RTL Scan Design for Skewed-Load At-Speed Test under Power Constraints

Ho Fai Ko and Nicola Nicolici
Department of Electrical and Computer Engineering
McMaster University, Hamilton, ON, L8S 4K1, Canada
Email: henryko@grads.ece.mcmaster.ca, nicola@ece.mcmaster.ca

Abstract

This paper discusses an automated method to build scan chains at the register-transfer level (RTL) for power-constrained at-speed testing. By analyzing a circuit at the RTL, where design complexity is lower than at the gate netlist level, one can divide a circuit into multiple partitions, which can be tested independently in order to reduce test power. Despite activating one partition at a time, we show how through conscious construction of scan chains, high transition fault coverage can be achieved, while reducing test time of the circuit when employing third party test generation tools. Furthermore, as shown in experimental results, by constructing scan chains for the partitioned circuit at the RTL, area and performance penalty of the design-for-test hardware may be reduced.

I. Introduction

Structural tests targeting the single stuck-at fault model applied through scan chains (SCs) have been successfully used to detect physical defects that affect the static circuit behavior [1]. As the geometric feature size of digital integrated circuits decreases, the number of physical defects that affect the dynamic behavior (i.e., timing failures) is on the rise. One problem with the existing current-based test methods used to screen these type of defects, e.g., IDDQ, is that it is becoming increasingly difficult to distinguish the quiescent current of a faulty device from the fault-free one [2]. As early as in [3], it has been shown that by applying the same set of stuck-at test vectors at the operational frequency, timing-related faults can also be detected. As a result, at-speed testing has established itself as an essential step in manufacturing test. However, applying at-speed tests using scan poses unique challenges as discussed next.

To detect timing-related defects, two test patterns V_1 and V_2 need to be used to initialize the logic into a known state, and to trigger the targeted transitions in the circuit at the operating frequency [4]. In this paper we consider the Skewed-Load test application strategy, where the second pattern V_2 is obtained by shifting the the first pattern V_1. Despite the need for a scan enable which can switch between the scan and capture mode at-speed, this method can reuse the existing infrastructure for testing stuck-at faults and it also eliminates the need for sequential automatic test pattern generation (ATPG) [1]. However, it is important to note that for the Skewed-Load approach the coverage of delay faults (i.e., fault models of timing-related defects) is limited by the correlation between V_1 and V_2. This problem is further aggravated when additional constraints are imposed on SCs by the available power budget during test.

The elevated power dissipation during test has become a major concern that limits the test throughput and manufacturing yield and, consequently, new power-conscious test methodologies have emerged in the past decade [5]. A method independent of the test vectors that guarantees to reduce the test power in the circuit under test (CUT) is to divide the circuit into multiple partitions, such that each partition can be tested separately. This, however, influences the correlation between at-speed vectors applied using Skewed-Load, thus adversely affecting the delay fault coverage. The focus of this paper is to enable the use of scan chain divisions for power-constrained at-speed test using the Skewed-Load test application strategy. By utilizing information obtained from the circuit description at the RTL, the circuit can be partitioned by consciously controlling the flip-flops (FFs) from different SCs via separate scan enables.

The rest of the paper is organized as follows. Section II discusses the related work and gives the motivation for the proposed solution. Section III details our proposal, while results and conclusion are given in Sections IV and V.

II. Related Work and Motivation

A power-aware ATPG algorithm is proposed by Wen et al. in [6]. By filling the don't cares in the test patterns

1-4244-9707-X/06/$20.00 ©2006 IEEE

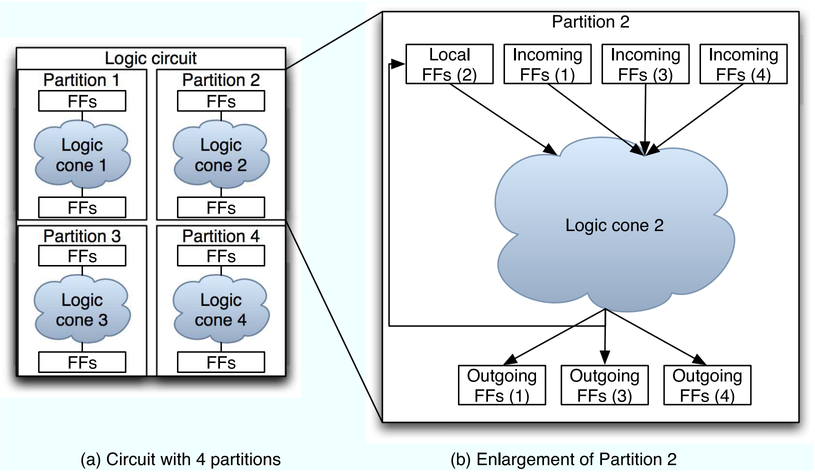


Fig. 1. Example of a partitioned circuit

consciously, the amount of transitions in the circuit during capture is decreased. However, this method may increase the number of test patterns, and is more complex than a regular ATPG algorithm. Butler et al. [7] combine an ATPG algorithm with design partitioning, which limits the amount of active FFs at a given time. However, this method requires the circuit to be partitioned manually. Lee et al. proposed a method to reduce capture power [8] by assigning to each SC a time when it should capture the test responses. To solve the problem of data dependencies between FFs, a new ATPG algorithm is introduced to find the upper bound of the number of capture cycles needed. Despite reducing both shift and capture power, this method may introduce more test patterns and area overhead from the partitioning of SCs. To lower the power during shift, Whetsel proposed to divide the SCs in a design such that they are shifted at different times in [9]. Since the number of FFs that are active during shift is reduced, the amount of transitions in the combinational logic are also decreased. This can effectively reduce the power consumption during shift. Rosinger et al. proposed a method for reducing both the shift power and capture power [10]. By employing a scan architecture with mutually exclusive scan segments, a circuit is divided such that the test patterns and responses for each segment will be shifted, and captured respectively at different times.

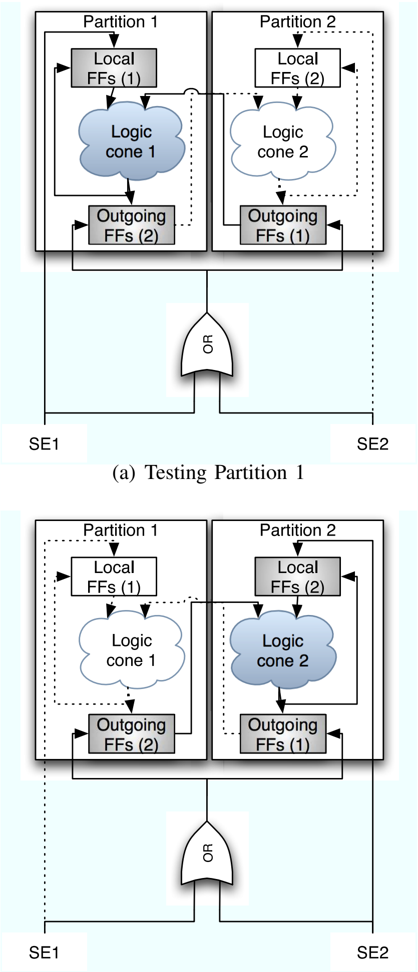
None of the existing methods for scan chain division are suitable for at-speed test using the Skewed-Load test application strategy. This is because when generating patterns that trigger the targeted transitions, all the FFs driving a logic cone must be controlled by an ATPG tool. As a consequence, as shown later in this paper, scan chain divisions need to share FFs between them. Thus, one will need to carefully analyze the design to create the multiple scan chain divisions such that shared FFs between partitions are minimized. Since sizes of designs will continue to increase in the future, the complexity for analyzing and identifying scan chain divisions at the gate level of design abstraction can become prohibitively high. As a consequence, to ensure the complexity of the analysis algorithm remains low, it is apparent that such investigation should be done at the RTL, rather than at the gate netlist level. Although there are a number of methods proposed in the literature, such as [11, 12], to construct functional SCs at the RTL, none of them explore CUT partitioning for managing test power. Therefore, the main motivation for this paper is to investigate the suitability of creating scan chain divisions for power-constrained at-speed test.

III. Creating Scan Chain Divisions at RTL using a New Partitioning Algorithm

In this section we first explain the problem of circuit partitioning. We then introduce the architecture for controlling the partitioned circuit during test. Next, we present an algorithm for dividing a circuit into smaller partitions such that they can be tested independently. After determining how each FF and the associated combinational logic should be allocated to the appropriate partition, the scan synthesis algorithm for inserting SCs to the design at RTL in [13] is applied. This algorithm helps create SCs with reduced correlation between test pattern pairs for Skewed-Load test application strategy. Once the scan circuit is constructed, the corresponding RTL description of the partitioned circuit can be interfaced to a RTL-to-GDSII tool flow for logic synthesis and test pattern generation.

A. Architecture for the Partitioned Circuit

The idea of partitioning is to locate the independent logic cones that are driven and captured by mutually exclusive sets of FFs. When such logic cones are found, they can be put into different smaller sections in a design. In order to better understand the problems, an example of a partitioned circuit is shown in Figure 1(a). In this circuit, there are four partitions, each with its own targeted logic cone and



(a) Testing Partition 1

(b) Testing Partition 2

Fig. 2. Architecture for controlling partitions during test application

its corresponding triggering and capturing FFs. Since each of the four partitions are independent of each other, testing this whole circuit becomes the task of testing four smaller circuits, each with its own logic cone. Since the partition-under-test is smaller than the whole circuit, the power dissipated during shift and capture will be lowered.

There is one major problem that needs to be solved when locating the independent partitions of a circuit. Due to the functional data dependencies in a circuit, it is difficult to locate the mutually exclusive triggering and capturing FFs for the selected logic cones in different partitions. An easy way to solve this problem is to insert dummy FFs between partitions to break these conflicts of shared FFs. However, this could incur excessive amount of area overhead since the dummy FFs are inserted just for the purpose of test.

Figure 1(b) shows an enlargement of Partition 2 in the circuit. As in other partitions, the selected logic cone, and the corresponding FFs that trigger transitions and capture circuit responses are identified. As mentioned above, it is difficult to find the mutually exclusive set of triggering and capturing FFs for a selected logic cone in a circuit. Thus, we divide the FF sets into three categories in order to identify the conflicting FFs. The three categories are *(i) local FFs, (ii) outgoing FFs and (iii) incoming FFs*. *Local FFs* drive and capture responses for the logic cone only in the specified partition. *Outgoing FFs* capture responses from the logic cone in the specified partition, but drive the logic in another partition. The FFs labeled *Outgoing FFs (1)* in Figure 1(b) represent FFs that capture responses of the logic cone in Partition 2, but also drive the logic cone in Partition 1. *Incoming FFs* drive the logic cone in this partition, but are located in another partition. Notice that *Incoming FFs (1)* in Partition 2 are the same set of FFs in Partition 1 labeled *Outgoing FFs (2)*.

When testing a partition, the local FFs, the incoming FFs and the outgoing FFs will have to be activated together. Thus, the incoming FFs and outgoing FFs will not only be active when the targeted partition is under test, but also when testing the adjacent partitions in order to maintain the desired level of fault coverage. As a result, it is obvious that it will be beneficial to have a large number of local FFs, and a small number of incoming FFs and outgoing FFs in order to reduce the test power of a single partition. It is also desired that the outgoing FFs of a partition should be shared with the least amount of adjacent partitions. This not only simplifies the control of the partitioned circuit, but also guarantees the outgoing FFs will be active in as little time as possible. This can be done by carefully identifying the independent logic cones when partitioning the circuit. The algorithm for doing so will be presented in the next subsection. One point the reader should note is that if a logic cone is too large to be partitioned, the number of incoming and outgoing FFs can become excessive if two partitions are created. In this case, since all these shared FFs need to be active at the same time, partitioning a large cone will likely not help reduce its test power.

In order to activate the appropriate sets of FFs for each partition during test, the architecture in Figure 2 will have to be employed. In this architecture, a separate scan enable (SE) signal is assigned to each partition. The local FFs in a partition are controlled by the corresponding SE signal. An OR gate is inserted in order to combine the SE signals from multiple partitions since the outgoing FFs of a partition will also be activated when the adjacent partitions are being tested. Figure 2(a) shows the configuration when Partition 1 is tested. The SE signal for Partition 1 (labeled $SE1$) is enabled as shown in the figure with a solid line. This enables the FF sets *Local FFs (1), Outgoing FFs (2)* in Partition 1, as well as *Outgoing FFs (1)* in Partition 2 in order to test *Logic cone 1*. When the SE signal for Partition 2 (labeled $SE2$) is activated as shown in Figure 2(b), the FF sets *Local FFs (2), Outgoing FFs (1)* in

Partition 2 and *Outgoing FFs (2)* in Partition 1 will be enabled to test *Logic cone 2*. To prevent excessive tester channel occupation, a simple decoder can be inserted to activate the SE signals one at a time, since, to lower test power, only one partition should be active at a time.

B. The Partitioning Algorithm

A pre-processing step consists of extracting information about the data dependencies between FFs in the design from the RTL description. This can easily be done by building a sequential graph (S Graph), where the nodes represent FFs in the design and data dependencies are shown as edges. Once the *S Graph* is built, the problem of dividing a circuit into multiple small partitions with the emphasis of having a large amount of local FFs and small number of outgoing FFs in a partition becomes equivalent to splitting the *S Graph* into smaller sub-graphs with the least amount of cross-edges between each sub-graph. This is because each cross-edge between two sub-graphs represents an outgoing FF between two partitions in a circuit. As a result, the partitioning problem can be easily formulated as the minimal cut set problem, which is known to be NP-hard in graph theory. However, since we have an additional constraint that the outgoing FFs should be shared by a minimum number of neighboring partitions, we have developed a simple greedy algorithm instead of reusing the existing heuristics.

We define four variables in Table I. Before detailing the algorithm, the gain function is described as:

$$Gain = (NFO + NFI) - (NLO + NLI) \qquad (1)$$

This equation gives a higher gain when a node in the *S Graph* has more edges connecting to a node in a neighboring partition than to a node in the local partition.

The algorithm for partitioning a circuit is shown in Algorithm 1. The designer can specify how many partitions are needed in order to meet the power constraint. The function *PartitionSize* returns the number of FFs in the specified partition. At the beginning of the algorithm, all nodes in the *S Graph* are assigned to Partition 1. Then, line 2 will use the gain formula in Equation 1 to calculate the gain of all nodes in the *S Graph*. At this point, the greedy algorithm starting at line 5 of Algorithm 1 will be applied repeatedly until all Partitions are created. It starts by selecting a node with the highest gain from Partition $i-1$ at line 6. The reason for choosing the highest gain can be shown using Figure 3. In this figure, the algorithm is trying to select a node in Partition 1 and move it to Partition 2. The two candidate nodes will be $FF3$ and $FF4$ with gains -1 and 2 respectively. By moving $FF4$ to Partition 2, the number of cross edges between the two partition can be decreased by 2, while moving $FF3$ will increase the number of cross edges by 1.

TABLE I. Variables for the gain function

Variable Name	Representation
NLI	Number of incoming edges from local partition
NLO	Number of outgoing edges to local partition
NFI	Number of incoming edges from foreign partition
NFO	Number of outgoing edges to foreign partition

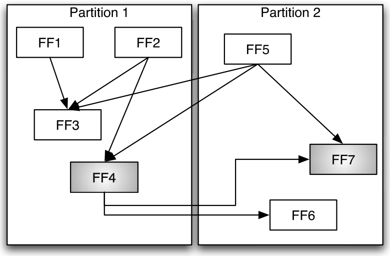


Fig. 3. Example of an S Graph

After a node is selected, lines 8 and 9 will update the list that contains all the incoming FFs and outgoing FFs of a partition. These incoming FF list and outgoing FF list will be used to update the gain of the parent and child nodes of the selected node at line 10 of the algorithm. This is repeated until Partition $i-1$ reaches the targeted size, at which point line 11 will lock the nodes that are driven by Partition $i-1$ in Partition i. The reason for this is to prevent a FF to be shared by more than two partitions. This can be shown using $FF7$ in Figure 3. Assuming the algorithm is trying to select a node from Partition 2 for creating Partition 3, by locking $FF7$, which is driven by $FF4$ in Partition 1, it avoids $FF4$ being shared between Partition 1, 2 and 3 at the same time, since it also drives $FF6$ which is located in Partition 2. Algorithm 1 will terminate at line 11 if the amount of locked FFs is larger than the targeted size.

IV. Experimental Results

In this section we discuss our results for a DMA circuit [14]. It is important to note that this circuit contains 2050 FFs when using gate level scan and 2115 FFs when

Algorithm 1: Partition Algorithm

Input : $S\ Graph$, $Num_{partition}$

Output : Partitioned $S\ Graph$

```
1  Calculate target size of each partition;
2   Assign all nodes to Partition 1;
3  Calculate gain of all nodes;
4  for ($i$ = 2; $i$ < $Num_{partition}$; $i$++) do
5      while ($PartitionSize(i-1)$ > desired size) do
6          Pick node in Partition $i-1$ with highest gain;
7          Move selected node to Partition $i$;
8          Update the incoming list of Partition $i-1$ and $i$;
9          Update the outgoing list of Partition $i-1$ and $i$;
10         Update gain of parent and child nodes of the selected
           node;
11     Lock FFs in Partition $i$;
12 Return $S\ Graph$;
```

Period	SC	Gate Full						RTL Full						△					
(ns)	#	TF	AU	FC	CTP	ST	CPU	TF	AU	FC	CTP	ST	CPU	TF	AU	FC	CTP	ST	CPU
1.85	6	77138	4591	91.80	2354	805	14.23	103730	4974	93.26	4260	629	5.42	-26592	-383	1.46	-1906	176	8.81
	9	77150	4727	91.62	2329	531	15.00	103186	4993	93.23	4218	415	5.45	-26036	-266	1.61	-1889	116	9.55
	12	77162	9241	85.77	2041	349	13.07	104558	5128	92.80	4292	315	5.47	-27396	4113	7.02	-2251	34	7.60
1.80	6	77884	4582	91.61	2323	794	14.39	108166	5243	93.26	4096	604	5.21	-30282	-661	1.65	-1773	190	9.18
	9	77896	4720	91.44	2278	519	12.35	108428	5283	93.27	4140	406	5.37	-30532	-563	1.83	-1862	113	6.98
	12	77908	9246	85.63	2066	353	12.87	107610	5230	93.26	4028	296	5.16	-29702	4016	7.64	-1962	57	7.71
1.75	6	79450	4579	91.69	2304	787	18.97	102658	4987	93.21	4313	637	5.48	-23208	-408	1.52	-2009	150	13.49
	9	79462	4692	91.55	2326	530	12.40	102886	5012	93.20	4331	427	5.37	-23424	-320	1.65	-2005	103	7.03
	12	79474	9297	85.76	2062	352	13.53	104904	5389	92.89	4316	318	5.52	-25430	3908	7.14	-2254	34	8.01
1.70	6	80506	4276	92.62	2295	784	14.61	104204	5114	93.27	4333	641	5.50	-23698	-838	0.65	-2038	143	9.11
	9	80518	4368	92.51	2291	522	12.51	104218	5117	93.28	4308	423	5.22	-23700	-749	0.77	-2017	99	7.29
	12	80530	8869	86.93	2045	349	13.62	104696	5413	93.01	4328	316	5.33	-24166	3456	6.08	-2283	33	8.29
Avg		78756	6099	89.91	2226	556	13.96	104937	5157	93.16	4247	452	5.38	-26180	942	3.25	-2021	104	8.59

TABLE II. Test generation results for the DMA core with three partitions

Period	SC	Gate Full						RTL Full						△					
(ns)	#	TF	AU	FC	CTP	ST	CPU	TF	AU	FC	CTP	ST	CPU	TF	AU	FC	CTP	ST	CPU
1.85	16	77178	4539	91.87	2295	296	15.66	92204	3964	93.60	7426	155	7.37	-15026	575	1.73	-5131	141	8.29
	24	77210	9333	85.66	2074	178	14.53	93339	3776	93.81	7811	106	7.67	-16129	5557	8.14	-5737	72	6.86
	32	77242	5008	91.27	2294	149	16.00	92008	4111	93.41	7428	74	7.58	-14766	897	2.14	-5134	75	8.42
1.80	16	77924	4537	91.67	2305	297	14.05	94289	4004	92.71	7545	157	7.81	-16365	533	1.04	-5240	140	6.24
	24	77956	9339	85.52	2040	175	13.32	93954	3781	93.96	7891	108	7.86	-15998	5558	8.44	-5851	67	5.46
	32	77988	5020	91.06	2271	147	13.23	93947	4128	92.54	7583	76	8.35	-15959	892	1.48	-5312	71	4.88
1.75	16	79490	4526	91.76	2291	295	13.98	93860	4050	92.55	7487	154	7.57	-14370	476	0.79	-5196	141	6.41
	24	79522	9387	85.65	2067	177	13.77	94314	3715	93.94	7758	105	7.75	-14792	5672	8.29	-5691	72	6.02
	32	79554	5029	91.13	2278	148	12.92	93379	4309	92.43	7447	73	7.91	-13825	720	1.29	-5169	75	5.01
1.70	16	80546	4226	92.69	2338	301	12.88	89196	3934	92.32	7540	155	7.42	-8650	292	-0.37	-5202	146	5.46
	24	80578	8985	86.79	2038	175	13.27	89372	3758	93.48	7918	108	7.35	-8794	5227	6.69	-5880	67	5.92
	32	80610	4723	92.08	2264	147	12.31	88423	4005	92.42	7509	75	8.08	-7813	718	0.34	-5245	72	4.23
Avg		78816	6221	89.76	2213	207	13.83	92357	3961	93.10	7612	112	7.73	-13541	2260	3.33	-5399	95	6.10

TABLE III. Test generation results for the DMA core with eight partitions

employing the proposed RTL scan. The number of FFs is different between the two circuits because for gate level scan the synthesis tool will be able to remove the redundant FFs before introducing scan. However, with RTL scan, these redundant FFs are already included in the scan paths.

When dividing a circuit into k partitions, it is expected that the number of active FFs in a partition should be 1/k of the total number of FFs. However, in the proposed solution, for the purpose of launching patterns for detecting delay faults, not only the local FFs, but also the incoming FFs in neighboring partitions must be activated. For example, for three partitions, there will be 1044 (or 49.4%) active FFs in the largest partition; for eight partitions, the largest partition will have 522 (or 24.7%) FFs. Note, it is assumed that the power when testing a single partition will be directly proportional to the scan chain division size. This is consistent with prior literature [9].

Tables II and III show the testability results generated by a commercial ATPG tool [15] for the DMA circuit with three and eight partitions. The timing constraints and number of SCs are listed in columns 1 and 2 respectively. Note that the number of SCs for the DMA circuit with three and eight partitions are chosen to be different to show that the benefit of our approach is irrespective to the number of SCs in the design. The column labeled TF represents the total number of transition delay faults in the circuit. The column labeled AU represents ATPG untestable faults, which are faults that are untestable due to the limitation of scan cell arrangement. FC corresponds to fault coverage for transition faults, CTP denotes the number of compressed test patterns, ST is the scan time in thousands of clock cycles, and CPU represents test generation time in seconds. For the columns labeled *Gate Full*, full scan is inserted at the gate level after the circuit has been synthesized. All the columns for the gate level case are generated without partitioning the circuit. For the columns named *RTL Full*, the testability results for TF, AU, CTP, ST and CPU are calculated by summing the corresponding data between the multiple partitions that are obtained by our approach. It is to be noted that the TF for RTL scan is higher than that of gate level scan due to the presence of redundant FFs and the added DFT logics for controlling the multiple partitions. Despite the increase in TF, the amount of AU for RTL scan with three and eight partitions are actually 942 and 2260 faults less than that of gate level scan on average. This decrease in AU faults in turn improves the fault coverage of the RTL scan by 3.25% and 3.33%. This improvement is due to reusing the method from our prior work [13]. However, by employing the scan chain division method proposed in this paper, although the amount of CTP increases by over 2000 and 5000 on average, the scan time is actually reduced on average by 104 and 95 thousand clock cycles. This translates also into 18.7% and 45.9% reduction in volume of test data for the DMA circuit with three and eight partitions respectively. This is because within a single partition, there are fewer active FFs that need to be scanned. Besides, note that our proposed solution also improves the test generation time.

Tables IV and V show the area and performance results comparison between gate level scan and RTL scan with three and eight partitions for the DMA circuit. Column 1 shows the timing constraints used by the synthesis tool [16]. Column 2 provides the total number of SCs and Columns 3, 5 and 7 indicate whether the timing constraints were met during synthesis for the non-scan circuit, the circuit with gate level scan and the RTL scan circuit. Columns 4 and 6 show the area overhead when compared to the non-scan circuit for gate level scan and

	#	Original	Area OH				
Period	of	(No DFT)	Gate Full		RTL Full		Δ
(ns)	SC	Timing Met?	%	Met?	%	Met?	%
1.85	6	Yes	11.47	Yes	11.21	Yes	0.26
	9		11.47	Yes	10.52	Yes	0.96
	12		11.47	Yes	11.03	Yes	0.44
1.80	6	Yes	13.02	No	11.67	Yes	1.35
	9		13.02	No	11.14	Yes	1.88
	12		13.02	No	10.13	No	2.89
1.75	6	Yes	14.50	No	8.95	Yes	5.55
	9		14.50	No	8.84	Yes	5.66
	12		14.50	No	9.97	Yes	4.53
1.70	6	No	18.19	No	12.42	Yes	5.78
	9		18.19	No	12.48	Yes	5.71
	12		18.19	No	12.49	Yes	5.70
Average			14.30		10.90		3.39

TABLE IV. Area data for DMA with three partitions

	#	Original	Area OH				
Period	of	(No DFT)	Gate Full		RTL Full		Δ
(ns)	SC	Timing Met?	%	Met?	%	Met?	%
1.85	16	Yes	11.47	Yes	12.11	Yes	-0.64
	24		11.47	Yes	10.99	Yes	0.48
	32		11.47	Yes	11.91	Yes	-0.44
1.80	16	Yes	13.02	No	12.88	Yes	0.14
	24		13.02	No	11.27	No	1.75
	32		13.02	No	11.98	Yes	1.04
1.75	16	Yes	14.50	No	11.23	Yes	3.27
	24		14.50	No	11.28	No	3.22
	32		14.50	No	12.29	Yes	2.21
1.70	16	No	18.19	No	15.18	Yes	3.01
	24		18.19	No	14.77	Yes	3.42
	32		18.19	No	14.22	Yes	3.97
Average			14.30		12.51		1.79

TABLE V. Area data for DMA with eight partitions

RTL scan accordingly. Column 8 shows the difference in area overhead between gate level scan and RTL scan. As can be seen in the table, despite the presence of the redundant FFs and the additional logic for controlling the partitions during test, the area for RTL scan is 3.39% and 1.19% respectively less on average than that of gate level scan. Moreover, from Columns 3 and 5 of Tables IV and V, the non-scan circuit and gate level scan fail to meet timing beyond 1.75 ns and 1.80 ns respectively. However, the performance is improved to 1.7 ns for the RTL scan with three and eight partitions as indicated in Columns 7 in both tables. We attribute this contribution to the fact that the logic synthesis tool can better optimize the circuit by generating the scan paths and functional logic simultaneously when the scan infrastructure is provided in the RTL description. However, one point to note is that the synthesis tool failed to meet the timing constraints at 1.8 ns for the DMA circuit with three partitions and 12 SCs, and at 1.8 ns and 1.75 ns for the DMA circuit with eight partitions and 24 SCs. We consider this anomaly to be caused by the heuristic nature of the logic synthesis engine. It is also important to note that the computational time for partitioning the circuit and inserting scan at the RTL only takes a few minutes when performed on the DMA circuit with 2115 FFs on a 1.5GHz PowerPC G4 with 1GB of RAM.

V. Conclusion

This paper described how by dividing a circuit into multiple partitions at the RTL for power-constrained at-speed testing, testability of the circuit can be improved by consciously constructing the SCs.

References

[1] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Boston: Kluwer Academic Publishers, 2000.

[2] X. Lin, R. Press, J. Rajski, P. Reuter, T. Rinderknecht, B. Swanson, and N. Tamarapalli, "High-Frequency, At-Speed Scan Testing," *IEEE Design and Test of Computers*, vol. 20, no. 5, pp. 17–25, Sept-Oct 2003.

[3] P. Maxwell, R. Aitken, V. Johansen, and I. Chiang, "The Effect of Different Test Sets on Quality Level Prediction: When is 80% Better Than 90%?" in *Proceedings of International Test Conference*, 1991, p. 358.

[4] J. Savir and S. Patil, "Scan-Based Transition Test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 8, pp. 1232–1241, 1993.

[5] N. Nicolici and B. M. Al-Hashimi, *Power-Constrained Testing of VLSI Circuits*. Kluwer Academic Publishers, 2003.

[6] X. Wen, Y. Yamashita, S. Morishima, S. Kajihara, L.-T. Wang, K. K. Saluja, and K. Kinoshita, "Low-Capture-Power Test Generation for Scan-Based At-Speed Testing," in *Proceedings of International Test Conference*, 2005, pp. 4–10.

[7] K. Butler, J. Saxena, A. Jain, T. Fryars, J. Lewis, and G. Hetherington, "Minimizing Power Consumption in Scan Testing: Pattern Generation and DFT Techniques," in *Proceedings of International Test Conference*, 2004, pp. 355–364.

[8] K.-J. Lee, S.-J. Hsu, and C.-M. Ho, "Test Power Reduction with Multiple Capture Orders," in *Proceedings of the 13th Asian Test Symposium*, 2004, pp. 26–31.

[9] L. Whetsel, "Adapting Scan Architectures for Low Power Operation," in *Proceedings of International Test Conference*, 2000, pp. 863–872.

[10] P. Rosinger, B. M. Al-Hashimi, and N. Nicolici, "Scan Architecture With Mutually Exclusive Scan Segment Activation for Shift- and Capture-Power Reduction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 7, pp. 1142–1153, 2004.

[11] T. Asaka, S. Bhattacharya, S. Dey, and M. Yoshida, "H-SCAN+: A Practical Low-Overhead RTL Design-for-Testability Technique for Industrial Designs," in *Proceedings of International Test Conference*, 1997, pp. 265–274.

[12] Y. Huang, C.-C. Tsai, N. Mukherjee, O. Samman, D. Devries, W.-T. Cheng, and S. M. Reddy, "On RTL Scan Design," in *Proceedings of International Test Conference*, 2001, pp. 728–737.

[13] H. F. Ko and N. Nicolici, "Functional Scan Chain Design at RTL for Skewed-load Delay Fault Testing," in *Proceedings of the 13th Asian Test Symposium*, 2004, pp. 454–459.

[14] Faraday Technology Corporation. (2002) Faraday Structured ASIC Benchmarks. [Online]. Available: http://www.faraday-tech.com/StructuredASIC/download/

[15] Synopsys Test Tools, "TetraMAX ATPG," http://www.synopsys.com/products/test/tetramax_dsA4.pdf, 2003. [Online]. Available: http://www.synopsys.com/products/test/tetramax_dsA4.pdf

[16] Synopsys Synthesis Tools, "Design Compiler," http://www.synopsys.com/products/logic/design_compiler.html, 2003. [Online]. Available: http://www.synopsys.com/products/logic/design_compiler.html