IEICE TRANS. INF. & SYST., VOL.E87–D, NO.3 MARCH 2004

PAPER Special Section on Test and Verification of VLSI

Preemptive System-on-Chip Test Scheduling*

Erik LARSSON[†,††], *Nonmember and* **Hideo FUJIWARA**[††], *Fellow*

SUMMARY In this paper, we propose a preemptive test scheduling technique (a test can be interrupted and later resumed) for core-based systems with the objective to minimize the test application time. We make use of reconfigurable core test wrappers in order to increase the flexibility in the scheduling process. The advantage with such a wrapper is that it is not limited to a single TAM (test access mechanism) bandwidth (wrapper chain configuration) at each core. We model the scheduling problem as a Bin-packing problem, and we discuss the transformation: number of TAM wires (wrapper-chains) versus test time in combination with preemption, as well as the possibilities and the limitations to achieve an optimal solution in respect to test application time. We have implemented the proposed preemptive test scheduling algorithm, and we have through experiments demonstrated its efficiency.

key words: *test scheduling, test access mechanism design, preemptive scheduling, system-on-chip testing*

1. Introduction

The increasing complexity of digital systems has lead to the development of the core-based design technique, and the technology evolution has lead to device size miniaturization. The core-based design technique in combination with device size miniaturization make it possible to design a complex system, which is placed on a single chip as a SOC (system-on-chip). The idea behind the core-based design technique is to compose a system by integrating pre-defined and pre-verified modules of logic, so called cores. The advantage with the approach is that systems are designed by making use of reusable cores, and not design the system from scratch, which reduces the design time and makes it possible to design complex systems in a reasonable time.

SOC designs show similarities with the PCB (printed circuit board) designs, however, from a testing perspective, there are differences; one important difference is the amount of test data. In both cases, SOC and PCB, test data (test stimuli and test response) are transported in and out of the system. However, for a PCB system the amount is less, mainly due to that the components are tested prior to mounting. System testing is limited to testing of interconnections. In SOC designs, on the other hand, the complete system is tested in a single phase: cores and interconnections. Furthermore, due to the increasing design complexity, a substantial amount of test data are required to test an SOC.

Several test scheduling techniques have been proposed [1], [3], [4], [10], [18], [19], [23], [24]. The objective with the techniques is to organize the execution of the tests as concurrent as possible in order to reduce the test application time, but without violating constraints and limitations. Recently, the problem of assigning TAM (Test Access Mechanism) wires to cores in SOC designs, a special case of test scheduling, has gained interest [6]–[8], [10], [11], [14], [16], [17]. The approaches assume that each core has a fixed set of scan elements (scan-chains and wrapper cells) and the basic problems are to determine:

- the number of wrapper-chains for each core,
- in which wrapper-chain each scanned element (scan-chain and wrapper cells) should be included,
- the TAM wires to connect the wrapper-chains, and
- the start time for each test,

in such a way that the system's test application time is minimized. The assumption that at least some cores have a few and fixed number of unbalanced scan-chains makes the problem complicated.

A core test wrapper is the interface between a core and the TAM [20], [21]. Wrapper approaches, such as P1500, assume that the scanned elements at each core are configured into a single set of wrapper chains, which are to be connected to the TAM. Recently, Koranne proposed a reconfigurable core wrapper where the scanned elements can be configured into different number of wrapper chains over time [16]. The advantage is that it increases the flexibility in the scheduling process.

In this paper, we propose a preemption-based test scheduling technique (a test can be interrupted and resumed later). We model the problem as a Bin-packing problem and we make use of reconfigurable core wrappers, which is useful to allow flexible number of wrapper-chains at each core. In contrast to previous work, we focus on systems designed with so called soft cores, synthesizable cores, where the number of scanned elements is given, but not the length of each individual scan-chain. It means that we have the possibility to balance the wrapper chains at each core. The motivation is that we believe that future hard cores (cores with fixed number of scan-chains) will be designed with a relatively high number of balanced scan-chains, which makes it easier to balance the wrapper chains. Therefore they will have a similar behaviour as soft cores.

Manuscript received June 28, 2003.
Manuscript revised October 7, 2003.

[†]The author is with the Embedded Systems Laboratory, Linköpings Universitet, SE-581 83 Linköping, Sweden.
[††]The authors are with the Computer Design and Test Laboratory, Nara Institute of Science and Technology, Ikoma-shi, 630–0101 Japan.

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

In this paper we also discuss the possibility to achieve an optimal solution. This discussion is important because we allow preemption and reconfigurable wrappers, and that makes it is possible to partition the test sets into smaller partitions and make use of a different number of TAM wires for each of the partitions.

We have made an analysis of previously proposed test architectures for different TAM bandwidths. The objective is to analyze the behaviour of the architectures at different TAM widths.

We propose a test scheduling technique and the advantage with the approach is that it determines the cores that require flexible wrapper and also the number of wrapper-chain configurations. It should be compared to the approach by Koranne where the cores allowed to have reconfigurable wrappers are determined prior to scheduling.

The rest of the paper is organized as follows. An overview of related work is in Sect. 2, and preliminaries are given in Sect. 3. The system model and the problem formulation are given in Sect. 4. In Sect. 5, we analyse previous proposed techniques, and our approach is described in Sect. 6. Experimental results are presented in Sect. 7, and the paper is concluded in Sect. 8.

2. Related Work

Scheduling the tests in a system means that the start time and the end time are determined for all tests in such a way the test application time is minimized. Several techniques have been proposed and they can be divided into:

- *Non partitioned testing* - new tests are not started until all tests in the session are completed. Zorian [24] and Chou *et al.* [4] have proposed such techniques,
- *Partitioned testing with run to completion* - tests may start as soon as possible. Examples of such techniques are the ones proposed by Chakrabarty [3] and Muresan *et al.* [23],
- *Partitioned* (*preemptive*) *testing* - a test may be interrupted and resumed later. Iyengar and Chakrabarty [10] proposed such a technique.

All proposed scheduling approaches are minimizing the systems test application time but are taking different issues into consideration. Chakrabarty focus on test conflicts imposed by external tests and BIST (Built-In Self-Test) tests [3]. Zorian's technique minimizes the number of control lines for BIST systems [24]. The number of control lines is determined by the number of time points when tests are scheduled to starts. In non partitioned testing all tests in a session start at the same time, which minimizes the number of control lines since all tests in a session can share the same control line. For general systems, Chou *et al.* [4] and Muresan *et al.* [23] have proposed techniques.

The above test scheduling approaches focus on a fixed test time for every test sets. Iyengar and Chakrabary proposed a preemption-based test scheduling technique [10] where each test set can be interrupted and resumed later. All test vectors are applied but they can be partitioned into several sub test sets.

In scan testing each test vector is shifted in (scanned in), and after a capture cycle, the test response is shifted out (scanned out), and at the same time the next test vector is shifted in. The shift process contributes to a major part of the testing time. The shift time at a core depends on the number of wrapper-chains (the number of partitions of the scanned elements, *i.e.* the scan-chains and the wrapper cells). The test time can be reduced by assigning a higher number of wrapper-chains to the core, which will make each wrapper-chain shorter (it includes less scanned elements). For systems composed of hard cores (a fixed number of scan-chains of fixed length), several test scheduling approaches have been proposed [6]–[9], [11]–[14], [16]. Aerts and Marinssen [1] investigated scan-chain partitioning for soft cores (only flip-flops are given) where the constraints are defined by available pins (bandwidth).

Test access is eased by placing the core in a wrapper such as Boundary scan [2], TestShell [20], or IEEE P1500 [21]. These approaches assume one single TAM bandwidth (one configuration of wrapper-chains) per core. Koranne has recently proposed a flexible bandwidth test wrapper where the number of wrapper-chains can vary during the testing [16]. In order to minimize the introduced overhead, only a few cores are allowed to have a flexible wrapper [16]. However, it is a difficult problem to determine which cores should be allowed to have a flexible wrapper and also all bandwidth configurations at the flexible wrapper might not be needed. These problems are not addressed in a systematic way by Koranne.

3. Preliminaries

The cores in a core-based design are given as [2]:

- soft cores, which comes in the form of synthesizable RTL (register-transfer level) descriptions,
- firm cores, supplied as gate-level netlists, or as
- hard cores, available as non modifiable layouts.

For soft cores the number of scanned elements are known, however, the length and the number of scan-chains are not determined. In the case with hard cores, both the length and the number of scan-chains are fixed. It means that soft cores allow higher flexibility when determining the number of scan-chains and their length compared to firm cores and hard cores. However, when creating a hard core flexibility to determine the number of scan-chains and their length can be achieved. Consider an example of a hard core and its scan-chain implementation in Fig. 1. In Fig. 1 (a) a single scan-chain is used, while in (b) a fixed set of n scan-chains is used. In both cases the number of scan chains are fixed, however, in Fig. 1 (b) the chains can externally be configured into a variation of scan chain lengths. Furthermore, in order to design a hard core, which is easier to reuse, a high number of shorter scan-chains of equal length is to be preferred compared to few longer scan-chains of unequal length. The

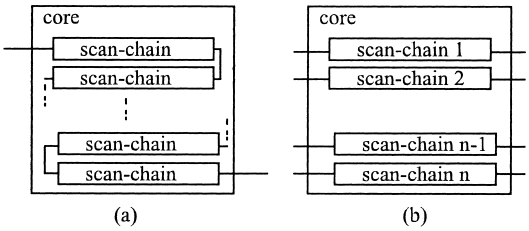


Fig. 1 Scan-chains design at a core, (a) a single fixed scan-chain and (b) n fixed scan-chains.

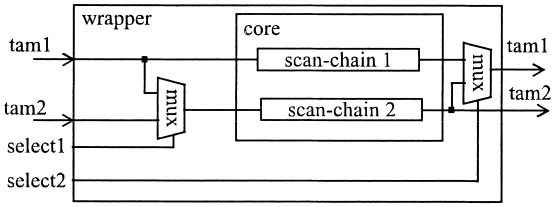


Fig. 2 Flexible scan-chains design at a core test wrapper.

former approach gives a higher possibility to configure the scanned elements into balanced wrapper chains, which are easier to schedule efficient. It also means that from the perspective of length and number of scan-chains, future hard cores are likely to be similar to soft cores. Therefore, we assume soft cores in this work.

In Fig. 2, we illustrate how to achieve flexible scan-chain length for a hard core. Depending on the selectors (select 1 and select 2) the two scan-chains can form either a single wrapper-chain or two wrapper-chains. If a single wrapper-chain is used, the test vectors are loaded through tam1. In the case when two wrapper-chains are used, both chains are loaded at the same time, test data is loaded in scan-chain 1 through tam1 and in scan-chain 2 through tam2. The selectors make it is possible to direct the test vector to either scan-chain1 or scan-chain 2. The multiplexer on the output is used to direct the test response to correct TAM wire when the scan chains are configured into a single chain.

4. System Modelling and Problem Formulation

An example of a system under test is given in Fig. 3. Each core is placed in a wrapper, which serves as the interface between the to the wrapper-chains and the TAM. The system is tested by applying several sets of tests where each set is created at a test generator (source), and the test response is analysed at a test response evaluator (sink). A system under test, such as the one shown in Fig. 3, can be modelled as:

$C = \{c_1, c_2, \ldots, c_n\}$ is a finite set of n cores.

Each core $c_i \in C$ is characterized by:

tv_i: number of test vectors,

ff_i: number of scanned flip-flops.

For the system:

N_{TAM}: bandwidth of the TAM.

The formula to compute the test time at a core is by Aerts and Marinissen [1] defined as:

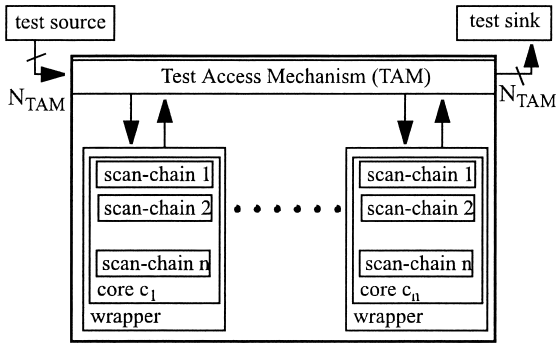


Fig. 3 Embedded cores, wrappers and TAM.

$$t_{test}(c_i) = (tv_i + 1) \times \lceil ff_i / n_i \rceil + tv_i \quad (1)$$

where the core c_i has ff_i scanned flip-flops divided into n_i scan chains and tested with tv_i test vectors. The division of flip-flops into scan-chains has to be rounded upwards since the length of a scan-chain must be an integer.

For each core, a set of test vectors is given and for a given TAM bandwidth, we can compute its test time using Eq. (1). This can be illustrated using a 2-dimensional cube for each test set, test time versus TAM width (number of wrapper-chains). Each test set has such cubes and for every core one cube (test time for a certain number of TAM wires) has to be selected. All selected cubes have to be packed in such a way that the test application time is minimized. This can be modelled as a Bin-packing problem [5].

In preemptive scheduling, the test vectors at each core do not have to be scheduled as a single test set. Each test set can be divided into several sub test sets. Furthermore, the TAM bandwidth for each sub test set can be different if a reconfigurable wrapper is assumed. For instance, if we have a test set of 10 test vectors and we apply 5 test vectors in the first sub set and the other 5 test vectors in a second sub set, we can have one TAM bandwidth for the first set and another bandwidth for the second test set. It means that each of the cubes used to illustrate a test can be split up into several cubes of smaller size.

To support preemption and flexibility in TAM bandwidth, we introduce; for a core c_i with test vectors to be applied in session j:

sc_{ij}: number of test vectors,

tt_{ij}: test time,

tam_{ij}: number of TAM wires required.

An example of a test schedule is in Fig. 4, where the number of wrapper-chains sc_{1k} from core c_1, sc_{3k} from core c_3 and sc_{5k} from core c_5 are scheduled in session k.

For each test session we have to select:

- from which cores to include test vectors,
- the number of test vectors in each partition,
- the number of wrapper-chains for each partition,
- which TAM wires to use for each partition,
- the end time for each of the partitions.

with the objective to minimize the total test application time.

We have a set of transformations that we can apply to

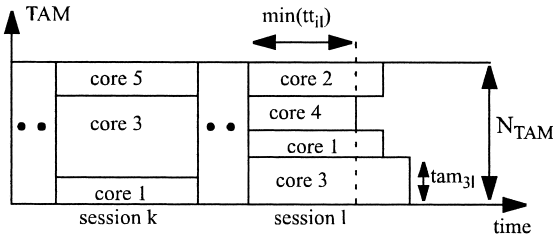


Fig. 4 Session length based on preemption.

each test set. We can sub-divide the test sets for preemptive scheduling and we can modify the TAM usage for each test set. Combining the transformations and preemption means that we have a high degree of flexibility in the test scheduling process when it comes to determine the test time at each core. It also means that we have to check for the possibility of achieving an optimal solution by either assigning all TAM wires to each core and schedule all tests in a sequence, or by dividing each test set into several very small test sets, which easily can be scheduled. However, there are a number of factors limiting both of these approaches, which justifies our approach:

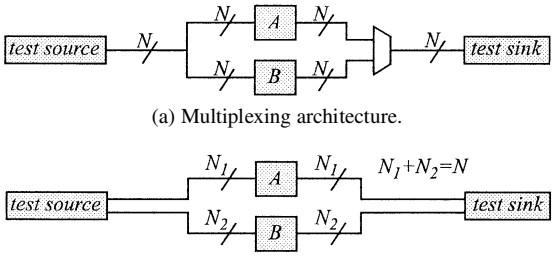
1. scan-chains cannot be too short since it would lead to high routing inside the core. Aerts and Marinissen set a limit of 20 scanned elements in each scan-chain [1]. In theory, each scanned element could form a scan-chain. However, the routing overhead would be unacceptable since each flip-flop has to be directly accessible.
2. the assignment of TAM wires for a core may not always result in an integer result:

$$\Delta_i = \left\lceil \frac{ff_i}{n_i} \right\rceil - \frac{ff_i}{n_i} \tag{2}$$

   For instance, if a core has 8 flip-flops, which should form 3 scan-chains, the length of the chains has to be an integer, the result is rounded to 3. The loss is given by Δ_i.
3. dividing the test set into several test sets increases the total test application time. Assume we have a core with a test set of 10 vectors, 20 flip-flops and a single TAM wire. Its test time is given by: $(10+1)\times 20/1+10 = 230$. If the test set is divided into two sets, each with 5 test vectors the test time is: $(5 + 1) \times 20/1 + 5 + (5 + 1) \times 20/1 + 5 = 250$.
4. a high TAM size results in a higher "area" per test. If we compute the product ("area") given by *test time* × *TAM wires* for the test set above assuming a single TAM wire and 10 TAM wires. In the case with one single TAM wire the product is: $((10+1)\times 20/1+10)\times 1 = 230$ and in the case with 10 TAM wires the product is: $((10 + 1) \times 20/10 + 10) \times 10 = 320$.

5. Analysis of Previous Test Architectures

In this section, we analyse the MA (Multiplexing architecture) and the DA (Distribution architecture) (Fig. 5), both



(a) Multiplexing architecture.

(b) Distribution architecture.

Fig. 5 Multiplexing architecture and distribution architecture [1].

Table 1 Design data for benchmark IC [1].

Core c_i	Flip-flops ff_i	Test vectors tv_i
1	6000	1100
2	3000	900
3	2600	1100
4	1500	1000
5	1500	800
6	800	1000
7	800	400
8	600	500
9	300	300
10	150	400
11	120	150

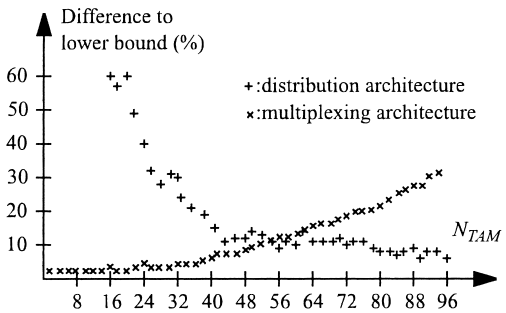


Fig. 6 Difference to lower bound for MA and DA.

are proposed by Aerts and Marinissen [1]. In MA each core is given the full TAM bandwidth when it is to be tested, which means the tests are scheduled in a sequence. For cores where the number of scan-chains is smaller than the TAM bandwidth, the TAM is not fully utilized.

In DA, each core is given its dedicated part of the TAM, which means that initially all cores occupy a part of the TAM. The approach assumes that the bandwidth of the TAM is at least as large as the number of cores, ($|C| \leq N_{\text{TAM}}$).

We have made an analysis of the test application time on the IC benchmark (Table 1) for the MA and the DA. We assume that each scan chains must include at least 20 flip-flops (same assumption as [1]) and where the size of the TAM is in the range $|C| \leq N_{\text{TAM}} \leq 96$, Fig. 6. The lower bound of the test application time, excluding the capture cy-

cles and the shift out of the last response, is defined by Aerts and Marinissen [1] as:

$$\sum_{i=1}^{|C|} \frac{ff_i \times tv_i}{N_{\text{TAM}}} \tag{3}$$

The results in Fig. 6 indicates that the DA is not efficient for low TAM size while MA is less efficient as the TAM size increases.

6. The Preemptive Test Scheduling Algorithm

In this section, we describe the PTS (preemptive test scheduling) algorithm, which is outlined in Fig. 7. The objective is to minimize the test application time. The idea is to assign TAM wires to the cores in each session such that Eq. (2) is minimized. The algorithm starts by trying to find sessions with a single core fully utilizing the TAM. The number of cores in a session is increased until $|C|$. If not all vectors are scheduled, the allowed fault (Δ) increases and the algorithm restarts. The algorithm terminates when all test vectors are scheduled.

In Fig. 4 core 1, 2, 3 and 4 have been chosen to be included in session l. The TAM assignment for each core has been completed and the session length (preemption time) is determined by $min(tt_{il})$, see Fig. 4.

Figure 4 also illustrates that each core can be assigned to a different number of TAM wires when its test vectors are split up into several sessions, *i.e.* tam_{3k} is not equal to tam_{3l}.

Our algorithm schedules the tests in sessions which minimizes the number of needed control steps. Each session will form a control step. However, we are making use of the flexible core wrapper, which will make use of additional control lines. In the experimental results we therefore report the number of wrapper configurations.

7. Experimental Results

We have made a comparison between the MA (multiplexing architecture) [1], the DA (distributed architecture) [1] and our proposed PTS (preemptive test scheduling) technique (Fig. 7). The three approaches have been implemented and the benchmarks we have used are the IC benchmark [1] and the ITC'02 benchmarks, D695, G1023, P22810, P34392 and P93791 [15]. The data for the IC benchmark is in Table 1. For the ITC'02 benchmarks, we excluded all non-scan tested cores and assumed that for each core, only the number of flip-flops and the number of test vectors are given. The ITC'02 benchmarks as we used them are presented in Table 2.

For every benchmark, we made experiments at 12 different TAM bandwidths. In the cases where there is no result for the DA, it is due to the technique cannot be used when the TAM size is less than the number of cores. All experiments were performed on a SunBlade 1000, 900 MHz with 1024 Mb RAM memory and all experimental results are collected in Table 4 and the results are summarized in Table 5.

```
j=0 /session number/
Δ=initial value;
R_TAM=Σ N_i
until |C| = 0 begin / all test vectors at all cores scheduled /
  for k= 1 to |C| begin
    for all possible SC_j where |SC_j|=k and
        tam_ij≤N_i and Σ tam_ij=min(N_TAM,R)
        and Σ Δ_i≤Δ and Σ tp_ij≤P_max
    begin
        t=min(tt_ij)  / length of session /
        for all sc_ij ∈ SC_j begin
            sc_ij=(t×tam_ij-ff_i)/(tam_ij+ff_i)  / vectors in session j /
            tv_i=tv_i-sc_ij  / preempt test vectors /
            if tv_i=0 then begin
                R=R-N_i
                remove c_i from C
            end
        end
        j=j+1  / new session /
    end
  end
  Δ=Δ+δ  / increase allowed fault /
end
```

Fig. 7 The preemptive test scheduling algorithm.

The experimental results in Table 4 are organized as follows. For each benchmark, we have made experiments at 12 different TAM bandwidths. We have for each of the scheduling techniques collected the test application time, the difference to lower bound and the computational cost (CPU time). The test application time is given as the time when the last test finish and the lower bound is computed using Eq. (3). For our approach, we have also computed the number of cores with flexible wrappers and the number of flexible configurations, which are used to indicate the additional overhead. The overhead is counted as follows. For all cores requiring one TAM bandwidth, there is no cost. However, as soon as more than one configuration is required, we count all needed configurations including the first. For instance, at design IC at TAM bandwidth 8, 7 wrapper configurations are needed, which are distributed over 3 wrappers where 2 wrappers have 2 configurations each and 1 wrapper has 3 configurations (2 + 2 + 3). For each benchmark, we have also computed the average test application time, average CPU time and for our approach the average number of wrapper configurations and the average number of flexible wrappers.

For every bandwidth at all benchmarks our approach produces a solution with the test application time closest to lower bound. The computational cost of using the MA and the DA is extremely low. We have a slightly higher computational cost, however, in most of the cases we only require a few seconds. Our approach assumes flexible wrappers, which has a cost but both the number of flexible wrappers and the number of wrapper configurations are low.

For the IC benchmark (first group in Table 4) we let the TAM width be in the range from 8 to 96 in steps of 8 and we did not allow any scan chain include less than 20 scan

Table 2 Design data for the ITC '02 benchmarks D695, G1023, P22810, P34392 and P93791.

Core	flip-flops	test vectors	Core	flip-flops	test vectors
Design: D695			Design: G1023		
3	32	75	1	592	273
4	211	105	2	167	215
5	1426	110	3	53	171
6	638	234	4	216	155
7	534	95	5	127	27
8	179	97	6	94	18
9	1728	12	7	94	18
10	1636	68	8	104	80
Design: P22810			9	64	34
1	1122	785	10	13	377
5	2255	202	11	9	191
9	2234	175	12	13	161
10	209	38	Design P34392		
11	589	94	1	806	210
12	714	93	2	8856	514
13	280	1	10	4731	454
14	78	108	18	6555	745
15	422	37	Design: P93791		
16	109	8	1	6801	409
17	118	25	4	108	11
18	315	644	6	23789	218
19	100	58	11	576	187
20	231	124	12	4265	391
21	1054	465	13	9527	194
22	166	59	14	9527	194
23	289	40	17	6391	216
24	180	27	19	4349	210
25	2322	215	20	7450	416
26	11485	181	23	7639	234
27	34	2	27	3026	916
28	417	26	29	6525	172

flip-flops. Our approach finds the solution with the test application time closest to the lower bound for all bandwidths. The computational cost for the MA and the DA approaches are below 1 second, however, our approach requires only 2.4 seconds on average.

A TAM schedule on the IC benchmark at TAM size 40 is presented in Table 3. The schedule consists of 11 sessions and for each session, its test time is shown as well as the cores tested, their TAM assignment and the number of test vectors. The total test application time is equal to the summation of the test time at each test session.

The TAM schedule (Table 3) demonstrates also how our algorithm proceeds. Our algorithm starts by trying to assign all TAM wires to a single core. After trying all cores, the algorithm tries combinations with two cores and continues to increase the number of cores on until $|C|$ is reached. If not all test vectors are scheduled, the algorithm restarts

Table 3 Test schedule for IC at TAM size 40.

S	Time	Core - c, TAM wires - t, Vectors - v		
0	8420	c:7 t:40 v:400		
1	21020	c:6 t:40 v:1000		
2	72665	c: 3 t:40 v:1100		
3	68475	c: 2 t:40 v:900		
4	166250	c: 1 t:40 v:1100		
5	9330	c: 8 t:30 v:443	c: 9 t:10 v:300	
6	3537	c: 5 t:30 v:68	c: 8 t:10 v:57	
7	51050	c: 4 t:30 v:1000	c: 5 t:10 v:337	
8	4680	c: 5 t:30 v:90	c:10 t: 6 v:179	c:11 t: 4 v:150
9	5771	c: 5 t:34 v:124	c:10 t:6 v:221	
10	7097	c: 5 t:40 v:181		
Σ	418295			

with a little higher Δ (allowed fault), with one core and continuous until all test vectors are scheduled. A re-start, is performed after session 8 (there are two cores scheduled at session 9 and one in session 10).

The TAM schedule (Table 3) also demonstrates the use of flexible wrappers. For cores that appear only once in the schedule, only one TAM bandwidth is assigned to those cores and a flexible wrapper is not needed. An example of such a core is core 11 (Table 3). An example of a core that appears several times is core 5, which appears in 5 different sessions. However, core 5 does not require 5 different TAM bandwidths, only 4 since it uses 30 TAM wires in both session 6 and 8. For this TAM schedule (Table 3), we need 2 flexible wrappers; one for core 5 and one for core 8. The flexible wrapper at core 5 requires 4 configurations and the flexible wrapper at core 8 requires 2 configurations; in total 6 configurations.

For D695, G1023, P22810, P34392 and P93791 the results in Table 5 shows that our approach finds a solution closest to the lower bound. In all cases, the computational cost is low. Only at P22810 the computational cost is higher, however, it is still in the range of a few minutes. Furthermore, the additional overhead due to the flexible wrapper is low. Only a few cores require a flexible wrapper with only a few configurations at each such core wrapper.

8. Conclusions

In this paper, we have proposed a preemptive test scheduling technique for scan tested core-based systems. We have made use of a core wrappers allowing a flexible number of wrapper-chain configurations at each core. The advantage with the possibility of modifying the bandwidth is that it increases the flexibility in the scheduling process.

We have made an analysis of previously proposed techniques and modelled the scheduling problem as a Bin-packing problem. Our test scheduling technique makes use of the possibility of using preemption and flexible wrappers. Experiments comparing our implementation with other approaches show that our technique produces solutions with

Table 4 Experimental results of IC [1], D695, G1023, P22810, P34392, and P93791 [15] using MA [1], DA [1], and our preemptive TAM scheduling (PTS) technique.

Design	TAM width	Distribution Architecture (DA)			Multiplexing Architecture (MA)			Our PTS, Preemptive TAM Scheduling					Lower bound
		Test application time	Difference to lower bound (%)	CPU (s)	Test application time	Difference to lower bound (%)	CPU (s)	Test application time	Difference to lower bound (%)	CPU (s)	#flexible wrapper configurations	#flexible wrappers	Test time
IC	8	Not applicable			2068931	0.6	<1	2065770	0.5	1	7	3	2056000
	16	1652600	60.8	<1	1045764	1.7	<1	1036738	0.8	<1	6	2	1028000
	24	945758	38.0	<1	707115	3.2	<1	693288	1.2	<1	5	2	685333
	32	661700	28.7	<1	537240	4.5	<1	521038	1.4	<1	5	2	514000
	40	478934	16.5	<1	436338	6.1	<1	418295	1.7	2	6	2	411200
	48	389753	13.7	<1	376179	9.8	<1	349524	2.0	3	9	4	342666
	56	321200	9.4	<1	331535	12.9	<1	299025	1.8	<1	8	3	293714
	64	288461	12.2	<1	297802	15.9	<1	263468	2.5	1	5	2	257000
	72	251250	10.0	<1	272477	19.3	<1	240453	5.3	6	11	4	228444
	80	221300	7.6	<1	252758	22.9	<1	211582	2.9	<1	6	2	205600
	88	201200	7.6	<1	240146	28.5	<1	194506	4.1	3	7	3	186909
	96	181100	5.7	<1	228635	33.4	<1	176344	2.9	8	11	4	171333
Average:			19.1	1		14.1	1		2.3	2.4	7.1	2.8	
D695	4	Not applicable			135360	2.0	<1	135360	2.0	<1	0	0	132696
	8	158396	138.7	<1	68422	3.1	<1	68422	3.1	<1	0	0	66348
	12	75199	70.0	<1	46174	4.4	<1	46174	4.4	<1	0	0	44232
	16	50289	51.6	<1	35077	5.7	<1	34806	4.9	<1	4	2	33174
	20	31856	20.0	<1	28193	6.2	<1	27898	5.1	<1	6	3	26539
	24	25727	16.3	<1	23900	8.1	<1	23347	5.6	1	5	2	22116
	28	22476	18.6	<1	20649	8.9	<1	20411	7.7	<1	7	3	18956
	32	19034	14.8	<1	18199	9.7	<1	17759	7.1	<1	8	4	16587
	36	17183	16.5	<1	16384	11.1	<1	15933	8.1	<1	2	1	14744
	40	15274	15.1	<1	14977	12.9	<1	14371	8.3	<1	0	0	13269
	44	13319	10.4	<1	14054	16.5	<1	13218	9.6	1	7	3	12063
	48	12320	11.4	<1	13131	18.7	<1	12250	10.8	<1	2	1	11058
Average:			34.9	1		8.9	1		6.4	1	3.4	1.6	
G1023	10	Not applicable			29448	10.7	<1	28361	6.6	<1	3	1	26608
	12	162481	632.8	<1	24935	12.5	<1	24904	12.3	6	2	1	22173
	14	54525	186.9	<1	21379	12.5	<1	21041	10.7	<1	2	1	19006
	16	36287	118.2	<1	18997	14.2	<1	18483	11.1	1	0	0	16630
	18	32879	122.4	<1	17120	15.8	<1	16400	10.9	<1	0	0	14782
	20	23563	77.1	<1	15860	19.2	<1	15324	15.2	1	0	0	13304
	22	18359	51.8	<1	14522	20.1	<1	13735	13.6	43	0	0	12094
	24	17003	53.4	<1	13564	22.4	<1	12965	16.9	55	0	0	11086
	26	15069	47.2	<1	12907	26.1	<1	12212	19.3	1	0	0	10234
	28	12877	35.5	<1	12089	27.2	<1	11491	20.9	2	2	1	9503
	30	12055	35.9	<1	11541	30.1	<1	10722	20.9	123	2	1	8869
	32	11233	35.1	<1	11010	32.4	<1	10073	21.1	2	2	1	8315
Average:			126.9	1		21.9	1		16.4	24.8	1.1	0.5	

Table 4 (Continued.)

Design	TAM width	Distribution Architecture (DA)			Multiplexing Architecture (MA)			Our PTS, Preemptive TAM Scheduling					Lower bound
		Test application time	Difference to lower bound (%)	CPU (s)	Test application time	Difference to lower bound (%)	CPU (s)	Test application time	Difference to lower bound (%)	CPU (s)	#flexible wrapper configurations	#flexible wrappers	Test time
P22810	8	Not applicable			661774	1.3	<1	661774	1.3	75	0	0	653454
	16	Not applicable			333432	2.1	<1	333418	2.0	136	0	0	326727
	24	882677	305.2	<1	223983	2.8	<1	223037	2.4	150	21	7	217818
	32	393359	140.8	<1	169653	3.9	<1	168550	3.2	150	15	6	163363
	40	229186	75.4	<1	137365	5.1	<1	136753	4.6	237	6	2	130690
	48	167399	53.7	<1	115305	5.9	<1	114014	4.7	150	7	2	108909
	56	130857	40.1	<1	100268	7.4	<1	98413	5.4	150	5	2	93350
	64	110291	35.0	<1	88372	8.2	<1	86621	6.0	150	7	2	81681
	72	95367	31.3	<1	79676	9.7	<1	77274	6.4	252	8	4	72606
	80	80957	23.9	<1	73362	12.3	<1	70808	8.4	352	5	2	65345
	88	71927	21.1	<1	66788	12.4	<1	63742	7.3	353	7	2	59404
	96	65771	20.8	<1	62250	14.3	<1	59359	9.0	145	18	6	54454
Average:			74.7	1		7.1	1		5.1	226.2	8.2	2.9	
P34392	8	2153059	46.6	<1	1474419	0.4	<1	1474419	0.4	<1	0	0	1469074
	16	816123	11.1	<1	740855	0.9	<1	739207	0.6	1	2	1	734537
	24	538719	10.0	<1	499534	2.0	<1	493126	0.7	<1	3	1	489691
	32	380584	3.6	<1	377930	2.9	<1	370504	0.9	<1	3	1	367268
	40	306605	4.4	<1	305824	4.1	<1	297019	1.1	<1	2	1	293814
	48	253894	3.7	<1	257527	5.2	<1	247498	1.1	<1	5	2	244845
	56	216124	3.0	<1	223593	6.5	<1	213427	1.7	<1	4	2	209867
	64	189483	3.2	<1	197098	7.3	<1	187342	2.0	<1	4	2	183634
	72	169434	3.8	<1	177012	8.4	<1	165515	1.4	<1	3	1	163230
	80	152954	4.1	<1	161097	9.7	<1	149510	1.8	<1	2	1	146907
	88	137263	2.8	<1	150725	12.9	<1	135687	1.6	3	4	2	133552
	96	126819	3.6	<1	142129	16.1	<1	125887	2.8	1	4	2	122422
Average:			8.3	1		6.4	1		1.3	1.1	3.0	1.3	
P93791	8	Not applicable			3081717	0.6	<1	3081717	0.6	<1	0	0	3064398
	16	2775758	81.2	<1	1544043	0.8	<1	1544043	0.8	<1	0	0	1532199
	24	1394819	36.6	<1	1032830	1.1	<1	1032446	1.1	<1	0	0	1021466
	32	929174	21.3	<1	776871	1.4	<1	776487	1.4	<1	0	0	766099
	40	744599	21.5	<1	624244	1.9	<1	623860	1.8	<1	0	0	612879
	48	598779	17.2	<1	522517	2.3	<1	522133	2.2	<1	0	0	510733
	56	473915	8.3	<1	450159	2.8	<1	449775	2.7	<1	0	0	437771
	64	444521	16.0	<1	394439	3.0	<1	394055	2.9	1	0	0	383049
	72	372518	9.4	<1	352728	3.6	<1	352344	3.5	<1	0	0	340488
	80	345692	12.8	<1	318273	3.9	<1	317889	3.7	<1	0	0	306439
	88	306818	10.1	<1	290426	4.3	<1	290042	4.1	1	0	0	278581
	96	278767	9.2	<1	266769	4.5	<1	266385	4.3	1	0	0	255366
Average:			20.3	1		2.5	1		2.4	1	0	0	

Table 5 Summery of the experimental results.

Design	#cores	minimal scan-chain length	Multiplexing architecture		Distribution architecture		Our preemptive TAM scheduling algorithm			
			average % from LB	average CPU time	average % from LB	average CPU time	average % from LB	average CPU time	#flexible configurations	#flexible wrappers
IC	11	20	19.1	1	14.1	1	2.3	2.4	7.1	2.8
D695	8	5	34.9	1	8.9	1	6.4	1	1.1	0.5
G1023	12	1	126.9	1	21.9	1	16.4	1	3.4	1.6
P22810	22	5	74.7	1	7.1	1	5.2	226.2	8.2	2.9
P34392	4	60	8.3	1	6.4	1	1.4	1	3.0	1.3
P93791	13	30	20.3	1	2.5	1	2.4	1	0	0

the lowest test application time at a low computational cost.

References

[1] J. Aerts and E.J. Marinissen, "Scan chain design for test time reduction in core-based ICs," Proc. IEEE International Test Conference (ITC), pp.448–457, Washington, DC, Oct. 1998.

[2] M.L. Bushnell and V.D. Agrawal, Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits, Kluwer Academic Publ., ISBN 0-7923-7991-8.

[3] K. Chakrabarty, "Test scheduling for core-based systems using mixed-integer linear programming," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.19, no.10, pp.1163–1174, Oct. 2000.

[4] R. Chou, K. Saluja, and V. Agrawal, "Scheduling tests for VLSI systems under power constraints," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.5, no.2, pp.175–185, June 1997.

[5] T. Cormen, C. Leiserson, and R. Rivest, Introduction to Algorithms, The MIT Press, ISBN 0-262-03141-8, 1989.

[6] S.K. Goel and E.J. Marinissen, "Effective and efficient test architecture design for SOCs," Proc. International Test Conference (ITC), pp.529–538, Oct. 2002.

[7] S.K. Goel and E.J. Marinissen, "Cluster-based test architecture design for system-on-chip," Proc. VLSI Test Symposium (VTS), pp.259–264, Monterey, CA, April 2002.

[8] Y. Huang, W.-T. Cheng, C.-C. Tsai, N. Mukherjee, O. Samman, Y. Zaidan, and S.M. Reddy, "Resource allocation and test scheduling for concurrent test of core-based SOC design," Proc. IEEE Asian Test Symposium (ATS), pp.265–270, Kyoto, Japan, Nov. 2001.

[9] 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," Proc. International Test Conference (ITC), pp.74–82, Baltimore, MD, Oct. 2002.

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

[11] V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "Test wrapper and test access mechanism co-optimization for system-on-chip," Proc. IEEE International Test Conference (ITC), pp.1023–1032, Baltimore, MD, Nov. 2001.

[12] V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "Co-optimization of test wrapper and test access architecture for embedded cores," J. Electron. Test., Theory Appl. (JETTA), pp.213–230, April 2002.

[13] V. Iyengar K. Chakrabarty, and E.J. Marinissen, "Efficient wrapper/TAM co-optimization for large SOCs," Proc. Design and Test in Europe (DATE), pp.491–498, Paris, France, March 2002.

[14] V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "On using rectangle packing for SOC wrapper/TAM co-optimization," Proc. VLSI Test Symposium (VTS), pp.253–258, Monterey, CA, April 2002.

[15] E.J. Marinissen, V. Iyengar, and K. Chakrabarty, ITC'02 SOC Benchmarks, http://www.extra.research.philips.com/itc02socbenchm

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

[17] S. Koranne and V. Iyengar, "On the use of k-tuples for SoC test schedule representation," Proc. International Test Conference (ITC), pp.539–548, Oct. 2002.

[18] E. Larsson and Z. Peng, "An integrated system-on-chip test framework," Proc. Design, Automation and Test in Europe Conference (DATE), pp.138–144, Munchen, Germany, March 2001.

[19] E. Larsson and Z. Peng, "The design and optimization of SOC test solutions," Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp.523–530, San Jose, CA, Nov. 2001.

[20] E.J. Marinissen, R.G.J. Arendsen, G. Bos, H. Dingemanse, M. Lousberg, and C. Wouters, "A structured and scalable mechanism for test access to embedded reusable cores," Proc. IEEE International Test Conference (ITC), pp.284–293, Washington, DC, Oct. 1998.

[21] E.J. Marinissen, Y. Zorian, R. Kapur, T. Taylor, and L. Whetsel, "Towards a standard for embedded core test: An example," Proc. IEEE International Test Conference (ITC), pp.616–627, Atlantic City, NJ, Sept. 1999.

[22] E.J. Marinissen, R. Kapur, M. Lousberg, T. McLaurin, M. Ricchetti, and Y. Zorian, "On IEEE P1500's standard for embedded core test," J. Electron. Test., Theory Appl. (JETTA), vol.18, pp.365–383, Aug. 2002.

[23] V. Muresan, X. Wang, V. Muresan, and M. Vladutiu, "A comparison of classical scheduling approaches in power-constrained block-test scheduling," Proc. IEEE International Test Conference (ITC), pp.882–891, Atlantic City, NJ, Oct. 2000.

[24] Y. Zorian, "A distributed BIST control scheme for complex VLSI devices," Proc. IEEE VLSI Test Symposium (VTS), pp.4–9, Atlantic City, NJ, April 1993.

Erik Larsson is Assistant Professor at the Department of Computer and Information Science at Linköpings Universitet, Linköping, Sweden. He received his M.Sc. and Ph.D from Linköpings University in 1994, 2000, respectively. From October 2001 to December 2002 he was at a Post Doctoral position at the Computer Design and Test Laboratory at Nara Institute of Science and Technology (NAIST), Nara, Japan. The Post Doctoral position that was funded by the Japan Society for the Promotion of Science. 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.

Hideo Fujiwara received the B.E., M.E., and Ph.D. degrees in electronic engineering from Osaka University, Osaka, Japan, in 1969, 1971, and 1974, respectively. He was with Osaka University from 1974 to 1985 and Meiji University from 1985 to 1993, and joined Nara Institute of Science and Technology in 1993. In 1981 he was a Visiting Research Assistant Professor at the University of Waterloo, and in 1984 he was a Visiting Associate Professor at McGill University, Canada. Presently he is a Professor at the Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan. His research interests are logic design, digital systems design and test, VLSI CAD and fault tolerant computing, including high-level/logic synthesis for testability, test synthesis, design for testability, built-in self-test, test pattern generation, parallel processing, and computational complexity. He is the author of Logic Testing and Design for Testability (MIT Press, 1985). He received the IECE Young Engineer Award in 1977, IEEE Computer Society Certificate of Appreciation Award in 1991, 2000 and 2001, Okawa Prize for Publication in 1994, IEEE Computer Society Meritorious Service Award in 1996, and IEEE Computer Society Outstanding Contribution Award in 2001. He is an advisory member of IEICE Trans. on Information and Systems and an editor of IEEE Trans. on Computers, J. Electronic Testing, J. Circuits, Systems and Computers, J. VLSI Design and others. Dr. Fujiwara is a fellow of the IEEE, a Golden Core member of the IEEE Computer Society, and a member of the Information Processing Society of Japan.