A Methodology for Partial Scan Design Using Balanced Sequential Structures*

by
Rajesh Gupta, Rajiv Gupta and
Melvin A. Breuer

Technical Report No. CRI 88-59

University of Southern California
Department of Electrical Engineering-Systems
Los Angeles, CA 90089-0781

* This work was supported in part by the Defense Advanced Research Projects Agency and monitored by the Office of Naval Research under Contract No. N00014-87-K-0861, and in part by the Semiconductor Research Corporation under Contract No. 88-DP-075.

A Methodology for Partial Scan Design Using Balanced Sequential Structures*

Rajesh Gupta, Rajiv Gupta and Melvin A. Breuer

University of Southern California, MC 0781
Department of Electrical Engineering–Systems
Los Angeles, CA 90089-0781.

Abstract

In the new partial scan methodology presented in this paper, the scan path is constructed such that the rest of the circuit belongs to a class of circuits called *balanced sequential structures*. Test patterns for this structure are generated by treating it as being combinational. To test the circuit, each test pattern is applied to the circuit by shifting it into the scan path, holding it in the scan path for a fixed number of clock cycles, loading the test result into the scan path and then shifting it out. This technique achieves full coverage of all detectable faults with a minimal number of scannable storage elements and using only combinational test pattern generation.

Key words: Balanced sequential structures, sequential circuit testing, design for testability, partial scan design, scan path register selection.

*This work was supported in part by the Defense Advanced Research Projects Agency and monitored by the Office of Naval Research under Contract No. N00014-87-K-0861, and in part by the Semiconductor Research Corporation under Contract No. 88-DP-075. The views and conclusions contained 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 Defense Advanced Research Projects Agency or the U. S. Government.

1 Introduction

Test pattern generation (TPG) for sequential circuits is usually a computationally intensive task [1]. Full scan design techniques attempt to reduce the complexity of this problem by connecting all storage elements into a scan path that is externally controllable and observable [2]. Thus in a circuit designed using full scan the portion of the circuit excluding the scan path, which we shall refer to as the **kernel** of the circuit, is fully combinational. Since all inputs and outputs of the kernel must be connected either to scan path elements or to circuit primary I/O, tests can be generated using combinational TPG techniques. The scan design concept is illustrated in Figure 1(a), and the kernel is indicated using broken lines. Scan design involves a logic overhead for modifying the storage elements and configuring them into a scan path. Partial scan techniques have been proposed for reducing this overhead [3]. The general concept of partial scan is illustrated in Figure 1(b), in which not all storage elements are included in the scan path. In this case the kernel is itself a sequential circuit.

In the partial scan technique proposed by Agrawal et al. [4], each test for the kernel consists of a single pattern shifted into the scan path and applied to the kernel for one clock cycle. The combinational portion of the kernel is used for TPG, and the storage elements inside it are considered to be inaccessible with unknown states. Complete fault coverage cannot be obtained using this technique; in fact, initial functional tests are required for it to be effective.

A recently proposed partial scan technique attempts to rectify this limitation by using multiple-pattern test sequences [5]. Given a fault to be detected, the technique uses sequential TPG combined with a judicious selection of scannable storage elements to bring the circuit to a desirable initial state from which the rest of the test sequence can be easily determined. Thus full coverage of kernel faults is obtained at the cost of high TPG complexity.

In this paper an alternative partial scan methodology, BALLAST (BALAnced structure Scan Test), is presented. In our approach each test consists of a single input pattern applied to a sequential kernel; however, the pattern is *held constant* in the scan path at the inputs of the kernel for a fixed number of clock cycles before the output pattern of the kernel is sampled and shifted out. The test patterns for the kernel are obtained by treating it as being combinational, with registers replaced by delayless wires. If the storage elements that are to be included in the scan path are selected appropriately, this technique achieves full coverage of all detectable faults in the kernel, and only combinational TPG is required.

An example of a circuit employing the BALLAST methodology is shown in Figure 2. The circuit has six registers and three combinational blocks. Registers R1, R2, R4 and R5 are constructed using D-flip-flops. The only registers in the scan path are R3 and R6, and they must have the ability to hold data across consecutive clock cycles. The set of test patterns for the kernel is obtained simply by replacing the registers within it by wires, and running combinational TPG on the resulting combinational circuit. In order to test the kernel, each test pattern is shifted into the scan path and held in the scan path for two clock cycles before the kernel output is loaded back into the scan path and shifted out. We will prove that using this approach we can detect all faults in the kernel that can cause errors in the logical operation of the circuit.

Now consider a variation on this example. Assume that the register R5 is also capable of holding data according to an externally controllable HOLD signal. In this case the simple test scheme described earlier does not work. The BALLAST methodology would require that *either one* of R1, R2, R4 and R5 be also removed from the kernel and included in the scan path along with R3 and R6. This statement, along with other claims about the fault coverage in the kernel, will be proved in a later section.

We shall describe the BALLAST methodology by focussing on two issues.

1. What circuit structures are acceptable as kernels, i.e., can be fully tested for all detectable faults using the test method described above?
2. Given an arbitrary sequential circuit with no scan path, how should a scan path be constructed such that the resulting kernel, which is that portion of the original circuit with the scan path removed, has the desired structure?

In Section 2 we shall formally define a class of sequential structures called *B-structures* that answer the first question above. Based on this definition, an outline of the BALLAST methodology will be presented in Section 3. The fact that B-structures get fully tested by the proposed test method will be proved in Section 4. Section 5 will answer the second question posed above by describing a procedure for constructing a scan path. Implementation aspects of the BALLAST methodology are discussed in Section 6. Section 7 deals with testing the functional modes of internal registers. The results of applying BALLAST to an actual circuit are described in Section 8. Finally, concluding remarks are presented in Section 9.

2 B-Structures and their Properties

In this paper we consider only synchronous sequential circuits in which every cyclic path contains at least one clocked storage element. Further, the storage elements are assumed to be edge-triggered flip-flops (FFs). The circuit may have any number of clocks. However, the FF clock signals must all be controlled by primary inputs, and no clock signal may feed the data input of any FF either directly or through combinational logic. The FFs may have *Reset* and *Load Enable* control signals, with similar restrictions as for the clock signals.

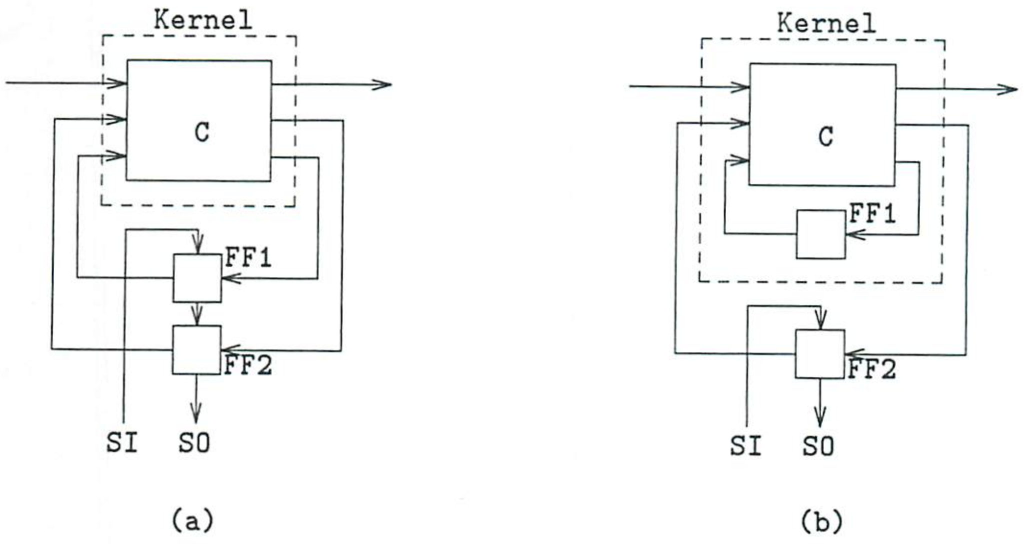


Figure 1: Scan design. (a) Full scan, (b) Partial scan.

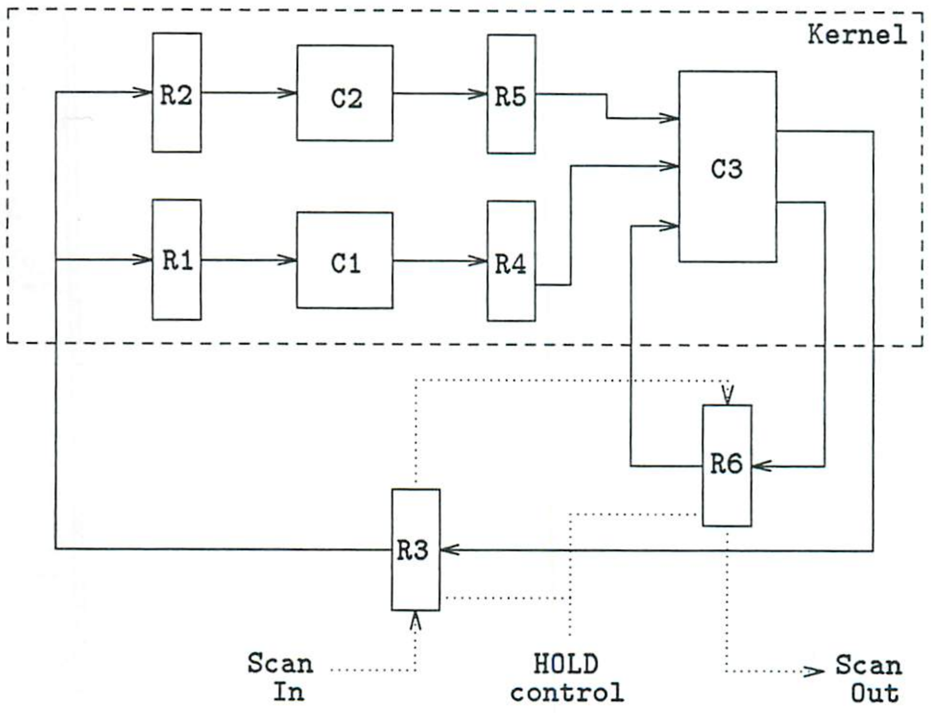


Figure 2: Illustration of BALLAST methodology.

Circuit Model

In general a synchronous sequential circuit S consists of blocks of combinational logic connected with each other either directly or through registers, as illustrated in Figure 3(a). A *register* is essentially a collection of one or more FFs. All FFs in a register must be driven by the same clock signal and must be controlled by the same mode control signals, if any exist. Any subset of the FFs in a register forms a valid register.

The set of registers in the circuit can be partitioned into two subsets based on the presence or absence of explicit *Load Enable* controls on the FFs comprising them. We define the **load set** L as the set of registers in the circuit whose FFs have no explicit load enable control; these registers always operate in the LOAD mode (in which data is read from the data input during every clock cycle). Similarly, we define the **hold set** H as the set of registers whose FFs have an explicit *Load Enable* control signal; these registers have two modes of operation: a HOLD mode (in which they retain their value across consecutive clock cycles) as well as a LOAD mode.

The combinational logic in S can be partitioned into maximal regions of connected combinational logic, known as **clouds** [6]. The inputs to these regions are either primary inputs or outputs of FFs; the outputs of these regions are either primary outputs or inputs to FFs. Clouds can be constructed by clustering combinational logic blocks in a bottom-up fashion based on the following rule: *Two combinational blocks A and B belong to the same cloud if either (1) one of them directly feeds the other, (2) they have a common input signal that fans out to both, or (3) one is fed by the output of a flip-flop (Q) and the other is fed by the inverted output of the same flip-flop ($\overline{Q}$).* The clouds of S_1 determined using this rule are indicated using dotted lines in Figure 3(a). Also shown is a special **vacuous cloud** which consists of wires with no logic. *A group of wires is considered to form a vacuous cloud if either (1) it connects the output of one register directly to the input of another, (2) it represents a circuit primary input feeding the input of a register, or (3) it represents the output of a register that is a circuit primary output.*

From the way in which clouds are defined, it follows that no two clouds can be directly connected together; they must be separated by one or more registers. Further, each FF in a register must receive data from exactly one cloud and must feed exactly one cloud. We can therefore constrain the grouping of FFs into registers (by splitting registers where necessary) such that each register receives data from exactly one cloud and feeds exactly one cloud. Under this constraint the topology of the circuit can be modeled by a directed **topology graph** $G = (V, A, H, w)$ in which nodes in V represent clouds, each arc in A represents a connection between two clouds through a register, arcs in $H \subset A$ represent HOLD registers, and $w : A \rightarrow Z^+$ (positive integers) defines the number of FFs in each register. $w(a)$ also represents the cost of converting the register a into a scan path register. Note that the topology graph differs from the classical concept of the directed graph in that it may have multiple arcs between the same pair of nodes. Figure 3(b) shows the topology graph G_1 of the circuit S_1.

B-Structures

Let S be an arbitrary synchronous sequential circuit with topology graph $G = (V, A, H, w)$.

Definition: S is said to be a **balanced sequential structure (B-structure)** 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 length[1]; and
3. $\forall h \in H$, if h is removed from G, the resulting graph is disconnected. □

Figure 4 shows an example of a topology graph that is a B-structure. Bold arcs represent HOLD registers and others represent LOAD registers.

Given a balanced sequential structure S^B, we define its **combinational equivalent,** C^B as the combinational circuit formed by replacing each FF in every register in S^B by a wire (if the output of the register uses the Q output of the FF) or an inverter (if its uses the $\overline{Q}$ output of the FF). Define the **depth**, d, of S^B as the longest directed path in its topology graph. Also, given an input vector I applied to S^B, define the **single-pattern output** of S^B for I as the steady-state output of S^B when I is held constant at the inputs to S^B and all its registers are operated in LOAD mode for at least d clock cycles. Further, given some fault f in S^B, if the single-pattern outputs for I of the good and the faulty circuits are different, then I is a **single-pattern test vector** for f.

B-structures have two interesting properties which allow them to be used as kernels in a BALLAST partial scan design: (1) they are single-pattern testable, and (2) a complete single-pattern test vector set can be derived using combinational test generation techniques. Both properties will be proved in Section 4. Next we present an overview of the proposed BALLAST methodology.

3 Scan Design Using B-Structures

When any register in a sequential circuit is included in a scan path, it serves as a control and observation point for the rest of the circuit. In effect it becomes a primary output of the cloud feeding it and as a primary input of the cloud it drives. Thus in our circuit model, the inclusion of a register in the circuit scan path corresponds to its removal from the topology graph of the circuit; and the reduced topology graph represents the kernel, i.e., the portion of the circuit to be tested using the scan path.

[1]Condition 2 actually includes condition 1.

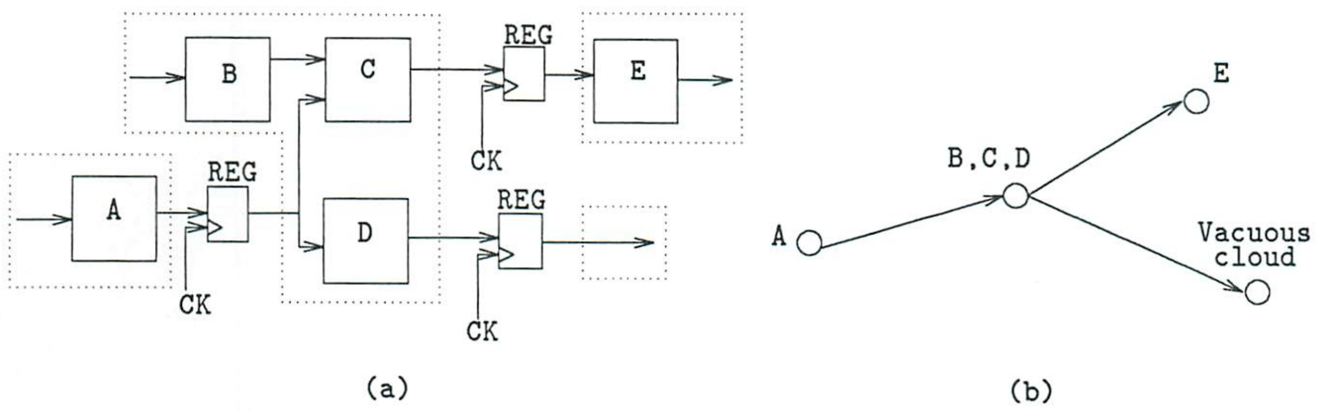


Figure 3: (a) Synchronous sequential circuit S_1; (b) Topology graph G_1 of S_1.

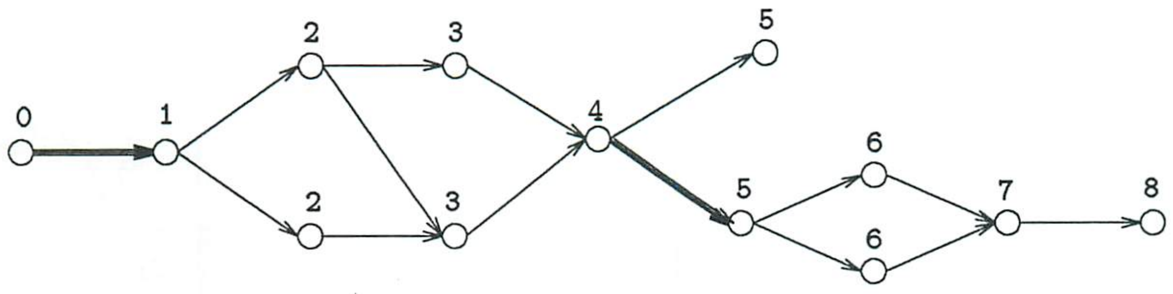


Figure 4: Example of balanced structure.

BALLAST selects a minimal number of FFs to be made scannable such that the remainder of the circuit is a B-structure. As will be proved later, this guarantees that full coverage of all detectable faults in the kernel can be obtained using single-pattern test vectors shifted in through the scan path. The following is an outline of the BALLAST methodology.

1. Construct G, the topology graph of the circuit, as defined in Section 2.
2. Select a minimal cost set of arcs, R, to be removed from G such that the remaining topology graph is balanced. Arcs in R represent the registers that must be included in the scan path. Let S^B be the B-structure corresponding to the resulting topology graph which represents the kernel of the circuit. Algorithms for implementing this step are presented in Section 5.
3. Determine the combinational equivalent C^B of the kernel S^B. Using traditional combinational test pattern generation techniques, determine a complete test vector set for C^B. This will constitute a complete single-pattern test vector set for the kernel S^B. (See Section 4 for proof of correctness of this statement.)
4. Construct a scan path containing the registers in R so that they are capable of (a) shifting test patterns in/out, (b) holding a pattern constant at the kernel inputs for d clock cycles (where d is the depth of the B-structure comprising the kernel), and (c) loading in the test results from the kernel. The circuit modifications will be discussed further in Section 6.

Given a circuit designed in the above manner, the test plan for applying a sequence of single-pattern test vectors to the circuit is as follows. N is the number of test vectors to be applied and l is the total number of FFs in the scan path.

1. *Operate all scan registers in the* SHIFT *mode for l clock cycles. (Scan in the first test vector.)*
2. *Repeat N times:*
   (a) *Place all scan registers in the* HOLD *mode and all non-scan registers in the* LOAD *mode for d clock cycles. (This allows test data to propagate through the circuit.)*
   (b) *Operate all scan registers in the* LOAD *mode for 1 clock cycle. (This loads the test result into the scan registers.)*
   (c) *Operate all scan registers in the* SHIFT *mode for l clock cycles. (Scan out the test result and scan in the next test vector.)* □

In general, the total test time for a BALLAST circuit may be higher than that for the corresponding full scan circuit. This may be attributed to two factors: first, the higher number of test vectors required for full fault coverage of the kernel, caused by the size of the combinational equivalent circuit as well as fewer controllable inputs and observable outputs of the kernel; and second, the need to hold each test pattern for d clock cycles in step 2a above.

In the following section the testability properties of B-structures are studied with the objective of proving that any B-structure is a valid kernel in the BALLAST test methodology.

4 Proof of Correctness

We shall now focus on the kernel of the circuit under consideration in isolation from the scan path. We shall treat all connections of the kernel from/to scan path registers as primary I/O lines of the kernel. The BALLAST methodology ensures that the kernel is a B-structure. In this section two important properties of B-structures will be proved: that they are single-pattern testable, and that complete single-pattern test sequences can be determined using combinational test generation techniques.

In the following discussions, S^B represents an arbitrary B-structure and C^B represents its combinational equivalent. The set of faults of a B-structure refers to the union of the sets of faults of the individual clouds. Stuck-at faults in a register can be considered to be equivalent to a stuck-at fault in one of the clouds adjacent to it. A *detectable* fault is one for which some sequence of test patterns exists.

Lemma 1 *For any input vector I, the output of C^B and the single-pattern output of S^B are identical.*

Proof: By induction on the depth of S^B.

Let G be the topology graph of S^B and let its depth be d.

$\underline{d = 0:}$ If d is 0, S^B is combinational, and the statement is trivially true.

$\underline{d > 0:}$ Assume that the statement is true for $0 < d \leq n-1$, and consider $d = n$. Remove from G all "first-level" nodes, i.e., nodes that have no incoming arcs, and all corresponding first-level arcs. There must be at least one first-level node because G is acyclic. Let the resulting graph be G_1, representing a corresponding balanced structure S_1^B of depth $n-1$ with combinational equivalent C_1^B. When the input vector I is applied to S^B, all clouds corresponding to first-level nodes must settle at constant values within the first clock cycle; hence the values loaded in by the first-level registers must in fact be their final steady-state values. Thus after the first clock cycle, S_1^B receives a constant input pattern. Since the depth of S_1^B is $n-1$, and both C_1^B and S_1^B receive the same input pattern after the first clock cycle, therefore the output of C_1^B and the single-pattern output of S_1^B must be identical, by assumption. Thus the statement of the lemma is true for $d = n$.

Thus the lemma holds for all $d \geq 0$. $\square$

Note that the only property of B-structures used in the above lemma is the fact that the topology graph is acyclic, hence it is true of all acyclic structures.

Lemma 2 *Let f_S be a fault in S^B and let f_C be the corresponding fault in C^B. Then any test pattern t for f_C in C^B is a single-pattern test for f_S in S^B.*

Proof: Let C_f^B and S_f^B be the faulty circuits produced by f_C and f_S respectively. Since t detects f_C, the outputs of C^B and C_f^B must differ for input t. Hence due to Lemma 1, the single-pattern outputs of S^B and S_f^B must differ for input t. Thus t is a single-pattern test for f_S in S^B. $\square$

Note that the above lemma does not prove that there is a single-pattern test for *every* detectable fault in S^B.

Theorem 1 *Every B-structure is fully testable for all* detectable *faults using single-pattern test vectors.*

Proof: It is sufficient to prove that given any detectable fault f in the balanced structure S^B, there exists a single-pattern test vector for f. We first present a formal proof and then illustrate the proof using an example.

Let $G = (V, A, H, w)$ be the topology graph of S^B. During any clock cycle t, let the *state* of S^B be defined by the tuple $G^t = (G, I^t, h^t, x^t)$, where $I^t(v), v \in V$ represents the input pattern applied at the circuit primary inputs (if any) of the cloud v during clock cycle t; $h^t(a), a \in H$ represents the mode signal of the HOLD register a during clock cycle t; and $x^t : A \rightarrow \{0, 1, D, \overline{D}, \times\}$ is an assignment of logic values to all registers in the circuit. In this 5-valued logic [1], D and $\overline{D}$ represent erroneous states due to a fault, and $\times$ represents an unspecified or *don't-care* value. Assume (without loss of generality) that for a HOLD register a, $h^t(a) = 0$ represents the LOAD mode and $h^t(a) = 1$ represents the HOLD mode.

Since f is detectable, some test sequence T for f must exist. Let T consist of a sequence of patterns applied to the circuit primary inputs feeding the clouds and to the control signals feeding the HOLD registers. Let the first pattern be applied at clock cycle

1 and let the fault be first observed at a circuit primary output at clock cycle m. The application of T to S^B can be fully described by the sequence of states $(G^1, G^2, \ldots, G^m)$ that the circuit experiences.

We shall now show that based on T it is possible to derive a single-pattern test for f. The following procedure transforms T into a single-pattern test T'.

```
procedure transform ($T = (G^1, \ldots G^m)$):

  1. Pick a node $v_0 \in V$ such that the corresponding cloud in $S^B$ has one or more outputs
     that are circuit primary outputs and at least one primary output has value $D$ or $\overline{D}$
     in state $G^m$.

  2. Define a relation called activation time, $\alpha : V \rightarrow \{1, 2, \ldots, m\}$ where $\alpha(v)$ represents
     the clock cycle during which the cloud $v$ is is actively involved in sensitizing or
     propagating the effect of the fault $f$; i.e., there is a functional dependency between
     the output of cloud $v$ during clock cycle $\alpha(v)$ and the output of $v_0$ during clock cycle
     $m$.
     (It will be shown that every cloud has a unique activation time.)
     Set $\alpha(v_0) \leftarrow m$.

  3. Construct a set $\mathcal{F}$, representing a frontier of nodes being processed, and set $\mathcal{F} \leftarrow \{v_0\}$.

  4. Repeat the following until $\mathcal{F} = \phi$:

     (a) Pick some $v \in \mathcal{F}$ having the highest activation time $\alpha(v)$, and remove it from
         $\mathcal{F}$.
         $k \leftarrow \alpha(v)$.
     (b) For all arcs $a$ incident onto $v$, do the following:
           i. $u \leftarrow$ source node of $a$;
              Add $u$ to $\mathcal{F}$, with $\alpha(u) \leftarrow k - 1$.
          ii. If $a \in H$ and $h^k(a) = 1$ then
                A. $t \leftarrow$ min. $j$ such that $h^{j+1}(a) = h^{j+2}(a) = \ldots = h^k(a) = 1$;
                   If there is no such $j$, skip steps B and C.
                   $t$ represents the clock cycle during which cloud $u$ is active, and register
                   $a$ loads the active data, in the original test plan.
                B. $\tau \leftarrow k - t =$ number of HOLD cycles of $a$ in the current sequence.
                   The next step eliminates the HOLD cycles while keeping the test valid.
                C. The removal of $a$ from $G$ must disconnect $G$ into separate components,
                   by definition of balanced structures. Let $G_u$ be the component that
                   contains $u$. Modify the test by delaying all electrical activity within $G_u$
                   by $\tau$ clock cycles as follows.
```

$\forall v \in V$ in G_u, $I^j(v) \leftarrow I^{j-\tau}(v)$, $\tau < j \leq k$;
$\forall h \in H$ in G_u, $h^j(h) \leftarrow h^{j-\tau}(h)$, $\tau < j \leq k$;
$\forall a \in A$ in G_u, $x^j(a) \leftarrow a^{j-\tau}(a)$, $\tau < j \leq k$;
$h^k(a) \leftarrow 0$.
It can easily be seen that the sequence of modified states $(G^1, \ldots, G^m)$ is still a valid test for f.

The preceding portion of the procedure transforms the test sequence such that in the final state sequence every register is in the LOAD mode at the time when the cloud driving it becomes active, and adjacent clouds are active during adjacent clock cycles. Note that the activation time $\alpha(v)$ of each cloud v depends only on its distance from the output node v_0. Since G is balanced, this implies that *for each cloud, there is a unique clock cycle during which it is active.*

5. Let $t_0 \leftarrow \min_v [\alpha(v)]$ = earliest activation time among clouds that have been assigned an activation time.
Then $T' = (G^{t_0}, \ldots, G^m)$ is a valid test for f.

6. Any circuit primary input feeding a cloud v must have the value determined by $I^{\alpha(v)}(v)$ during the activation time $\alpha(v)$ of v; at other times its values do not affect the test. Hence we can transform the input patterns as follows to obtain a valid test:
$\forall v \in V$, $I^t(v) \leftarrow I^{\alpha(v)}(v)$, $t_0 \leq t \leq m$.
In other words, the input pattern consisting of the pattern $I^{\alpha(v)}(v)$ applied to each cloud v is a single-pattern test for f in S^B.

7. Return $I^{t_0}(v)$, $\forall v \in V$; this represents the single-pattern test vector T'.

The above procedure demonstrates that every fault in S^B has a single-pattern test; this proves the theorem. □

Example

Figure 5(a) shows a typical B-structure with three clouds and three registers. For simplicity in this example, each register consists of a single FF. R3 is a HOLD register while R1 and R2 are LOAD registers. The lines a, b, c, d are circuit primary inputs to the various clouds and g is a circuit primary output. h controls the HOLD mode of R3 such that if $h = 0$ during clock cycle t, then R3 loads new data between clock cycles t and $t+1$, and if $h = 1$, it holds the data present during clock cycle t for an additional cycle.

Consider the fault f which makes the line e stuck at 1. f is detectable, and Figure 5(a) shows a test sequence T, consisting of three test patterns, that detects f. The patterns shown at the primary inputs and at h must be applied in order from left to right over consecutive clock cycles. The states of the internal signals during this time are also shown. Note that the fault is first detected at primary output g in clock cycle 3.

We shall now show that based on T it is possible to derive a single-pattern test for f. We first transform the test so that all HOLD registers (i.e., R3) operate only in the LOAD mode, and then further transform it so that the value applied to each primary input is constant throughout the test.

Given the test sequence T, we say that a cloud C is *active* during a clock cycle k if it is actively involved in sensitizing or propagating the effect of the fault during clock cycle k. In other words, there is a functional dependency between the output of cloud C3, at whose output the fault is first detected at clock cycle 3, and the output of C at clock cycle k. The cycles during which the various clouds in Figure 5 are active are indicated by underlining the corresponding logic values in the state sequences. Clearly, C3 is active at clock cycle 3. C1 must be active at clock cycle 2, since it feeds C3 through the LOAD register R1 which has a constant delay of 1 clock cycle. Note that every path between C1 and C3 must have exactly the same number of LOAD registers, because this structure is balanced; hence C1 must be active at clock cycle 2 and at no other time. The other cloud feeding C3 is C2, and in this case the connection is via a HOLD register, which introduces a variable delay. The control sequence applied to R3 in test sequence T is '01×', which means that R3 is actually in the HOLD mode at the time (clock cycle 2) just before C3 becomes active. The most recent clock cycle at which new data was loaded into R3 is 1; hence the erroneous output of C3 actually depends on the output of C2 in clock cycle 1. Thus C2 is active in clock cycle 1.

Figure 5(b) shows a transformed test in which C2 is active in clock cycle 2 instead of 1, and the HOLD operation of R3 is eliminated. In effect, all logical activity in C2 and in all clouds feeding C2 (directly or indirectly) is delayed by one clock cycle, so that the HOLD cycle of R3 can be eliminated. Note that by delaying the activity in the portion of the circuit feeding C2, the activity in the rest of the circuit is unaffected. This follows from the fact that all paths between this portion of the circuit and the rest of the circuit must pass through the HOLD register R3. Thus the sequence of states shown in Figure 5(b) is a valid test for f in which the error is first observed at clock cycle 3.

The transformation above ensures that every cloud feeding C3 (which is active at clock cycle 3) is active at clock cycle 2. The transformation process must be repeated for all clouds feeding C1 and C2, respectively (none in this case), until (i) the clock cycles during which the various clouds are active have been determined, and (ii) every HOLD register operates in the LOAD mode in the transformed test. From the preceding arguments it follows that every cloud is active during some unique clock cycle; further, no cloud can be active earlier than clock cycle 2, since adjacent clouds are active during consecutive clock cycles and C3 is active at clock cycle 3 and the depth of the B-structure is 1. Hence the test can be further transformed such that (a) it consists of only two vectors, applied at clock cycles 2 and 3; and (b) the input pattern required at the primary inputs of each cloud

C during its *active* cycle are actually applied during *both* cycles 2 and 3. The resulting single-pattern test vector T' for f is $a = 1$, $b = 0$, $c = 0$ and $d = 1$.

The above example illustrates the procedure for transforming an arbitrary test into a single-pattern test, and demonstrates that every detectable fault is single-pattern testable.

Corollary to Theorem 1 *The maximum required duration of any single-pattern test is d clock cycles, where d is the depth of the B-structure.*

Proof: This follows from the fact that adjacent clouds are active during adjacent clock cycles in the transformed test. □

We now proceed to show that a complete single-pattern test set for a B-structure can be obtained by combinational test pattern generation techniques.

Lemma 3 *Let f_S be a fault in S^B and let f_C be the corresponding fault in C^B. If f_S is detectable in S^B then f_C is detectable in C^B.*

Proof: Let C^B_f and S^B_f be the faulty circuits. Since f_S is detectable in S^B, by Theorem 1 there must be a single-pattern test vector t for this fault. Hence S^B and S^B_f must have different single-pattern outputs for t. This implies, by Lemma 1, that C^B and C^B_f must have different outputs for t. Thus t is a test for f_C in C^B. □

Theorem 2 *Given a balanced structure S^B, any complete test set for all detectable faults in its combinational equivalent C^B is a complete single-pattern test set for all detectable faults in S^B.*

Proof: Let T be a test set for all detectable faults in C^B. We need to show that T is a single-pattern test set for all detectable faults in S^B.

Let f_S be a detectable fault in S^B and f_C the corresponding fault in C^B. Since f_S is detectable, by Lemma 3 f_C must also be detectable. Since T detects all detectable faults in C^B, it must contain some vector t that detects f_C. Hence, by Lemma 2, t must detect f_S. Thus T detects all detectable faults in S^B. □

The two theorems reduce the problem of test generation for B-structures to the simpler problem of combinational logic test pattern generation, and from which a complete single-pattern test set can be obtained. The single-pattern nature of the test sequences makes it possible, in a partial scan circuit, to easily test a kernel that is a B-structure. Each single-pattern vector can be shifted into the scan path and held constant for the required number of clock cycles before the test results are loaded into the scan path and shifted out. The implementation issues related to the scan path will be discussed briefly in Section 6.

5 Algorithm for Scan Register Selection

Given an arbitrary sequential circuit to which partial scan is to be applied, BALLAST selects a set of registers to be made scannable such that the kernel is a B-structure and the cost of modifying the circuit is minimized. The selection is carried out using the algorithms presented in this section.

Let $G = (V, A, H, w)$ be the topology graph of the circuit. Formally, we need to determine a set of registers $R \subseteq A$ such that the topology graph of the kernel, $(V, A - R, H - R, w)$, is balanced and $\sum_{a \in R} w(a)$ is minimized. The solution can be obtained using the following steps.

1. Transform $G = (V, A, H, w)$ into an acyclic topology graph G_A by removing a set of arcs R_A such that $\sum_{a \in R_A} w(a)$ is minimized.
2. Transform G_A into a balanced topology graph G_B by removing a set of arcs R_B such that $\sum_{a \in R_B} w(a)$ is minimized.
3. $R = R_A \cup R_B$ is the desired set of arcs, and the resulting topology graph $G_B = (V, A - R, H - R, w)$ represents the kernel.

The first problem is known to be NP-complete [7]. An algorithm for solving this problem has been presented in [8]. This algorithm first determines the set of all cycles and then finds a minimal set of arcs that break all cycles. The reader is referred to [8] for further details.

A simplified form of the second problem has also been shown to be computationally intractable [9]. In this section we present a heuristic procedure, **balance**, for solving this problem. **balance** uses a verification procedure, **check**, to verify that a given structure is balanced.

Verification Procedure

Given the topology graph G of a sequential structure S, the following procedure checks whether S is balanced. First the procedure checks whether the removal of each arc in the hold set would disconnect the topology graph. Then, starting at each root node in turn, it levelizes the portion of the graph reachable from the current root node. If every node is found to be at a unique distance from each root, the graph is balanced.

function check $(G = (V, A, H, w))$: Returns SUCCESS if G is balanced, FAILURE otherwise.

1. Construct the graph $G' = (V, A - H, \phi, w)$, by removing all HOLD registers from G. Determine the connected components $\{C_1, C_2, \ldots, C_K\}$ of G'.
2. Collapse each connected component C_i of G' into a single node in G, the original topology graph. Let the collapsed topology graph be G''. Note that the arc set of G'' is H.
3. Determine whether G'' is a tree. If it is, condition 3 of the definition of B-structures is satisfied; proceed to the next step. Otherwise return FAILURE.
4. Repeat for each C_i:
   (a) Determine the set $ROOTS$ of root nodes of C_i, where *root nodes* are defined as nodes with no incoming arcs. If there are one or more root nodes, proceed to the next step; otherwise return FAILURE since condition 1 in the definition must be violated.
   (b) Pick a root node v_1 in $ROOTS$. Starting at v_1, carry out a breadth-first traversal of all nodes in C_i reachable from v_1 by a directed path, and assign each node visited a *level number* equal to its distance from v_1. If at any time a node v_2 needs to be assigned a level number when it has been previously assigned a different level number with respect to v_1, stop the search and return FAILURE, since C_i must violate either condition 1 or condition 2. Continue the traversal until no more nodes can be visited.
   (c) Repeat step 4b for all root nodes in $ROOTS$.
5. S is a B-structure; return SUCCESS. □

Figure 4 indicates the level number assigned by the procedure to each node. In the above procedure, the time complexity of step 4b is $O(m)$, where $m = |A|$. This is repeated for all root nodes of all components C_i, hence an upper bound on the complexity of step 4 and of this procedure is $O(nm)$, where $n = |V|$.

Balancing Arbitrary Sequential Circuits

An acyclic topology graph $G = (V, A, H, w)$ is balanced if and only if all paths between any given pair of nodes are of equal length and the removal of any arc in H disconnects G. A heuristic procedure, **balance**, for balancing G is presented below. It returns a set of arcs $R \subseteq A$ such that the derived topology graph $(V, A - R, H - R, w)$ is balanced.

The procedure works in a recursive manner by partitioning the topology graph into two smaller topology graphs, balancing them independently, and then merging the solutions. The partition is obtained by determining a minimum cost cutset (mincut) CS of the topology graph. The idea behind using a mincut is to minimize the sensitivity of the overall solution to the degree of optimality of the merging process (which is based on a greedy heuristic). The mincut is determined by applying the maximum flow algorithm [10] to the topology graph, with capacity of arc $a \in A$ defined by $w(a)$.

A greedy algorithm is used for merging the two balanced sub-structures. First we consider only registers without HOLD modes. Arcs in the mincut that represent such registers are introduced one by one into the merged topology graph in order of decreasing weight w. Arcs which cause an imbalance in the merged graph (verified using **check**) are rejected, implying that the corresponding registers must be made scannable. The "accepted" arcs represent a maximal set of LOAD registers that can join the two balanced sub-structures while keeping the merged topology graph balanced.

From the definition of B-structures it follows that if any HOLD arc is used to connect the two balanced sub-structures, no other arc should connect them. Thus an alternative to using the set of LOAD arcs derived above is to use the maximum-weight HOLD arc in the mincut.

Hence there are two ways of merging the two balanced subgraphs: one using only LOAD arcs and the other using a single HOLD arc. The costs of these two solutions are compared to determine the set of arcs to be actually used for merging.

function balance ($G = (V, A, H, w)$: acyclic topology graph): Returns $R \subseteq A$ such that $(V, A - R, H - R, w)$ is balanced.

1. If (**check**(G) = SUCCESS) then return $(R \leftarrow \phi)$, else proceed.
2. CS = minimal cost cutset of G. Let G_s and G_d be the subgraphs of G induced by CS.
3. Balance G_s and G_d separately; $R \leftarrow$ **balance**$(G_s) \cup$ **balance**$(G_d) \cup CS$.
4. Let $CS_H \leftarrow CS \cap H$, $CS_L \leftarrow CS \cap (A - H)$ be a partition of CS into its HOLD and LOAD registers.
   Sort the arcs in CS_L in order of decreasing cost.
5. $CS'_L \leftarrow \phi$, the set of LOAD arcs retained in the topology graph when merging G_s and G_d.
6. For all arcs a in CS_L, in order of decreasing cost:

```
    Check whether the inclusion of $a$ makes the merged graph unbalanced:
    If [check($V$, $(A - R) \cup CS'_L \cup \{a\}$, $H - CS$, $w$) = SUCCESS]
        then $CS'_L \leftarrow CS'_L \cup \{a\}$.

7. Let $a_H \leftarrow$ highest-cost arc in $CS_H$.

8. Compare the solutions obtained in the previous two steps. In case of a tie, make the
   HOLD register scannable.
    If $\sum_{a_L \in CS'_L} w(a_L) > w(a_H)$
        then $R \leftarrow R - CS'_L$
        else $R \leftarrow R - \{a_H\}$.

9. Return $R$, the set of arcs to be removed from $G$ to make it balanced.    □
```

The time complexity of the above algorithm is bounded by $O(nm^3)$, where $n = |V|$ and $m = |A|$. This follows from the following observations: (i) in the worst case, the minimum cutset may need to be computed $O(m)$ times as the procedure calls itself recursively; (ii) the size of each minimum cutset is bounded by m; and (iii) for each arc in the cutset, the procedure **check** is invoked once with complexity $O(nm)$. Note that this is a very loose bound and the average performance is expected to be much better.

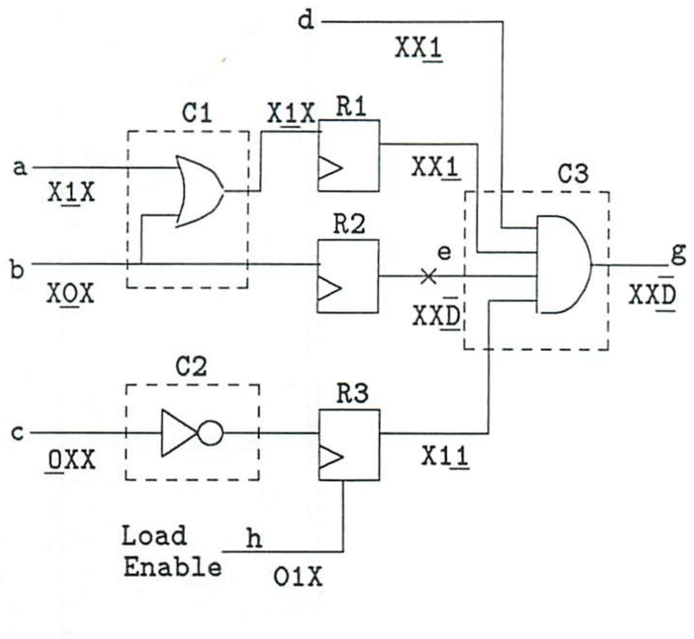
Figure 6(a) shows an arbitrary topology graph that is not balanced, and Figure 6(b) shows the result of applying **balance** to it. The arcs removed are (e, f) and (g, i), and the registers in the circuit that correspond to these arcs must be included in the scan path.

6 BALLAST Implementation

In Section 5 we described a procedure for determining a minimal set of registers to be connected into a scan path. The resulting kernel is a B-structure, and therefore single-pattern testable. Hence it can be tested by scanning in appropriate test vectors and holding them constant for *d* clock cycles, where *d* is the depth of the B-structure. The exact test plan is described in Section 3. The operations in steps 1, 2b and 2c of the test plan are the same as in traditional scan path techniques [2]. The operation in step 2a, however, is peculiar to BALLAST and has several implications.

First, some scan registers which do not already have a HOLD mode must be provided with this mode. Depending on the implementation, this may introduce a slightly higher logic overhead for some scan registers. However, typically this overhead is likely to be acceptable as a reduced number of FFs need to be made scannable.

Up to this point we have maintained in the interest of simplicity that a test pattern must be held at all the inputs to the kernel for *d* clock cycles. This is not strictly true; for example the scan path register R3 in Figure 2 does not actually need to hold its pattern



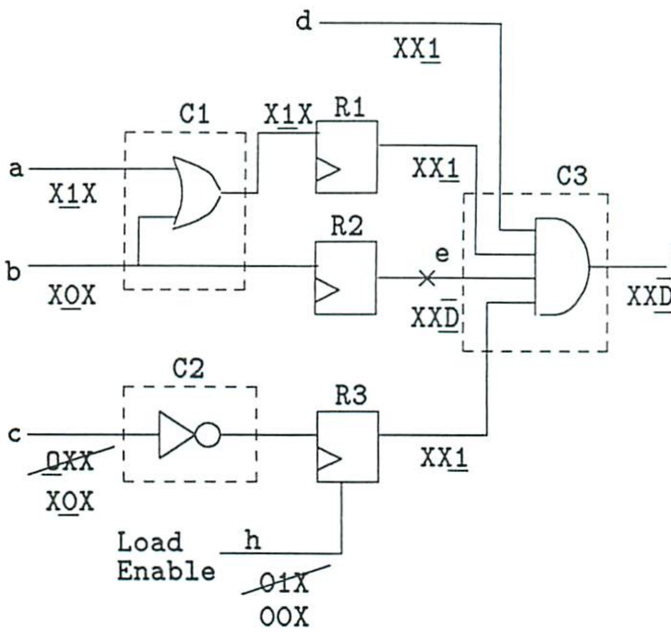
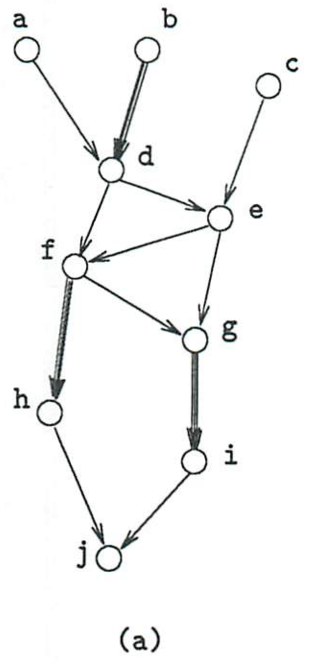


Figure 5: (a) Test for e stuck-at-1, (b) Transformed test with no HOLD operations.



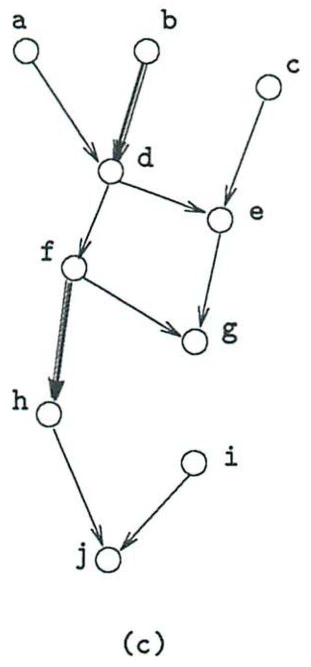


Figure 6: Illustration of balance procedure. (a) Original topology graph, (b) Balanced topology graph.

during the test. The reader can easily verify that for a given test vector applied using the scan path, the kernel outputs after the 2nd clock cycle are independent of whether or not R3 holds its pattern after the first cycle. In general the registers that need to be provided with the HOLD mode can be identified using the following rule:

Let $G = (V, A, H, w)$ be the topology graph of the kernel with depth d. Let V_{IN} (V_{OUT}) be the set of clouds in V whose inputs (outputs) are connected to scan registers or to circuit primary I/O. *If for cloud v_i in V_{IN}, any path from v_i to any cloud v_o in V_{OUT} is of length less than (but not equal to) d, then all scan registers feeding v_i must have* HOLD *modes.*

A second implication of the BALLAST test procedure is that explicit control over HOLD control signals, including those present in the original circuit, is necessary. These controls must be externally accessible through circuit primary inputs. This requirement is met by most circuits that are designed to have separate data path and control units. In circuits whose control signals are derived from within the data path logic, additional logic and I/O pins are required to make these signals externally controllable. In BALLAST, while a test pattern is being applied to the kernel, registers within the kernel continously load their input data at each clock cycle. Therefore, it is essential that no signal simultaneously control the HOLD modes of both a scan register and a non-scan register.

Circuit Modifications

The registers in the original circuit may be classified into four types depending on (1) whether or not they have a HOLD capability, and (2) whether or not they are to be included in the scan path. The modifications required for each type of register so that they can perform the appropriate functions according to the test plan are listed below.

Non-Scan Registers Without HOLD Mode: These require no modification.

Non-Scan Registers With HOLD Mode: The individual HOLD controls of all such registers should be externally controllable so that they can be operated in LOAD mode while the kernel is being tested.

Scan Registers With HOLD Mode: These registers need to be converted into scan path registers by the addition of SHIFT modes as in traditional scan design techniques [2]. Their HOLD mode signals must be externally controllable so that they can be made to hold test patterns for the required number of clock cycles. No HOLD control signal for a scan register may serve as a control signal for a non-scan register.

Scan Registers Without Hold Mode: In addition to being augmented with SHIFT modes, some of these registers need to be provided with HOLD modes as well. The HOLD control signals for all such registers may be controlled by a single external pin.

These modifications may result in a small logic overhead and possibly additional I/O pins for ensuring that the HOLD signals are appropriately configured.

7 Testing Register Functional Modes

As we have mentioned previously, a complete set of single-pattern test vectors can be obtained for covering all detectable faults in the combinational logic. While exercising the combinational logic, every FF must operate in the LOAD mode; hence other built-in modes of operation of the FFs may not get exercised. In this section we deal with the problem of testing the RESET (or CLEAR), PRESET and HOLD modes of operation, if present in any FFs in the circuit. Further, we focus our attention on faults in non-scan FFs, i.e., FFs within the kernel. FFs in the scan path are either tested for free during tests for the kernel or can be easily tested using special patterns shifted in and out of the scan path.

The three functional modes listed above give rise to six possible fault modes: stuck-at-RESET, cannot-RESET, stuck-at-PRESET, cannot-PRESET, stuck-at-HOLD and cannot-HOLD. We shall map faults in the functional operation of FFs on to structural faults on the control lines for the FFs. Given the B-structure S^B under test, we generate the combinational equivalent C^B using the functional combinational equivalent of each FF, depending on its built-in functions, rather than simple wires and/or inverters.

Figure 7(a) shows a FF connecting two clouds C_1 and C_2, and Figures 7(b) and (c) show how the combinational equivalents of FFs having RESET and PRESET modes, respectively are used. Each fault in these functional modes is functionally equivalent to a stuck-at-0/1 fault on the corresponding control line. Thus it is sufficient to detect both stuck-at-0 and stuck-at-1 faults on the control lines using the combinational equivalent. Note that some of the faults may be tested for free while testing the clouds of the kernel. The control lines may be treated as primary inputs for the purpose of test pattern generation. If a control line fans out to more than one FF, as shown in Figure 7, faults on all the lines marked $\times$ (i.e., the fanout stem as well as the fanout branches) must be tested. This ensures that all appropriate fault modes are covered.

Figure 7(d) shows the combinational equivalent of a HOLD FF. Unlike the faults considered earlier, the manifestation of HOLD faults depends on the previous state of the circuit. Two single-pattern test vectors are required to detect the cannot-HOLD fault. While the first vector is applied all FFs operate in the LOAD mode, and while the second vector is applied the FF under test (and possibly other FFs) operate in the HOLD mode. Two copies of the combinational equivalent of the B-structure, viz. C_1^B and C_2^B, are required for generating a test. This is illustrated in Figure 8. Note that multiplexers are not required to model the HOLD FFs in C_1^B since all FFs operate in the LOAD mode while the first vector is applied. Test pattern generation on the combined combinational structure yields two patterns t_1 and t_2 corresponding to C_1^B and C_2^B, respectively. During test, first t_1 is scanned in and applied to the kernel in the normal way with all FFs in the LOAD mode. After the kernel outputs have stabilized the control signal under test is switched to the HOLD mode. t_2 is now scanned in and applied to the kernel in the normal way except for the control signal under test being in the HOLD mode. The output of the kernel for the second pattern is used to detect the fault.

8 Case Study

The BALLAST technique was applied to a sequential module MCOMP, 32 copies of which were used in a larger circuit VC. Each copy of MCOMP contained 14 FFs; hence with full scan design, MCOMP would contribute a total of 448 FFs to the scan path of VC. By using the BALLAST approach, the MCOMP circuit was reduced to a B-structure of depth 4 by converting only 8 FFs into scan path FFs; thus the total contribution of MCOMP to the scan path of VC was reduced to 256 FFs. In order to satisfy the restrictions on FF mode control the I/O pin overhead increased from 3 pins to 4 pins.

The number of test patterns for MCOMP increased from 13 (for the full scan version) to 21 (for the BALLAST version). This would imply an increased test time (from 208 clock cycles to 280 clock cycles) if MCOMP were to be tested in isolation. However, the scan FFs connected to MCOMP were a part of the scan path for the whole chip VC. Since the length of the circuit scan path was reduced by 192 FFs by using BALLAST for the 32 MCOMP units, this implied a large saving in shifting time for each vector applied to the chip as a whole, with a favorable impact on the overall test time.

Comparable results were obtained on other case studies.

9 Conclusion

In this paper we have described a class of synchronous sequential circuits called balanced structures. They have the following properties which make them useful as kernels in a partial scan circuit: (i) they are single-pattern testable for all detectable faults in the combinational logic and nearly all faults in the storage elements; (ii) the FFs internal to these networks need not be made scannable; and (iii) they can be treated as combinational circuits for the purpose of test pattern generation.

The concept of balanced structures can be used to reduce various overheads in partial scan design for arbitrary sequential circuits. By identifying the balanced sub-structure of

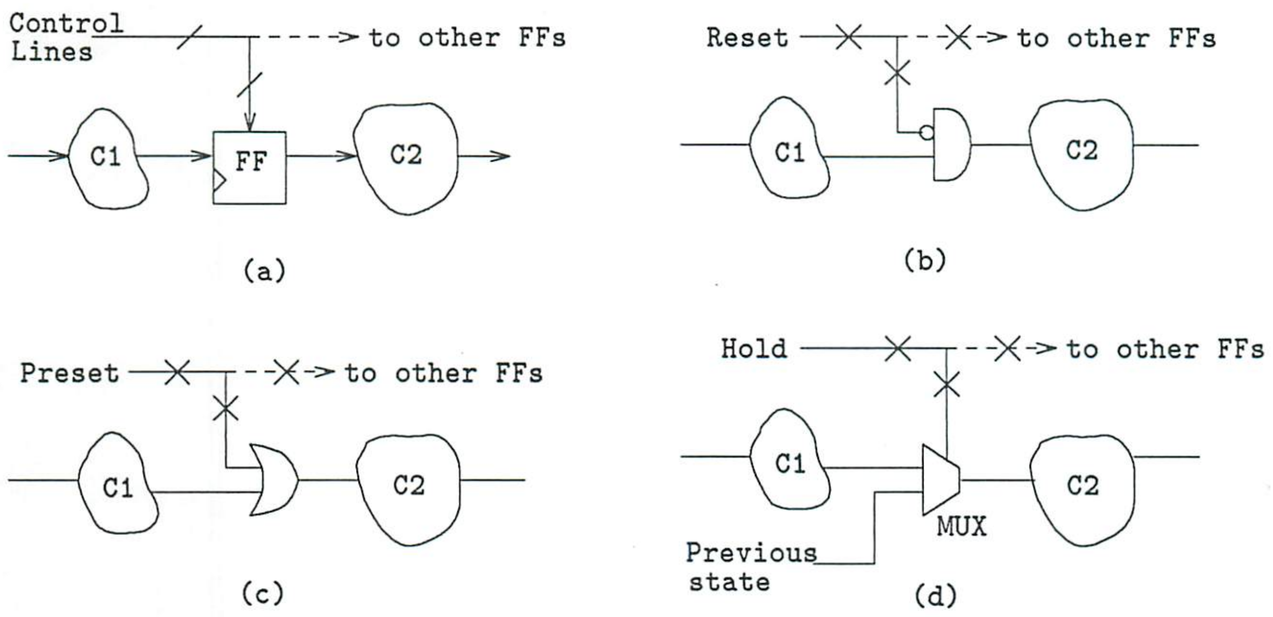


Figure 7: Modeling FF functional modes. (a) FF connecting two clouds, (b) Model of RESET operation, (c) Model of PRESET operation, (d) Model of HOLD operation.

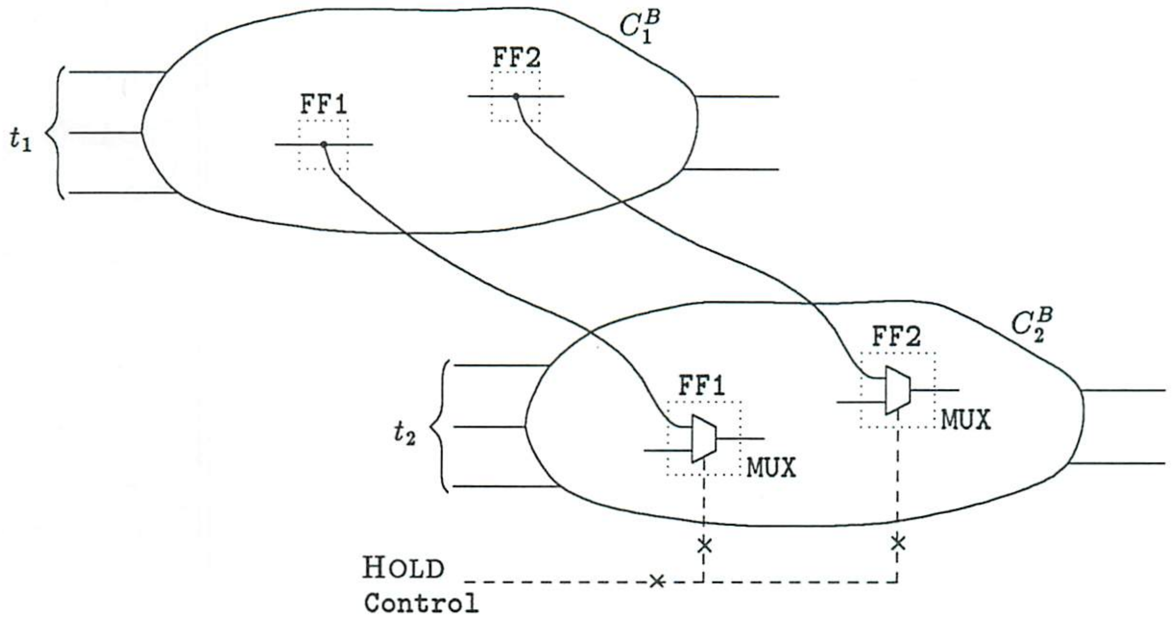


Figure 8: Combinational equivalent for detecting HOLD faults.

the circuit that has the largest number of FFs, it is possible to minimize the test overhead by configuring the balanced sub-structure as a kernel to be tested using a scan path without any loss in fault coverage. Heuristic procedures for determining the minimal set of scan FFs were presented. Based on these procedures the shortest required scan path can be constructed. FFs in the scan path need to have a HOLD mode so that single-pattern test vectors can be applied to the kernel. Case studies indicate that the logic overhead can be reduced significantly using this partial scan technique, particularly in pipelined circuits such as those which often occur in digital signal processing chips.

References

[1] M. A. Breuer and A. D. Friedman. *Diagnosis and Reliable Design of Digital Systems.* Computer Science Press, Rockville, Md., 1976.

[2] E. J. McCluskey. A survey of design for testability scan techniques. *VLSI Systems Design*, V(12):38–61, December 1984.

[3] E. Trischler. Design for testability using incomplete scan path and testability analysis. *Siemens Forsch.- u. Entwickl.-Ber.*, 13(2):56–61, 1984.

[4] V. D. Agrawal, K.-T. Cheng, D. D. Johnson, and T. Lin. A complete solution to the partial scan problem. In *Proceedings, IEEE International Test Conference*, pages 44–51, 1987.

[5] H.-K. T. Ma, S. Devadas, A. R. Newton, and A. Sangiovanni-Vincentelli. An incomplete scan design approach to test generation for sequential machines. In *Proceedings, IEEE International Test Conference*, pages 730–734, September 1988.

[6] L. M. McDonald, G. C. Chen-Ellis, and D. O. Lahti. A test generation environment for VLSI chips. In *Proceedings, IEEE Workshop on Simulation & Test Generation Environments*, pages 39–47, 1985.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, 1979.

[8] H. Walther. *Ten Applications of Graph Theory*, chapter 9, pages 197–216. D. Reidel Pub. Co., Boston, 1984.

[9] A. Kunzmann. Produktionstest synchroner Schaltwerke auf der Basis von Pipelinestrukturen. 1988. Private communication.

[10] R. E. Tarjan. *Data Structures and Network Algorithms.* SIAM, Philadelphia, 1983.