IEEE the 12th Asian Test Symposium (ATS'03), pp.306-311, Nov. 2003.

Optimal System-on-Chip Test Scheduling

Erik Larsson*+ and Hideo Fujiwara*

Embedded Systems Laboratory+
Linköpings Universitet
SE-581 83 Linköping,
Sweden
erila@ida.liu.se

Computer Design and Test Laboratory*
Nara Institute of Science and Technology,
8916-5 Takayama, Ikoma, Nara 630-0101,
Japan
fujiwara@is.aist-nara.ac.jp

Abstract[1]

In this paper, we show that the scheduling of tests on the test access mechanism (TAM) is equivalent to independent job scheduling on identical machines and we make use of an existing preemptive scheduling algorithm to produce an optimal solution in linear time. We extend the algorithm to handle (1) test conflicts due to interconnection tests and (2) cases when a test limits an optimal usage of the TAM by using reconfigurable core test wrappers. Our extensions preserve the production of an optimal solution in respect to test time and minimizes the number of wrapper configurations as well as the TAM usage at each core, which implicitly minimizes the TAM routing. Experiments with our implementation shows its efficiency in comparison with previous approaches.

1 Introduction

The high test times of core-based systems is becoming a problem. It can be minimized by an efficient scheduling of the tests on the test access mechanism (TAM), which is added for the transportation of test data from and to the automatic test equipment (ATE).

A core test wrapper is the interface connecting a core with the TAM. A *wrapped* core is a core with an interface to the TAM while a core without a dedicated wrapper is defined as *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, and
- *interconnection test* - tests unwrapped cores, interconnections between cores, and user-defined logic (UDL).

For *core tests,* several scheduling techniques of the tests on the TAM minimizing the total test time have been proposed [3,4,5,6,8,10,11]. Common drawbacks are that *interconnection tests* and costs as added logic and TAM routing are not considered.

The main contributions of this 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),
- we use of a preemptive scheduling algorithm producing optimal solution in $O(n)$ time for n jobs (tests) [2],
- we propose an extension to the algorithm to adjust the test times to fully utilize wide TAMs by using reconfigurable test wrappers proposed by Koranne [10],
- we extend the optimal preemptive TAM scheduling algorithm to handle interconnection tests.

The advantages of our approach, besides that we achieve optimal test time in linear time, are that we implicitly minimize the TAM bandwidth at each core, which implicitly minimizes the routing independently of the floorplan, 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 of most importants in future designs [1].

The rest of the paper is organized as follows. In Section 2, we present an overview of related work and in Section 3 we formulate the problem, introduce our system model and preemptive test scheduling. Our scheduling approach is presented in Section 4 and experimental results are in Section 5. The paper is concluded in Section 6.

2 Related Work

Several scheduling techniques for *core tests* where TAM wires are selected for each wrapper with the objective to minimize the total test time have been proposed [3,4,5,6,8,10,11]. All approaches assume that: (1) any TAM wire can be assigned to any core, (2) once a test is started it cannot be interrupted, *i.e.* preemption is not allowed, and finally in all approaches but Koranne's [10] (3) one fixed set of TAM wires (bandwidth) is allowed at each core wrapper.

The basic idea in the proposed techniques is that each test defines an area given by its test time and its usage of TAM wires. Transformations can be made for each test set where increasing the TAM width reduces the test time and vice versa. Different techniques are used for the selection of number of TAM wires per core wrapper. For instance, Huang *et al*. use a best-fit algorithm [8].

The standard wrappers allow a single bandwidth (a fixed set of TAM wires at each wrapper). To increase the flexibility in the scheduling process, Koranne proposed a reconfigurable wrapper that allows all possible bandwidths at each wrapped core [10]. The cost of extra TAM routing and logic is minimized by selecting cores with flexible wrapper prior to applying the scheduling algorithm.

1. The research is supported by the Japan Society for the Promotion of Science (P01735) and the Swedish National Program on Socware.

3 Preliminaries

The amount of logic required to specify a core is not strictly defined. We assume that all parts to be tested, core logic, UDL and interconnections, are cores. Cores with an interface to the TAM are *wrapped* (core c_1, c_3 and c_5 Figure 1), otherwise *unwrapped* (core c_2 and c_4 in Figure 1). A wrapper has the modes; normal operation, internal (core) test or external (interconnection) test.

The execution of a *core test* (always at a wrapped core) and an *interconnection test* (always at an unwrapped core) differs. For a core test the wrapper is put in *internal mode* and a set of TAM wires are used for the transportation of test vectors to the core and test responses from the core. For instance, in testing the wrapped core c_1 (Figure 1) the wrapper is placed in internal test mode and test data is directly transported to and from the core using the TAM.

For an interconnection test, on the other hand, there is no direct connection to the TAM. For instance, when testing the unwrapped core c_2 (Figure 1) the test vectors are transported from the TAM through the wrapper at core c_1 and the test response from c_2 is transported to the TAM via the wrapper at core c_3. For the testing of c_2, the wrappers at core c_1 and c_3 have to be in *external mode.* A wrapper can only be in one mode at a time and therefore can the testing of c_1 and c_3 not be performed at the same time as the testing of c_2; there is a test conflict.

The test time of a test can often be modified. In scan testing, assigning a high number of TAM wires means that the scan chains can be partitioned into a higher number of *wrapper chains* and parallel loading reduces the test time [8,5]. These modifications increases the flexibility in the scheduling process and to further increase flexibility, preemption can be used. In preemptive scheduling a job can be interrupted and resumed at a later time, which is to be compared to non-preemptive scheduling where each job when started runs until completion. For scheduling where a set of test vectors are to be applied, each individual test vector in the test set is independent of the other vectors; we can partition the test set into several sub test sets and apply them one at a time as long as all test vectors are applied.

A system under test, as in Figure 1, can be modelled as:

$C = \{c_1, c_2, ..., c_n\}$ is a finite set of n cores where each core $c_i \in C$ is characterized by:

τ_i: total test time (assuming a single wrapper chain),
tv_i: number of test vectors,
s_i: the core sending the test vectors,
r_i: the core receiving the test response,
sc_{ij}: flip-flops in chain j at core i.

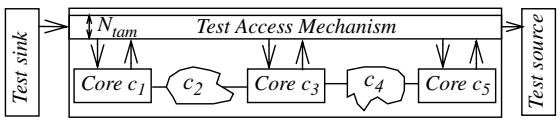


Figure 1. Example system.

For the system:

$N_{tam}=\{w_1,w_2,...,w_m\}$: is a finite set of m wires.

If we have a core test, it means $c_i=s_i=r_i$. For instance, in Figure 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 s_1 (c_1) and the receiving core is $r_3(c_3)$.

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

- the number of test vectors tv_{ik} to apply,
- the number of TAM wires n_{ik} to use,
- the test time τ_{ik},
- the start time ($start_{ik}$) and end time (end_{ik}).

with the main objective is to minimize:

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

The added logic comes basically from the reconfigurable wrapper and it depends on the number of configurations (discussed below). It means that minimizing the number of configurations, minimizes the added wrapper logic. The routing of TAM wires depends on the number of wires connected to each core and it is minimized by minimizing the number of TAM wires at each core.

4 Optimal Scheduling

In this section we describe a preemptive scheduling algorithm producing optimal solution in linear time, $O(n)$ for n jobs (tests) [2]. We extend the algorithm to: (1) increase the utilization of the TAM by using reconfigurable wrappers proposed by Koranne [10] for tests with long test times and (2) handle interconnection tests. The use of reconfigurable wrappers and the number of configurations are determined and minimized by our algorithm.

4.1 Optimal TAM Scheduling of Core Tests

All core tests are independent of each other besides a possible resource conflict due to TAM wire sharing. Each TAM wire is also independent of the other; it does not matter which wires that are used at a core. 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,…, n)) each with a testing time τ_i, on the identical machines (TAM wires) w_j (j=1,…N_{tam}).

All test sets are known in advance as well as the TAM bandwidth. The LB (lower bound) can be computed as [2]:

$$LB = max\left\{max(\tau_i), \sum_{i=1}^{n} \tau_i / N_{tam}\right\} \qquad 1$$

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, 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.

An example will illustrate the approach (Figure 2) [2]. The LB is computed to 7 (Equation 1) 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 a test at c_1 which is assigned to wire w_1 at time 0. At time 4, 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 0 on wire w_2. Executing the test at c_2 starts with the use of wire w_2 during time period 0 to 2 followed by the use of wire w_1 during time period 4 to 7. 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 to set up a job and also to save its state. 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 but 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.

Finally, in some cases, such as for some types of memories such as DRAMs, the testing cannot be preempted. For instance, assume that test t_2 cannot be preempted as in Figure 2. In such a case, when LB is met, the scheduling algorithm restarts at LB (and not at time 0) and moves towards zero. The resulting schedule is in Figure 3. Note that, test t_2 now makes use of one wire during time point 4 to 5 and two wires during time 5 to 7, which is possible using the reconfigurable wrapper. This overlapping is further discussed below.

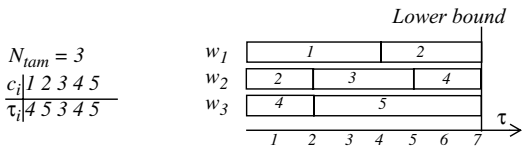


Figure 2. Optimal preemptive TAM assignment.

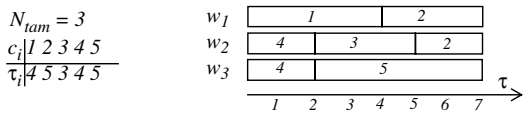


Figure 3. Optimal TAM assignment and preemptive scheduling where test t_2 cannot be interrupted.

4.2 Transformations for Optimal TAM Utilization

A long test time for one of the test in the system may limit the solution, *i.e.* LB is given by the test time of a test ($max(\tau_i)$ in Equation 1). In such a case, the test time can be reduced by assigning more TAM wires to the core that limits the solution so that the length of the wrapper chains becomes shorter, which reduces the core's test time. Our approach is straight forward, we remove the $max(\tau_i)$ part from in Equation 1:

$$LB = max \sum_{i=1}^{n} \tau_i / N_{tam}. \qquad 2$$

When LB is computed, we use the scheduling approach illustrated above (Figure 2). To illustrate it, we use the same example (Figure 2) but with a wider TAM (N_{tam}=7). The scheduling result is presented in Figure 4. A test may overlap in using the wires (machines). For instance, the test at c_1 uses wire w_1 and w_2 during time period 0 to 1 and wire w_1 during period 1 to 3.

We need a mechanism to handle the different TAM bandwidths at each core. Several core test wrappers such as Boundary scan, TestShell and P1500 have been proposed. Recently Koranne proposed a reconfigurable wrapper that allows several bandwidths at each core [10]. We will use a core with 3 scan chains of length {10,5,4} to illustrate Koranne's approach. The scan-chains and their partitioning into wrapper chains are in Table 1. It means when a single wrapper chain is assumed, a single TAM wire is needed and all scan-chains are forming a single wrapper chain. The reconfigurable wrapper allows different TAM bandwidth assignment over time.

For each TAM bandwidths (number of wrapper chains) (1,2, and 3) a directed (di)-graph is generated where each node is a scan-chain and node I is an input TAM [10]. An edge is added between nodes (scan-chains) in the same partition and the shaded nodes are to be connected to the

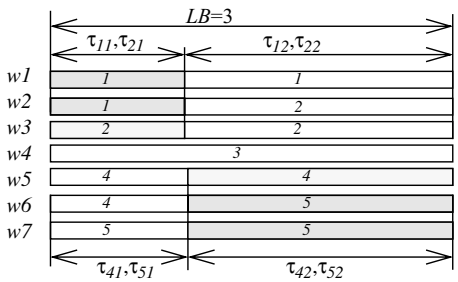
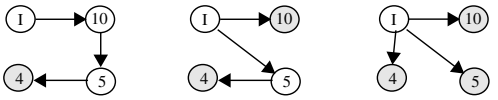


Figure 4. Partitioning of the schedule in Figure 4.

Tam width	Wrapper chain partitions	Max length
1	[10,5,4]	19
2	[(10),(5,4)]	10
3	[(10),(5),(4)]	10

Table 1. Scan chain partitions at the example core.



(a) 1 wrapper-chain (b) 2 wrapper-chains (c) 3 wrapper-chains

Figure 5. Di-graph representations [10].

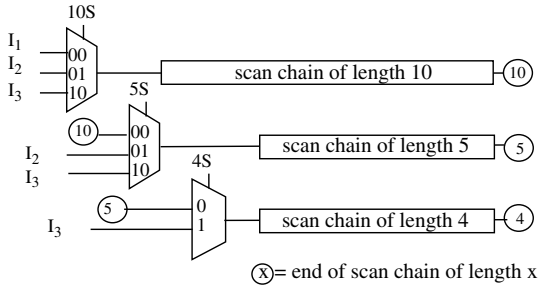


Figure 6. Multiplexing strategy [10].

output TAM. Based on the di-graph (Figure 5) a combined di-graph is generated by taking the union of all di-graphs. The indegree for each node in the combined di-graph gives the number of signals to multiplex, which is outlined in Figure 6 where I_i indicate the input at bandwidth i. We observe that in the case of i=1, a single wrapper chain is assumed and one TAM wire is needed. In cases when I_n and I_m in Figure 6 are mapped to the same TAM wire, the multiplexing logic is reduced as well as the control logic. The control logic is generated from the control signals given in Table 2.

The multiplexing logic and the control logic are also reduced when the number of configurations is reduced. For instance, if only one configuration exists, let say one wrapper chain, no multiplexing logic is required at all.

Returning to the example (Figure 4) where it is clear that a test uses different bandwidth as the test is executed. Taking the test at c_2, which uses only wire w_3 during time period 0 to 1 but w_2 and w_3 during time period 1 to 3. We solve the overlapping problem in two consecutive steps; partitioning of the tests and usage of reconfigurable

TAM width	10S	5S	4S
1	00	00	0
2	01	01	0
3	10	10	1

Table 2. Select signals for multiplexers in Figure 6.

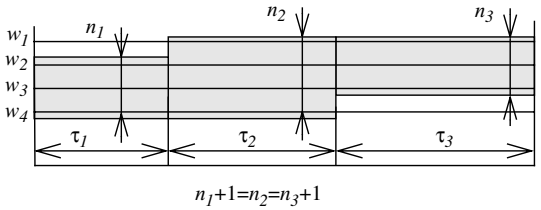


Figure 7. Bandwidth requirement for a general test.

wrappers. After assigning TAM wires to all tests, we determine the partitions, which is illustrated in Figure 4. For instance, in partition 1 of the test at c_2, w_3 is used during τ_{21} and in partition 2 of the test at c_2, w_2 and w_3 are used during τ_{22}. It means that during τ_{21} a single wrapper chain is used and during τ_{22} two wrapper chains are used.

In our approach, the maximal number of partitions per test is three. In the example (Figure 4), no test is partitioned into more than two partitions. However, if the test at c_2 would terminate after time 1 but before 3 (LB) another partition would be created. In Figure 7 we show the general TAM bandwidth requirement for a test. The TAM bandwidth is equal for two of the sessions since n_1=n_3 (Figure 7). It means that only two configurations are needed at each core and a multiplexer for selection between w_1 and w_4 in the example in Figure 7. The added logic due to reconfigurable wrappers depends on the number of cores and the number of configurations. It means that in our approach, the added logic is in the worst case in the range 3×|C| (maximum 3 configurations per core). In the approach by Koranne the added logic, if a reconfigurable wrapper is added at all cores, given by N_{tam}×|C|.

The TAM routing is minimized by minimizing the number of TAM wires routed to each core regardless of the floor-plan. If a floor-plan is known, the tests can be soreted based on placement starting, for instance, in the upper-left corner and ending in lower-right corner. We take the system in Figure 2 with N_{tam}=7 resulting in a test schedule as in Figure 4 where the cores are sorted (and numbered) clock-wise as in Figure 8. The advantage is that neighbouring cores share TAM wires. For instance core 2, which makes use of TAM wire w_2 as soon as core 1 finish its use of w_2. Cores placed far away from each other are not sharing TAM wires, such as core 5 and core 3.

4.3 Interconnection Test

There are no test conflicts among core tests since each core has its dedicated interface to the TAM (Section 4.1 and 4.2). However, the interconnection tests must also be scheduled, which means that the test conflicts must be handled. The test conflicts in a system such as the example system (Figure 1) can be modelled using a resource graph (Figure 9) [9]. An arc between a test and a resource (core or a wrapper cell) indicates that the test requires the resource during testing. The test conflicts comes from that the wrapper cells only

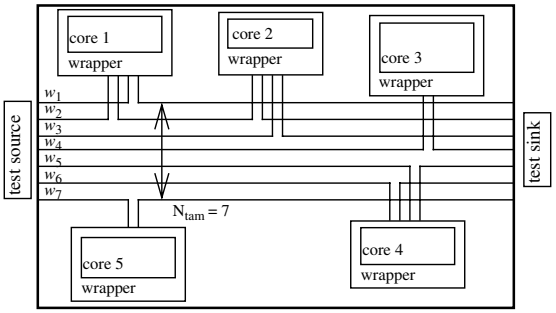


Figure 8. The example system assuming the five wrapped cores to be floor-planned.

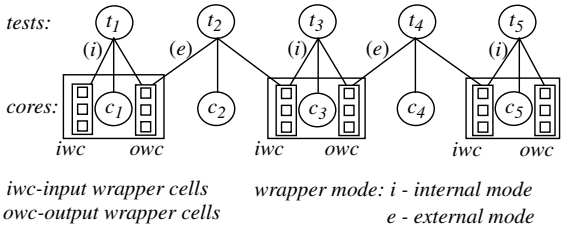


Figure 9. A resource graph of the system in Figure 1

can be in one mode at a time; in *core testing* the wrapper cells are in internal test mode while in *interconnection test* they are in external test mode. In Figure 9, 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.

Figure 1 shows an interconnection test at c_2, which is performed by setting the wrappers at c_1 and c_3 in external test mode and then test vectors are transported to c_2 through the wrapper at c_1 and the test response from c_2 is transported to the TAM using the wrapper at c_3. This demonstrates an interconnect test with a *one-to-one* mapping where the wrapper cells at the functional outputs at c_1 are connected via c_2 to the functional input wrapper cells at c_3. Several other mapping combinations are possible for the wrapper input and wrapper output cells; *one-to-many*, *many-to-one* and *many-to-many*. These mappings cover all combinations and we assume that each functional input and each functional output can be in only one such mapping and in only one test set. In Figure 10 for instance, it means that a functional output wrapper cell let say at c_1 cannot be in one test set with an input wrapper cell at c_5 and in another test set with an input cell at c_3. However, a wrapper cell at c_1 can be in the same test set as a wrapper cell at c_3 and at c_5. In some cases, the functional inputs and outputs at a wrapped core may be connected to different cores. Figure 10 shows such an example where the outputs at c_1 are partitioned into two sets, one set used by c_2 and c_3 and another set used by c_4 and c_5. However, these partitions

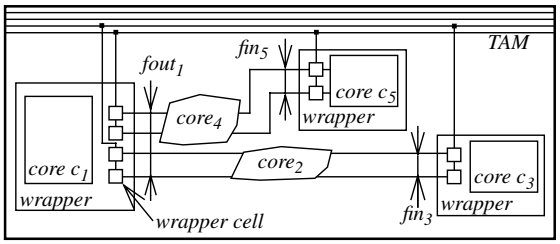


Figure 10. Example of interconnection test.

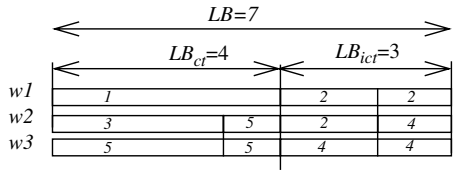


Figure 11. Partitioning of the schedule in Figure 4.

operates independently when the wrapper is in external test mode; no conflict.

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_{ict}. To illustrate, we take the example in Figure 4 assuming that the test at c_2 and c_4 are interconnection tests, which means that executing the test at c_2 inhibit concurrent testing at c_1 and at c_3 and t_4 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 Figure 11.

The test scheduling algorithm consists of four steps:

1. Compute LB_{ct} (lower bound) for the core tests,
2. Schedule all core tests,
3. Compute LB_{ict} for the interconnection tests, and
4. Schedule all interconnection tests.

The algorithm starts at time (τ) zero and at wire (w) zero by selecting a core scheduled to start at time zero. If its test time is higher than LB, a new wire is used. The test time is reduced until it reaches zero and each time LB is reached, a new wire is added to the test. The start time and the end time of the tests are used when creating the partitions. We observe that the LB defines the test application time and also that all TAM wires are fully utilized, all tests ends at the same time (Figure 4). It means that partitioning the tests into two partitions (core tests and interconnection tests) will still produce an optimal solution.

5 Experimental Results

We have above shown that the test scheduling problem for core-based systems can be solved in linear time using preemptive scheduling and reconfigurable test wrappers. We have, nevertheless, implemented our approach allowing preemption and reconfigurable wrappers and using the P93791 design we have made a comparison with previous approaches. In our approach, each test set can be partitioned in maximally three partitions (Section 4.2) and for each partition we have used the wrapper chain algorithm presented by Iyengar *et al.* [3]. The results are collected in Table 3 and for each bandwidth the best solution is marked in bold (solutions better than lower bound computed by Goel and Marinissen are not considered).

6 Conclusions

We have shown that the scheduling of tests on the test access mechanism 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. The main advantages of our approach, besides that optimal test time is found in linear time, are that we minimize the TAM bandwidth at each core, which implicitly minimize the routing of TAM wires, and for the cores with reconfigurable wrapper the number of configurations is minimized, which minimizes the added test logic. We have implemented the algorithm for a comparison with previous approaches.

References

[1] A.L. Crunch, "Design for Test", *Prentice Hall,* 1999.

[2] P. Brucker, "Scheduling Algorithms", *Springer-Verlag*, ISBN 3-540-64105-X, 1998.

[3] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "Co-Optimization of Test Wrapper and Test Access Architecture for Embedded Cores", *Journal of Electronic Testing; Theory and Applications (JETTA),* pp 213-230, April 2002.

[4] V. Iyengar K. Chakrabarty, and E. J. Marinissen, "Efficient Wrapper/TAM Co-Optimization for Large SOCs", *Proceedings of Design and Test in Europe (DATE),* pp. 491-498, Paris, France, March 2002.

[5] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "On Using Rectangle Packing for SOC Wrapper/TAM Co-Optimization", *Proceedings of VLSI Test Symposium (VTS)*, pp. 253-258, Monterey, California, April 2002.

[6] S. K. Goel and E. J. Marinissen, "Cluster-Based Test Architecture Design for System-On-Chip, *Proceedings of VLSI Test Symposium (VTS)*, pp. 259-264, Monterey, California, April 2002.

[7] S. K. Goel and E. J. Marinissen, "Effective and Efficient Test Architecture Design for SOCs", *Proceedings of International Test Conference (ITC),* pp 529-538, Oct. 2002.

[8] Y. Huang *et al.*, "Resource Allocation and Test Scheduling for Concurrent Test of Core-based SOC Design", *Proceedings of Asian Test Symposium (ATS)*, pp 265-270, Kyoto, Japan, November 2001.

[9] C.R. Kime and K.K. Saluja, "Test Scheduling in Testable VLSI Circuits", *Proceedings of International Symposium on Fault-Tolerant Computing,* pp. 406-412, June 1982.

[10] S. Koranne, "A Novel Reconfigurable Wrapper for Testing Embedded Core-Based and its Associated Scheduling", *Journal of Electronic Testing; Theory and Applications (JETTA)*, pp. 415-434, August 2002.

[11] S. Koranne and V. Iyengar, "On the Use of k-tuples for SoC Test Schedule Representation", *Proceedings of International Test Conference (ITC),* pp. 539-548, October 2002.

Approach	Test application time: *T*						
	TAM=16	**TAM=24**	**TAM=32**	**TAM=40**	**TAM=48**	**TAM=56**	**TAM=64**
Lower bound [7]	1746657	1164442	873334	698670	582227	499053	436673
Enumerate [3]	1883150	1288380	944881	929848	835526	537891	551111
ILP [3]	1771720	1187990	887751	(698583)	599373	514688	460328
Par eval [4]	1786200	1209420	894342	741965	599373	514688	473997
GRP[5]	1932331	131084	988039	794027	669196	568436	517958
Cluster [6]	-	-	947111	816972	677707	542445	467680
Binpack[8]	1791860	1200157	900798	719880	607955	521168	459233
CPLEX[10]	1818466	(1164023)	919354	707812	645540	517707	453868
ECTSP[10]	1755886	(1164023)	919354	707812	**585771**	517707	453868
ECTSP1[10]	1807200	1228766	967274	890768	631115	562376	498763
TB-serial [7]	1791638	1185434	912233	718005	601450	528925	455738
TR-serial [7]	1853402	1240305	940745	786608	628977	530059	461128
TR-parallel [7]	1975485	1264236	962856	800513	646610	540693	477648
K-tuple [11]	2404341	1598829	1179795	1060369	717602	625506	491496
Our approach	**1752336**	**1174252**	**877977**	**703219**	592214	**511925**	**442478**

Table 3. Test time comparison on P93791.