A Practical BIST TPG Design Methodology

Chih-Ang Chen and Sandeep K. Gupta

CENG Technical Report 94-34

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California 90089-2562
(213)740-2251

December 1994

A Practical BIST TPG Design Methodology *

Chih-Ang Chen and **Sandeep K. Gupta**
Electrical Engineering – Systems
University of Southern California
Los Angeles CA 90089-2562

Abstract

This paper describes a new technique for the design of BIST TPGs. The TPG design technique identifies *compatible* circuit inputs that can be connected to the same TPG stage. The key idea is that *compatibility* between the circuit inputs is determined by analyzing the circuit logic. Unlike pseudo-exhaustive testing, circuit inputs that fanout to the same output can be compatible, provided that connecting them to the same TPG stage does not cause any loss of fault coverage. Experimental results show that TPGs designed with the proposed technique achieve 100% stuck-at fault coverage in practical test length without adding extra hardware.

*This research was funded by NSF Research Initiation Award no. MIP-9210871

1 Introduction

Built-in self-test (BIST) is gaining acceptance in the VLSI industry as an alternative to conventional scan-based *design-for-testability (DFT)* techniques because of its many advantages. Test application under the conventional scan-based DFT techniques requires the use of an *automatic test equipment (ATE)* to store the test stimuli and the circuit responses. ATEs are expensive, require expertise to operate and maintain, and become outdated in short time as the technology advances. Testing with this approach is inherently slow because test stimuli are serially scanned in, and their responses serially scanned out. Conventional ATEs are normally operated at slow speed because high speed ATEs are very expensive. BIST eliminates the need for ATEs by building test circuitry on the chip. Test stimuli are generated by an on-chip *test pattern generator (TPG)*. The circuit responses to the test stimuli are captured and analyzed by an on-chip *output response analyzer (ORA)*. A *self-testable* chip can be easily tested by initializing the internal test circuitry (which then applies the tests and compresses the responses) and waiting for the pass/fail signal. The testing can be performed *at-speed* — tests are generated and applied using the operating clock — and hence is very fast. Also, the test resources are available during the entire life time of the chip. This significantly simplifies the diagnosis and maintenance of digital systems.

One major issue in BIST is the design of efficient TPGs. An efficient TPG must generate a test sequence of **practical length**, that achieves **high fault coverage** with **minimal hardware overhead**. Many TPG design techniques for stuck-at faults have been proposed. These techniques make trade-offs between test time, fault coverage, and hardware overhead.

Exhaustive testing can achieve high fault coverage with minimal hardware overhead, but the test time is impractically large for circuits with more than 30 inputs. In *pseudo-exhaustive testing* [5, 17], logic feeding each circuit output (referred as a *structural cone* or, simply, a *cone*) is tested exhaustively. Many techniques [2, 17, 29, 30, 36, 37] exist to design TPGs that can generate pseudo-exhaustive test sequences. However, the lower bound on the test length for any pseudo-exhaustive approach is 2^k, where k is the largest *cone size* (the number of inputs in the largest cone). If k is larger than 30, then the circuit must be partitioned into *segments* with fewer inputs to make pseudo-exhaustive testing possible. In *sensitized partitioning* [18], suitable pattern are applied to some primary inputs to isolate segments so that the inputs and outputs of the segment are fully controllable and observable. TPG designs based on this approach require high area overhead because the outputs of some TPG stages must be held constant, while other stages are required to generate exhaustive

patterns for the segments. Besides the high area required for reconfiguring the BIST TPG, complicated BIST control circuitry is required. In *hardware partitioning* [18, 32, 33], multiplexers or *segmentation cells* are inserted in the CUT to isolate the segments. Insertion of these hardware components introduces significant area overhead and *deteriorates circuit performance.*

Pseudorandom testing employs simple TPGs such as *linear feedback shift registers (LFSRs)* to generate pseudorandom sequences. However, long test sequences are required to achieve acceptable fault coverage for *random-pattern resistant* circuits (circuits with many hard-to-detect faults). Use of *multiple seeds* or LFSRs with reconfigurable feedback polynomials can increase fault coverage. But this results in complicated BIST control circuits. In *weighted random testing* [19, 20, 38], a *weight circuit* biases the pseudorandom sequence using precomputed weights. The hardware complexity is normally high due to the need for multiple weights.

Another category of TPG designs are based on *deterministic test set embedding.* Even for random-pattern resistant circuits, the number of tests in the complete deterministic test sets is often small. Many techniques have been proposed to *synthesize* TPGs that can generate short sequences that include these tests. In *store and generate* [1], the entire test set is stored in a *read only memory (ROM)* and a counter is used to apply each test pattern sequentially. Though high fault coverage can be obtained in short test time, the high hardware overhead makes this approach impractical. Techniques to search LFSRs with the best seeds and/or characteristic polynomials to cover a deterministic test set are proposed in [11, 16, 21, 35]. These approaches require low area overhead since only LFSRs are used, but they can only be applied to circuits of small sizes or with regular structures because of high computational complexity. Test set embedding by non-linear feedback shift register (NLFSR) is proposed in [10, 34]. Due to the non-linear nature of the TPG, the area overhead of this approach is high. The method presented in [3] converts a deterministic test set into a linear group, which can then be implemented by a counter and an XOR array obtained by solving linear equations. Though the number of XOR gates in the array can be minimized with algebraic techniques, the area overhead is still high.

There is a fundamental problem associated with deterministic test set embedding. The optimality of the TPGs designed by this approach relies heavily on the test set chosen. Selecting the most suitable test set for a particular embedding scheme is difficult and is seldom addressed. Most existing techniques simply use compact test sets [24, 31] (test sets with minimal number of tests). However, the compact test sets are usually hard to embed and the resulting TPG normally requires longer test length and/or more hardware than

necessary.

In spite of all the BIST TPG design techniques, there is still a lack of a practical solution which, at reasonable hardware overhead, can achieve 100% stuck-at fault coverage in reasonable test time. In this paper, a new TPG design technique for stuck-at testing is introduced. Unlike pseudo-exhaustive testing which only merges inputs that do not belong to the same cone into a *test signal*, our technique explores the logic and determines more *compatible* inputs that can be merged into test signals, even if the inputs belong to the same cone. TPGs designed by the proposed technique guarantee the detection of all detectable single stuck-at faults in the circuit. The proposed TPG designs are LFSRs with primitive feedback polynomials. No segmentation cells are added to the circuit, thereby, keeping the hardware overhead very low. Experimental results show that TPGs that guarantee 100% stuck-at fault coverage in reasonable test length are obtained with the proposed technique. Most importantly, our technique performs well on data path circuits, control circuits, and even circuits that are random-pattern resistant. These circuits have very different characteristics and differ significantly in their test requirements.

Experimental results reported in this paper are obtained by using an ATPG and a *fault simulator (FSIM)* to verify compatibility of each pair of inputs. A greedy strategy is used to merge compatible inputs. In practice, compatibility relations of many inputs can be identified with efficient algorithms based on structural information, testability analysis, and fast redundancy identification algorithms. These techniques can be used as *filters* to verify most input compatibility before resorting to the computationally intensive ATPG/FSIM approach. Some simple filters are described in this paper.

2 General Model of BIST TPGs

A general model of BIST TPGs is shown in Figure 1. The *feedback network* can be *linear* or *non-linear*. A linear feedback network consists of only XOR gates. In general, a TPG with a linear feedback network requires minimal area overhead, but long test time for random-pattern resistant circuits. On the other hand, a TPG with a non-linear feedback requires high area overhead, but may achieve high fault coverage with a short test sequence.

The interconnect network H is a set of logic functions that map the m TPG outputs to the n CUT inputs in the test mode. An interconnect network H is said to be *passive* if H consists of direct connections only. (A connection via an inverter is considered passive because a TPG flip-flop has both inverted and non-inverted outputs.) The *test signals* in

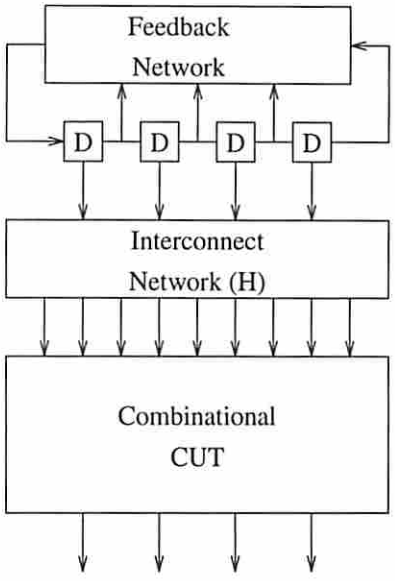

Figure 1: BIST TPG model

verification testing [17] are examples of passive interconnect networks. An interconnect network H is *active* if H realizes some Boolean functions. Examples of active networks are the XOR arrays in linear sums [2] and the weight circuits in weighted random testing [19, 20, 38].

An active interconnect network normally requires high area overhead, but can achieve high fault coverage with a short test sequence. A passive interconnect network is preferred due to its low area overhead. However, existing techniques based on passive interconnection networks normally require long test time for acceptable fault coverage. In the following, we will show that 100% fault coverage can be achieved in reasonable test time by using passive connections.

We will use `c17`, the simplest ISCAS85 benchmark circuit [7], to illustrate the proposed approach. In all the TPG designs that follow, LFSRs with primitive feedback polynomial are used to generate maximal length sequences (or *M-sequences*). If the all zero pattern is required, then a *complete LFSR* (also called a *de Bruijn counter* [4]) can be used. A de Bruijn counter coverts a M-sequence generated by an LFSR (of test length $2^m - 1$) to a *de Bruijn sequence* of length 2^m by inserting the all zero pattern between the pattern $00\cdots01$ and pattern $10\cdots00$.

Example 1 *The circuit* `c17`*, shown in Figure 2, has 5 inputs and 2 outputs. The largest*

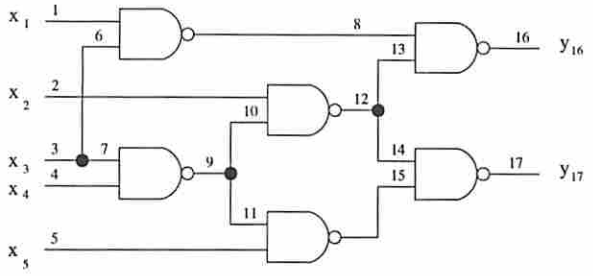

Figure 2: Benchmark Circuit `c17`

	x_1	x_2	x_3	x_4	x_5
y_{16}	1	1	1	1	0
y_{17}	0	1	1	1	1

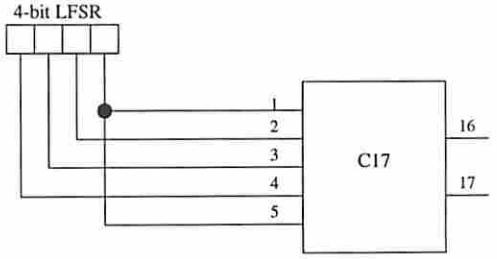

Figure 3: (a) Dependence matrix, and (b) TPG based on verification testing for `c17`.

	x_1	x_2	x_3	x_4	x_5
t_1	1	0	0	1	0
t_2	0	1	1	1	1
t_3	1	1	0	1	0
t_4	1	0	1	0	1

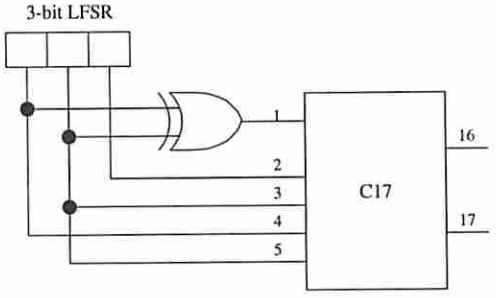

Figure 4: (a) Complete test set-1, and (b) TPG based on embedding test set-1 for `c17` [3].

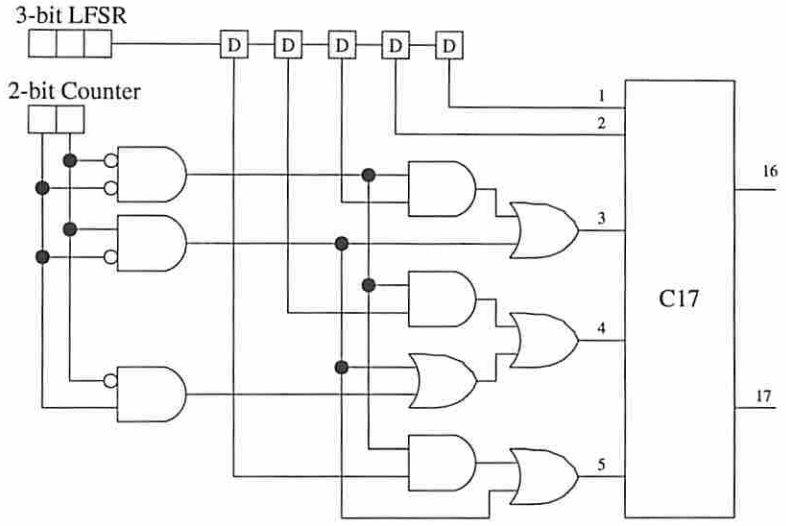

Figure 5: 3-Weight Pseudo-Random TPG for `c17`

cone size is 4. The dependence matrix for `c17` *is shown in Figure 3 (a). Application of verification testing [17] leads to 4 test signals* $\{x_1, x_5\}$, $\{x_2\}$, $\{x_3\}$, *and* $\{x_4\}$. *The TPG design is shown in Figure 3 (b). The pseudo-exhaustive test length for this TPG design is* 2^4. *Since* `c17` *is a* maximal test concurrent *[17] circuit, this test length is also the lower bound for any pseudo-exhaustive approach. With deterministic test set embedding, the test length can be further reduced. In [3], the complete test set given in Figure 4 (a) is used to synthesize a TPG design that uses XOR gates, shown in Figure 4 (b). The test length for this design is* $2^3 = 8$. *The same test set is used in [20] to calculate weights for a weighted random TPG. The resulting circuit is shown in Figure 5. For this small circuit, the weighted network is actually larger than the CUT. (In circuits of practical sizes, the weight circuits are large but not larger than the CUT.)*

Two complete test sets, test set-2 and test set-3 shown in Figure 6 (a) and 7 (a), are given for `c17`. *Compared to the test set in Figure 4 (a), the two test sets will be shown to be "easier to generate". In test set-2, column 1 is identical to column 4, and column 3 is identical to column 5. This implies that the circuit can be tested with a 3-stage LFSR with outputs* $\{q_1, q_2, q_3\}$ *together with the following passive connections:* $q_1 \rightarrow \{x_1, x_4\}$, $q_2 \rightarrow \{x_2\}$, *and* $q_3 \rightarrow \{x_3, x_5\}$. *The TPG design is shown in Figure 6 (b). The test length for this design is* $2^3 = 8$. *If inverted passive connections are allowed, then test set-3 shown in Figure 7 (a) can be used. In test set-3, column 2 and 3 are identical to column 4 and 5 respectively. In addition, column 1 and column 2 are complements of each other. The circuit can be*

	x_1	x_2	x_3	x_4	x_5
t_1	1	1	1	1	1
t_2	1	0	0	1	0
t_3	1	0	1	1	1
t_4	0	0	1	0	1
t_5	1	1	0	1	0

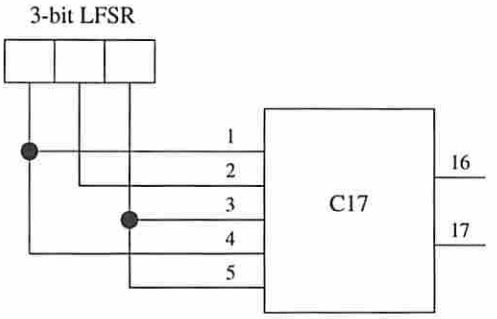

Figure 6: (a) Complete tes set-2, and (b) TPG without inverted interconnections for `c17`

	x_1	x_2	x_3	x_4	x_5
t_1	1	0	1	0	1
t_2	0	1	0	1	0
t_3	0	1	1	1	1
t_4	1	0	0	0	0

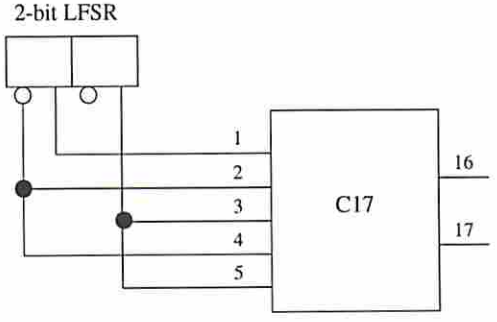

Figure 7: (a) Complete tes set-3, and (b) TPG with inverted interconnections for `c17`

tested with a 2-stage complete LFSR with outputs $\{q_1, q_2\}$ *together with the following passive connections:* $q_1 \rightarrow \{x_1\}$, $\overline{q}_1 \rightarrow \{x_2, x_4\}$, *and* $q_2 \rightarrow \{x_3, x_5\}$. *A complete LFSR is required for this case because all* 2^m *patterns are included in the complete test set. The TPG design is shown in Figure 7 (b). The test length is* $2^2 = 4$. *Hence, by carefully selecting the complete test set, a TPG with lower hardware overhead and shorter test length than all the previously known schemes can be constructed for* `c17`, *using only passive connections.*

3 Proposed TPG Design Technique

A *complete test set*, suitable for TPG designs based on passive interconnection networks, has many identical columns and/or columns which are complemented. Two approaches are proposed to derive such complete test sets which are easier to generate by TPGs. The first approach uses repeated ATPG/FSIM to ensure that no redundancy is introduced if two columns in a complete test set are made identical or complemented. Various techniques can be used to determine compatibility relations of many inputs before resorting to the computationally intensive ATPG/FSIM approach. The second approach uses a modified ATPG to generate complete test sets suitable for TPG designs. This paper describes the first approach.

If the columns corresponding to inputs x_i and x_j are identical in a test set T for an n-input circuit, then an $(n-1)$-stage TPG (capable of generating an M-sequence), obtained by connecting x_i and x_j to the same TPG stage, can generate a test sequence that includes every test in T. If T is a complete test set, then the TPG can generate a test sequence that guarantees 100% fault coverage in half the test length compared to an n-stage TPG. Input connected to the same TPG stage are physically shorted in the test mode. If there exists a complete test set T with identical columns corresponding to the two inputs x_i and x_j, then all detectable faults in the original circuit should remain detectable in the the circuit obtained by shorting inputs x_i and x_j together. An important observation follows.

Observation 1 *If two circuit inputs can be shorted together without introducing redundancy into the circuit, then the two inputs can be merged into a test signal [17] and connected to the same TPG stage.*

Definition 1 *Two inputs* x_i *and* x_j *that can be shorted together without introducing any redundant stuck-at fault are said to be* **compatible.**

Note that compatible inputs defined here may belong to the same cone. Hence, the definition of compatibility is more general than the one used in pseudo-exhaustive testing [2, 17, 29, 30, 36]. Sometimes x_i and x_j cannot be connected to the same TPG output y_k. However, while x_i is connected to y_k, x_j can be connected to $\overline{y}_k$ without any loss of fault coverage. Such connections are called *inverted connections* and considered passive because both y_k and $\overline{y}_j$ are available for all TPG stages.

Definition 2 *Two inputs that can be shortered via an inverter without introducing any redundant single stuck-at fault are said to be* **inversely compatible**.

Compatibility of two inputs x_i and x_j (with or without an inverter) can be checked by actually shorting x_i and x_j and use various techniques to check if any redundancy is introduced. If all detectable faults in the original circuit remain detectable in the new circuit after merging x_i and x_j, then x_i and x_j are compatible and can be combined into a test signal. The proposed technique is called *input reduction.* Starting with completely separate inputs, the compatibilities of each pair of inputs are checked in topological order. Compatible inputs are merged into test signals (hence *reducing* the number of inputs) until no more compatible inputs exist.

One simple approach to determine whether inputs x_i and x_j are compatible is to use ATPG/FSIM to check if any redundancy is introduced in the circuit after merging x_i and x_j. However, direct application of ATPG/FSIM to determine compatible inputs by generating a test for every detectable fault in the original circuit is expensive. Various techniques can be used to reduce the computational effort required to determine compatibility of circuit inputs. For example, a modified version of the complete test set for the original circuit can be used to perform fault simulation on the modified circuit. Since only one pair of inputs are merged in each pass, most faults in the circuit can still be detected by the modified test set. ATPG needs to be performed only on a small subset of faults in each pass. Modern ATPG algorithms use improved implication procedures [9, 12, 13, 14, 26, 25] and can identify circuit redundancies in very short time. These techniques can be used to reduce computation time of ATPG.

For an n-input circuit, there are $2\binom{n}{2} = n^2 - n$ possible pairwise connections. In the worst case, ATPG/FSIM must be used for each pair of inputs to check if they are compatible. This approach is computationally intensive even with efficient redundancy identification algorithms. In practice, compatibility (or incompatibility) of some inputs can be determined in polynomial time. Such polynomial time *filters* can dramatically reduce ATPG effort required for the TPG design.

3.1 Compatibility and Incompatibility Filters

Each input reduction step can change the compatibility relations among the remaining inputs. Hence, the procedures to identify compatibility must be repeated for all pairs of inputs or incrementally for the inputs that may no longer be compatible. Therefore, an efficient implementation of input reduction must use powerful filtering techniques. There are two types of filters:

1. **Compatibility filters:** determine if two inputs are compatible.
2. **Incompatibility filters:** determine if two inputs are incompatible.

A simple filter that can be used is the compatibility filters derived from pseudo-exhaustive testing. In verification testing [17], inputs that do not fanout to the same cones are both compatible and inversely compatible. These compatible inputs can be found in polynomial time by using the structural information. The passive connections derived by techniques based on pseudo-exhaustive testing can be used to construct an initial circuit for further input reduction. This filtering technique can significantly decrease the computation time for circuits with small cones. However, deriving the optimal test signals in pseudo-exhaustive testing is itself NP-complete. Also, cases can be found that starting with a initial circuit with optimal connections calculated by verification testing often does not lead to an optimal TPG based on passive input substitution. For example, if modified version of the circuit `c17` obtained by applying these techniques is used, a 3-stage LFSR is required (instead of a 2-stage LFSR shown in Figure 7).

Another filtering technique that provide incompatibility criterion is based on the following lemma.

Lemma 1 *If primary inputs x_i and x_j are connected to the same gate G, whose input faults are all detectable, then x_i and x_j are both incompatible and inversely incompatible.*

Proof:
Assume that G is a 2-input AND gate. A complete test for a 2-input AND gate must contain the three patterns: 11, 10, and 01. Since all input faults of G are detectable, the three patterns must appear at the two inputs x_i and x_j in order to achieve complete fault coverage. If the inputs x_i and x_j are merged, then the two patterns 10 and 01 can not be applied to the inputs x_i and x_j. If x_i and x_j are merged via an inverter, then the

pattern 11 can not be applied to x_i and x_j. In both cases, at least one stuck-at fault at the inputs of G can not be detected by the TPG design. Hence, the two inputs x_i and x_j are both incompatible and inversely incompatible. The same argument holds for other types of primitive gates with arbitrary number of inputs. □

Therefore, primary inputs connected to the same gate can not be merged into a test signals. Lemma 1 can dramatically reduce computation time for circuits which have many inputs that are connected to the same gates. Note that circuit inputs connected to the same gate G may still be compatible, if G has redundant input faults in the original circuit.

Filters based on pseudo-exhaustive testing and Lemma 1 are derived from structural information. These filters can identify many compatible/incompatible inputs in large circuits. More complicated filters based on fast redundancy identification algorithms that vary in efficiency and accuracy are being designed to speedup our technique. However, the problem of redundancy identification is NP-complete. Therefore, no procedure can guarantee the identification of all redundant faults in polynomial time. ATPG/FSIM must be applied to identify compatible inputs if all filters fail.

3.2 Procedure for Input Reduction

The proposed `input_reduction` procedure is shown in Figure 8. First, a complete test set T for the CUT C is generated. Aborted and redundant faults are removed from the fault list and not considered further. No attempt is made to distinguish between aborted and redundant faults during the ATPG phase. The procedure designs a TPG that guarantees the detection of all detectable faults F. (Modern ATPGs [9, 12, 26, 28] can generate a test set with 100% fault coverage with a small number of backtracks. Aborted faults can always be eliminated by increasing the backtrack limit of the ATPGs.) Next, two candidate PIs x_i and x_j are selected. Various filters are applied to determine compatibility/incompatibility of x_i and x_j. If the filtering is not successful, ATPG/FSIM must be performed to determine the compatibility of the inputs. Assume that the connection is $x_i \rightarrow x_j$. Let C' be the circuit corresponding to C with inputs x_i and x_j shorted. Note that C' has one less input than C. Let T' be the reduced test set obtained by removing the column corresponding to the input x_j. Fault simulation is then performed for the circuit C' using the test set T'. Since only x_i and x_j are merged, most detectable faults in C remains detectable in C' by the tests in the reduced test set T'. Let F' be the remaining faults in C' that can not be detected by any test in T'. The ATPG program is then used to generate new tests for the faults in F'. If any fault $f \in F'$ in the circuit C' is aborted or proven redundant, inputs x_i

```
INPUT_REDUCTION(C)
    # C : input circuit
    F ← {list of testable faults in C}
    T ← {complete test for F}
    for every input x_i
        free(x_i) ← true
    for each pair of CUT inputs (x_i, x_j) where i < j and free(x_j) is true
        compatible(x_i, x_j) ← false
        C' ← C(x_j ← x_i)    # short the two inputs
        if FILTER(x_i, x_j) is not successful then
            T' ← T with j column removed
            if REDUNDANCY_CHECKING(C', T', F) is false then
                compatible(x_i, x_j) ← true
            else
                C' ← C(x_j ← x̄_i)    # short the two inputs via an interter
                if REDUNDANCY_CHECKING(C', T', F) is false then
                    compatible(x_i, x_j) ← true
        if compatible(x_i, x_j) is true then
            C ← C'
            T ← T'
            free(x_j) ← false
    end
```

Figure 8: Pseudo-code for input reduction based on repeated ATPG/FSIM

```
REDUNDANCY_CHECKING(C, T, F)
    F' ← FAULT_SIMULATE(C, T)
    T' ← ATPG(C, F-F')
    T ← T ⋃ T'
    if ∀f ∈ F − F' is testable return false
    else return true
    end
```

Figure 9: Redundancy Checking by ATPG/FSIM

and x_j are incompatible. Consequently, the connection $x_i \rightarrow x_j$ is not valid. A new pair of PIs is selected and the program continues with the original test set T. On the other hand, if all faults are detected, the circuit is updated by shorting x_i and x_j and the original test set T is replaced by the new test set T'. In the procedure, inputs are checked for compatibility in topological order. Compatible inputs are merged greedily. This may affect optimality of the resulting TPGs. But from the experimental results as will be shown in Section 4, this greedy strategy works well for most of the benchmark circuits.

There are several features that reduce the run time of this approach. In early iterations, there exist many inputs that can be merged without introducing redundancy. Hence, the effort to find a pair of compatible inputs is low. At this stage, less time is spent on ATPG, which is computationally expensive because there are many inputs. As more inputs are combined, it becomes more difficult to find compatible inputs. However, since many inputs are already combined into test signals at this stage, ATPG becomes easier because the circuit has fewer inputs and a smaller space is searched. Typically, the number of faults in F' is relatively small. ATPG only needs to be performed on a small fraction of the faults in each pass. Also, the objective is to design a TPG with 100% fault coverage for all detectable faults. If any fault is proven redundant (or aborted), then the two inputs are incompatible. A new iteration is started with a new pair of inputs. Heuristic techniques to identify the faults that are likely to become redundant by merging two inputs are being developed. The ATPG can first target at these faults. If x_i and x_j are indeed incompatible, then the ATPG can be terminated in early stage

4 Experimental Results

Procedure 1 was implemented and applied to a large number of benchmark circuits. The program is implemented inside `SIS` [27], the Berkeley synthesis and optimization tool. We also adopt the ATPG program inside `SIS` as our basic ATPG tool. The ATPG program [28] inside `SIS` is an efficient implementation of [15], a method based on the *Boolean differences.*

The program was used to design TPG for all ISCAS85 [7] and smaller ISCAS89 [6] benchmark circuits. The experimental results are shown in Table 1 and 2. Column 1 gives the circuit name. Column 2 and 3 are the number of PIs and POs respectively. Column 4 gives the largest cone size k. Note that the minimal test length for pseudo-exhaustive exhaustive is greater than or equal to 2^k. Column 5 gives the number of TPG stages to achieve 100% fault coverage with input reduction. Column 6 gives the number of inverted

Table 1: Experimental Results for ISCAS85 Benchmark Circuits

Ckt	No. of inputs	No. of outputs	Largest cone size	No. of TPG stages	No. of inverters
c17	5	2	4	2	1
c432	36	7	36	12	1
c499	41	32	41	9	0
c880	60	26	45	13	7
c1355	41	32	41	11	0
c1908	33	25	33	13	1
c2670	233	140	122	22	7
c3540	50	22	50	17	1
c5315	178	123	67	13	18
c6288	32	31	32	6	7
c7552	207	108	194	28	23

connections (i.e. connections to $\overline{Q}$ of D flip-flops). For example, `c7552` has 207 inputs, 108 outputs, and the largest cone size 194. Note that the lower bound on the number of TPG stages for any pseudo-exhaustive approach is 194. However, with the proposed technique, the number of TPG stages is 28 with 23 inverted connections. Hence, the proposed TPG design *guarantees* the detection of all detectable single stuck-at faults with a test sequence that has 2^{28} or fewer tests.

Most ISCAS85 benchmark circuits are data path circuits, while most ISCAS89 benchmark circuits are control circuits. These two kinds of circuits have very different characteristics, but our program performs equally well on both sets of benchmark circuits. Among the ISCAS85 circuits, the two circuit `c2670` and `c7552` are known to be random-pattern resistant. Table 1 shows great improvement on the number of TPG stages for both circuits. For `c2670`, a 22-stage LFSR TPG is designed by our approach, though the largest cone size is 122. Note that our technique shows great improvements on circuits with large cone size.

We also applied our procedure on selected MCNC logic synthesis benchmark circuits. The circuits are synthesized in `SIS` with two scripts: one is the standard script `rugged`; the other is a script based on the fast extraction algorithm `fx` [22, 23]. It is observed that these synthesized circuits are random-pattern resistant compared to manually-designed circuits [8]. The results are shown in Table 3 and 4. Except for the circuit `rckl`, the proposed procedure obtains improvements on these random-pattern resistant circuits as well.

Table 2: Experimental Results for ISCAS89 Benchmark Circuits

Ckt	No. of inputs	No. of outputs	Largest cone size	No. of TPG stages	No. of inverters
s27	7	4	6	3	2
s208	19	10	18	11	1
s298	17	20	8	7	4
s344	24	26	13	7	3
s349	24	26	13	7	3
s382	24	27	14	7	1
s386	13	13	12	11	1
s400	24	27	14	7	2
s420	35	18	34	19	1
s444	24	27	14	8	2
s510	25	13	20	8	0
s526	24	27	14	13	0
s526n	24	27	14	13	0
s641	54	42	27	15	3
s713	54	42	27	15	3
s820	23	24	21	13	0
s832	23	24	21	13	0
s838	67	34	66	35	1
s953	45	52	18	16	0
s1196	32	32	23	15	4
s1238	32	32	23	15	4
s1423	91	79	59	13	7
s1488	14	25	14	12	0
s1494	14	25	14	12	0
s5378	199	213	60	16	10
s9234	247	250	83	30	15

Table 3: Experimental Results for Circuits Synthesized with Script `fx`

Ckt	No. of inputs	No. of outputs	Largest cone size	No. of TPG stages	No. of inverters
b4	33	23	17	14	4
b10	15	11	15	13	2
chkn	29	7	26	22	1
gary	15	11	15	13	0
in2	19	10	19	13	1
in4	32	20	31	19	2
in5	24	14	21	14	1
in6	33	23	17	14	4
in7	26	10	21	17	0
misg	56	23	15	14	0
rckl	32	7	32	32	0
vg2	25	8	25	17	2
x1dn	27	6	27	15	5

Table 4: Experimental Results for Circuits Synthesized with Script `rugged`

Ckt	No. of inputs	No. of outputs	Largest cone size	No. of TPG stages	No. of inverters
b4	33	23	17	15	0
b10	15	11	15	13	0
chkn	29	7	26	20	2
gary	15	11	15	13	0
in2	19	10	19	13	1
in4	32	20	31	12	7
in5	24	14	21	13	2
in6	33	23	17	15	0
in7	26	10	21	14	0
misg	56	23	15	10	4
rckl	32	7	32	32	0
vg2	25	8	25	12	5
x1dn	27	6	27	14	3

4.1 Discussion

The proposed technique combines compatible inputs into test signals in a manner similar to verification testing. The TPGs designed by this approach have the advantage of minimal area overhead. However, our definition of compatible inputs is more general than the one in verification testing. In verification testing, inputs in the same cone can never be combined into a test signal. This restriction introduces the lower bound on the test length to be 2^k, where k is the largest cone size. In practical circuits, k is normally large. Partitioning techniques are resorted to reduce k. This introduces significant area overhead and may deteriorate circuit performance. In our definition, two input are compatible as long as combining them into a test signal does not make any fault in the circuit redundant. With this relaxation, many inputs become compatible. Practical test length can be achieved by simply combining compatible inputs.

In input reduction, we start with completely separate inputs and merge compatible inputs until no more compatible inputs exist. This is a more natural approach. However, the proposed technique can be applied from the other extreme. We can assume that all inputs are compatible and then separate incompatible inputs until complete fault coverage is achieved. This approach is called *input expansion.* For circuits which can be tested with very few test signals, it may be more beneficial to apply input expansion than input reduction. Consider `c7552` in Table 1. The 207 inputs are "reduced" to 28 test signals. However, it may be faster to "expand" a single test single to 28. This alternative approach is under investigation.

Techniques based on Observation 1 use only passive connections, where a passive interconnect network H is constructed between TPG outputs and CUT inputs. By using Procedure 1, tremendous improvements in test time have already been obtained for most benchmark circuits. If the test length is still not acceptable, then an active interconnect network can be adopted. A new TPG design technique, referred to as *input substitution*, can be developed based on the following generalization of Observation 1.

Observation 2 *If a circuit input x can be represented as a Boolean function f of the remaining inputs without introducing redundancy into the circuit, then x can be connected to a Boolean network that realizes f.*

The function f is called a *substituting function* for the the input x. Here, we substitute a circuit input with a Boolean function of the TPG outputs, and hence the name "input

	x_1	x_2	x_3	x_4	x_5
x_1	-	-	-	1	**1**
x_2	0	-	1	1	-
x_3	-	-	-	-	1
x_4	0	-	-	-	1
x_5	**0**	-	-	-	-

Figure 10: Compatibility matrix for c17

substitution". Observation 1 is a special case of input substitution where the function f is either an identify or an inversion function. Selection of suitable substituting functions are being studied.

The compatibility relation of an n-input circuit can be specified by an $n \times n$ *compatibility matrix*. For example, the compatibility matrix for c17 is shown in Figure 10. The upper triangular matrix is used to represent the compatibility relations, while the lower triangular matrix to represent the inverse compatibility relations. The entry (i, j) is 1 (0) if inputs i and j are (inversely) compatible. A "-" in the entry (i, j) indicates that inputs i and j are incompatible. Various filtering techniques can be used to determine most of the compatibility relations. The remaining entries can be determined by repeated application of ATPG/FSIM. The bold-faced entries in Figure 10 represent compatible inputs as defined by verification testing. More entries in the compatibility matrix imply that more inputs can be combined into test signals.

Our current implementation is based on repeated application of ATPG/FSIM to determine compatible inputs in topological order. The first pair of inputs found to be compatible are merged. Though very encouraging results were obtained with this greedy approach, we believe better solutions can be obtained in shorter time with more systematic approaches. Deriving the optimal passive connections is closely related to the *compatibility theory* which is at the heart of many applications such as logic minimization, state minimization, state assignment, etc. The objective of compatibility theory is to find a minimal number of *compatibility classes*. In our terminology, the objective of input reduction is to find minimal number of test signals, each consisting of compatible inputs. However, there are several issues that make our problem more difficult. In input reduction, each pair of inputs may be compatible and/or inversely compatible. Compatibility theory must be modified to consider both types of compatibility. In verification testing, the compatibility matrix M is constructed from the input/output dependency. Two columns in M which has no conflicting

"1" on every row are compatible and can be merged. Merging two compatible columns does not change values in other columns. In input reduction, inputs belonging to the same cone can still be compatible. However, merging two compatible inputs may make other compatible inputs incompatible. The compatibility matrix must be updated after merging two compatible inputs. Properties of these dynamically-updated compatibility matrices must be carefully explored to obtain more efficient TPG designs. These issues are currently under investigation.

5 Conclusion

A new TPG design technique for testing of stuck-at faults is proposed. Unlike pseudo-exhaustive testing which only merges inputs that do not belong to the same cone into a test signal, our technique explores the circuit logic and determines more compatible inputs that can be merged into test signals, even if the inputs belong to the same cone. A procedure based on repeated ATPG/FSIM to determine compatible inputs is given. Experimental results show that TPGs with **100% fault coverage**, **low area overhead**, and **practical test time** can be constructed by only passive connections. The proposed technique can be used to construct efficient TPGs for data path circuits, control circuits, and synthesized circuits which are random-pattern resistant. Various speedup techniques can be used to reduce the computational time for TPG design. Design of more efficient filters, selection of substituting functions, and systematic approaches to obtain optimal TPGs using compatibility theory will be reported in the future.

References

[1] V. K. Agarwal and E. Cerny. Store and Generate Built-In-Testing Approach. In *Proc. IEEE Int. Conf. on Fault-Tolerant Computing*, pages 35–40, 1981.

[2] S. B. Akers. On the Use of Linear Sums in Exhaustive Testing. In *Proc. IEEE Int. Conf. on Fault-Tolerant Computing*, pages 148–153, June 1985.

[3] S. B. Akers and W. Jansz. Test Set Embedding in a Built-In Self-Test Environment. In *Proc. IEEE Int. Test Conf.*, pages 257–263, 1989.

[4] P. H. Bardell, W. H. McAnney, and J. Savir. *Built-In Test for VLSI: Pseudorandom Techniques.* John Wiley & Sons, New York, NY, 1987.

[5] Z. Barzilai, J. Savir, G. Markowsky, and M. G. Smith. The Weighted Syndrome Sums Approach to VLSI Testing. *IEEE Trans. on Computers*, C-30(12):996–1000, Dec. 1981.

[6] F. Brglez, D. Bryan, and K. Kozminski. Combinational Profiles of Sequential Benchmark Circuits. In *IEEE Int. Symp. on Circuits and Systems*, pages 1929–1934, 1989.

[7] F. Brglez and H. Fujiwara. A Neutral Netlist of 10 Combinational Circuits and a Target Translator in FORTRAN. In *IEEE Int. Symp. on Circuits and Systems*, pages 663–698, June 1985.

[8] C.-H. Chiang and S. K. Gupta. Random Pattern Testable Logic Synthesis. In *Proc. IEEE Int. Conf. on Computer-Aided Design*, 1994.

[9] H. Cox and J. Rajski. On Necessary and Nonconflicting Assignments in Algorithmic Test Pattern Generation. *IEEE Trans. on CAD*, 13(4):515–230, Apr. 1994.

[10] W. Daehn and J. Mucha. Hardware Test Pattern Generation for Built-In Testing. In *Proc. IEEE Int. Test Conf.*, pages 110–113, 1981.

[11] S. Hellebrand, S. Tarnick, J. Rajski, and B. Courtois. Generation of Vector Patterns through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers. In *Proc. IEEE Int. Test Conf.*, pages 120–129, 1992.

[12] T. Kirkland and M. R. Mercer. A Topological Search Algorithm for ATPG. In *Proc. IEEE-ACM Design Automation Conference*, pages 502–508, 1987.

[13] W. Kunz and D. K. Pradhan. Recursive Learning: An Attractive Alternative to the Decision Tree for Test Generation in Digital Circuits. In *Proc. IEEE Int. Test Conf.*, pages 816–825, 1992.

[14] W. Kunz and D. K. Pradhan. Accelerated Dynamic Learning for Test Pattern Generation in Combinational Circuits. *IEEE Trans. on CAD*, 12(5):684–694, May 1993.

[15] T. Larrabee. Efficient Generation of Test Patterns Using Boolean Difference. In *Proc. IEEE Int. Test Conf.*, pages 795–801, 1989.

[16] M. Lempel, S. K. Gupta, and M. A. Breuer. Test Embedding with Discrete Logarithms. In *IEEE VLSI Test Symposium*, pages 74–80, 1994.

[17] E. J. McCluskey. Verification Testing — A Pseudoexhaustive Test Technique. *IEEE Trans. on Computers*, C-33(6):541–546, June 1984.

[18] E. J. McCluskey and S. Bozorgui-Nesbat. Design for Autonomous Test. In *Proc. IEEE Int. Test Conf.*, pages 15–21, 1980.

[19] F. Muradali, V. K. Agarwal, and B. Nadeau-Dostie. A New Procedure for Weighted Random Built-In Self-Test. In *Proc. IEEE Int. Test Conf.*, pages 660–669, 1990.

[20] I. Pomeranz and S. M. Reddy. 3-Weight Pseudo-Random Test Generation Based on a Deterministic Test Set for Combinational and Sequential Circuits. *IEEE Trans. on CAD*, 12(7):1050–1058, July 1993.

[21] I. Pomeranz and S. M. Reddy. A Learning-Based Method to Match a Test Pattern Generator. In *Proc. IEEE Int. Test Conf.*, pages 998–1007, 1993.

[22] J. Rajski and J. Vasudevamurthy. Testability Preserving Transformations in Multi-level Logic Synthesis. In *Proc. IEEE Int. Test Conf.*, pages 265–273, 1990.

[23] J. Rajski and J. Vasudevamurthy. The Testability-Preserving Concurrent Decomposition and Factorization of Boolean Expressions. *IEEE Trans. on CAD*, 11(6):778–793, June 1992.

[24] L. N. Reddy, I. Pomeranz, and S. M. Reddy. Compact Test Sets for Digital Logic Circuits. Technical Report 7-1-1992, Univ. of Iowa, Iowa, 1992.

[25] M. H. Schulz and E. Auth. Improved Deterministic Test Pattern Generation with Applications to Redundancy Identification. *IEEE Trans. on CAD*, 8(7), July 1989.

[26] M. H. Schulz, E. Trischler, and T. M. Sarfert. SOCRATES: A Highly Efficient Automatic Test Pattern Generation System. In *Proc. IEEE Int. Test Conf.*, pages 1016–1026, 1987.

[27] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli. SIS: A System for Sequential Circuit Synthesis. Technical Report Memorandum No. UCB/ERL M92/41, Univ. of California, Berkeley, 1992.

[28] P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Combinational Test Generation using Satisfiability. Technical Report UCB/ERL M92/112, Univ. of California, Berkeley, 1992.

[29] D. T. Tang and C.-L. Chen. Logic Test Pattern Generation Using Linear Codes. *IEEE Trans. on Computers*, C-33(9):845–849, Sept. 1984.

[30] D. T. Tang and L. S. Woo. Exhaustive Test Pattern Generation with Constant Weight Vectors. *IEEE Trans. on Computers*, C-32(12):1145–1150, Dec. 1983.

[31] G. Tromp. Minimal Test Sets for Combinational Circuits. In *Proc. IEEE Int. Test Conf.*, pages 204–209, 1991.

[32] J. G. Udell, Jr. and E. J. McCluskey. Efficient Circuit Segmentation for Pseudo-Exhaustive Testing. In *Proc. IEEE Int. Conf. on Computer-Aided Design*, pages 148–151, 1987.

[33] J. G. Udell, Jr. and E. J. McCluskey. Pseudo-Exhaustive Test and Segmentation: Formal Definitions and Extended Fault Coverage Results. In *Digest of Papers 19th Int. Symp. on Fault-Telerant Computing*, pages 292–298, 1989.

[34] S. J. Upadhyaya and L.-C. Chen. On-Chip Test Generation for Combinational Circuits by LFSR Modification. In *Proc. IEEE Int. Conf. on Computer-Aided Design*, pages 84–87, 1993.

[35] S. Venkataraman, J. Rajski, S. Hellebrand, and S. Tarnick. An Efficient BIST Scheme Based on Reseeding of Multiple Polynomial Linear Feedback Shift Registers. In *Proc. IEEE Int. Conf. on Computer-Aided Design*, pages 572–577, 1993.

[36] L.-T. Wang and E. J. McCluskey. Circuits for Pseudo-Exhaustive Test Pattern Generation. In *Proc. IEEE Int. Test Conf.*, pages 25–37, 1986.

[37] L.-T. Wang and E. J. McCluskey. Condensed Linear Feedback Shift Register (LFSR) Testing — A Pseudoexhaustive Test Technique. *IEEE Trans. on Computers*, C-35(4):367–370, Apr. 1986.

[38] H.-J. Wunderlich. Multiple Distributions for Biased Random Test Patterns. In *Proc. IEEE Int. Test Conf.*, pages 236–244, 1988.