An Integrated Technique for Test Vector Selection and Test Scheduling under Test Time Constraint

Stina Edbom and Erik Larsson

Department of Computer Science
Linköpings Universitet, Sweden
erila@ida.liu.se

Abstract[1]

The quality of test is highly related to the number of faults that can be detected during the testing (fault coverage) and the defect probability of each testable unit. High test quality is reached by applying an excessive number of good test vectors, however, such a high test data volume can be problematic to fit in the ATE's (automatic test equipment) limited memory. We therefore propose, for core-based designs, a scheme that selects test vectors for each core, and schedule the test vectors in such a way that the test quality is maximized under a given test time constraint given by the ATE memory depth.

1. Introduction

The test data volume required to test an IC (integrated circuit) is stored in an ATE (automatic test equipment). If the test data volume does *not* fit the ATE memory, the test cost increases either due to the division of test data into multiple parts, which enforce additional time consuming ATE re-loads, or to the need to purchase a new ATE with additional memory [10].

A way to make test data fit the ATE memory is to employ an effective organization of the application of test data. Several architectures have been proposed [1,6,8], as well as scheduling techniques [3,9]. Scheduling techniques taking defect probability into account have also been proposed [2,4,7]. Goel *et al*. showed that by using an efficient test scheduling technique compared to using a less effective one on a real industrial (Nexperia Home Platform PNX8550) the test data volume can be made to fit the ATE memory [9].

However, currently the increase in test data volume is even faster than the number of transistors on the IC test, and scheduling alone may not be sufficient to handle the problem [10]. Modern process technologies suffer from defects that are not detected with the standard stuck-at fault model. For instance, delay fault testing is required for timing faults, which means that the test data volume increases due to the need of additional vectors [9, 10].

In this paper, we explore techniques to reduce the test data volume for modular systems in order to make it fit the ATE memory while maximizing the test quality. In a modular design approach modules (cores, blocks of logic), are integrated to become a system, which is placed on a single die, a SOC (system-on-chip). The modules may be provided from different core providers; previous designs, designed from scratch, or bought from core vendors. Each core is tested by its test set (a set of test vectors).

The quality of a test set is determined by the fault coverage. If all test vectors are applied, a high number of faults can be checked, and the quality of the test is high. If less vectors are applied, the quality is reduced. Fault simulation can give the fault coverage curve for some modules, but it can be difficult to get the fault coverage curve for other modules (so called hard cores). However, it is known that the first vectors in a test of a module detects a higher number of faults compared to test vectors applied later. For the estimation of the fault coverage, we make use of the assumption that the fault detection (fault coverage) can be approximated to an exponential function. The defect probability for each core has also to be taken into account when optimizing for quality. Modules with high defect probability should be tested more extensively compared to modules with lower defect probability. We assume that the defect probabilities are collected from the production line and are known prior to scheduling.

We assume that given is an SOC system with a set of modules (cores, testable units), a test set per module, a defect probability per module, and a constraint on the test application time given by the memory depth of the ATE and the clock frequency. Our problem is to select the number of test vectors for each core and schedule the selected vectors in such a way that: (1) the given constraint on test application time (ATE memory) is met, and (2) the probability to detect faults are maximized. We solve the problems by proposing an integrated test vector selection and test scheduling technique. We demonstrate on an example the usefulness of the technique.

2. Problem Formulation

The quality of a test set (a set of test vectors) for a core is determined by the:

- fault coverage,
- defect probability, and
- number of applied test vectors.

If all test vectors in a test set are applied, long test time is required but the highest number of possible faults are checked. The test time can be reduced, if not all test vectors

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

are applied, however, the quality of the test is reduced since not all possible faults are checked. The possibility to detect faults depends on the fault coverage curve versus the number of test vectors. And it is commonly so that the first test vectors in a test set detects more faults than the last test vectors. From Figure 1 where the fault coverage over the number of test vectors is plotted for a set of ISCAS'89 designs, it is obvious that the initial test vectors detect a higher number of faults compared to test vectors applied at the end of the test. If vectors are to be removed, the last vectors should be removed in favor for the initial vectors at another module.

The defect probability has also to be taken into account when discussing test quality. A module with a high defect probability is more likely to hold a fault compared to a module with a low defect probability. If test vectors are to be removed, they should be excluded from modules with a low defect probability.

We assume that given is a core-based architecture with n cores $i=\{1..n\}$ and for each core i 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 achieved 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} and a upper-bound (time constraint) τ_{max} are given. The τ_{max} is given by the ATE memory depth and the clock speed of the ATE. The TAM bandwidth can be partitioned into a set of k TAMs each of width $W_{tam}=\{w_1, w_2, ..., w_k\}$ in such a way that:

$$W_{tam} = \sum_{i=1}^{k} w_i \qquad 1$$

and we assume sequential execution of tests on each TAM.

Our model supports systems with both soft cores - a set of flip-flops is given for each core (sc_{ij}=1 and m=number of scanned flip-flops) and hard cores - a set of scanned elements (scan-chains and wrapper cells) is given for each core (sc_{ij} is the length of scan-chain j at core i and m is the number of scan-chains).

For solving the problem of partitioning (grouping) scanned elements (scan-chains, input cells, output cells and bidirectional cells) at a core into a balanced number of w_j wrapper chains that can be connected to the w_j TAM wires at TAM j, we make use of the *Design_wrapper* algorithm proposed by Iyengar *et al*. [3]. For a wrapper-chain configuration where $si_i(w)$ ($so_i(w_j)$) is the longest wrapper scan-in (scan-out) chain, the test time $\tau_i(w_j, tv_i)$ for core i is

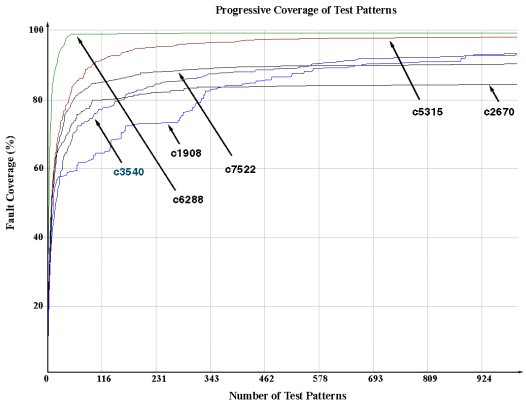

Figure 1. Fault coverage versus the number of applied number of test patterns for a set of ISCAS designs.

given by [3]:

$$= (1 + max(si_i(w_j), so_i(w_j))) \times tv_i + min(si_i(w_j), so_i(w_j)) \quad 2$$

where tv_i is the number of vectors and w_j is the TAM width.

Out of the factors that have an impact on the test quality (fault coverage fc_i, probability of a defect dp_i) it is possible to impact on the number of test vectors that are applied, which indirectly has an impact on the fault coverage. We therefore define for each core i:

- stv_i - selected number of test vectors, and
- $fc_i(stv_i)$ - fault coverage when stv_i vectors are applied.

The fault coverage varies over the number of applied test vectors. In some cases the fault coverage curve might be available. However, in some cases (due to core protection, limited design time etc) it might be more difficult. Hence, an estimation technique is required. Figure 1 shows the fault coverage for a set of ISCAS benchmarks. The following observation can be made: *the curves are similar to an exponential function* (Figure 2). We therefore assume that the fault coverage 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 so it passes the origin. For each core i we introduce the CTQ_i (core test quality) given 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:

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

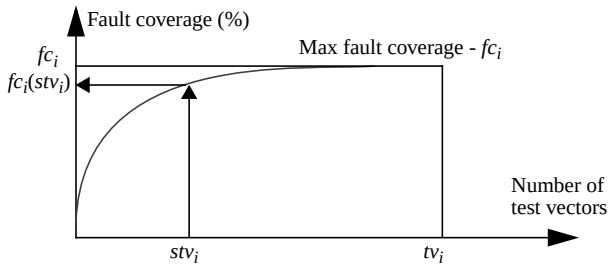

Figure 2. Fault coverage versus number of test vectors estimated as an exponential function.

The CTQ_i value depends on the defect probability (dp_i) and increases according to the exponential function fc_i (see Eq. 3 and 4) with the number of applied test vectors (stv_i).

Our problem is to:

- for a given W_{tam} find the number of k TAMs and their widths (w_1, ..., w_k),
- for each core i select the number of vectors (stv_i) and
- assign the selected test vectors to the designed TAMs,

in such a way that the test application time for the system meets τ_{max} and STQ is maximized.

3. Test Scheduling and Test Vector Selection

First, it is important to note that there is the difference between the order the test vectors are *selected* and the actual *execution order* of the test vectors. Our algorithm may select and assign the test vectors in one order, but when it comes to execution all selected vectors for each core are always executed as a single set for each core.

For a given value of the maximum number of TAMs (k), our algorithm finds the number of TAMs, the TAM widths, the assignment of the cores and the assignment of the test vectors in such a way that the test quality (STQ) is maximized (see Equation 6). In order to decide to which TAM a core should be assigned, we introduce WDC_i (wrapper design cost):

$$= max(si_i((w_j), so_i(w_j))) \times w_j - max(si_i(1), so_i(1)) \quad 7$$

The WDC_i reflects the imbalance cost for a wrapper design, and it is based on Eq. 2.

We assume that each core i is assigned to the TAM with a width that will give the least contribution to WDC. During test scheduling, we are always assigning test vectors to the core that will increase the CTQ per clock cycle the most. If such an assignment is impossible due to exceeding the total available test time at one of the TAMs, we consider the scheduling to be finished for this TAM and continue to select the best option that do not exceed the time limit for the remaining TAMs. We continue until test scheduling is finished for all TAMs (all TAMs are fully occupied). Since we want to find the solution with best system test quality, we repeat the algorithm described above for every possible set of TAM widths. The algorithm for our test scheduling technique is outlined in Figure 3.

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. 7)
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. 2)
8. Compute CTQ_i assuming $stv_i=stv_i+1$
9. **end**
10. **for core** with highest CTQ/$\tau(w_j,1)$ and $stv_i<tv_i$
11. $stv_i=stv_i+1$ //take one additional vector
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
17. **end**

Figure 3. Scheduling and vector selection algorithm.

Core i	1	2	3	4
Number of scan-chains	0	4	6	2
Scan-chain length sc_{ij}	-	8,7,7,7	32,32,32,32,32,1	9,9
Inputs w_i	233	45	214	32
Outputs w_o	140	52	228	32
Test vectors tv_i	2206	346	374	336
Pass probability pp_i	99	90	82	81
Max fault coverage fc_i (%)	95	98	99	45

Table 1. Data for the example design.

We make use of an example with data as in Table 1, is based on core 2, 4, 5 and 6 from the ITC'02 benchmark d281.

We illustrate in Figure 4 the following four alternatives:

1. Test scheduling without considering defect probability and fault coverage,
2. Test scheduling considering defect probability but not fault coverage,
3. Test scheduling considering defect probability and fault coverage,
4. Test scheduling and test vector selection when considering defect probability and fault coverage.

In this example we assume the time constraint τ_{max} to be 15% of the maximal test time, where the maximal test time is the time when all test vectors are applied. In option 1,2, and 3 where test set selection is not applied, the testing is terminated at the time constraint τ_{max}. As seen in Figure 4(a), option 1 results in a low system test quality (STQ). All available test time is spent on one single core with high pass probability (since defect probability is not taken into account), while the other cores are not given any test time at all. Option 2 and 3 in Figure 4 improves the test quality

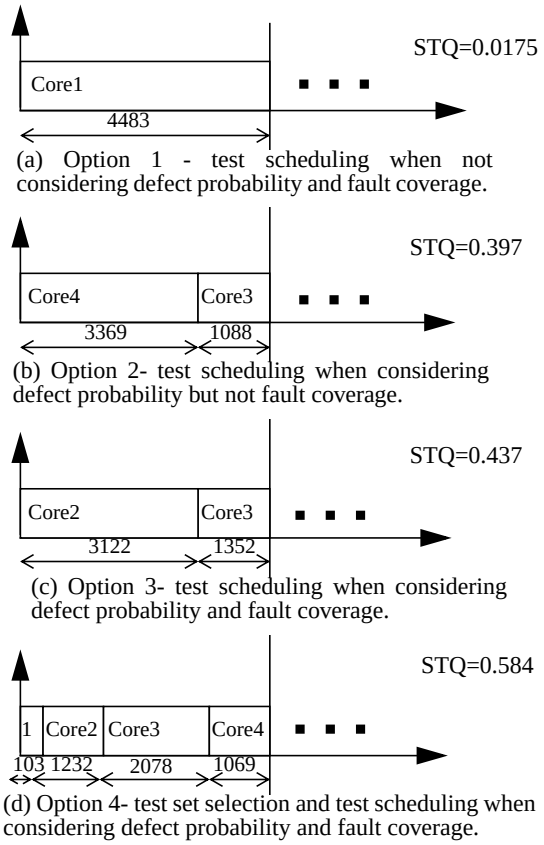

Figure 4. Test quality (STQ) for different techniques.

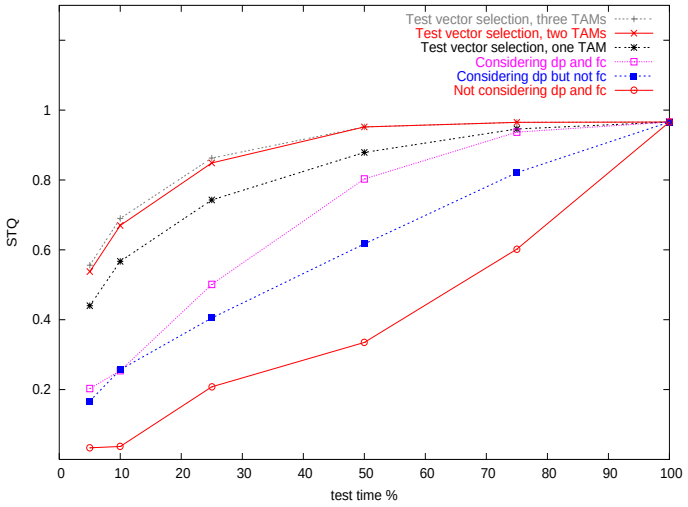

Figure 5. The STQ (system test quality) versus the test application time.

significantly by considering defect probability respective defect probability and fault coverage. By taking these parameters into consideration we avoid assigning a high number of test vectors to cores with high pass probability, but still we are wasting test time since we are not exploiting the fact that the first test vectors in a test set usually detects more faults than the last ones. A more efficient way is to considering test scheduling and test vector selection while taking both defect probabilities and fault coverage into account as in Figure 4 (d).

Results on the STQ for several approaches at several test time constraints are collected in Figure 5. Interesting to note is that the same quality (STQ) (0.7) using our approach at 10% of the test time (ATE memory) is achieved at 80% of the test time (ATE memory) using the simplest approach.

4. Conclusions

The test data volume of SOC designs increases at a an higher pace than the transistor count. It is therefore becoming a problem to fit the test data in the ATE memory. In this paper we have defined a system quality metric that depends on fault coverage and defect probability. We have also introduced an estimation technique for fault coverage versus test time since fault simulation might not always be possible to perform. We have proposed an integrated technique for test vector selection and test scheduling where defect probability and the fault coverage are taken into account. Our technique selects test vectors and schedule the tests in such a way that the test quality is maximized while the constraint on ATE memory depth (test time) is met. We have implemented our technique and an illustrative example shows that high test quality can be achieved at shorter testing times if test set selection is integrated with test scheduling.

References

[1] P. Harrod, “Testing Reusable IP-a Case Study”, *Proc. of ITC*, Atlantic City, NJ, USA, pp. 493-498, 1999.

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

[3] V. Iyengar *et al.*, “Test Wrapper and Test Access Mechanism Co-optimization for System-on-Chip”, *Proc. of ITC*, Baltimore, MD, USA, pp. 1023-1032, 2001.

[4] W.J. Jiang and B. Vinnakota, “Defect-Oriented Test Scheduling”,*Trans. on VLSI.,Vo. 9,No.3*,pp. 427-438, 2001.

[5] E. Larsson, J. Pouget, and Z. Peng, “Defect-Aware SOC Test Scheduling“, *Proc of VTS*, Napa Valley, CA, USA, 2004.

[6] 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, 1998.

[7] L. Milor and A. L. Sangiovanni-Vincentelli, “Minimizing Production Test Time to Detect Faults in Analog Circuits”, *Trans. on CAD of IC & Sys.,*Vol.13, No. 6, pp 796-813, 1994.

[8] 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, 1998.

[9] 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.

[10] 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.