Reducing Test Application Time by Scan Flip-Flops Sharing

Abstract

In this research, we attempt to reduce the test application time while preserving the test quality or the fault coverage for circuit testing. The goal is achieved by reducing the number of scan flip-flops required for a scan-based design, and the basic idea is to look for groups of "s-independent inputs." The s-independent inputs in a group have the property that, when these inputs are combined together to share a scan flip-flop, the originally detectable faults are still detectable under the new scan structure. Though the number of test vectors may slightly increase, this can be offset by the significant reduction in the scan test width. Thus, the goal of test time reduction for scan test can be accomplished. There are also circuits which have few s-independent inputs. For these circuits, we propose to add bypass storage cells to increase the s-independencies among all inputs. Experiments have been performed on MCNC benchmarks and the results are surprisingly good. Several benchmark circuits have shown more than 90% of test time reduction.

1 Introduction

Scan-based design is a structural design for test (DFT) technique that has been widely accepted by industry [4][5][8][17]. The basic idea of scan test is to convert all or part of internal flip-flops of the circuit under test (CUT) into scan flip-flops such that the controllability and observability of the CUT can be greatly enhanced. Consider the scan procedure to test a CUT. Before a test vector is applied to the input of a module, each bit of the test vector must be serially shifted into the scan flip-flops. Then output responses are latched into the scan flip-flops and serially shifted out for verification. Normally, the scan shift clock is faster than the evaluation clock. However, for simplicity of discussion, we assume the scan shift clock is the same as the evaluation clock. The test time complexity for scan test is, therefore, proportional to the number of test vectors multiplied by the number of scan flip-flops. However, in a modern VLSI circuit, the number of internal flip-flops can be in the range of thousands or even higher. Hence, high product test cost may become a major concern in determining whether a scan-based design should be used or not.

There have been a lot of interests in developing techniques to reduce the test application time while preserving the test quality or the fault coverage for a circuit. Let the number of test vectors be the *test length,* and the size of a test vector be the *test width* which is equal to the number of scan flip-flops serially chained. In this respect, much research [7][16] attempts to reduce the test length by performing *dynamic* or *static compaction* to merge compatible test vectors during or after the test generation process.

On the other hand, techniques of multiple scan chains have been developed to alleviate the long test application time problem by reducing the test width [12][13][14][19][20]. The basic idea is to partition a single scan chain into a number of shorter scan chains, so test patterns and test responses can be shifted parallel to reduce the test application time. This method, unfortunately, will require a much higher number of extra I/O pins if a pair

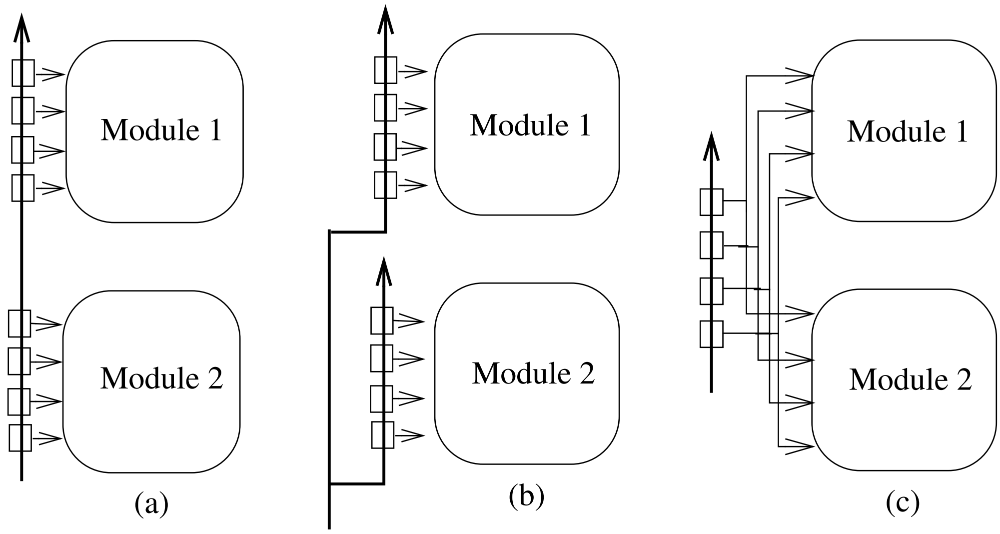


Fig. 1 Basic idea of test vectors broadcasting and scan flip-flop sharing.

of I/O pins is used for each scan chain. Moreover, currently boundary scan has become a standard for board-level testing [2][6][11][15]. However, the boundary scan test architecture allows only one pin for test data input and another one for test data output; hence cannot support multiple scan chains efficiently. Although multiplexers and de-multiplexers can be used to partly remedy these inadequacies, this will give up the most important advantage of multiple scan chains.

Instead of using the techniques of multiple scan chains, the work [1] proposed to reduce the test width by a novel scan structure which allows a test pattern to be shared by multiple circuits or modules. In this technique, it is assumed that the modules to be tested are **independent**. Traditionally, scan flip-flops of different modules are chained together as in Fig. 1(a) and the test width is equal to the summation of the test widths of all modules. In [1], the new test structure allows a test vector to be “broadcasted” to all modules as shown in Fig. 1(b), and therefore the test width can be dramatically reduced to the maximum test width among all modules. The concept of “virtual circuits” is developed in [1] for the automatic test pattern generation (ATPG) process which leads to the results that the test length for the new scan structure is generally increased by only a little amount or even no increase when compared to Fig. 1(a). According to their experiments, more than 60% of total test time reduction can be achieved in most cases compared to the conventional single scan chain technique.

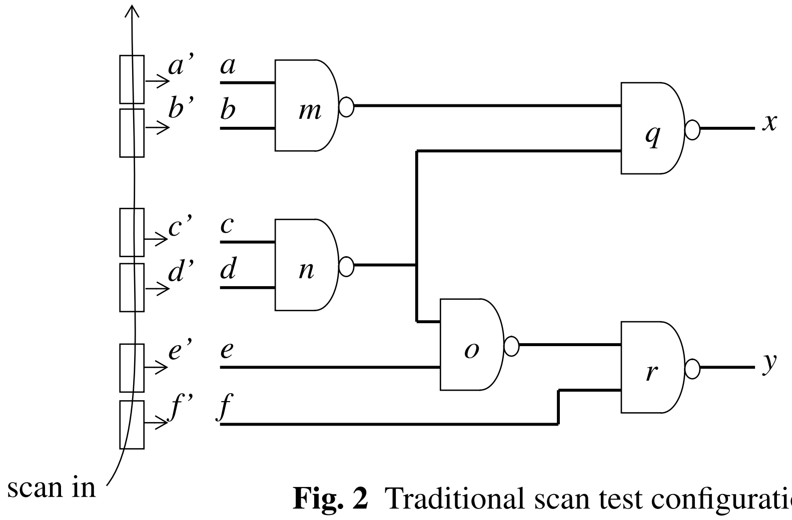
The technique in [1] assumed that a flip-flop exists at each input of all modules, and each flip-flop is allowed to be replaced by a scan flip-flop. In the case where neither condition is satisfied, we may still apply the similar test reduction concept using the test structure shown in Fig. 1(c). Here only one scan chain is employed to broadcast test vectors to all modules. It is easy to see that the same test results as those of [1] can be obtained using this “scan flip-flop sharing” structure. In fact, by a more careful observation, we find that the structure of Fig. 1(c) has the additional benefit of reducing the number of required scan flip-flops. However it will require a multiplexer for each input to each module to distinct the test data and the normal operation data. Clearly, which structure should be used will depend on the flip-flop distribution at the primary inputs of the CUT. In this paper all our discussion will be based on the structure of Fig. 1(c).

In this work, the concept of scan flip-flop sharing will be extended to the application domain for a **single** circuit (e.g., C6288), instead of multiple independent modules. The basic idea is to identify several groups of primary inputs whose output cones do not interfere with one another. It can be found that all primary inputs within each group can share the same scan flip-flop without sacrificing the fault coverage. In most cases, the benefit of test width reduction by scan flip-flop sharing can far offset the loss in test length increase. Thus, the test time can be greatly reduced. Of course, there are cases where scan flip-flop sharing does not work well. To tackle this problem, we propose a method to add a small number of bypass storage cells (bscs)[21] such that more inputs can share the scan flip-flops. This work assumes that the CUT is a combinational circuit embedded in a chip and all inputs can be accessed by a scan chain. Further, we assume that the output responses are collected and checked by a MISR as in [1] which may lose the ability of debugging. The major purpose of this research is to verify the concept and the performance of scan flip-flop sharing for a single circuit module. Of course, we will extend the idea to sequential circuits where the scan chain accesses primary inputs, outputs, and internal scan registers. Experimental results demonstrate that mostly the proposed method can dramatically reduce test application time. In some cases, more than 90% of test application time reduction can be achieved.

Section 2 uses several examples to motivate this work. A method of identifying structurally independent inputs is developed in Section 3 to share scan flip-flops. Section 4 presents the concept of adding bscs for the purpose of increasing the degree of flip-flop sharing. Experimental results are provided in Section 5, and the concluding remarks are given in Section 6.

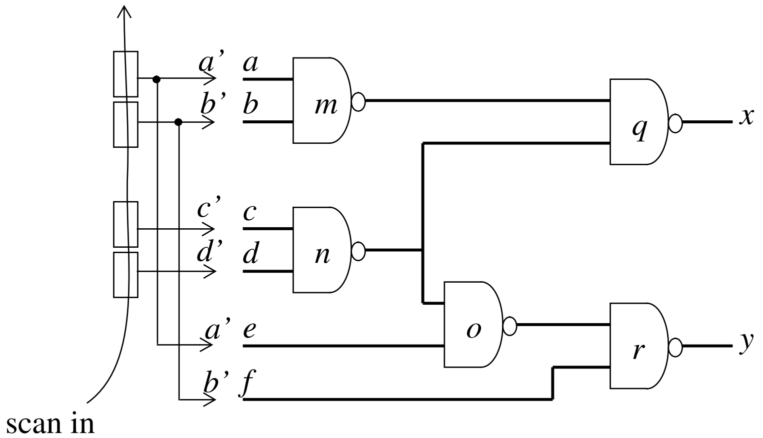
2 Main Concepts

In the following, we will use examples to illustrate our main concepts. We assume that the module under consideration is inside a circuit and test vectors for primary inputs of the module will be scanned in under a full-scan environment. Consider the example shown in Fig. 2. In this circuit, if the traditional serial scan style is applied, we have the test width equal to 6 and a set of test vectors is also shown in the figure. One can find that the test length is equal to 5. Among inputs of the circuit, input *a* and input *e* are “s-independent” in the sense that all paths from *a* to primary outputs (i.e., the output cone of *a*) do not intersect with paths from *e* to primary outputs (i.e., the output cone of *e*). If two or more inputs are s-independent, by our technique, we share the output of a scan flip-flop by all these s-independent inputs. We can guarantee that all faults can still be detected by certain test vectors after scan flip-flop sharing. In the example shown in Fig. 3, we bind together (*a,e*) and (*b,f*) and the test width is reduced from 6 to 4. For this example, after running the test generation program, a set of test vectors is shown in the figure and the test length equals to 6. In this way, we can easily reduce the test application time from 6*5=30 to 4*6=24. We will later show that maximally reducing the test width by binding s-independent inputs is an NP-complete problem.



Test Vectors					
a’	b’	c’	d’	e’	f’
1	0	0	1	1	0
1	1	0	0	1	1
0	1	1	1	1	1
0	0	1	0	0	1
0	1	1	0	0	0

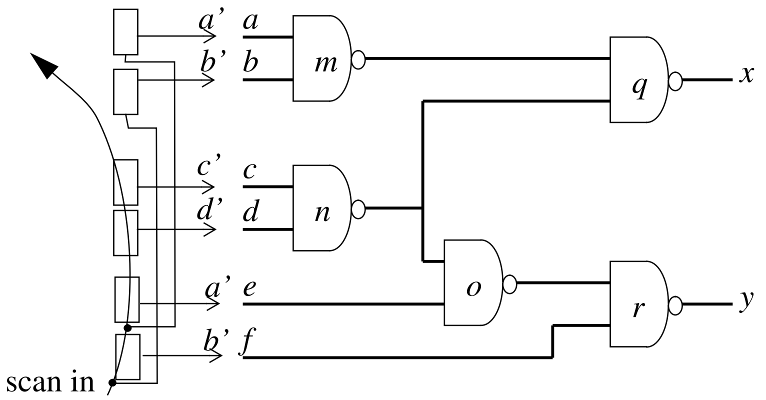
Fig. 2 Traditional scan test configuration.



Test Vectors			
a’	b’	c’	d’
1	0	0	1
1	1	0	0
0	1	1	1
0	0	1	0
0	1	1	0
1	1	1	1

Fig. 3 Test time reduction by scan flip-flop sharing. (sharing test configuration.)

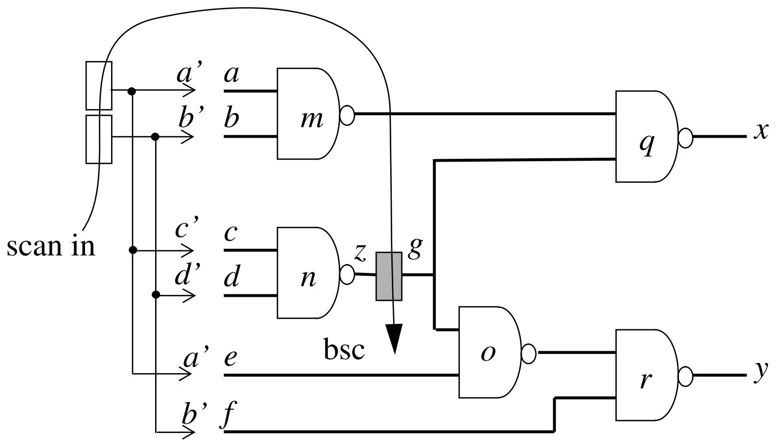
The flip-flop sharing configuration needs less scan flip-flops than the traditional style but needs to add some muxes. Another configuration in Fig. 4, named as the broadcasting test



Test Vectors			
a’	*b’*	*c’*	*d’*
1	0	0	1
1	1	0	0
0	1	1	1
0	0	1	0
0	1	1	0
1	1	1	1

Fig. 4 . Test time reduction by scan flip-flop sharing. (broadcasting test configurat

configuration carefully connects together some scan inputs of scan flip-flops. In Fig. 4, the inputs *b* and *f* will have the same value during the test mode because their scan flip-flops have the same scan inputs. In the broadcasting test configuration, we do not add additional muxes to the circuit. As a result, the broadcasting test configuration has less delay penalty than the sharing test configuration style.



Test Vectors		
a’	*b’*	*g’*
1	0	0
1	1	0
0	1	1
1	1	1
1	0	1

Fig. 5 Bsc addition for more scan flip-flop sharing.

In benchmark circuits, there are cases where we can find very few s-independent inputs and, as a result, the benefit of test width reduction is insignificant. In this paper, we also propose a technique which adds *bscs* to the CUT to increase independencies among inputs. Adding a bsc will create a new pseudo primary input and a new pseudo primary output in the test mode. These bscs can isolate some logic gates and cause more primary inputs to become s-independent during testing. Each group of s-independent primary inputs can then share the same scan flip-flop for the purpose of test application time reduction. Consider the same circuit in Fig. 2. After inserting a bsc onto node *n*, inputs *a, c*, and *e* become s-independent (so do inputs *b, d,* and *f*) and can be bound together as shown in Fig. 5. Thus, the test width has been further reduced to 3. After re-running the test generation program, we can obtain a set of vectors with test length equal 5. Finally, the test application time has been further reduced from 4*6=24 to 3*5=15.

3 Scan Flip-Flop Sharing

In this section, we first define the concept of structurally independent inputs. Then, we discuss a method to maximally reduce the number of scan flip-flops required for scan testing.

3.1 Structurally independent inputs

If there is a path from node *p* to node *q,* then node *q (p)* is said to be in the *transitive fanout (fanin) set* of node *p (q)*. The *output (input) cone* of node *n,* denoted by *outCone(n) (inCone(n)),* contains all nodes which are in the transitive fanout (fanin) set of node *n.* Two input nodes *n* and *m* are said to be *structurally independent* or *s-independent* if the output cones of node *n* and node *m* do not intersect. Similarly, a set of nodes are *s-independent* if any two inputs in the set are s-independent. For example, in Fig. 2, inputs *a* and *e* are s-independent because the output cone, {*m, q*}, of node *a* do not intersect with the output cone, {*o, r*} of *e*. Similarly, primary inputs *b* and *f* are s-independent as well.

Lemma 1: If two inputs *a* and *b* are s-independent, any test vector for a combinational fault does not require to simultaneously satisfy constraints on inputs *a* and *b.*

Proof. Suppose inputs *a* and *b* are s-independent, and outCone(*a*) and outCone(*b*) do not intersect as shown in Fig. 6(a). We discuss two conditions for a combinational fault *f:* (1) *f* is inside either one of the output cones and (2) *f* is outside both output cones. Suppose fault *f* is inside outCone(*a*) in Fig. 6(b). Then, inCone(*f*) used to sensitize *f* does not contain input *b;* otherwise outCone(*a)* and outCone(*b)* will intersect. In addition, because outCone*(f)* is a subset of outCone(*a)*, node *b* will not be in the transitive fanin set of outCone(*f*). Note that the transitive fanin set of outCone(*f*) is used to propagate fault *f.*

Since node *b* is neither in inCone(*f*), nor in the transitive fanin set of outCone(*f*), a test vector for fault *f* has no constraint on input *b.*

Suppose fault *f* is located outside outCone(*a*) and outCone(*b*). First, inCone(*f*) must not contain inputs *a* and *b.* Thus, there is no constraint on inputs *a* and *b* to excite the fault. Since we only need to propagate the fault effect of *f* to one primary output which cannot be both in outCone(*a*) and outCone(*b*)*,* there is no simultaneous constraints on inputs *a* and *b.* As a result, any test vector for a fault in the CUT does not need to simultaneously satisfy constraints on inputs *a* and *b.* ***Q.E.D.***

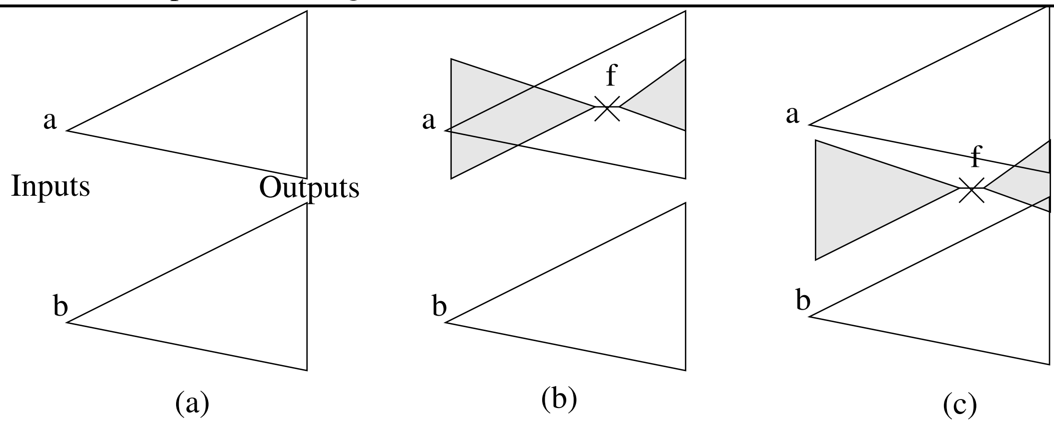


Fig. 6 The test behavior of s-independent inputs.

Lemma 2: Suppose *a* and *b* are two s-independent primary inputs. If a vector $v=(a=x, b=\bar{x}, ...)$ can detect fault *f,* then either $v'=(a=x, b=x, ...)$ or $v''=(a=\bar{x}, b=\bar{x}, ...)$ can also detect *f* where *x* is a binary value 1 or 0 and $\bar{x}$ is its complement. In other words, for any fault in the CUT, there must exist a test vector whose assignments on both inputs *a* and *b* are the same.

Proof. This can be directly derived from Lemma 1.

For example, consider the test vector *v*= *(*0,1,1,1,1,1) in Fig. 2 where *v* can detect faults {*x s-a-0*, *y s-a-1*, *n s-a-1*, *n->q s-a-1, n->o s-a-1*} after fault collapsing. Since input *a* and input *e* are s-independent, according to Lemma 2, one can easily verify that both test vector *v'*=(0,1,1,1,0,1) and *v"*=(1,1,1,1,1,1) can also detect these faults.

Consider two s-independent inputs. To test a fault, from Lemma 2, there must exist a test vector which assigns the same logic value to these two inputs. And, for all test vectors, we can enforce this requirement so that each test vector has the same logic value on these two s-independent inputs. Since each vector on these two inputs has the same logic value, we can share the output of a scan flip-flop to these two inputs. Therefore, one scan flip-flop can be saved and the test width can also be reduced by one. In fact, if a set of primary inputs are s-independent, they can share the output of a scan flip-flop and the test width reduction will be equal to (the number of s-independent inputs) -1.

On the other hand, the test length may increase when scan flip-flops are shared. The reason is that, to test a fault, the value assignments for the shared inputs are no longer don't cares in the shared structure. In traditional methods, however, these don't cares may be used for the test compaction purpose. In our experience, we find that only a very small number of test vectors will be increased and this can be offset by the significant reduction in the test width. The fact of slight increase in the number of test vectors is consistent to the results in [1].

3.2 Scan Flip-Flop Reduction by Clique Partitioning

Sharing scan flip-flops by different inputs can reduce the test width and in many cases can largely reduce the test time. In this section, we give a technique which tries to maximally reduce the test width. First, we show that the problem of maximally reducing the test width can be transformed to that of finding the minimum number of cliques in a graph.

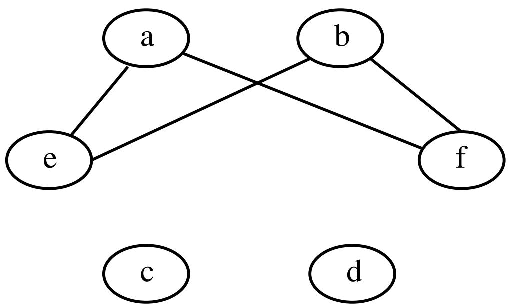


Fig. 7 The s-independent graph of the circuit in Fig. 2.

Let us define a graph called the *s-independent graph* to represent the relationship among s-independent inputs. In the graph, we create a node for each primary input; if two inputs are s-independent, an edge is created between the corresponding nodes in the graph. For example, the *s-independent* graph for the circuit in Fig. 2 is shown in Fig. 7. Because primary inputs *b* and *f* are s-independent, there is an edge connecting nodes *b* and *f* in the graph. By definition, a *clique* in a graph is a subset of the graph such that any two nodes in the subset are connected. In the s-independent graph, a clique represents a set of s-independent inputs which can share the same scan flip-flop. Thus, the minimum number of scan flip-flops required to test the CUT equals to the minimum number of cliques required to cover all nodes in the s-independent graph. This problem is equivalent to the well-known clique partitioning problem. To find the minimum number of scan flip-flops required, we first generate the s-independent graph by examining the s-independencies among all inputs. Then, we try to find the minimum number of cliques to cover all nodes in the s-independent graph. The corresponding inputs represented by the nodes in a clique can thus share the same scan flip-flop. In Fig. 6, four scan flip-flops are required since the minimum number of cliques to cover all nodes is 4.

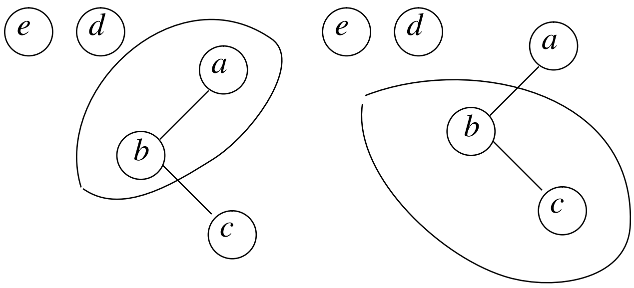
Lemma 3: The problem of maximally reducing the number of scan flip-flops by sharing is an NP-complete problem.

Proof. Omitted.

During solving the clique partitioning problem, there often exist choices for a node to be grouped by different cliques. Consider the example of an s-independent graph in Fig. 8(a). There are two choices: grouping node *a* with node *b* or grouping node *b* with node *c*. For either choice, the number of cliques required is the same. However, it may be different for the overall test time complexity which needs to consider the number of test vectors. In other words, the test length may be different even though the test width, i.e., the minimum number of cliques required to cover all nodes, is the same.

To determine the choice of grouping nodes in the clique partitioning problem, we extract hints by a trial round of test generation from the original circuit. Consider the example in Fig. 8. Assume that a set of test vectors is obtained in Fig. 8(b). The same value assignments between inputs *b* and *c* are highlighted and, in this example, the values assigned differ only in the last vector. Since inputs *b* and *c* are s-independent, the faults covered by the last vector, {01010}, must also be covered by two vectors, {01110, 00010}. If we substitute {01010} by {01110, 00010} in Fig. 8(b), all bit assignments for inputs *b* and *c* will be the same. Therefore, by binding inputs *b* and *c* together, we can immediately derive a set of new test vectors in Fig. 8(c). The set of test vectors we derived can be the worst-case estimation.

Let primary inputs *a* and *b* be s-independent. Given a set of test vectors, the *correlation value* between inputs *a* and *b* is defined as the total number of same bit assignments in the test vectors for inputs *a* and *b*. For the set of test vectors in Fig. 8(b), the correlation value between *b* and *c* is four because the value assignments for *b* and *c* are the same in the first four test vectors. The correlation value between *a* and *b* is two, however. Since two inputs with a larger correlation value will have a smaller number of estimated test vectors, when they are bound together. Thus, it is intuitive that we should group the inputs with a larger correlation value into the same clique. In this example, we group *b* and *c* rather than *a* and *c* into the same clique. In summary, for the clique partition process, whenever there is a



(a)

a	b	c	d	e
1	1	1	1	1
0	1	1	1	0
1	0	0	1	1
1	1	1	0	1
0	1	0	1	0

(b)

a	c b	d	e
1	1	1	1
0	1	1	0
1	0	1	1
1	1	0	1
0	1	1	0
0	0	1	0

(c)

Fig. 8 Clique selection for primary inputs.

choice to select a clique for a node, we always use the correlation value as our criteria.

4 BSC Insertion for Test Width Reduction

By sharing a scan flip-flop to several inputs of a circuit, we can significantly reduce the test widths for many benchmark circuits and still can maintain the same fault coverage. However, for circuits with few s-independent primary inputs, the reduction of test widths may be limited. For example, consider an irredundant circuit with only one primary output. Since the output cones of all primary inputs must contain the primary output node, there are no s-independent primary inputs. Therefore, no scan flip-flop can be shared. To tackle this problem, we propose to insert bscs to the CUT so that the number of s-independent inputs can be increased. In the normal mode, these bscs are transparent; while in the test mode, they act as pseudo primary inputs and pseudo primary outputs. In the following, we will not distinguish between pseudo primary inputs (outputs) and primary inputs (outputs) because they behave in the same way during the test mode.

A bsc can isolate some portion of circuits from others in the test mode, and may cause some originally **non**-s-independent inputs to become s-independent. For example, in Fig. 9(a), primary inputs *a* and *c* are not s-independent. In Fig. 9(b), we insert a bsc to the circuit where a pseudo input *d* and a pseudo output *y* have been created in the test node. By the bsc insertion, inputs *a* and *c* become s-independent since their output cones do not intersect.

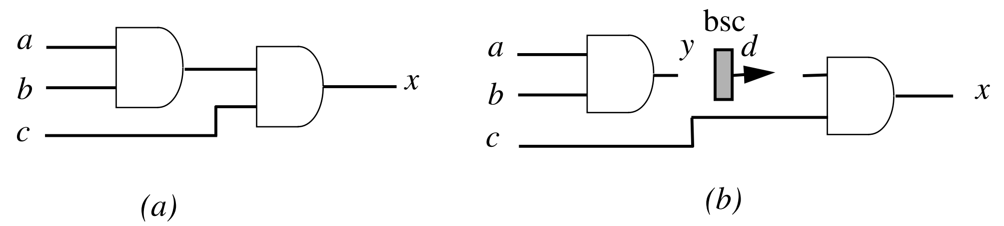


Fig. 9 The test behavior for a bsc.

Now, we discuss our strategy of adding bscs for increasing the number of s-independent inputs. Let the *support* of a primary output *Po* be all primary inputs in the input cone of *Po,* and the size of a support be the number of inputs in the support. Obviously, no two primary inputs in the support of a primary output are s-independent, so they cannot share the same scan flip-flop. As a result, if a PO has *n* supports, we must use *n* distinct scan flip-flops to drive these inputs. Among the supports of all primary outputs, assume the maximum size is MAX_SUPPORT. Then, a lower bound for the minimum number of scan flip-flops required is equal to MAX_SUPPORT.

Since MAX_SUPPORT is a lower bound, in our algorithm, bscs are added to the CUT so that the value of MAX_SUPPORT can be reduced. Though reducing MAX_SUPPORT does not necessarily result in the reduction of test width, it can be found that this strategy works well in many cases. Further, it should be emphasized that the problem of reducing MAX_SUPPORT is similar to the problem of partitioning for pseudo-exhaustive testing [9] [18]. In the following, we first describe the scenario of pseudo-exhaustive testing and then discuss the differences in circuit partitioning between our method and pseudo-exhaustive testing.

In pseudo-exhaustive testing, the CUT is partitioned into sub-circuits which will be tested exhaustively. The time required for pseudo-exhaustively testing the CUT is proportional to $2^{\text{MAX\_SUPPORT}}$. Most research in this area try to partition the CUT so that the value of MAX_SUPPORT is limited to a specified number (e.g., 20) and, at the same time, the number of bscs inserted is minimized. Consider the example in Fig. 10(a) where the circuit has MAX_SUPPORT equal 6. After adding two bscs into the circuit, we create two pseudo outputs *b1_o* and *b2_o* and two pseudo inputs *b1_i* and *b2_i* as shown in Fig. 9(b). Note that *b1_i* and *b2_i* are the outputs of both bscs while *b1_o* and *b2_o* are the inputs of both bscs. It can be easily verified that all outputs have the support sizes smaller than or equal to 3. In Fig. 10(a), the circuit has no s-independent inputs. However, in the new circuit of Fig. 10(b), each of the three groups {(*a*, *c, e*), (*b, d, f*), (*b1_i, b2_i*)} contains s-independent inputs, so the total number of scan flip-flops required is 3.

Though the objective of pseudo-exhaustive testing is quite similar to ours, there are still

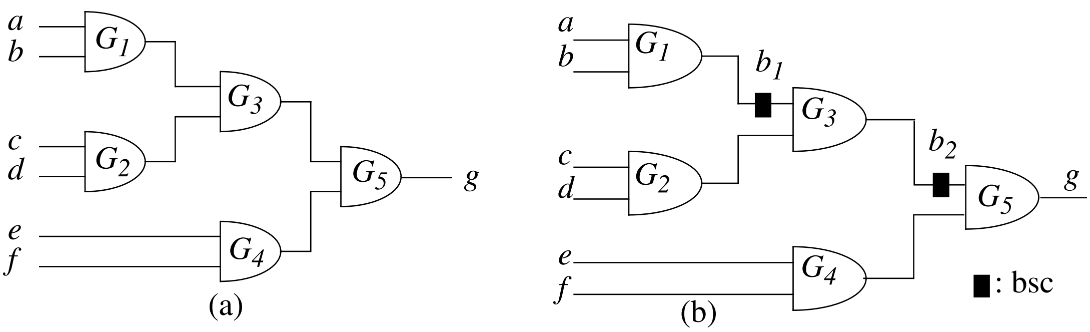


Fig. 10 A complete example for bsc insertion.

some differences. First, MAX_SUPPORT in pseudo-exhaustive testing cannot be too large because test time is exponential to this number; however in our case there is no restriction to the test width and MAX_SUPPORT for scan test. Traditionally, the test width can be as large as the number of inputs to the CUT which can be hundreds or even thousands. The value of MAX_SUPPORT for scan test is not so critical as in the case of pseudo-exhaustive testing, thus it is highly dependent on the trade-off between hardware overhead and test application time. Additionally, in pseudo-exhaustive testing, there is no advantage to add bscs to the CUT without reducing MAX_SUPPORT. On the contrary, in our case, it is still beneficial to add bscs to the CUT where MAX_SUPPORT is not reduced, because more primary inputs may turn to be s-independent and the test width can thus be greatly reduced. The bsc insertion may incur additional delay penalty. Since there are many places

```
share_scan_ff_no_bsc(circuit)
{
  extract_s_independcy();
  build_s_independent_gragh();
  min_clique_partitioning();
}
```

(a)

```
share_scan_ff_add_bsc(circuit, maxSup)
{
   insert_bscs_max_support_less_n(maxSup);
   share_scan_ff_no_bsc(circuit);
}
```

(b)

Fig. 11 Pseudo codes of the proposed method.

of adding bscs, we can take into account the delay information of the circuit during the bsc insertion process so that bsc cells are not added to the critical paths.

5 Experimental results

We implemented two programs to reduce scan flip-flops for testing a circuit. In the first program shown in Fig. 11(a), s-independencies among primary inputs are extracted; then the s-independent graph is built. After that, we try to generate the minimum number of cliques from the graph using a heuristic method. Finally, for each clique, we output the corresponding primary inputs and let them share the same scan flip-flop. In the second program shown in Fig. 11(b), bscs are added to the CUT for increasing the s-independencies. The program adds bscs so that the support size of each primary output and each pseudo primary output does not exceed input variable *maxSup*. Then, the test width reduction method of Fig. 11(a) is performed. Note that the smaller the value of *maxSup* is, the more bscs are required; usually, this results in more test width reduction. In our experiments, *maxSup* is determined to half the size of total primary inputs in the original circuit. The value can be carefully tuned by designers for trading off among delay, area and test time.

The programs are applied to a set of MCNC and ISCAS benchmarks. In our experiments, we generate test patterns using a commercial ATPG tool. Table 1 shows the experimental results of the proposed method without adding bscs. The first column shows the name of each circuit, the second column gives the size of each circuit using the number of literal count. The third column gives the number of inputs and the forth column shows the number of test vectors using the ATPG tool. The fifth column shows the test complexity by multiplying the number of inputs and the number of test vectors assuming the traditional way of scan design. The sixth column shows the number of scan flip-flops required by our method and the seventh column provides the number of test vectors required by the shared scan flip-flops environment. In the eighth column, we also show the test complexity similar to the fifth column. Finally, we show the improvement in test time complexity by our method. For example, circuit C5315 has 178 primary inputs and the number of test vectors required is 209 using traditional scan methods. In our solution, we only need 69 shared scan flip-flops and 204 test vectors are sufficient to test the circuit with the same fault coverage. The improvement rate is 62%. Several circuits highlighted in the table have improvement rate more than 90%.

It is to our great surprise that the total number of test vectors can also be reduced in several examples. We suspect that it is due to inefficiency of the test compaction algorithm used by the tool or the circuits may contain special modularized structures. The ATPG tool from SIS [3] is also tried and the improvement ratios are consistent with those of the commercial tool. For space limit, the tables for SIS results are not given here. It should be emphasized that all results have been carefully verified through the following procedure. For a circuit, we first generate a set of test vectors for the CUT with flip-flops shared. Each test vector from the shared structure will then be expanded by duplicating the logic value(s) to all s-independent inputs. For example, the first vector (*a',b',c',d'*)=(1,0,0,1) in Fig. 3 will be expanded to (*a,b,c,d,e,f*)= (1,0,0,1,1,0) where the value of input *a'* will be copied to input *a*, and input *e*. Fault simulation using the expanded test vectors is performed and this will check whether all faults are covered by the set of shared test vectors. In the tenth and eleventh columns, we also report the delay results of the circuits. For each circuit, we compute the delay of the original circuit by the delay script, script.delay in SIS [3] and the delay of the circuit with the sharing test configuration which adds some muxes to achieve the input sharing. We also would like to mention that the delay of a circuit with the broadcasting test configuration has the same delay as the original circuit.

For circuits which have few s-independent inputs, we try to add bscs to increase s-independencies and the results are shown in Table 2. In addition to all the information shown in Table 1, in the ninth column, the number of bscs added to each circuit is demonstrated. For example, circuit C3540 has 50 primary inputs, and the number of test vectors required is 259. In our solution, the s-independencies of primary inputs are increased by adding 51 bscs. As a result, only 31 scan flip-flops are needed and only 222 test vectors are sufficient to test the circuit with the same fault coverage. The improvement rate is 43%. For circuits in the table, the average improvement rate is 51%.

Table 1: The results of Fig. 11a (no bscs) using a commercial ATPG.

circuit	literal count	# of FFs (before)	# of vect. (before)	total bits (before)	# of FFs (after)	# of vectors (after)	total bits (after)	improvement (%),	delay (before)	delay (after)
b9	256	41	71	2911	14	70	980	66	6.99	8.96
C2670	2043	233	171	39843	122	180	21960	44	24.29	24.48
C5315	4369	178	209	37202	69	204	14076	62	31.13	31.93
c8	362	28	81	2268	13	82	1066	52	9.19	10.38
C880	703	60	109	6540	45	100	4500	31	17.13	17.54
cc	105	21	34	714	7	25	175	75	5.97	6.33
cm163a	69	16	24	384	9	21	189	50	7.26	7.29
cm85a	68	11	23	253	10	27	270	6	7.39	7.4
cmb	69	16	40	640	12	33	396	38	9.41	10.21
count	174	35	33	1155	20	29	580	49	19.71	19.71
cu	98	14	52	728	13	45	585	19	5.56	5.58
example2	432	85	88	7480	17	84	1428	80	9.84	10.07
frg1	792	28	291	8148	25	310	7750	4	10.45	10.45
frg2	2851	143	262	37466	26	233	6058	83	15.65	16.46
i1	72	25	27	675	11	27	297	56	5.42	5.74
i3	1816	132	98	12936	32	33	1056	91	3.72	4.52
i4	1040	192	180	34560	47	57	2679	92	6.04	6.84
i5	556	133	103	13699	19	51	969	92	8.01	8.79
i6	1037	138	58	8004	5	15	75	99	7.36	8.16
i7	1311	199	61	12139	6	16	96	99	7.41	8.21
cht	374	47	49	2303	6	24	144	93	5.87	6.67
des	7412	256	257	65792	22	219	4818	92	15.33	16.13
tcon	48	17	17	289	3	4	12	95	3.24	4.04
i8	4626	133	172	22876	21	162	3402	85	17.27	17.27
i9	1453	88	73	6424	17	59	1003	84	15.02	15.26
i10	5376	257	631	162167	108	654	70632	56	36.81	36.81
lal	257	26	91	2366	13	86	1118	52	8.14	8.34
pair	2667	173	340	58820	57	294	16758	71	19.47	19.69
pcle	78	19	25	475	12	23	276	41	9.32	9.32
pcler8	102	27	29	783	13	25	325	58	10.46	10.98
pm1	98	16	39	624	9	36	324	48	4.83	4.96
rot	1424	135	371	50085	65	355	23075	53	22.9	22.9

Table 1: The results of Fig. 11a (no bscs) using a commercial ATPG.

circuit	literal count	# of FFs (before)	# of vect. (before)	total bits (before)	# of FFs (after)	# of vectors (after)	total bits (after)	improvement (%),	delay (before)	delay (after)
sct	234	19	77	1463	14	80	1120	23	6.73	6.73
term1	997	34	343	11662	20	341	6820	41	15.94	15.83
ttt2	719	24	141	3384	14	134	1876	44	10.73	10.77
unreg	144	36	28	1008	6	13	78	92	5	5.8
x1	2141	51	647	32997	27	613	16551	49	10.92	10.92
x3	1816	135	269	36315	24	234	5616	84	13.5	14.3
x4	1040	94	161	15134	16	133	2128	85	9.26	10.25
mux	142	21	68	1428	142	21	1428	0	7.12	7.12
cm85a	68	11	23	253	11	23	253	0	6.6	6.6
Total	49439	3347	5866	704393	1142	5168	222942	68	472.39	489.74

Table 2: The results of Fig. 11b (add bscs) using a commercial ATPG.

circuit	**literal count**	# of inputs (before)	# of vectors (before)	total bits (before)	# of inputs (after)	# of vectors (after)	total bits (after)	# of FFs	improvement (%)
C1355	**1032**	41	109	4469	24	96	2304	8	48
C1908	**1497**	33	171	5643	28	150	4200	18	25
C3540	**2934**	50	259	12950	31	222	7326	51	43
C432	**372**	36	77	2772	21	89	1869	22	32
C499	**616**	41	109	4469	24	96	2304	8	48
C6288	**4800**	32	40	1280	23	56	1232	126	3
C7552	**6098**	207	344	71208	113	341	38533	4	45
cm150a	**92**	21	57	1197	10	22	220	4	81
cm151a	**45**	12	26	312	6	14	84	2	73
comp	**200**	32	75	2400	14	30	420	6	82
cordic	**194**	23	43	989	11	43	473	5	52
dalu	**3588**	75	263	19725	40	321	11877	14	39
i2	**268**	201	226	45426	76	171	12996	4	71
mux	**142**	21	68	1428	7	26	182	4	87
my_adder	**305**	33	27	891	15	27	405	2	54
parity	**60**	16	9	144	8	7	56	2	61
C880	703	60	109	6540	34	87	2958	6	54
cm85a	68	11	23	253	5	16	80	7	68
count	174	35	33	1155	17	27	459	1	60
pcle	78	19	25	475	9	21	189	1	60
Total	**23266**	999	2093	183726	516	1862	88167	295	52

6 Conclusions and Future Research

In this paper, to reduce the test application time for a CUT, we have proposed the idea of s-independent inputs which can share the same scan flip-flop. We also showed that the problem of minimizing the number of scan flip-flops can be transformed to that of minimizing the number of cliques in a graph. For circuits with few s-independent inputs, bscs are added to increase the s-independencies among primary inputs. Experimental results show that significant improvement in test application time can be obtained using our method. Although the discussion is mainly based on the domain of combinational circuits whose primary inputs are accessed by a scan chain, we can expect the success of extending the idea to sequential circuits with internal flip-flips scanned. Adding bscs can significantly improve the performance for circuits containing very few s-independent inputs. Of course, this also produces side effects of hardware overhead and performance degradation. To alleviate the disadvantages, we will investigate a more powerful method to insert as small number of bscs as possible.

7 Reference

[1] K. J. Lee, J. J. Chen, and C. H. Huang, “Using a Single Input to Support Multiple Scan Chains”, accepted to the 1998 ICCAD.

[2] H. Bleeker et al., “Boundary-Scan Test: a Practical Approach”, Kluwer Academic Publishers, 1993.

[3] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, A. R. Wang, “MIS: Multi-level Interactive Logic Optimization System,” IEEE Trans. on CAD, CAD-6(6), pp. 1062-1081, Nov. 1987.

[4] H. Courjon, “Scan Design in the Philips ASIC Test Environment”, Euro ASIC 1990, pp.370-375.

[5] S. Funatsu, M. Kawai, and A. Yamada, “Scan Design at NEC”, IEEE Design & Test of Computers, vol. 6, no. 3, pp.50-57, June 1989.

[6] S. Hronik, “Industry Acceptance of Boundary Scan”, Proc. of WESCON 1994, pp.518-524.

[7] P. Goel and B. C. Rosales, “Test Generation and Dynamic Compaction of Tests”, ITC 1979, pp.189-192.

[8] Y. Hagihara, C. Ohkubo, F. Okamoto, H. Yamada, M. Takada ,and T. Enomoto, “Design For Testability In A 200MFLOPS Vector-Pipelined Processor (VPP)-ULSI”, Proc. of First Asian Test Symposium 1992, pp.223-228.

[9] W. B. Jone and C. A. Papachristou, “A Coordinate Circuit Partitioning and Test Generation Method for Pseudo-Exhaustive Testing of VLSI Circuits”, IEEE Trans. on CAD, vol. 14, no. 3, pp.374-384, Mar, 1995.

[10] S. Kobayashi, M. Edahiro, and M. Kubo, “Scan-Chain Optimization Algorithms for Multiple Scan-Paths”, Proc. of Asia and South Pacific Design Automation Conference 1998, pp.301-306.

[11] C. M. Maunder, R. E. Tullos, “The Test Access Port and Boundary-Scan Architecture”, IEEE Computer Society Press, 1990.

[12] S. Narayan et al., “Configuring Multiple Scan Chains for Minimum Test Time”, ICCAD 1992, pp.4-8.

[13] S. Narayanan and M. A. Breuer, “Asynchronous Multiple Scan Chains”, Proc. of 13th IEEE VLSI Test Symposium 1995, pp.270-276.

[14] S.Narayanan, R. Gupta, and M. A. Breuer, “Optimal Configuring of Multiple scan chains”, IEEE Trans. on Computers, September 1993.

[15] A. Osseiran, “Test Standards (with Focus on IEEE1149.1)”, Proc. of 38th Midwest Symposium on Circuit and System, vol. 2, pp.708-711, 1996.

[16] I. Pomeranz et al., “COMPACTEST: A Method to Generate Compact Test Sets for Combinational Circuits”, ITC 1991, pp.194-203.

[17] T. J. Powell, F. Hwang, and B. Johnson, “Testability features in the TMS 370 family of microcomputers” Proc. of New Frontiers in Testing ITC 1988, pp.153-160.

[18] R. Srinivassan, S. K. Gupta, and M. A. Breuer, “An Efficient Partitioning Strategy for Pseudo-Exhaustive Testing”, Proc. of ACM/IEEE DAC 1993, pp.242-248.

[19] B. Ye, Z. Zheng, J. Hu, and W. Li, “Testability Design with Multiple Scan Chains”, Proc. of 4th International Conference on Solid-State and IC Technology 1995, pp.637-639.

[20] S. Bhatia, N. Jha, “Synthesis for Parallel Scan: Applications to Partial Scan and Robust Path-Delay Fault Testability”, TCAD Feb. 1996, pp.228-243.

[21] M. Abramovici, M. A. Breuer, A. D. Friedman, “Digital Systems Testing and Testable Design”, pp.476-480.