A Low Cost BIST Methodology & Associated Novel Test Pattern Generator

Sen Pin Lin, Sandeep K. Gupta & Melvin A. Breuer

CENG Technical Report 93-33

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

December 1993

A Low Cost BIST Methodology and Associated Novel Test Pattern Generator[*]

Sen-Pin Lin, Sandeep K. Gupta and Melvin A. Breuer

Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, CA 90089-2560

Email: splin@poisson.usc.edu
Phone: (213) 740-2353

[*]This work was supported by the Advanced Research Projects Agency and monitored by the Federal Bureau of Investigation under Contract No. J-FBI-90092. The views and conclusions considered in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

A Low Cost BIST Methodology and Associated Novel Test Pattern Generator

Abstract

The conventional BILBO methodology requires every combinational logic block be driven by a BILBO register (TPG) and drive another BILBO register(SA). The area overhead and performance degradation associated with these BILBO registers can often be excessive. This technical report presents a new BILBO-oriented methodology, called Built-In test for Balanced Structure (BIBS), that significantly reduces the number of BILBO registers used in creating a testable circuit, and thus decreases the area overhead and performance degradation. The concepts of k-pattern detectable faults and k-step functionally testable circuits are introduced. When the BIBS methodology is employed, circuits under test are guaranteed to be 1-step functionally testable and thus a high fault coverage can be achieved. A novel test pattern generator design employing a combination of linear feedback shift registers and shift registers is used to achieve the 1-step functional testability. A new concept, called functionally pseudo-exhaustive testing, is presented that can reduce test time of a balanced BISTable structure when applicable. Preliminary experiments have been performed on several digital filter designs to demonstrate the saving in terms of test hardware when using the BIBS TDM.

Keyword: *Built-in self-test, test pattern generator design, BILBO, functionally exhaustive testing.*

1 Introduction

In the conventional BILBO[1] testable design methodology(TDM)[2] every combinational logic block is driven by a BILBO register and its output drives another BILBO register. To test a combinational logic block the driving BILBO register operates as a test pattern generator(TPG) and the driven BILBO register operates as a signature analyzer(SA). Each combinational logic block in the circuit is a test primitive, called a *kernel*, in the sense that test patterns are applied and output responses compressed outside of the kernel. For a complex circuit the area overhead of these BILBO registers can be excessive. These registers may also increase circuit delay and thus adversely affect circuit performance. In general, kernels need not be restricted to single combinational logic blocks. Several researchers have addressed the problem of reducing high area overhead and/or performance degradation associated with the conventional BILBO TDM[3, 4, 5, 6]. Larger kernels, usually sequential structures, are identified in these approaches that lead to a fewer number of kernels and reduced test hardware. These previous techniques either suffer from poor fault coverage or long test time[4], or lack of a systematic way to identify kernels[5, 6]. In this technical report we present a new TDM, called *Built-In test for Balanced Structure*(BIBS), that significantly reduces test hardware and performance degradation. In addition, good fault coverage and acceptable test time can usually be obtained by employing the BIBS TDM. The methodology presented in [3] is a special case of the BIBS TDM, and may lead to higher area overhead than that of the BIBS TDM. In this study CBILBO registers[7] are only used when necessary since these registers introduce a significant amount of hardware overhead and performance degradation.

This report is organized as follows. In Section 2 the motivation for this study is presented, and the concepts of k-pattern detectable faults and k-step functionally testable circuits are introduced. The circuit graph model employed in this work and the BIBS TDM are described in Section 3. Section 4 presents a novel TPG design scheme employing a combination of LFSRs (linear feedback shift registers) and SRs (shift registers) for 1-step functional testability. Finally Section 5 concludes this report with a brief summary and an outline of current research.

2 Motivation

Increasing the size of kernels usually leads to fewer kernels and thus less test hardware. However, as the size of a kernel increases and sequential kernels are formed, it becomes more difficult to excite a fault and propagate its faulty effect to an observable output. This may lead to a decrease in fault coverage, an increase in test length, and an increase in TPG complexity. It is important to identify kernels that tend to minimize the above problems.

A synchronous sequential circuit is said to be *balanced* if it is acyclic and the sequential lengths of all paths between every pair of combinational blocks in the circuit are the same. Testability of a sequential kernel may be reduced due to circuit imbalance. For example, consider the circuit shown in Figure 1. C is a combinational logic block. F is a fanout block that is fed by a primary input PI and transfers data to its outputs unaltered. R is a register. This circuit is unbalanced since the two paths from F to C have different sequential lengths. To detect a fault f in C may require a test pattern consists of two vectors, u and v, be applied at the inputs to C. This in turn requires that a test sequence of two vectors (v followed by u) be applied at PI. In general, some faults in an unbalanced circuit require a test sequence of vectors for their detection. On the other hand, it has been shown that all detectable stuck-at faults in balanced circuits are *single pattern detectable*[8]. An arbitrary stuck-at fault in such a circuit is tested by applying a test pattern to the inputs of the circuit, clocking the circuit one or more times allowing data to propagate through the circuit, and finally observing the output response. Note that this property of single pattern testability assumes all registers consist of D-type flip-flops(F/F). Single pattern testability is usually no longer guaranteed when designs contain F/Fs that have a hold mode of operation, such as JK or SR F/Fs. In general, a fault is *k-pattern detectable* if there exists an input sequence of length k (or less) that will detect this fault. Every detectable stuck-at fault in the circuit shown in Figure 1 is 2-pattern detectable.

Consider the circuit shown in Figure 2 where C_1 and C_2 are combinational logic blocks and assume that R_1 and R_2 are n-bit registers. By applying all 2^n patterns at R_1 we say that C_2 is tested *1-step functionally exhaustively* even though the output of C_1 may not cover all possible 2^n patterns at the input of C_2. The concept of *1-step* follows because each detectable stuck-at

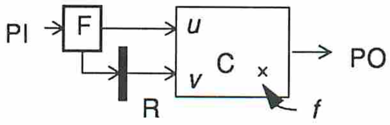


Figure 1: An unbalanced circuit

fault only requires a single test pattern for its detection. The phrase *functionally exhaustive* is used since any test pattern that may occur under functional operation also occurs during the test mode. In fact C_1 is tested both exhaustively and 1-step functionally exhaustively. This circuit is said to be *1-step functionally testable*. That is, by applying all possible test patterns at PI, all detectable stuck-at faults are detected. In practice, it may not be necessary or feasible to apply all possible test patterns to a 1-step functionally testable circuit. For example, in many data path circuits that are 1-step functionally testable, a small portion of the functionally exhaustive test set suffices to detect every detectable stuck-at fault. In addition, when the input width of a kernel is large, say n equals 40 in Figure 2, it may not be feasible to apply all possible test patterns to the kernel.

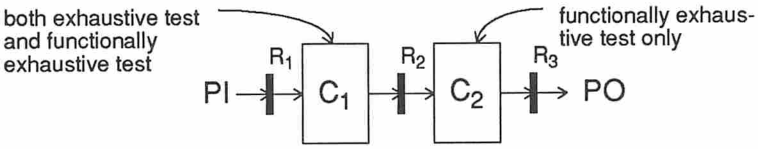

both exhaustive test and functionally exhaustive test
functionally exhaus-tive test only
R1
R2
R3
PI
C1
C2
PO

Figure 2: A 1-step functionally testable circuit

The circuit shown in Figure 1 is 2-step functionally testable since by applying all possible pairs of test patterns at PI, all detectable stuck-at faults are detected. In general, an acyclic circuit is said to be *k-step functionally testable* if for each detectable fault that does not modify the sequential aspects of the circuit, there exists a test sequence of length k (i.e. consists of k vectors) that detects this fault. Note that after applying the last test sequence it may be necessary to apply d random patterns to flush the data through the circuit, where d is the sequential depth of the circuit.

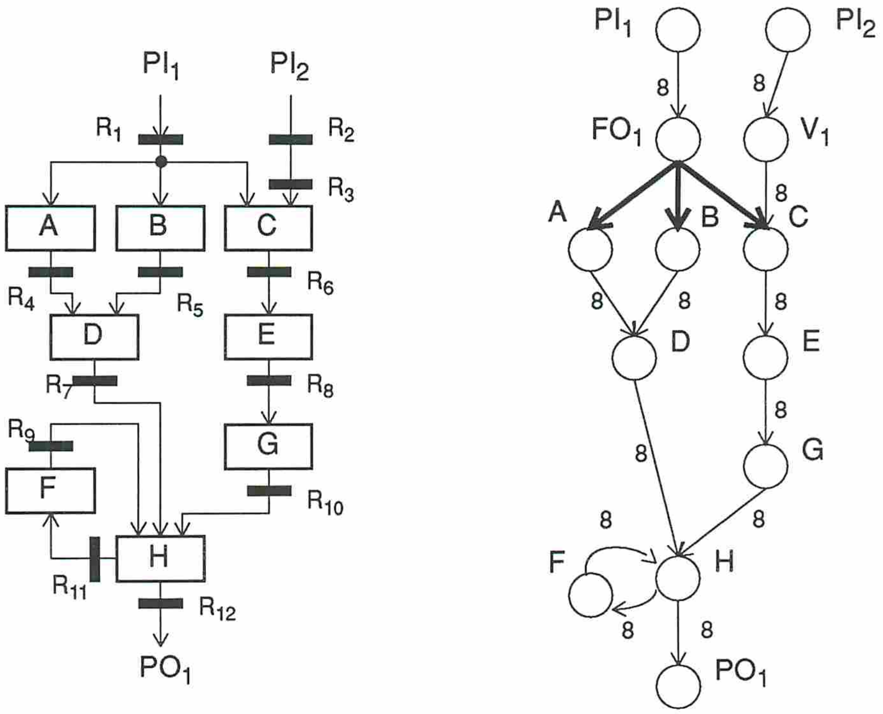
A circuit that is not 1-step functionally testable is considerably harder to test than a 1-step functionally testable circuit since for the former circuit sequences of test patterns are required to ensure a comprehensive fault coverage[8, 9]. In BIST test patterns are generated within the circuit itself often using an LFSR. Conventional LFSRs usually cannot efficiently and effectively generate test sequences in a particular order. Therefore, if conventional LFSRs are used as TPGs then the fault coverage may be low due to the low coverage of faults that require sequence of test vectors. On the other hand, TPG designs that can generate a large subset of all the test sequences of length k are complicated, even for $k = 2$[10]. The area overhead required for such TPGs and the test application time are usually excessive and unacceptable. Therefore, 1-step functionally testable sequential kernels are desired to ensure BISTable circuits with acceptable area overhead, test time and fault coverage.

3 The BIBS TDM

In this section we will first present a circuit graph model that is employed in this report. The definition of a balanced BISTable kernel and its properties are then described, followed by a comparative study of the BIBS TDM and the TDM in [3].

3.1 Circuit Graph Model

A CUC consists of RTL components such as combinational logic blocks, fanout blocks, registers, PIs/POs, and the connections between them. An additional class of logic blocks, called *vacuous blocks*, are introduced which simply consists of wires connecting its inputs and outputs. A vacuous block is present between a pair of registers when one of the registers directly feeds the other register and there is no fanout. A CUC is modeled as a directed graph $G = (V, E, w)$. $v \in V$ represents a combinational block, a PI/PO, a fanout block, or a vacuous block; $e \in E$ represents a connection between two vertices through a register or wires; and $w : E \rightarrow \mathcal{Z}^+$ defines the weight of each edge. A vertex is called a *logic vertex*, *I/O vertex*, *fanout vertex*, or *vacuous vertex* if it represents a combinational logic block, a PI/PO, a fanout block, or a vacuous block, respectively. An edge is called a *register edge* or *wire edge* if it represents a connection between vertices through register or wires, respectively. When e is a register edge, $w(e)$ denotes the width of the register, otherwise $w(e)$ is ∞ (a large number in practice). An example circuit is shown in Figure 3(a) where every register is assumed to be 8 bits wide. The corresponding circuit graph is illustrated in Figure 3(b) where wire edges are represented as bold arcs and the weights for register edges are shown next to the edges in Figure 3(b).



(a)

(b)

Figure 3: (a) An example circuit; (b) corresponding circuit graph

Notice that there is a fanout from the outputs of R_1 the blocks A, B and C in the circuit, hence a fanout vertex FO_1 in required in the circuit graph. Between this fanout vertex and the logic vertices A, B and C are wire edges as denoted by the bold arcs. Furthermore, there is no logic between the two registers R_2 and R_3, thus a vacuous vertex V_1 is required in the circuit graph. A path in G is said to form a *cycle* if it contains at least one register edge and starts and ends at the same vertex. It should be noted that combinational cycles (i.e. cycles composed of vertices and wire edges only) are not allowed in this report since they may cause a circuit to behave asynchronously. A subgraph of G is called an *unbalanced reconvergent-fanout structure (URFS)* if it contains two vertices where two or more paths between them have an unequal number of register edges. The subgraph ($V = \{F, H\}$, $E = \{(F,H), (H,F)\}$) of the circuit graph shown in Figure 3(b) is a cycle; the subgraph ($V = \{FO_1, A, C, D, E, G, H\}$, $E = \{(FO_1, A), (A, D), (D, H), (FO_1, C), (C, E), (E, G), (G, H)\}$) constitutes an URFS.

An *input port* of a logic block C is a collection of inputs at C that are directly driven by logics originating from a common block C'. Similarly an *output port* of C is a collection of outputs at C that directly drive logics in a common block C'. Note that the blocks C and C' must be distinct, otherwise there exists a combinational cycle that is not allowed in this study. An input port is represented by the entrance of an in-coming edge on a vertex in a circuit graph; and an output port is represented by the exit of an out-going edge on a vertex in a circuit graph. For example, there are two input ports and one output port on the combinational logic block D in Figure 3(a). In its corresponding circuit graph shown in Figure 3(b), there are two in-coming edges and one out-going edge connecting to the vertex D.

3.2 Balanced BISTable Kernels

Let S be an arbitrary synchronous sequential circuit with a circuit graph $G = (V, E, w)$.

Definition 1 *S is said to be* **balanced BISTable** *if*

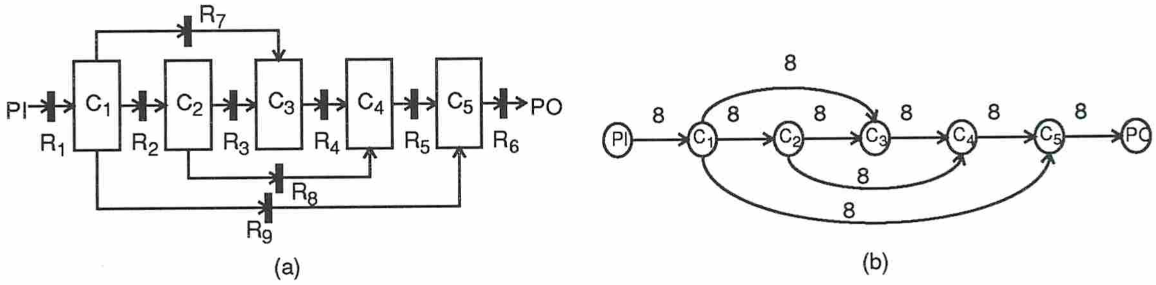
1. *G is acyclic,*

2. *$\forall v_1, v_2 \in V$, all directed paths (if any) from v_1 to v_2 are of equal sequential length, and*

3. *there does not exist a pair of input and output ports on S that is directly driven by and directly drives a common register.*

Clearly the first two requirements force S to be a *balanced structure* as defined in [11]. BALLAST[8] is a design-for-test partial scan TDM that employs balanced structures. It has been shown in [8] that every balanced structure is 1-step functionally testable. Due to this property, only an ATPG system for combinational logic is required. A procedure is presented in [11] to modify a CUC by converting a set of registers to scan registers so that the circuit becomes balanced. A scan register can operate as both a pseudo PI and a pseudo PO simultaneously during the testing of a kernel. On the other hand, a normal BILBO register can operate as either a TPG or a SA, but not both, during the testing of a kernel. Due to this reason the third requirement is necessary to ensure that each pair of input and output ports of a kernel is driven by and drives distinct BILBO registers. This concept can be illustrated by the example below.

Example 1 Consider a circuit S and its corresponding circuit graph shown in Figures 4(a) and (b), respectively. S is not balanced since the sequential lengths of the paths from C_1 to C_3 are different. In a partial scan design a minimal cost solution to balance the circuit is to convert R_3 and R_9 to scan registers. The resulting circuit is illustrated in Figure 5. The BIST equivalence of this solution is to convert R_3 and R_9 (along with R_1 and R_6) to BILBO registers. However, the modified circuit is not balanced BISTable since R_3 and R_9 are used simultaneously as both a TPG and a SA. To make this circuit balanced BISTable, additional registers have to be converted to BILBO registers. One solution is to convert two additional registers, R_7 and R_8, to BILBO registers. This leads to two kernels, each being balanced BISTable, as shown by the shaded blocks in Figure 6. To test the circuit, two test sessions are required. In the first session kernel 1 is tested by configuring R_1 as a TPG and R_3, R_7, R_8 and R_9 as SAs. In the second session kernel 2 is tested by configuring R_3, R_7, R_8 and R_9 as TPGs and R_6 as a SA. □



(a)

(b)

Figure 4: (a) Circuit for Example 1; (b) corresponding circuit graph

BIBS is a BIST TDM that requires kernels under test to be balanced BISTable. Given a CUC, it is usually necessary to modify the circuit by converting a set of registers to BILBO registers in order to satisfy the requirements of the BIBS TDM. The modified circuit is said to be *BIBS testable*. Every edge in the circuit graph that represents a BILBO register is called a *BILBO edge*.

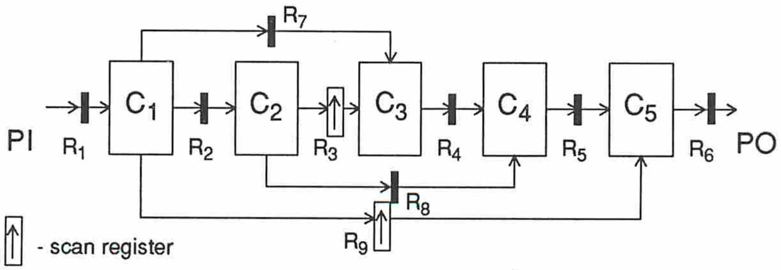


Figure 5: A partial scan design for circuit in Figure 4(a)

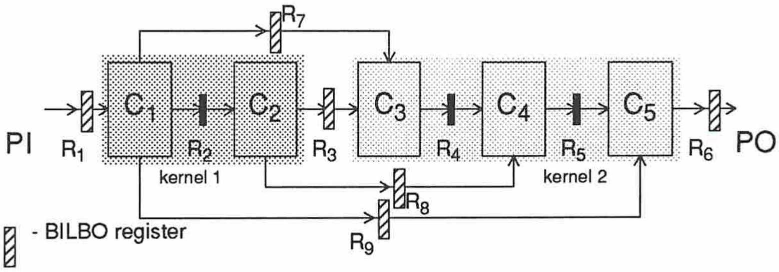


Figure 6: A BISTable design for circuit in Figure 4(a)

3.3 Properties of Balanced BISTable Kernels

Theorem 1 A balanced BISTable kernel is 1-step functionally testable.

Proof By definition a balanced BISTable kernel is a balanced structure. Therefore, the theorem follows from the results presented in [8]. □

Theorem 2 Given a circuit S with a circuit graph G, at least two BILBO edges are required in every cycle and URFS (if any) in G to make S BIBS testable.

Proof Clearly at least one BILBO edge is required in every cycle and URFS to balance G since a balanced circuit has to be free of cycles and URFSs. Let G' be the modified circuit graph (and S' the modified circuit) of G so that there is one BILBO edge in every cycle and URFS. Consider a cycle in G as shown in Figure 7(a) where (u, v) represents a BILBO edge. Let U and V be the combinational logic blocks represented by the vertices u and v, respectively, and R be the BILBO register represented by the edge (u, v). Clearly the input port of V is an input port of S' and the output port of U is an output port of S'. This pair of input and output ports of S' is fed by and feeds the same BILBO register R, thus S' is not balanced BISTable. Similarly, consider an URFS in G that contains only one BILBO edge, say (u, v), as shown in Figure 7(b). Again let U and V be the combinational logic blocks represented by the vertices u and v, respectively, and R be the BILBO register represented by the edge (u, v). The input port of V and output port of U form a pair of input/output ports of S' that is fed by and feeds the same BILBO register and thus S' is again not balanced BISTable. Consequently, at least two BILBO edges are required in every cycle and URFS in G to make S balanced BISTable. □

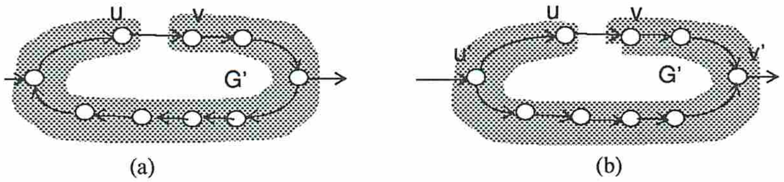


Figure 7: (a) A cycle with one BILBO edge; (b) an URFS with one BILBO edge

Note that if a cycle contains only one register edge, then either an extra register needs to be added in the circuit or a CBILBO register is required. In the former case, the extra register can be transparent during normal functional mode and operates as an LFSR during test mode.

3.4 Comparison of the BIBS TDM and the TDM in [3]

The problem of adding BILBO registers to a CUC so that the circuit can be tested *functionally exhaustively* has been addressed by Krasniewski and Albicki[3]. The criteria used in [3] for adding BILBO registers are:

1. a BILBO register is required for every input port of a combinational logic block if the block has more than one input port,

2. a BILBO register is required for every PI/PO port, and

3. at least two BILBO registers are required in any cycle of the circuit.

Theorem 3 The methodology presented in [3] is a special case of the BIBS TDM.

Proof To prove this theorem it suffices to show that every circuit satisfying the three criteria in [3] consists of only balanced BISTable structures, and the converse is not always true. Assume that there exists a circuit satisfying the three criteria, but containing a non-balanced BISTable structure S. Note that S is connected. Based on the definition of a balanced BISTable structure, S must either (1) contain a cycle, (2) contains an URFS, or (3) feed and be fed by the same BILBO register.

Case 1 is clearly impossible since there will be at least two BILBO registers in any cycle of the original circuit, thus no cycles exist in the modified circuit. Since a BILBO register is required for every input port of a combinational logic block if the block has more than one input port, these BILBO registers essentially break all reconvergent fanout paths, therefore case 2 is not possible either. For case 3, suppose a BILBO register R feeds a combinational logic block V in S and is fed by another combinational logic block U in S (see Figure 8(a)). Since S is connected, U and V must be connected as shown in either Figure 8(b) or Figure 8(c). In Figure 8(b) there is a path from V to U. In other words a cycle exists before register R is converted to a BILBO register. It is necessary to convert another register in this path to a BILBO register as stated by the third criterion above. The extra BILBO register breaks the path and disconnects S, leading to a contradiction. In Figure 8(c) there exists a pair of combinational logic blocks U' and V' that can be the same blocks as U and V, respectively, that form a reconvergent fanout structure. Since V' has more than one input port, BILBO registers are required in front of both input ports of V'. This breaks the path between U' and V' and disconnects S. Thus the assumption that there exists a circuit satisfying the three criteria in [3] but containing a non-balanced BISTable structure is not possible. Therefore every circuit that satisfies the three criteria in [3] consists of only balanced BISTable structures.

Next Consider the circuit shown in Figure 4(a). Clearly all nine registers in the circuit are required to be converted to BILBO registers according to the methodology in [3]. On the other hand, only six BILBO registers are required by the BIBS TDM as shown in Example 1. Therefore a balanced BISTable structure need not satisfy the criteria in [3] and this completes the proof. □

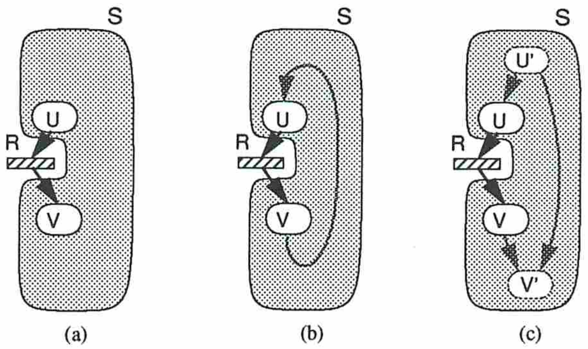


Figure 8: (a) A connected structure that feeds and is fed by a BILBO register R; (b) and (c) possible configurations

Next we consider the example circuit employed in [3] which is shown in Figure 9(a). To make the circuit BISTable using the technique presented in [3], 10 BILBO registers are required as shown in Figure 9(b). The total number of F/Fs that require modification is 52. On the other hand, only 8 BILBO registers that consists of a total of 43 F/Fs are required if the BIBS TDM is employed. Both TDMs partition the circuit into two kernels as illustrated by the shaded blocks in the figure.

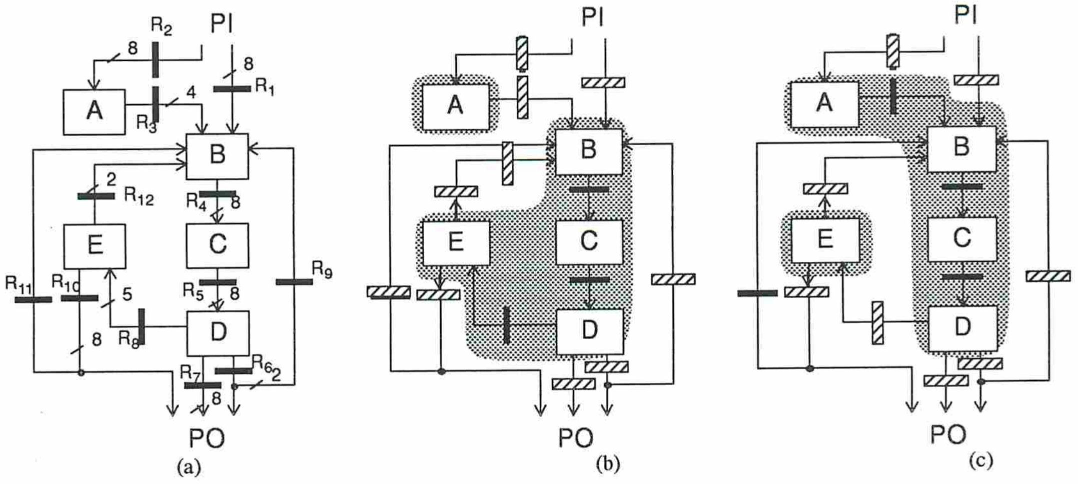


Figure 9: (a) The example circuit employed in [3]; (b) BISTable design using technique in [3]; (c) BISTable design using BIBS TDM

To further compare these two techniques in terms of area overhead, performance degradation,

Circuit	c5a2m	c3a2m	c4a4m

Function	o=(a+b)*(c+d)+(e+f)*(g+h)	o=((a+b)*c+d)*e+f	o=a*(f+g)+e*(b+c)
			p=d*(b+c)+h*(f+g)

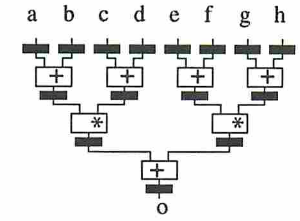
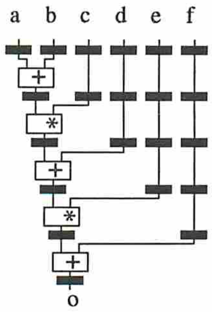
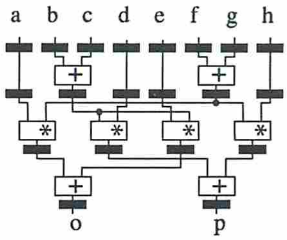
Implementation			
# of gates	2,542	2,218	4,096

Table 1: Summary of the data path circuits

fault coverage and test time, we performed a series of experiments on several data path circuits consisting of adders and multipliers as summarized in Table 1. These circuits are portions of a digital filter design synthesized by MABAL, a CAD tool developed by the USC design automation group[12]. These data paths are all 8 bits wide. Due to the specific application of these circuits, only the 8 least significant output lines of each multiplier feed the next stage in the data path. The comparison between these two TDMs is summarized in Table 2.

In Table 2 for each circuit the first column shows the results of applying the BIBS TDM and the second column shows the results of the approach presented in [3]. We assume pseudo-random testing is employed rather than 1-step functionally exhaustive testing. Since every circuit in Table 1 is balanced BISTable, only registers at the PI and PO need be converted to BILBO registers when employing the BIBS TDM. The remaining circuit is a single kernel and only one test session is required. Whenever a BILBO register is used, it introduces a certain amount of delay, say 1 time unit, due to the extra hardware required. We assume that each BILBO register introduces a delay of 1 time unit. A *maximal delay* is thus calculated for each BISTable circuit that is equal to the maximal number of BILBO registers from a PI to a PO. The maximal delay for each of the circuits is 2 time unites when using the BIBS TDM. For the methodology presented in [3], every register that feeds an input port of each adder and multiplier, as well as the PI and PO

	Circuit	c5a2m		c3a2m		c4a4m	
		BIBS	[3]	BIBS	[3]	BIBS	[3]
1	# of kernels	1	7	1	5	1	7
2	# of test sessions	1	2	1	2	1	2
3	# of BILBO registers	9	15	7	15	10	20
4	Maximal delay	2	4	2	6	2	4
5	# of patterns to achieve 99.5% fault coverage	1,440	1,660	2,060	1,596	1,900	4,128
6	Test time to achieve 99.5% fault coverage	1,440	782	2,060	782	1,900	1,037
7	# of patterns to achieve 100% fault coverage	7,300	4,440	9,240	4,376	19,120	8,688
8	Test time to achieve 100% fault coverage	7,300	2,172	9,240	2,172	19,120	2,172

Table 2: Summary of the experimental results

registers, need be converted to BILBO registers. Each adder and multiplier is a kernel by itself during the test. By using the test scheduler presented in [13], it can be observed that the optimal test schedule for each circuit requires two test sessions. For example, in circuit `c5a2m` the two multipliers can be tested in the first test session and the five adders can be tested in the second test session. The maximal delay for each circuit is significantly larger when using the procedure in [3] than for the BIBS TDM. For instance, the maximal delay for circuit `c3a2m` is 6 time units (from PI a to PO o) in [3] while it is 2 time units when using the BIBS TDM. As can be seen from rows 3 and 4 in Table 2, the BIBS TDM requires significantly less test hardware and circuit delay than the approach in [3] for the data path circuits considered.

The effects on fault coverage and test time for these two approaches are also analyzed and compared as follows. 100% fault coverage of detectable faults can be achieved by both TDMs. Row 7 in the table shows that both TDMs achieve 100% fault coverage if a sufficient number of patterns are applied. The number of patterns required is substantially less than that required by functionally exhaustive testing. The number of patterns required by the approach in [3] is generally less than that by the BIBS TDM. The total test time in [3] can be further reduced if the tests for kernels are properly scheduled. For example, 2,140 and 32 patterns are needed to detect all detectable faults in each multiplier and adder, respectively. To test each kernel of c5a2m in sequence requires 4,440 patterns (i.e. $2,149 \times 2 + 32 \times 5$.) If the tests are scheduled so that the two multipliers are tested in one test session and the five adders are tested in another test session, the total test time is only 2,172 clock cycles. It should be noted that in practice it is not always necessary to achieve 100% fault coverage. In such cases, the differences in test time between these two TDMs are substantially less as can be observed from rows 5 and 6.

The increase in test time for the BIBS TDM seems to be inevitable because larger and more complex structures are tested as kernels. On the other hand, this phenomenon provides a designer with trade-offs between test time, test hardware and performance degradation. In a high performance circuit where performance degradation and area overhead usually have higher priority than test time, the BIBS TDM is an attractive option in making the design BISTable. In this experiment the fault coverage and test time analysis is performed by using a fault simulator where random patterns instead of pseudo-random patterns generated by LFSRs are used. The actual number of test patterns required may vary slightly if LFSRs are employed. It should also be noted that if functionally exhaustive testing is desired, the differences in test time between these two TDMs may be greater since the input widths of the kernels in the BIBS TDM are usually larger. However, for the data path circuits considered, it is not necessary to apply functionally exhaustive testing to achieve 100% fault coverage.

From the above theoretical analysis and experimental results we conclude that the BIBS TDM is very attractive when the area overhead and performance degradation are important attributes in a BISTable circuit. In addition, good fault coverage can be achieved if adequate test patterns are applied to a circuit under test.

4 TPG Design for 1-step Functional Testable Circuits

In the previous section we presented the BIBS TDM that ensures every kernel under test to be 1-step functionally testable. To guarantee a comprehensive fault coverage, all possible patterns (i.e. functionally exhaustive patterns) should be applied to a kernel under test. When a kernel K under test is a combinational logic block with a single input port, a TPG design that configures the input register of K as a maximal length LFSR is sufficient to provide such an exhaustive test set. When K has n input ports and is driven by a set of registers R_1, R_2, ..., R_n, these registers need to be concatenated and made into a single maximal length LFSR. When a kernel is sequential, the TPG design that can provide a functionally exhaustive test set when the sequential lengths of paths from the TPG to blocks in the kernel are unequal may be more complicated. We address the issue of TPG design for balanced BISTable kernels in this section.

Consider the balanced BISTable kernel shown in Figure 10(a). When R_1, R_2 and R_3 are concatenated and made into a maximal length LFSR, a functionally exhaustive test set may not be obtained at the inputs of C_2 and C_3. This is due to the unequal sequential lengths of the paths from the input registers R_1, R_2 and R_3 to C_2 and C_3. To compensate for the imbalance in the sequential lengths of these paths, one and two clock cycles of delay can be added to the paths by inserting transparent registers to the circuit which balance the sequential lengths of these paths (see Figure 10(b)). This, however, would add significant area overhead and the circuit performance may be adversely affected. On the other hand, the same balancing effect can be achieved by adding one D-type F/F before each of R_2 and R_3 if a type 1 LFSR[14] is employed (see Figure 10(c)). This modification works because in a type 1 LFSR L, the data present in the i^{th} stage of L at time t is the same as the data present in the $(i-1)^{st}$ stage of L at time $t-1$ for $i > 1$, where the most significant bit of the LFSR is the first stage. The area overhead for this approach is much smaller than the previous one and no additional performance degradation is introduced apart from the LFSR circuitry.

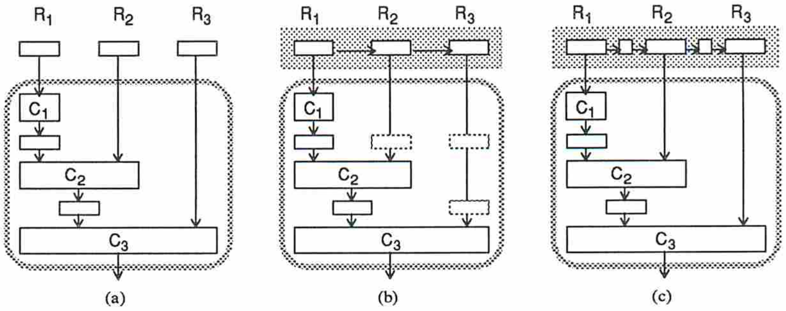


Figure 10: (a) a balanced BISTable kernel; (b) a simple approach to test kernel in (a); (c) proposed approach

As can be seen from the above example, more complicated TPG design may be required to balance the sequential lengths of paths in a sequential kernel. In this section we will first introduce a novel TPG design scheme using a combination of LFSRs and SRs. This scheme can be easily applied to a balanced BISTable kernel with single output port (single cone). We will then extend the design scheme for circuits with multiple output ports (multiple cones). A cone in a circuit consists of all the logic associated with an output port. In some circuits having multiple cones, when the input width of each cone is less than the input width of the entire circuit, the test time for these circuits can often be reduced by using a technique similar to *pseudo-exhaustive testing*[14]. A new concept, called *functionally pseudo-exhaustive testing*, will be introduced to deal with these circuits.

4.1 TPG Design for Single-cone Kernels

A balanced BISTable kernel having a single cone can be represented as a generalized structure shown in Figure 11(a), where C is the combinational logic block that feeds the output port of the kernel. d_1, d_2, ..., d_n are the sequential lengths of the paths from R_1, R_2, ..., R_n, respectively, to C. Each path in Figure 11(a) can be a data path between R_i and C, or a set of parallel data paths between R_i and C all having the same sequential length. For example, referring to the circuit shown in Figure 12(a) the data paths from R_1 through C_1 and C_2 to C_3 and from R_1 through C_1 and C_4 to C_3 are represented by a single path with a sequential length of 2 in Figure 12(c). Note that different balanced BISTable kernels may have the same generalized structure. For example, both the balanced BISTable kernels shown in Figures 12(a) and (b) are represented by the same structure shown in Figure 12(c). It should also be noted that the paths in a generalized structure may not be disjoint. For example, the paths from R_1 to C_3 and from R_2 to C_3 in Figure 12(c) correspond to two data paths in Figure 12(b), namely the path from R_1 through C_1, C_2 to C_3 and the path from R_1 through C_2 to C_3, that share C_2 and the register fed by C_2.

Without loss of generality assume $d_1 \geq d_2 \geq \ldots \geq d_n$. A TPG design for the generalized structure employing type 1 LFSR is illustrated in Figure 11(b). Δ_i extra D-type F/Fs are added in front of R_i, where $\Delta_i = d_{i-1} - d_i$ for $1 < i \leq n$. Therefore, the total number of extra F/Fs in the TPG is $\sum_{i=2}^{n}(d_{i-1} - d_i) = d_1 - d_n$. In other words the TPG consists of $(M + d_1 - d_n)$

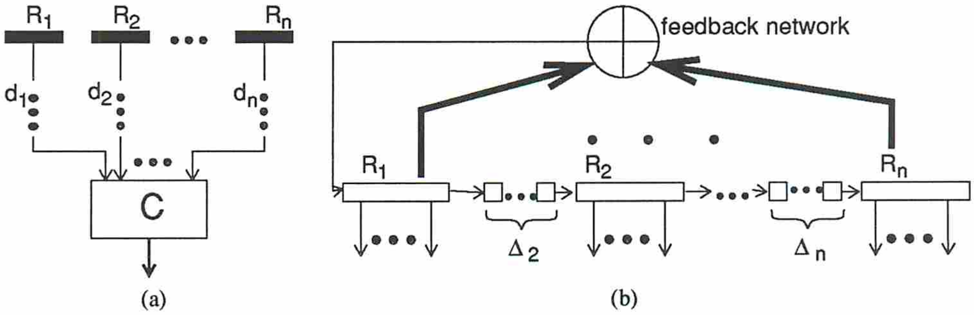


Figure 11: (a) a generalized structure for representing balanced BISTable kernels; (b) a TPG design for the structure

F/Fs, where $M = \sum_{i=1}^{n} |R_i|$ and $|R_i|$ is the width of R_i. Among the string of $(M + d_1 - d_n)$ F/Fs, the first M F/Fs are connected as a type 1 LFSR. In general, the number of extra F/Fs depends on the difference of sequential lengths between the longest path and the shortest path from the TPG to C. For simplicity, we ignore the all-0 pattern in the following discussion. The all-0 pattern can be provided by modifying an LFSR to be a *complete LFSR*[15].

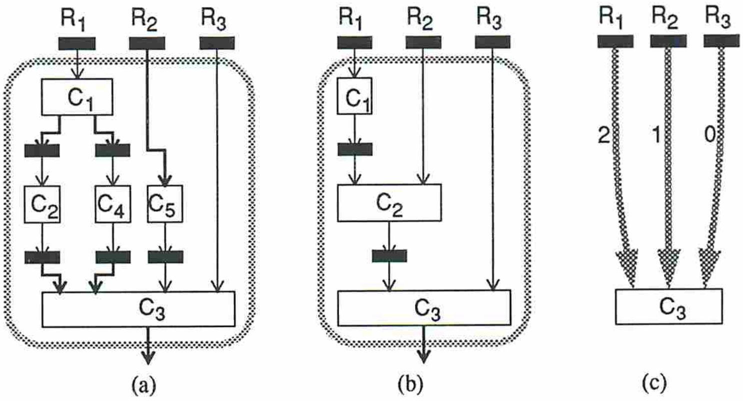


Figure 12: (a) and (b) two different balanced BISTable kernels; (c) a generalized structure for (a) and (b)

Example 2 Consider the balanced BISTable kernel shown in Figure 12(a). Suppose R_1, R_2 and R_3 are all 4-bit registers. A TPG design for this kernel is shown in Figure 13 where the primitive polynomial $x^{12} + x^7 + x^4 + x^3 + 1$ is employed in the LFSR configuration. Only 2 extra D-type F/Fs are required, thus adding 7.2% extra area to a 12-bit BILBO register based on the `magic` layout tool. The test time for the kernel is $2^{12} - 1 + 2$ clock cycles, where the extra 2 clock cycles are needed to flush the last test pattern through the circuit. □

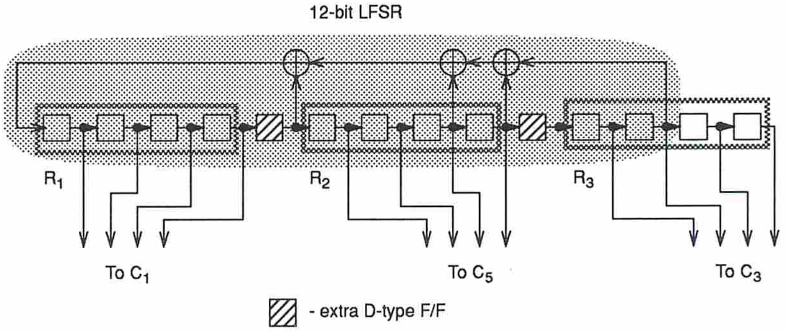


Figure 13: A TPG design for the balanced BISTable kernel in Figure 12(a)

Theorem 4 The TPG design scheme presented above provides a functionally exhaustive test set to a balanced BISTable kernel having a single cone.

Proof To show that a TPG provides a functionally exhaustive test set to a kernel we need to prove that every combinational logic block in the kernel is tested functionally exhaustively when the TPG goes through all possible states of its state space. In a single-cone balanced BISTable kernel, each combinational logic block having a single input port, such as C_5 in Figure 12(a), clearly can be tested functionally exhaustively. Due to the balance property of the kernel, the difference in sequential lengths of the paths from a pair of input registers, say R_i and R_j, to a combinational logic block with fan-ins, such as C_2 or C_3 in Figure 12(b), is the constant $(d_j - d_i)$. For example, the difference in sequential lengths of the paths from R_1 to C_3 and from R_2 to C_3 in Figure 12(b) is 1; and the difference in sequential lengths of the paths from R_1 to C_2 and from R_2 to C_2 is also 1.

We first consider the combinational logic block C that drives the output port of the kernel as shown in Figure 11(a). Let $Q_1(t)Q_2(t)\ldots Q_n(t)$ be a pattern at the inputs of C that occurs during normal functional operation. Then there exists a vector in R_i, say $P_i(t - d_i)$ where $1 \leq i \leq n$, which when present at time $t - d_i$, will produce $Q_i(t)$ at the inputs of C. To exercise all possible functionally patterns at the input of C during test mode, the pattern $P_1(t-d_1)P_2(t-$

$d_2) \ldots P_n(t - d_n)$ should go through all $(2^M - 1)$ combinations. Note that the M-stage LFSR in the TPG shown in Figure 11(b) can be divided into n segments where segment S_i contains $|R_i|$ F/Fs (see Figure 14). For each register R_i, $1 < i \leq n$, there exists a displacement of $(d_1 - d_i)$ F/Fs with respect to segment S_i due to the extra D F/Fs added in the TPG construction. This displacement is equivalent to a time shift between the vectors in R_i and S_i. In other words, the vector in R_i at time t is the same as the vector in S_i at time $t - (d_1 - d_i)$. Therefore, the pattern $P_1(t-d_1)P_2(t-d_2) \ldots P_n(t-d_n)$ in R_1, R_2, ..., R_n is equivalent to $P_1(t-d_1)P_2(t-d_1) \ldots P_n(t-d_1)$ in S_1, S_2, ..., S_n. Clearly the latter can go through all $(2^M - 1)$ combinations since S_1, S_2, ..., S_n are connected as a maximal length LFSR of degree M, thus functionally exhaustive patterns can be applied to C. As was seen above, the difference in the sequential lengths of the paths from R_i and R_j in the TPG to any block C_k is the constant $(d_j - d_i)$ for a balanced kernel. Consequently, the same TPG will generate all patterns required to functionally exhaustively test C_k. Therefore, the proposed TPG can provide a functionally exhaustive test set to a single-cone balanced BISTable kernel. □

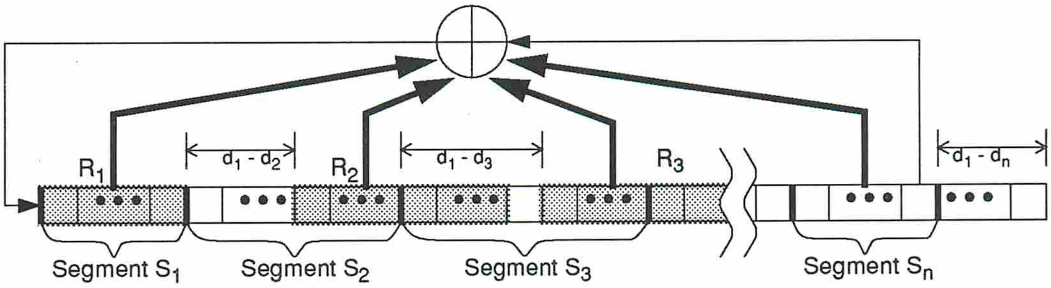


Figure 14: TPG for a balanced BISTable kernel

Corollary 1 The test time to functionally exhaustively test a single-cone balanced BISTable kernel is $2^M - 1 + d$, where M and d are the input width and sequential depth, respectively, of the kernel.

Proof It follows from Theorem 4 that a TPG containing a maximal length LFSR of degree M exists for each single-cone balanced BISTable kernel. From the construction of the TPG, the TPG can apply one test pattern per clock cycle to the kernel. Therefore, it takes $(2^M - 1)$

clock cycles to generate a functionally exhaustive test set and d clock cycles to propagate the last pattern through the kernel. □

Note that even though extra F/Fs are needed in a TPG, they do not increase the test time required to functionally exhaustively test a balanced BISTable kernel. This scheme can be contrasted with the circular self-test path (CSTP) TDM proposed by Krasniewski and Palarski[4]. Here kernels can also be sequential and need not be balanced. It is estimated that to apply an exhaustive test set requires about $T \cdot 2^M$ test patterns, where T varies from 4 to 8. Since kernels need not be balanced, they may not be tested functionally exhaustively.

In general, the sequential lengths of the paths from input registers to an output cone may not be in descending order. We have developed a procedure described below to construct a TPG for a general single-cone balanced BISTable kernel. The procedure first creates a string of N D-type F/Fs (step 2), where N is the kernel width plus the number of required extra D F/Fs. Register cells are then assigned to the created F/Fs (steps 3 and 4). We denote the j^{th} cell of register R_i as $R_{i,j}$. The assignment is done in the order of R_1, R_2, …, R_n, and is based on the difference in the sequential lengths of the paths from each pair of consecutive input registers to the output. For example, let $\Delta_i = d_{i-1} - d_i$, where d_i is the sequential length from R_i to the output. Let $P_{i-1}(t)$ and $P_i(t)$ be the vectors present in R_{i-1} and R_i, respectively, at time t. If $\Delta_i > 0$, $P_i(t)$ will propagate to the output Δ_i clock cycles earlier than $P_{i-1}(t)$ does. To compensate for this imbalance so that functionally exhaustive patterns can be applied to the circuit, R_i should be shifted Δ_i stages away from R_{i-1} in the TPG. This can be achieved by allocating Δ_i F/Fs between the last cell of R_{i-1} and the first cell of R_i (step 4(c)). On the other hand, if $\Delta_i < 0$ then $P_i(t)$ will propagate to the output $|\Delta_i|$ clock cycles later than $P_{i-1}(t)$ does. To compensate for this imbalance, R_i should be shifted $|\Delta_i|$ stages closer to R_{i-1} in the TPG. This can be accomplished by making the first and the last $|\Delta_i|$ cells of R_i and R_{i-1}, respectively, share the same signals (step 4(b)). Two cells are said to share the same signal if they are fed by the same fanout stem. Due to the separation or sharing of register cells, a pair of consecutive input registers R_{i-1} and R_i may not be physically adjacent to each other in the TPG. In other words, a displacement of R_i with respect to R_{i-1} may exist. For a single-cone kernel, the displacement of R_i with respect to R_{i-1} is equivalent to Δ_i. Once register cells are assigned to F/Fs, an M-stage

LFSR can then be constructed (steps 5 and 6). When a signal is shared by two or more register cells, only one of the cells needs to be in the LFSR and the remaining cells are simply fed by the same fanout stem (step 6). This procedure, known as *SC_TPG*, is presented next.

Procedure *SC_TPG*

```
/* INPUT: Input registers $R_1$, $R_2$, ..., $R_n$ and the associated sequential lengths $d_1$, $d_2$, ..., $d_n$.
*/
/* OUTPUT: a TPG design. */

1. Let $r_i$ be the width of $R_i$ and $M$ be $\sum_{i=1}^{n} r_i$.
2. Let $D$ be the difference between $\max_{i=1}^{n} d_i$ and $\min_{i=1}^{n} d_i$. Create a string $S$ of $N$ unlabelled
   D-type F/Fs, where $N = M + D$.
3. Let $R_{1,j}$ be the $j^{th}$ F/F in $S$ and label it as $L_j$ for $j = 1$ to $r_1$. $k = r_1$.
4. For $i = 2$ to $n$ do
   (a) let $\Delta_i = d_{i-1} - d_i$.
   (b) if $(\Delta_i < 0)$ then $k = k - |\Delta_i|$.
   (c) otherwise for $l = k+1$ to $k+\Delta_i$, label the first unlabelled F/F in $S$ as $L_l$. $k = k+\Delta_i$.
   (d) let $R_{i,j}$ be the first unlabelled F/F in $S$ and label it as $L_{k+j}$ for $j = 1$ to $r_i$.
   (e) $k = k + r_i$.
5. Let $L_m$ be the last labelled F/F in $S$. If $(M > m)$ then for $k = m+1$ to $M$, label the first
   unlabelled F/F in $S$ as $L_k$.
6. Connect F/Fs labelled $L_1$, $L_2$, ..., $L_M$ as a maximal length LFSR. If two or more F/Fs have
   the same label, only connect the last F/F (based on their order in the TPG). Configure the
   remaining F/Fs as shift registers where a F/F labelled $L_k$ is fed by a F/F labelled $L_{k-1}$.
```

Theorem 5 Procedure *SC_TPG* generates a minimal test time TPG to functionally exhaustively test a single-cone balanced BISTable kernel.

Proof The TPG generated by Procedure *SC_TPG* employs an LFSR of M stages whose test time $(2^M - 1)$ is minimal for a circuit with M inputs. Therefore, what remains to be shown is the coverage of the patterns generated.

Given a single-cone balanced kernel, the condition that a displacement $(d_j - d_i)$ of R_i with respect to R_j for every pair of input registers R_i and R_j $(j < i)$ is sufficient to functionally exhaustively test the kernel. Procedure *SC_TPG* assigns register cells to the F/Fs in a TPG incrementally where a displacement $(d_{i-1} - d_i)$ of R_i with respect to R_{i-1} is ensured in each iteration. Consequently, the displacement of R_i with respect to R_j obtained by Procedure *SC_TPG* is $(d_{i-1} - d_i) + (d_{i-2} - d_{i-1}) + \ldots + (d_j - d_{j+1}) = d_j - d_i$, thus satisfying the above condition and the kernel can be tested functionally exhaustively. □

Example 3 Consider the generalized structure as shown in Figure 12(c). Assume that each register is 4 bits wide and the sequential lengths from R_1, R_2 and R_3 to C_3 are 1, 2, and 0, respectively. The TPG generated by Procedure *SC_TPG* is shown in Figure 15, where the primitive polynomial $x^{12} + x^7 + x^4 + x^3 + 1$ is employed. The line below the F/Fs illustrates the assignment of register cells to F/Fs. The displacement of R_2 with respect to R_1 is -1. Therefore, $R_{1,4}$ and $R_{2,1}$ share the same signal of LFSR stage L_4. On the other hand, R_2 and R_3 are separated by two F/Fs due to the positive displacement (+2) of R_3 with respect to R_2. Neither of the two F/Fs labelled L_4 can be deleted since both are used in the normal mode of operation of the circuit. □

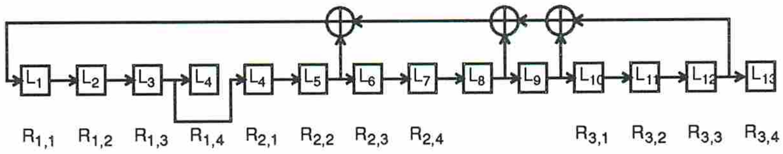


Figure 15: A TPG design for Example 3

It should be noted that if there exists a $\Delta_i < 0$ and $|\Delta_i| > r_{i-1}$, the number of signals shared by register R_i and R_{i-1} is less than $|\Delta_i|$. This can be illustrated by the following example.

Example 4 Consider a kernel with a generalized structure as shown in Figure 16(a). Suppose that both R_1 and R_2 are 4 bits wide. The difference in sequential lengths of the paths from R_1 and R_2 to C results in a displacement -5 of R_2 with respect to R_1. Since the width of R_1 is only 4, R_1 and R_2 can share at most 4 signals in the TPG. A TPG design for a kernel with this generalized structure is shown in Figure 16(b), where the first stage in the LFSR is L_0 instead of L_1. As can be seen from this TPG design, R_1 and R_2 in fact share only 3 LFSR stages L_1,

L_2 and L_3. Note that this type of kernels rarely happen in practical design since the difference in sequential lengths is normally small compared with register widths. Therefore, this example is primarily for theoretical interests and we will assume the absence of this type of kernels in the remaining of this section. □

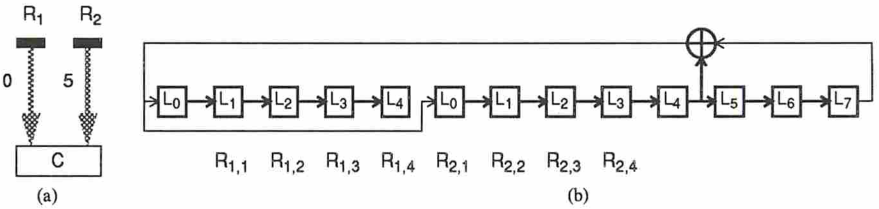


Figure 16: (a) an example generalized structure; (b) TPG design for (a)

4.2 TPG Design for Multiple-cone Kernels

In the previous section, given a pair of consecutive registers R_{i-1} and R_i, the displacement of R_i with respect to R_{i-1} in a TPG is simply the difference in their associated sequential lengths. This may not be true for kernels having multiple cones since a pair of registers may both drive two cones and the differences in the associated sequential lengths from these registers to the cones are unequal. This can be illustrated by the example below.

Example 5 Consider a balanced BISTable kernel with 2 cones as shown in Figure 17(a). We assume that both R_1 and R_2 are 4 bits wide. The first cone, denoted by Ω_1, is associated with output port O_1 and depends on R_1 and R_2. The sequential lengths of the paths from R_1 and R_2 to O_1 are 2 and 0, respectively. The second cone, denoted by Ω_2, is associated with port O_2 and also depends on R_1 and R_2. However, the sequential lengths of the paths from R_1 and R_2 to O_2 are 1 and 0, respectively. To test this kernel with a TPG consisting of R_1 and R_2, the differences in sequential lengths of paths from R_1 and R_2 to O_1 and O_2 should be considered to determine the displacement of R_2 with respect to R_1 in the TPG. Consider O_1 first. Since the difference in sequential lengths of the paths from R_1 to O_1 and from R_2 to O_1 is +2, a displacement +2 of R_2 with respect to R_1 in the TPG is sufficient. Next we consider O_2. A displacement +1 of R_2

with respect to R_1 in the TPG is sufficient due to the sequential length difference between the paths from R_1 to O_2 and from R_2 to O_2. Therefore, a displacement +2 of R_2 with respect to R_1 in the TPG is sufficient to compensate for the sequential length imbalance. This displacement implies that R_1 and R_2 are separated by 2 F/Fs in the TPG as shown in Figure 17(b).

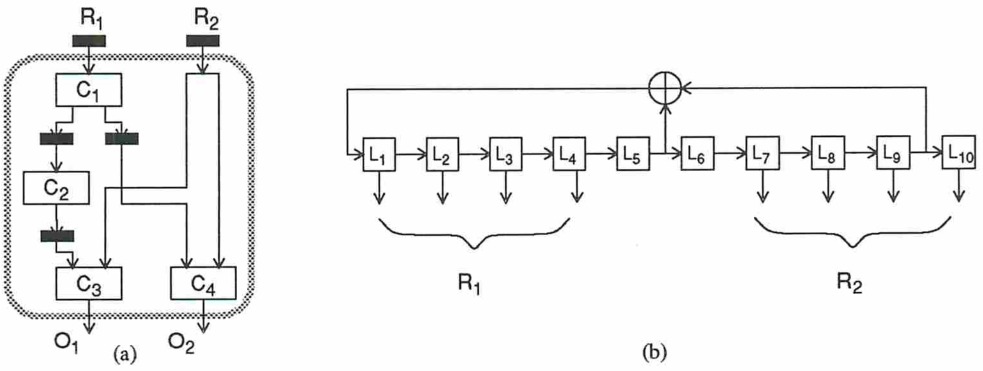


Figure 17: (a) an example multiple-cone kernel; (b) TPG design for (a)

Once the displacement between registers are determined, the size of the required LFSR can then be determined as described below. To simplify the following analysis, we assume that C_1 and C_2 are vacuous blocks. It should be noted that this assumption is only for illustration and the result of the analysis is valid without the assumption. Suppose that at time t, the pattern in the 10 F/Fs shown in Figure 17(b) is $P_1(t)P_2(t)\ldots P_{10}(t)$. At time $t+1$, the pattern present at the inputs of C_4 is $P_1(t)\ldots P_4(t)P_7(t+1)\ldots P_{10}(t+1)$, which is equivalent to $P_1(t)\ldots P_4(t)P_6(t)\ldots P_9(t)$. To functionally exhaustively test C_4 (or cone Ω_2), an LFSR of at least 9 stages is required. Similarly at time $t+2$, the pattern present at the inputs of C_3 is $P_1(t)\ldots P_4(t)P_7(t+2)\ldots P_{10}(t+2)$, which is equivalent to $P_1(t)\ldots P_4(t)P_5(t)\ldots P_8(t)$ and thus an LFSR of at least 8 stages is required to functionally exhaustively test cone Ω_1. Therefore, although the maximal cone size (width of the inputs on which a cone depends) of the circuit is 8, an LFSR of degree 9 is required (see Figure 17(b)). □

Next we extend the procedure presented in the previous section to deal with multiple-cone kernels. Usually we cannot determine the number of F/Fs to be used in a TPG design for a multiple-cone kernel *a priori*, hence in step 1 a string of "adequate" F/Fs are created. In the register cell assignment process (steps 2 and 3), the sequential length information for every pair of input registers (not necessarily consecutive) has to be considered for each cone that depends on both registers (steps 3(a) and 3(b)). For example, suppose $d_{j,x} - d_{i,x} = \delta_{i,j}(x)$ where R_i, R_j $(j < i)$ are a pair of registers on which O_x depends, and $d_{i,x}$ is the sequential length from R_i to output port O_x. There can be more than one cone that depends on both R_i and R_j. Let $\Delta_{i,j} = \max_x \delta_{i,j}(x)$ for all x, where cone Ω_x depends on both R_i and R_j. Clearly the displacement of R_i with respect to R_j must be at least $\Delta_{i,j}$. When R_i and R_j are consecutive registers (i.e. $j = i - 1$), $\Delta_{i,j} > 0$ implies a separation of $\Delta_{i,j}$ F/Fs between R_i and R_j in the TPG; and $\Delta_{i,j} < 0$ implies a sharing of $|\Delta_{i,j}|$ signals between R_i and R_j. When R_i and R_j are not consecutive registers, let L_{k_j} and $L_{k_{i-1}}$ be the labels of the last cells of R_j and R_{i-1}, respectively. Then the sufficient displacement of R_i with respect to R_{i-1} is simply $k_j + \Delta_{i,j} - k_{i-1}$ and the separation or sharing between R_i and R_{i-1} can then be determined.

Theorem 6 The above register assignment guarantees that every register cell is assigned to the F/F with the minimal label.

Proof Since cells of the same register are assigned to F/Fs with contiguous labels, we will consider only the last cell of each register in this proof. We first claim that if the last cell of R_{i-1}, namely $R_{i-1,r_{i-1}}$, is assigned to the F/F with the minimal allowable label, this will lead to the assignment of the last cell of R_i, namely R_{i,r_i}, to the F/F with the minimal label.

Assume the above claim is false and let $L_{k_{i-1}}$ be the minimal allowable label for $R_{i-1,r_{i-1}}$. Then assigning $R_{i-1,r_{i-1}}$ to a F/F labelled $L_{k'_{i-1}}$ will lead to the assignment of R_{i,r_i} to a F/F labelled $L_{k_{i1}}$, while assigning $R_{i-1,r_{i-1}}$ to a F/F labelled $L_{k_{i-1}}$ will lead to the assignment of R_{i,r_i} to a F/F labelled $L_{k_{i2}}$, where $k'_{i-1} > k_{i-1}$ and $k_{i1} < k_{i2}$. However, it can be observed from the assignment process presented above that if there exists no output cone that depends on both R_{i-1} and R_i, R_{i,r_i} will be assigned to the same F/F (i.e. $k_{i1} = k_{i2}$) independent of the assignment of $R_{i-1,r_{i-1}}$; and if there exists an output cone that depends on both R_{i-1} and R_i then $k_{i1} > k_{i2}$. Therefore, the assumption cannot be true and this proves our claim. Notice that registers are assigned incrementally, namely register R_i is assigned only if registers R_1, R_2, ..., R_{i-1} have been assigned. This claim is true for all i, where $1 < i \leq n$, and thus the theorem is true. □

Once register cells are assigned to F/Fs, a sufficient number of LFSR stages for every cone is then determined(step 4). Consider a cone Ω_x that depends on p input registers R_{x_1}, R_{x_2}, ..., R_{x_p} as shown in Figure 18, where R_{x_k} proceeds $R_{x_{k+1}}$ in the TPG for $k = 1$ to $p-1$. Let L_{l_k} and L_{u_k} be the labels assigned to the first and the last cells, respectively, of R_{x_k}.

Theorem 7 $(u_p - l_1 + 1 + d_{x_p,x} - d_{x_1,x})$ LFSR stages are sufficient to functionally exhaustively test the logic in cone Ω_x.

Proof Every pair of input ports of C that depend on register R_{x_i} and R_{x_j}, where $j < i$, are connected to LFSR stages having a span of $(u_i - l_j + 1)$ F/Fs, as can be seen from Figure 18. This number is called the *physical span* of these two input ports. Due to the difference in sequential length, the same ports depend on a span of $(u_i - l_j + 1 + d_{x_i,x}$ - $d_{x_j,x})$ LFSR stages as was illustrated in Example 5. This number is called the *logical span* of these two input ports. The *logical span* of the output port O_x is defined as the the maximum among the logical spans of all (i, j) pairs. This number determines a sufficient number of LFSR stages to functionally exhaustively test the logic associated with cone Ω_x.

We claim that $(u_p - l_1 + 1 + d_{x_p,x} - d_{x_1,x}) \geq (u_i - l_j + 1 + d_{x_i,x} - d_{x_j,x})$ for any (i, j) pair. This is true by examining the following equation.

$$\begin{aligned}
& (u_p - l_1 + 1 + d_{x_p,x} - d_{x_1,x}) - (u_i - l_j + 1 + d_{x_i,x} - d_{x_j,x}) \\
= \;& (u_p - u_i) - (d_{x_i,x} - d_{x_p,x}) + (l_j - l_1) - (d_{x_1,x} - d_{x_j,x}) \\
\geq \;& \Delta_{x_p,x_i} - \delta_{x_p,x_i}(x) + \Delta_{x_j,x_1} - \delta_{x_j,x_1}(x) \\
\geq \;& 0
\end{aligned}$$

It should be noted that the value $(u_p - u_i)$ is the displacement of R_{x_p} with respect to R_{x_i} in the TPG design, which is obtained by considering every i that is less than p. Therefore, it is at least as large as Δ_{x_p,x_i}, namely the required displacement of R_{x_p} with respect to R_{x_i} when only a particular i is considered. Similarly the value $(l_j - l_1)$ is the displacement of R_{x_j} with respect to R_{x_1} and is at least as large as Δ_{x_j,x_1}. Furthermore, $\Delta_{x_p,x_i} \geq \delta_{x_p,x_i}(x)$ and $\Delta_{x_j,x_1} \geq \delta_{x_j,x_1}(x)$ since $\Delta_{x_p,x_i} = \max_x \delta_{x_p,x_i}(x)$ and $\Delta_{x_j,x_1} = \max_x \delta_{x_j,x_1}(x)$. Therefore, $(u_p - l_1 + 1 + d_{x_p,x} - d_{x_1,x})$ LFSR stages are sufficient to functionally exhaustively test cone Ω_x. □

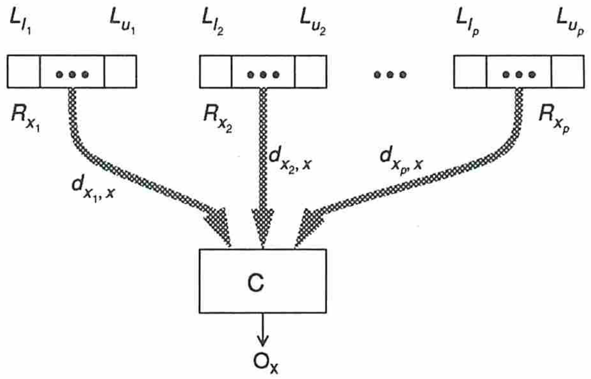


Figure 18: An output cone that depends on R_{x_1}, R_{x_2}, ..., R_{x_p}

Once the sufficient number of LFSR stages to functionally exhaustively test each cone has been determined, the sufficient number of LFSR stages to functionally exhaustively test the entire circuit can then be obtained by taking the maximum among the sufficient number of LFSR stages for individual cones. The procedure for generating a TPG for a multiple-cone kernel, known as *MC_TPG*, is presented next. Note that in this procedure L_{k_i}, $1 \leq i \leq n$, denotes the label for the last cell of register R_i.

Procedure *MC_TPG*

```
/* INPUT: Input registers $R_1$, $R_2$, ..., $R_n$ and output cones $\Omega_1$, $\Omega_2$, ..., $\Omega_m$. Each $\Omega_x$ contains
a set of registers on which $\Omega_x$ depends and the associated sequential lengths. */
/* OUTPUT: a TPG design. */
  1. Let $r_i$ be the width of $R_i$ and create a string $S$ of adequate unlabelled F/Fs.
  2. Let $R_{1,j}$ be the $j^{th}$ F/F in $S$ and label it as $L_j$ for $j = 1$ to $r_1$. $k_1 = r_1$.
  3. For $i = 2$ to $n$ do
       (a) for $j = 1$ to $i - 1$ do
             i. for each cone $\Omega_x$ if $\Omega_x$ depends on both $R_i$ and $R_j$, let $\delta_{i,j}(x) = d_{j,x} - d_{i,x}$.
            ii. let $\Delta_{i,j}$ be the maximum among $\delta_{i,j}(x)$ for all $x$.
           iii. let $\Delta_i(j)$ be $\Delta_{i,j} + k_j - k_{i-1}$.
       (b) let $\Delta_i$ be the maximum among $\Delta_i(j)$ for $j = 1$ to $i - 1$.
       (c) if $(\Delta_i < 0)$ let $k_i = k_{i-1} - |\Delta_i|$.
       (d) otherwise for $k = k_{i-1} + 1$ to $k_{i-1} + \Delta_i$, label the first unlabelled F/F in $S$ as $L_k$.
           $k_i = k_{i-1} + \Delta_i$.
```

   (e) let $R_{i,j}$ be the first unlabelled F/F in S and label it as L_{k_i+j} for $j = 1$ to r_i.

   (f) $k_i = k_i + r_i$.

4. Let $M = 0$. For each cone Ω_x in the circuit do

   (a) determine the sufficient number of LFSR stages to functionally exhaustively test Ω_x, say M', as follows. Let R_{x_1} and R_{x_p} be the first and last registers on which Ω_x depends. Let L_{l_1} and L_{u_p} be the labels of the first and the last cells of R_{x_1} and R_{x_p}, respectively. Then $M' = u_p - l_1 + 1 + d_{x_p,x} - d_{x_1,x}$.

   (b) if $(M' > M)$ then $M = M'$

5. Let L_m be the last labelled F/F in S. If $(M > m)$ then for $k = m + 1$ to M, label the first unlabelled F/F in S as L_k.

6. Connect F/Fs labelled $L_1, L_2, \ldots, L_M$ as a maximal length LFSR. If two or more F/Fs have the same label, only connect the last F/F (based on their order in the layout). Configure the remaining labelled F/Fs as shift registers where a F/F labelled L_k is fed by a F/F labelled L_{k-1}.

Procedure *MC_TPG* is a polynomial time algorithm with a complexity of $O(mn^2)$, where m is the number of cones and n is the number of input registers.

Example 6 Consider the circuit shown in Figure 19(a). The registers are assigned as shown in Figure 19(b) where a displacement of R_2 with respect to R_1 is +2 due to the sequential length difference in Ω_1. Based on this register assignment, the physical span of Ω_2 is 10 (from L_1 to L_{10}). Due to the sequential length difference (i.e. +1), the logical span of Ω_2 is 11 (from L_1 to L_{11}). Therefore, an 11-stage LFSR as shown in Figure 19(b) is sufficient to functionally exhaustively test the circuit. The TPG shown in Figure 19(b) can be obtained by using Procedure *MC_TPG*. □

It should be noted that if the 2 cones in Figure 19(a) are tested separately (in 2 test sessions), the test time is approximately $2 \times 2^8 = 2^9$, which is much smaller than the test time 2^{11} using the TPG described above. In such a case, a *reconfigurable TPG*[16] that can test different cones at different test sessions by changing the configurations of the LFSRs might be desired to reduce the test time. For the kernel in Figure 19(a), a reconfigurable TPG as shown in Figure 20 can be used. When the control line Ω_1/Ω_2 is 0, the TPG is configured to test cone Ω_1 and when the control line Ω_1/Ω_2 is 1, the TPG is configured to test cone Ω_2. Although a reconfigurable

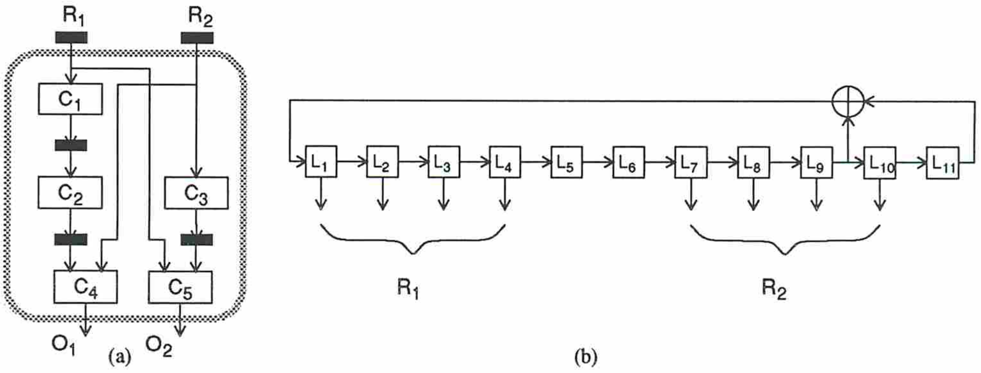


Figure 19: (a) a multiple-cone kernel for Example 6; (b) TPG design for (a)

TPG may reduce the test time for a multiple-cone kernel, the area overhead and performance degradation of such design are usually high. Therefore, a designer can make trade-offs between test time and area overhead when a reconfigurable TPG is more time efficient.

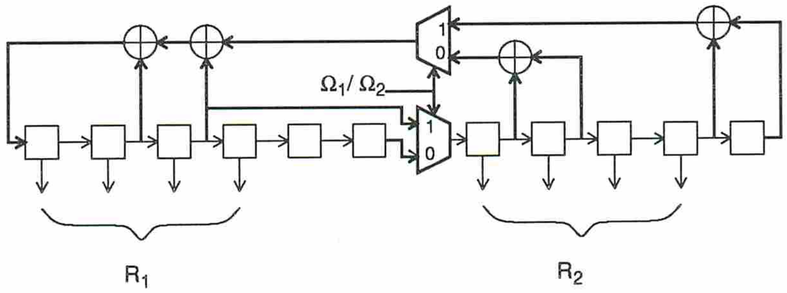


Figure 20: A reconfigurable TPG design for kernel in Figure 19(a)

4.3 Functionally Pseudo-Exhaustive Testing

Example 7 Consider the circuit with 3 cones as shown in Figure 21(a). According to Procedure *MC_TPG*, two F/Fs are required between R_1 and R_2 and one F/F is required between R_2 and R_3. Based on this register assignment, the logical span of Ω_1 is 8 (from L_1 to L_8); the logical span of Ω_2 is 16 (from L_1 to L_{16}); and the logical span of Ω_3 is 8 (from L_7 to L_{14}). Therefore, a maximal length LFSR of degree 16 is sufficient in the TPG design as shown in Figure 21(b). The test time for the circuit using this TPG is approximately 2^{16}.

If the input registers are allowed to be permuted, significant reduction in the LFSR size and test time can be achieved. For example, suppose the input registers are ordered as R_1, R_3 and R_2. A TPG design that employs an LFSR of degree 8 can be obtained by using Procedure *MC_TPG* as shown in Figure 21(c). It should be noted that although the input width of the circuit is 12, the test time to functionally exhaustively test the entire circuit using this TPG design is approximately 2^8. This is due to the fact that each cone only depends on a subset of the inputs. □

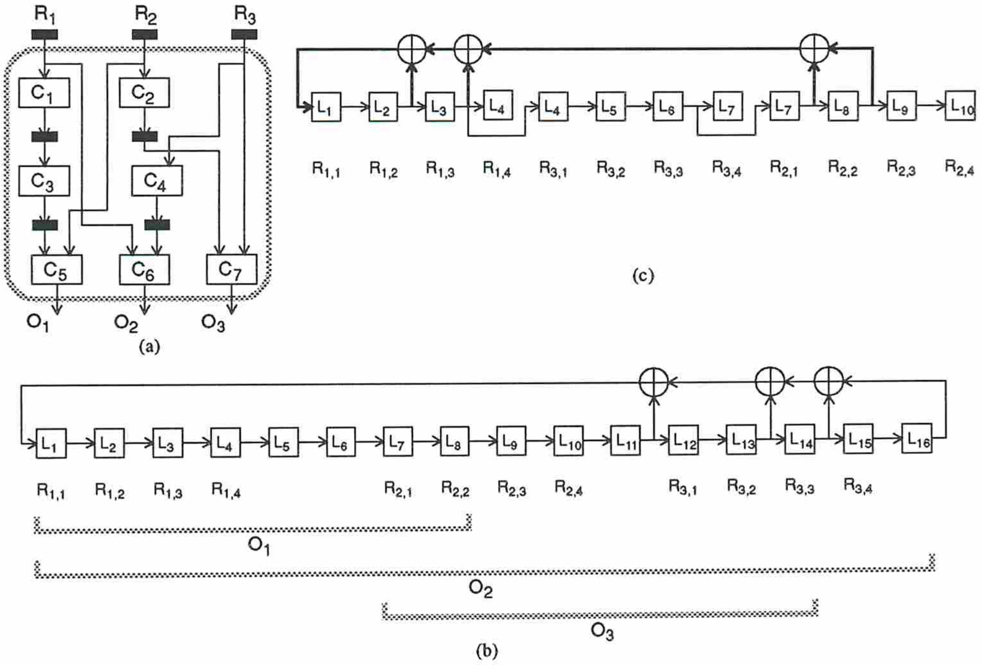


Figure 21: (a) a three cones kernel; (b) a TPG design for (a); (c) an alternative TPG design for (a)

The concept of pseudo-exhaustive testing has been widely employed in testing combinational circuits, where exhaustive patterns are applied to individual output cones of a combinational circuit. This technique ensures the detection of all detectable stuck-at faults in the circuit and the associated test time is usually less than that of exhaustive testing. As can be observed from Example 7, functionally exhaustive patterns can be applied to individual output cones of a balanced BISTable kernel and the test time of this technique may be less than that of applying exhaustive patterns to the entire kernel. This test strategy is called *functionally pseudo-exhaustive testing.* Functionally pseudo-exhaustive testing ensures the detection of all stuck-at faults in the circuit that would interfere with normal operation.

Research problems in pseudo-exhaustive testing, such as finding the minimal test signal set and test sets[17, 18] and TPG design for pseudo-exhaustive testing[19, 17, 16] have been extensively studied. It has been shown that the problem of generating an optimal pseudo-exhaustive test set is NP-complete[17] and the problem of minimal test time TPG design is also hard. The problem of TPG design for functionally pseudo-exhaustive testing differs from this previous work in two aspects. Namely register level instead of gate level components are considered, and the concepts of time shift and sequential length difference must be taken into account. We can easily extend the procedure in finding the minimal number of test signals presented in [17] to deal with register level signals efficiently as illustrated below.

Example 8 Consider the the circuit shown in Figure 21(a). The cone dependency[17] is shown in the dependency matrix D.

$$D = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$D_{ij} = 1$ if cone Ω_i depends on R_j. Clearly 3 test signals, each of which being a group of 4 wires, are required which implies an LFSR of 12 stages is needed in the TPG design. Therefore, the test time using this approach will be approximately 2^{12} which is much greater than 2^8 obtained in Example 7 when employing Procedure *MC_TPG* and the register permutation. □

As was seen in the above example, the result obtained by the extended minimal test signal procedure is not optimal in terms of the LFSR size and test time. This is due to the lack of the ability to handle sequential length information in this procedure. In addition, once a TPG is configured based on the RTL test signals generated, the connection between the TPG and the kernel under test needs to be done in the gate level based on the sequential length differences. Therefore, this simple extension of an existing procedure that finds the minimal number of test signals[17] does not always generate a minimal test time TPG design for a multiple-cone kernel.

As illustrated in Example 7, the test time for a multiple-cone kernel can vary according to the ordering of the input registers. Under certain register orderings, test time to detect all detectable stuck-at faults may be less than that of applying a functionally exhaustive test set and thus functionally pseudo-exhaustive testing can be applied. It is often desirable to reduce the test time for a kernel by finding a best register ordering. This can be achieved by executing Procedure *MC_TPG* once for each input register ordering. In practice, the number of input registers of a multiple-cone kernel is usually small, say less than 5. In addition, Procedure *MC_TPG* is a polynomial time algorithm. Therefore, it is feasible to employ the above approach to reduce the test time for a multiple-cone kernel. It should be noted that the test time for a multiple-cone kernel is lower bounded by 2^w, where w is the maximal cone size of the kernel. Therefore, if this lower bound is achieved by a particular register ordering, a minimal test time TPG has been obtained and the test time reduction process can be terminated.

5 Conclusion

We have presented a BILBO-oriented methodology, called BIBS, which in general requires less test hardware and performance degradation than the conventional BILBO methodology. The BIBS TDM employs balanced BISTable structures that are 1-step functionally testable, thus TPGs with low area overhead and acceptable test time can be used to achieve high fault coverage. The methodology presented in [3] has been shown to be a special case of the more general BIBS methodology. A novel TPG design scheme for the BIBS TDM that can be used in functionally exhaustive testing and functionally pseudo-exhaustive testing was presented. From this study it can be observed that the size of a TPG depends on the difference in sequential lengths from the TPG to outputs of a balanced BISTable structure. The test time for such a structure also depends on this sequential length difference and the maximal cone size of a structure. In practice these two factors need to be considered during the process of finding balanced kernels to ensure acceptable test hardware overhead and test time. Preliminary experiments have been performed on three different digital filter designs to compare the BIBS TDM with the methodology presented in [3]. The experimental results show that the BIBS TDM requires fewer BILBO registers and leads to less performance impact than the TDM in [3]. Both approaches achieve 100% fault coverage of detectable faults using pseudo-random patterns and the test application time is substantially less than that of using functionally exhaustive testing.

Currently we are working on procedures to allocate test hardware and identify kernels in making a circuit BIBS testable. A polynomial time algorithm for generating minimal cost BIBS testable design has been implemented for a class of circuits. Procedures to generate BIBS testable design for more general circuits are being implemented. The necessary and sufficient condition for a k-stage LFSR to functionally exhaustively test a balanced BISTable kernel having n inputs, where $k \geq n$, has been identified. A procedure to generate a TPG using the minimal number of F/Fs and LFSR stages to functionally exhaustively test a balanced BISTable kernel can be developed using this condition. The development of such a procedure remains an open problem. The BIBS TDM, procedures to generate BIBS testable circuits, and the TPG design procedure for BIBS testable circuits are integrated with the Built-In Test System(BITS) developed by the USC test group. BITS is an integrated CAD test system that reads in a circuit (in EDIF description) to be made BISTable, reorganizes the circuit into a RTL description useful for the BISTable design process, systematically explores the BISTable design space to provide a family of solutions, generates an optimal test schedule, designs low area and high fault coverage TPGs and SAs, synthesizes a test controller, and finally exports the fully testable circuit to another EDIF description for layout. More experiments are being performed using the BITS system to demonstrate the advantages of employing the BIBS TDM.

References

[1] B. Konemann, J. Mucha, and G. Zwiehoff. Built-In Logic Block Observation Technique. In *Proc. Int'l. Test Conf.*, pages 37–41, 1979.

[2] M.S. Abadir and M.A. Breuer. A Knowledge Based System for Designing Testable VLSI Chips. *IEEE Design and Test of Computers*, 3(4):56–68, August 1985.

[3] A. Krasniewski and A. Albicki. Automatic Design of Exhaustively Self-Testing Chips with BILBO Modules. In *Proc. 1985 Int'l. Test Conf.*, pages 362–371, 1985.

[4] A. Krasniewski and S. Pilarski. Circular Self-Test Path: A Low-Cost BIST Technique for VLSI Circuits. *IEEE Transactions on Computer-Aided Design*, pages 46–55, January 1989.

[5] M.S. Abadir, J. Newman, D. D'Souza, and S. Spencer. Partitioning Hierarchical Designs for Testability. In *Proc. of Intl. test Conf.*, pages 174–183, October 1991.

[6] S.P. Lin. A Testable Design System Using BILBO-oriented Methodologies. Ph.D Thesis Proposal, University of Southern California, June 1992.

[7] L.T. Wang and E.J. McCluskey. Concurrent Built-In Logic Block Observer (CBILBO). In *Int'l. Symp. on Circuits and Systems*, volume 3, pages 1054–1057, 1986.

[8] R. Gupta, R. Gupta, and M.A. Breuer. BALLAST: A Methodology for Partial Scan Design. In *Proc. 19th Int'l. Symp. on Fault-Tolerant Computuing*, pages 118–125, June 1989.

[9] D. Vir Das, S.C.Seth, and V.D. Agrawal. Estimating the Quality of Manuafactured Digital Sequential Circuits. In *Proc. 1991 Int'l. Test Conf.*, pages 210–217, 1991.

[10] Chih-Ang Chen and Sandeep K. Gupta. BIST Test Pattern Generators for Stuck-Open and Delay Testing. to appear in European Design and Test Conf., 1994.

[11] R. Gupta. Advanced Serial Scan Design for Testability. Technical Report Tech. Report CRI-91-10, University of Southern California, March 1991.

[12] K. Kucukcakar and A.C. Parker. MABAL: A Software Package for Module And Bus ALlocation. *Int'l. J. of Computer Aided VLSI Design*, 2:419–426, 1990.

[13] S.P. Lin, C.A. Njinda, and M.A. Breuer. Generating A Family of Testable Design Using the BILBO Methodology. *Journal of Electronic Testing: Theory and Applications*, pages 71–89, April 1993.

[14] M. Abramovici, M.A. Breuer, and A.D. Friedman. *Digital Systems Testing and Testable Design*. W.H. Freeman and Company, 1990.

[15] L.T. Wang and E.J. McCluskey. Complete Feedback Shift Register Design for Built-In Self-Test. In *Proc. Int'l. Conf. on Computer-Aided Design*, pages 56–59, November 1986.

[16] R. Srinivasan, S.K. Gupta, and M.A. Breuer. Novel Test Pattern Generators for Pseudo-Exhaustive Testing. to appear in Proc. Int'l. Test Conf., 1993.

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

[18] 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, December 1981.

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