Preemptive Power Constrained TAM Scheduling for Scan-based System-on-Chip

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]

We investigate power constrained test access mechanism (TAM) scheduling for core-based system by combining scan-chain partitioning and preemption. We discuss scan-chain partitioning in core-based design to minimize test time while satisfying test power constraint and we outline a wrapper supporting a variable number of scan-chains at a core. We demonstrate that optimal test time can be achieved both for a fixed TAM and under power constraint. We also investigate practical limitations, model the problem as a Knapsack problem and propose an algorithm. Experiments using our implementation shows its efficiency compared to previous approaches.

1 Introduction

To manage the increasing complexity of digital systems, the core-based design technique, SOC (system-on-chip), has been developed placing a complete system on a single chip. The approach shows similarities with traditional PCB (printed circuit board) design technique, however, from a testing perspective, there is a major difference. In both approaches test data is transported in and out of the system. However, for PCB systems the amount is less since components are tested prior to mounting which is not the case for cores in core-based designs. In addition, due to the increasing design complexity, substantial amount of test data is to be transported in and out of a SOC design leading to long testing times.

Several techniques have been proposed to minimize the test time, including test scheduling [1,2,3,4,5] and test vector set reduction [6]. Techniques have also been proposed to reduce test power dissipation allowing testing at higher clock frequencies [7,8].

The basic idea in test scheduling is to schedule tests in parallel so that many test activities can be performed concurrently. However, there are usually many conflicts, such as sharing of common resource, in a system under test, which inhibit parallel testing. Therefore the test scheduling issue must be taken into account during the design of the system under test, in order to maximize the possibility for parallel testing. Further, test power constraints must be considered carefully, otherwise the system under test may be damaged due to overheating.

We have proposed an integrated SOC test framework for early design space exploration as well as for extensive optimization of the final test solution [9,10]. The framework provides a design environment to treat test scheduling under test conflicts and test power constraints as well as test set selection, test resource placement and test access mechanism (TAM) design in a systematic way. We have also investigated scan-chain division under power constraint [11].

In this paper, we investigate preemptive TAM scheduling under power constraint. We discuss and outline a wrapper for scan-based design for different types of cores minimizing test power while allowing flexible scan chain length. We demonstrate that optimal test time can be achieved using test set reduction for TAM scheduling as well as under power constraint. We also investigate practical limitations. We analyse test set reduction techniques for different TAM bandwidths and model the problem as a Knapsack problem. We propose a technique combining preemption-based test scheduling technique [5] and test set reduction [6] under power constraint.

The rest of the paper is organized as follows. An overview of related work is in Section 2, and preliminaries are given in Section 3. In Section 4 we analyse previous proposed techniques and in Section 5 we demonstrate how to achieve optimal test time under power constraint. In Section 6 we show how to achieve optimal test time under TAM bandwidth limitation. The details of our approach are presented in Section 7. Finally, the experimental results are presented in Section 8 and paper is concluded in Section 9.

2 Related Work

2.1 Test Scheduling

Scheduling the tests in a system means the start time and end time are determined for all tests in the system while satisfying all constraints minimizing the test time. Several techniques have been proposed and they can be divided into:

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

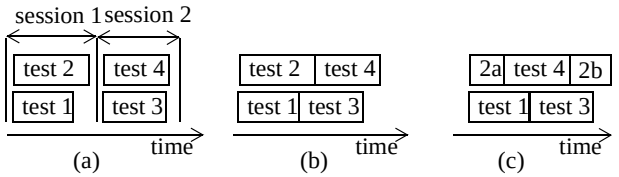


Figure 1. Test scheduling strategies.

- *Non partitioned testing* with techniques proposed by Zorian [2] and Chou *et al*.[3], see Figure 1(a),
- *Partitioned testing with run to completion* with work done by Chakrabarty [1] and Muresan *et al.* [4], see Figure 1(b) for illustration, and
- *Partitioned (preemptive) testing* where Iyengar and Chakrabarty [5] proposed a technique, see Figure 1(c).

All approaches minimizes test time but take different issues in consideration. For instance, Chakrabarty proposed a test scheduling technique for core-based systems considering external and BIST test conflicts [1]. The technique proposed by Zorian minimizes the number of control lines for BIST (Built-In Self-Test) systems under power constraint [2]. The test conflicts in such systems are few due to that each testable unit has its dedicated test resources. For general systems, Chou *et al.* [3] and Muresan *et al.* [4] have proposed techniques to minimize test time under power limitations and conflicts. In the approach by Chou *et al.* [3] a resource graph is used to model the system where an arc between a test and a resource indicate that the resource is required for the test, Figure 2. From the resource graph, a TCG (test compatibility graph) is generated (Figure 3) where each test is a node and an arc between two nodes indicate that the tests can be scheduled concurrently. For instance t_1 and t_2 can be scheduled at the same time. Each test is attached with its test time and its power consumption and the maximal allowed power consumption is 10. The tests t_1, t_2, t_3 are compatible, however, due to the power limit they can not be scheduled at the same time.

The above test scheduling approaches focus on a fixed test time for all test sets. Iyengar and Chakrabary proposed a preemption-based test scheduling technique [5] where each test set can be interrupted and resumed later.

2.2 Scan-chain Partitioning

By scan-chain partitioning or test set reduction the test vectors in a given test are rearranged in such a way that test can be executed in parallel. In scan testing each test vector is shifted in (scanned in), and after a capture cycle, the test response is shifted out (scanned out). Even if pipelining is used shifting in a new test vector at the same time as the test response from the previous test vector is shifted out, the shift-in and shift-out process contributes to a major part of the test time. It can be reduced by partition the scan flip flops into several chains of shorter length.

Another advantage with test partitioning, beside test time minimization, is that the time when a resource is required

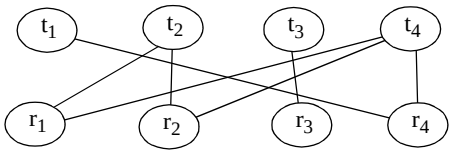


Figure 2. An example of a resource graph.

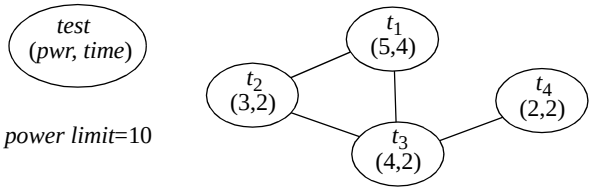


Figure 3. Test compatibility graph of the example (Figure 2).

for a particular test is reduced, which reduces the impact of test conflicts. For instance, if test t_4 that requires r_4, in the example given in Figure 2, is parallelized by a factor n, the time when r_4 is used by t_4 is also reduced by a factor n.

Aerts and Marinssen [6] investigated the problem of dividing scan-chains for test time minimization where the constraints are defined by available pins (bandwidth).

Iyengar *et al*. proposed a co-optimization for wrapper and TAM co-optimization where either the TAM is optimized for a set of pre-defined wrappers or the wrapper is optimized for a given TAM [12].

2.3 Scan Chain Power Consumption

The shift in and shift out process of scan-based systems does not only contribute to a major part of the test time, it also contributes to a major part of the test power consumption [7]. Gerstendörfer and Wunderlich [7] proposed a technique to isolate the flip flops in the scan chain during the shift in and shift out process. However, the introduction of an extra level of gates per flip-flop may cause an effect on the critical path.

An approach to minimize test power is proposed by Nicoli and Al-Hashimi by identify useless test power consumption [8].

3 Preliminaries

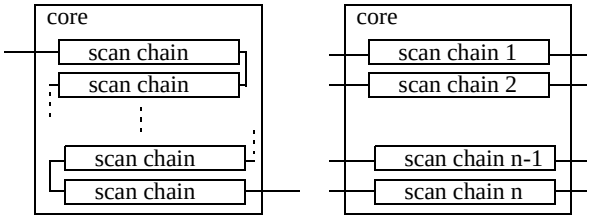
In this section we discuss scan chain partitioning in core-based systems, wrappers, test power consumption and design for low power, and our system model.

3.1 Scan Chain Partitioning in Core-based Designs

In a core-based design environment each of the cores to be used can be given to the designer in different format. There are basically three types of cores [13]:

- Soft cores comes in the form of synthesizable RTL (register-transfer level) descriptions,
- Firm cores are supplied as gate-level netlists, and
- Hard cores are available as non modifiable layouts.

The soft cores allow more flexibility for the designer



(a) fixed scan chain length. (b) variable scan chain length

Figure 4. Scan-chains design at a core.

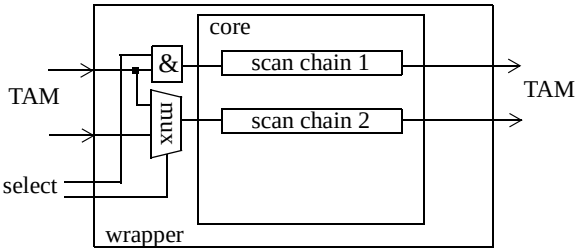


Figure 5. Scan-chains design at a core.

compared to firm cores and hard cores. This is also true when determining the type of test method. For scan-based testing, soft cores allow a higher flexibility when it comes to determine the number of scan-chains and their length. However, when designing 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 Figure 4. In Figure 4(a) a single scan chain is used while in (b) a fixed set of *n* scan chains are used. In both cases the number of scan chains are fixed, however, in Figure 4(b) the chains can externally be con figured into a variation of scan chain lengths.

The advantages of Figure 4(b) is not only that a variety of scan chain lengths can be achieved but also the test power dissipation decreases. In Figure 4(b), when a single scan-chain is assumed, only one partition of the scan-chain is active at any time. By dividing the scan chain into several of shorter length, the activity in the scan chain is reduced and since the test power highly depends on the activity the consumed power is only 1/*n* in Figure 4(b) compared to (a).

A disadvantage of scan chain partitioning is that the number of required primary inputs and primary outputs increases. However, the core is aimed to be embedded in an SOC where the additional pin cost is not comparable with pin cost for components in PCB designs.

3.2 Test Wrapper Design

Test access is eased by placing the core in a wrapper such as the TestShell proposed by Marinissen *et al.* [15]. A standard under development is the IEEE P1500 Standard for Embedded Core Test, consisting of a Core Test Language and a Core Test Wrapper [16]. The P1500 wrapper is similar to the TestShell. A major difference between TestShell and P1500 is that the latter only allow a single bit bypasses while the TestShell allows a TAM wide bypass.

Recently, Marinissen *et al*. proposed a library of wrapper cells allowing a flexible design [17]. For instance, it helps the test designer when using all types of cores, to determine the length and number of scan chains. In Figure 5 we demonstrate how to achieve a flexible scan chain length for a hard core. Depending on the selectors the two partitions can either form a single scan chain or two scan chains.

Another advantage with the approach proposed by Marinissen *et al*. is that it also allows the test designer to design wrappers with no delayed (clocked) bypass [17].

3.3 Test Power Consumption

Generally speaking, there are more switching activities during the testing mode of a system than when it is operated under the normal mode. The power consumption of a CMOS circuit is given by a static part and a dynamic part. The dynamic part dominates and can be characterized by:

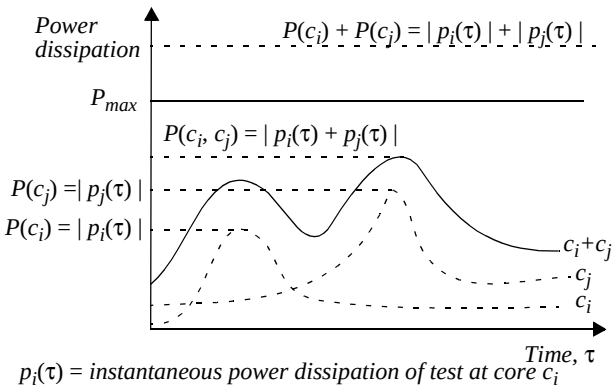
$$p = C \times V^2 \times f \times \alpha \qquad 1$$

where the capacitance *C*, the voltage *V*, and the clock frequency *f* are fixed for a given design [14]. The switch activity α, on the other hand, depends on the input to the system, which during the test mode are the test vectors and therefore the power dissipation vary depending on them.

An example illustrating the test power dissipation variation over time τ for the tests at two cores c_i and c_j are given in Figure 6. Let $p_i(\tau)$ and $p_j(\tau)$ be the instantaneous power dissipation of two compatible tests at c_i and c_j, respectively, and $P(c_i)$ and $P(c_j)$ be the corresponding maximal power dissipation.

If $p_i(\tau) + p_j(\tau) < P_{max}$, the two tests can be scheduled at the same time. However, instantaneous power of each test vector is hard to obtain. To simplify the analysis, a fixed value tp_i is usually assigned for all test vectors at a core c_i such that when the test is performed the power dissipation is no more than tp_i at any moment.

The tp_i can be assigned as the average power dissipation over all test vectors at c_i or as the maximum power dissipation over all test vectors at c_i. The former approach



$p_i(\tau)$ = *instantaneous power dissipation of test at core* c_i

$P(t_i) = | p_i(\tau) |$ = *maximum power dissipation of test at core* c_i

Figure 6. Power dissipation as a function of time [3].

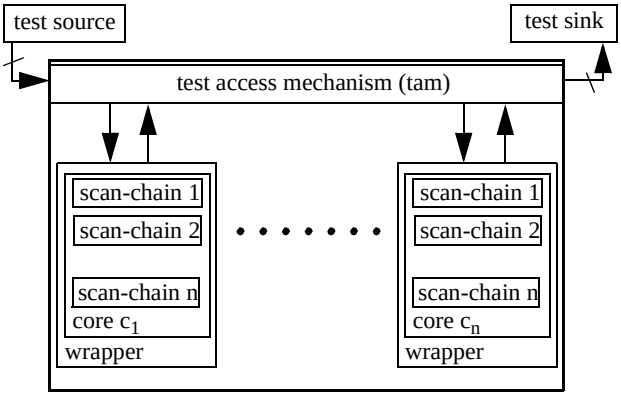


Figure 7. Embedded cores, wrappers and TAM.

could be too optimistic, leading to an undesirable test schedule which exceeds the test power constraints. The latter could be too pessimistic; however, it guarantees that the power dissipation will satisfy the constraints. Usually, in a test environment the difference between the average and the maximal power dissipation for each test is often small since the objective is to maximize the circuit activity so that it can be tested in the shortest possible time [3]. Therefore, the definition of power dissipation tp_i for a test at c_i is usually assigned to the maximal test power dissipation ($P(c_i)$) when test vectors at c_i alone is applied to the device. This simplification was introduced by Chou *et al.* [3] and has been used by Zorian [2] and by Muresan *et al.* [4]. We will also use this assumption also in our approach.

3.4 System Modeling

An example of a system under test is given in Figure 7 where each core is placed in a wrapper in order to ease test access. The system is tested by applying several set of tests to the system 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 Figure 7, can be modelled as:

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

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

tp_i: test power when active,

tv_i: number of test vectors,

ff_i: number of scanned flip-flops,

N_i: maximal allowed bandwidth,

N_{tam}: bandwidth of the test access mechanism, and

P_{max}: maximal allowed power at any time.

The test time and the test power for a set of test vectors activating n_i scan chains are defined below. The test time for a scan tested core c_i is given by [6]:

$$t_{test}(c_i) = (tv_i + 1) \times \lceil ff_i / n_i \rceil + tv_i \qquad 2$$

at a core with ff_i scanned flip-flops partitioned into n_i scan chains and tv_i test vectors. The formulas assume that a new test vector is scanned in at the same time as the test response is shifted out. This scheme is applicable for all test vectors but when the test response from the last test vector is shifted out and therefore the term +1 is added in Equation 2 [6].

Based on the discussion above (Section 3.1 and 3.3), the test power at a core c_i depends on the number of active scan chains:

$$p_{test}(c_i) = tp_i \times n_i \qquad 3$$

4 Test Vector Set Reduction Analysis

Aerts and Marinissen have proposed three scan chain design architectures, multiplexing, distributed and daisychain [6]. However, due to that bypass can be designed at no clock delay (discussed in Section 3.2), the daisychain architecture is similar to the multiplexing architecture.

In multiplexing architecture each core is given all TAM bandwidth when it is to be tested. It basically means that 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. Furthermore, since the test time is minimized at each core, the test power is maximized, which could damage the core.

In distributed architecture, each core is given its dedicated number of scan chains. That means that initially all cores occupy a part of the TAM. This approach assume that the bandwidth of the TAM is at least as large as the number of cores in the system, ($N_{tam} > |C|$).

We use the benchmark IC to analyse multiplexing and distributed architecture, see Table 1, and let the minimal number of flip flops in a scan chain be 20 [6].

We have made an analysis of the test time on the IC benchmark for multiplexing architecture and distribution architecture where scan chains must include at least 20 flip flops and where the size of the TAM is in the range $|C| < N_{tam} < 96$, Figure 8. The lower bound of the test time excluding the capture cycles and the shift out of the last

Core c_i	ff_i	tv_i
A	6000	1100
B	3000	900
C	2600	1100
D	1500	1000
E	1500	800
F	800	1000
G	800	400
H	600	500
I	300	300
J	150	400
K	120	150

Table 1. Design data for benchmark IC [6].

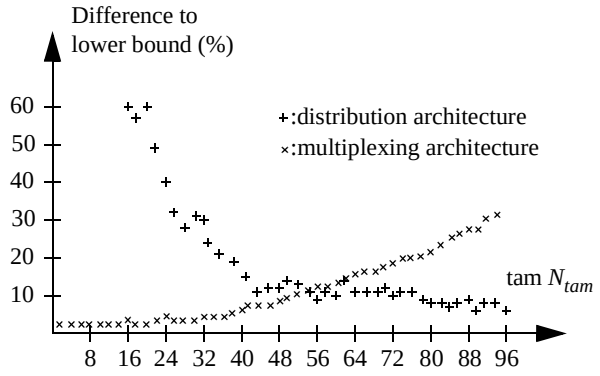


Figure 8. Difference to lower bound for Multiplexing Architecture and Distribution Architecture.

response is given by [6]:

$$\sum_{i=1}^{|C|} \frac{ff_i \times tv_i}{N_{tam}} \qquad 4$$

An obvious conclusion from the analysis (Figure 8) is that the distribution architecture is not efficient for low TAM bandwidth while multiplexing architecture is not efficient as the TAM size increase.

5 Optimal Power Constrained Test Schedule

In this section, we demonstrate how to achieve optimal test time using scan-chain division under power constraints.

5.1 Optimal Test Time

We assume a given system to be modelled as described in Section 3.4 where the test at each core has a test time and a test power consumption attached to it. This can be illustrated using a rectangle for each test (as shown in Figure 9) where the horizontal side corresponds to its test time while the vertical side corresponds to its test power consumption.

A test schedule can be illustrated by placing all tests in a diagram as in Figure 9. At any moment the test power consumption must be below the maximal allowed power limit p_{max}. The rectangle where the vertical side is given by p_{max} and the horizontal side is defined by the total test application time t_{total} characterizes the test feature of a given system under test.

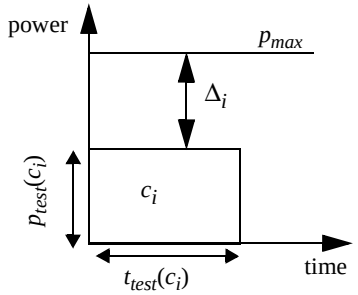


Figure 9. Illustration of the test time and power consumption for a test at a core c_i.

If the rectangle defined by $p_{max} \times t_{total}$ is equal to the summation of $t_{test}(c_i) \times p_{test}(c_i)$ for the tests at all cores, as given by the following equation, we have the optimal solution.

$$\sum_{\forall i} t_{test}(c_i) \times p_{test}(c_i) = p_{max} \times t_{opt} \qquad 5$$

The optimal test time for a system under test is thus:

$$t_{opt} = \sum_{\forall i} \frac{t_{test}(c_i) \times p_{test}(c_i)}{p_{max}} \qquad 6$$

Usually, the optimal test time cannot be achieved due to test conflicts. The worst case occurs when all tests are in conflicts with each other and all tests must be scheduled in sequence. The total test time is then given by:

$$t_{sequence} = \sum_{\forall i} t_{test}(c_i) \qquad 7$$

For a scan-based design the scan-chains can be divided into several, which reduces the test time. If the tests at all cores c_i are allowed to be divided by a factor n_i, the total test time when the tests are scheduled in a sequence is:

$$\sum_{\forall i} t_{test}(c_i)/n_i \qquad 8$$

The lower bound of the degree of partitioning is n_i=1. For a scan-based core, it means a single scan-chain. The upper bound of the degree of partitioning is defined by the maximal test power consumption (Figure 9):

$$n_i = p_{max}/p_{test}(c_i) \qquad 9$$

By using the upper bound as the degree of partitioning in combination with Equation 8, the following is obtained:

$$\sum_{\forall i} \frac{t_{test}(c_i)}{n_i} = \left\{ n_i \rightarrow \frac{p_{max}}{p_{test}(c_i)} \right\} =$$

$$\sum_{\forall i} \frac{t_{test}(c_i) \times p_{test}(c_i)}{p_{max}} = t_{opt} \qquad 10$$

5.2 Optimal Power Constrained Test Schedule

The optimal test scheduling algorithm is illustrated in Figure 10. The time τ determines when a test is to start and it is initially set to zero. In each iteration over the set of cores, the degree of partitioning n_i is computed for the test at c_i; its new test time is calculated; and the starting time for the test is set to τ. Finally τ is increased by $t_{test}(c_i)/n_i$.

The algorithm consists of a loop over the set of cores resulting in a complexity of $O(|C|)$ where $|C|$ is the number of cores.

5.3 Practical limitations

The optimal degree of partitioning for a test at a core c_i has been defined as $p_{max}/p_{test}(c_i)$ (Equation 9). However, such

$\tau = 0$;
for all cores c_i
$n_i = p_{max} / p_{test}(c_i)$
start test at c_i *at time* τ;
$\tau = \tau + t_{test}(c_i)/n_i$;

Figure 10. Optimal scan-chain partitioning algorithm.

partitioning does not usually give an integer result. For instance, assume a system with a maximal test power consumption as p_{max}=10 and the test power for a test at a scan-based core c_i as $p_{test}(c_i)$ = 4. In this case n_i = 2.5. However, the number of scan-chains in a core can not be 2.5. In practice, n_i should be rounded down, in this case into 2 (rounding up to 3 leads to a test power of 12, which is bigger than p_{max}). The practical degree of partitioning for a test at c_i is given by:

$$n_i = \lfloor p_{max} / p_{test}(c_i) \rfloor \qquad 11$$

For the test at c_i, the difference between the optimal and the practical degree of partitioning is given by:

$$P_{max} = p_{test}(c_i) \times n_i + \Delta_i \qquad 12$$

and the difference Δ_i (see Figure 9) for each test at c_i is:

$$\Delta_i = p_{test}(c_i) \times n_i - p_{test}(c_i) \times \lfloor n_i \rfloor = p_{test}(c_i) \times (n_i - \lfloor n_i \rfloor) \qquad 13$$

Δ_i reaches its maximum when n_i-$\lfloor n_i \rfloor$ is approximately 1 which occur when n_i = 0.99... leading to $\Delta_i \approx p_{test}(c_i)$. The worst case test time occurs when $\Delta_i \approx p_{test}(c_i)$ for tests at all c_i and n_i = 1, resulting in a test time given by Equation 8 which is equal to $t_{sequence}$ computed using Eq. 7 since n_i = 1.

We now show the difference between the worst case test time for the system and its optimal test time. The worst case occurred when $\Delta_i = p_{test}(c_i)$ and n_i= 0.99... which in Equation 13 results in the following:

$$P_{max} = p_{test}(c_i) + p_{test}(c_i) \qquad 14$$

which only has one solution, $p_{test}(c_i)=P_{max}/2$ (assuming $P_{max}>p_{test}(c_i)>0$). However, we can not make any conclusions in respect to test time since the tests at two cores c_i and c_j may have equal test power consumption but different test time. The difference between the optimal test time and the worst total test time given by:

$$\sum_{\forall i} t_{test}(c_i) - \sum_{\forall i} \frac{t_{test}(c_i)}{2} = \sum_{\forall i} \frac{t_{test}(c_i)}{2} \qquad 15$$

This motivates the use of an integrated test scheduling and scan chain partitioning approach. Furthermore, scan-chain partitioning could have bandwidth limitations and it is also not feasible to assume that the length of each number chains at a core can be close to the number of flip-flops which would mean a scan-chain per flip flop. These limitations are investigated in the following section.

6 Optimal Test Access Mechanism Scheduling

In this section we demonstrate that TAM scheduling can achieve optimal test time. We investigate practical limitations and analyse test set pipe-lining.

6.1 Optimal Test Time

We assume a given system to be modelled as described in Section 3.4 where each the test at a core has a test time and a required test bandwidth attached to it. This can be illustrated using a rectangle for each test set (as shown in Figure 11) where the horizontal side corresponds to its test time while the vertical side corresponds to its test bandwidth.

A test schedule can be illustrated by placing all tests in a diagram as in Figure 11. At any moment we can not schedule tests requiring more test bandwidth than the maximal bandwidth of the TAM.

If the rectangle defined by $N_{tam} \times t_{total}$ is equal to the summation of $t_{test}(c_i) \times tam_{test}(c_i)$ for all tests, as given by the following equation, we have the optimal solution.

$$\sum_{\forall i} t_{test}(c_i) \times tam_{test}(c_i) = N_{tam} \times t_{opt} \qquad 16$$

The optimal test time for a system under test is thus:

$$t_{opt} = \sum_{\forall i} \frac{t_{test}(c_i) \times tam_{test}(c_i)}{N_{tam}} \qquad 17$$

Usually, the optimal test time cannot be achieved due to test conflicts. The worst case occurs when all tests are in conflicts with each other and all tests must be scheduled in sequence. The total test time is then given by:

$$t_{sequence} = \sum_{\forall i} t_{test}(c_i) \qquad 18$$

For a scan-based design the scan-chains can be divided into several which reduces the test application time. If every test t_i is allowed to be partitioned by a factor n_i, the total test time when all tests are scheduled in sequence is:

$$\sum_{\forall i} t_{test}(c_i)/n_i \qquad 19$$

The lower bound of the degree of partitioning is n_i = 1. For a scan-based core, it means a single scan-chain. The

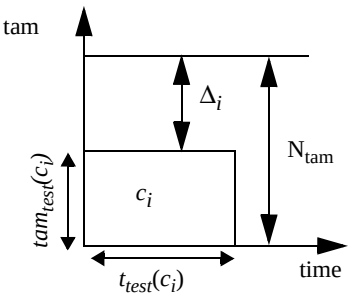


Figure 11. Illustration of the test time and the tam requirement for a test at a core c_i.

upper bound of the degree of partitioning is defined by the maximal TAM bandwidth:

$$n_i = N_{tam} / tam_{test}(c_i) \qquad 20$$

By using the upper bound as the degree of partitioning in combination with Equation 8, the following is obtained:

$$\sum_{\forall i} \frac{t_{test}(c_i)}{n_i} = \left\{ n_i \rightarrow \frac{N_{tam}}{tam_{test}(c_i)} \right\} =$$

$$\sum_{\forall i} \frac{t_{test}(c_i) \times p_{test}(c_i)}{N_{tam}} = t_{opt} \qquad 21$$

The above equation indicates the possibility to obtain optimal test time by scan chain partitioning, in theory.

6.2 Optimal TAM Schedule

The optimal test scheduling algorithm is illustrated in Figure 12. The time τ determines when a test is to start and it is initially set to zero. In each iteration over the set of cores, the degree of partitioning n_i is computed for the tests at c_i; its new test time is calculated; and the starting time for the test is set to τ. Finally τ is increased by $t_{test}(c_i)/n_i$.

The algorithm consists of a loop over the set of cores resulting in a complexity of $O(|C|)$ where $|C|$ is the number of cores.

```
τ = 0;
for all cores c_i
    n_i = N_tam / tam_test(c_i
    start test at c_i at time τ;
    τ=τ+t_test(c_i)/n_i;
```

Figure 12. Optimal scan-chain partitioning algorithm.

6.3 Practical limitations

The optimal degree of partitioning for a test at c_i has been defined as $N_{tam}/tam_{test}(c_i)$ (Equation 20). However, such partitioning does not usually give an integer result. For instance, assume a system with a maximal TAM as N_{tam} = 10 and the test bandwidth for a test c_i at a scan-based core as $tam_{test}(t_i) = 4$. In this case $n_i = 2.5$. However, the number of scan-chains in a core can not be 2.5. In practice, n_i should be rounded down, in this case into 2 (rounding up to 3 leads to a test power of 12, which is bigger than N_{tam}). The practical degree of partitioning for a test c_i is given by:

$$n_i = \lfloor N_{tam} / tam_{test}(c_i) \rfloor \qquad 22$$

For each test c_i, the difference between the optimal and the practical degree of partitioning is given by:

$$N_{tam} = tam_{test}(c_i) \times n_i + \Delta_i \qquad 23$$

and the difference Δ_{ij} for each test t_{ij} is given by:

$$\Delta_i = tam_{test}(c_i) \times n_i - tam_{test}(t_i) \times \lfloor n_i \rfloor = tam_{test}(c_i) \times (n_i - \lfloor n_i \rfloor) \qquad 24$$

Δ_i reaches its maximum when n_i-$\lfloor n_i \rfloor$ is approximately 1 which occur when n_i= 0.99.. leading to $\Delta_i \approx tam_{test}(c_i)$. The worst case test time occurs when $\Delta_i \approx tam_{test}(t_i)$ for all test c_i and $n_i = 1$, resulting in a test time given by Equation 19 which is equal to $t_{sequence}$ computed using Equation 18 since $n_i = 1$.

We now show the difference between the worst case test time for the system and its optimal test time. The worst case occurred when $\Delta_i = tam_{test}(c_i)$ and n_i= 0.99... which in Equation 24 results in the following:

$$N_{tam} = tam_{test}(c_i) + tam_{test}(c_i) \qquad 25$$

which only has one solution, $tam_{test}(p_i)=N_{tam}/2$ (assuming $0<tam_{test}(c_i)\leq N_{tam}$). However, we can not make any conclusions in respect to test time since two test c_i and c_j may have equal test bandwidth but different test time. The difference between the optimal test time and the worst total test time given by:

$$\sum_{\forall i} t_{test}(c_i) - \sum_{\forall i} \frac{t_{test}(c_i)}{2} = \sum_{\forall i} \frac{t_{test}(c_i)}{2} \qquad 26$$

This motivates the use of an integrated test scheduling and test partitioning approach.

Furthermore, the bandwidth at a core (number of possible number of scan chains) may put a limit and $N_{tam}/tam_{test}(c_i)$ can not be achieved.

6.4 Test Set Pipelining

In order to minimize the test time in scan based cores, the new test vector is shifted in at the same time as the previous test response is shifted out (see Section 2.2). Pipelining could be used between cores and not only within cores. It means that while the last test response of a core c_i is shifted out, the new test vector of core c_j can be shifted in.

We make two observation:

- The order cores are tested in may impose a conflict on TAM usage, and
- the test time gain using test set pipelining between cores is limited.

In distributed architecture test set pipelines between cores can not be used since each core is given its dedicated number of TAM wires. However it can be used in multiplexing architecture, where each core is given all TAM wires when it is tested.

The maximal time reduction is given by $min(ff_i/n_i, ff_j/n_j)$ and in relation to the total number of test vectors at the cores its impact is limited.

When using a TAM for the test data transportation, the order of the cores are also important (Figure 13). If core c_j is tested after c_i, there is a conflict. The part of the wires

between c_i and c_j should then both transport a vector to c_j and test response from c_i. In the multiplexing architecture this is not a problem since the other of tests can be determined afterwards. However, in the general case, it is not trivial. In this paper, we do not consider the test set pipelining between cores.

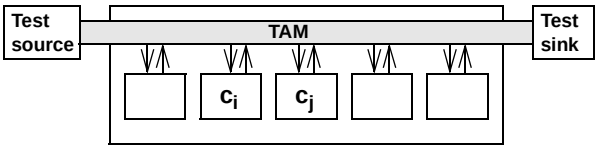


Figure 13. Pipelining and TAM conflicts.

7 Proposed Approach

In this section, we model the TAM scheduling problem under power constraint as an Knapsack problem. We describe our power constrained preemptive TAM scheduling algorithm and discuss the support for systems with both on-chip and off-chip resources.

7.1 The Problem

The Knapsack problem is where a set of n items each with a cost c_i and a weight w_i are to be selected to be included in a knapsack bounded by W minimizing the total cost C. The selection problem can be discrete (0-1), where an object is either in or out, or continuous (fractional) where the items can be divided. The discrete selection (0-1) makes the problem is NP-complete but not if it is fractional [18].

For the test set at a core, we can make transformations to change the characteristics of its test time, power consumption and TAM usage. For the TAM assignment we have Equation 2:

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

and for power consumption we have Equation 3:

$$p_{test}(c_i) = tp_i \times n_i$$

The number of assigned TAM wires can not exceeds N_{tam} and the power consumption can at any moment not be above P_{max}.

The assignment of TAM wires for a core may not always result in an integer result (also discussed in Section 6.3):

$$\Delta_i = \left\lceil \frac{ff_i}{n_i} \right\rceil + \frac{ff_i}{n_i} \qquad 27$$

and we may also have a limitation of lower limit on the length of the scan chains. Since the transformations do not always result in an integer result, we can not guarantee fractional assignment and therefore the problem is NP-complete.

The basic idea in preemptive scheduling is that a task or job can run for a period of time and then it is interrupted and resumed later. For preemptive test scheduling it means that all test vectors in a test set are not applied as a single set but as several tests all starting at different time. Naturally, all test vectors must be applied in order to finish the schedule. An example illustrating preemption based scheduling is in Figure 1(c) where the test 2 is split into two partitions, 2a and 2b.

To support preemption, 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,
tp_{ij} ($=tp_i * tam_{ij}$): test power consumed when active.

For each core we have a set of test vectors which take time (cost) to apply requiring TAM wires (weight) and a power consumption (weight), illustrated in Figure 14.

An example is in Figure 15, where 3 scan chain partitions 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,
- select the number of test vectors in each partition,
- determine the number of scan-chain for each partition,
- determine a start time for each of the partitions.

7.2 The Preemptive TAM Scheduling Algorithm

Our preemptive TAM scheduling algorithm is outlined in Figure 16. The objective is to minimize the total test time while satisfying all constraints and the power limitation. The idea is to assign TAM wires to the cores in each session such that Equation 27 is minimized. The algorithm starts by trying to find sessions with a single core to fully utilize the TAM. The number of cores in a session is increased until

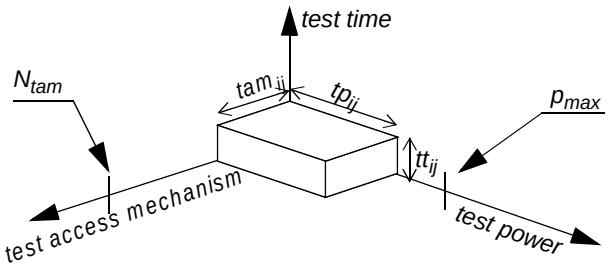


Figure 14. A three dimensional view of the problem.

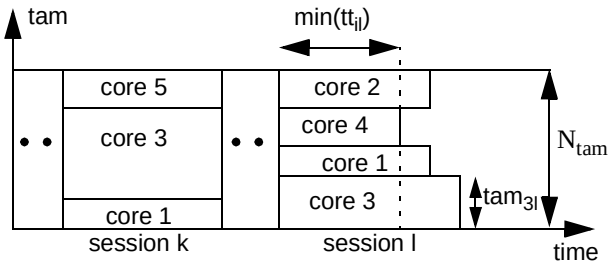


Figure 15. Session length based on preemption.

```
j=0 /session number /
Δ=0
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    / increase to a new session /
      end
   end
   Δ=Δ+δ   / increase allowed fault /
end
```

Figure 16. Preemptive scan partitioning algorithm.

|C|. If not all vectors are scheduled, the allowed fault increases (δ) and the algorithm restarts. The algorithm terminates when all test vectors are scheduled.

In Figure 15 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 Figure 15.

Figure 15 also illustrates that each core can be assigned to a different number TAM wires when its test vectors are split up into several sessions, *i.e* tam_{3k} is not equal to tam_{3l}. We outlined in Section 3.2 a wrapper design to support different number of scan chains and length of scan chains for all types of cores (soft, firm and hard).

For a given core c_i where the test vectors are split up into sc_{i1} and sc_{i2} the degree of partitioning calculated as n_{i1} and n_{i2}. It means that the number of scan-chains at c_i should, when test sc_{i1} is applied, be n_{i1} and, when sc_{i2} is applied, n_{i2}. For instance if n_{i1}=10 and n_{i2}=15 the number of scan-chains are given by 2×5×3=30 which is *least common multiplier* (*lcm*). This means that we also generalize our solution to make it applicable to an arbitrary number of tests per testable core.

7.3 On-chip and Off-chip Test Sets

We have for each core considered a set of test vectors of which origin we did not specify. We only assumed that all test vectors required a mean of transportation (part of the TAM). However, BIST (Built-In Self-Test) does not always require TAM access since each core can have its dedicated BIST structure. In these cases, we only have to keep control of the test power budget and if there are sharing among the BIST tests. For the algorithm, it only means that when a session is created, we also search among the BIST tests to see if any can be added in the session without violating constraints.

8 Experimental Results

We have made a comparison between the multiplexing architecture [6], the distributed architecture [6] and our proposed technique (Figure 16). We implemented the three approaches and made experiments using the IC benchmark, Table 1,[6] where we let the TAM width be in the range from 8 to 96 in steps of 8 with the limitation of not allowing scan chains including less than 20 scan flip flops. We used a SunBlade 1000, 900 MHz with 1024 Mb RAM for the experiments and all results are collected in Table 2.

There is no result for the distributed architecture when the TAM is 8 since the technique can not be used when the TAM size is less than the number of cores, i.e N_{tam}<|*C*|. For all TAM bandwidths our technique produces the solution closest to the lower bound. In average our technique is 2.1% from the lower bound while the distributed architecture is 19.1% from it and multiplexing architecture is 12.8% from it. The computational cost of the multiplexing architecture and the distribution architecture are extremely low, on average only 1 second. However, our technique produces results on average after 30.1 seconds which also is low.

9 Conclusions

In this paper, we have investigated TAM scheduling for scan tested core-based systems under power constraint. We have demonstrated that optimal test time can be achieved both for TAM scheduling and for scheduling under power constraints. We have investigated practical limitations and we have outlined a wrapper design allowing flexible scan chain lengths for all types of cores (soft, firm and hard). We model the problem as a Knapsack problem and we have proposed and implemented an algorithm which we have used to make a comparison between with previously proposed approaches.

References

[1] 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.

[2] Y. Zorian, “A distributed BIST control scheme for complex VLSI devices”, *Proceedings of IEEE VLSI Test Symposium (VTS)*, pp. 4-9, Atlantic City, NJ, April 1993.

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

[4] V. Muresan, X. Wang, V. Muresan, and M. Vladutiu, "A Comparison of Classical Scheduling Approaches in Power-Constrained Block-Test Scheduling", *Proceedings of IEEE International Test Conference (ITC)*, pp. 882-891, Atlantic City, NJ, October 2000.

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

[6] J. Aerts and E. J. Marinissen, "Scan Chain Design for Test Time Reduction in Core-Based ICs", *Proceedings of IEEE International Test Conference (ITC)*, pp. 448-457, Washington, DC, October 1998.

[7] S. Gerstendörfer and H-J Wunderlich, "Minimized Power Consumption for Scan-Based BIST", *Proceedings of IEEE International Test Conference (ITC)*, pp. 77-84, Atlantic City, NJ, September 1999.

[8] N. Nicolici and B. M. Al-Hashimi, "Power Conscious Test Synthesis and Scheduling for BIST RTL Data Paths", *Proceedings of IEEE International Test Conference (ITC)*, pp. 662-671, Atlantic City, NJ, October 2000.

[9] 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.

[10] E. Larsson and Z. Peng, "The Design and Optimization of SOC Test Solutions", *Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 523-530, San Jose, CA, November 2001.

[11] E. Larsson and Z. Peng, "Test Scheduling and Scan-Chain Division Under Power Constraint", *Proceedings of IEEE Asian Test Symposium (ATS)*, pp. 259-264, Kyoto, Japan, November 2001.

[12] V. Iyengar, K. Chakrabarty and E. J. Marinissen, "Test Wrapper and Test Access Mechanism Co-Optimization for System-on-Chip", *Proceedings of IEEE International Test Conference (ITC)*, pp. 1023-1032, Baltimore, MD, November 2002.

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

[14] N. Weste and K. Eshraghian, Principles of CMOS VLSI Design, *Addison-Wesley*, ISBN 0-201-53376-6, 1993.

[15] 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.

[16] E. J. Marinissen, Y. Zorian, R. Kapur, T. Taylor, and L. Whetsel, "Towards a Standard for Embedded Core Test: An Example", *Proceedings of IEEE International Test Conference (ITC)*, pp. 616-627, Atlantic City, NJ, September 1999.

[17] E. J. Marinissen, S. K. Goel, and M. Lousberg, "Wrapper Design for Embedded Core Test", *Proceedings of IEEE International Test Conference (ITC)*, pp. 911-920, Atlantic City, NJ, October 2000.

[18] T. Cormen, C. Leiserson, and R. Rivest, "Introduction To Algorithms", *The MIT Press*, ISBN 0-262-03141-8, 1989.

TAM width, N_{tam}	Distribution Architecture			Multiplexing Architecture			Our Technique			Lower bound
	Test time	Difference to lower bound (%)	CPU (s)	Test time	Difference to lower bound (%)	CPU (s)	Test time	Difference to lower bound (%)	CPU (s)	Test time
8	-	-	-	2067728	0.6	<1	2065604	0.5	1	2056000
16	1652600	60.8	<1	1044561	1.6	<1	1036493	0.8	1	1028000
24	945758	38.0	<1	705912	3.0	<1	692518	1.0	1	685333
32	661700	28.7	<1	536037	4.3	<1	521015	1.4	1	514000
40	478934	16.5	<1	435135	5.8	<1	418127	1.7	7	411200
48	389753	13.7	<1	374976	9.4	<1	348772	1.8	21	342666
56	321200	9.4	<1	330332	12.5	<1	298977	1.8	1	293714
64	288461	12.2	<1	296599	15.4	<1	262958	2.3	1	257000
72	251250	10.0	<1	271274	18.7	<1	237348	3.9	42	228444
80	221300	7.6	<1	251555	22.4	<1	211398	2.8	1	205600
88	201200	7.6	<1	238943	27.8	<1	194506	4.1	106	186909
96	181100	5.7	<1	227432	32.7	<1	176640	3.1	187	171333
Average:		19.1	1		12.8	1		2.1	30.1	

Table 2. Test time comparison of Multiplexing Architecture, Distribution Architecture and our technique.