,1), (2,3), \cdots, ((2k-2), (2k-1))\}$, where $k = \lfloor n/2 \rfloor$. Moreover, if the code pair $(i, j) \in \{CP\}$ is assigned to the state pair $(S_k, S_l) \in \{VP\}$, then the mapping of i and j to S_k and S_l are to be considered. The objective is to find a state assignment which minimizes the cost

$$1/2 \sum_{S_i \ne S_j} c(S_i, S_j) \times h(enc(S_i), enc(S_j))$$

This is an instance of constrained graph embedding problem. We implemented a genetic algorithm based scheme to solve this optimization problem.

A. Construction of Intimacy Graph

An estimate of the gate area of multi-level logic circuits is the number of literals in factored form of the logic function. The literal count of combinational logic (CC) of an FSM can be reduced by extracting the common subexpressions within the minterms. For example, $f(a,b,c,d) = abcd + abc'd'$ is expressed as $f(a,b,c,d) = ab(cd + c'd')$, thereby reducing the literal count from 8 to 6. The algorithms, *Fanin* and *Fanout* [8], are implemented as an attempt to minimize the literal count in a logic function.

A.1 The Fanout Algorithm

The *Fanout* algorithm works well for FSMs with a large number of outputs and a few inputs. If $PO\ z$ is asserted by two states S_i and S_j then in the logic function for z, the expression for S_i and S_j will appear as product terms within two minterms. Similarly, if a certain state S_k is asserted by two present states S_i and S_j then the expression for S_i and S_j shall appear as product terms in two minterms for the next state (NS) lines where S_k has an 1. Therefore, by assigning codes with low hamming distance to S_i and S_j a larger common product term could be obtained. The *Fanout* algorithm tries to find for each pair of states a weight, considering all the output lines and all the states of the NS column, that gives an estimation how close their codes should be. The logical steps of the algorithm is noted below.

Algorithm 3: Fanout
Input: An FSM, N_i= no. of PIs, N_o= no. of POs,
N_s= no. of states.
Output: $c(S_i, S_j)\ \forall\ (S_i, S_j),\ i < j$.
Step 1: Construct O_SET_i for each output O_i
For each row of the STT of the form
$< inputpattern > \quad S_k \quad S_l \quad < outputpattern >$ do
{If O_i in $< outputpattern >$ is 1 then
$O_SET_i = O_SET_i \cup \{S_k\}$
$nw(O_SET_i, S_k) = nw(O_SET_i, S_k) + 1$
endif
}
$//nw(O_SET_i, S_k)$ is the frequency of S_k in $O_SET_i//$
Step 2: Construct NS_SET_i for each FSM state $S_i \in NS$
For each row of the STT compute
$\{NS_SET_l = NS_SET_l \cup \{S_k\}$
$nw(NS_SET_l, S_k) = nw(NS_SET_l, S_k) + 1$ }
Step 3: Compute $c(S_i, S_j)$ for each pair of FSM states (S_i, S_j)
(i) For each FSM state S_k do
$c(S_i, S_j) += nw(NS_SET_k, S_i) \times nw(NS_SET_k, S_j)$
(ii) $c(S_i, S_j) = c(S_i, S_j) \times N_b/2$
For each output O_k do
$c(S_i, S_j) += nw(O_SET_k, S_i) \times nw(O_SET_k, S_j)$

In *Step 3(ii)* a multiplying factor $N_b/2$ (N_b= no. of bits required to encode a state) is used as the average number of 1s in a state code is $N_b/2$ and a common product term appears only when a state code bit is 1.

A.2 The Fanin Algorithm

The *Fanin* algorithm is suitable for FSMs with large number of inputs and outputs. It concentrates on the 1st and the 3rd columns of the STT and tries to give high edge weights to states which are produced by similar inputs and similar sets of present states in order to maximize the number of the largest common subexpressions in the next state lines. The steps are described below.

Algorithm 4: Fanin
Input: An FSM, N_i= no. of PIs, N_o= no. of POs,
N_s= no. of states.
Output: $c(S_i, S_j)\ \forall\ (S_i, S_j),\ i < j$.
Step 1: Construct the sets $I_SET_i(0)$ and $I_SET_i(1)$ for each primary input I_i
For each row of the STT of the form
$< inputpattern > \quad S_k \quad S_l \quad < outputpattern >$ do
{If I_i in $< inputpattern >$ is 0 then
$I_SET_i(0) = I_SET_i(0) \cup \{S_l\}$
$nw(I_SET_i(0), S_l) = nw(I_SET_i(0), S_l) + 1$
endif
If I_i in $< inputpattern >$ is 1 then
$I_SET_i(1) = I_SET_i(1) \cup \{S_l\}$
$nw(I_SET_i(1), S_l) = nw(I_SET_i(1), S_l) + 1$
endif }
Step 2: Construct PS_SET_i for each FSM-state $S_i \in PS$
For each row of the STT do
$\{PS_SET_k = PS_SET_k \cup \{S_l\}$
$nw(PS_SET_k, S_l) = nw(PS_SET_k, S_l) + 1$ }
Step 3: Compute $c(S_i, S_j)\ \forall\ (S_i, S_j), i < j$
For each FSM-state S_k do
$c(S_i, S_j) += nw(PS_SET_k, S_i) \times nw(PS_SET_k, S_j)$
$c(S_i, S_j) = c(S_i, S_j) \times N_b$
For each input I_k do
$\{c(S_i, S_j) += nw(I_SET_k(0), S_i) \times nw(I_SET_k(0), S_j)$
$c(S_i, S_j) += nw(I_SET_k(1), S_i) \times nw(I_SET_k(1), S_j)$ }

B. State Encoding Using GA

A state code assignment is the mappings $f_1 : \{CP\} \rightarrow \{VP\}$ and $f_2 : C \rightarrow S$, where C={0,1,..., n-1} and $S = \{S_0, S_1, \cdots, S_{n-1}\}$ are the set of reachable state codes and the set of FSM states respectively. Here f_2 is subject to f_1. Therefore, our aim is to find f_2, constrained by f_1, such that the cost

$$1/2 \sum_{S_i \neq S_j} c(s_i, s_j) \times h(enc(s_i), enc(s_j))$$

is minimized. We present a GA based solution of this optimization problem.

Genetic Algorithms (GA): Genetic Algorithms (GA) [7] are stochastic search algorithms inspired by natural selection and genetics. The parameters of the search space are encoded in string like structures known as *chromosomes*. The chromosomes (population) undergo evolution for a number of generations. The reproductive plan consists of two genetic operators, namely,

crossover and *mutation*. A pair of parent chromosomes take part in a *crossover* operation. This operator helps to explore the search space by generating new solutions from old ones. The *mutation* operator is applied to impart random change in a solution to ensure that the search space is not closed under crossover. The probability of crossover is normally kept high and that of mutation is kept low. A fitness criterion is applied on each chromosome of a generation. A selection policy, that mimics the Darwinian principle of survival of the fittest, determines which chromosomes should be propagated to the subsequent generation. In the *elitist* model of GA, the best fit chromosome is preserved till the last generation and then presented as the solution. The formulation of GA for the graph embedding problem is being described below.

Chromosomes- A valid pair is actually an unordered pair of states. In this paper an FSM state is represented by an integer. By convention, a valid pair (S_i, S_j) follows the relation $S_i < S_j$. An ordering relation $<$ is then established between two valid pairs.

Definition 5: Given two valid pairs (S_i, S_j) and (S_k, S_l), $(S_i, S_j) < (S_k, S_l)$ if and only if $S_i < S_k$.

For an FSM, the valid pairs can be arranged as $(S'_0, S'_1) < (S'_2, S'_3) < \cdots$, so that it is meaningful to refer $(S'_0, S'_1), (S'_2, S'_3), \cdots$ as the $0^{th}, 1^{st}, 2^{nd}, \cdots$ valid pairs. Similarly, the code pairs are arranged as $(0,1), (2,3), \cdots$ and are being referred to as the $0^{th}, 1^{st}, 2^{nd}, \cdots$ code pairs.

In this paper, a composite chromosomes of the form $< g_0 g_1 \cdots g_{k-1} >< b_0 b_1 \cdots b_{k-1} >$, where $\forall i$, $g_i \in \{0, 1, \cdots, k-1\}$, $b_i \in \{0, 1\}$, and $k = \lfloor n/2 \rfloor$, has been employed to encode the solution. The substring $< g_0 g_1 \cdots g_{k-1} >$ expresses f_1, while $< b_0 b_1 \cdots b_{k-1} >$ expresses f_2. Each g_i (b_i) is obtained by generating a random integer between 0 and $k-1$ (0 and 1).

Interpretation of a chromosome: The ith valid pair is mapped to the ${g_i}^{th}$ code pair provided it has not already been assigned to some other valid pair.

Suppose, the ith valid pair (S_p, S_q) has been mapped to the code pair $(m, m+1)$ (m is even). Here, two distinct code assignments are possible, viz., (i) $\{S_p \leftarrow m, S_q \leftarrow m+1\}$ or (ii) $\{S_p \leftarrow m+1, S_q \leftarrow m\}$. In the current design, we follow assignment (i) if $b_i = 0$ else assignment (ii).

Example 2: Consider an FSM with 7 states $S_0, \cdots, S_6$. The pool of state codes is $\{0, \cdots, 6\}$. Code 7 is the unreachable state. Let $\{VP\} = \{(S_0, S_3), (S_1, S_6), (S_2, S_4)\}$. S_5 is the *lonely state*. Since $\{CP\} = \{(0,1), (2,3), (4,5)\}$, 6 is the *lonely code*.

The lonely code is assigned to the lonely state at the outset. Consider the chromosome $< 220 >< 010 >$. Since $g_0 = 2$, the 0th valid pair (S_0, S_3) is mapped to the 2nd code pair $(4, 5)$. Moreover, since $b_0 = 0$ the assignment would be $S_0 \leftarrow 4$ and $S_3 \leftarrow 5$. The complete state assignment for this chromosome is: $\{S_0 \leftarrow 4$, $S_1 \leftarrow 1$, $S_2 \leftarrow 2$, $S_3 \leftarrow 5$, $S_4 \leftarrow 3$, $S_5 \leftarrow 6$, $S_6 \leftarrow 0\}$.

Fitness function- The fitness function is based on the intimacy graph obtained from algorithms *Fanin* or *Fanout*. A solution S to the constrained graph embedding problem has the *cost*

$C(S) = 1/2 \sum_{S_i \neq S_j} c(s_i, s_j) \times h(enc(s_i), enc(s_j))$

In order to define a suitable fitness function a reference worst cost C_w is used.

$C_w = C_{av} + f\sigma$

where C_{av} is the average cost of the population, σ is the standard deviation and f is a user defined sigma scaling factor. The fitness is then defined by

$F(S) = C_w - C(S)$ if $C(S) < C_w$

$= 0$, otherwise.

Selection, Crossover and Mutation- We followed the *Roulette-wheel* method for selection. Crossover was done on consecutive pairs of chromosomes. For mutation, the gene of the g-part of a chromosome undergoing mutation was replaced by it's $(k-1)^{th}$ complement. The b-part of a chromosome does not take part in mutation.

V. Experimental Results

The proposed scheme has been carried out in the framework of SIS [9]. Extensive experimentation was done on *MCNC benchmark* circuits. The fault efficiencies of the proposed scheme are shown in *Table I*. The single stuck-at fault coverage of the original circuit (synthesized from SIS [9]) is shown in *column 2*. *Column* 3 denote the fault coverage of the circuits synthesized after DOF analysis. *Column 4* reports the fault efficiencies of the circuits synthesized out of *final* state encoding (obtained through the GA based scheme) targeting $BIST$ quality and gate area. A synthesized FSM is tested with a fixed number of test vectors generated by the cellular automata (CA) based pseudo random test pattern generator ($PRPG$) for all the three cases noted in *Column* 2, 3 and 4. To encode FSM states, the DOF in states are computed using the heuristic proposed in *Section III C*.

Table II reports the comparative study of gate area for the circuit synthesized from our scheme and the original circuit, estimated using the output dominance algorithm in *JEDI* [9], noted in *Column 2*. For all the cases the gate area is computed with $MCNC$ $GENLIB$. *Column 3* of *Table II* depicts the gate areas of the sequential circuits synthesized from DOF analysis, whereas the gate areas resulted for *final* state code assignment are noted in *Column 4*. These two results includes the area due to two control XORs placed in the most and least significant PS lines of the FSM. The area overhead (%) in the proposed scheme, pro-

TABLE I

Fault Coverage results for MCNC benchmarks

Circuit name	Fault cov(%) with CA based TPG		
	Original circuit	proposed scheme	
		DOF	Final
ex1	57.66	99.73	99.70
s1	69.07	99.81	99.84
s27	58.59	85.48	84.98
s208	53.03	91.42	92.54
s386	60.30	98.55	98.55
s420	46.67	100	100
s820	36.18	95.71	96.34
s832	43.74	97.09	97.47
s1488	23.95	97.91	97.91
s1494	25.97	97.20	97.32
bbsse	62.69	100	100
styr	42.78	96.27	96.48
keyb	19.31	67.89	67.77
opus	52.63	89.90	89.90
sse	62.64	99.63	99.63
tma	66.38	98.84	98.78
kirkman	67.36	99.21	99.21

TABLE II

Gate Area for Synthesized $MCNC$ benchmarks

Circuit name	Gate Area of Syn. FSM			Overhead (%)
	Original circuit	proposed scheme		
		DOF	Final	
ex1	345	321	321	-7.48
s1	269	309	288	7.06
s27	71	72	67	-5.97
s208	184	128	128	-43.75
s386	199	211	190	-4.74
s420	157	136	136	-15.44
s820	437	509	438	0.23
s832	450	431	418	-7.66
s1488	901	920	883	-2.04
s1494	892	905	870	-2.53
bbsse	199	206	202	1.49
styr	740	662	631	-17.27
keyb	319	312	312	-2.24
opus	153	152	148	-3.38
sse	197	211	202	2.48
tma	264	277	272	2.94
kirkman	262	270	262	0

vided in *column 5*, is computed as

$$area\ \ overhead = \frac{\text{area after final encoding - area in JEDI}}{\text{area after final encoding}}$$

A negative (-) entry in *Column 5* implies that the synthesized area in the proposed scheme is less compared to the area requirement in $JEDI$ (*Column 2*).

The results shown in the *Table I* & II are obtained through a set of similar logic optimization steps (available in SIS) for all the test cases. For fault simulation the synthesized FSM is first converted to 'blif' format and then to bench using public domain tool *blif2bench*.

Performance of the proposed $BIST$ scheme is shown in *Table I*. It is observed that the proposed design employing DOF analysis enhances the detectability of faults by a large extent for all the circuits. Moreover, the design targeting fault efficiency as well as gate area achieves higher fault coverage and simultaneously lesser gate area of the synthesized circuit.

In the current implementation the GA parameters are kept as follows. The population size =50, crossover and mutation probabilities are 0.8 and 0.02 respectively. The simulation was done on the platform of fault simulator *hope*, and the fault coverage figures are expressed in terms of

$$faultcoverage = \frac{\text{Total no. of detected faults}}{\text{Total no. of faults in the CUT}}.$$

VI. Conclusion

The paper presents an elegant technique for $BIST$ structure for sequential machines. The fault efficiencies of $BIST$ed finite state machines are enhanced through the analysis of reachability and emitability parameters of the machine states. The design permits the improvement of parameter values for a set of selected states and thus enhances the detectability of the circuit faults with $BIST$ test pattern generator. The GA based optimization scheme is employed to assign the state codes which effectively minimizes gate area of the synthesized circuit. The experimental results establish the claim of enhanced $BIST$ quality of the proposed design while maintaining the implementation cost. This design can also be further improved by considering the delay constraints at the time of assigning the state codes among the valid pairs of states.

References

[1] V. D. Agrawal, C. R. Kime and K. K. Saluja *'A tutorial on built-in-self-test part 1 & part 2'*, IEEE Design and Test of Computers, March & June 1993.

[2] H. Wunderlich *'The design of random testable sequential circuits'*, Proceedings of 19th Fault-Tolerant Computing Symposium, pp. 110-117, 1989.

[3] F. Muradali, T. Nishida and T. Shimizu *'A structure and technique for pseudorandom-based testing of sequential circuits'*, Journal of Electronic Testing: Theory and Applications, pp. 107-115, 1995.

[4] L. Nachman, K. K. Saluja, S. Upadhaya and R. Reuse *'Random pattern testing for sequential circuits revisited'*, Proceedings of 19th Fault-Tolerant Computing Symposium, pp. 44-52, 1996.

[5] I. Pomeranz and S. M. Reddy *'Improved Built-in test pattern generators based on comparison units for synchronous sequential circuits'*, Proceedings of International Conference on Computer Design, pp. 26-31, 1998.

[6] A. P. Storele and H. Wunderlich *'Hardware-optimal test register insertion'*, IEEE Transactions on Computer-Aided Design, vol. 17, no. 6, pp. 531-539, 1998.

[7] L. Levis *'Handbook of Genetic Algorithms'*, Newyork, Van Nostrand Reinhold, 1991.

[8] S. Devadas, Hi-Keung MA, A. R. Newton and A. S. Vincentelli *'MUSTANG: State Assignment of Finite State Machines Targeting Multilevel Logic Implementations'*, IEEE tcad, Vol. 7, No. 12, pp. 1290-1300, 1988.

[9] *SIS: Asystem for sequential circuit synthesis*, University of California, Berkley, Rep. M92/41, 1992.