Defect-Aware SOC Test Scheduling

Erik Larsson, Julien Pouget, Zebo Peng

Defect-Aware SOC Test Scheduling

Erik Larsson[+], Julien Pouget*, and Zebo Peng[+]

Embedded Systems Laboratory[+]
Department of Computer Science
Linköpings universitet
Sweden

LIRMM*
Montpellier 2 University
CNRS
France

Abstract[1]

In this paper we address the test scheduling problem for system-on-chip designs. Different from previous approaches where it is assumed that all tests will be performed until completion, we consider the cases where the test process will be terminated as soon as a defect is detected. This is common practice in production test of chips. The proposed technique takes into account the probability of defect-detection by a test in order to schedule the tests so that the expected total test time will be minimized. We investigate different test bus structures, test scheduling strategies (sequential scheduling vs. concurrent scheduling), and test set assumptions (fixed test time vs. flexible test time). We have also made experiments to illustrate the efficiency of taking defect probability into account during test scheduling.

1. Introduction

The cost of developing electronic systems is increasing and a significant part of the cost is related to the testing of the systems. One efficient way to reduce the total cost is therefore to reduce the testing cost. Test cost reduction can be achieved by minimizing the testing time of the system. An efficient ordering, test scheduling, of the execution of the test sets will minimize the total testing time.

The core-based design technique is another approach to reduce the increasing development costs. With such a technique, pre-designed and pre-verified blocks of logic, so called cores, are integrated to a complete system, which can be placed on a single die to form a system-on-chip (SOC). To test a SOC, a test bus is used for the transportation of test data in the system and its organization often has a great impact on the test schedule. SOC test scheduling can be performed assuming: *sequential scheduling*, *i.e.* only one test at a time, or *concurrent scheduling*, with a possibility to execute several tests at the same time. The testing time for the execution of each test set can be *fixed,* or *flexible* where it is possible to adjust the test time for a test.

In a large volume production test for SOC, an *abort-on-fail* approach is usually used, which means that the test sequence is aborted as soon as a fault is detected. This approach is used to reduce the test application time. With the abort-on-fail assumption, the tests should be ordered in such a way that tests with a high probability to fail are scheduled before tests with a lower probability to fail since this will minimize the average testing time.

In this paper we discuss test scheduling considering defect detection probability, which is collected from the production line or generated based on inductive fault analysis. We have defined models to compute the expected test time as well as scheduling heuristic taking the defect probabilities into account and we have performed experiments to show the efficiency of the proposed approach.

The rest of the paper is organized as follows. An overview of related work is in Section 2. Sequential test scheduling is discussed in Section 3 and concurrent test scheduling is described in Section 4. The paper is concluded with experimental results in Section 5 and conclusions in Section 6.

2. Related Work

Test scheduling determines the execution order of the tests in a system. A common objective is to minimize the test application time while considering test conflicts. In SOC systems, where each core is equipped with a wrapper, an interface to the test access mechanism (TAM), a test conflict is due to the sharing of the TAM or the test bus. The TAM, used for the transportation of test data, is used to connect the test source, the cores and the test sink. The test source is where the test vectors are generated or stored, and the test sink is where the test responses are analyzed or stored. An automatic test equipment (ATE) is a typical example of a test source and test sink.

A TAM can be organized in different ways, which impacts the test scheduling. An example is the AMBA test bus, which makes use of the existing functional bus, however, the tests have to be scheduled in a sequence [2]. An alternative is the approach proposed by Varma and Bhatia where several test buses are used. The tests on each bus are scheduled in a sequence, however, since several buses are allowed, testing can be performed concurrently [11]. Another approach is the TestRail, which allows a high degree of flexibility [8].

The TestRail approach has recently gained interest and several scheduling techniques for scan tested SOCs have been proposed [3,5,7]. The objective is to arrange the scan-

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

chains into wrapper chains, which then are connected to TAM wires. Iyengar *et al.* made use of integer-linear programming [5] and Huang *et al.* used a bin-packing algorithm. Both these approaches assume that the tests will always be performed until completion. Koranne proposed an *abort-on-fail* technique to minimize the average-completion time by scheduling tests with short test time early [7]. For sequential testing, several abort-on-fail test scheduling techniques considering the defect probability have been proposed [4,6]. Huss and Gyurcsik made use of a dynamic programming algorithm to order the tests [4]. Milor and Sangiovanni-Vincentelli proposed a technique for the selection and ordering of the test sets [10], which is based on the dependencies between the test sets. For SOC testing with cores in wrappers, however, there is no dependency between the testing of different cores. In the approach proposed by Jiang and Vinnakota the actual fault coverage is extracted from the manufacturing line [6]. The technique minimizes the average completion time by ordering of the tests based on the failure probability.

3. Sequential Test Scheduling

In sequential testing, all tests are scheduled in a sequence, one test at a time. When the *abort-on-fail* approach is assumed, if a defect is detected, the testing should be terminated at once. In order to account for the case when the test responses are compacted into a single signature after a whole test set is applied, we assume that the test abortion occurs at the end of a test set even if the actual defect is detected in the middle of applying the test set. This assumption is also used in our formula to compute the expected test time. Note, this means that the computational results are pessimistic, or the actual test time will be smaller than the computed one, in the case when the tests are actually aborted as soon as the first defect is detected.

Given a core-based system with n cores, for each core i there is a test set t_i with a test time τ_i and a probability of passing p_i (*i.e.* the probability that test t_i will detect a defect at core i is $1\text{-}p_i$). For a given schedule, the *expected test time* for sequential testing is given by:

$$\sum_{i=1}^{n}\left(\left(\sum_{j=1}^{i}\tau_j\right)\times\left(\prod_{j=1}^{i-1}p_j\right)\times(1-p_i)\right)+\left(\sum_{i=1}^{n}\tau_i\right)\times\prod_{i=1}^{n}p_i \qquad (1)$$

For illustration of the computation of the expected test time, we use an example with four tests (Table 1). The tests are scheduled in a sequence as in Figure 1. For test t_1, the expected test time is given by the test time τ_1 and the probability of success p_1, $\tau_1\times p_1=2\times0.7=1.4$. Note if there is only one test in the system, our above formula will give the expected test time to be 2 since we assume that every test set has to be fully executed before we can determine if the test is a successful test or not.

The expected test time for the completion of the complete test schedule in Figure 1 is:

Core i	Test t_i	Test time τ_i	Probability to pass, p_i
1	t_1	2	0.7
2	t_2	4	0.8
3	t_3	3	0.9
4	t_4	6	0.95

Table 1. Example data.

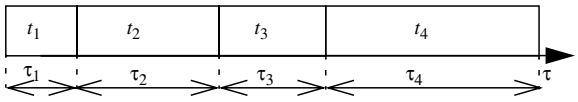

Figure 1. Sequential schedule of the example (Table 1).

$\tau_1\times(1-p_1)$ +
$(\tau_1+\tau_2)\times p_1\times(1-p_2)$ +
$(\tau_1+\tau_2+\tau_3)\times p_1\times p_2\times(1-p_3)$ +
$(\tau_1+\tau_2+\tau_3+\tau_4)\times p_1\times p_2\times p_3\times(1-p_4)$ +
$(\tau_1+\tau_2+\tau_3+\tau_4)\times p_1\times p_2\times p_3\times p_4$ =
$2\times(1-0.7) + (2+4)\times0.7\times(1-0.8) + (2+4+3)\times0.7\times0.8\times(1-0.9) + (2+4+3+6)\times0.7\times0.8\times0.9\times(1-0.95) + (2+4+3+6)\times0.7\times0.8\times0.9\times0.95 = 9.5$

As a comparison, for the worst schedule, where the test with highest passing probability is scheduled first, the order will be t_4, t_3, t_2, t_1, and the expected test time is 13.6. In the case of executing all tests until completion, the total test time does not depend on the order, and is $\tau_1+\tau_2+\tau_3+\tau_4=15$.

The algorithm for test scheduling based on defect probability in the sequential case is straight forward, it sorts the tests in descending order based on $\tau_i\times(1-p_i)$ and schedule the tests in this order (Figure 2).

1. Compute the cost $c_i=(1\text{-}p_i)\times\tau_i$ for all tests t_i
2. Sort the costs c_i descending in L
3. until L is empty (all tests are scheduled) begin
4. select, schedule and remove the first test in L
5. end

Figure 2. Sequential test scheduling algorithm.

4. Concurrent Test Scheduling

The total test time of a system can be reduced by executing several tests at the same time, *concurrent testing*. Concurrent testing is for instance possible in systems with several test buses. In this section, we analyze concurrent scheduling with fixed test time per test set and flexible test time per test set.

A concurrent test schedule of the example system used in Section 3 with data as in Table 1 assuming 3 TAMs (test buses) is in Figure 3. The test schedule (Figure 3) consists of a set of sessions, S_1, S_2, S_3, and S_4. Test session S_1 consists of test t_1, t_2 and t_3; $S_1=\{t_1,t_2,t_3\}$. The length of a session S_k is given by l_k. For instance $l_1=2$. We assume now that the abortion of the test process can occur at any time during the application of the tests. To simplify the

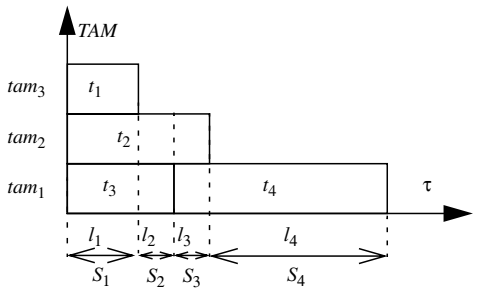

Figure 3. Concurrent schedule of the example (Table 1).

computation of expected test time, it is assumed that the test process will terminate at the end of a session (note this is again a pessimistic assumption). The probability to reach the end of a session depends in the concurrent test scheduling approach not only on a single test, but on all tests in the session. For instance, the probability to complete session 1 depends on the tests in session 1 (S_1): t_1, t_2 and t_3. As can be observed in Figure 3, only test t_1 is fully completed at the end of session 1. For a test t_i that is not completed at the end of a session, the probability p_{ik} for it to pass all test vectors applied during session k is given by:

$$p_{ik} = p_i^{l_k/\tau_i} \qquad 2$$

It can be seen that for a test set t_i, which is divided into m sessions, the probability that the whole test set is passed is equal to:

$$\prod_{k=1}^{m} p_{ik} = p_i^{\frac{l_1}{\tau_i}} \times p_i^{\frac{l_2}{\tau_i}} \times \ldots \times p_i^{\frac{l_k}{\tau_i}} = p_i^{\sum_{k=1}^{m} \frac{l_k}{\tau_i}} = p_i \qquad 3$$

since:

$$\sum_{k=1}^{m} \frac{l_k}{\tau_i} = \frac{1}{\tau_i} \times \sum_{k=1}^{m} l_k = 1 \qquad 4$$

For example, the probability for the tests in session S_1 are (Figure 3):

$p_{11} = p_1 = 0.7$
$p_{21} = 0.8^{2/4} = 0.89$
$p_{31} = 0.9^{2/3} = 0.93.$

The formula for computing the expected test time for a complete concurrent test schedule is given as:

$$\sum_{i=1}^{n} \left(\left(\sum_{j=1}^{i} l_j \right) \times \left(\prod_{j=1}^{i-1} \prod_{\forall t_k \in S_j} p_{kj} \right) \times \left(1 - \prod_{\forall t_k \in S_i} p_{ki} \right) \right) + \left(\sum_{i=1}^{n} \tau_i \right) \times \prod_{i=1}^{n} p_i \qquad 5$$

As an example, the computation of the expected test time for the test schedule in Figure 3 is given below. First we compute the probability for each test set in each session.

The probabilities p_{11}, p_{21}, p_{31} are computed to 0.7, 0.89, and 0.93, respectively (see above).

$p_{22} = 0.7^{1/4} = 0.91.$
$p_{32} = 0.9^{1/3} = 0.96.$
$p_{23} = 0.8^{1/4} = 0.95.$
$p_{43} = 0.95^{1/6} = 0.99.$
$p_{44} = 0.95^{5/6} = 0.96.$

From the formula we get:

$l_1 \times (1 - p_{11} \times p_{21} \times p_{31}) +$
$(l_1 + l_2) \times p_{11} \times p_{21} \times p_{31} \times (1 - p_{22} \times p_{32}) +$
$(l_1 + l_2 + l_3) \times p_{11} \times p_{21} \times p_{31} \times p_{22} \times p_{32} \times (1 - p_{23} \times p_{43}) +$
$(l_1 + l_2 + l_3 + l_4) \times p_{11} \times p_{21} \times p_{31} \times p_{22} \times p_{32} \times p_{23} \times p_{43} \times (1 - p_{44}) +$
$(l_1 + l_2 + l_3 + l_4) \times p_{11} \times p_{21} \times p_{31} \times p_{22} \times p_{32} \times p_{23} \times p_{43} \times p_{44} =$
$2 \times (1 - 0.7 \times 0.89 \times 0.93) +$
$(2+1) \times 0.7 \times 0.89 \times 0.93 \times (1 - 0.91 \times 0.96) +$
$(2+1+1) \times 0.7 \times 0.89 \times 0.93 \times 0.91 \times 0.96 \times (1 - 0.95 \times 0.99) +$
$(2+1+1+5) \times 0.7 \times 0.89 \times 0.93 \times 0.91 \times 0.96 \times 0.95 \times 0.99 \times$
$(1 - 0.96) + (2+1+1+5) \times 0.7 \times 0.8 \times 0.9 \times 0.95 = 5.66.$

As a comparison, if all tests are assumed to be executed until completion, the total test time will be 9.

In concurrent scheduling with fixed test times, we sort the tests based on $\tau_i \times (1 - p_i)$ and select n tests for the n TAMs based on the sorted list. The selected tests are scheduled and removed from the list. As soon as a test terminates, a new test from the list of unscheduled tests is selected. The process continues until all tests are scheduled (sketch of the algorithm is given in Figure 4.)

1. Compute the cost $c_i = \tau_i \times (1 - p_i)$ for all tests t_i
2. Sort the costs c_i descending in L
3. f=number of TAMs
4. τ=0 // current time,
5. **Until** L is empty (all tests are scheduled) begin
6. at time τ **Until** f=0 **Begin**
7. select tests from list in order and reduce f accordingly
8. **End**
9. τ=time when first test terminates.
10. **End**

Figure 4. Concurrent test scheduling algorithm for tests with fixed test times.

A way to further reduce the test application time is to modify, if possible, the test times of the individual test sets. For instance, in scan tested cores the test times at each core can be modified by loading several scan chains in parallel. The scan-chains and the wrapper cells are to form a set of wrapper chains. Each wrapper chain is then connected to a TAM wire. If a high number of wrapper chains are used, their length is shorter and the loading time of a new test vector is reduced. However, the higher number of wrapper chains requires more TAM wires.

In Figure 5 the TAM bandwidth |W| is 4, there are four wires in W={w_1, w_2, w_3, w_4}. The testing of each core is performed by transporting test vectors on the assigned TAM wires to a core and the test response is also transported from the core to the test sink using the TAM. The testing of cores sharing TAM wires cannot be executed concurrently. For instance, the testing of core 1 and core 2 cannot be

performed concurrently due to the sharing of TAM wire w_3 (Figure 5). A test schedule for the system is given in Figure 6 and the computation of the expected test time can be done using formula 3 in Section 4. In the case with flexible test times, the number of assigned TAM wires will affect the expected test time.

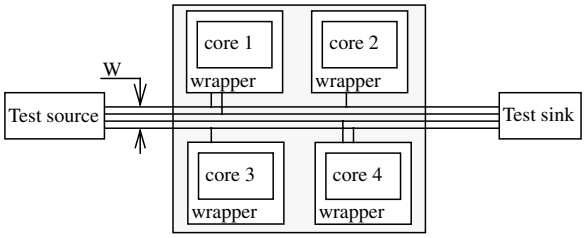

Figure 5. SoC example.

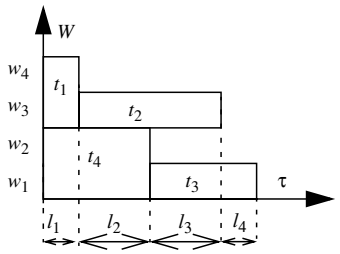

Figure 6. SOC test schedule of the example (Table 1).

5. Experimental Results

We have compared three approaches to demonstrate the importance of considering the defect probability during test scheduling. We have implemented (1) sequential scheduling where the tests are ordered in a sequence, (2) one where a fixed test time is assigned to each test prior to scheduling (for details see [13]), and (2) one where the testing time versus the number of used TAM wires is flexible (for details see [12]). For all three approaches, we have included the consideration of defect probabilities.

We have used the ITC'02 designs [9] and for all experiments we have used an AMD 1800 machine (1.53 GHz, 512 MB RAM) and the computational cost is usually a few seconds, and never exceeds 15 seconds. The defect probability for each core is collected in Table 2.

The experimental results are collected in Table 3. We have for each of the benchmarks made experiments at various TAM bandwidths. We have compared the three approaches. The results indicate that an efficient ordering taking the defect probabilities into account can reduce the average testing times up to nearly 90% compared to sequential testing (also taking the defect probabilities into account).

6. Conclusions

In this paper we have discussed test scheduling techniques for system-on-chip (SOC) that take into account the defect probability of each test. The advantage with our approach is that by considering defect probabilities during the test scheduling process, the expected test time can be minimized, which is important in large volume production of SOC where the testing process is terminated as soon as a defect is detected (abort-on-fail).

We have analyzed several different test bus structures and scheduling approaches, and defined models to compute the expected test times and test scheduling algorithms for several types of test buses. We have also performed experiments to demonstrate the efficiency of our approach.

Core	Design						
	d695	h953	g1023	t512505	p22810	p34392	p93791
1	98	95	99	99	98	98	99
2	99	91	99	95	98	98	99
3	95	92	99	97	97	97	97
4	92	92	98	93	93	91	90
5	99	97	94	90	91	95	91
6	94	90	95	98	92	94	92
7	90	94	94	98	99	94	98
8	92	96	97	96	96	93	96
9	98		92	92	96	99	91
10	94		92	91	95	99	94
11			96	91	93	91	93
12			92	92	91	91	91
13			93	91	92	90	91
14			96	91	93	95	90
15				99	99	94	99
16				95	99	96	98
17				97	99	96	97
18				95	95	97	99
19				94	96	92	99
20				99	97	90	99
21				91	93	92	90
22				99	99	99	99
23				91	96	96	90
24				97	98	98	98
25				92	99		92
26				96	92		96
27				95	91		95
28				92	91		91
29				90	93		90
30				91	94		96
31				95			

Table 2. The pass probability in percentage for cores in the systems.

Design	TAM Width	Expected Test Time			Comparison	
		1 - Sequential Testing	2 - Fixed Testing Times [13]	3 - Flexible Testing Times [12]	(3) vs (1)	(3) vs (2)
g1023	128	41 807	24 878	17 904	- 57,2%	- 28,0%
	96	43 289	23 443	18 741	- 56,7%	- 20,1%
	80	44 395	23 112	18 229	- 58,9%	- 21,1%
	64	44 395	27 358	20 773	- 53,2%	- 24,1%
	48	46 303	27 997	21 501	- 53,6%	- 23,2%
	32	55 562	27 662	26 867	- 51,6%	- 2,9%
	24	56 711	29 410	28 795	- 49,2%	- 2,1%
	20	60 609	36 979	35 431	- 41,5%	- 4,2%
	16	75 100	44 728	44 657	- 40,5%	- 0,2%
	12	95 679	67 549	60 239	- 37,0%	- 10,8%
d695	128	31 113	10 884	9 468	- 69,6%	- 13,0%
	96	31 158	14 716	11 712	- 62,4%	- 20,4%
	80	31 158	14 881	14 509	- 53,4%	- 2,5%
	64	40 586	25 483	16 652	- 59,0%	- 34,7%
	48	40 692	27 388	23 983	- 41,1%	- 12,4%
	32	70 411	50 998	33 205	- 52,8%	- 32,9%
	24	70 598	62 367	42 165	- 40,3%	- 32,4%
	20	70 696	68 611	50 629	- 28,4%	- 26,2%
	16	131 178	123 164	61 473	- 53,1%	- 50,1%
	12	131 465	131 465	82 266	- 37,4%	- 37,4%
h953	128	104 382	87 339	82 358	- 21,1%	- 5,7%
	96	104 466	82 733	82 437	- 21,1%	- 0,4%
	80	104 466	85 307	82 448	- 21,1%	- 3,4%
	64	104 466	87 349	82 466	- 21,1%	- 5,6%
	48	104 508	87 443	82 495	- 21,1%	- 5,7%
	32	104 549	92 245	84 169	- 19,5%	- 8,8%
	24	104 591	99 888	104 290	- 0,3%	+ 4,4%
	20	104 633	125 262	92 159	- 11,9%	- 26,4%
	16	159 657	137 089	135 438	- 15,2%	- 1,2%
	12	189 740	185 016	183 359	- 3,4%	- 0,9%
p22810	128	423 852	71 628	50 484	- 88,1%	- 29,5%
	96	423 968	93 921	59 177	- 86,0%	- 37,0%
	80	423 993	122 641	71 995	- 83,0%	- 41,3%
	64	443 459	141 999	92 218	- 79,2%	- 35,1%
	48	510 795	213 995	121 865	- 76,1%	- 43,1%
	32	535 586	355 646	160 237	- 70,1%	- 54,9%
	24	707 813	480 480	294 612	- 58,4%	- 38,7%
	20	836 491	756 138	328 270	- 60,8%	- 56,6%
	16	877 443	855 355	383 034	- 56,3%	- 55,2%
	12	1 341 549	1 336 251	401 720	- 70,1%	- 69,9%
t512505	128	9 724 227	1 073 413	889 677	- 90,9%	- 17,1%
	96	9 724 227	1 217 641	894 924	- 90,8%	- 26,5%
	80	9 724 227	1 269 333	928 499	- 90,5%	- 26,9%
	64	9 724 227	2 810 847	1 062 112	- 89,1%	- 62,2%
	48	14 883 557	8 938 649	1 828 281	- 87,7%	- 79,5%
	32	14 883 609	8 940 193	1 955 361	- 86,9%	- 78,1%
	24	25 202 194	16 090 266	2 891 241	- 88,5%	- 82,0%
	20	25 202 230	16 308 884	3 652 388	- 85,5%	- 77,6%
	16	25 202 298	21 716 978	3 961 341	- 84,3%	- 81,8%
	12	46 296 336	27 526 848	5 394 939	- 88,3%	- 80,4%

Table 3. Experimental results comparing three approaches (1), (2) and (3).

Design	TAM Width	Expected Test Time			Comparison	
		1 - Sequential Testing	2 - Fixed Testing Times [13]	3 - Flexible Testing Times [12]	(3) vs (1)	(3) vs (2)
p34392	128	1 168 630	258 038	265 777	- 77,3%	+ 3,0%
	96	1 168 630	343 408	248 170	- 78,8%	- 27,7%
	80	1 212 761	374 916	262 563	- 78,3%	- 30,0%
	64	1 212 899	470 976	268 010	- 77,9%	- 43,1%
	48	1 232 116	627 558	389 813	- 68,4%	- 37,9%
	32	1 525 655	1 271 626	563 470	- 63,1%	- 55,7%
	24	1 559 706	1 389 689	823 435	- 47,2%	- 40,7%
	20	1 640 812	1 596 012	1 382 206	- 15,8%	- 13,4%
	16	2 888 061	2 677 967	1 604 297	- 44,5%	- 40,1%
	12	2 960 587	2 926 044	2 154 610	- 27,2%	- 26,4%
p93791	128	491 279	431 628	124 278	- 74,4%	- 71,2%
	96	524 537	488 083	192 940	- 63,2%	- 60,5%
	80	942 900	852 477	197 393	- 79,1%	- 76,8%
	64	983 943	922 505	270 979	- 72,5%	- 70,6%
	48	1 072 900	1 003 672	360 045	- 66,4%	- 64,1%
	32	1 941 982	1 941 892	682 101	- 64,9%	- 64,9%
	24	2 125 118	2 125 118	826 441	- 61,1%	- 61,1%
	20	3 546 031	3 546 031	1 023 667	- 71,1%	- 71,1%
	16	3 854 386	3 854 386	1 353 034	- 64,9%	- 64,9%
	12	4 238 379	4 238 379	3 768 819	- 11,1%	- 11,1%

Table 3. Experimental results comparing three approaches (1), (2) and (3).

References

[1] M. L. Flottes, J. Pouget, and B.Rouzeyre, "Sessionless Test Scheme: Power-constrained Test Scheduling for System-on-a-Chip", *Proceedings of the 11th IFIP on VLSI-SoC*, pp. 105-110.Montpellier, June 2001.

[2] P. Harrod, "Testing reusable IP-a case study", *Proceedings of International Test Conference (ITC)*, Atlantic City, NJ, USA, pp. 493-498, 1999.

[3] Y. Huang, W.-T. Cheng, C.-C. Tsai, N. Mukherjee, O. Samman, Y. Zaidan and S. M. Reddy, "Resource Allocation and Test Scheduling for Concurrent Test of Core-based SOC Design", *Proceedings of IEEE Asian Test Symposium (ATS)*, pp 265-270, Kyoto, Japan, November 2001.

[4] S. D. Huss and R. S. Gyurcsik, "Optimal Ordering of Analog Integrated Circuit Tests to Minimize Test Time", *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pp. 494-499, 1991.

[5] V. Iyengar, and K. Chakrabarty, and E. J. Marinissen, "Test wrapper and test access mechanism co-optimization for system-on-chip", *Proceedings of International Test Conference (ITC)*, Baltimore, MD, USA, pp. 1023-1032, 2001.

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

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

[8] E. J. Marinissen, R. Arendsen, G. Bos, H. Dingemanse, M. Lousberg, C. Wouters, "A structured and scalable mechanism for test access to embedded reusable cores", *Proceedings of International Test Conference (ITC)*, Washington, DC, USA, pp. 284-293, October 1998.

[9] E. J. Marinissen, V. Iyengar, and K. Chakrabarty, "A Set of Benchmarks for Modular Testing of SOCs", *Proceedings of International Test Conference (ITC)*, pp. 519-528, Baltimore, MD, USA, October 2002.

[10] L. Milor and A. L. Sangiovanni-Vincentelli, "Minimizing Production Test Time to Detect Faults in Analog Circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 13, No. 6, pp 796-, June 1994.

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

[12] J. Pouget, E. Larsson, and Z. Peng, "SOC Test Time Minimization Under Multiple Constraints", *Proceedings of Asian Test Symposium (ATS),* Xian, China, 2003.

[13] J. Pouget, E. Larsson, Z. Peng, M. L. Flottes, and B. Rouzeyre,An Efficient Approach to SoC Wrapper Design, TAM Configuration and Test Scheduling", *Formal Proceedings of European Test Workshop 2003* (*ETW*'03), Maastricht, The Netherlands, May 25-28, 2003, pp 51-56.