Optimal Test Time for System-on-Chip Designs using Preemptive Scheduling and Reconfigurable Wrappers

Erik Larsson and Hideo Fujiwara

Graduate School of Information Science

Nara Institute of Science and Technology,

8916-5 Takayama, Ikoma, Nara 630-0101, Japan

{erila,fujiwara}@is.aist-nara.ac.jp

Abstract[1]

In this paper, we show that the test access mechanism (TAM) scheduling is equal to the independent job scheduling on independent machines. We make use of an existing preemptive scheduling algorithm, which produces an optimal solution in O(n) time. We extend the algorithm to handle test conflicts due to interconnection test and cases when a test limits an optimal usage of the TAM by using reconfigurable test wrappers, which in contrast to standard approaches allows several TAM bandwidths per core. Our extension produces optimal solution in respect to test time in O(n) time and it also minimizes the TAM usage, its routing as well as the number of wrapper configurations.

1 Introduction

The development of an efficient test solution for digital systems is becoming a complicated task due to the increasing design complexity, which also leads to a high test data volumes and long testing times. A way to tackle the complexity problem is to use a core-based design environment where a system is composed of a set of cores, defined test set and a test access mechanism (TAM) used to transport test data into the system and out of the system. An important mechanism is the interface, the core test wrapper, between the core and the TAM added to ease test access.

A *wrapped* core is a core placed in a wrapper while a core without 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, or
- *interconnection test*, tests unwrapped cores, interconnections between cores, and user-defined logic (UDL).

Several test scheduling techniques minimizing the total test application time have been proposed [2,3,4,5]. Recently, TAM scheduling of core tests, a special case of test scheduling, has gained interest [6,7,11]. However, interconnections, unwrapped cores and UDL must also be tested, which means test conflicts must be considered. Furthermore, the techniques minimize test time but do not consider costs related to added logic or TAM routing.

In this paper, we address the TAM scheduling problem and the main contributions of our work are that we:

- demonstrate that the TAM scheduling problem is equal to the independent job scheduling on independent machines,
- make use of a preemptive scheduling algorithm producing optimal solution in $O(n)$ time for n jobs (tests) [8],
- propose an extension to the algorithm to adjust the test time of the tests to fully utilize the TAM for wide TAMs bandwidths by using the reconfigurable test wrapper proposed by Koranne [10], which in contrast to previous approaches allows several TAM bandwidths at each core,
- extend the optimal preemptive TAM scheduling algorithm to handle test conflicts, which must be considered to test interconnections and UDL.

The advantage of our approach, besides that we achieve optimal test time in $O(n)$ time, is that we implicitly minimize the TAM usage and its routing, and we only make use of a minimum of reconfigurable configurations when using the reconfigurable wrapper, in other words; the overhead is minimized.

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 scheduling. The scheduling approach is presented in Section 4 and conclusions are in Section 5.

2 Related Work

2.1 Test Scheduling

The basic problem in test scheduling is to determine a start time for all tests while minimizing the total test time. The scheduling approaches can be grouped into:

- *non partitioned testing*, where all tests are scheduled in sessions and no new tests are started until all tests in the session are fully executed (Figure 1(a)). Chou *et al.*[3] and we [5] have proposed such techniques,
- *partitioned testing with run to completion*, where tests are allowed to start as soon as possible, *i.e.* no sessions

1. This work has been supported by the Japan Society of Promotion of Science (JSPS) under grant P01735.

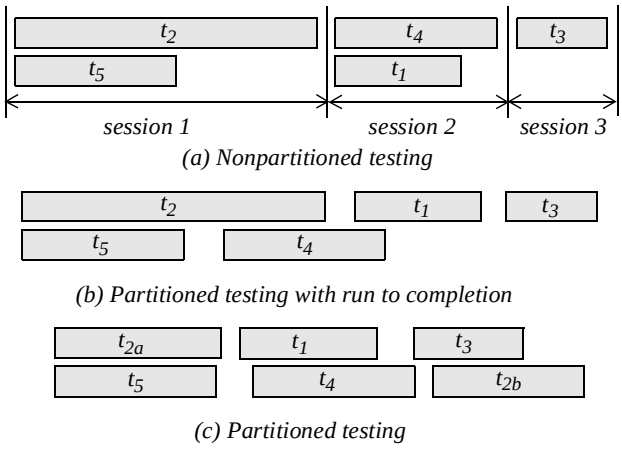


Figure 1. Scheduling approaches.

exists (Figure 1(b)). The advantage is that it increases the flexibility in the scheduling process. Chakrabarty [2] and also we [5] have proposed such techniques,

- *partitioned (preemptive) testing*, where each test set can be partitioned into several sub sets applied at different times (Figure 1(c)). The flexibility while scheduling the tests is increased further. Iyengar and Chakrabarty [4] and also we [11] have proposed such techniques.

Huang *et al*. have proposed a TAM scheduling approach minimizing the test time for wrapped cores, i.e. no test conflicts exists. In the approach, transformations for each test can be made and it is assumed that each test defines an area given by its test time and its TAM usages. Increasing the TAM width for a test, reduces the test time and the tests are scheduled using a best-fit algorithm [6].

Another TAM scheduling approach for core tests is proposed by Iyengar *et al*., where the test time variations for cores over assigned TAM bandwidth is analyzed. The analysis indicates that the linear dependency does not always exist between test time and TAM bandwidth assigned to a core. Iyengar *et al.* uses Pareto-optimum to guide the proposed rectangular packing algorithm [7].

2.2 Test Wrapper

The core test wrapper, needed to ease test access, is the interface between a core and the TAM. If such interface exists, the core is said to be *wrapped*, otherwise *unwrapped.* Several wrappers such as Boundary scan, TestShell and P1500 have been proposed and they can basically be in three different modes; normal operation, internal (core) test or external (interconnection) test [12].

Recently Koranne proposed a reconfigurable wrapper with the feature to allow several bandwidths [10]. The main advantage of the approach is that it increases the flexibility in the scheduling process. 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

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.

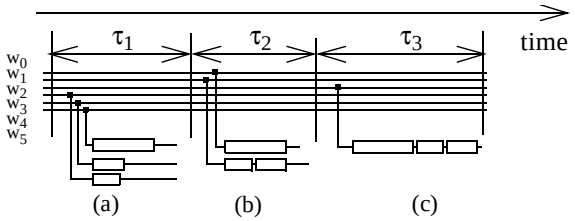
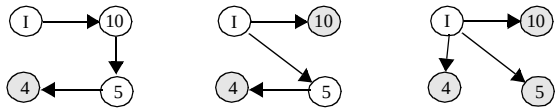


Figure 2. TAM assignment at the example core at time τ_1 - 3 wrapper chains, at time τ_2 - 2 wrapper chains, and at time τ_3 - 1 wrapper chain.



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

Figure 3. Di-graph representations.

are forming a single scan chain. The reconfigurable wrapper allows different TAM bandwidth assignment over time. For instance if we only consider this example core and its TAM wire assignment over time (Figure 2) we see that during time τ_1 the 3 TAM wires w_3, w_4 and w_5 are assigned to the core and three wrapper chains are assumed. Later, during time τ_2, the two TAM wires w_0 and w_1 are assigned and two wrapper chains are used and finally during τ_3 w_2 is assigned where all chains form a single scan-chain.

For each TAM bandwidths (number of wrapper chains) (1,2, and 3) a 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 output TAM. Based on the di-graph (Figure 3) 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 4 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 1 TAM wire is needed. In Figure 2, the example of the TAM wire assignment is shown for the 3 configurations. In cases when I_n and I_m in Figure 4 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 are reduced. For instance, if only 1 configuration exists, let say 1 wrapper chain, no multiplexing logic is required at all.

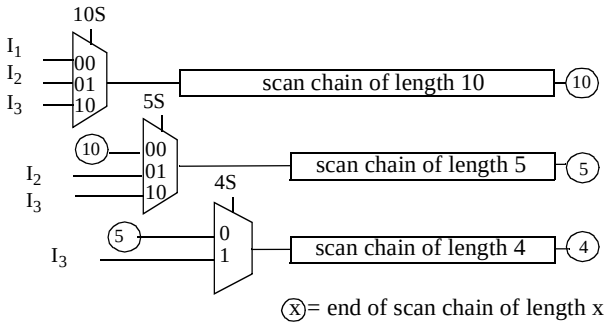


Figure 4. Multiplexing strategy [10].

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 4.

3 Preliminaries

The amount of logic required at a core in order to call it a core is not strictly defined. We assume that all parts to be tested, cores, UDL and interconnections, are seen as cores (Figure 5), which can be either *wrapped* or *unwrapped.*

The transportation of test vectors to a wrapped core and transportation of test response from a wrapped core require a set of TAM wires. The unwrapped cores on the other hand require a wrapped core for the feeding of the test vectors and a wrapped core to send the test response to since an unwrapped core has no interface to the TAM. For instance, the unwrapped core c_2 in Figure 5 is tested by an *interconnection test* where the test vectors are transported through the wrapper at core c_1 and the test response from c_2 transported to the test sink using the wrapper at core c_3. It means that the wrappers at core c_1 and c_3 are in *external test mode*, which means the logic at core c_1 and c_3 cannot be tested at the same time as core c_2 is tested. Testing the logic at c_1 and c_3 (*core test*) requires that respective wrapper to be in *internal test mode*. It means that a test conflict occurs since testing of c_1 and c_3 cannot be done at the same time as c_2 is tested.

The test time of a test at a core can often be modified. For instance, scan testing is often used and by assigning a high number of wrapper chains (TAM wires) to a core will reduce the test time since it reduces the time due to the shift process [6,7,11]. 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 test scheduling where a set of test vectors are to be applied, we observe that each individual test vector is independent on the other vectors. It means that we can partition the test set into several sub test sets and apply them

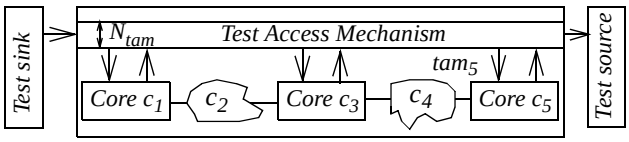


Figure 5. Example system.

one at a time.

A system under test as the one in Figure 5 with support of preemptive scheduling 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,
tv_i: 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.

For the system:

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

If we have a core test, it means $c_i=s_i=r_i$. For instance, in Figure 5 when testing c_1, no other core is required ($c_1=s_1=r_1$). On the other hand, when testing core c_2, core c_1 and core c_3 are required, i.e. the sending core is s_1 and the receiving core is r_3.

For each core c_i at each partition k we have to determine:

- the number of test vectors tv_{ik},
- the TAM wires n_{ik},
- the test time τ_{ik},
- the start time ($start_i$) and end time (end_i).

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 above). It means that minimizing the number of configurations, minimizes the 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 test scheduling technique for core tests, which achieves optimal test time. We extend the technique to increase the utilization of the TAM by using test transformations for tests with long test times and we outline a technique to support interconnection tests.

4.1 Optimal TAM Scheduling

The tests at wrapped cores in a core-based system are independent of each other and each of the TAM wires is also independent of each other. It means that the independent job scheduling on independent machines is equal to the TAM scheduling by letting each TAM wire be an independent

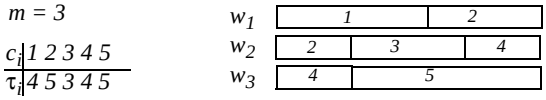


Figure 6. Optimal preemptive TAM assignment.

machine and each test be a job. We then have to schedule the independent jobs (the tests at core c_i, (i=1,…, n)) each with a testing time τ_i, on the independent machines (TAM wires) w_j (j=1,…m).

All test sets are known in advance as well as the TAM bandwidth. It means that the LB (lower bound) is given [8]:

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

The independent job scheduling on independent machines problem can be solved in $O(n)$ time by using preemption [8]: 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 [8] (Figure 6). 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 scheduled at time 0 on wire w_1. At time 4, when the test at c_1 is finished, the test at c_2 is scheduled to start and at time 7 when LB is reached, it is preempted and the rest of the test is scheduled to start on wire w_2 at time 0. The test at c_2 is rearranged when executed where the first part of the test is using wire w_2 during time period 0 to 2 and the second part of the test is applied at time point 4, where wire w_1 is used. At preemption of a test, a new 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 due to set up a job and also to save its state. The main reson 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 continous. The advantage is that the state of the core is already set and testing of it can start at once. Figure 7 illustrates the clocking at a core where *normal*/*test* selection determines the mode of the core. The test controller determines when the cores are to be active during testing by assigning values to the *run*/*stop selection.* Assume that core c_2 has a wrapper-chain of length l (l cycles are needed to perform a shift in of a new vector and a shift out of the previous test response). If the test is preempted when x% of the l cycles are applied it means that when the test is restared 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.

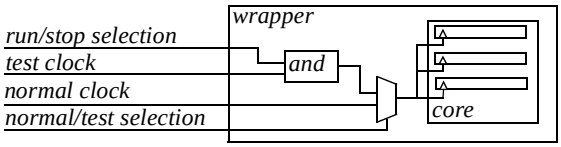


Figure 7. Clock-gating at preemption.

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, a wider TAM can be assigned to these tests reducing their test time, which leads to a more efficient utilization of the TAM. Our approach is straight forward, we remove the $max(\tau_i)$ part from in Equation 1 and get:

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

(N_{tam}=m) and after computation of LB, we use the scheduling approach illustrated above (Figure 6).

To illustrate, we use the same example as above but assuming a wider TAM (N_{tam}=m=7). The scheduling result is presented in Figure 8 and in this case, a test may overlap in using the wires (machines). For instance, the test at c_2 uses wire w_2 and w_3 during the time period 1 to 3.

In the example, it is also 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. The overlapping problem is solved in two consecutive steps; partition of the tests and use of reconfigurable wrappers. After assigning the TAM wires to all tests in the system, we determine the partitions, which is illustrated in Figure 8. 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}.

The maximal number of partitions per test is 3. In the example in Figure 8, no test is partitioned into more than 2 partitions. However, if the test at c_2 would terminate after time 1 but before 3 (LB) another partition would be created. In Figure 9 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 9). It means that only two configurations are needed at each core.

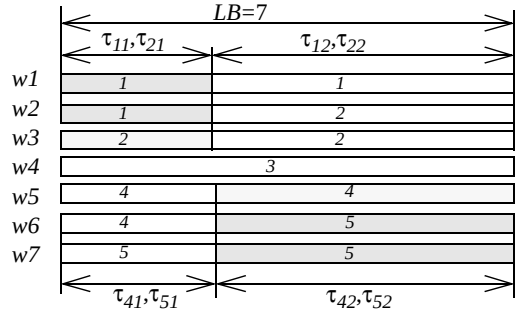
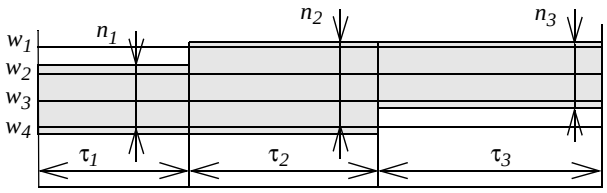


Figure 8. Partitioning of the schedule in Figure 8.



$n_1+1=n_2=n_3+1$

Figure 9. Bandwidth requirement for a general test.

To handle several bandwidths we make use of a reconfigurable core wrapper allowing flexible TAM bandwidth (described above) [10]. For each core, we can have no more than two different configurations. The advantage is that it minimizes the overhead since the overhead depends on the number of configurations. Koranne has addressed the overhead using the reconfigurable wrapper by limiting the number of cores with reconfigurable wrapper [10]. However, such approach requires that the cores with reconfigurable wrapper have to be determined in advance. Furthermore, the approach assumes that once a core is placed in a reconfigurable wrapper, all configurations are possible, which might not be needed and also all TAM wires are routed to these cores.

4.3 Interconnection Test

In the approaches above (Section 4.1 and 4.2) no test conflicts existed and the motivation for that was that all cores were efficiently isolated and test will not interfere, i.e. core tests where considered (testing of wrapped cores). However, in general also interconnections and unwrapped UDL must be tested.

The tests and the test conflicts in a system such as the example system (Figure 5) can be modelled using a resource graph (Figure 10). An arc between a test and a core (resource) indicates that the test requires that core during testing. The tests, above classified as *core test* and *interconnection test* where the wrapper is put in internal test mode for the former and in external test mode for the latter.

In Figure 10, all core tests at wrapped cores where the wrapper has to be in internal mode are marked (i) and interconnection tests requiring needed wrappers to in external test mode are marked (e). Making two resource graphs, one for core tests and one for interconnection tests makes it possible to apply all core tests.

A typical scenario for an interconnection test is as follows, test vectors are transported on the TAM to a wrapped core placed in external test mode, which is feeding the test vectors to the unwrapped core (the target for the interconnection test). The test responses are captured at a wrapped core also in external test mode and then fed to the TAM. In Figure 5 such a test at c_2 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

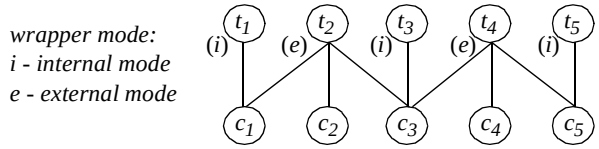


Figure 10. A resource graph of the system in Figure 5

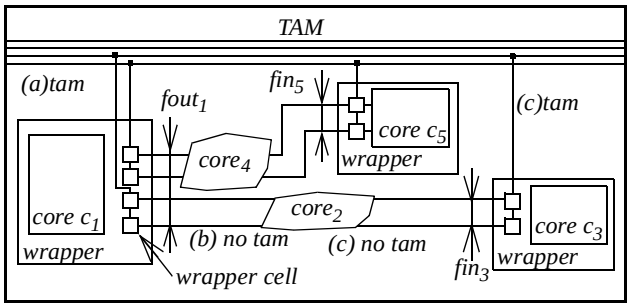


Figure 11. Example of external test.

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 mapings cover all combinations and we assume that each functional input and each functional output can be in one such mapping and in one test set. 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, the wrapper cell at c_1 can be in the same test set as a wrapper cell at c_3 and at c_5. The grouping actually means that each interconnection test set is independent of all other interconnection tests.

In some cases, the functional inputs and outputs at a wrapped core may be connected to different cores. Figure 11 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. In these cases, the partitions can operate independently when the wrapper is in external test mode.

Separating *core test* and *interconnection test* eliminates the test conflict. In our approach, we divide the testing into two consecutive parts, core test followed by interconnection test. 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 8). It means that partitioning the tests into two partitions (core tests and interconnection tests) will still produce an optimal solution.

We partition the TAM schedule into a core test part given by LB_{ct} and an interconnection test part given by LB_{ict}. LB_{ct}. To illustrate, we take the example in Figure 8 and we assume 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

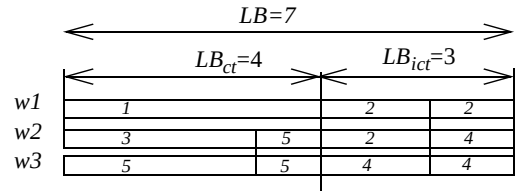


Figure 12. Partitioning of the schedule in Figure 8.

```
w=0; // wire
τ=0; // time
select any core c_i where s_i=r_i=c_i // core test
    start_i= τ; // the start time of the test
    until τ_i=0 { // test time of test at core c_i
        let t_i start at time t using wire w;
        if τ+ τ_i> LB {
            τ_i= τ_i-(LB- τ); // reduce the unscheduled time of t_i
            w=w+1; // increase to next wire
            τ=0;
        } else {
            τ= τ+ τ_i; // t_i ends at τ
            end_i= τ;
        }
    }
    if start>end then {
        τ_i1=end_i; τ_i2=start_i;
    } else {
        τ_i1=start_i; τ_i2=end_i;
    }
}
```

Figure 13. Test scheduling algorithm.

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. LB is computed to 7 ((4+5+3+5+4)/7) and LB_{ct}=(4+3+5)/3=4 and LB_{ict}=(5+4)/3=3, i.e. $LB=LB_{ct}+LB_{ict}$. The test schedule is presented in Figure 12.

The test scheduling algorithm consists of four steps:

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

The LB is computed (Equation 2) and the scheduling algorithm for core tests is outlined in Figure 13. The algorithm for interconnection tests is similar except that in the core selection step:

select any core c_i *where* $s_i=r_i=c_i$

is changed to:

select any core c_i and the time is initially set to: $\tau=LB_{ct}$

The algorithm starts at time (τ) zero and at wire (w) zero. A core is selected and its test is scheduled to start at time zero. If the test time is higher than LB, a new wire is used. The test time is reduced until it reach zero and each time LB is reached, a new wire is used. We keep track on the start time and the end time of the test, which is useful when we are to create the partitions.

5 Conclusions

In this paper we have related the TAM scheduling problem to the independent job scheduling on independent machines problem, for which it is known that an optimal solution can be found in $O(n)$ time using preemptive scheduling. We have extended the algorithm for cases where a specific test limits the solution by making use of a reconfigurable core wrapper. We have also extended the algorithm to include interconnection test. The main advantages of the approach, besides that optimal test time is found in $O(n)$ time, are that it also minimizes the TAM bandwidth at each core, which leads to a minimal routing of TAM wires, and for the cores with reconfigurable wrapper and the number of configurations is minimized by the algorithm.

References

[1] E. J. Marinissen, R. Arendsen, G. Bos, H. Dingemanse, M. Lousberg, and C. Wouters, "A Structured And Scalable Mechanism for Test Access to Embedded Reusable Cores", *Proceedings of IEEE International Test Conference (ITC)*, pp. 284-293, Washington, DC, October 1998.

[2] K. Chakrabarty, "Test Scheduling for Core-Based Systems Using Mixed-Integer Linear Programming", *IEEE Transactions on CAD of IC and Systems*, Vol. 19, No. 10, pp. 1163-1174, October 2000.

[3] R. Chou, K. Saluja, and V. Agrawal, "Scheduling Tests for VLSI Systems Under Power Constraints", *IEEE Trans. on VLSI Systems*, Vol. 5, No. 2, pp. 175-185, June 1997.

[4] V. Iyengar and K. Chakrabarty, "Precedence-based, preemptive, and power-constrained test scheduling for system-on-a-chip", *Proc. of IEEE VLSI Test Symposium (VTS)*, pp. 42-47, Marina Del Rey, CA, April 2001.

[5] E. Larsson and Z. Peng, "An Integrated System-On-Chip Test Framework", *Proceedings of Design, Automation and Test in Europe Conference (DATE)*, pp. 138-144, Munchen, Germany, March 2001.

[6] Y. Huang, W-T Cheng, C-C Tsai, N. Mukherjee, O. Samman, Y. Zaidan, and S. Reddy, "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.

[7] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "On Using Rectangle Packing for SOC Wrapper/TAM Co-Optimization", *Proc. of VLSI Test Symposium (VTS)*, 2002.

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

[9] H. Bleeker, P. Van Den Eijnden, and F. De Jong, "Boundary-Scan Test: A Practical Approach", *Kluwer Academic Publishers*, ISBN 0-7923-9296-5, 1993.

[10] S. Koranne, "Design of Reconfigurable Access Wrappers for Embedded Core Based SOC Test", *Proceedings of IEEE International Symposium on Quality Electronic Design (ISQED)*, pp 106-111, San Jose, California, March 2002.

[11] E. Larsson and H. Fujiwara, "Power Constrained Preemptive TAM Scheduling", *Informal Digest of IEEE European Test Workshop (ETW)*, pp 411-416, Corfu, Greece, May 2002.

[12] S. Mourad and Y. Zorian, "Principles of Testing Electronic Systems", *John Wiley & Sons, Inc*, ISBN 0-471-31931-7, 2000.