Transactions Briefs

System-on-Chip Test Scheduling With Reconfigurable Core Wrappers

Erik Larsson and Hideo Fujiwara

***Abstract*—The problem with increasing test application time for testing core-based system-on-chip (SOC) designs is addressed with test architecture design and test scheduling. The scan-chains at each core are configured into a set of wrapper-chains, which by a core wrapper are connected to the test access mechanism (TAM), and the tests are scheduled in such a way that the test time is minimized. In this paper, we make use of reconfigurable core wrappers that, in contrast to standard wrappers, can dynamically change (reconfigure) the number of wrapper-chains during test application. We show that by using reconfigurable wrappers the test scheduling problem is equivalent to *independent job scheduling on identical machines*, and we make use of an existing preemptive scheduling algorithm that produces an optimal solution in linear time ($O(n)$; n is the number of tests). We also show that the problem can be solved without preemption, and we extend the algorithm to handle: 1) test conflicts due to interconnection testsre and 2) cases when the test time of a core limits an optimal usage of the TAM. The overhead in logic is given by the number of configurations, and we show that the upper-bound is three configurations per core. We compare the proposed approach with the existing technique and show, in comparison, that our technique is 2% less from lower bound.**

***Index Terms*—Preemptive scheduling, reconfigurable core wrapper, system-on-chip (SOC), test access mechanism (TAM) design, test scheduling, test time minimization.**

I. Introduction

Testing is required to ensure the production of fault-free chips. However, the cost of test application is increasing and it is highly related to the test application time. The problem is that the increasing test data volume, due to complex chips, leads to long test application time. It is, therefore, important to minimize the test application time.

Complex chips are often designed in a modular fashion where reusable blocks of logic, so called cores, are integrated to a system-on-chip (SoC) [17]. Examples of cores are CPUs, DSPs, mixed-signal modules, and memories. Each core in an SoC has its dedicated test. In order to reduce the test application time, concurrent testing where multiple cores are tested at the same time, instead of sequential testing, only one test at a time, can be used; however, constraints must be considered carefully. In a core-based design, major constraints are related to the interface between the test access mechanism (TAM) and the core, the core wrapper.

Zorian *et al.* [17] introduced the concept: 1) test source and test sink; 2) TAM; and 3) wrapper. The test source is where test stimuli are stored, and the test sink is where the test responses are analyzed. Automatic test equipment (ATE) is often used as both test source and test sink. The TAM is used for the transportation of test stimuli from the test source to a core under test, and for the transportation of test responses from a core under test to the test sink. The core wrapper isolates the core from the surrounding logic during testing and acts as the interface between the wrapper-chains (the chains of scanned elements at each core) and the TAM.

A core with a dedicated interface to the TAM is said to be *wrapped*, while a core such as a user-defined logic (UDL) block or the interconnection wires between cores, is said to be *unwrapped*. This can be used to classify the tests in the system:

- *core test*—tests the core logic of a wrapped (isolated) core or fully isolated logic block;
- *interconnection test*—tests unwrapped cores, interconnections between cores, and UDL.

The TAM is organized according to an architecture such as multiplexing architecture [1], daisy-chain architecture [1], distribution architecture [1], bus-based architecture [16], or a bus-based architecture with or without fork and merge of TAMs [7].

For cores with a dedicated wrapper, several test scheduling techniques have been proposed [4]–[6], [8]–[13]. However, cores without dedicated interface (wrapper) to the TAM are not considered.

In this paper we explore the use of reconfigurable core wrappers, which in contrast to standard wrapper can be reconfigured during test application.

The main contributions of the paper are that we:

- demonstrate that the problem of scheduling tests on the TAM is equivalent to the independent job (=tests) scheduling on identical machines (=TAM wires);
- show that a preemptive scheduling algorithm producing optimal solution in $O(n)$ time for n jobs (tests) [2] can be used;
- demonstrate that the algorithm [2] can be modified such that preemption is not used while producing optimal solution;
- modify the algorithm [2] such that if the test time of a core limits an optimal solution, the test time is modified such that optimal solution is reached;
- extend the algorithm to handle cores without dedicated wrappers.

The advantages of the proposed approach are that we implicitly minimize the TAM bandwidth at each core, which implicitly minimizes the routing cost independently of the floor-plan, and we only make use of a minimum of reconfigurable configurations when using the reconfigurable wrapper, which minimizes the added logic, where the cost of routing minimization is most important in future designs [3].

The rest of the paper is organized as follows. In Section II, we formulate the problem, introduce the system model, and preemptive test scheduling. The scheduling approach is presented in Section III and experimental results are in Section IV. The paper is concluded in Section V.

II. Problem Definition

We assume that given a core-based system where for each core the number of test vectors, the length, and number of scan-chains, as well as the number of wrapper input cells, output cells, and bidirectional cells are given, i.e., $C = \{c_1, c_2, \ldots, c_n\}$ is a finite set of n cores where each core $c_i \in C$ is characterized by:

$sc_{j,i}$	length of scan-chain j at core i;
wi_i	number of wrapper input cells;

Manuscript received August 7, 2002; revised October 18, 2004. The work was supported in part by the Japan Society for the Promotion of Science (P01735) and the Swedish National Program STRINGENT.

E. Larsson is with Embedded Systems Laboratory, Department of Computer Science, Linköpings Universitet, SE-58183 Linköpings, Sweden (e-mail: erila@ida.liu.se).

H. Fujiwara is with the Department of Information Processing, Nara Institute of Science and Technology (NAIST), Nara 630-0192, Japan.

Digital Object Identifier 10.1109/TVLSI.2006.871757

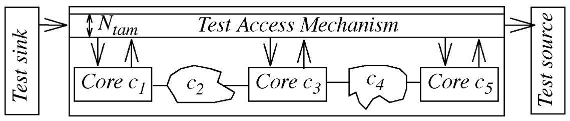

Fig. 1. Core-based system with five cores (c_1, c_2, c_3, c_4, and c_5).

wo_i	number of wrapper output cells;
wb_i	number of wrapper bidirectional;
tv_i	number of test vectors;
s_i	core sending test stimuli to core i;
r_i	core receiving test responses from core i.

For the system $W_{\rm tam} = \{w_1, w_2, \ldots, w_m\}$ is a finite set of m wires, and $N_{\rm tam} = |W_{\rm tam}| = m$.

The notation of s_i and r_i is used to handle interconnection tests, where it is important to keep information on which wrapper is used to receive test stimuli from the TAM and to send the produced test responses on the TAM. If we have a core test, it means that $c_i = s_i = r_i$, the wrapper at core i is responsible for both receiving test stimuli from the TAM and sending the test response on the TAM. For instance, in Fig. 1 when testing c_1, no other core is required $(s_1 = r_1 = c_1)$. On the other hand, when testing core c_2, both core c_1 and core c_3 are required, i.e., the sending core is $s_2 = c_1$ and the receiving core is $r_2 = c_3$.

The test time for core c_i, assuming w wrapper-chains is computed by [4]

$$\tau_i(w) = (\max\{sc_{\rm in}, sc_{\rm out}\} + 1) \times tv_i + \min\{sc_{\rm in}, sc_{\rm out}\} \quad (1)$$

where $sc_{\rm in}$ is the shift-in time and $sc_{\rm out}$ is the shift-out time.

We let τ_i denote the test time at core c_i, assuming one (1) single wrapper-chain, $\tau_i = \tau_i(w)$ where $w = 1$.

For preemptive scheduling, which means that each test can be split into k number of partitions, we have to determine for each core c_i at each partition k:

$tv_{i,k}$	number of test vectors to apply;
$n_{i,k}$	number of TAM wires to use;
$str_{i,k}$	start time;
${\rm end}_{i,k}$	end time.

For each core i, all test vectors must be applied, that is

$$tv_i = \sum_{\forall k} tv_{i,k}. \quad (2)$$

Given the above, the main objective is to minimize:

- total test application time;
- added logic due to reconfigurable wrappers;
- wiring of TAM wires connecting the cores.

The added logic comes from the reconfigurable wrapper and it depends on the number of configurations/partitions k (discussed next). It means that minimizing the number of configurations, in turn, minimizes the added wrapper logic. The routing of TAM wires depends on the number of wires connected to each core (the number of wrapper-chains) and, hence, the routing is minimized by minimizing the number of TAM wires at each core.

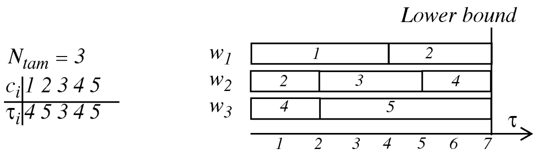

Fig. 2. Core-based system with five cores (c_1, c_2, c_3, c_4, and c_5).

III. Test Scheduling

A. Scheduling Core Tests

For a core-based design with core tests, each core can be tested independent of all other cores. The main problem is to assign TAM wires to each core and a start time for each test in such a way that the test application time is minimized.

The test data (test stimuli and test responses) for a core test are transported to and from the cores via the TAM, where the TAM is a set of wires. The problem is the configuration of scanned elements into wrapper-chains for each core, the assignment of a number of TAM wires to each test, and the assignment of a start time for each test. The number of TAM wires is a limiting factor since in order to minimize the test time each core cannot be assigned to its private TAM wires. As soon as wires are shared between cores, there is a conflict due to TAM wire sharing. Overlapping in time is not allowed.

In a system with core tests only, the tests are not dependent on each other; hence, they are independent. Each core has a wrapper that isolates it from the rest of the system during testing. The tests can, therefore, be seen as independent jobs. Furthermore, the TAM wires are all identical. Test stimuli/responses can be transported from/to the ATE from/to a core on any TAM wire; hence, the TAM wires are identical.

As the core tests are independent and each TAM wire is identical, it means that the *independent job scheduling on identical machines* is equivalent to the scheduling of tests on the TAM by letting each TAM wire be an identical machine and each core test be a job.

The problem is to schedule the independent jobs (the tests at core c_i, $(i = \{1, \ldots, n\})$) each with a testing time τ_i, on the identical machines (TAM wires) $w_j (j = \{1, \ldots, N_{\rm tam}\})$.

All tests are known in advance, as well as the TAM bandwidth, and it makes it possible to compute the lower bound (LB) as [2]

$$LB = \max\left\{\max_{\forall i}(\tau_i), \left\lceil \frac{\sum_{i=1}^{n} \tau_i}{N_{\rm tam}} \right\rceil\right\}. \quad (3)$$

The independent job scheduling on identical machines problem can be solved in $O(n)$ time by using preemption [2]: assign tests to the TAM wires successively and assign the tests in any order and preempt tests into two parts whenever the LB is met. Assign the second part of the preempted test on the next TAM wire at zero time. Repeat until all tests are assigned to TAM wires.

An example with five core tests will illustrate the approach (Fig. 2). The LB is computed to 7 by (3) and due to that, $\tau_i \leq LB$ for all tests; the two parts of a preempted test will not overlap. The scheduling proceeds as follows: the tests are selected in order, starting with the test at c_1, which is assigned to wire w_1 at time zero. At time four, at the end of the testing of c_1, the test at c_2 is assigned to wire w_1. The full test cannot be assigned to wire w_1 at time 7 when LB is reached; it is preempted and the rest of the test is assigned to start at time zero on wire w_2. The assignment of tests/cores to wires continues until all cores/tests are assigned.

Executing the test at c_2, for instance, starts with the use of wire w_2 during time period zero to two, followed by the use of wire w_1 during

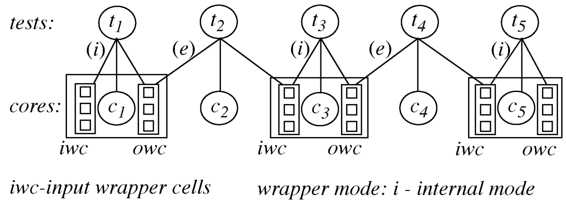

Fig. 3. Resource graph for test conflict illustration.

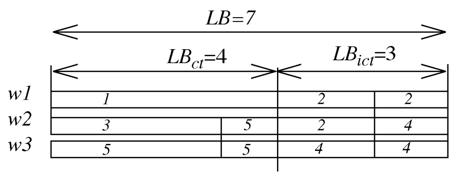

Fig. 4. Core testing and interconnection/cross-core testing.

time period four to seven. At preemption of a test, another wire is assigned to the core and a multiplexer is to be added for wire selection. For the test of c_2, a multiplexer is added to select between w_1 and w_2.

In general preemptive scheduling, extra time is introduced at a preemption point to set up a new job and to save the state of the old (preempted) job. The main reason is that the machine is to be used by another job. However, in testing, no other tasks are performed at the cores besides testing. It means that the core's state can be left as it is until the testing continues. The advantage is that the state of the core is already set and testing of it can start at once. Assume that core c_2 has a wrapper-chain of length l (l cycles are needed to shift in a new vector and shift out the previous test response). If the test is preempted when $x\%$ of the l cycles are applied, it means that when the test restarts $x\%$ of the new test vector is already loaded and $x\%$ less cycles are needed in the first shift process, i.e., there is no time overhead due to setting up and saving the state of a core; all tests can be stopped at LB.

B. Scheduling Core Tests and Interconnection Tests

The interconnection tests must also be scheduled, which means that the test conflicts, due to wrapper conflicts, must be handled. The test conflicts in a system such as the example system (Fig. 1), can be modeled using a resource graph (Fig. 3). An arc between a test and a resource (core or a wrapper cell) indicates that such test requires the resource during testing. In Fig. 3, we mark core tests with (i), internal mode, and interconnection tests with (e), external mode.

Our approach is to break the resource graph into two resource graphs, one for core tests and one for interconnection tests. We then schedule all core tests first and after that, all interconnection tests are scheduled. By doing so, the conflicts are eliminated. We have now shown that partitioning the tests into two partitions, core tests and interconnection tests, eliminates the test conflict. We make use of that and divide the test scheduling into two consecutive parts, core testing followed by interconnection testing. The partition of the tests means we divide the tests into a core test part given by LB_{ct}, and an interconnection test part given by $LB_{\rm ict}$. To illustrate, we take the example in Fig. 1, where the test at c_2 and c_4 are interconnection tests, which means that executing the test at c_2 will inhibit concurrent testing at c_1 and at c_3; and c_4 will inhibit concurrent testing at c_3 and at c_5. The core tests are at core c_1, c_3, and c_5, and the interconnection tests are at core c_2 and c_4. The LB is computed to 7 $((4+5+3+5+4)/3)$ and for the core tests; $LB_{ct} = (4+3+5)/3 = 4$ and for the interconnection tests: $LB_{ict} = (5+4)/3 = 3$, i.e., $LB = LB_{ct} + LB_{ict}$. The test schedule is presented in Fig. 4.

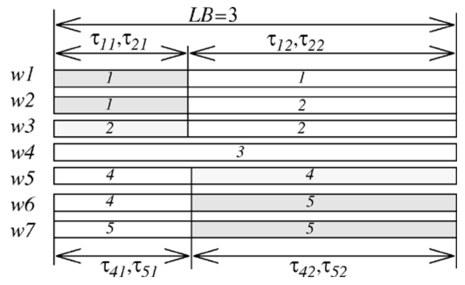

Fig. 5. Test schedule on a wider TAM.

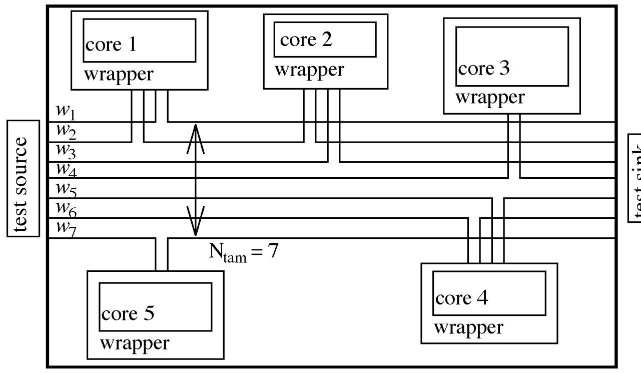

Fig. 6. TAM routing of the example design Fig. 1 with test schedule as Fig. 5.

The test scheduling algorithm consists of four steps:

- compute LB_{ct} (lower bound) for the core tests;
- schedule all core tests;
- compute $LB_{\rm ict}$ for the interconnection tests;
- schedule all interconnection tests.

C. Transformations for Optimal TAM Utilization

Long test times for one of the tests in the system may limit the solution, i.e., LB is given by the test time of a test [$\max(\tau_i)$ in (3)]. In such a case, the testing time for the core that limits the solution can be reduced by increasing its number of wrapper-chains. It then means that (3) will be modified to

$$LB = \left\lceil \frac{\sum_{i=1}^{n} \tau_i}{N_{\rm tam}} \right\rceil . \quad (4)$$

After the computation of LB, we use the scheduling approach described above. For an illustration of the test schedule at a wider TAM ($N_{tam} = 7$), we use the same example (Fig. 2), and the scheduling result is presented in Fig. 5. Above, a test could not overlap since $\tau_i \leq LB$, however, now overlapping is possible. For instance, the test at c_1 uses wire w_1 and w_2 during time period zero to one and wire w_1 during period one to three.

We make use of a reconfigurable wrapper that allows the bandwidths at each core to be changed dynamically during the execution [11], [13]. The multiplexing logic and the control logic is reduced when the number of configurations is reduced. For instance, if only one configuration exists, i.e., one wrapper-chain, no multiplexing logic is required at all.

D. Logic due to Reconfigurable Wrapper and TAM Routing

In the approach, the maximum number of partitions per test is three. In the example (Fig. 5), no test is partitioned into more than two parti-

TABLE I
TEST TIME COMPARISON ON P93791

Approach	Test Application Time						
	$N_{tam} = 16$	$N_{tam} = 24$	$N_{tam} = 32$	$N_{tam} = 40$	$N_{tam} = 48$	$N_{tam} = 56$	$N_{tam} = 64$
Lower bound [9]	1746657	1164442	873334	698670	582227	499053	436673
Enumerate [4]	1883150	1288380	944881	929848	835526	537891	551111
ILP [4]	1771720	1187990	887751	(698583)	599373	514688	460328
Par-eval [5]	1786200	1209420	894342	741965	599373	514688	473997
GRP [6]	1932331	1310841	988039	794027	669196	568436	517958
Cluster [8]	-	-	947111	816972	677707	542445	467680
Binpack [10]	1791860	1200157	900798	719880	607955	521168	459233
CPLEX [11]	1818466	(1164023)	919354	707812	645540	517707	453868
ECTSP [11]	1755886	(1164023)	919354	707812	**585771**	517707	453868
ECTSP1 [11]	1807200	1228766	967274	890768	631115	562376	498763
TB-serial [9]	1791638	1185434	912233	718005	601450	528925	455738
TR-serial [9]	1853402	1240305	940745	786608	628977	530059	461128
TR-parallel [9]	1975485	1264236	962856	800513	646610	540693	477648
K-tuple [12]	2404341	1598829	1179795	1060369	717602	625506	491496
Our approach	**1752336**	**1174252**	**877977**	**703219**	592214	**511925**	**442478**

tions. However, if the test at c_2 would terminate after time 1 but before time 3, (LB) another partition would be created.

The added logic due to reconfigurable wrappers depends on the number of cores and the number of configurations. It means that, the added logic is in the worst case in the range $3 \times |C|$ (maximum three configurations per core). In the approach by Koranne the added logic, if a reconfigurable wrapper is added at all cores, is given by $N_{\mathrm{tam}} \times |C|$.

The TAM routing is minimized by minimizing the number of TAM wires routed to each core regardless of the floor-plan. If the floor-plan is known, the tests can be sorted based on placement (the technique above does not depend on any sorting) and the assignment can start, for instance, in the upper left corner and end in the lower-left corner. We take the system in Fig. 1 with $N_{\mathrm{tam}} = 7$, resulting in a test schedule as in Fig. 5, where the cores are sorted (and numbered) clockwise as in Fig. 6. The advantage is that neighboring cores share TAM wires. For instance, core c_2, which makes use of TAM wire w_2 as soon as c_4 finishes its use of w_2. Cores placed far away from each other are not sharing TAM wires, such as c_5 and c_3.

E. Scheduling Without Preemption

In some cases, for some types of memories such as DRAMs, the testing cannot be preempted since these cores cannot hold their state (data is lost). For instance, it means that tests at c_2 cannot be preempted as in Fig. 2. In such a case, when LB is met, we let the TAM wires assignment algorithm restart at LB (and not at time zero) and move towards zero.

IV. EXPERIMENT RESULTS

We have shown above that the test scheduling problem for core-based systems can be solved in linear time using reconfigurable core test wrappers. We have, nevertheless, implemented our approach and using the P93791 [14] design, we have made a comparison with previous approaches where we assume that all cores are core tests.

In our approach, as discussed above, each test set can be partitioned into no more than three partitions (Section III-D), and to assign wrapper-chains and compute the test time at each partition, we have used the algorithm presented by Iyengar *et al.* [4]. The results are collected in Table I, where for each bandwidth the best solution is marked in bold (solutions with a test time lower than the lower bound computed by Goel and Marinissen are not considered). The best results are collected in Fig. 7, and it is clear that the approaches in general

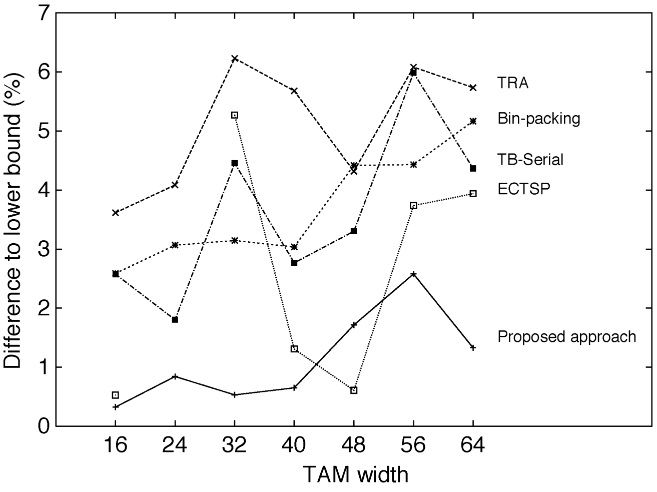

Fig. 7. Best test scheduling results in relation to lower bound.

produce good results since all results are in the range of 6% from lower bound.

The proposed approach produces results that are less than 2% from lower bound. In one case (at TAM width 48), the proposed approach is not the one that produces the best result. And actually, Fig. 7 shows that the results from the proposed technique, relative to LB, gets worse at higher TAM width. The explanation is as follows. The test time for a core decreases when a higher number of wrapper-chains are used. It is further shown in Fig. 8, that the difference $(\tau_i(w) \times w - \tau_i)$ varies over the TAM width. The main reason is that the scan-chains at a core can be few and of unequal length, and that results in unbalanced wrapper-chains. Fig. 8 shows that a high number of wrapper-chains at a core, increases the term $(\tau_i(w) \times w)$. As few wrapper-chains as possible is usually best, however, at certain $u > w$, the term $\tau_i(w) \times w < \tau_i(u) \times u$, and, hence, our solution will not be the best (as illustrated in Fig. 7 at TAM width $w = 48$). Note, that our technique tries to assign as few TAM wires as possible, and this problem only occurs when the TAM width is relatively high or if one core dominates the testing times. The problem with unbalanced scan-chains can be avoided by designing cores with a high number of short scan-chains. It would make it easier to reuse such a core and use it at a different number of wrapper-chains at different designs. We have shown for one core that by redesign of the scan-chains into a higher number of shorter and balanced scan-chains, the test time decreases nearly linearly with the

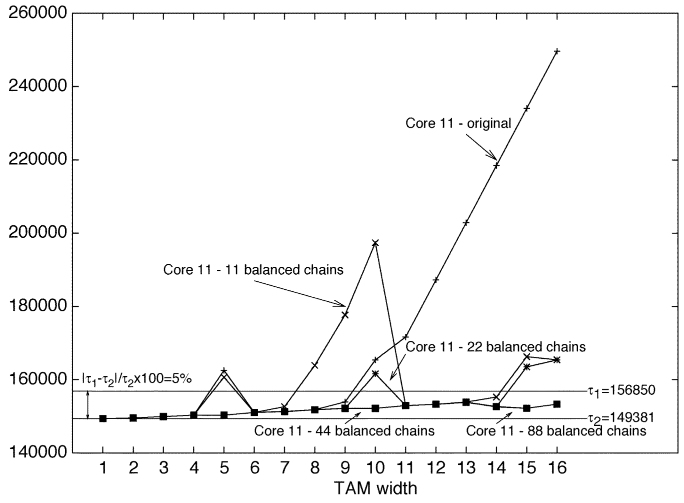

Fig. 8. Test time in relation to lower bound $(\tau_i(w) \times w - \tau_i)$ at various TAM widths.

number of wrapper-chains ($\tau_i(w) \times w$ becomes constant) as shown in Fig. 8.

V. Conclusion

We have shown that the scheduling of tests on the TAM is equal to the independent job scheduling on identical machines for which it is known that an optimal solution can be found in linear time using preemptive scheduling. We have extended the preemptive scheduling algorithm: 1) for cases where a test limits the solution by making use of reconfigurable core test wrappers and 2) to consider test conflicts due to interconnection test; the testing of interconnections and user-defined logic. We have also shown how the problem can be solved without preemption. The main advantages of our approach, are that we minimize the TAM bandwidth at each core, which implicitly minimizes the routing of TAM wires, and for the cores with reconfigurable wrapper, the number of configurations is minimized, which minimizes the added test logic. The TAM routing can easily be minimized further by sorting the cores based on the floor-plan. We have implemented the algorithm for a comparison with previous approaches. The results show that our technique is only 2% from the theoretical lower bound.

References

[1] J. Aerts and E. J. Marinissen, "Scan chain design for test time reduction in core-based ICs," in *Proc. Int. Test Conf. (ITC)*, 1998, pp. 448–457.

[2] P. Brucker, *Scheduling Algorithms*. New York: Springer-Verlag, 1998.

[3] A. L. Crunch, *Design for Test*. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[4] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "Co-optimization of test wrapper and test access architecture for embedded cores," *J. Electron. Testing: Theory Appl. (JETTA)*, vol. 18, no. 2, pp. 213–230, Apr. 2002.

[5] ——, "Efficient wrapper/TAM co-optimization for large SOCs," in *Proc. Des. Test Eur. (DATE)*, 2002, pp. 491–498.

[6] ——, "On using rectangle packing for SOC wrapper/TAM co-optimization," in *Proc. VLSI Test Symp. (VTS)*, 2002, pp. 253–258.

[7] ——, "Test access mechanism, test scheduling, and test data reduction for system-on-chip," *IEEE Trans. Comput.*, vol. 52, no. 12, pp. 1–14, Dec. 2003.

[8] S. K. Goel and E. J. Marinissen, "Cluster-based test architecture design for system-on-chip," in *Proc. VLSI Test Symp. (VTS)*, 2002, pp. 259–264.

[9] ——, "Effective and efficient test architecture design for SOCs," in *Proc. Int. Test Conf. (ITC)*, 2002, pp. 529–538.

[10] Y. Huang, W. T. Chen, 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," in *Proc. Asian Test Symp. (ATS)*, 2001, pp. 265–270.

[11] S. Koranne, "A novel reconfigurable wrapper for testing embedded core-based and its associated scheduling," *J. Electron. Testing: Theory Appl. (JETTA)*, vol. 18, pp. 415–434, Aug. 2002.

[12] S. Koranne and V. Iyengar, "On the use of k-tuples for SoC test schedule representation," in *Proc. Int. Test Conf. (ITC)*, 2002, pp. 539–548.

[13] S. Koranne, "Design of reconfigurable access wrappers for embedded core based SoC test," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 5, pp. 955–960, Oct. 2003.

[14] E. J. Marinissen, V. Iyengar, and K. Chakrabarty, "A set of benchmarks for modular testing of SOCs," in *Proc. Int. Test Conf. (ITC)*, 2002, pp. 519–528.

[15] *IEEE 1500 Standard for Embedded Core Test (SECT)*, [Online]. Available: http://grouper.ieee.org/groups/1500/

[16] P. Varma and S. Bhatia, "A structured test re-use methodology for core-based system chips," in *Proc. Int. Test Conf. (ITC)*, 1998, pp. 294–302.

[17] Y. Zorian, E. J. Marinissen, and S. Dey, "Testing embedded-core based system chips," in *Proc. Int. Test Conf. (ITC)*, 1998, pp. 539–548.