Test data truncation for test quality maximisation under ATE memory depth constraint

E. Larsson and S. Edbom

Abstract: Testing is used to ensure production of high quality integrated circuits. High test quality implies the application of high quality test data; however, technology development has led to a need to increase test data volumes to ensure high test quality. The problem is that the high test data volume leads to long test application times and high automatic test equipment memory requirement. For a modular core-based system-on-chip, a test data truncation scheme is proposed, that selects test data for each module in such a way that the system test quality is maximised while the selected test data are guaranteed to overcome constraints on time and memory. For test data selection, a test quality metric is defined based on fault coverage, defect probability and number of applied test vectors, and a scheme that selects the appropriate number of test vectors for each core, based on the test quality metric, defines the test architecture and schedules the transportation of the selected test data volume on the test access mechanism such that the system's test quality is maximised. The proposed technique has been implemented, and the experimental results, produced at reasonable CPU times on several ITC'02 benchmarks, show that high test quality can be achieved by careful selection of test data. The results indicate that the test data volume and test application time can be reduced to about 50% while keeping a high test quality.

1 Introduction

Technology has made it possible to develop integrated circuits (ICs) where a complete system, with an enormous number of transistors, which are clocked at an immense frequency and partitioned into a number of clock-domains, is placed on a single die. Even as such highly advanced system chips or system-on-chip (SOC) are designed, the electronic design automation (EDA) tools must keep up; to enable the design of a highly advanced system with a reasonable effort in a reasonable time. New design methodologies are under constant development. At the moment, a modular design approach in which modules are integrated into a system shows promise. The advantage of such an approach is that pre-designed and pre-verified modules, blocks of logic or cores, with technology-specific details, can at a reasonable time and effort be integrated into a system. The core provider designs the cores, and the system integrator selects the appropriate cores for the system where the cores may originate from previous in-house designs or from different core vendors (companies). The cores can be delivered in various formats. In general, they can be classified as soft cores, firm cores and hard cores. Soft cores are general high-level specifications where the system integrator can, if necessary, apply modifications. Hard cores are gate-level specifications where, if any, only minor modifications are possible. Firm cores are somewhere between soft cores and hard cores. Soft cores allow more flexibility compared with hard cores. The main advantage is that the system integrator can modify a soft core. On the other hand, hard cores can be made highly protected by the core provider, which often is desirable.

A produced chip is tested to determine if it is faulty or not. In the test process, a number of test vectors, stored in the automatic test equipment (ATE), are applied to the chip under test. If the produced test response from the applied vectors corresponds to the expected response, the chip is considered to be fault-free and can be shipped. However, testing complex chips is becoming a problem, and one major problem is the increasing test data volume. Currently, the test data volume increases faster than the number of transistors in a design [1]. The increasing test data volume is due to (1) high number of fault sites because of the large numbers of transistors, (2) new defect types introduced by nanometer process technologies and (3) faults related to timing and delay since systems have higher performance and make use of multiple-clock domains [1].

The problem with high test data volume is that it leads to long test application times and high ATE memory requirements. The cost of test is highly related to test application time. Test time in a tester is purchased; hence the given test cost budget must be kept. It is also known that the purchase of a new ATE with higher memory capabilities is costly; hence, it is desirable to make use of the existing ATE instead of investing in a newone.

Vranken *et al.* [1] discussed three alternatives to make the test data fit the ATE: (1) test memory reload – the test data are 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 do not fit the ATE are simply not applied – leads to reduced test quality and (3) test data compression – the test stimuli is compressed – however, does not guarantee that the test data will fit the ATE. As test memory reload is not practical,

doi:10.1049/iet-cdt:20050209

Paper first received 13th December 2005 and in revised form 17th September 2006

The authors are with the Embedded Systems Laboratory, Department of Computer Science, Linköpings University, Sweden

E-mail: erila@ida.liu.se

the alternatives are test data truncation and test data compression. Test data compression does reduce the problem but does not make sure that the test data will fit the ATE memory. Test data truncation is an alternative to make sure that the test cost constraints: ATE memory requirement and test application time are overcome. The draw-back with test data truncation is lower test quality. However, ITRS roadmap states 'relaxing the requirement of 100% correctness in both transient and permanent failures of signals, logic values, devices, or interconnects may reduce the cost of manufacturing, verification, and testing' [2]. Hence, there is a need to explore the test quality against test cost.

The test data must also be organised or scheduled in the ATE. A recent industrial study showed that by making use of test scheduling, the test data were made to fit the ATE [3]. The study demonstrated that the ATE memory limitation is a real and critical problem. The basic idea in test scheduling is to reduce the amount of idle bits to be stored in the ATE, and therefore scheduling must be considered in combination with the test data truncation scheme. Further, when discussing memory limitations, the ATE memory depth in bits is equal to the maximal test application time for the system in clock cycles [4]. Hence, the memory constraint is seen as a time constraint.

This paper focuses on test data truncation, where the aim is a technique that maximises test quality while making sure that the test data volume fits the ATE memory and the test application time overcomes the time constraint. We assume that given a core-based design and for each core the defect probability, the maximal fault coverage when all its test vectors have been applied, and the size of the test set (the number of test vectors) are given.

We define for a core, a core test quality (CTQ) metric, and for the system, a system test quality (STQ) metric. The CTQ metric reflects that test data should be selected for 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, and STQ combines all CTQs to a system quality metric. For the fault coverage function, we make use of an estimation function. Fault simulation can be used to extract the 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 intellectual property (IP) protection, for instance. McLaurin and Potter discuss ways to select test vectors for a core [5]. The test vectors in a test set can be applied in any order. However, regardless of the order, it is well known in the test community that the first test vectors detect a higher number of faults compared to the last applied test vectors and that the function fault coverage against number of test vectors has an exponential/logarithmic behaviour. We therefore assume that the fault coverage over time (number of applied test vectors) for a core can be approximated to an exponential/logarithmic function.

We integrate the test data selection with test scheduling and test access mechanism (TAM) design in order to verify that the selected test data actually fit the ATE memory. We have implemented our technique and we have made experiments on several ITC'02 benchmarks to demonstrate that high test quality can be achieved by applying only a subset of the test data. The results indicate that the test data volume and the test application time can be reduced to 50% whereas the test quality remains high. 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 minimise the test application time.

The advantage of our technique is that given a core-based system and for each core a test set per core, a number on maximal fault coverage and defect probability, we can select test data for the system, design the TAM and schedule the selected test data in such a way that the test quality is maximised and the selected test data fit the ATE memory or overcome the time constraint. In this paper, we assume a single test per core. However, the technique can easily be extended to allow multiple tests per core by introducing constraint considerations in the scheme.

2 Related work

Test scheduling and test data compression are approaches proposed to address the high test data volumes that must be stored in the ATE in order to test SOCs. The basic principle in test scheduling is to organise the test bits in the ATE in such a way that the number of introduced so-called idle bits (not useful bits) is minimised. The gain is reduced test application time and a reduced test data volume. A scheduling approach depends highly on the test architecture in the system. Examples of test architectures are the AMBA test bus [6], the test bus [7] and the TestRail [8].

Iyengar *et al.* [9] proposed a technique to partition the set of scan chain elements (scan chains and wrapper cells) at each core into wrapper scan chains and connect them to TAM wires in such a way that the total test time is minimised. 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 are scheduling techniques that make use of an abort-on-fail strategy; the testing is terminated as soon as a fault is detected. The idea is that as soon as a fault is present, the chip is faulty and therefore the testing can be terminated. Huss and Gyurcsik [10] proposed a sequential technique making use of a dynamic programming algorithm for ordering the tests, whereas Milor and Sangiovanni-Vincentelli [11] presented a sequential technique based on selection and ordering of test sets. Jiang and Vinnakota [12] proposed a sequential technique, where the information about the fault coverages provided by the tests are extracted from the manufacturing line. For SOC designs, Larsson *et al.* [13] proposed a technique based on ordering of tests, considering different test bus structures, scheduling approaches (sequential and concurrent) and test set assumptions (fixed test time and flexible test time). The technique takes defect probability into account; however, the probability of detecting a fault remains constant through the application of a test.

Several compression schemes have been used to compress the test data [14–19] . For instance, Ichihara *et al.* [20] used statistical codes, Chandra and Chakrabarty [21] made use of Golomb codes, Iyengar *et al.* [22] explored the use of run-length codes, Chandra and Chakrabarty [23] tried frequency-directed run-length codes, and Volkerink *et al.* [24] have investigated the use of Packet-based codes.

All these approaches (test scheduling and test data compression techniques) reduce the ATE memory requirement. In the case of test scheduling, effective organisation means that both the test time and needed test data volume are reduced, and in the case of test data compression, less test data are required to be stored in the ATE. The main advantage of these two approaches is that the highest possible test quality is reached since the whole test data volume is applied. However, the main disadvantage is that these techniques do not guarantee that the test data volume fits the ATE. 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 maximised while the test data are guaranteed to fit the ATE memory or overcome the test time constraint.

3 Problem formulation

We assume that a core-based system with n cores denoted by i is given, and for each core i in the system, the following are given:

- sc_{ij} – length of scanned element j at core i is given where $j = 0, \ldots, m$ is the number of scanned elements
- wi_i – number of input wrapper cells
- wo_i – number of output wrapper cells
- wb_i – number of bidirectional wrapper cells
- tv_i – number of test vectors
- fc_i – fault coverage reached when all the tv_i test vectors are applied
- pp_i – pass probability per core
- dp_i – 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 an 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 denoted by j each of width $W_{tam} = \{w_1, w_2, \ldots, w_k\}$ in such a way that

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

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) [4], the memory constraint is actually a time constraint τ_{max}

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

Our problem is to:

- select the number of test vectors (stv_i) for each core i,
- partition the given TAM width W_{tam} into no more than k TAMs,
- determine the width of each TAM (w_j), $j = 1, \ldots, k$,
- 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), TAM design and the test scheduling should be done in such a way that the test quality of the system (defined in Section 4) is maximised while the memory constraint (M_{max}) (time constraint τ_{max}) is overcome.

4 Test quality metric

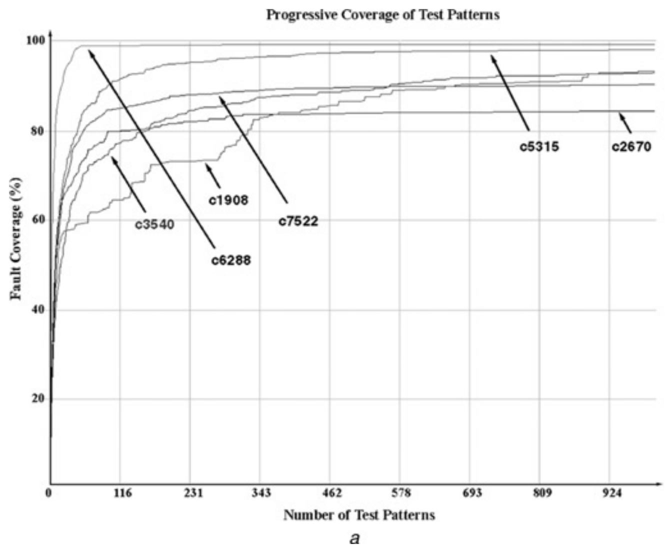
We need a test quality metric to (1) select test data for each core and (2) measure the final system test quality. In this section, we describe the metric where we take the following parameters into account to measure test quality:

- defect probability
- fault coverage
- 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 than to select test data for a core with a low defect probability since the core with high defect probability is more likely to hold a defect.

The possibility to detect faults depends on the fault coverage against 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 detects. 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. A core provider may want to protect the core, which makes fault simulation impossible. We therefore make use of an estimation technique. It is known that the fault coverage does not increase linearly over the number of applied test vectors. For instance, Fig. 1*a* shows the fault coverage for a set of ISCAS benchmarks. The following observation can be made: the curves have an exponential/logarithmic behaviour (as in Fig. 1*b*). This can also be observed in the results by others [25, 26]. We therefore assume that the fault coverage after applying stv_i test vectors for core i can be estimated as (Fig. 1*b*)

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



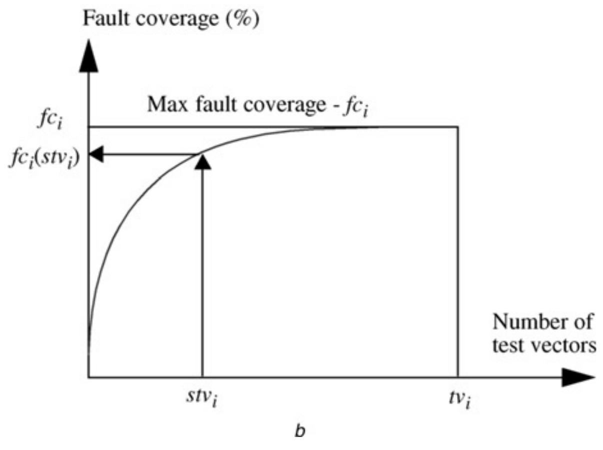


Fig. 1 *Fault coverage against number of test vectors*

a For a set of ISCAS designs
b Estimated as an exponential function

where the slopeConst is given as follows

$$\text{slopeConst} = \frac{\log(\text{tv}_i + 1)}{\text{fc}_i} \tag{4}$$

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

For a system, we assume that the test quality can be estimated as

$$\text{P(we find a defect|we have a defect in the SOC)} \tag{5}$$

The test quality describes the probability of finding a defect when we have the condition that the SOC has one defect. By introducing this probability, we find a way to measure the probability of finding a defect if a defect exists in the SOC and hence the test quality. However, it is important to note that our metric only describes the test quality, and hence we are not introducing any assumptions about the number of defects in the SOC.

In order to derive an equation for the test quality using information about defect probability, fault coverage and the number of test vectors, we make use of definitions from basic probability theory [27]:

Definition 1: If A and B are independent events $=>$ $\text{P}(A \cap B) = \text{P}(A)\text{P}(B)$.

Definition 2: If $A \cap B$ is the empty set $\varnothing =>$ $\text{P}(A \cup B) = \text{P}(A) + \text{P}(B)$.

Definition 3: $\text{P}(A \cap B) = \text{P}(A)\text{P}(B|A)$, where $\text{P}(B|A)$ is the probability of B conditioned on A.

Furthermore, we assume (Section 3) that the quality of a test set (a set of test vectors) for a core i is composed by the following:

- fault coverage fc_i
- probability of defect dp_i

Since the number of applied test vectors indirectly has an impact on the fault coverage, we define for each core i:

- stv_i is the selected number of test vectors
- $\text{fc}_i(\text{stv}_i)$ is the fault coverage after stv_i test vectors have been applied

We assume the following:

- dp_i and fc_i are independent events

Since we assume one defect in the system when we introduced test quality (5), we can only have one defect in a core at a time in the system. Therefore we can say:

- The intersection of any of the events dp_i is the empty set $\varnothing$

For a system with n cores, we can now derive STQ (system test quality) from (5) by using Definitions 1, 2 and 3

$$\text{STQ} = \text{P(defect detected in the SOC | defect in the SOC)}$$

$$\Rightarrow \frac{\text{P(defect detected in the SOC} \cap \text{defect in the SOC)}}{\text{P(defect in the SOC)}}$$

$$\Rightarrow \frac{\sum_{i=1}^{n} \text{P(defect detected in core } i \cap \text{defect in core } i)}{\text{P(defect in the SOC)}}$$

$$\Rightarrow \frac{\sum_{i=1}^{n} \text{P(defect detected in core } i)\text{P(defect in core } i)}{\text{P(defect in the SOC)}}$$

$$\Rightarrow \frac{\sum_{i=1}^{n} \text{dp}_i \times \text{fc}_i(\text{stv}_i)}{\sum_{i=1}^{n} \text{dp}_i} \tag{6}$$

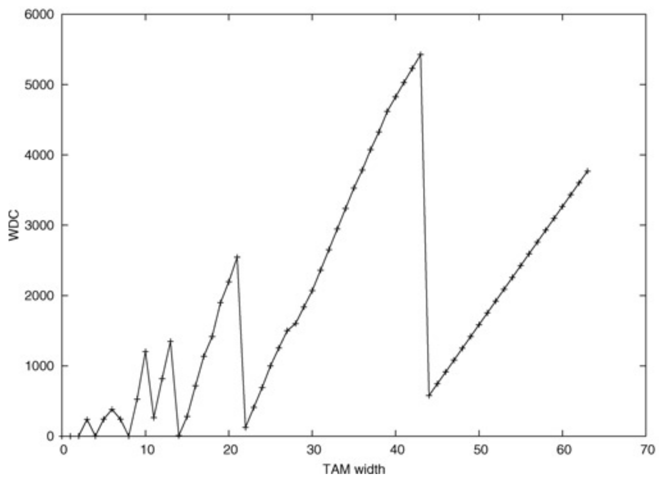
From STQ, we can derive the quality metric for a single core, the CTQ (core test quality) is

$$\text{CTQ} = \sum_{i=1}^{n} \text{dp}_i \times \text{fc}_i(\text{stv}_i) \tag{7}$$

5 Test vector selection, test scheduling and TAM design

In this section, we describe our technique to optimise test quality by selecting test vectors for each core, design the TAM and schedule the selected vectors for an SOC under the time constraint given by the ATE memory depth ((2) and [4]). We assume that given is a system as described in Section 3, and 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 one after the other [7]. We make use of the test quality metric defined in Section 4.

The scanned elements (scan-chains, input cells, output cells and bidirectional cells) at a core have to be configured into a set of wrapper chains, which are to be connected to a corresponding number of TAM wires. The wrapper scan chains, which are to be connected to the TAM wires, w_j, should be as balanced as possible, and we make use of the Design_wrapper algorithm proposed by Iyengar *et al.* [9].



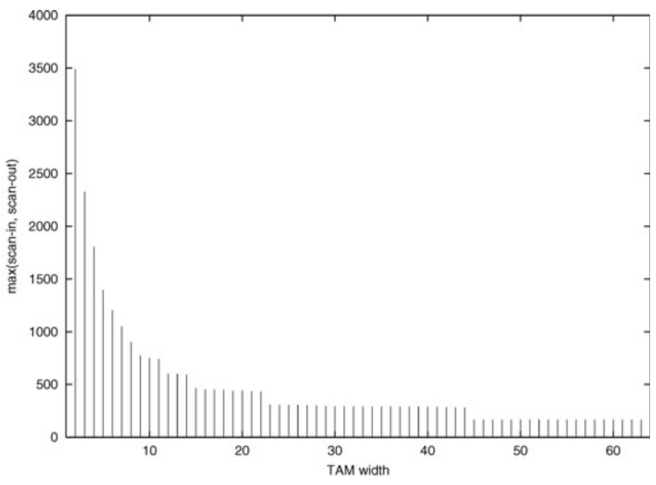


Fig. 2 *Variation of WDC and maximum value of scan-in and scan-out lengths at different TAM widths at core 1 (p93791)*
a WDC
b max(scan-in, scan-out)

```
1. Given:
   $\tau_{max}$ - the upper bound on 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. 10)
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. 8)
8.         Compute $CTQ_i$ assuming $stv_i=stv_i+1$ (Eq. 6)
9.     end
10.    for core with highest $CTQ/\tau(w_j,1)$ and $stv_i<tv_i$ //best contribution to increase STQ
11.        $stv_i=stv_i+1$
12.    for all cores where $stv_i>0$ begin// some selected vectors
13.        Vectors for each core are grouped and TAT is computed
14.        if a TAM is full ($<\tau_{max}$) - mark TAM as unavailable.
15.    end
16. end
17. Compute and return $STQ$ (Eq. 6).
18. end
```

Fig. 3 *Test vector selection and test scheduling algorithm*

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 for core i is given by [9]

$$\tau_i(w, \mathrm{tv}) = (1 + \max(\mathrm{si}_i(w), \mathrm{so}_i(w))) \times \mathrm{tv} + \min(\mathrm{si}_i(w), \mathrm{so}_i(w)) \qquad (8)$$

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

We need a technique to partition the given TAM width W_{tam} into a number of TAMs k and to determine which core should be assigned to which of the designed TAMs. The number of different ways we can assign n cores to k TAMs grows with k^n, and therefore the number of possible alternatives will be huge. We also need a technique to guide the assignment of cores to the TAMs. We make use of the fact that Iyengar *et al.* [9] made use of, which is 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 utilisation) for a core i at a TAM of width w as

$$\mathrm{TWU}_i(w) = \max(\mathrm{si}_i(w), \mathrm{so}_i(w)) \times w \qquad (9)$$

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

$$\mathrm{WDC}_i = \mathrm{TWU}_i(w) - \mathrm{TWU}_i(1) \qquad (10)$$

For illustration of the variations in the number of ATE idle bits, we plot in Fig. 2*a* the value of WDC for different TAM widths (number of wrapper chains), obtained by using core 1 of the ITC'02 benchmark p93791. We also plot the maximum value of the scan-in and scan-out lengths at

Table 1: Data for benchmark d695

	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 wi	0	32	207	34	36	38	62	77	35	35	28
Outputs wo	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	0	98	99	95	92	99	94	90	92	98	94
Max fault coverage fc_i (%)	0	93	99	98	96	96	99	94	99	95	96

Table 2: Selected test vectors (%) for the cores in design d695 considering different scheduling techniques

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

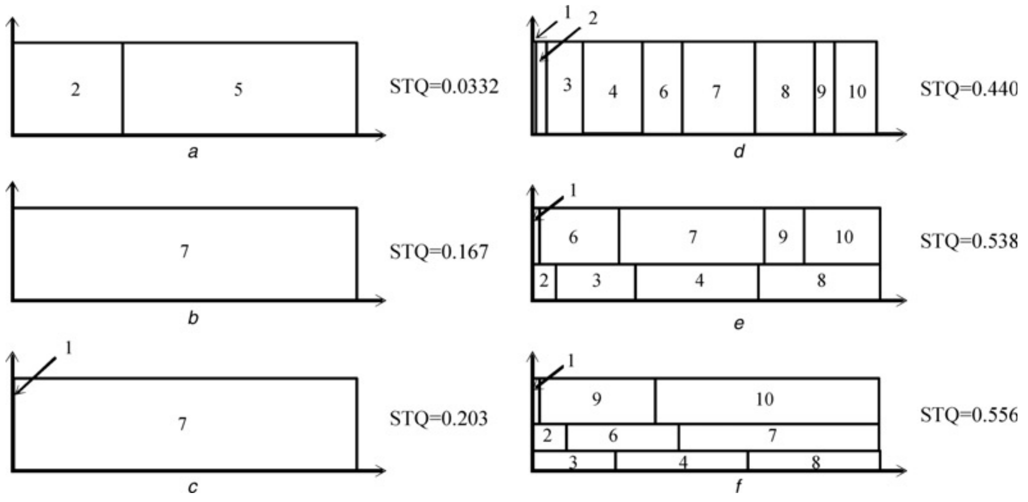


Fig. 4 *Results for different scheduling techniques*

a Test scheduling without test vector selection when defect probability and fault coverage are not considered
b Test scheduling considering defect probability
c Test scheduling considering defect probability and fault coverage
d Test scheduling using test vector selection and one TAM
e Test scheduling using test vector selection and two TAMs
f Test scheduling using test vector selection and three TAMs

various TAM widths for the previous design in Fig. 2*b*. In Fig. 2*b*, several TAM widths have the same test time. For a set of TAM widths with the same test time, a pareto-optimal point is the one with lowest TAM [9]. We can notice that the TAM widths having a low value of the WDC, and hence a small number of idle bits, correspond to the pareto-optimal points. Hence, we make use of WDC to guide the selection of wrapper chains at a core.

The algorithm for our scheme is outlined in Fig. 3. Given is a system, the upper bound on the test time ($\tau_{\max}$) and the TAM width (W_{tam}) distributed over k TAMs $w_1, w_2, \ldots, w_k$. Initially no test vectors are selected for any core ($\text{stv}_i = 0$ for all i) and the test time for the test schedule is zero ($\text{TAT} = 0$).

At line 4, the cores are assigned to TAMs and the while loop continues until $\tau_{\max}$ is overcome. In each iteration of the loop (lines 7 and 8), the algorithm explores for all cores the contribution in improving CTQ if an additional vector is added to a core. The test vector for a core that contributes most to improving CTQ is selected (line 11).

Note that the test vectors for a core might not be selected in order (line 13 in Fig. 3). For instance, in a system with two cores A and B, the first vector can be selected from core A, the second from core B and the third from core A. However, at scheduling (application), selected test vectors for each core are grouped and scheduled as a single set. If grouping is not applied, we would not benefit from the scan-in/scan-out pipelining.

The algorithm (Fig. 3) assumes a given TAM architecture (number of TAMs and their width). We have therefore added an outer loop that makes sure that we explore all possible TAM configurations.

5.1 Illustrative example

To illustrate the proposed technique for test scheduling and test vector selection, we make use of an example where the time constraint is set to 5% of the maximal test application time (the time when all available test vectors are applied). For the example, we make use of the ITC'02 benchmark [28] d695 with the data as in Table 1. As the maximal fault coverage for a core when all test vectors are applied and the pass probability per core are not given in the ITC'02 benchmarks, we have added these numbers. In order to show the importance of combining test scheduling, TAM design and test vector selection, we compare our proposed technique to a naive approach where we order the tests and assign test vectors according to the initial sorted order until the time limit (ATE memory depth) is reached.

Table 3: Pass probability and maximal fault coverage numbers for the cores in the used SOCs

		Core																																
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
d281	pp (%)	—	98	99	95	92	99	94	90	92																								
	fc (%)	—	98	97	95	98	98	96	99	97																								
d695	pp (%)	—	98	99	95	92	99	94	90	92	98	94																						
	fc (%)	—	93	99	98	96	96	99	94	99	95	96																						
p22810	pp (%)	98	98	97	93	91	92	99	96	96	95	93	91	92	93	99	99	99	95	96	97	93	99	96	98	99	92	91	91	93				
	fc (%)	95	99	97	98	94	99	99	97	95	97	97	99	99	94	97	94	99	98	94	95	99	99	95	98	95	99	99	97	98				
p34392	pp (%)	98	98	97	91	95	94	94	93	99	99	91	91	90	95	94	96	96	97	92	90													
	fc (%)	97	97	99	98	99	99	97	98	94	96	98	98	99	94	97	95	98	98	95	95													
p93791	pp (%)	—	99	99	97	90	91	92	98	96	91	94	93	91	91	90	99	98	97	99	99	99	90	99	90	98	92	96	95	91	90	96	99	99
	fc (%)	—	99	95	98	98	99	97	99	95	96	97	99	99	94	98	94	97	97	95	95	99	98	96	98	94	99	99	98	99	97	98	99	94

Table 4: Comparison of different TAM widths using ITC'02 benchmark p93791

SOC	% of max test time	Technique 1 STQ	Technique 2 STQ	Technique 3 STQ	Technique 4 STQ	Technique 5 STQ	Technique 6 STQ
p93791 TAM width 16	5	0.00542	0.118	0.560	0.719	0.720	0.720
	10	0.0248	0.235	0.618	0.793	0.796	0.796
	25	0.0507	0.458	0.747	0.884	0.885	0.885
	50	0.340	0.619	0.902	0.945	0.945	0.945
	75	0.588	0.927	0.958	0.969	0.969	0.969
	100	0.976	0.976	0.976	0.976	0.976	0.976
p93791 TAM width 32	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
p93791 TAM width 64	5	0.00535	0.118	0.499	0.703	0.752	0.752
	10	0.00606	0.235	0.567	0.780	0.827	0.827
	25	0.0356	0.461	0.739	0.878	0.918	0.918
	50	0.335	0.620	0.901	0.944	0.965	0.965
	75	0.566	0.927	0.961	0.969	0.975	0.975
	100	0.976	0.976	0.976	0.976	0.976	0.976

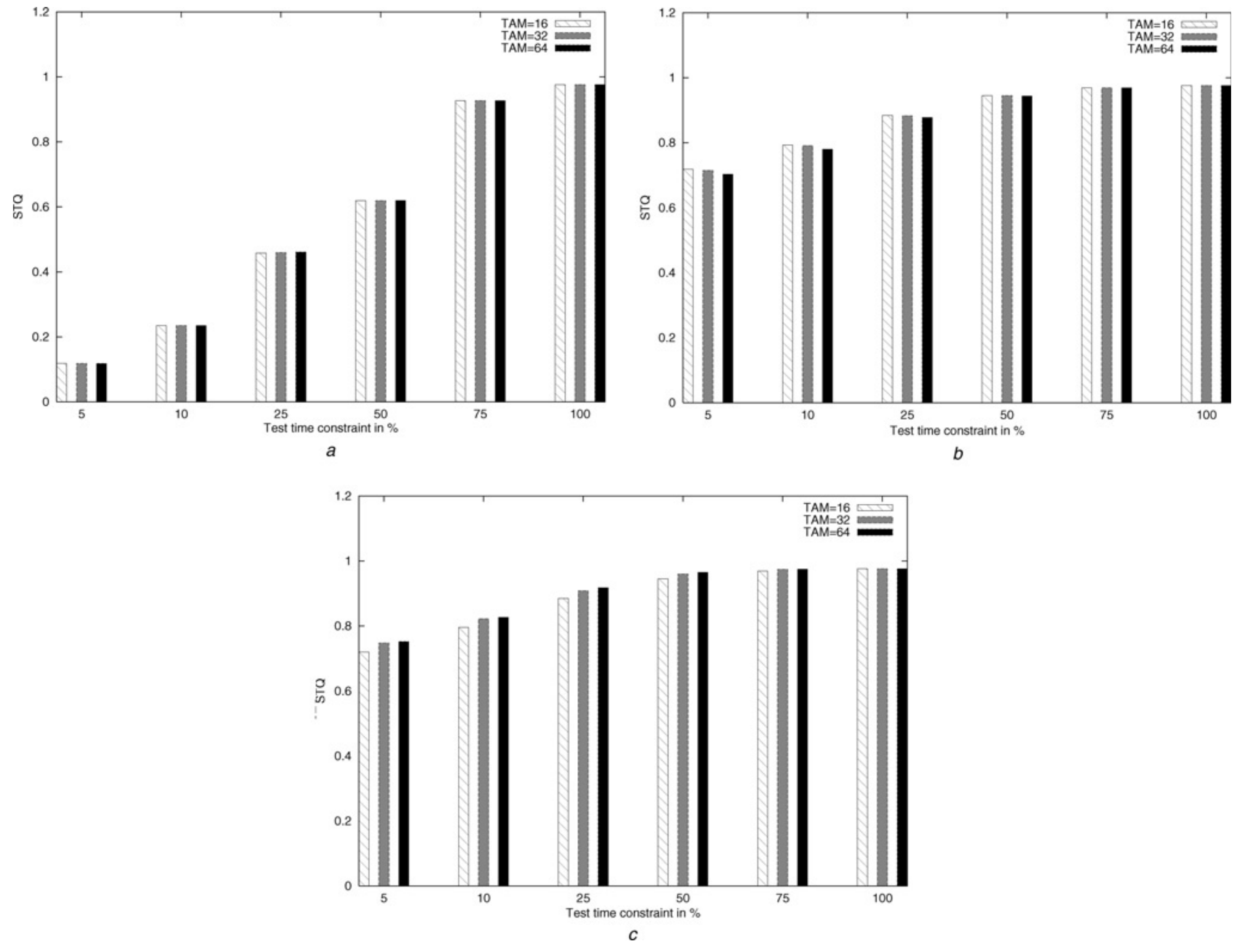


Fig. 5 *Comparing STQ at TAM width 16, 32 and 64 on design p93791*

a For Technique 2
b For Technique 4
c For Technique 6

For this naive approach, we consider three different techniques:

1. Sorting when defect probability and fault coverage are not considered (Technique 1).
2. Sorting when defect probability is considered but not fault coverage. The cores are sorted in descending order according to defect probability (Technique 2).
3. Sorting when defect probability in combination with fault coverage is considered. In this technique, we make use of the STQ equation (6) to find a value of the test quality for each core. The cores are then sorted in descending order according to test quality per clock cycle. The sorting constant is described in (11) (Technique 3)

$$\text{sortConst} = \frac{\text{dp}_i \times \text{fc}_i(\text{tv}_i)}{\tau(w, \text{tv}_i) \times \sum_{i=1}^{n} \text{dp}_i} \tag{11}$$

For our technique, we consider three cases where we divide the TAM into one (Technique 4), two (Technique 5) or three test buses (Technique 6). The selected test data volume per core for each of the six scheduling techniques is reported in Table 2 and the test schedules with the corresponding STQ are presented in Fig. 4. Fig. 4*a* illustrates the case when no information about defect probability and fault coverage is used in the test ordering. As seen in the figure, such technique produces a schedule with an extremely low system test quality (STQ). By making use of the information on defect probability (Fig. 4*b*), respective defect probability and fault coverage (Fig. 4*c*) in the ordering, we can improve the test quality significantly. Although it is possible to increase the STQ by using an efficient sorting technique, we are still not exploiting the fact that the first test vectors in a test set detect more faults than the last test vectors. In Figs. 4*d–f*, we make use of this information as we are using our proposed technique for test scheduling and test vector selection. We note that it is possible to further improve the STQ by dividing the TAM into several test buses (Figs. 4*e–f*).

5.2 Optimal solution for single TAM

The above algorithm can easily be improved to produce an optimal solution in the case of a single TAM. The above

Table 5: Experimental results

SOC	% of max test time	Technique 1 STQ	Technique 2 STQ	Technique 3 STQ	Technique 4 STQ	Technique 5 STQ	Technique 6 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

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

algorithm aborts the assignment of test vectors immediately when the time constraint (memory constraint) is reached – a selected test vector cannot be assigned since it violates the constraint. However, test vectors from other cores (not from the core that violates the time constraint) could have been selected while making sure that they do not violate the ATE constraint. Note that the selection of test vectors is based on a monotonically increasing function. The test vector that contributes most to the test quality is first selected. That process continues on an updated list until the constraint is reached. In the case of a single TAM, the scheme is optimal.

6 Experimental results

The aim with the experiments is to demonstrate that the test quality can be kept high by using the proposed scheme. We have implemented the technique described earlier, and we have in the experiments made use of five ITC'02 benchmarks [28], d281, d695, p22810, p34392 and p93791.

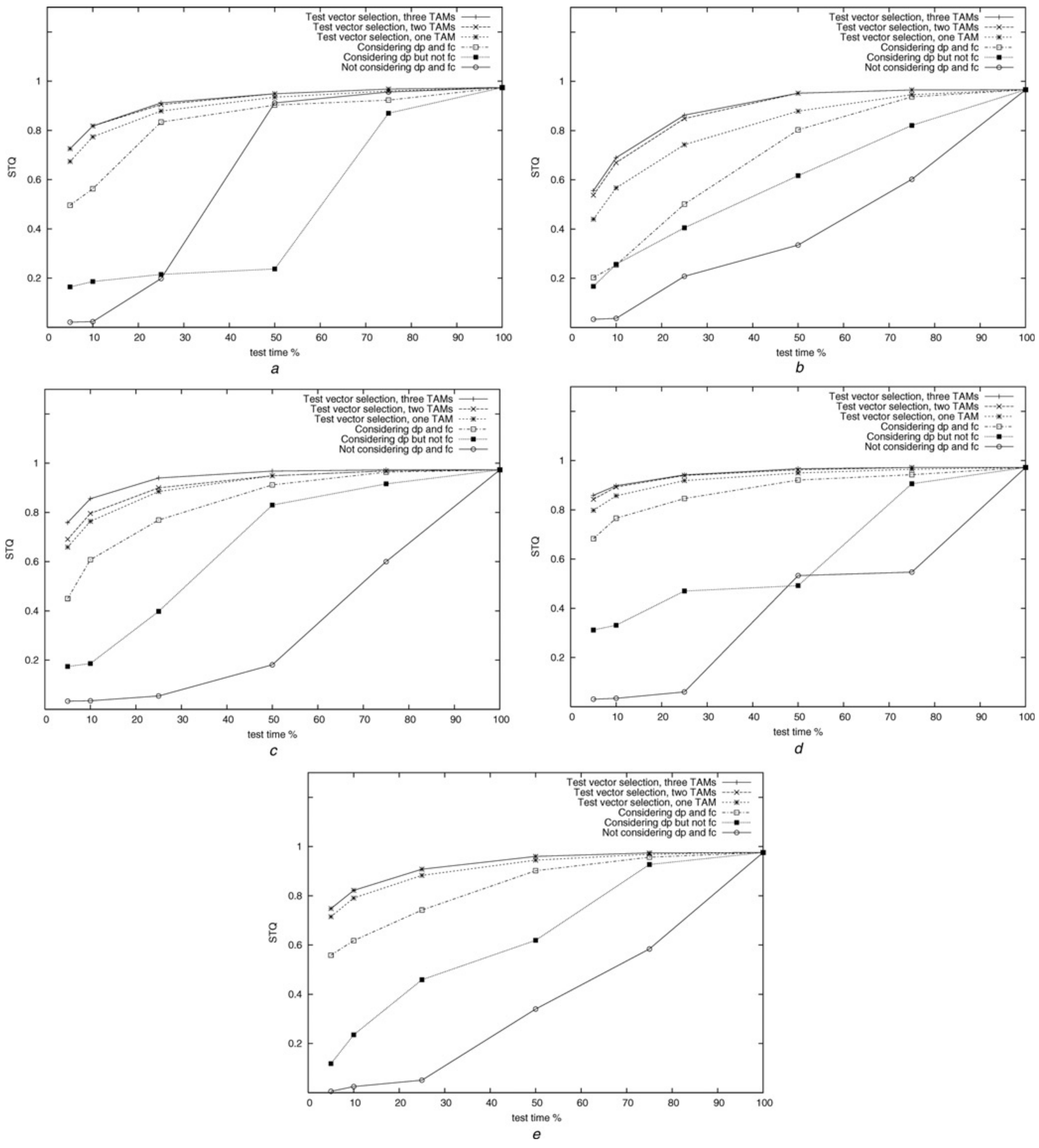


Fig. 6 *Experimental results*

a d281
b d695
c p22810
d p34392
e p93791

Given for each core in these benchmarks are the number of test vectors, the number of scanned elements (number and length of the scan-chains), the number of input pins, bidirectional pins and output pins. The netlists for the ITC'02 benchmarks are not publicly available, and therefore we have, in order to perform experiments, added for each core a pass probability and a maximal fault coverage number when all its test vectors are applied (Table 3). Note that core 0 in d281, d695 and p93791 does not contain any logic; hence we have not assigned a pass probability and a fault coverage.

In order to have a memory (time) constraint from the ATE, we performed for each design a schedule where all vectors are applied and that test application time refers to 100%. We have performed experiments at various ATE memory depths constraints (equal to time constraints (2) and [4]) and these constraints are set as a percentage of the time it would take to apply all test vectors.

We identify six techniques:

1. Test scheduling when defect probability or fault coverage is not considered and testing is aborted at $\tau_{\max}$ – Technique 1.
2. Test scheduling when defect probability is considered but not fault coverage and testing is aborted at $\tau_{\max}$ – Technique 2.
3. Test scheduling when defect probability as well as fault coverage is considered and testing is aborted at $\tau_{\max}$ – Technique 3.
4. Test scheduling and test-vector selection when defect probability and fault coverage are considered, using one TAM – Technique 4.
5. Test scheduling and test-vector selection when defect probability and fault coverage are considered, using up to two TAMs – Technique 5.
6. Test scheduling and test-vector selection when defect probability and fault coverage are considered, using up to three TAMs – Technique 6.

In the first experiment, we analyse the importance of TAM width. We have made experiments on benchmark p93791 at TAM width 16, 32 and 64 at time constraint 5, 10, 25, 50, 75 and 100% of the test application time if all test data are applied. The results are presented in Table 4 and illustrated for Techniques 2, 4 and 6 in Fig. 5. The results show that the produced results (STQ) are at a given time constraint, rather similar at various TAM widths. Therefore for the rest of the experiments, we assume a TAM bandwidth W_{tam} of 32.

The results from the experiments on d281, d695, p22810, p34392 and p93791 are presented in Table 5 and also plotted in Fig. 6. In column 1 the design name is given, in column 2 the percentage of the test time is given, and in columns 3–8 the produced STQ is reported for each technique (1–6). The computational cost for every experiment is in the range of a few seconds to a few minutes.

From the experimental results collated in Table 5 and Fig. 6, we learn that the STQ value increases with the time constraint (a larger ATE memory results in a higher STQ), which is obvious. It is also obvious that the STQ value for a design is the same at 100% test time, all test data are applied. From the results, we also see that test set selection improves the test quality when comparing STQ at the same test time limit. That is, Techniques 4, 5 and 6 have significant higher STQ value compared to Techniques 1, 2 and 3. 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, TAM design and test scheduling, to reduce the test application time while keeping the test quality high. Also, we have selected rather high pass probabilities and rather high fault coverage as these numbers are not publicly available. For designs with lower pass probabilities and lower fault coverage, and also, for designs where the variations in these numbers are higher, our technique becomes more important.

7 Conclusions

The technology development has made it possible to design extremely advanced chips where a complete system is placed on a single die. The requirement to test these system chips increases, and especially, the growing test data volume is becoming a problem. Several test scheduling techniques have been proposed to organise the test data in the ATE in such a way that the ATE memory limitation is not violated, and several test compression schemes have been proposed to reduce the test data volume. However, these techniques do not guarantee that the test data volume fits 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 maximised while the constraint on ATE memory depth is overcome. We have defined a test quality metric based on defect probability, fault coverage and the number of applied vectors that is used in the proposed test data selection scheme. We have implemented our technique and the experiments on several ITC'02 benchmarks [28] 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.

8 Acknowledgment

This work has been partially supported by the Swedish Foundation for Strategic Research (SSF) under the Strategic Integrated Electronic Systems Research (STRINGENT) Program.

9 References

1 Vranken, H., Hapke, F., Rogge, S., Chindamo, D., and Volkrink, E.: 'ATPG padding and ATE vector repeat per port for reducing test data volume'. Proc. Int. Test Conference (ITC), Charlotte, NC, USA, 2003, pp. 1069–1078

2 International Technology Roadmap for Semiconductors (ITRS), 2003, http://public.itrs.net/Files/2003ITRS/Home2003.htm

3 Goel, S.K., Chiu, K., Marinissen, E.J., Nguyen, T., and Oostdijk, S.: 'Test infrastructure design for the Nexperia^{TM} Home Platform PNX8550 system chip'. Proc. Design, Automation and Test in Europe Conference (DATE), Paris, France, 2004, pp. 1530–1591

4 Iyengar, V., Goel, S.K., Marinissen, E.J., and Chakrabarty, K.: 'Test resource optimization for multi-site testing of SOCs under ATE memory depth constraints'. Proc. Int. Test Conference (ITC), Baltimore, USA, October 2002, pp. 1159–1168

5 McLaurin, T.L., and Potter, J.C.: 'On-the-shelf core pattern methodology for ColdFire(R) microprocessor cores'. Proc. Int. Test Conference (ITC), Atlantic City, NJ, USA, October 2000, pp. 1100–1107

6 Harrod, P.: 'Testing reusable IP–a case study'. Proc. Int. Test Conference (ITC), Atlantic City, NJ, USA, 1999, pp. 493–498

7 Varma, P., and Bhatia, S.: 'A structured test re-use methodology for core-based system chips'. Proc. Int. Test Conference (ITC), Washington, DC, USA, October 1998, pp. 294–302

8 Marinissen, E.J., Arendsen, R., Bos, G., Dingemanse, H., Lousberg, M., and Wouters, C.: 'A structured and scalable mechanism for test

access to embedded reusable cores'. Proc. Int. Test Conference (ITC), Washington, DC, USA, October 1998, pp. 284–293

9 Iyengar, V., Chakrabarty, K., and Marinissen, E.J.: 'Test wrapper and test access mechanism co-optimization for system-on-chip'. Proc Int. Test Conference (ITC), Baltimore, MD, USA, 2001, pp. 1023–1032

10 Huss, S.D., and Gyurcsik, R.S.: 'Optimal ordering of analog integrated circuit tests to minimize test time'. Proc. Design Automation Conference (DAC), San Francisco, CA, USA, June 1991, pp. 494–499

11 Milor, L., and Sangiovanni-Vincentelli, A.L.: 'Minimizing production test time to detect faults in analog circuits', *IEEE Trans. Computer-Aided Design Integrated Circuits Syst.*, 1994, **13**, (6), p. 796

12 Jiang, W.J., and Vinnakota, B.: 'Defect-oriented test scheduling', *Trans. Very-Large Scale Integration (VLSI) Syst.*, 2001, **9**, (3), pp. 427–438

13 Larsson, E., Pouget, J., and Peng, Z.: 'Defect-aware SOC test scheduling'. Proc VLSI Test Symposium (VTS), Napa Valley, CA, USA, April 2004, pp. 359–364

14 El-Maleh, A., and Al-Abaji, R.: 'Extended frequency-directed run length code with improved application to system-on-a-chip test data compression'. Proc. 9th IEEE Int. Conf. Electronics, Circuits and Systems, September 2002, pp. 449–452

15 Gonciari, P.T., Al-Hashimi, B.M., and Nicolici, N.: 'Variable-length input Hoffman coding for system-on-a-chip test', *Trans. Computer-Aided Design Integrated Circuits Syst.*, 2003, **22**, (6), pp. 783–796

16 Iyengar, V., and Chandra, A.: 'Unified SOC test approach based on test data compression and TAM design', *IEE Proc.-Comput. Digit. Tech.*, 2005, **152**, (1), pp. 82–88

17 Li, L., and Chakrabarty, K.: 'Test data compression using dictionaries with selective entries and fixed-length indices', *ACM Trans. Design Automation Electron. Syst.*, 2003, **8**, (4), pp. 470–490

18 Tehranipoor, M., Nourani, M., and Chakrabarty, K.: 'Nine-coded compression technique for testing embedded cores in SoCs', *IEEE Trans. Very Large Scale Integration (VLSI) Syst.*, 2005, **13**, (6), pp. 719–731

19 Wurtenberger, A., Tautermann, C.S., and Hellebrand, S.: 'Data compression for multiple scan chains using dictionaries with corrections'. Proc. Int. Test Conference (ITC), Charlotte, NC, USA, 2004, pp. 926–935

20 Ichihara, H., Ogawa, A., Inoue, T., and Tamura, A.: Dynamic test compression using statistical coding'. Proc. Asian Test Symposium (ATS), Kyoto, Japan, November 2001, pp. 143–148

21 Chandra, A., and Chakrabarty, K.: 'System-on-a-chip test data compression and decompression architectures based on Golomb codes', *Trans. CAD IC. Syst.*, 2001, **20**, (3), pp. 355–367

22 Iyengar, V., Chakrabarty, K., and Murray, B.: 'Built-in self-testing of sequential circuits using precomputed test sets'. Proc. VLSI Test Symposium (VTS), Princeton, NJ, USA, 1998, pp. 418–423

23 Chandra, A., and Chakrabarty, K.: 'Frequency-directed-run-length (FDR) codes with application to system-on-a-chip test data compression'. Proc. VLSI Test Symposium (VTS), Marina Del Rey, CA, USA, April 2001, pp. 42–47

24 Volkerink, E.H., Khoche, A., and Mitra, S.: 'Packet-based input test data compression techniques'. Proc. Int. Test Conference (ITC), Baltimore, MD, USA, October 2002, pp. 154–163

25 Lin, X., Rajski, J., Pomeranz, I., and Reddy, S.: 'On static compaction and test pattern ordering for scan designs'. Proc. Int. Test Conference, 2001, pp. 1088–1098

26 El-Maleh, A., and Osais, Y.: 'On test vector reordering for combinational circuits'. Proc. 16th Int. Conf. Microelectronics, 6–8 December 2004, pp. 772–775

27 Blom, G.: 'Sannolikhetsteori och statistikteori med tillämpningar', Studentlitteratur, 1989

28 Marinissen, E.J., Iyengar, V., and Chakrabarty, K.: 'A set of benchmarks for modular testing of SOCs'. Proc. Int. Test Conference (ITC), Baltimore, MD, USA, October 2002, pp. 519–528