JOURNAL OF ELECTRONIC TESTING: Theory and Applications 21, 599–611, 2005

Multiple-Constraint Driven System-on-Chip Test Time Optimization*

JULIEN POUGET, ERIK LARSSON AND ZEBO PENG
Embedded Systems Laboratory, Department of Computer and Information Science, Linköpings Universitet, Sweden
erik.larsson@ida.liu.se

Received March 1, 2004; Revised June 7, 2005

Editor: H.J. Wunderlich

Abstract. The cost of testing SOCs (systems-on-chip) is highly related to the test application time. The problem is that the test application time increases as the technology makes it possible to design highly complex chips. These complex chips include a high number of fault sites, which need a high test data volume for testing, and the high test data volume leads to long test application times. For modular core-based SOCs where each module has its distinct tests, concurrent application of the tests can reduce the test application time dramatically, as compared to sequential application. However, when concurrent testing is used, resource conflicts and constraints must be considered. In this paper, we propose a test scheduling technique with the objective to minimize the test application time while considering multiple conflicts. The conflicts we are considering are due to cross-core testing (testing of interconnections between cores), module testing with multiple test sets, hierarchical conflicts in SOCs where cores are embedded in cores, the sharing of the TAM (test access mechanism), test power limitations, and precedence conflicts where the order in which tests are applied is important. These conflicts must be considered in order to design a test schedule that can be used in practice. In particular, the limitation on the test power consumption is important to consider since exceeding the system's power limit might damage the system. We have implemented a technique to integrate the wrapper design algorithm with the test scheduling algorithm, while taking into account all the above constraints. Extensive experiments on the ITC'02 benchmarks show that even though we consider a high number of constraints, our technique produces results that are in the range of results produced be techniques where the constraints are not taken into account.

Keywords: SOC testing, multiple constraints, wrapper and TAM design, test scheduling, power constraint

1. Introduction

The IC technology development has made it possible to produce extremely complex chips. The cost of testing these chips is increasing, and it is important to develop techniques to reduce the cost of testing. The cost of test is highly related to the test application time, and the testing times for chips are increasing due to the growing complexity of chips. In order to handle the design of complex system within a reasonable design time, the use of core-based SOC design methodology, where pre-defined logic blocks, cores, are integrated with UDL (user-defined logic) to form a system, has been developed. These system chips require excessive test data volumes for their testing, hence, long test application times.

The long test application time can be reduced by allowing tests to be executed concurrently. However, when allowing concurrent application of the tests, conflicts and limitations must be carefully considered.

*The research is partially supported by the Swedish National Program on Socware. An earlier version of the paper has been presented at the Asian Test Symposium (ATS), 2003 [16].

Test conflicts due to cross-core testing (interconnection testing), unit testing with multiple test sets, hierarchical SOCs where cores are embedded in cores, and the sharing of test access mechanism (TAM) wires, must be considered during the test scheduling process in order to develop a test schedule that can be applied in practice. Further, executing tests concurrently increases the activity in the system, which leads to higher power consumption. And it is important that the test power constraints are not violated since it might otherwise damage the system under test.

Several approaches have been proposed for SOC test scheduling [2–8, 10–13, 15]. The basic problem is to minimize the test application time for a design where the test sets are stored in an Automatic Test Equipment (ATE) and the main limitation is the number of available pins in the system. Goel and Marinissen [11], for instance, proposed for systems where each core has a dedicated wrapper, a technique to schedule the test data transportation on the TAM wires in such a way that the total test application time is minimized. Huang et al. proposed a method to address the test power consumption [4], where the test time for a system with wrapped cores is minimized while test power limitations are considered and tests are assigned to TAM wires. Recently, Iyengar et al. proposed a scheduling technique to minimize testing time while taking hierarchical constraints into account [9]. We have in our previous work considered design hierarchy constraints, power limitations, precedence constraints, multiple test sets and interconnection test [15]. However, the wrapper design and the test scheduling tasks were considered as two sequential steps, which have the consequence that even if locally optimal wrapper configurations are selected, a global system optimum is usually not achieved [15].

In this paper we address the SOC test scheduling problem by proposing a test scheduling technique that minimizes the test application time while considering test power consumption and test conflicts. In our approach we consider:

- TAM wire assignment: each test must be assigned a start time and an end time as well as TAM wires in the case of tests stored in the ATE;
- power constraints: in test mode, cores can dissipate more power than in functional mode and the system power limit has to be respected in order not to damage the chip;
- hierarchical constraints: in some designs, cores (children) may be embedded in other cores (parents), and they can not be tested simultaneously;
- cores with multiple test sets: a core can, for instance, be tested using one test set generated by an LFSR and another test set stored in the ATE;
- cross-core (interconnection) testing: the logic and interconnections placed between wrapped cores should also be tested; and
- precedence constraints: a particular order has sometimes to be enforced between some of the tests.

The main advantage of our proposed approach, compared to previous work, is that we integrate the wrapper design algorithm with the test scheduling algorithm, which makes it possible to explore the design space in a more efficient way since the wrapper configuration is not fixed prior to test scheduling.

The rest of the paper is organized as follows. In Section 2, we give the background to the work and formulate precisely the problem. Our combined wrapper design and test scheduling technique is then described in Section 3. The experimental results are reported in Section 4, and the conclusions in Section 5.

2. Background and Problem Formulation

In this section we give the background and our problem formulation. Let us consider a core-based system as given in Fig. 1. Such a system is said to be testable if every testable unit in the system is equipped with a test method and corresponding test sets. A testable unit can be a core, UDL, or interconnections. It is also assumed that a set of pins can be used for the TAM (the total number of wires in the TAM is denoted by $W_{\max}$ as in Fig. 1) and in order to connect the cores to the TAM some cores are equipped with wrappers.

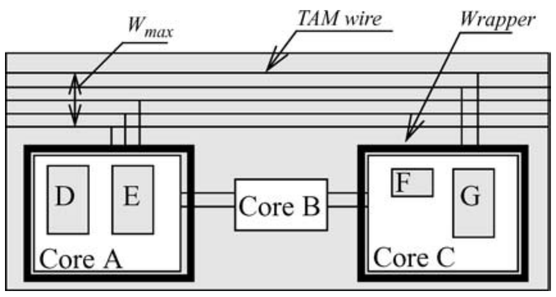


Fig. 1. A core-based design with a TAM, cores in wrappers (core A and core C) and hierarchy (core D and E are embedded in core A, and core F and G in core C). Core B is the interconnection between cores A and C.

The problem we focus on is how to assign a start time, an end time and if needed the set of TAM wires for each test in such a way that the total test time is minimized. The assignment should consider the constraints discussed below.

A wrapper serves as the interface between a core and the TAM and it can normally be in one of the following modes at a time: *normal operation mode, internal test mode, external test mode*, or *bypass mode*. Some cores are equipped with wrappers while others are not. In order to access test data on the TAM, a wrapper must be used. If a testable unit does not have its own wrapper, some other wrapper must be used in order to get access to the TAM. For example, core B has no wrapper and in order to test core B in Fig. 1 with a test stored in the ATE, the wrapper at core A can be used to feed test stimuli to core B and core C can be used to receive test responses from core B. Note that since a wrapper can only be in one mode at a time, testing of core B cannot be performed concurrently with the testing of core A and core C. In this particular example, core B is actually used to model the interconnection between cores A and C. The testing of core B is therefore an example of cross-core (interconnection) testing. In our approach, an interconnection that is to be tested will always be modelled by a special core, as core B here. In this way, interconnection test is treated as ordinary core test with some special constraints as in this example.

Another conflict illustrated in Fig. 1 is the design hierarchy conflict. The two cores named F and G are embedded within core C. Such embedding of cores may lead to test conflicts since concurrent testing of core F and/or core G with core C may not be possible. Furthermore, each testable unit can be tested by one or several test sets. If more than one test set exists for a testable unit, there is a test set conflict since only one test set can be applied at a time to a testable unit.

We assume that a test set for a testable unit is either stored in an ATE or generated at a dedicated BIST engine placed at the testable unit. This means that if a testable unit is tested by only a BIST test set there is no need to make use of TAM wires. On the other hand, for a test stored at the ATE, TAM wires are required for the transportation of test stimuli from the ATE via the TAM to the testable unit, and for the transportation of test response from the testable unit to the ATE. At any time, only one testable unit can use a TAM wire, which is captured as a sharing conflict, and illustrated in Fig. 2. In Fig. 2 the assignment of TAM wires to three test sets over time is given. Each test set is assigned to several TAM wires for a certain period of time.

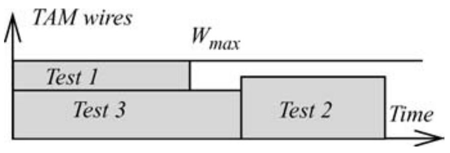


Fig. 2. TAM wire-constrained test scheduling.

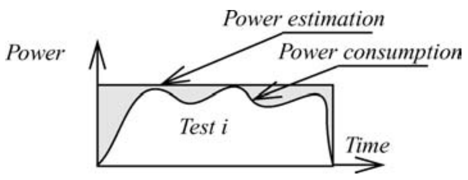


Fig. 3. Modelling of test power consumption.

The execution of a test results in switching activities, which consume power. Fig. 3 shows the execution of a test and its power consumption. The power consumption varies over time; however, to simplify the analysis, we will use a power model introduced by Chou et al. [1] that assumes a fixed power value attached to each test. And the total power consumed by a system under test at a certain point is the summation of the power of the tests' that are executed at the point. At no time it is allowed to consume more power than the maximal power constraint.

In some cases, the order in which the tests are executed is important. Such an order imposes precedence constraints, which means that some tests must be executed prior to others.

The test time of a testable unit can often be modified. One such example is illustrated with the scan-tested core given in Fig. 4, where the scan-chains and the wrapper cells can be configured into two wrapper-chains. An increasing number of wrapper-chains reduce the test time at the expense of more TAM wires

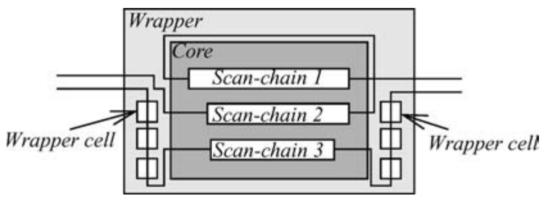


Fig. 4. A wrapped scan tested core where the scan-chains and wrapper cells are configured into two wrapper chains.

and vice versa. Iyengar et al. showed that the problem of TAM wire and wrapper optimization is NP-hard [7] and proposed a technique to address the problem. Goel and Marinissen [10] also proposed an approach to solve this problem.

3. Proposed Test Scheduling Technique

In this section we describe our wrapper design technique (scan-chain configuration), our test scheduling algorithm and how they are integrated. The wrapper design algorithm configures the scan elements (scan-chains, input wrapper cells, output wrapper cells and bidirectional wrapper cells) into a given number of wrapper chains and computes the test time for the wrapper configuration. And the scheduling algorithm selects the most appropriate wrapper design for each core in the system and assign TAM wires and a start time in such a way that the test application time is minimized while all constraints are satisfied.

3.1. Wrapper Design Algorithm

The wrapper design algorithm assigns the scanned elements at a core into a given number of wrapper-chains and computes the test time. The proposed wrapper design heuristic is illustrated in Fig. 6 and the aim with the wrapper chaining function is to balance the wrapper-chains in order to reduce the longest wrapper chain. The longest wrapper chain is the one that determines the testing time as shown by Pouget et al. [15]. The generated wrapper designs are memorized so that all possible configurations for each core are available during the TAM design and the test scheduling steps.

A small example illustrating the algorithm is given in Fig. 7. In the example, we have a core with 10 I/Os, 5 scan chains of lengths 10, 9, 5, 3, and 2, respectively, see Fig. 7(a) and we will in the example create three wrapper chains. The three longest scan chains are assigned to a wrapper chain each, see Fig. 7(b). Then, the two shortest chains are combined into one chain. First, scan-chain of length 2 and 3 are chained (Fig. 7(c)). At this point, we have 4 chains of length 10, 9, 5, $3 + 2$, and we let the shortest chains be chained again, i.e. 5 and $3 + 2$ (Fig. 7(d)). At this point, we have chained all scan-chains into 3 wrapper-chains of length 10, 9, $5 + 3 + 2$, and we only have the input/output cells to distribute, which is trivial as they are each of length one and can be assigned to the chains in a balanced way.

The test time at a core usually, but not always, decreases when a higher number of wrapper-chains are allowed. One example of the obtained results is shown in Fig. 5 for a very simple core (core 5) in design d695. If the wrapper design algorithm results in the same test time for a number of wrapper-chain configurations, we have a Pareto optimal point, which among these

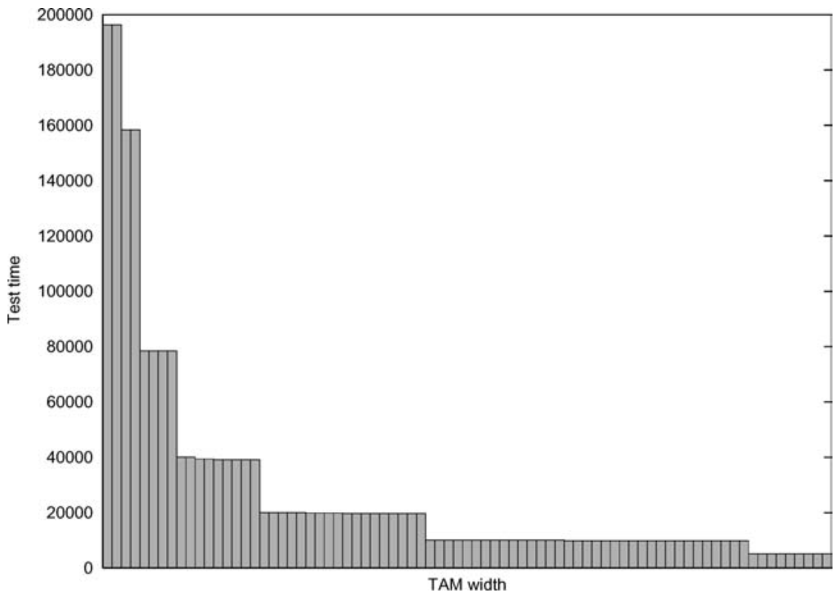


Fig. 5. Test time for a set of wrapper configurations for core 5 in design d695.

```
Wmax=number of TAM connections
NbLines=(int)(Wmax/2)
#SC=number of Scan Chains

Process 'Internal chaining'
Sort the internal scan chain in decreasing length order
Select the (NbLines) longest scan chains as the (NbLines) lines
While (#SC>NbLines)
    Chain the shortest line with the shortest scan chain
    Update #SC (#SC=#SC-1)
    Update length of the longest scan chain
    Sort scan chains in decreasing length order
End process
Add functional I/Os (inputs/outputs) to balance the scan chains
End
```

Fig. 6. Wrapper design heuristic.

wrapper designs is the configuration where the lowest number of wrapper-chains is used. If there is a configuration where the test time is unique, it is also a Pareto optimal point. Obviously, only Pareto-optimal points are of interest since they make use of the lowest possible number of TAM wires to reach a certain test time. We compute all Pareto optimal points for each core by the wrapper design heuristic, and in the scheduling step, we use a heuristic aiming at minimizing the total test time by selecting wrapper design among the Pareto optimal points.

Among the Pareto-optimal points for a core, we observe that the "area" given by its *test time × number of warpper chains* is not constant. We therefore define the *bestPareto* for a core as:

$$bestPareto = \min_{\forall i}\{T_i \times W_i\} \tag{1}$$

where i is a wrapper configuration, and the loss for a given configuration as:

$$loss = \frac{bestPareto - T_i \times W_i}{bestPareto} \tag{2}$$

Optimal wrapper design for a selected core leads to a local optimum at each core; however, from a global system perspective a local optimized solution will rarely lead to the global optimum. Hence, the wrapper design selection must consider all cores in the system.

3.2. Scheduling Algorithm

The scheduling algorithm selects wrapper design for each core, assigns a start time, an end time and which TAM wires to use for each core in such a way that the test application time is minimized while all constraints are satisfied.

The *OptimalTime* is a lower bound that represents the "ideal" situation, but, due to the TAM structure and the wrapper design, this limit is almost never reached. The *OptimalTime* is calculated using the formula:

$$OptimalTime = \left\lceil \frac{\sum_i bestPareto}{W_{\max}} \right\rceil$$

where $W_{\max}$ is the number of available pins for test access (the TAM bandwidth).

The *OptimalTime* gives the lower bound of the total test time of the system when no constraints but TAM width limitations are considered. In the ideal case, the schedule does not contain any idle times (i.e. there is no cost loss between tests in the test schedule), and it is therefore the best test application time that can ever be achieved. It assumes that the optimal wrapper design can be selected for each core and that the selected optimal wrapper designs can be assigned to TAM wires in such a way that no idle to is found in the schedule. In practice, it is usually not possible to find a test schedule with the *OptimalTime* test time because there exist test incompatibilities due to design hierarchy and test resource sharing constraints. However, the OptimalTime gives a feeling for how good a test schedule is.

The scheduling heuristic is outlined in Fig. 8. The algorithm makes use of two lists (L1 and L2). The first list (L1) contains the tests that are to be scheduled, and L2 is a temporary list where tests that the algorithm has tried but could not schedule yet. All tests are first sorted and placed into list L1. If a test cannot be scheduled for a given reason due to one of the constraints, it is placed in an auxiliary list L2 to be scheduled later. When L1 is empty, i.e. all tests are scheduled or placed in L2, the tests in L2 are moved to L1 and the process is re-iterated until all tests are scheduled. We first sort the tests in decreasing test time order (line 1 in Fig. 8). For each test, one Pareto optimal point of wrapper design is selected, considering the maximal TAM width usage, i.e. the pair $< T_i, W_i >$ where T_i is the test time and W_i is the TAM usage with W_i being the closest to $W_{\max}$, the given maximal number of TAM wires. In step 3, we use the cost loss value to select a subset of the Pareto optimal points to be considered for scheduling. The Pareto optimal points correspond to optimal wrapper designs for a given width constraint and are pre-calculated by our wrapper design heuristic.

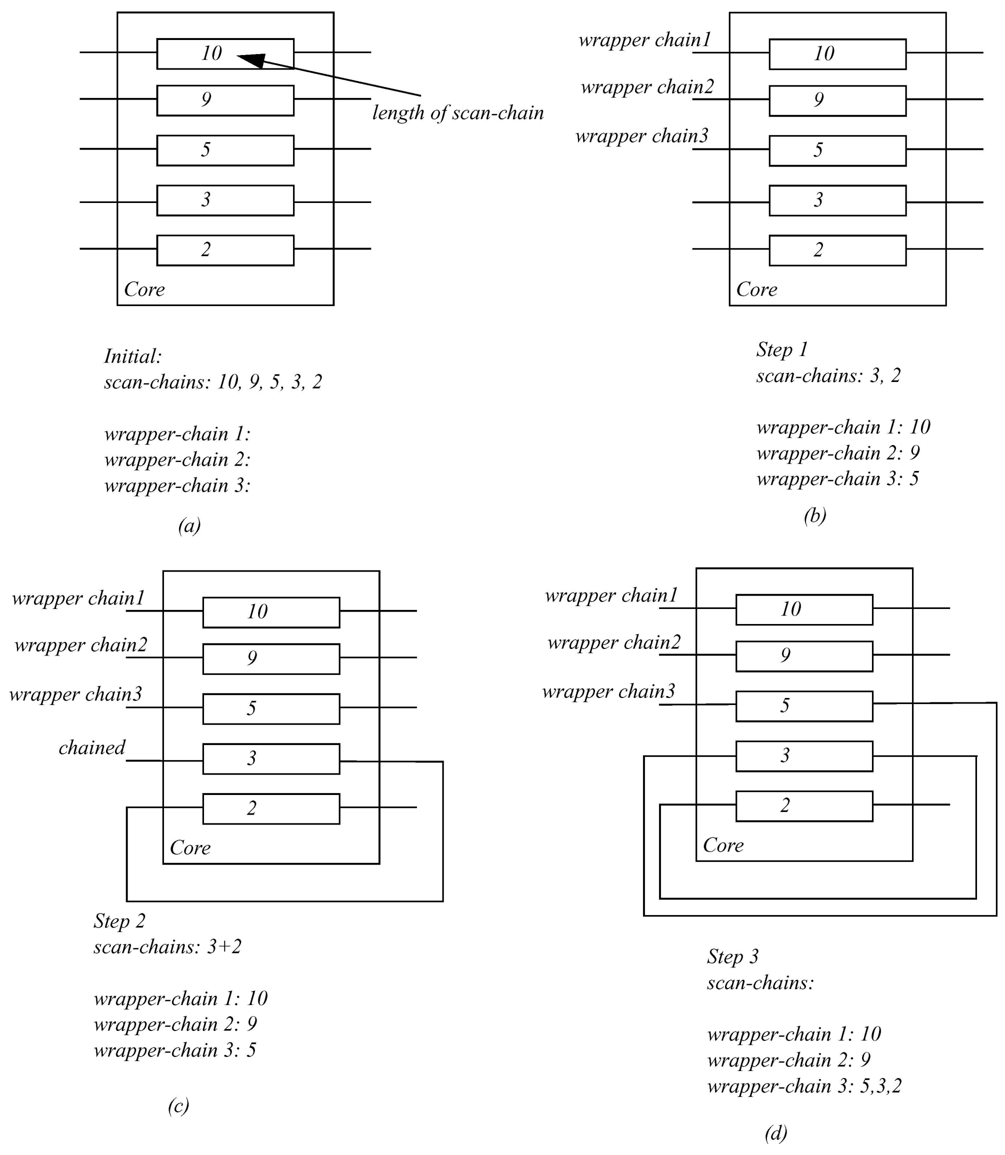


Fig. 7. A wrapper design process to configure 5 scan-chains at a core into 3 wrapper-chains.

As there is a cost loss (discussed above) for some Pareto-optimal points, favour is given to Pareto optimal points with low cost loss. For instance, if the cost loss is 10%, the choice of the Pareto optimal point for a core is a subset where all wrapper designs have a cost loss between 0 and 10%. The heuristic creates one schedule and one TAM configuration for each cost loss (e.g. 81 schedules and TAM configurations will be selected from 0% to 80%) and finally the best schedule is returned as the final solution. The idea is that wrapper design for each core should be selected with as small local loss as possible; however, the selected wrapper designs have to fit the schedule in an effective way.

From step 4 to step 8, the heuristic schedules the tests as soon as possible using the Pareto optimal points defined in the wrapper design heuristic depicted in Fig. 6. At step 8, for each test, the heuristic tries to place each test in a session. A session is given as when any scheduled test ends. Note that the sessions are not fixed and will be modified as the test scheduling algorithm proceeds. The algorithm is starting from time $t = 0$, and by trying all the Pareto optimal points (i.e. changing

```
1. L1=list of tests sorted in decreasing order of test time
2. Compute OptimalTime (Eq. 3. )
3. For cost loss=0 to cost loss=80 do
4.    While all tests are not scheduled
5.       While L1 not empty
6.          For each test T in L1
7.             For each time point t defining the beginning of a test session
8.                Select the best Pareto optimal point such that
                     a) it respects the tolerance;
                     b) the width constraint is satisfied,
                     c) the test time does not exceed OptimalTime, and
                     d) precedence, power, cross-core conflicts, hierarchical conflicts,
                        incompatibilities constraints are respected.
9.                If (the current total test time will not change when T is scheduled to start at t)
10.                  Schedule T at t with the selected Pareto point; remove T from L1.
11.               Else
12.                  If T is the first test of L1
13.                     Schedule T at t with any selected Pareto point; remove T from L1.
14.                  Else
15.                     Place test T in L2; remove T from L1.
16.   L1<=L2
17. End
```

Fig. 8. The test scheduling heuristic.

the values of W_i and T_i) of the considered tests that do not violate the constraints and have a cost loss lower or equal to the allowed cost loss. A test that is checked if it can be schedule is checked at the beginning of all the available sessions. Once the best Pareto optimal point is chosen the test is scheduled (steps 9 and 10) and removed from L1.

Steps 12 and 13 of the test scheduling algorithm are for the first test of list L1, and accessed at each iteration when L2 goes into L1. It means that if no test has been scheduled when L1 is traversed, the first test is forced to be scheduled in order to make the scheduling proceed. And finally, when L1 is empty (step 16) list L2 goes to L1, and the process is re-iterated.

3.3. *Illustrative Example*

We make use of an example with data in Table 1 to illustrate the algorithm. The example assumes that all cores are wrapped and that the only conflict to be considered is the TAM wire assignment at TAM width (W_{tam}) limitation set to 32. The Pareto-optimal points are first computed using the algorithm in Fig. 6 and the results; the test time at a given TAM width, the cost (test time $\times$ TAM width) and the cost loss (the cost difference to the Pareto-optimal point with the lowest cost) for each core are presented in Table 2. The algorithm explored cost loss in the range from 0% to 80%; however, here we show only one at cost loss 16% (Fig. 9). The final reported schedule is the schedule for all created schedules with the lowest test application time.

The cores are sorted based on cost (corel, core2, core3, and core4) and placed in L1 and the OptimalTime is calculated to $(1780 + 736 + 586 + 250)/32 = 3352/32 = 104.75 = 105$ using Eq. (3). The OptimalTime and the TAM constraint are shown in the empty schedule in Fig. 9(a). The first core in L1 (core1) is selected and scheduled in such a way that it maximizes the TAM usage and minimizes the test time. For the first core the cost loss limit is not considered, test time minimization is regarded as more important Fig. 9(b). The first session is created when corel is scheduled. The list (L1) is iterated in order to find tests that can be scheduled without increasing the test application time

Table 1. Data for the illustrative example.

	Scan-chains	Input cells	Output cells	Test vectors
Core 1	10 10 10 10 10 10 10 10	0	0	10
Core 2	8 8 8 8 4 4	0	0	8
Core 3	8 8 6 6 6 6 4 4	0	0	4
Core 4	6 6 3 2	0	0	6

Table 2. The Pareto-optimal points for each of the four cores in the illustrative example.

TAM width (W)	Test time (T)	Cost (T × W)	Cost loss (%)
	Core 1		
2	890	1780	0
4	450	1800	1.1
6	340	2040	14.6
8	230	1840	3.4
16	120	1920	7.9
	Core 2		
2	368	736	0
4	188	752	2.2
6	152	912	23.9
8	116	928	26.1
10	80	800	8.7
	Core 3		
2	293	586	0
4	149	596	1.7
6	113	678	15.7
8	89	712	21.5
10	77	770	26.6
12	65	780	31.4
14	53	742	33.1
	Core 4		
2	125	250	0
4	69	276	10.4
6	48	288	15.2

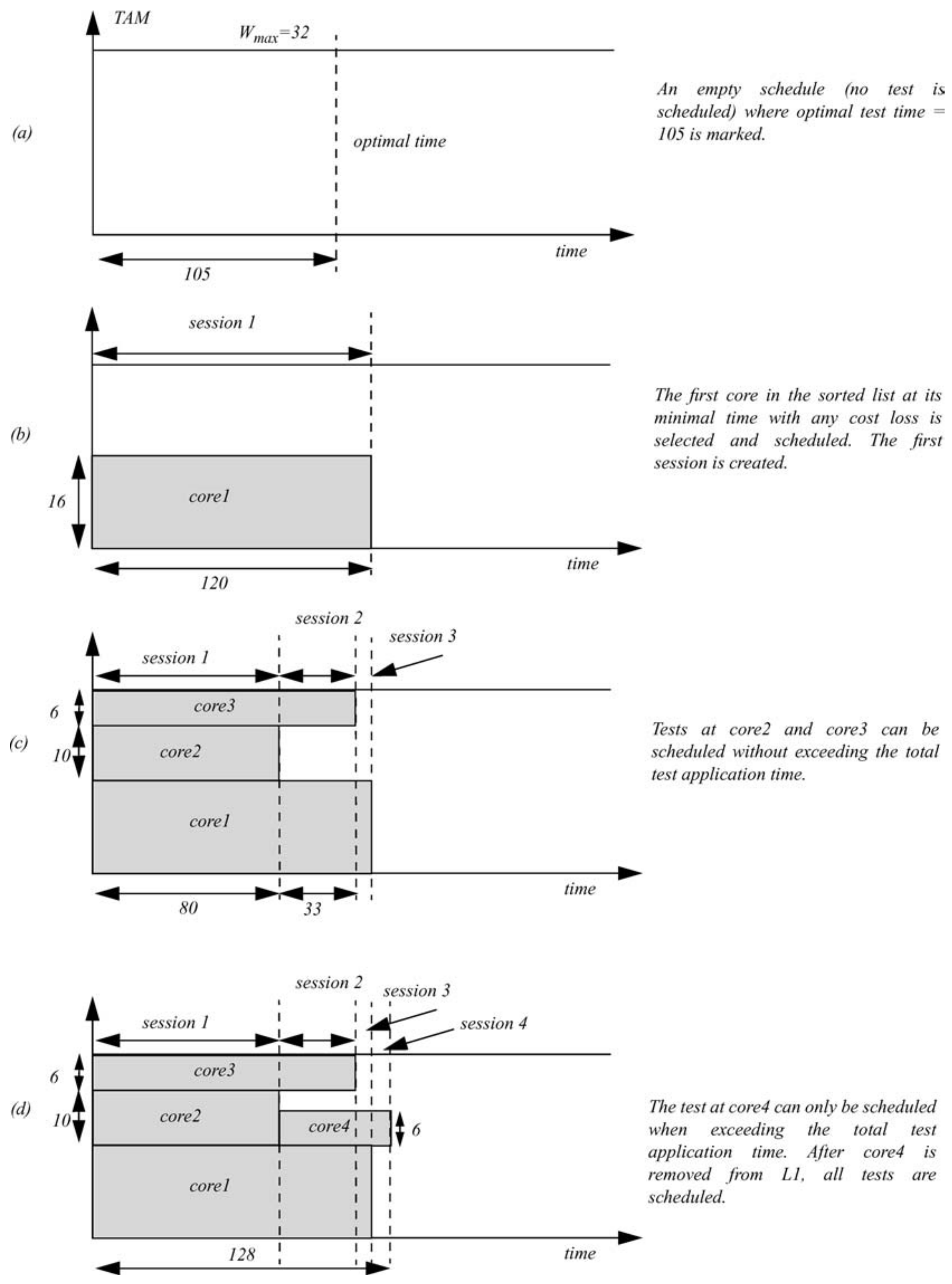
(finding tests that fit session 1). Core2 is selected at a configuration at TAM = 10, a test time of 80 and a cost loss of 8.7% (lower than the cost loss limit (16%)). After core2 has been scheduled, core3 can be scheduled. A configuration at TAM = 6 with a test time of 113 and a cost loss of 15.7% (lower than 16%) is selected. New sessions are created as core2 and core3 are scheduled (Fig. 9(c)). These sessions defines the time points that should be explored when L1 has been traversed and a new time point (t) must be found. For instance, when core4 is explored at $t = 0$, there are no available TAM-wires; hence a new t has to be found. The new t is set to the 80 (the end of session 1 and the start of session 2). It is possible to find a configuration at $t = 80$ for core4 in such a way that the cost loss is not violated (Fig. 9(d)). When core4 has been removed from L1, there are no more cores to schedule and the algorithm terminates. The total test application time for the schedule at cost loss 16% is 128. The test application time is computed for each cost loss in the range from 0 to 80% and the best test application time for all schedules is reported as the final test application time.

Table 3. Power consumption values for the tests in design d695, p22810, and p93791.

Test	d695	p22810	p93791
1	660	173	7014
2	602	173	74
3	823	1238	69
4	275	80	225
5	690	64	248
6	354	112	6150
7	530	2489	41
8	753	144	41
9	641	148	77
10	1144	52	395
11	–	2505	862
12	–	289	4634
13	–	739	9741
14	–	848	9741
15	–	487	78
16	–	115	201
17	–	580	6674
18	–	237	113
19	–	442	5252
20	–	441	7670
21	–	167	113
22	–	318	76
23	–	1309	7844
24	–	260	21
25	–	363	45
26	–	311	76
27	–	2512	3135
28	–	2921	159
29	–	413	6756
30	–	508	77
31	–	–	218
32	–	–	396

4. Experimental Results

We have implemented our test scheduling technique and performed experiments using the ITC'02 benchmarks. Note that none of the previous approaches



An empty schedule (no test is scheduled) where optimal test time = 105 is marked.

The first core in the sorted list at its minimal time with any cost loss is selected and scheduled. The first session is created.

Tests at core2 and core3 can be scheduled without exceeding the total test application time.

The test at core4 can only be scheduled when exceeding the total test application time. After core4 is removed from L1, all tests are scheduled.

Fig. 9. Test schedule on the example design at allowed cost loss of 16% at TAM width $W_{\max} = 32$.

considers more test conflicts than TAM wire sharing, except Iyengar et al. [9] who consider design hierarchy constraints in the benchmarks and Huang et al. [4] who consider test power. All other approaches except Iyengar et al. [9] assume that the designs are flat and without hierarchy constraints. We are, as discussed above, considering a variety of test conflicts, including cases when a core is tested by several tests, power limitations, and precedence constraints. These realistic assumptions, obviously, make the problem more complicated.

In the first experiment, we compared our technique with the approach presented by Huang et al. [4]. We make use of the d695 circuit with the power values used by Huang et al. [4], and given in Table 3. The results are given in Table 4 for different TAM bandwidth at different power limits. We note that the results by the two approaches are similar even if we in our approach handle different test conflicts.

In our second experiment, we compared our approach to several previously proposed techniques using d695, p22810 and p93791 without considering any power limitation. The results are given in Table 5 for a range of TAM bandwidths. We list first the lower bound given by Goel and Marinissen [11] and the *Optimal-Time* extracted from our formula above (in the columns LB [11] and Optimal Tune respectively). Then we give the test times produced by the Multiplexed approach

Table 4. Power constrained test time on design d695—Comparison between Huang et al. [4] and our approach.

Design: d695	TAM width-32		TAM width = 48		TAM width = 64		TAM width = 80		TAM width = 96		TAM width = 112		TAM width = 128	
approach:	[4]	Our	[4]	Our	[4]	Our	[4]	Our	[4]	Our	[4]	Our	[4]	Our
$P_{max} = 1500$	45560	43541	31028	32663	27573	26973	20914	24369	20914	23425	16841	19402	16841	19402
$P_{max} = 1800$	44341	42450	29919	32054	24454	23864	20467	18774	18077	18774	14974	18774	14899	16804
$P_{max} = 2000$	43221	42450	29419	29106	24171	21942	19206	18691	17825	17467	14128	14563	14128	14469
$P_{max} = 2500$	43221	41847	29023	29106	23721	21931	19206	18691	15847	17257	14128	13963	12993	13394

Table 5. Experimental results. Comparison between the Multiplexed approach [15], Pouget et al. [15], Huang et al. [4], lyengar et al. [6, 7, 9] and our approach.

		Test time								
Design	TAM width	LB [11]	Optimal time	Multiplexed [15]	Static [15]	[4]	[7]	[6]	[9]	Our
d695	128/64	10247	9584	36158	13348	11279	11604	12941	–	13348
	96/48	13659	12780	36232	19932	15142	15698	15300	–	17257
	80/40	16388	15335	36232	19932	17366	18459	18448	–	18691
	64/32	20482	19169	45798	32857	21389	23021	22268	–	20512
	48/24	27305	25559	45972	33031	28639	30317	30032	–	29106
	32/16	40951	38339	78077	65136	42716	43723	42644	–	41847
p22810	128/64	104868	105493	503088	142360	128512	136941	153990	–	128332
	96/48	139823	140578	503534	215339	167858	167256	232049	–	159994
	80/40	167787	168790	503635	223463	184951	197293	232049	–	195733
	64/32	209734	210988	531631	294046	223462	246150	246332	–	236186
	48/24	279644	281317	619537	418226	300723	307780	313607	–	352834
	32/16	419466	421976	664665	574120	446684	452639	468011	–	473418
p93791	128/64	436673	413565	639827	618150	459233	511286	473997	481896	457862
	96/48	582227	551420	672119	650402	607955	627934	599373	635710	639217
	80/40	698670	661704	1174475	1155800	719880	794020	741965	758156	787588
	64/32	873334	827131	1240170	1221495	900798	975016	894342	863765	945425
	48/24	1164442	1102841	1377123	1358448	1200157	1248795	1209420	1293990	1220469
	32/16	1746657	1654261	2432511	2432511	1791860	1851135	1786200	1927010	1827819

Table 6. Power-constrained scheduling on p22810.

p22810 TAM width	Optimal time	No P_{max} Test time	$P_{max} = 10000$ Test time	$P_{max} = 8000$ Test time	$P_{max} = 6000$ Test time	$P_{max} = 5000$ Test time	$P_{max} = 4000$ Test time	$P_{max} = 3000$ Test time
128	103 344	128 332	128 332	142 056	157 568	246 110	268 856	293 021
112	118 108	138 410	138 542	147 535	159 686	257 600	268 272	293 528
96	137 792	159 994	159 994	159 994	174 928	266 166	285 814	311 632
80	165 351	195 733	195 733	195 733	209 559	264 038	285 307	356 215
64	206 688	236 186	236 186	236 186	250 487	321 930	324 478	309 255
48	275 584	352 834	352 834	352 834	346 461	382 507	389 243	392 525
32	413 376	473 418	473 418	473 418	475 951	472 026	480 223	482 963
24	551 168	635 583	635 583	635 583	638 116	638 316	653 699	680 622
20	661 402	819 465	819 465	819 465	819 530	845 469	845 469	845 469
16	826 753	892 713	892 713	892 713	893 050	891 457	891 457	948 481
12	1 102 337	1 206 986	1 206 986	1 206 986	1 206 986	1 206 986	1 206 986	1 206 986

Table 7. Power-constrained scheduling on p93791.

p93791 TAM width	Optimal Time	No P_{max} Test time	$P_{max} = 30000$ Test time	$P_{max} = 25000$ Test time	$P_{max} = 20000$ Test time	$P_{max} = 15000$ Test time	$P_{max} = 10000$ Test time
128	424 847	457 862	457 862	493 599	472 653	486 469	568 734
112	485 539	515 020	515 020	549 669	549 669	598 487	629 051
96	566 462	639 217	639 217	639 217	658 132	631 214	691 866
80	679 755	787 588	787 588	821 475	821 575	848 050	1 091 210
64	849 694	945 425	945 425	965 383	957 921	1 014 616	1 117 385
48	1 132 924	1 220 469	1 220469	1 220 469	1 220 469	1 220 469	1 220 469
32	1 699 387	1 827 819	1 827 819	1 827 819	1 827 819	1 827 819	1 827 819
24	2 265 850	2 399 834	2 399 834	2 399 834	2 399 834	2 399 834	2 399 834
20	2 719 020	2 951 651	2 951 651	2 951 651	2 951 651	2 951 651	2 951 651
16	3 398 775	3 574 150	3 574 150	3 574 150	3 574 150	3 574 150	3 574 150
12	4 531 700	4 728 023	4 728 023	4 728 023	4 728 023	4 728 023	4 728 023

[15], Pouget et al. [15], Huang et al. [4], Iyengar et al. [6, 7, 9], and our approach for the TAM widths [4, 6, 7, 15], respectively. Note that in p22810 and p93791 there are design hierarchy constraint that we have taken into account. We give also an example of the schedule our generates at TAM width of 128 in Fig. 10 where the TAM bandwidth is on the y-axe and the test time on the x-axe.

In our last experiment, we applied our algorithm assuming different power constraint values. We made use of two designs with a high number of tests: p22810 containing 30 tests and p93791 containing 32 tests. As power values are not given for these benchmarks, we added them as depicted in Table 3 (in columns 3 and 4). The power limitations for p93791 are in the range from 30000 down to 10000 and for p22810 the range is from 10000 down to 3000 units. The results are presented in Tables 6 and 7.

The computation times of our algorithm including the wrapper design and test scheduling for the different designs reported here are all within a few seconds using an AMD 1800 machine (1.53 GHz and 512 MB RAM).

5. Conclusions

The technology development has made it possible to design and manufacture extremely complex systems. These systems have an increasing number of fault sites; hence, a high test data volume is needed to test them. In order to reduce the test cost, the test time should be reduced. In this paper we have proposed a test scheduling

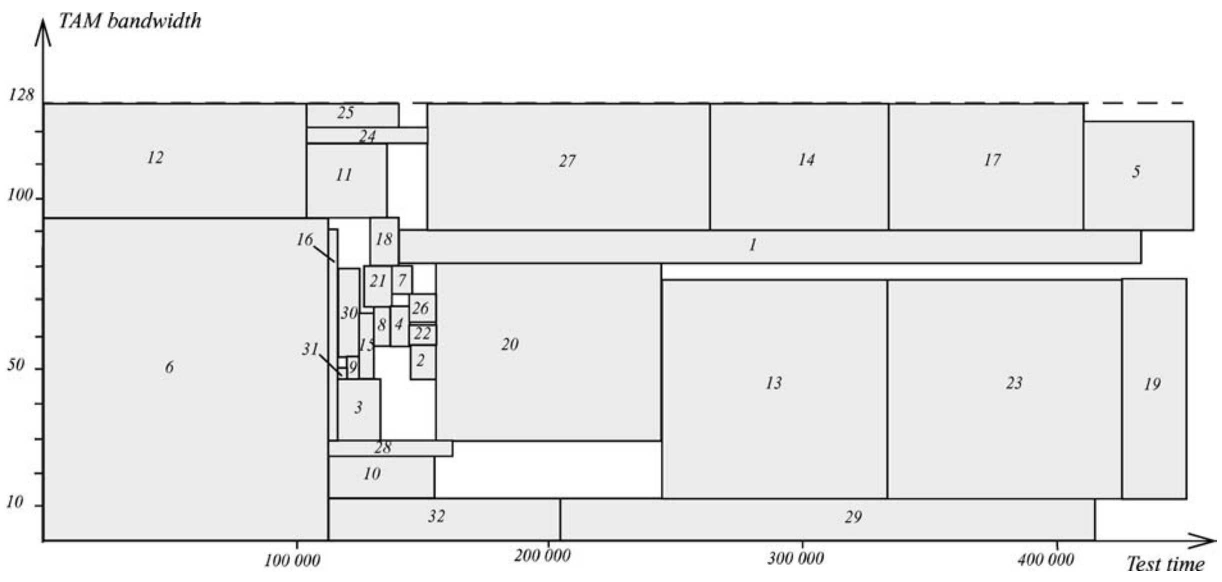


Fig. 10. Test schedule for p93791with $W_{\max} = 128$ bits.

technique that minimizes the test application time by allowing tests to be applied as concurrently as possible. The technique takes test power consumption and test conflicts into account when minimizing the test application time. It is important to consider test power consumption since exceeding the system power budget might damage the system. It is also important to take the test conflicts into account since they appear in many SOC designs. The test conflicts we consider are due to cross-core testing (interconnection testing), unit testing with multiple test sets, hierarchical SOCs where cores are embedded in cores, and the sharing of test access mechanism (TAM) wires. Another important conflict that we consider is precedence constraints, which is the order in which the tests are to be applied.

We have implemented our technique and performed several experiments to compare our technique with previous proposed approaches. The experiments show that our technique has a low computational cost and the results are comparable with other techniques which do not consider all the constraints and limitations that we are handling.

References

1. R.M. Chou, K.K. Saluja, and V.D. Agrawal, "Scheduling Tests for VLSI Systems Under Power Constraints," *IEEE Transactions on VLSI Systems*, vol. 5, no. 2, pp. 175–185, 1997.
2. E. Cota, L. Cairo, M. Lubaszewski, and A. Orailoglu, "Test Planning and Design Space Exploration in a Core-based Environment," in *Proceedings of the Design, Automation and Test in Europe Conference (DATE)*, Paris, France, 2002, pp. 478–485.
3. H-S Hsu, J-R Huang, K-L Cheng, C-W Wang, C-T Huang, and C-W Wu, "Test Scheduling and Test Access Architecture Optimization for System-on-Chip," in *Proceedings of IEEE Asian Test Symposium (ATS)*, Tamuning, Guam, USA, 2002, pp. 411–416.
4. Y. Huang, S.M. Reddy, W-T Cheng, P. Reuter, N. Mukherjee, C-C Tsai, O. Samman, and Y. Zaidan, "Optimal core wrapper width selection and SOC test scheduling based on 3-D bin packing algorithm," in *Proceedings IEEE of International Test Conference (ITC)*, Baltimore, MD, USA, 2002, pp. 74–82.
5. V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "Test Wrapper and Test Access Mechanism Co-Optimization for System-on-Chip," *Journal of Electronic Testing; Theory and Applications (JETTA)*, pp. 213–230, 2002.
6. V. Iyengar K. Chakrabarty, and E.J. Marinissen, "Efficient Wrapper/TAM Co-Optimization for Large SOCs," in *Proceedings of Design and Test in Europe (DATE)*, Paris, France, 2002, pp. 491–498.
7. V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "On Using Rectangle Packing for SOC Wrapper/TAM Co-Optimization," in *Proceedings of IEEE VLSI Test Symposium (VTS)*, Monterey, California, USA, 2002, pp. 253–258.
8. V. Iyengar, S.K. Goel, E.J. Marinissen, and K. Chakrabarty, "Test Resource Optimization for Multi-Site Testing of SOCs under ATE Memory Depth Constraints," in *Proceedings of IEEE International Test Conference*, Baltimore, MD, USA, 2002, pp. 1159–1168.
9. V. Iyengar, K. Chakrabarty, M.D. Krasniewski, and G.N. Kuma, "Design and Optimization of Multi-level TAM Architectures for Hierarchical SOCs," in *Proceedings of IEEE VLSI Test Symposium (VTS)*, 2003, pp. 299–304.

10. S.K. Goel and E.J. Marinissen, “Cluster-Based Test Architecture Design for System-On-Chip,” in *Proceedings of IEEE VLSI Test Symposium (VTS)*, Monterey, California, USA, 2002, pp. 259–264.
11. S.K. Goel and E.J. Marinissen, “Effective and efficient test architecture design for SOCs,” in *Proceedings of IEEE International Test Conference (ITC)*, Baltimore, MD, USA, 2002, pp. 529–538.
12. S. Koranne, “On Test Scheduling for Core-based. SOCs,” in *Proceedings of International Conference on VLSI Design*, Bangalore, India, 2002, pp 505–510.
13. S. Koranne and V. Iyengar, “On the use of k - tuples for SoC test schedule representation,” in *Proceedings of International Test Conference (ITC)*, Baltimore, MD, USA, 2002, pp. 539–548.
14. E.J. Marinissen, R. Kapur, and Y. Zorian, “On Using IEEE P1500 SECT for Test Plug-n-play,” in *Proceedings of IEEE International Test Conference (ITC)*, Atlantic City, NJ, USA, 2000, pp. 770–777.
15. J. Pouget, E. Larsson, Z. Peng, M.-L. Flottes, and B. Rouzeyre, “An Efficient Approach to SoC Wrapper Design, TAM configuration, and Test Scheduling,” in *Proceedings of IEEE European Test Workshop (ETW)*, Maastricht, The Nederlands, 2003, pp. 117-122.
16. J. Pouget, E. Larsson, and Z. Peng, “SOC Test Time Minimization Under Multiple Constraints,” in *Proceedings of Asian Test Symposium (ATS)*, Xian, China, 2003, pp. 312–317.

Julien Pouget received his M.Sc degree and PhD degree from Montpellier University, LIRMM, France, in 1999 and 2002, respectively. He was a guest researcher at Embedded Systems Laboratory, Linköpings Universitet, Sweden, from September 2002 to October 2003, and during 2004 he was an assistant professor at ISIM, The Microelectronic Engineering School, Montpellier, France. Currently, he is a DFT engineer at ST Microelectronics, Grenoble, France. His main research interest is on system-on-chip test scheduling and test architecture co-optimization.

Erik Larsson received his M.Sc., Tech. Lic and Ph.D from Linköping University in 1994, 1998, 2000, respectively. From October 2001 to December 2002 he was at a Japan Society for the Promotion of Science (JSPS) funded Post Doc position at the Computer Design and Test Laboratory at Nara Institute of Science and Technology (NAIST), Nara, Japan. Currently, he is Assistant Professor and Director of Studies of the Division for Software and Systems at the Department of Computer and Information Science, Linköpings Universitet, Sweden.

His current research interests include the development of tools and design for testability methodologies to facilitate the testing of complex digital systems. The main focuses are on system-on-chip test scheduling and test infrastructure design.

He is author of the book “Introduction to Advanced System-on-Chip Test Design and Optimization” (Springer 2005) and is co-guest editor for the IEE Computers & Digital Techniques special issue on “Resource-Constrained Testing of System Chips”. He received the best paper award for the paper ”Integrated Test Scheduling, Test Parallelization and TAM Design” at IEEE Asian Test Symposium (ATS), 2002, and he has supervised the thesis, which was selected as the best thesis by Föreningen Svenskt Näringsliv, 2002, and the thesis, which was selected as the best thesis at the Department of Computer Science, 2004.

He is a member of the program committee of Design and Test Automation in Europe (DATE), 2004, 2005, 2006, IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2004, 2005, 2006, and The Workshop on RTL ATPG & DFT (WRTLT), 2005, 2006.

Zebo Peng received the B.Sc. degree in Computer Engineering from the South China Institute of Technology, China, in 1982, and the Licentiate of Engineering and Ph.D. degrees in Computer Science from Linköping University, Sweden, in 1985 and 1987, respectively.

Currently, he is Professor of Computer Systems, Director of the Embedded Systems Laboratory, and Chairman of the Division for Software and Systems in the Department of Computer Science, Linköping University. His research interests include design and test of embedded systems, electronic design automation, SoC testing, design for testability, hardware/software co-design, and real-time systems. He has published over 180 technical papers in these areas and co-authored the books “System Synthesis with VHDL” (Boston: Kluwer, 1997), ”Analysis and Synthesis of Distributed Real-Time Embedded Systems” (Boston: Kluwer, 2004), and “System-level Test and Validation of Hardware/Software Systems” (London: Springer, 2005).

Prof. Peng was co-recipient of four best paper awards, two at the European Design Automation Conferences (EURO-DAC’92 and EURO-DAC’94), one at the IEEE Asian Test Symposium (ATS’02), and one at the Design, Automation and Test in Europe Conference (DATE’05), as well as a best presentation award at the IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (2003). He has served on the program committee of a dozen international conferences and workshops, including ATS, ASP-DAC, DATE, DDECS, DFT, ETS, ITSW, MEMOCDE and VLSI-SOC. He was the General Chair of the 6th IEEE European Test Workshop (ETW’01), the Program Chair of the 7th IEEE Design & Diagnostics of Electronic Circuits & Systems Workshop (DDECS’04), and the Test Track Chair of the 2006 Design Automation and Test in Europe Conference (DATE’06). He is the Vice-Chair of the IEEE European Test Technology Technical Council (ETTTC).