

# K Longest Paths Per Gate (KLPG) Test Generation for Scan-Based Sequential Circuits

Wangqi Qiu, Jing Wang, D. M. H. Walker, Divya Reddy<sup>+</sup>, Xiang Lu<sup>\*</sup>, Zhuo Li<sup>\*</sup>, Weiping Shi<sup>\*</sup>, Hari Balachandran<sup>+</sup>

Dept. of Computer Science  
Texas A&M University  
College Station TX 77843-3112  
Tel: (979) 862-4387  
Fax: (979) 847-8578  
Email: walker@cs.tamu.edu

<sup>\*</sup>Dept. of Electrical Engineering  
Texas A&M University  
College Station TX 77843-3124  
Tel: (979) 458-0093  
Fax: (979) 845-2630  
Email: wshi@ee.tamu.edu

<sup>+</sup>Texas Instruments, Inc.  
12500 TI Boulevard MS 8761  
Dallas TX 75243  
Tel: (214) 480-3783  
Fax: (214) 480-3449  
Email: {d-reddy2, harib}@ti.com

## Abstract

*To detect the smallest delay faults at a fault site, the longest path(s) through it must be tested at full speed. Existing test generation tools are inefficient in automatically identifying the longest testable paths due to the high computational complexity. In this work a test generation methodology for scan-based synchronous sequential circuits is presented, under two at-speed test strategies used in industry. The two strategies are compared and the test generation efficiency is evaluated on ISCAS89 benchmark circuits and industrial designs. Experiments show that testing transition faults through the longest paths can be done in reasonable test set size.*

## 1. Introduction

Delay test has been investigated for many years. At-speed test significantly increases the delay fault coverage in industrial applications. The transition fault model [1], which is the simplest delay fault model, is usually used in these applications. However, the transition fault model targets large delay faults which cause all the sensitizable paths through the fault site to be slow. Recent research shows that resistive opens are one of the major defect types which cause delay faults [2], and that small delay faults cannot be neglected [3]. To detect the smallest delay fault at a fault site, the longest sensitizable paths through it must be tested. But (longest) path delay fault test generation is much more expensive than transition fault test generation, because a transition fault test can be composed by pairing stuck-at-0 and stuck-at-1 vectors [4] and transition fault test generation for sequential circuits has been extensively investigated [5][6][7].

Recently some research significantly decreased the cost of path delay fault test generation [8][9] and these methodologies are able to integrate some path selection criteria, such as the longest paths through each line. However, they assume the circuits are combinational, i.e. there is no dependence between the two test vectors or between two bits within a vector. These methodologies cannot be applied to sequential circuits directly. The reason is that the commonly-used design-for-testability (DFT) structures, such as muxed scan, rarely support combinational enhanced-scan, which allows the two vectors to be independent but requires more silicon area

and introduces more delay. Therefore, a new automatic test pattern generation (ATPG) tool for path delay faults in sequential circuits has to be developed, to target practical DFT structures.

The ATPG was developed by extending a path generation algorithm for combinational circuits [9] to handle scan-based synchronous sequential circuits. This tool is able to generate K longest paths through the input and output pins of each gate (KLPG) for both slow-to-rise and slow-to-fall faults. In this work test generation is limited to K=1 because it is assumed the industry cannot afford a test set much larger than a transition fault test set. At speed testing that utilizes scan, often called AC scan, uses two common approaches, “launch-on-shift” and “launch-on-capture”. The constraints from these approaches result in sequential false paths [10] which are combinationally testable. The delays of the longest combinational and sequential testable paths through each line are compared in the experiments.

The remainder of the paper is organized as follows. Section 2 introduces the two practical test approaches to apply at-speed test in a scan-based circuit. Section 3 describes the test generation algorithm using the two approaches. Section 4 includes experimental results on the ISCAS89 benchmark circuits and industrial designs. Section 5 concludes with directions for future research.

## 2. Scan-Based At-Speed Test Approaches

In low-cost automatic test equipment (ATE), the test speed is usually much slower than the functional speed of the circuit under test. This is not a problem to test stuck-at and large delay faults, but small delay faults may escape. Therefore at-speed test is preferred to increase the realistic delay fault coverage.

However, due to the low-cost ATE speed limitation, the at-speed tests primarily in use in industry are built-in self-test (BIST) and AC scan. Evidence has shown that BIST can achieve very high fault coverage for stuck-at and transition faults [11], but it has low probability to sensitize enough critical paths, e.g. the longest path through each line. On the other hand, functional tests running at full speed are becoming unattractive due to the high cost of development and application [12]. Therefore, this paper focuses on high-quality delay test generation using existing scan designs.

In this paper the muxed scan design is assumed, with a scan enable signal selecting either serial scan data or functional logic data. The flip-flops are clocked with the system clock. Two scan-based at-speed test methodologies, which have found increasing usage in industry, will be briefly introduced in the next two sections.

### 2.1. Launch-on-Shift (Skewed Load)

The procedure of the launch-on-shift (or skewed load [13][14]) test approach is:

1. The circuit is set to scan mode. The first test vector is scanned into the scan chains using the slow scan clock, and the values are set on primary inputs (PIs).
2. The second test vector is obtained by shifting the scan chain by one bit. Usually the PIs do not change values due to the constraints from low-cost ATEs.
3. The circuit is set to the functional mode by flipping the scan-enable signals and pulsing the system clock to capture the circuit values in the flip-flops. The values on primary outputs (POs) are captured if necessary.
4. The circuit is set to scan mode and the values in the scan chains are scanned out using the slow scan clock. This step can be overlapped with step 1.

The advantage of this approach is that fast test generation methodologies for combinational circuits can be applied without many modifications. Scanned flip-flops are considered primary inputs in the ATPG for combinational circuits, and the adjacent scan bit dependencies must be added to the existing ATPG. These constraints may result in some paths being untestable.

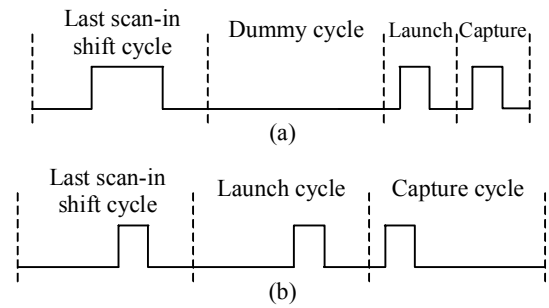
The disadvantage of this approach is that the scan enable signals must operate at full speed. In addition, many of the sensitizable paths under the launch-on-shift constraints are sequential false paths, i.e. these paths are not sensitizable in functional mode, so some redundant faults would be detected.

### 2.2. Launch-on-Capture (Functional Justification)

The procedure of the launch-on-capture (or functional justification, broadside [15]) test approach is:

1. Same as the launch-on-shift approach step 1.
2. The circuit is set to functional mode. A dummy cycle is inserted if the scan-enable signal cannot operate at full speed or the system clock frequency is very high, so that the launch clock pulse width is too large. Figure 1(a) shows the clock waveform. For comparison, Figure 1(b) shows the clock waveform if the time is sufficient for the scan enable signal to propagate. In this approach, the launch cycle is kept identical to the shift cycle with respect to period, rising edge, and pulse width.
3. The system clock is pulsed twice. At the first clock pulse, the second test vector is derived from the first vector. At the second clock pulse, the test is performed and the output values are captured in the scanned flip-flops. The values on POs are captured if necessary.

4. Same as the launch-on-shift approach step 4.



**Figure 1. Launch-on-capture clock waveforms.**

The advantage of this approach is that it does not require the scan enable signal to operate at full speed. And the sensitizable paths under the launch-on-capture constraints are also sensitizable in functional mode, unless the first vector represents an illegal state.

Though the launch-on-capture approach is more promising and practical for industrial use [12], the launch-on-shift approach is also included in this work because it may have lower data volume, it may detect some delay faults that are not functional, and test generation only requires combinational test.

## 3. Test Generation

The test generation algorithm was developed from a fast ATPG for combinational circuits [9]. In this section, the path generation engine is introduced, and the constraints from the launch-on-shift/capture approach are applied to eliminate sequential false paths from the combinationally testable path set, and the time frame expansion method is used for the launch-on-capture approach.

### 3.1. KLPG Path Generation Engine

Figure 2 is the algorithm used in the KLPG path generation engine [9]. In this paper, a *launch point* (of a path) is a primary input or scanned flip-flop, and a *capture point* is a primary output or a scanned flip-flop. In the preprocessing phase, static timing analysis computes the maximum delay from each gate to capture points, without considering any logic constraint. This value is termed the *PERT delay* or *STA delay*. In the path generation phase, *partial paths* are initialized from launch points. A partial path is a path which originates from a launch point but has not reached any capture point. A value called *esperance* [16] is associated with a partial path. The *esperance* is the sum of the delay of the partial path and the STA delay from its last node to a capture point. In other words, the *esperance* of a partial path is the upper bound of its delay when it becomes a *complete path*, which reaches a capture point.

In each iteration of the path generation phase, the partial path with the maximum *esperance* value is extended by adding one gate. If the last gate of the partial path has more than one fanout, the partial path splits. Then the

constraints to propagate the transition on the added gate, such as non-controlling side input values required under the robust [17] or non-robust [18] sensitization criterion, are applied. *Direct implications* [16] are then used to propagate the constraints throughout the circuit. A direct implication on a gate is one where an input or output of that gate can be directly determined from the other values assigned to that gate. Figure 3 shows some examples of direct implications on an AND gate. The values in boxes are implied from the existing values. If there are any conflicts, the whole search space which contains the partial path is trimmed off. If the partial path does not reach a capture point, some false path elimination techniques [9] are applied to prevent it from growing to a false path. Then its esperance value is updated and it is inserted back into the partial path store. If a partial path becomes a complete path, final justification is performed to find a vector. Details of final justification are provided in Section 3.4. This process repeats until enough longest testable paths are generated. Because the longest path through a fault site is very possibly the longest path through other fault sites along the path, fault dropping is performed when a new path is generated.

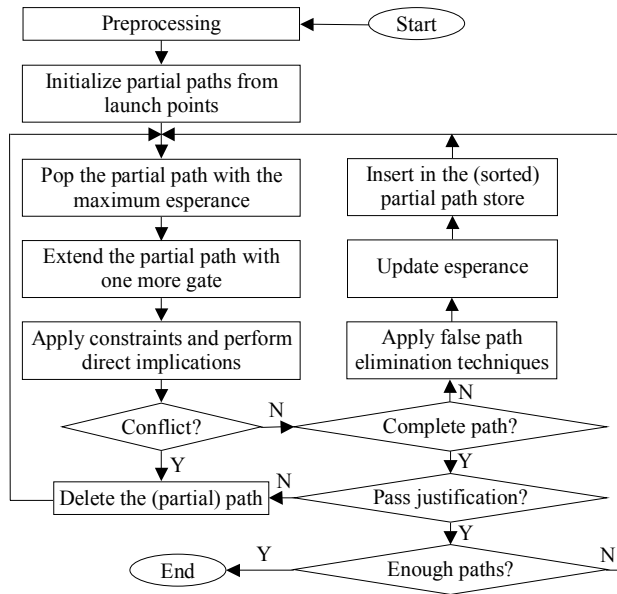


Figure 2. Path generation algorithm.

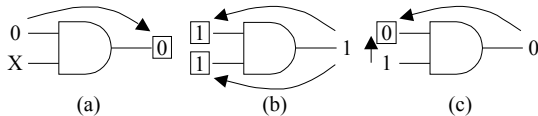


Figure 3. Examples of direct implications [9].

In this work, the goal is to generate the K longest paths through the input and output pins of each gate for both slow-to-rise and slow-to-fall faults, termed KLPG patterns. So the search space is limited to the fanin and fanout cones

of the fault site and the test generation does not stop until the K longest paths for both faults are generated.

### 3.2. Implications on Scanned Flip-Flops

Direct implications can be performed on scanned flip-flops as well as regular gates to detect most local conflicts and eliminate sequential false paths. Since local conflicts are the fundamental reason for false paths in most circuits [16], performing direct implications as much as possible can identify most false paths and significantly speed up the test generation process.

If the launch-on-shift approach is used, the logic values on neighboring scanned flip-flops are dependent on each other. For example, in Figure 4, the logic value of cell A in the first vector is the same as that of cell B in the second vector. The relation between cell B and C is the same. Therefore, if there is a rising transition assigned to cell B, direct implications would try to assign a logic 1 to cell A in the first vector and a logic 0 to cell C in the second vector, and propagate the new assignments throughout the circuit. If there are any conflicts, the partial path is a sequential false path under the launch-on-shift constraints. It is assumed that the scan chain design cannot be modified to reduce the dependence, such as inserting dummy cells between the scanned flip-flops.

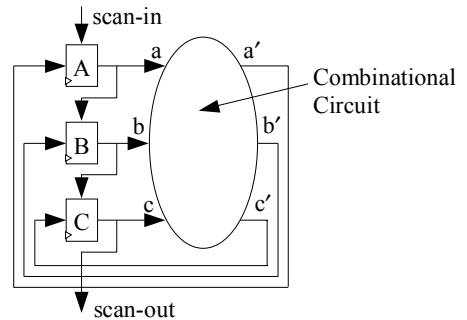


Figure 4. Implications on scanned flip-flops.

If the launch-on-capture approach is used, dependence exists between the two vectors. Even if the circuit has a pipeline structure, in which the two vectors are independent, the structure can also be seen as the general structure shown in Figure 4. The conversion is shown in Figure 5. Thus the second vector is the output of the combinational circuit, derived from the first vector, excluding the primary input and output bits. In other words,  $V_2=C(V_1)$ , where  $V_1$  and  $V_2$  are the two vectors and  $C$  is the logic of the combinational circuit. For example, if it is assumed that a testable path has a rising transition launching from cell A and a rising transition captured on cell B, in Figure 4, then for the first vector, output  $a'$  must be a logic 1 (then it becomes the value for input  $a$  in the second vector); and for the second vector, input  $b$  must be a logic 0 because it is derived from the first vector. Then more direct implications can be performed from  $a'$  and  $b$ .

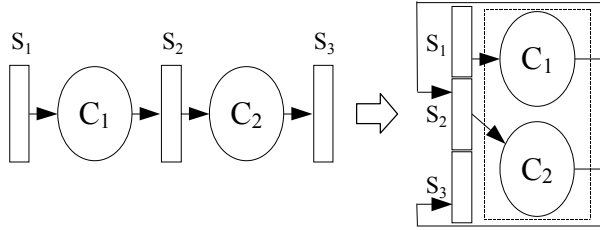


Figure 5. A pipeline structure.

### 3.3. Constraints from Non-Scanned Memories

If the circuit is not full scan, the non-scanned flip-flops may or may not be initialized after the first vector is scanned in. Logic simulation of the test setup sequence and scan procedure is done before the ATPG is performed. If a non-scanned flip-flop gets initialized by its set/reset signal during the test setup procedure, the constant value is used as a known value in the first vector during the ATPG. Logic values of the set/reset signals of each flip-flop are also checked by logic simulation, to ensure the set/reset signals are in their off state during the scan and test procedure. Clocks are checked in the same way. Flip-flops not controlled by the system clock are considered uncontrollable. Industrial designs also contain embedded memories, whose values cannot be easily initialized during the test for the logic. Extensive research [19][20] has been done to solve the initialization problem. However, there may still be many non-scanned flip-flops which cannot be initialized. These bits are considered “uncontrollable” in the test generation. In commercial tools, embedded memories are usually considered “black-boxes” as well.

	0	1	$x$	$u$	0/ $u$	1/ $u$	$x/u$
0	0	0	0	0	0	0	0
1	0	1	$x$	$u$	0/ $u$	1/ $u$	$u$
$x$	0	$x$	$x$	0/ $u$	0/ $u$	$x/u$	$x/u$
$u$	0	$u$	0/ $u$	$u$	0/ $u$	$u$	0/ $u$
0/ $u$	0	0/ $u$	0/ $u$	0/ $u$	0/ $u$	0/ $u$	0/ $u$
1/ $u$	0	1/ $u$	$x/u$	$u$	0/ $u$	1/ $u$	$x/u$
$x/u$	0	$u$	$x/u$	0/ $u$	0/ $u$	$x/u$	$x/u$

Figure 6. Truth table of an AND gate.

The algebra used in this work includes seven values: logic 0/1,  $x$  (unknown/unassigned),  $u$  (uncontrollable), 0/ $u$  (0 or uncontrollable), 1/ $u$  (1 or uncontrollable) and  $x/u$  (unknown or uncontrollable). At the beginning of test generation, the lines from the non-scanned memories are  $u$  and all the other lines are  $x$ . Both  $u$  and  $x$  have “don’t know” values but  $x$  may be assigned a value in the test generation process (assuming controllable). Figure 6 shows the truth table of a 2-input AND gate. For example, if one input is  $x$  and the other is  $u$ , the output is 0/ $u$  because if the input with  $x$  is assigned a logic 0 the output becomes 0, but if this input is assigned a logic 1 the output becomes uncontrollable. Before the test generation, logic simulation is performed throughout the circuit to reduce the number of  $x$ ’s. Figure 7 shows two examples, assuming  $M_1$  is a

non-scanned memory cell and  $M_2$  is a scanned flip-flop. The logic values assigned by simulation are shown. If the conventional 3-value algebra is used, all the lines are assigned  $x$ ’s.

Using this 7-value algebra significantly speeds up the test generation because it divides unknown values into controllable and uncontrollable categories. In the example shown in Figure 7(a), since the logic value on line  $n_3$  can never be a logic 1, all the paths through line  $n_4$  are false. Thus the test generation stops growing partial paths at line  $n_4$  and all the gates in the fanout cone of line  $n_4$  are pruned. If the conventional 3-value algebra is used, the test generation may have to generate all the paths through line  $n_4$  and find there is no test pattern for any of them. Moreover, by looking at the logic values on line  $n_5$ , it can be learned that it is impossible to intentionally make a transition on this line because logic 1 is not achievable, therefore both slow-to-rise and slow-to-fall faults on this line are untestable. Since all the paths through line  $n_4$  must contain line  $n_5$ , it can also be known that both delay faults on line  $n_4$  are untestable. In summary, many faults can be proven untestable by simple analysis, before the test generation is performed, and non-solution search space is more efficiently pruned during the test generation.

Figure 7(b) shows another example. When a partial path reaches gate  $g_2$ , the value of the second vector on side input  $n_3$  is set to logic 0 (non-controlling value, for both robust and non-robust test requirement). Then direct implications are performed backward. The value of the second vector on line  $n_2$  is set to 0 and further direct implications for the first vector can be performed from  $M_2$  (see the previous section). Thus conflicts can be found earlier. If the conventional 3-value logic is used, direct implications stop at gate  $g_1$ . Some conflicts may be hidden.

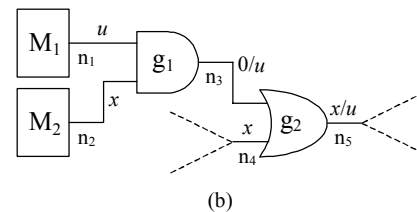
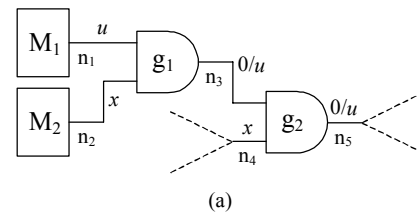


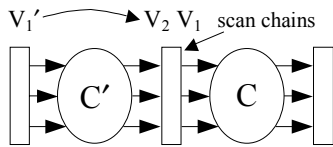
Figure 7. Application of 7-value algebra.

### 3.4. Final Justification

A PODEM [21] based justification process is performed to find a vector pair when a complete path is found. Because most conflicts are eliminated by direct implications, this process is likely to succeed. Since the

two vectors are dependent, whenever a decision (a logic value on any bit in either vector) is made at a primary input or scanned flip-flop, direct implications have to be performed to trim the search space. For the launch-on-shift approach, both vectors can be justified in this way.

For the launch-on-capture approach, because the second vector is derived as the circuit response to the first vector, one time frame expansion is used. In Figure 8, both the circuit and scan chains are duplicated. The first vector  $V_1$  can be generated within one time frame, but since the second vector  $V_2=C(V_1')$ , the goal is to find a satisfying  $V_1'$ . Because  $V_1$  and  $V_1'$  are identical excluding the “don’t care” bits, in the justification process there must be no conflicts between  $V_1$  and  $V_1'$ , i.e. a bit is logic 1 in  $V_1$  but 0 in  $V_1'$  (it is consistent if one of them is a “don’t care”). Similarly, whenever a decision is made on any bit in either vector, direct implications must be performed to keep the logic assignments on any line in the two identical circuits consistent.



**Figure 8. Time frame expansion for final justification using launch-on-capture.**

## 4. Experimental Results

The proposed ATPG has been implemented in Visual C++ and run on Windows 2000 with a 2.6 GHz Pentium 4 processor and 2 GB of memory. Experiments are performed on the full scan versions of the largest ISCAS89 benchmark circuits and two industrial designs, *controller1* and *controller2*, which are partial scan. Muxed scan is used in all the designs. The nominal SDF model is used for computing path delays for *controller2*. The unit delay model is used for the ISCAS89 circuits and *controller1* because SDF models are not available.

### 4.1. Robust Test

Table 1 shows the results for generating the longest robustly-testable path [17] for each fault, under the launch-on-capture and launch-on-shift constraints. It is assumed that at each fault site there are slow-to-rise and slow-to-fall delay faults. The number of faults is twice the number of lines in a circuit, and the same as the number of transition faults. Column 3 shows the upper bound of detectable faults. This number is less than the total number of faults, because it is also assumed that the primary inputs cannot change their logic values, and the primary outputs are masked (not observed), due to the constraints from low-cost ATEs. Therefore no transition can happen at some of the fault sites and some transitions are not observable. Columns 4 and 5 show the number of primary inputs and scan flip-flops for each circuit. Control signals, such as clock and scan enable, are added into the standard ISCAS89 circuits, as primary inputs. It is assumed that the ISCAS89 circuits are full scan and there is only one scan chain for each circuit, in random order. The industrial design *controller1* contains 4 scan chains and *controller2* contains 16 scan chains. Both designs are partial scan. There are 38 non-scanned memory cells in *controller1* and 5 557 in *controller2*. Columns 6-8 show the results for the launch-on-capture approach and columns 9-11 for the launch-on-shift approach. Columns 6 and 9 show the number of paths generated by the ATPG. Before test compaction, each generated path has a test pattern. The number of patterns after compaction is shown in columns 7 and 10. The test patterns are compacted by a simple greedy static compaction algorithm, in which each new pattern is combined with the first compatible existing pattern. Columns 8 and 11 show the CPU time. It can be seen that dealing with uncontrollable signals from non-scanned flip-flops and embedded memories significantly increases the CPU time. However, without using the 7-value algebra, the test generation for *controller1* did not finish within 24 hours and resulted in many more aborts.

**Table 1. Robust test generation summary.**

Circuit	# Lines	UB # Detectable Faults	# Primary Inputs	# Scan Flip-Flops	Launch-on-Capture			Launch-on-Shift		
					# Paths Generated	# Test Patterns	CPU Time (m:s)	# Paths Generated	# Test Patterns	CPU Time (m:s)
s1423	1 423	2 420	20	74	395	215	0:13	666	191	0:07
s1488	1 488	1 310	11	6	192	87	0:01	206	81	0:01
s1494	1 494	1 324	11	6	193	85	0:02	204	79	0:01
s5378	5 378	7 564	38	179	1 799	406	0:07	1 110	94	0:04
s9234	9 234	16 166	39	211	2 376	790	3:59	3 608	681	2:52
s13207	13 207	22 886	65	638	3 220	909	2:25	6 469	1 635	1:03
s15850	15 850	24 338	80	534	2 637	472	2:35	5 828	645	1:08
s35932	35 932	59 246	38	1 728	9 762	36	14:31	12 194	44	8:15
s38417	38 417	74 926	31	1 636	14 905	949	14:21	17 554	655	2:46
s38584	38 584	59 454	41	1 426	9 723	526	11:20	21 047	679	4:28
controller1	86 612	130 692	38	3 503	12 275	2 275	130:10	19 626	657	102:41
controller2	1 966 204	1 815 222	201	57 352	493 779	70 670	132 hrs*	714 116	43 289	57 hrs

\*A commercial ATPG tool took >48 hours for transition fault test generation using launch-on-capture for *controller2*.

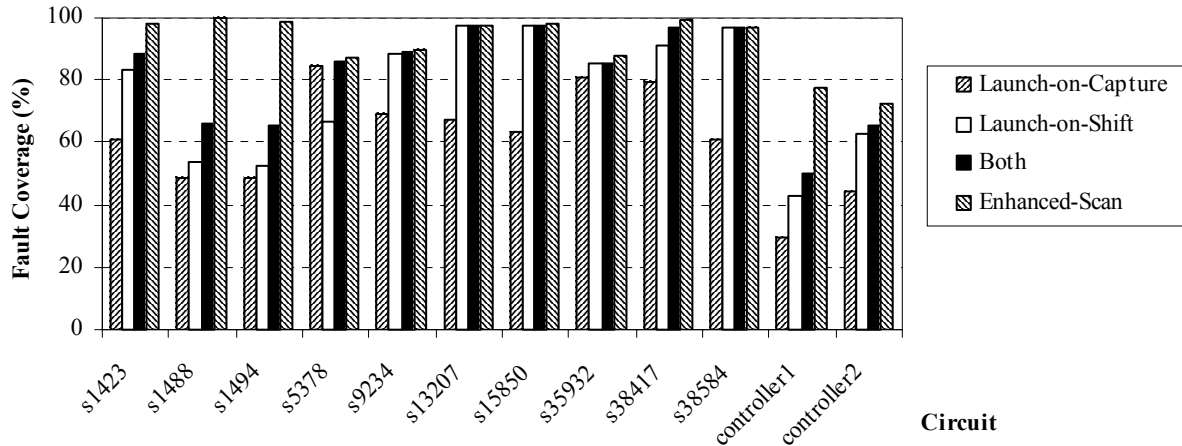


Figure 9. Fault coverage (percentage of robustly tested faults) comparison.

Figure 9 shows the fault coverage (percentage of faults which have at least one path robustly tested) using the launch-on-capture or launch-on-shift approach only, or both. The data in column 3 in Table 1 is used as the total number of detectable faults. For most circuits, the launch-on-shift approach can detect more faults robustly than the launch-on-capture approach, except for circuit s5378, in which the launch-on-capture approach does better. The fault coverage assuming combinational enhanced-scan is shown for comparison. In combinational enhanced-scan, two independent vectors can be stored in the scan chain, so the fault coverage for this method is an upper bound. Again in this mode it is assumed that the primary inputs hold their logic values from the first vector to the second vector, and the primary outputs are masked. Thus the coverage loss is purely due to the launch-on-capture and launch-on-shift constraints and uncontrollable values. Although the faults that the launch-on-capture approach cannot detect must be sequentially redundant in functional mode, the test patterns are still useful because these sequentially redundant faults may cause reliability problems.

Figure 10 is the comparison for robustly testable path length using the launch-on-capture and launch-on-shift approaches, for circuit s15850. The faults are indexed so that the length of the longest testable path for each fault, under the launch-on-capture constraints, is in increasing order. The longest robustly testable path for each fault assuming combinational enhanced-scan is also generated for comparison. Because the primary inputs hold and the primary outputs are masked, some faults have no coverage even if the circuit uses combinational enhanced-scan. For most faults, the maximum path length using the launch-on-shift approach is close to the upper bound, but this is not true for the launch-on-capture approach. All the other circuits have similar plots except for circuit s5378. This phenomenon indicates that the constraints from the launch-

on-capture approach are stronger than the constraints from the launch-on-shift approach for most circuits.

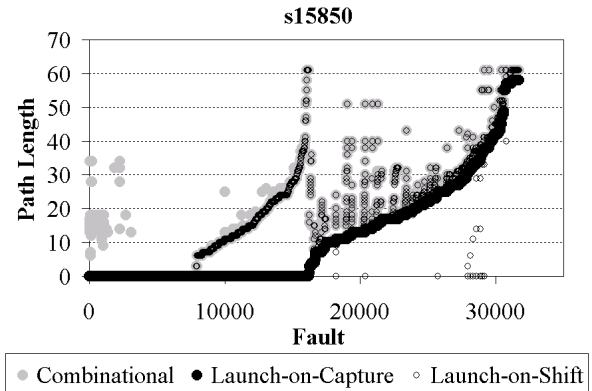


Figure 10. Path length comparison.

#### 4.2. Comparison to Transition Fault Tests

According to the combined delay fault model [22][23], a delay fault can be caused by the combination of a spot defect and process variation. A spot defect can be modeled as a slow-to-rise or slow-to-fall (local delay) fault at a certain site and process variation can cause small distributed delay fault along a path through the spot defect. For a local delay fault, our test strategy is to test the longest paths with a rising or falling transition at the fault site, so the smallest combination of local and distributed delay faults can be detected. In short, the fault space of this delay fault model is the same as that of the transition fault model, but it models smaller delay faults. Process variation can be handled by testing multiple (K) longest paths through each fault site, not only one. However, to keep the test set size comparable to the transition fault test set size, K is set to one in this work, and the resulting test set is termed a KLPG-1 test set.

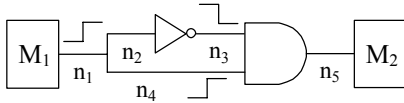
The KLPG-1 test set is constructed as follows: If a fault (slow-to-rise or slow-to-fall) has robustly testable

paths, the longest one is selected, because the delay fault can always be detected regardless of the delay of the other gates or interconnects in the circuit, though this may not be the longest sensitizable path. The results for robust tests are shown in the previous section. In Figure 9 it is shown that the fault coverage for robust test is low using the launch-on-capture approach. This indicates that many faults have no robust test. To construct a test set whose quality is higher than the transition fault test, these faults must be tested.

If a fault does not have a robust test, the longest restricted non-robustly testable path is selected, if it exists. The path selection has the following restrictions:

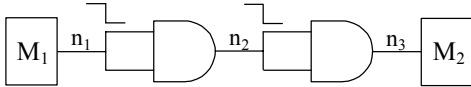
1. The path must be non-robustly testable [18];
2. It must have the required transition at the fault site;
3. The local delay fault must be detected at some capture points, if there is no other delay fault.

In short, if a test is a restricted non-robust test for a fault, it must also be a transition fault test.



**Figure 11. Restricted non-robust test.**

For example, path  $n_1-n_2-n_3-n_5$  is a non-robustly testable path in Figure 11. It is a valid non-robust test for line  $n_2$  and  $n_3$ . However, it is not a valid non-robust test for line  $n_5$  because the glitch (or transition) may not happen if the delay of path  $n_1-n_4$  is greater than the delay of path  $n_1-n_2-n_3$  (violation of restriction 2). Similarly, it is not a valid non-robust test for line  $n_1$  because the slow-to-rise fault may not be detected even if there are no other delay faults (violation of restriction 3).

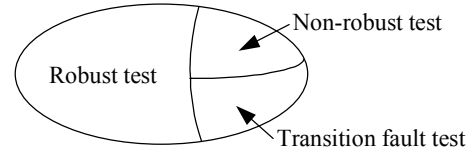


**Figure 12. Transition fault test.**

If a fault has no non-robust test either, a transition fault test which can detect a small local delay fault is generated. In other words, this test has higher quality than the traditional transition fault test because the traditional one assumes large local delay and propagates the fault through any path (usually a short path). In our test generation, this case usually happens when the local delay fault can only be activated or propagated through multiple paths, such as the slow-to-fall fault on line  $n_2$  in Figure 12. The test quality is determined by the length of the shortest paths in the activating or propagating path set. The longer the shortest path is, the smaller the local delay fault that can be detected. The best transition fault test, in terms of the detected local delay fault size, cannot be guaranteed to be generated by our tool but it should be better than the traditional transition fault test. If no high quality test can

be found, a transition fault test is generated. This happens for fewer than 0.05% of the faults in our test cases.

The KLPG-1 test set is composed of these 3 types of path delay fault tests, as shown in Figure 13. It has the same transition fault coverage as the commercial transition fault test set, but has higher quality, in terms of the detected delay fault size.



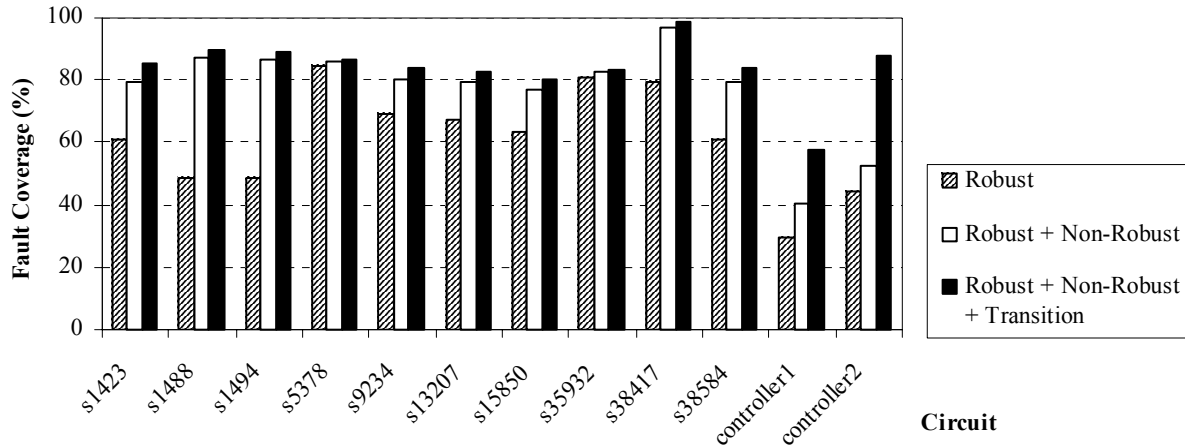
**Figure 13. Test composition.**

Table 2 shows the comparison of test set size (number of test patterns) between KLPG-1 test sets and the commercial transition fault test sets, using the launch-on-capture approach. Columns 2-4 show the number of statically compacted robust/non-robust/ transition fault test patterns in KLPG-1 test sets. Column 5 shows the total number of compacted KLPG-1 patterns. If longest paths are not required, some KLPG-1 patterns can be dropped but the transition fault coverage remains the same. Column 6 shows the number of transition fault effective patterns, which detect unique transition faults, out of the KLPG-1 test sets. The number of transition fault test patterns generated and dynamically compacted by the commercial tool is listed in column 7.

**Table 2. Comparison of test size (Launch-on-capture).**

Circuit	Robust	Non-Robust	Transition	Total	Effective	Commercial
s1423	215	35	12	262	208	95
s1488	87	40	2	129	119	102
s1494	85	41	2	128	116	101
s5378	406	2	2	410	341	194
s9234	790	69	6	865	697	465
s13207	909	32	117	1 058	612	382
s15850	472	31	4	507	397	231
s35932	36	3	1	39	37	68
s38417	949	51	1	1 001	724	365
s38584	526	443	50	1 019	945	528
controller1	2 275	825	311	3 411	2 600	1 900
controller2	70 670	1 856	4 025	76 551	16 284	11 702

For most circuits, the KLPG-1 test sets are 2-3x larger than the commercial transition fault test sets. The KLPG-1 test set for circuit s35932 is smaller, but the KLPG-1 test set for *controller2* is significantly larger. On the other hand, for *controller2*, the number of patterns which have unique transition fault detection is not much larger than the commercial transition fault test set. This phenomenon indicates that many transition faults in *controller2* are easy-to-detect but testing them through the longest paths results in many more necessary assignments and lower compaction rate. This is likely due to the fact that the average path contains 40 gates.



**Figure 14. Robust/non-robust/transition fault coverage of KLPG-1 test sets (launch-on-capture).**

Figure 14 shows the robust/non-robust/transition fault coverage of the KLPG-1 test sets, using launch-on-capture. For example, for circuit s1423, 60.66% of the faults are robustly tested, and 19.01% of the faults are non-robustly only tested. If the launch-on-shift or combinational enhanced-scan approach is used, adding a few transition fault test patterns to the robust test sets results in the same transition fault coverage as the commercial transition fault test sets, for most ISCAS89 circuits. But this is not true for the launch-on-capture approach. The strong constraints from this approach prevent many faults from having robustly testable paths. For the industrial designs *controller1* and *controller2*, robustly detected faults are even fewer due to the constraints from non-scanned memory cells. Figure 14 shows that many more faults are only non-robustly detected or can only be detected through multiple paths (detected by the transition fault test only). For example, in circuit s5378, 354 faults are non-robustly detected but not robustly detected. The number of non-robust test patterns included in the test set is 2. This does not indicate that these 2 test patterns detect the 354 faults, because the 460 robust test patterns also non-robustly detect some other faults by luck. Because the unit delay model is used for the ISCAS89 circuits in the experiments, many paths are of equal length and any of the longest restricted non-robustly testable paths can be the best non-robust test for the fault. Thus, the 2 non-robust test patterns for this circuit can be seen as a cleanup phase.

### 4.3. Compaction

Though KLPG-1 test has higher quality, in some cases a larger test set is not affordable. Then trade-offs between test quality and test size are necessary in compaction.

There are two factors which affect the test quality. One is the number of transition faults which are covered by the test set. The other is the path delay through the transition fault sites [23]. For a fault which has long sensitizable paths, testing a short path is not able to detect a small delay

fault. So a long path whose delay is close to the longest sensitizable path through the fault must be tested. Therefore, the patterns which test long paths are “must keep” patterns. For a fault which has only short paths, relative to the critical paths, the coverage loss can be neglected if one path, not necessarily the longest one, is tested. The reason is that the probability of a delay fault large enough to cause the longest short path to fail but small enough to cause the short path to pass is small. Thus, for these faults, transition fault tests are good enough if a small test size is required. Given a maximum test set size, first the “must keep” patterns, which test long paths, are selected. Then the remaining patterns are selected based on the number of faults they detect. Most of the remaining transition faults can be detected by randomly filling the don’t care bits in the selected patterns.

If the test size is fixed at 6 000 for circuit *controller2*, and all the paths whose delay exceeds 85% of the longest sensitizable path are kept, 3.02% of the non-redundant transition faults are not detected, assuming the commercial tool detects all non-redundant transition faults. This indicates that the transition fault coverage curve must have a long tail. The average care bit density of the compacted test set is 10.2% before randomly filling. Experiments show that if more long paths (with delay >75% of the longest sensitizable path) are kept, 3.05% of the non-redundant transition faults become undetected (an additional 0.03%). Trade-offs between transition fault coverage and path delays can be made accordingly.

## 5. Conclusions and Future Work

We have proposed a KLPG test pattern generation tool for scan-based synchronous sequential circuits, using the launch-on-shift and launch-on-capture at-speed test approaches. The generated test patterns can be applied to the commonly-used scan designs, and at-speed test can be performed using low-cost automatic test equipment. A 7-value algebra has been developed to handle non-scanned

flip-flops and embedded memories whose logic values cannot be initialized by the scan operation. Experiments have shown that this algebra significantly speeds up the test generation procedure.

Experiments have shown that for most circuits, the launch-on-capture approach results in stronger constraints and tighter dependence between the two vectors in a test pattern, than the launch-on-shift approach. The test quality using the launch-on-shift approach is close to the upper bound, in terms of the maximum path delay through each fault site. However, the launch-on-capture approach can eliminate most of the sequentially redundant faults in functional mode [24].

If the launch-on-shift or combinational enhanced-scan approach is used, adding a few transition fault test patterns to the robust test sets results in the same transition fault coverage as the commercial transition fault test sets, for most ISCAS89 circuits. However, due to the strong constraints from the launch-on-capture approach, robust tests alone do not achieve high transition fault coverage. We have constructed KLPG-1 test sets which include the robust/non-robust/transition fault test patterns for each fault and have the same transition fault coverage as the commercial transition fault test sets. Since KLPG-1 test sets test long paths through each fault site, smaller delay faults must be detected, compared to the commercial transition fault tests. We are currently evaluating our test sets on real chips to quantify this benefit.

This work is being extended to handle circuits which have more complicated timing features, such as clock gating and multi-cycle paths.

## Acknowledgements

This research was funded by the Semiconductor Research Corporation under contract 2000-TJ-844 and the National Science Foundation under contract CCR-0098329.

## References

- [1] Z. Barzilai and B. K. Rosen, "Comparison of AC Self-Testing Procedures," *IEEE Int'l Test Conf.*, Philadelphia, PA, Oct. 1983, pp. 89-94.
- [2] B. R. Benware, R. Madge, C. Lu and R. Daasch, "Effectiveness Comparisons of Outlier Screening Methods for Frequency Dependent Defects on Complex ASICs," *IEEE VLSI Test Symp.*, Apr. 2003, pp. 39-46.
- [3] R. R. Montanes and J. P. Gyvez, "Resistance Characterization for Weak Open Defects," *IEEE Design & Test of Computers*, vol. 19, no. 5, Sept. 2002, pp. 18-25.
- [4] J. Waicukauski, E. Lindbloom, B. K. Rosen and V. S. Iyengar, "Transition Fault Simulation," *IEEE Design & Test of Computers*, vol. 4, no. 5, Apr. 1987, pp. 32-38.
- [5] K. T. Cheng, S. Devadas and K. Keutzer, "Delay-Fault Test Generation and Synthesis for Testability Under a Standard Scan Design Methodology," *IEEE Trans. on Computer-Aided Design*, vol. 12, no. 8, Aug. 1993, pp. 1217-1231.
- [6] K. T. Cheng, "Transition Fault Testing for Sequential Circuits," *IEEE Trans. on Computer-Aided Design*, vol. 12, no. 12, Dec. 1993, pp. 1971-1983.
- [7] K. T. Cheng, "Test Generation for Delay Faults in Non-Scan and Partial Scan Sequential Circuits," *IEEE/ACM Int'l Conf. on Computer Aided Design*, Santa Clara, CA, Nov. 1992, pp. 554-559.
- [8] M. Sharma and J. H. Patel, "Finding a Small Set of Longest Testable Paths that Cover Every Gate," *IEEE Int'l Test Conf.*, Baltimore, MD, Oct. 2002, pp. 974-982.
- [9] W. Qiu and D. M. H. Walker, "An Efficient Algorithm for Finding the K Longest Testable Paths Through Each Gate in a Combinational Circuit," *IEEE Int'l Test Conf.*, Charlotte, NC, Sept. 2003, pp. 592-601.
- [10] T. J. Chakraborty and V. D. Agrawal, "Effective Path Selection for Delay Fault Testing of Sequential Circuits," *IEEE Int'l Test Conf.*, Washington, DC, Nov. 1997, pp. 998-1003.
- [11] M. P. Kusko, B. J. Robbins, T. J. Koprowski and W. V. Huott, "99% AC Test Coverage Using Only LBIST on the 1GHz IBM S/390 zSeries 900 Microprocessor," *IEEE Int'l Test Conf.*, Baltimore, MD, Oct. 2001, pp. 586-592.
- [12] J. Saxena, K. M. Butler, J. Gatt, R. Raghuraman, S. P. Kumar, S. Basu, D. J. Campbell and J. Berech, "Scan-Based Transition Testing - Implementation and Low Cost Test Challenges," *IEEE Int'l Test Conf.*, Baltimore, MD, Oct. 2002, pp. 1120-1129.
- [13] J. Savir, "Skewed-Load Transition Test: Part I, Calculus," *IEEE Int'l Test Conf.*, Baltimore, MD, Sept. 1992, pp. 705-713.
- [14] S. Patel and J. Savir, "Skewed-Load Transition Test: Part II, Coverage," *IEEE Int'l Test Conf.*, Baltimore, MD, Sept. 1992, pp. 714-722.
- [15] J. Savir and S. Patel, "Broad-Side Delay Test," *IEEE Trans. on Computer-Aided Design*, vol. 13, no. 8, Aug. 1994, pp. 1057-1064.
- [16] J. Benkoski, E. V. Meersch, L. J. M. Claesen and H. D. Man, "Timing Verification Using Statically Sensitizable Paths," *IEEE Trans. on Computer-Aided Design*, vol. 9, no. 10, Oct. 1990, pp. 1073-1084.
- [17] C. J. Lin and S. M. Reddy, "On Delay Fault Testing in Logic Circuits," *IEEE Trans. on Computer-Aided Design*, vol. 6, no. 9, Sept. 1987, pp. 694-701.
- [18] G. L. Smith, "Model for Delay Faults Based Upon Paths," *IEEE Int'l Test Conf.*, Philadelphia, PA, Oct. 1985, pp. 342-349.
- [19] W. T. Cheng and T. J. Chakraborty, "Gentest: An Automatic Test Generation System for Sequential Circuits," *IEEE Computer*, vol. 22, no. 4, Apr. 1989, pp. 43-49.
- [20] T. M. Niermann and J. H. Patel, "HITEC: A Test Generation Package for Sequential Circuits," *ACM/IEEE European Design Automation Conf.*, Amsterdam, The Netherlands, Feb. 1991, pp. 214-218.
- [21] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Trans. on Computers*, vol. C-30, no. 3, Mar. 1981, pp.215-222.
- [22] W. Qiu, X. Lu, Z. Li, D. M. H. Walker and W. Shi, "CodSim - A Combined Delay Fault Simulator," *IEEE Int'l Symp. on Defect and Fault Tolerance in VLSI Systems*, Boston, MA, Nov. 2003, pp. 79-86.
- [23] W. Qiu, X. Lu, J. Wang, Z. Li, D. M. H. Walker and W. Shi, "A Statistical Fault Coverage Metric for Realistic Path Delay Faults," *IEEE VLSI Test Symp.*, Napa Valley, CA, Apr. 2004, pp. 37-42.
- [24] X. Liu and M. S. Hsiao, "Constrained ATPG for Broadside Transition Testing," *IEEE Int'l Symp. on Defect and Fault Tolerance in VLSI Systems*, Boston, MA, Nov. 2003, pp. 175-182.