Combined Test Data Selection and Scheduling for Test Quality Optimization under ATE Memory Depth Constraint

Erik Larsson and Stina Edbom
Embedded Systems Laboratory
Linköpings Universitet, Sweden

Abstract[1]

The increasing test data volume required to ensure high test quality when testing a System-on-Chip is becoming a problem since it (the test data volume) must fit the ATE (Automatic Test Equipment) memory. In this paper, we (1) define a test quality metric based on fault coverage, defect probability and number of applied test vectors, and (2) a test data truncation scheme. The truncation scheme combines (1) test data (vector) selection for each core based on our metric, and (2) scheduling of the execution of the selected test data, in such a way that the system test quality is maximized, while the selected test data is guaranteed to fit the ATE's memory. We have implemented the technique and the experimental results, produced at reasonable CPU times, on several ITC'02 benchmarks show that high test quality can be achieved by a careful selection of test data.

1. Introduction

High test quality when testing a System-on-Chip (SOC) is a must. However, high quality requires a large test data volume to be stored in the limited memory of the ATE (Automatic Test Equipment). Currently, the test data volume for SOCs increases faster than the number of transistors in a design [16]. The increasing test data volume is due to (1) high number of fault sites because of the high amount of transistors, (2) new defect types introduced with nanometer process technologies, and (3) faults related to timing and delay since systems have higher performance and make use of multiple-clock domains [16]. In addition, updating to a new ATE is expensive, hence not an alternative. The problem is therefore, in order to have the highest possible test quality, to find techniques to make the test data volume fit the limited memory of the existing ATE.

Vranken *et al.* [16] discuss three alternatives to make the test data fit the ATE; (1) *test memory reload*, where the test data is divided into several partitions, is possible but not practical due to the high time involved, (2) *test data truncation*, the ATE is filled as much as possible and the test data that does not fit the ATE is simply not applied, leads to reduced test quality, and (3) *test data compression*, the test stimuli is compressed, however, it does not guarantee that the test data will fit the ATE. Thus there is a need for a usable technique that ensures high test quality while making sure the test data volume fits the ATE memory.

1. The research is partially supported by the Swedish National Program STRINGENT.

The test data must also be organized or scheduled in the ATE. A recent industrial study showed that by using test scheduling the test data was made to fit the ATE [3]. The study clearly showed that ATE memory limitations is a real and critical problem.

Test scheduling reduces the amount of idle bits to be stored in the ATE, and therefore scheduling must be considered in combination with test data reduction. Also when discussing memory limitations, the ATE memory depth in bits is equal to the maximal test application time for the system in clock cycles [9]. Hence, the memory constraint can be seen as a time constraint.

In this paper, we explore test data truncation. The aim is a technique that maximizes test quality while making sure that the selected test data fits the ATE. We assume that given is a core-based design and for each core defect probability, maximal fault coverage and size of the test set are given. We define for each core and its test data a CTQ (core test quality) metric and for the system a STQ (system test quality) metric. The metrics reflect that test data should be selected from a core (1) with high probability of having a defect and (2) where it is possible to detect a fault using a minimal number of test vectors. For the fault coverage function we make use of an estimation. Fault simulation can extract fault coverage at each test vector, however, it is a time consuming process and also it might not be applicable for all cores due to IP-protection, for instance.

The test vectors in a test set can be applied in any order. However, regardless of order, it is well-known in the test community that the first stimuli detects a higher number of faults compared to stimuli applied at the end of testing a testable unit, and that the function fault coverage versus number of test vectors is approximately exponential/ logarithmic. We therefore assume that the fault coverage over time for a core can be approximated to an exponential/ logarithmic function.

We make use of CTQ metric to select test data volume for each core in such a way that the test quality for the system is maximized (STQ), and we integrate the selection with test scheduling in order to verify that the selected test data volume really fits the ATE memory. We have implemented our technique and we have made experiments that demonstrate that high test quality can be achieved by applying only a sub-set of the test stimuli. Furthermore, it is possible to turn the problem (and our solution), and view it as: for a certain test quality, which test data should be selected to minimize the test application time.

The advantage with our technique is that given a core-based system with test sets, a number on maximal fault coverage, and defect probability per core, we can select test data for the system and schedule the selected test data in such a way that the test quality is maximized and the selected test data fits the ATE memory.

The paper is organized as follows. In Section 2 we present related work, and in Section 3 the problem definition is given. The test quality metric is defined in Section 4 and our test data selection and scheduling approach is described in Section 5. The experiments are presented in Section 6 and finally the paper is concluded in Section 7.

2. Related Work

Test scheduling and test data compression are examples of approaches proposed to reduce the high test data volumes that must be stored in the ATE in order to test SOCs.

The idea in test scheduling is to organize the test bits in the ATE in such a way that the number of introduced so called idle bits (not useful bits) is minimized. The gain is reduced test application time and reduced test data volume. A scheduling approach depends on a test architecture such as the AMBA test bus [4], the test bus [15] and the TestRail [13]. Iyengar *et al.* [8] proposed a technique to partition the set of scan chain elements (internal scan chains and wrapper cells) at each core into wrapper chains, which are connected to TAM wires in such a way that the total test time is minimized. Goel *et al*. [3] showed that ATE memory limitation is a critical problem. On an industrial design they showed that by using an effective test scheduling technique the test data can be made to fit the ATE.

There has also been scheduling techniques that make use of an abort-on-fail strategy, that is the testing is terminated as soon as a fault is detected. Koranne's minimizes the average-completion time by scheduling short tests early [11]. Other techniques have taken the defect probability for each testable unit into account [5,10,12]. Huss and Gyurcsik proposed a sequential technique using of a dynamic programming algorithm for ordering the tests [5], while Milor and Sangiovanni-Vincentelli presents a sequential technique based on selection and ordering of test sets [14]. Jiang and Vinnakota proposed a sequential technique, where the information about the fault coverages provided by the tests is extracted from the manufacturing line [10]. For SOC designs, Larsson *et al.* proposed a technique based on ordering of tests, considering different test bus structures, scheduling approaches (sequential vs. concurrent) and test set assumptions (fixed test time vs. flexible test time) [12].

Several compression schemes have been used to compress the test data. For instance, Ichihara *et al*. used statistical codes [6], Chandra and Chakrabarty Golomb codes[1], Iyengar *et al.* run-length codes [7], Chandra and Chakrabarty Frequency-directed run-length codes [2], and Volkerink *et al.* Packet-based codes [17].

All approaches above reduces the ATE memory requirement. The main advantage with the techniques is that the highest possible test quality is reached. However, the main disadvantage is that techniques do not guarantee that the test data volume fits the ATE. Hence, they might not be applicable in practice. It means that there is a need for a technique that in a systematic way defines the test data volume for a system in such a way that the test quality is maximized while the test data is guaranteed to fit the ATE memory.

3. Problem Formulation

We assume that given is a core-based architecture with n cores denoted by i. For each core i in the system, the following is given:

- $sc_{ij}=\{sc_{i1}, sc_{i2},..., sc_{im}\}$ - the length of the scanned elements at core i are given where m is the number of scanned elements,
- wi_i - the number of input wrapper cells,
- wo_i - the number of output wrapper cells,
- wb_i - the number of bidirectional wrapper cells,
- tv_i - the number of test vectors,
- fc_i - the fault coverage reached when all the tv_i test vectors are applied.
- pp_i - the pass probability per core and,
- dp_i - the defect probability per core (given as 1-pp_i).

For the system, a maximal TAM bandwidth W_{tam}, a maximal number of k TAMs, and a upper-bound memory constraint M_{max} on the memory depth in the ATE are given.

The TAM bandwidth W_{tam} is to be partitioned into a set of k TAMs that we denote by j each of width $W_{tam}=\{w_1, w_2, ..., w_k\}$ in such a way that:

$$W_{tam} = \sum_{j=1}^{k} w_j \qquad 1$$

and on each TAM, one core can be tested at a time.

Since the memory depth in the ATE (in bits) is equal to the test application time for the system (in clock cycles) [9], the memory constraint can be seen as a time constraint τ_{max}:

$$M_{max} = \tau_{max} \qquad 2$$

Our problem is to:

- For each core i select the number of test vectors (stv_i),
- Partition the given TAM width W_{tam} into no more than k TAMs,
- Determine the width of each TAM (w_j),
- Assign each core to one TAM, and
- Assign a start time for the testing of each core.

The selection of test data (stv_i for each core i) and the test scheduling should be done in such a way that the test quality of the system (defined in Section 4) is maximized while the ATE memory depth (M_{max}) (test application time τ_{max}) is met.

4. Test Quality Metric

For the truncation scheme we need a test quality metric to (1) select test data for each core and (2) to measure the system test quality. We take the following parameters into account to measure test quality: defect probability, fault coverage, and number of applied test vectors.

The defect probability, the probability that a core is defect, can be collected from the production line or set by experience. Defect probability has to be taken into account since it is better to select test data for a core with a high defect probability since it is more likely to hold a defect compared to a core with a low defect probability.

The possibility to detect faults depends on the fault coverage versus the number of applied test vectors, hence the fault coverage and the number of applied test vectors also have to be taken into account. Fault simulation can be used to extract which fault each test vector detect. However, in a complex core-based design with a high number of cores, fault simulation for each core is, if possible due to IP-protection, highly time consuming. We therefore make use of an estimation technique. The fault coverage does not increase linearly over the number of applied test vectors. Figure 1 shows the fault coverage for a set of ISCAS benchmarks. And the following observation can be made: the curves have an exponential/logarithmic behaviour (as in Figure 2). We assume that the fault coverage after applying stv_i for core i can be estimated to:

$$fc_i(stv_i) = \frac{\log(stv_i + 1)}{slopeConst} \qquad 3$$

where the *slopeConst* is given as follows:

$$slopeConst = \frac{\log(tv_i + 1)}{fc_i} \qquad 4$$

and the +1 is used to adjust the curve to passes the origin.

For each core i we use the CTQ_i (core test quality) as:

$$CTQ_i = dp_i \times fc_i(stv_i) \qquad 5$$

and for the system with n cores, we introduce the *STQ* (system test quality) metric given as:

$$STQ = \sum_{i=1}^{n} CTQ_i / \sum_{i=1}^{n} dp_i \qquad 6$$

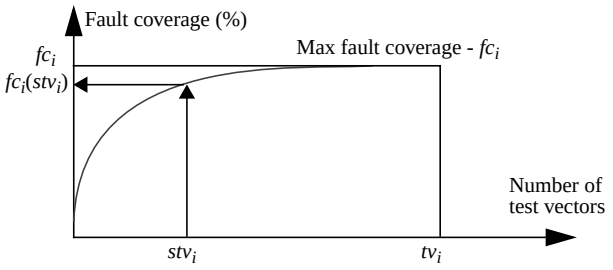


Figure 2. Fault coverage versus number of test vectors estimated as an exponential/logarithmic function.

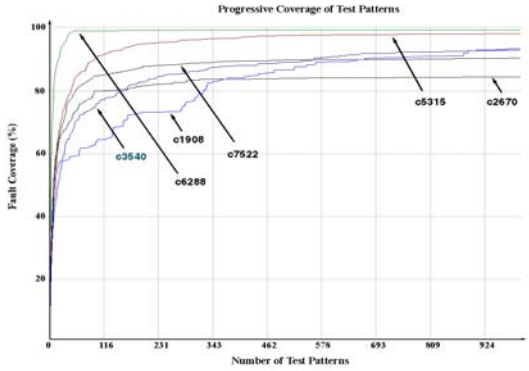


Figure 1. Fault coverage versus the number of applied test vectors for a set of ISCAS designs.

5. Test Scheduling and Test Vector Selection

In this section we describe our technique to optimize test quality by using test vector selection and test scheduling under time constraint given by the ATE memory depth (see Eq. 2 and [9]).

Given is a system as described in Section 3. We assume an architecture where the TAM wires can be grouped into several TAMs and the cores connected to the same TAM are tested sequentially [15]. For grouping the scanned elements (scan-chains, input cells, output cells and bidirectional cells) into a balanced number of so called wrapper scan chains, which are to be connected to the TAM wires w_j, we make use of the *Design_wrapper* algorithm proposed by Iyengar *et al*. [8]. For a wrapper chain configuration at a core i where si_i is the longest wrapper scan-in chain and so_i is the longest wrapper scan-out chain, the test time $\tau_i(w_j, tv_i)$ for core i is given by [8]:

$$= (1 + max(si_i(w), so_i(w))) \times tv + min(si_i(w), so_i(w)) \qquad 7$$

where tv_i is the number of applied test vectors for core i and w is the TAM width.

We need a way to guide the assignment of cores to TAMs in the case when multiple TAMs exist. We make use of the fact that balancing the wrapper scan-in chain and wrapper scan-out chain introduces different number of ATE idle bits as the TAM bandwidth varies. We define TWU_i (TAM width utilization) for a core i at a TAM of width w as:

$$TWU_i(w) = max(si_i(w), so_i(w)) \times w \qquad 8$$

and we make use of a single wrapper-chain (one TAM wire) as a reference point to introduce *WDC* (wrapper design cost) that measure the imbalance (introduced number of idle bits) for a TAM width w relative to TAM width 1:

$$WDC_i = TWU_i(w) - TWU_i(1) \qquad 9$$

For illustration of variations of ATE idle bits, we plot the value of *WDC* for different TAM widths, obtained by using core 1 of the ITC'02 benchmark p93791, in Figure 3.

The algorithm for our test truncation scheme is outlined in Figure 4. Initially no test vector is selected for any core. The vector that contributes most to improving STQ is

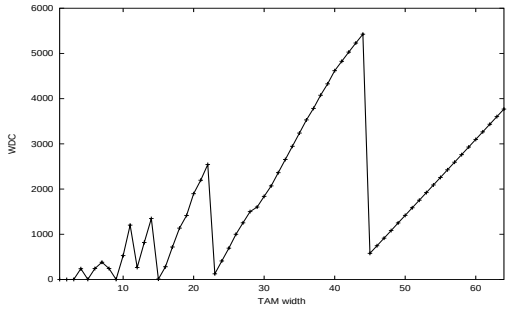


Figure 3. WDC at core 1 (p93791) at various TAM width.

1. Given: τ_{max} - the upper test time limit for the system
   W_{tam} - number of TAM wires - distributed over k TAMs w_1, w_2, ..., w_k in such a way that Eq. 1 holds.
2. Variables: $stv_i = 0$ //selected number of test vectors for core i
   $TAT = 0$ // test application time of the system
3. Compute WDC_i for all cores at all k TAMs (Eq. 9)
4. Select best TAM for each core based on WDC_i
5. **while** $TAT < \tau_{max}$ **at any TAM begin**
6. **for** i=1 to n **begin** // For all cores
7. Compute $\tau(w_j,1)$ (Eq. 7)
8. Compute CTQ_i assuming $stv_i=stv_i+1$ (Eq. 5)
9. **end**
10. **for core** with highest CTQ/$\tau(w_j,1)$ and $stv_i<tv_i$
11. $stv_i=stv_i+1$
12. **for all** cores where $stv_i>0$ **begin**// some selected vectors
13. Assign core to an available TAM with minimal WDC_i
14. **if** a TAM is full ($<\tau_{max}$) - mark TAM as unavailable.
15. **end**
16. Compute and return STQ (Eq. 6).
17. **end**

Figure 4. Test vector selection and test scheduling algorithm.

selected, assigned to a TAM where WDC is minimal and scheduled on the TAM. Additional vectors are selected one by one in such a way that STQ is maximized, and after each selection the schedule is created to verify that the time constraint (ATE memory depth) is met. Note that the test vectors for a core might not be selected in order. However, for the scheduling, the test vectors for each core are grouped and scheduled as a single set. The algorithm (Figure 4) assumes a fixed TAM partition (number of TAMs and their width). We have therefore added an outer loop that makes sure that we explore all possible TAM configurations.

6. Experimental Results

We have implemented the technique and made experiments using five ITC'02 benchmarks, d281, d695, p22810, p34392, and p93791 to demonstrate the importance of considering defect probability, fault coverage, and test data selection. We have added pass probabilities and maximal fault coverage numbers for each core (the data is omitted but for d695 (Table 2) due to space limitations). For the experiments we assume a TAM bandwidth W_{tam} of 32.

All experimental results are collected in Table 3, and the results on design p22810 are also plotted in Figure 6. The computational cost for every experiment is in the range of a few seconds to a few minutes. We have performed experiments at various ATE memory depths constraints (equal to time constraints (see Eq. 2 and [9])). The constraints are set as a percentage of the time it would take to apply all test vectors. At each time constraint we make the following experiments:

1. Test scheduling when not considering defect probability nor fault coverage and testing is aborted at τ_{max} (column three (Table 3) - technique 1).
2. Test scheduling when considering defect probability but not fault coverage and testing is aborted at τ_{max} (column four (Table 3) - technique 2).
3. Test scheduling when considering defect probability as well as fault coverage and testing is aborted at τ_{max} (column five (Table 3) - technique 3).
4. Test scheduling and test vector selection when considering defect probability and fault coverage, using one TAM (column six (Table 3) - technique 4).
5. Test scheduling and test vector selection when considering defect probability and fault coverage, using up to two TAMs (column seven (Table 3) - technique 5).
6. Test scheduling and test vector selection when considering defect probability and fault coverage, using up to three TAMs (column eight (Table 3) - technique 6).

We illustrate our technique on design d695 where the time constraint is set to 5% of the maximal test application time and where the selected test data volume per core is reported in Table 1, and the test schedules for technique 1, 4, and 5 and the corresponding STQ are presented in Figure 5.

From the experimental results collected in Table 3 we learn that the STQ value increases with the time constraint, which is obvious. We also see that test set selection improves the test quality when comparing STQ at the same test time limit. But also important, we note that we can achieve a high test quality at low testing times. Take design p93791, for example, where the STQ value (0.584) for technique 1 at 75% of the testing time is lower than the STQ value (0.748) at only 5% for technique 6. It means that it is possible, by integrating test set selection and test scheduling, to reduce the test application time while keeping the test quality high.

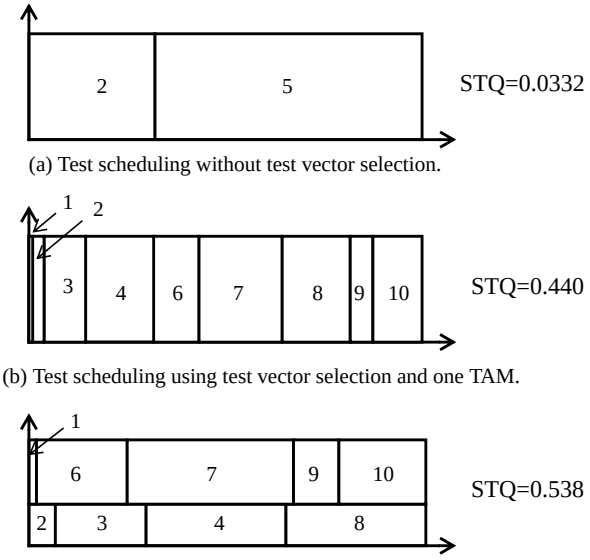
Technique	Selected % of test data for each core										
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Technique 1	0	0	100	0	0	20	0	0	0	0	0
Technique 2	0	0	0	0	0	0	0	54.7	0	0	0
Technique 3	100	0	0	0	0	0	0	52.6	0	0	0
Technique 4	0	100	9.6	6.7	4.8	0	1.7%	10.5	6.2	8.3	4.4
Technique 5	0	100	9.6	16.0	10.5	0	3.8	21.1	13.4	8.3	4.4
Technique 6	0	100	9.6	17.3	11.4	0	2.6	13.7	17.5	33.3	14.7

Table 1. Selected test vectors (%) for the cores in design d695 considering different scheduling techniques.

7. Conclusions

The test quality when testing a System-on-Chip must be kept high. However, high test quality requires a high test data volume that must fit the limited memory of the ATE (Automatic Test Equipment). A new ATE is expensive and and currently the test data volumes increases at a higher pace than the number of transistors in a system. Several compression schemes and test scheduling techniques have been developed to make the test data fit the ATE memory. However, these techniques reduces the test data volume but they do not guarantee that the volume will fit the ATE.

In this paper we have therefore proposed a test data truncation scheme that systematically selects test vectors and schedules the selected test vectors for each core in a core-based system in such a way that the test quality is maximized while the constraint on ATE memory depth is met. We have defined a test quality metric based on defect probability, fault coverage and the number of applied vectors, that is used in the test data selection scheme. We have implemented our technique and the experiments on several ITC'02 benchmarks, at reasonable CPU times, show that high test quality can be achieved by careful selection of test data. Further, our technique can be used to shorten the test application time for a given test quality value.



(a) Test scheduling without test vector selection.

(b) Test scheduling using test vector selection and one TAM.

(c) Test scheduling using test vector selection and two TAMs.

Figure 5. Results for different scheduling techniques. (a) - technique 1, (b) technique 4, and (c) technique 5.

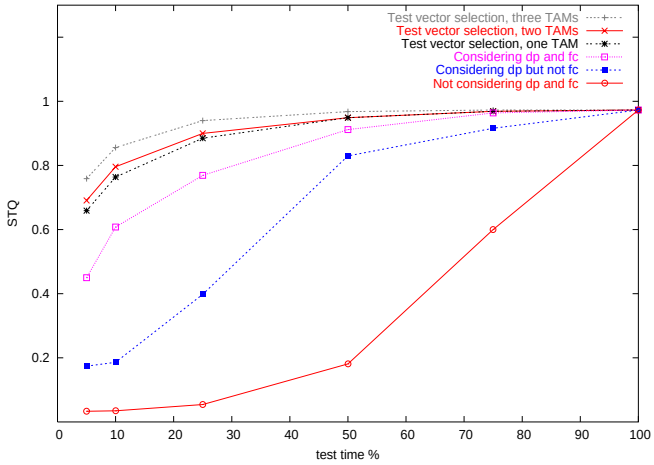


Figure 6. Experimental results on p22810.

References

[1] A. Chandra and K. Chakrabarty, "System-on-a-Chip Test Data Compression and Decompression Architectures Based on Golomb Codes", *Trans. on CAD of IC and Systems*, Vol. 20, No. 3, 2001, pages 355-367.

[2] A. Chandra and K. Chakrabarty, "Frequency -Directed-Run-Length (FDR) Codes with Application to System-on-a-Chip Test Data Compression", *Proc. of VTS*, 2001, pages 42-47.

[3] S. K. Goel *et al.*, "Test Infrastructure Design for the Nexperia^{TM}Home Platform PNX8550 System Chip", *Proc. of DATE*, Paris, France, 2004, pages 1530-1591.

[4] P. Harrod, "Testing reusable IP-a case study", *Proc. of ITC*, Atlantic City, NJ, USA, pp. 493-498, 1999.

[5] S. D. Huss and R. S. Gyurcsik, "Optimal Ordering of Analog Integrated Circuit Tests to Minimize Test Time", *Proceedings of DAC*, pp. 494-499, 1991.

[6] H. Ichihara *et al.*, "Dynamic Test Compression Using Statistical Coding", *Proc. of ATS*, 2001, pages 143-148.

[7] V. Iyengar *et al.*, "Built-In Self-Testing of Sequential Circuits Using Precomputed Test Sets", *Proc. of VTS*, 1998, pages 418-423.

[8] V. Iyengar *et al.*, "Test wrapper and test access mechanism co-optimization for system-on-chip", *Proc. of International Test Conf.*, Baltimore, MD, USA, pp. 1023-1032, 2001.

[9] V. Iyengar *et al.*, "Test resource optimization for multi-site testing of SOCs under ATE memory depth constraints", *Proc. of ITC*, pp. 1159 - 1168, Baltimore, USA, Oct. 2002.

[10] W. J. Jiang and B. Vinnakota, "Defect-Oriented Test Scheduling", *Transactions on Very-Large Scale Integration (VLSI) Systems, Vol. 9, No. 3*, pp. 427-438, June 2001.

[11] S. Koranne, "On Test Scheduling for Core-Based SOCs", *Proceedings of International Conference on VLSI Design (VLSID)*, pp. 505-510, Bangalore, India, January 2002.

[12] E. Larsson, J. Pouget, and Z. Peng, "Defect-Aware SOC Test Scheduling", *Proceedings of VLSI Test Symposium (VTS)*, Napa Valley, Ca, USA, April 2004.

[13] E. J. Marinissen *et al.*, "A structured and scalable mechanism for test access to embedded reusable cores", *Proc. of ITC*, Washington, DC, USA, pp. 284-293, October 1998.

[14] L. Milor and A. L. Sangiovanni-Vincentelli, "Minimizing Production Test Time to Detect Faults in Analog Circuits", *Trans. on CAD of IC &Sys.*, Vo. 13,No. 6, pp 796-,June 1994.

[15] P. Varma and S. Bhatia, "A Structured Test Re-Use Methodology for Core-based System Chips", *Proc of ITC*, pp. 294-302, Washington, DC, USA, October 1998.

[16] H. Vranken *et al.*, "ATPG Padding And ATE Vector Repeat Per Port For Reducing Test Data Volume", *Proc. of ITC*, Charlotte, NC, USA, 2003, pages 1069-1078.

[17] E. H. Volkerink *et al.*, "Packet-based Input Test Data Compression Techniques", *Proc. of ITC*, 2002, pp. 154-163.

	Core										
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Scan-chains	0	0	0	1	4	32	16	16	4	32	32
Inputs w_i	0	32	207	34	36	38	62	77	35	35	28
Outputs w_o	0	32	108	1	39	304	152	150	49	320	106
Test vectors tv_i	0	12	73	75	105	110	234	95	97	12	68
Pass probability pp_i	97	98	99	95	92	99	94	90	92	98	94
Max fault coverage fc_i (%)	95	93	99	98	96	96	99	94	99	95	96

Table 2 Data for benchmark d695.

SOC	% of max test time	Technique 1	Technique 2	Technique 3	Technique 4	Technique 5	Technique 6
		STQ	STQ	STQ	STQ	STQ	STQ
d281	5	0.0209	0.164	0.496	0.674	0.726	0.726
	10	0.0230	0.186	0.563	0.774	0.818	0.818
	25	0.198	0.215	0.834	0.879	0.905	0.912
	50	0.912	0.237	0.903	0.935	0.949	0.949
	75	0.956	0.870	0.923	0.960	0.968	0.968
	100	0.974	0.974	0.974	0.974	0.974	0.974
d695	5	0.0332	0.167	0.203	0.440	0.538	0.556
	10	0.0370	0.257	0.254	0.567	0.670	0.690
	25	0.208	0.405	0.510	0.743	0.849	0.863
	50	0.335	0.617	0.803	0.879	0.952	0.952
	75	0.602	0.821	0.937	0.946	0.965	0.965
	100	0.966	0.966	0.966	0.966	0.966	0.966
p22810	5	0.0333	0.174	0.450	0.659	0.691	0.759
	10	0.0347	0.186	0.608	0.764	0.796	0.856
	25	0.0544	0.398	0.769	0.885	0.900	0.940
	50	0.181	0.830	0.912	0.949	0.949	0.968
	75	0.600	0.916	0.964	0.969	0.969	0.973
	100	0.973	0.973	0.973	0.973	0.973	0.973
p34392	5	0.0307	0.312	0.683	0.798	0.843	0.859
	10	0.0341	0.331	0.766	0.857	0.893	0.898
	25	0.0602	0.470	0.846	0.919	0.940	0.942
	50	0.533	0.492	0.921	0.950	0.963	0.967
	75	0.547	0.906	0.943	0.965	0.972	0.972
	100	0.972	0.972	0.972	0.972	0.972	0.972
p93791	5	0.00542	0.118	0.559	0.715	0.748	0.748
	10	0.0249	0.235	0.618	0.791	0.822	0.822
	25	0.0507	0.459	0.742	0.883	0.908	0.908
	50	0.340	0.619	0.902	0.945	0.960	0.960
	75	0.584	0.927	0.957	0.969	0.974	0.974
	100	0.976	0.976	0.976	0.976	0.976	0.976

Table 3. Experimental results. 1 - only test scheduling, 2 - test scheduling and considering defect probability (dp), 3 - test scheduling considering dp and fault coverage (fc), 4 - test vector selection and test scheduling considering dp and fc at one TAM, 5 - as in 4 but two TAMs, 6 - as in 4 but three TAMs.