

# Defect Coverage Analysis of Partitioned Testing

Sreejit Chakravarty, Eric W. Savage and Eric N. Tran

Intel Corporation, Santa Clara, USA

## Abstract

Research in improving test quality has focused on identifying better fault models and coverage metrics and tools to achieve high coverage. The test generation and test application methodology is usually not considered. We attempt to understand the implication of a test generation and test application methodology, viz. partitioned testing, on product quality. In partitioned testing, patterns are applied to one part of the design while the other parts are maintained in a quiescent state.

Quantitative data on several aspects of partitioned testing, using some industrial test cases, are presented. It highlights the need for generating patterns using different partition sizes, generating long test sequences and the need to include functional testing in the test suite. In addition, we argue that a different metric is needed to evaluate functional pattern quality to cover the gap identified.

## 1. INTRODUCTION

Research in improving the quality of the test process has focused on identifying better fault models and coverage metrics based on those fault models. References [1] [10] is a partial list of numerous works published in this area. Better test generation methods to achieve these coverage targets has also been researched and discussed. An important issue in the domain of test quality that has not been researched is the impact of the test generation and test application methodology. Test generation methodology includes the kind of restrictions we generally specify to the tool, how we partition the design to ensure that the design is within the constraints imposed by the capacity of the ATPG tools, memory modeling, length of the vector sequence, etc. Test application refers to the mode used during test application like holding the predefined memory in a predetermined state, etc. We believe that test generation and test application also has a profound impact on product quality. In this paper we present an investigation of this problem.

We will be focusing on a particular test application methodology, and the associated test generation process, called partitioned testing. Consider Figure 1 where a design is shown to consist of four logical partitions. Partition A could be the on-chip cache while the other three partitions could be three partitions comprising the logic portion of the design. In another

scenario, these could be four embedded processors interconnected together.

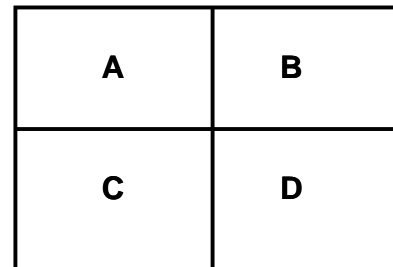


Figure 1. Illustration of Partitioned Testing

Partitioned testing is the process of testing one part of the design while the other parts are in a quiescent mode. In Figure 1, assume A, B, C and D to be four processors. In partitioned testing, one processor executes the functional test while the other three, potentially in a known state, are idle. As a second example, in Figure 1, assume A, B, C and D to be four partitions of a design. Assume each part has been scanned. Patterns are generated for one partition at a time. During test application of patterns for A, partitions B, C and D are held in a known state. In a clean design, the interface signals between partition A and the other partitions are set to known values during test application.

The use of partitioned testing is on the rise due to several reasons. The first of these is the constraint imposed by the design methodology. In SOC designs individual IP blocks and glue-logic are integrated to complete a design. In one of the most common scenarios, individual IP blocks have embedded in them their scan system and the associated scan patterns. Patterns for the glue logic are generated and partitioned testing is used. This could be due to pin limitations, power constraints etc.

In large designs, ATPG tool capacity could be the motivation for partitioning the design and using partitioned scan ATPG. Each partition is then tested individually. Designs using BIST typically partition the design and LFSRs apply patterns to individual partitions and compress the responses. For power reasons, only a handful of these partitions can be exercised at a time. In both these cases we are again using partitioned testing.

The current trend of increasing IP-block integration, larger monolithic designs, multi-processor designs, lagging ATPG tool capacity, power limitations during test, etc. implies that the use of partitioned testing will be on the rise. Given this, it is important to understand the limitations of this test methodology, identify the gaps and explore some potential approaches to cover these gaps. In this paper we present a case study that addresses the above question. The case study is based on test cases derived from one of our current micro-processor. We assume scan based partitioned testing, which is one of the most common scenarios in ASIC designs.

The main findings of this study can be summarized as follows.

- Scan test quality can be improved by generating and applying tests using larger blocks in that these tests have unique defect coverage. At the same time, patterns generated using smaller partitions also have unique coverage. The gain in test quality using larger partitions comes with a price. We are hitting the limits of the tool capacity faster than we anticipate. Even if we circumvent this limitation, there is a price to pay in terms of test data volume. These results highlight the need for a hybrid approach wherein patterns are generated and possibly applied using partitions of different sizes.
- The study identified the need for multi-cycle scan patterns since many of the faults that were left uncovered could not be detected by single cycle scan tests. This result may not be interesting for designs that are full scan but is important for high performance designs which are not typically full-scan. From a manufacturing perspective this is disconcerting since multi-cycle patterns tend to be more unstable in production. This result points to the need to pay special attention to generating multi-cycle scan patterns and making such patterns more production worthy.
- We analyzed the scan un-testable faults and show that even if we attain very high coverage using scan there remain potential quality gaps. This gap, which does not necessarily consist of uncovered at-speed failure defects, can only be covered by functional patterns. However, the conventional method of evaluating the functional patterns cannot identify these quality gaps. This issue needs further investigation.

The rest of the paper is organized as follows. In Section 2, we define the term defect coverage as used in this paper along with a rationale for using these coverage metrics. In Section 3 we discuss details of the case studies and the experimental setup. In this paper we refer to the larger chunk of the design as a cluster and each of the individual parts of a cluster as a partition. Accordingly we coin the terms partition-s@patterns, cluster-s@patterns, to refer to the s@patterns for a partition, s@patterns for the cluster respectively. In Section 4 we present the coverage data for partition-s@patterns and cluster-s@patterns. An

analysis of this data is presented in Section 5. In Section 6 we present the comparative data on test data volume for partition and cluster patterns. These data highlight the advantages and disadvantages of increasingly fragmenting the design for the purpose of testing and motivates the need for using partitions of varying sizes for test generation and, possibly, test application. In Section 7 we present data that point to the need for multi-cycle scan tests. Data pointing to the need for functional testing is discussed in Section 8. In this section we also discuss the limitations of the conventional method for evaluating functional patterns and propose some alternatives. We end with a brief summary.

## 2. DEFINITION OF DEFECT COVERAGE

We distinguish between two kinds of defects: **single-net defects** and **multi-net defects**. Single-net defects impact only one net in the design. Examples of this kind of defects include shorts to V<sub>CC</sub> or GND. Via failure is also an example of a single-net defect. On the other hand multi-net defects affect multiple nets. Bridges between signal nets is an example of such a defect. Cross-coupling between nets causes failures and for this discussion we categorize them as multi-net defects.

The move to copper processes implies that via failures, affecting only one net will increase [6]. This will increase the percentage of nodes causing single-net failures vis-à-vis bridges. At the same time, shrinking geometries, the use of multi-stacking, etc. will increase the probability of multi-node defects. Both these will increase both the probability of random defect induced failure mechanism like bridges. They will also increase the capacitive and coupling effects, between adjacent blocks, that lead to failures. This indicates that both single-net and multi-net defects will continue to be important. Therefore, in our study, we consider both single-net and multi-net defects.

Both single-net and multi-net defects can cause hard (sometimes referred to as static or slow-speed) failures. Both these defect types can also cause soft (sometimes referred to as dynamic or speed) failures. By some estimates, hard defects constitute a disproportionately large percentage (98%) of all defects [1]. It is likely that soft failures will increase in proportion but hard defects will continue to be dominant in the near future. Given this, in this study, we concentrate only on hard defects. A better study would have included soft metrics that measure soft failures – primarily the capacitive coupling type.

To simulate defect coverage we use approximate fault models. Stuck-at is a popular fault model that models single-net hard defects. In this study we will use stuck-at to model single net defects. We model multi-node hard defects by bridge fault models for two node bridges. A two-node bridge  $\langle a, b \rangle$  between signal nets a, b will be modeled by the 4-way bridge fault model  $\{1\}: \{as-a-0, b=0\}, \{as-a-1, b=1\}, \{bs-a-0, a=0\}$  and  $\{bs-a-1, a=1\}$ . The fault  $\{as-a-0, b=0\}$  implies that

the bridge is equivalent to a 0 provided the number of bridges is 0. The other three faults are similarly defined.

The target list of bridges, from which we derive the bridge fault models, is extracted by analyzing the design layout. The tool used implements inductive fault analysis [3] and has been discussed in [2] [7] [8] [9]. This fault list goes through additional refinement. In Figure 2 we show the % distribution of bridge defects to power (V<sub>CC</sub>, GND), clock lines and between signal lines. The latter is referred to as signal-signal bridges. This data is for one of our micro-processor products [1] and the distribution is given on a per-partition basis. Similar trends were discussed in [5].

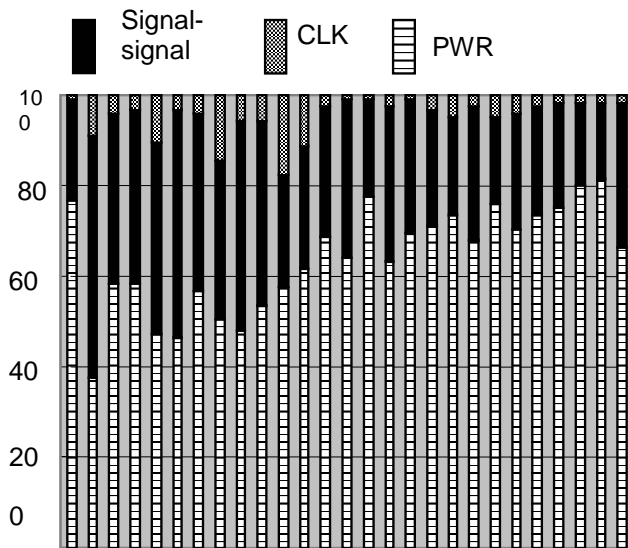


Figure 2. Example Breakdown of Signal-signal, Clock and Power bridge distribution

Bridges to power affect only one net and are treated as single-net defects. Assuming that clock drivers are usually very strong, bridges to clock lines are also treated as single-net defects. It is assumed that these two defect categories are covered by the s@ fault model for single-net defects. So, before generating the bridge fault list, the list of extracted bridges is filtered to remove all bridges to clock lines and power rails.

In our study we also break up the signal-signal bridge fault list into two categories: **likely** and **equi-probable** bridges. These two categories of bridges are defined as follows. Weighted Critical Area (WCA) [3], which is the basis for inductive fault analysis, is a measure of the likelihood of the occurrence of a bridge. It is calculated based on the defect profile of the underlying manufacturing process.

The two separate bridge fault lists are generated using cumulative WCA which is calculated as follows. First the list of extracted bridges is ranked with respect to their WCA. Let  $B_1, B_2, B_3, \dots, B_N$  be the ranked list of the bridge. Furthermore, let  $W_j$  be the WCA for bridge  $B_j$ . The cumulative WCA for  $B_1$  is  $W_1$ . The cumulative WCA for the first two faults  $B_1, B_2$  is  $W_1 + W_2$ ; the cumulative WCA for the first three faults  $B_1, B_2,$

$B_3$  is  $W_1 + W_2 + W_3$ ; and so on... Figure 3 is a typical plot of the cumulative WCA versus the number of bridges.

Note that the cumulative WCA initially grows very quickly and then grows at a very slow rate indicating a saturation of the cumulative WCA. The point at which the cumulative WCA saturates is referred to as the **knee of the WCA curve**. We define the knee of the curve to be the point at which the rate of increase of the cumulative WCA falls below  $1/N$  where  $N$  is the total number of bridges under consideration. The list of bridges prior to the knee is referred to as **likely-bridges**. The list of bridges after the knee is referred to as **equi-probable-bridges**.

Typically, the set of likely-bridges consists of a small percentage of the total number of bridges and accounts for a reasonably large percentage of the total WCA. The equi-probable-bridges consist of a large number of bridges each with a small WCA. However, the total WCA contribution of the equi-probable-bridges is usually quite significant. The bridge faults implied by the likely-bridges and equi-probable-bridges are referred to respectively as **likely-bridge faults** and **equi-probable-bridge faults**. We will also use the terms **pre-knee** and **post-knee** bridge faults respectively to refer to the likely and equi-probable bridge faults.

What we have just described are coverage based on layout analysis. In [10] a different metric, known as **bridge coverage estimate (BCE)** was proposed. BCE estimates the coverage of all possible bridges based on the number of times a stuck-at fault has been detected. It is defined as follows:

$$BCE = \sum_{i=1}^{\infty} \frac{f_i}{|F|} (1 - 2^{-i}),$$

where  $f_i$  is the set of stuck-at faults detected exactly  $i$  times and  $F$  is the set of all stuck-at faults.

Silicon data was presented to show that increasing the BCE coverage increases the fallout obtained from the test set.

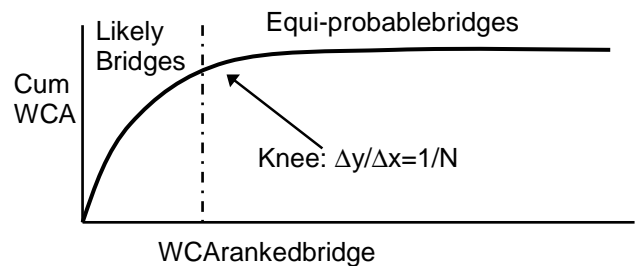


Figure 3. Example cumulative WCA plot

In our analysis, defect coverage will be gauged by four coverage metrics: s@, likely-bridge (pre-knee) and equi-probable-bridge (post-knee) and BCE coverage. The first is an approximation for single-net defect coverage where as the latter three are an approximation, respectively, for likely, equi-probable and broad multi-net defect coverage.

### 3. CASE STUDIES AND EXPERIMENTAL SETUP

In the experiment conducted we restrict ourselves to scan testing. We used three test cases named ExA, ExB and ExC from one of our latest Microprocessor. Some relevant details of these test cases are shown in Table 1, Table 2 and Table 3. Cluster ExA (respectively, ExB, ExC) consists of two partitions named A1, A2 (respectively, B1, B2; and C1, C2). Row "SIZE" gives the total of ATPG model gate count in millions, for each of the constituent partitions as well as the size of the cluster.

**Table1.** TestcaseA

EXA	A1	A2	Clust	Incr
s@	0.512	0.287	0.800	0.125
Size	1.78	0.383	2.17	

**Table2.** TestcaseB

ExB	B1	B2	Clust	Incr
s@	0.29	0.16	0.45	0.22
Size	0.28	0.53	0.81	

**Table3.** TestCaseC

ExC	C1	C2	Cluste	Incr
s@	0.357	0.355	0.735	3.13%
Size	0.480	0.602	1.215	

Row "s@ Faults" gives the total number of s@ faults for each partition and the cluster. The unit used is millions. Thus, column A1 of row "s@ Faults" states that the A1 partition of ExA has 0.512 million stuck-at faults. Column Incr gives the additional number of s@ faults that are present in the cluster s@ fault list but not in any of the partition fault lists. It is important to note here that the extra faults in the cluster model can potentially be mapped to one or more faults in the partition level model. However, these mapped faults are either not controllable or observable from the scan present in the partition itself and therefore not counted in the partition s@ fault list. We refer to them as **border-s@-faults**. The percentage calculation is with respect to the total cluster fault list.

The test cases have a generous amount of embedded memories in them and they are not fully scanned. Consequently, although the percentage of scan is pretty high, there are parts of the design that are not scan testable. The memories do not have any DFT around them and are blank-

boxed in the model. Constraints to avoid contention were also used in the ATPG process. Our un-testability analysis takes these into consideration. We will revisit this issue later on while analyzing the untestability data.

### 4. EXPERIMENTAL DATA

We first investigate the quality of cluster-s@ patterns and partition-s@ patterns. The first pattern set was generated using the cluster level ATPG model. The second pattern set was generated using the partition ATPG models of the constituent partitions and merged into one test set. The faults targeted during the ATPG process were the s@ faults in the corresponding models. Given that we are comparing scan patterns we should be careful in generalizing these results for other variations of partitioned testing. These two pattern sets were then simulated on a number of cluster level fault lists. In this section we tabulate the coverage data.

In Table 4, data on s@ fault coverage of cluster-s@ and partition-s@ patterns is presented. Column *Total Faults* is the total number of s@ faults in the fault list derived from the cluster level model. *Untestable Faults* includes s@ faults identified to be scan-untestable using structural analysis. We also included faults that were identified by the ATPG tool to be redundant while attempting to generate test forms, with the given ATPG constraints. The percentage number in column *Untestable Faults* is calculated as a percentage of the entire clusters@ fault list.

**Table4.** S@ coverage data of Partition and Clusters@-patterns

S@	Total Faults	Unt Faults	Both	Part Only	Cluster Only
ExA	800544	107919	598137	8359	27162
		13.5%	86.35%	1.2%	3.9%
ExB	453445	14302	435580	987	1346
		3.15%	99.12%	0.31%	0.23%
ExC	734846	203570	511110	7189	8201
		27.7%	96.20%	1.35%	1.54%

The partition-s@ patterns was generated by running the ATPG tools on the individual partitions A1, A2, B1, B2, C1 and C2 respectively. In the experiment, we combined the s@-patterns for A1, and A2 (respectively, B1 and B2; as well as C1, C2) and called it the partition patterns for cluster ExA (respectively, cluster ExB, ExC). The *cluster pattern* for a cluster is the set of patterns obtained by running the ATPG tool on the cluster model.

**Table5.** Realistic Bridge Coverage of cluster-s@ and partition-s@ patterns

	Total Faults	UnFaults	Detected		
			Both	Cluster Only	Part Only
<b>PRE-KNEEBRIDGES</b>					
ExA	5372410	3614602	1285638 (73.14%)	27815 (1.58%)	26123 (1.49%)
ExB	3047860	1204904	1639350 (88.95%)	28069 (1.52%)	20140 (1.09%)
ExC	4345700	2241378	1760789 (83.67%)	38868 (1.85%)	41451 (1.97%)
<b>POST-KNEEBRIDGES</b>					
ExA	4M	3099957 3211515	631874 (80.14%)	12977 (1.65%)	11679 (1.48%)
ExB	4M	1615282	2222746 (93.20%)	36520 (1.53%)	24157 (1.01%)
ExC	4M	2215588	1636557 (91.71%)	36520 (1.99%)	24157 (1.01%)

Column *Both* shows the total number of faults that are detected by both the cluster patterns and the partition patterns for each of the test cases. The percentage numbers shown were calculated by removing the un-testable faults from the total number of faults. Column *Part Only* shows the total number (percentage) of faults detected by the partition tests but not by the cluster test for the given cluster. Similarly, *Cluster Only* shows the total number (percentage) of faults detected by the cluster test but not the partition test for the given cluster.

In Table 5 we present the coverage of the two sets of patterns on the set of realistic bridges. This is broken down into two parts: the pre-knee bridges and the post-knee bridges. The entry in column *Total Faults* represents the number of bridge faults targeted in the simulation. For pre-knee this includes all bridges in the pre-knee region. A sample of 4M post-knee bridge faults was used in our study.

Column *Untestable Faults* lists the number of untestable faults. Untestable faults were identified in two ways. Using structural analysis, bridge faults that cannot be targeted using the available scan were marked as untestable. It also includes bridge faults identified to be untestable during a bridge ATPG run. The next column, labeled *Both*, is the number of bridge faults that were detected by both the cluster-s@ and partition-s@ patterns. Columns *Cluster Only*, *Partition Only* shows respectively the number of faults detected by the cluster-s@ patterns but not the partition-s@ patterns and the number of faults detected by the partition-s@ patterns but not the cluster-s@ patterns. The % numbers within brackets in the last

three columns are the percentage of the testable faults in the respective categories.

**Table 6.** BCE of cluster-s@ and partition-s@ patterns

BCE Cov	Both	Cluster Only	Part Only	Incr CL	Incr Part
ExA	75.57	73.88	71.31	4.26	1.69
ExB	96.21	93.93	95.09	1.12	2.28
ExC	69.83	66.94	68.24	1.59	2.89

In Table 6 we list the BCE coverage for the different pattern sets. Column *Both* shows the BCE coverage for the combination of the partition-s@ and cluster-s@ patterns. Column *Cluster Only* shows the BCE coverage for the cluster-s@ patterns only. Thus, the added value provided by the partition-s@ patterns is the difference between the *Both* column and the entry in column *Both* which is shown in column *Incr Part*. Analogously, column *Part Only* shows the BCE coverage of only the partition-s@ patterns. The difference of the entry in this column with the entry in column *Both* gives the incremental value of the cluster-s@ patterns and is shown in column *Incr CL*.

## 5. COVERAGE ANALYSIS OF CLUSTER AND PARTITION PATTERNS

An analysis of the data in Table 4-Table 6 is presented. This leads to the following observations.

- Cluster patterns often have significant unique coverage. For example, for ExA, the incremental s@, incremental pre-knee bridge, incremental post-knee bridge and incremental BCE coverage are respectively 3.9%, 1.58%, 1.65% and 4.26% respectively. In some cases, like ExB even if the incremental s@ coverage is minimal (0.23%) the incremental pre-knee, post-knee and BCE coverage are much larger being, respectively, 1.52%, 1.53% and 1.12%.
- These second observation is that increasing the partition size does not guarantee all the possible goodness in the quality of the patterns. This is exemplified by the unique coverage of the partition patterns of ExA, which respectively has unique s@, pre-knee, post-knee and BCE coverage of 1.2%, 1.49%, 1.48% and 1.69%.

The above data is somewhat counter intuitive in that, theoretically, any fault that has a test in the partition model is guaranteed to have a test in the cluster model. But the ATPG tools, which use various heuristics, cannot always find these tests when the larger cluster models are used.

- There are also a number of data points that show the total coverage of the partition patterns to be higher than the total coverage of the cluster patterns. Notable one is the higher BCE coverage, by approx. 1%, of the partition patterns for ExB and ExC. From Table 6, the total BCE

coverage of ExB (respectively, ExC) partition patterns is 97.37 (respectively, 71.13%) whereas for the cluster patterns the coverage is 96.12% (respectively, 69.83%).

## 6. A HYBRID PATTERN GENERATION METHODOLOGY

We have so far considered only the coverage aspect. We next look at the impact it has on test data volume, assuming no compression is used. This therefore also translates to impact on test time.

**Error! Not a valid bookmark self-reference.** shows data comparing the number of bits of storage required to store these patterns. It was assumed that the resulting patterns are fully specified and all the bits need to be stored in the tester. For each example we list the scan chain length in the model (cluster or partition) (column *Scan Chain Length*) and the number of scan patterns (column *No of Scan Loads*). The next column (*Total number of Bits of Storage*) is the product of the previous two columns. The last column is the ratio of the number of bits required to store the cluster-s@patterns versus the combined number of bits to store all the partition-s@patterns.

**Table 7.** Test Data Volume of cluster-s@ and partition-s@ patterns

	Scan Chain Length	No of Scan Loads	Total Number of Bits of Storage	Increase
ExA	24339	3869	94167591	1.98
A1	12476	1578	19687128	
A2	11863	2339	27747557	
ExB	27173	1253	34047769	1.96
B1	8957	645	5777265	
B2	18216	637	11603592	
ExC	32056	1421	45551576	1.44
C1	15761	926	14594686	
C2	16295	1048	17077160	

Table 7 implies that we pay a price when we increase the partition size for ATPG. For ExB, ExC there is a marginal difference in the s@ coverage of the partition and the cluster patterns. However, there is quite an increase in the size of the cluster patterns, being respectively 1.96X and 1.44X larger. For ExA, the increase in the test data volume, it could be argued, is due to the increase in the s@ coverage.

Given this and the coverage data in the previous section it makes sense to consider a hybrid approach. The hybrid approach first generates patterns using the smaller partitions and then adds to that incrementally the cluster patterns, after

dropping detected faults and generating patterns for the remaining ones. This could be done as we traverse up the design hierarchy and is limited only by the tool capacity. In this approach the goodness of both the smaller partition size and the larger partition size are incorporated into the quality of the patterns without paying unduly for the increase in the test data volume. We have not performed this study and leave it to the interested reader. This would also affect the test application methodology. Such patterns can be applied using the larger partition sizes but, for test application time, can be applied in multiple modes with different partition sizes.

## 7. NEED FOR MULTI-CYCLE SCAN TESTS

From Table 4 and Table 5 we note that a large number of both s@ and bridge faults are untestable. A sizeable part of those faults are untestable because they are neither controllable or observable by scan patterns. However there are faults in that category that are controllable and observable using scan but have been declared to be untestable. All these faults were removed from our coverage computation and therefore hide the possibility that further investigation could improve the defect detection capability of scan patterns.

**Table 8.** Multi-cycle Un-testability Data

	s@	Pre-Knee	Post-Knee
ExA	1.26	7.78	6.75
ExB	0.71	3.21	3.10
ExC	3.40	1.40	1.03

In Table 8, we tabulate the result for one such investigation. In Table 8, the untestable faults for which single cycle scan tests do not exist, although they are scan reachable, for each fault model are shown. These are faults that cannot be detected by single cycle scan tests and is a consequence of the design being partial scan. All entries in Table 9 are percentage of the total number of faults in the fault-list before removing any untestable faults. For ExC, a reasonably large percentage of s@ faults requires such patterns. For ExA and ExB, the numbers are large for both the pre-knee and post-knee bridges.

This data is significant because multi-cycle scan patterns tend to have robustness issue in high volume manufacturing testing and are generally avoided. The data shows that this practice is not advisable since we sacrifice a sizeable amount of test quality in the process. In addition, coverage calculations should consider these faults as testable faults in order to reflect this gap.

## 8. NEED FOR A BETTER EVALUATION MODEL FOR FUNCTIONAL PATTERNS

From Table 5 we note that over 50% of the extracted bridges are scan ATPG untestable! In Table 9 we summarize the untestable fault data for pre-knee and post-knee bridges for the three test cases. All data is presented as a percentage of the

total bridge faults extracted for the partition. Column CUI, CUO and DNG represents the number of faults that are not controllable, observable or involves a dangling node respectively. So, although these are scan untestable they are by no means benign faults and the number of faults in each of these categories is very high.

Since these are probable defects they have to be covered using at least one of the pattern classes in the HVM pattern suite. Using functional patterns is the obvious answer and most products add functional patterns to cover such defects. Unfortunately, functional patterns are often not evaluated as to their effectiveness. Even when they are evaluated they are evaluated against the s@ or transition fault model. These coverage metrics do not highlight the coverage of the gaps for multi-node defects. Note that using multiple defects using scan patterns, as proposed in the n-detect paradigm, does not lead to any better coverage for these faults since they are scan untestable.

**Table9.** UntestableBridgeData

		CUI	CUO	DNG
ExA	Pre-Knee	4.10%	58.57%	3.23%
	Post-Knee	2.75%	74.49%	0.40%
ExB	Pre-Knee	7.40%	24.34%	6.20%
	Post-Knee	8.65%	22.47%	8.57%
ExC	Pre-Knee	6.75%	37.61%	6.10%
	Post-Knee	6.67%	40.93%	6.88%

We investigated the untestable data further. The first of these was to identify what percentage of these scan untestable faults involved the scan system itself. The results are shown in Table 10. Once again the percentages are in terms of the total number of faults in the fault list. Note that the number of such faults is pretty large and is representative of a problem that arises when we have a large number of global test control lines. This scenario will also be true in the case of SOC designs where a large number of IP blocks are glued together by glue logic. The SOC community has focused on testing the glue logic but the interaction of the global lines with the individual IP blocks has been ignored. We typically do not analyze the coverage of such defects either for scan patterns or functional patterns.

**Table10.** UntestableScanBridgeData

		SCANCUI	SCANCUIO
ExA	Pre-Knee	0.3%	3.8%
	Post-Knee	0%	2.98%
ExB	Pre-Knee	0.5%	1.87%
	Post-Knee	0.3%	1.5%

ExC	Pre-Knee	2.83%	8.71%
	Post-Knee	2.40%	10.50%

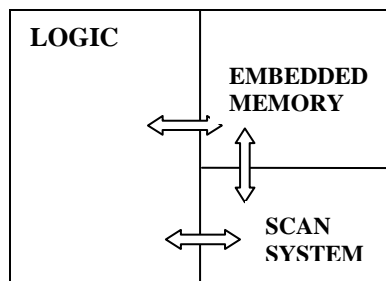
There is another category of defects that are ignored when we evaluate scan patterns and functional patterns. This includes defects arising from the interaction of the logic with embedded memories.

**Table11.** UntestableMemoryBridgeData

		MEMCUI	MEMCUO
ExA	Pre-Knee	0%	3.14%
	Post-Knee	0%	5.49%
ExB	Pre-Knee	2.27%	2.13%
	Post-Knee	4.18%	1.91%
ExC	Pre-Knee	1.07%	1.50%
	Post-Knee	1.15%	4.00%

We tabulate the percentage of bridges which involve at least one line in the embedded memories in Table 11. Once again we note that a sizeable percentage of defects involve a bridge between memory and logic. An important point to be noted is that most of these are unobservable. This is because the fault effector originates in the node that is part of the memory which is not observable in the scan mode.

All these point to a limitation of using simple, single-node fault models for evaluating functional patterns. It is often believed that much of the failures detected by the functional patterns, above and beyond those detected by the scan patterns, are speed failures. This data shows that there is a possibility that the value of functional tests might also lie in detecting defects that arise out of an interaction between various "partitions" in the design. One such scenario is shown in Figure 4 where in a chunk of logic is shown to be divided into three sub-parts viz. the pure logic, the embedded memories and the scan sub-system.



**Figure4.** Possible interaction between various partitions

It is generally assumed that the chain tests will identify defects in the scan system. However, such tests are not guaranteed to detect interactions between the logic and the embedded memories. They are collaterally detected by

functional patterns but while evaluating the functional patterns the scan system is often not even modeled.

A similar scenario exists for the embedded memories. They are often tested using direct-access test patterns but such patterns do not target defects that arise as a result of the interaction of the embedded memories and the logic for that matter the scan system.

One could take this a step further. In multi-core system or when multiple-IP blocks are integrated together the interaction of the head joining blocks are never factored into the evaluation of the functional patterns.

We presented data that shows that these interactions do represent a large percentage of the defects. We therefore need a coverage model for functional tests that identifies such gaps. The bridge model could be one such model but any other variant that appropriately captures the goodness of covering such interactions could also be investigated.

## SUMMARY

This paper is an attempt at broadening the scope of research into better test quality. In addition to identifying good coverage metric, we believe that test generation and test application methodology plays a significant role in test quality. This is often ignored.

There is a growing trend to fragment the design during test generation and test application. We studied the limitations of fragmenting the design for the purpose of test generation and test application. A model to evaluate the quality and effectiveness of such a testing scenario was proposed. We showed, through a case study, that increasing the partition size does lead to unique detects. However, the smaller partition size also contributes unique detects. There is an impact of the test data volume when the partition size is increased. These results point to a hybrid methodology for generating and applying such patterns.

We presented data that showed the need to expand the kind of scan patterns used to include multi-cycle scan tests.

A fairly large class of faults that cannot be detected by scan patterns and needs the entire system to be exercised using functional patterns was identified. However, existing methodologies cannot evaluate functional patterns to identify gaps in such interactions. We propose the need for an investigation of a better metric to model such interactions.

## REFERENCES

[1] S. Chakravarty, A. Jain, N. Radhakrishnan, E. W. Savage and S. T. Zachariah, "Experimental Evaluation of Scan Tests for Bridges," *IEEE International Test Conference*, 2002, pp.509-518.

[2] S. Chakravarty, S. T. Zachariah and Carl D. Roth, "Efficient Algorithms for Extracting Two-Node Bridges", *ACM Design Automation Conference*, 2000, pp.790-793.

[3] J.F. Ferguson and J.P. Shen, "Extraction and Simulation of Realistic Faults Using Inductive Fault Analysis," *IEEE International Test Conference*, 1988, pp.475-484.

[4] F.M. Goncalves, J.P. Teixeira, "Sampling techniques of non-equally probable faults in VLSI systems," *IEEE VLSI Test Symposium*, 1998, Page(s):283-288.

[5] V. Krishnamurthy, A. B. Ma and P. Vishakantiah, "A Study of Bridging Defect Probabilities on a Pentium™4 CPU," *IEEE International Test Symposium*, 2001, pp. 688-695.

[6] A. K. Stamper, T. L. McDevitt, S. L. Luce, "Sub-0.25-micron interconnect scaling: damascene copper versus subtractive aluminum," *IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, 1989, pp. 337-346.

[7] S. Zachariah and S. Chakravarty, "A Scalable and Efficient Methodology for Extracting Two Node Bridges from Large Industrial Circuits," *IEEE International Test Conference*, 2000, pp.750-759.

[8] S. T. Zachariah and S. Chakravarty, "Algorithms to Extract Two-Node Bridges," *IEEE Transactions on Very Large Scale Integrated Systems*, pp.741-744, Vol.11, No. 4, August 2003.

[9] S. T. Zachariah and S. Chakravarty, "Two Node Bridge Extraction for Large Industrial Circuits," *IEEE Transactions on Integrated Circuits and Systems*, to appear (2004).

[10] B. Benware, C. Schuermeyer, S. Ranganathan, R. Madge, P. Krishnamurthy, N. Tamarapalli, K. Tsai, J. Rajski, "Impact of Multiple Detect Test Patterns on Product Quality," *IEEE International Test Conference*, 2003, pp. 1031-1040.

[11] Grimaila, M.R.; Sooryong, Lee; Dworak, J.; Butler, K.M.; Stewart, B.; Balachandran, H.; Houchins, B.; Mathur, V.; Jaehong, Park; Wang, L.-C.; Mercer, M.R., "REDO-random excitation and deterministic observation-first commercial experiment," *IEEE VLSI Test Symposium*, pp.268-274, 1999.

[12] Dworak, J.; Wicker, J.D.; Lee, S.; Grimaila, M.R.; Mercer, M.R.; Butler, K.M.; Stewart, B.; Wang, L.-C., "Defect-oriented testing and defective-part-level prediction," *IEEE Design & Test of Computers*, Vol. 18, No.1, 2001, pp.31-41.

[13] Chao-Wen Tseng; McCluskey, E.J., "Multiple-output propagation transition fault test," *IEEE International Test Conference*, 2001, pp.358-366.

[14] McCluskey, E.J.; Al-Yamani, A.; Li, J.C.-M.; Chao-Wen Tseng; Volkerink, E.; Ferhani, F.-F.; Li, E.; Mitra, S., "ELF-Murphy data on defects and test sets," *IEEE VLSI Test Symposium*, pp.16-22, 2004.

[15] Dworak, J.; Cobb, B.; Wingfield, J.; Mercer, M.R., "Balanced excitation and its effect on the fault detection of dynamic defects,"

IEEE Design, Automation and Test in Europe Conference and Exhibition, pp.1066-1071, vol.2, 2004.

- [16] Dworak, J.; Dorsey, D.; Wang, A.; Mercer, M.R., Excitation, observation, and ELF-MD: optimization criteria for high quality test sets, IEEE VLSI Test Symposium, pp.9-15, 2004.
- [17] Dworak, J.; Grimaila, M.R.; Cobb, B.; Wang, T.-C.; Wang, L.-C.; Mercer, M.R., On the superiority of DO-RE-ME/MPG-D over stuck-at-based defective part level prediction, IEEE Asian Test Symposium, 2000, pp.151-157.
- [18] Dworak, J.; Grimaila, M.R.; Sooryong Lee; Wang, L.-C.; Mercer, M.R., "Enhanced DO-RE-ME based defect level prediction using defect site aggregation-MPG-D," IEEE Test Conference, 2000, pp.930-939
- [19] Wang, L.; Mercer, M.R.; Williams, T.W., "Using target faults to detect non-target defects," IEEE International Test Conference, 1996, pp.629-638.
- [20] Yuxin Tian; Grimaila, M.R.; Weiping Shi; Mercer, M. R., "Minimizing defective part level using a linear programming-based optimal test selection method," IEEE Asian Test Symposium, pp.354-359.