

Power Constrained and Defect-Probability Driven SoC Test Scheduling with Test Set Partitioning

Zhiyuan He, Zebo Peng, and Petru Eles
Embedded Systems Laboratory (ESLAB)
Linköping University, Sweden
{zhihe, zebpe, petel}@ida.liu.se

Abstract¹

This paper presents a test scheduling approach for system-on-chip production tests with peak-power constraints. An abort-on-first-fail test approach is assumed, whereby the test is terminated as soon as the first fault is detected. Defect probabilities of individual cores are used to guide the test scheduling and the peak-power constraint is considered in order to limit the test concurrency. Test set partitioning is used to divide a test set into several test sequences so that they can be tightly packed into the two-dimensional space of power and time. The partitioning of test sets is integrated into the test scheduling process. A heuristic has been developed to find an efficient test schedule which leads to reduced expected test time. Experimental results have shown the efficiency of the proposed test scheduling approach.

1. Introduction and Related Work

Increasing requirements of advanced applications have driven electronic systems to become more and more complex, consequently leading to longer time-to-market and higher production cost. In recent years, a system-on-chip (SoC) approach to the design of such complex systems has been proposed, which integrates pre-designed and pre-verified blocks, referred to as intellectual property (IP) cores, into a single die. Although the SoC design technique has decreased the design complexity and shortened the design time, it has posed great challenges to the testing of core-based systems [1]. A major problem is how to efficiently generate, transport and apply large quantities of test data to the cores.

One effective solution to reduce the testing cost is to shorten the test application time by employing advanced test scheduling techniques. In order to test IP cores in a SoC, a set of test resources, such as test pattern sources and test sinks as well as a test access mechanism (TAM), have to be available [2]. Depending on various TAM implementations, tests can be applied in parallel or sequentially. It is known that increasing the test concurrency can significantly reduce the test application time and this can be achieved through elaborate test schedules.

SoC test scheduling has been studied recently and some solutions have been proposed [3, 4]. Most of them assumed that the tests run to their completion. However, in production tests, an abort-on-first-fail (AOFF) test approach is often used, which stops the test process as soon as a fault is detected, and discards the faulty chip directly. When using the AOFF test approach, the test application time can be reduced if faults are detected,

especially in the early stages of manufacturing when the yield is low and defects are more likely to appear.

In our approach, it is assumed that several cores can be tested concurrently by utilizing the BIST mechanism in the cores and/or the external test infrastructure. Increasing the number of concurrent tests can directly lead to the reduction of the test application time. However, to maximize the test concurrency can be impractical, because of either limited power supply or potential damage to chips. It is known that testing usually consumes more power than normal operations. Extremely high power consumption may cause the chip to be overheated and even burnt due to the high temperature. Therefore, a power constraint is introduced to limit the total number of concurrent tests, which is defined as the maximum total power consumption that the chip can endure. In a SoC test framework, this means that the sum of the powers consumed by concurrently applied tests should never exceed the power constraint.

The power-constrained test scheduling problem is similar to the classical two-dimensional rectangular packing problem [5, 6] which is NP-complete. In our case, a test sequence composed of a number of test patterns is considered as a rectangle, with the height corresponding to the maximum power consumption of the test patterns and the width the time duration of the test sequence.

In a production test environment, defect probabilities of individual cores can be utilized for test scheduling in order to minimize the test time [7-11]. Those defect probabilities can be derived from the statistical analysis of the production process or generated based on inductive fault analysis. In [9], Ingelsson et al. proposed a test scheduling approach for external tests using defect probabilities, which aims to minimize the expected test time. In their work, test data are transported through TAM wires dedicated to every individual core. The widths of the TAM wires have a significant impact on the expected test time, and therefore are taken into account for the test scheduling. Pass probabilities of the individual test patterns are calculated through an assumed probability distribution function.

In this paper, we present a defect-probability driven test scheduling approach integrated with test set partitioning to minimize the test application time. The differences between our work and [9] are the following. (1) We have considered the power consumption issue and introduced a power constraint to limit the test concurrency. (2) A generic test architecture has been assumed, where built-in self-tests (BISTs) and/or external tests are utilized. (3) We have integrated the test set partitioning in the test scheduling in order to improve the efficiency of the generated test schedules. (4) The probabilities that the test process is terminated at different time moments are calculated based on the results of fault simulation, and hence are more accurate.

In our previous work [11], we presented a test scheduling approach for test time minimization considering the power

¹ This work has been partially supported by the Swedish Foundation for Strategic Research (SSF) under the Strategic Integrated Electronic Systems Research (STRINGENT) program.

constraint and defect probabilities of cores. A heuristic was proposed for test scheduling. The main contribution of this paper is the development of a test set partitioning method to divide a test set into several test sequences so that they can be packed into a tight schedule. This test set partitioning method is also integrated into the test scheduling approach, so that the efficiency of the test schedule is improved and the test application time can be further reduced. We have also used a test pattern reordering technique proposed in [12] as a pre-processing for the test set partitioning, which can reduce the power consumption of test patterns and make the power profiles smoother. For the test set partitioning, we have developed a heuristic to find an appropriate number of partitions such that the sum of the area sizes (the peak-power consumption multiplied by the time duration) of all the partitions is as small as possible.

The rest of this paper is organized as follows. The next section presents the assumed test architecture. In Section 3, a motivational example is given to illustrate the power-constrained test scheduling problem. Section 4 gives the problem formulation and Section 5 presents the test set partitioning approach. In Section 6, some basic partitioning and scheduling principles are presented and the proposed heuristic for test scheduling is demonstrated. Experimental results are shown in Section 7 and the paper is concluded in Section 8.

2. Test Architecture

We have assumed a BIST architecture which is extended with support to external tests. The cores are equipped with dedicated BIST logics and a test bus can be used to transport deterministic test patterns from external sources like an ATE or an on-chip memory. Such a hybrid BIST architecture [13, 14] can utilize the advantages of both pseudorandom and deterministically generated tests.

Figure 1 gives an example of the assumed test architecture for a five-core system. In this example, an embedded tester consisting of a test controller and a tester memory is integrated on the chip. Deterministic test patterns are applied from the tester memory to one core at a time. All the cores have their dedicated BIST logics that can apply pseudorandom tests concurrently. We have also assumed that a test bus connecting all cores, such as the advanced microcontroller bus architecture (AMBA) [15], is used for test data transportation.

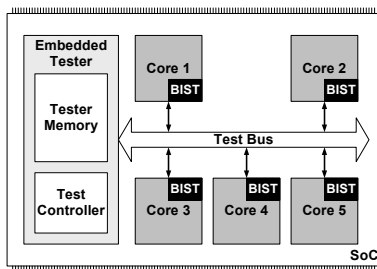


Figure 1. An example of the assumed test architecture

The assumed test architecture provides flexibility for system integrators to use our approach for different purposes. For example, in order to reduce the silicon area size, the on-chip tester can be substituted by an external ATE, or it can be eliminated if only BIST is required.

Suppose that a system S , consisting of n cores C_1, C_2, \dots, C_n , has the test architecture depicted in Figure 1. For every individual core C_i ($i = 1, 2, \dots, n$), a defect probability $DP(C_i)$, defined as the probability that the core has defects, is given. In order to test core C_i , a deterministic test set DT_i consisting of d_i ($d_i \geq 0$)

deterministic test patterns and a pseudorandom test set consisting of r_i ($r_i \geq 0$) pseudorandom test patterns are applied to C_i , where $d_i + r_i > 0$. Further more, a test set can be partitioned into a number of test sequences. Suppose that deterministic test set DT_i is partitioned into a_i ($0 \leq a_i \leq d_i$) deterministic test sequences, and pseudorandom test set PR_i is partitioned into b_i ($0 \leq b_i \leq r_i$) pseudorandom test sequences, where $a_i + b_i > 0$. With DT_{ij} ($j = 1, 2, \dots, a_i$) we denote the j -th deterministic test sequence for core C_i , and with PR_{ik} ($k = 1, 2, \dots, b_i$) the k -th pseudorandom test sequence for core C_i . Note that a test set is also a test sequence which originally has one partition ($a_i = d_i = 1$ and/or $b_i = r_i = 1$), so the term “test sequence” is used to indicate a test set as well, if not mentioned otherwise.

3. Motivational Example

The power consumption of a test pattern is proportional to the total amount of switching activities between the precedent test pattern and itself, which equals to the number of state transitions at all the primary inputs and outputs plus the number of state transitions at all the internal nodes in the circuit, during the application of the test pattern². The peak-power consumption of a test sequence is defined as the maximum power consumption of the test patterns in the test sequence. Figure 2 depicts the power profile of a test sequence and its peak-power consumption.

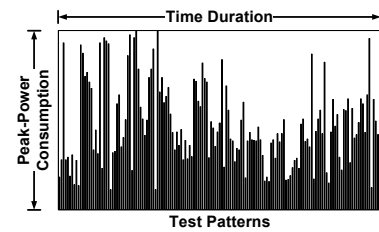


Figure 2. Peak-power consumption of a test sequence

Different test schedules can lead to different test application time. Figure 3(a) shows an example of a power-constrained test schedule for five deterministic test sequences DT_i ($i = 1, 2, \dots, 5$) and five pseudorandom test sequences PR_i ($i = 1, 2, \dots, 5$), illustrated with white and grey rectangles, respectively. Each test sequence is depicted as a rectangle with a height and a width corresponding to the peak-power consumption and the time duration of the test sequence, respectively. The area size of a test sequence is then equal to its peak-power consumption multiplied by the time duration. The peak-power constraint is denoted with POW_c . Note that test sequences belonging to the same core, like DT_1 and PR_1 , cannot be scheduled concurrently due to the test conflict.

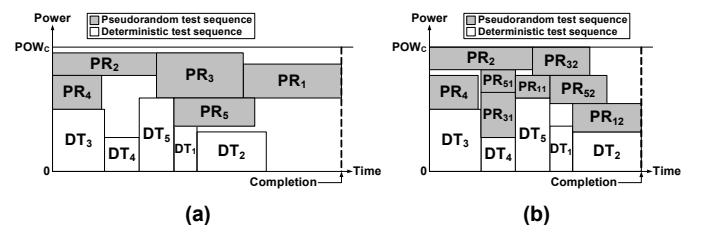


Figure 3. Power-constrained test schedule examples

Comparing the size of the effective scheduled area occupied by all test sequences to the size of the overall schedulable area restricted by the power constraint line and completion time line,

² We have assumed that a low-power scan chain technique is used [16] when a test-per-scan approach is employed.

one can find that the efficiency of the test schedule in Figure 3(a) is low since much of the space is wasted. One solution to improve the efficiency of the test schedule is to employ test set partitioning to decrease the granularities of test sequences. As shown in Figure 3(b), PR_1 , PR_3 , and PR_5 are partitioned into PR_{11} and PR_{12} , PR_{31} and PR_{32} , and PR_{51} and PR_{52} , respectively. The partitioned test sequences have shorter time duration and/or smaller peak-power consumption than the non-partitioned test sequence, thus can be scheduled at the time moments which were not possible for the non-partitioned test sequence due to its large area size. From the above example, it can be observed that using test set partitioning can significantly improve the efficiency of the test schedule and shorten the test time.

4. Problem Formulation

With the AOFF test approach, the test application time depends on two factors, one is the elapsed test time when a fault is detected, and the other is the probability to detect the fault. Given the defect probabilities of individual cores and the result of fault simulation, we can calculate the incremental fault coverage of every single test pattern, and the probability that the test aborts at every possible termination time moment [11]. Further, the expected total test time (ETTT), defined as the mathematical expectation of the test application time, is computed based on the calculated probabilities and used as a cost function for test time minimization.

A deterministic test can abort at the end of any deterministic test pattern, since the test response is available for every single test pattern. A pseudorandom test, on the other hand, can only abort at the end of the entire test sequence, when the signature is available. Here we define a possible test termination moment (PTTM) as the time moment when a deterministic test pattern or a pseudorandom test sequence has been applied, and the test response or signature has been analyzed. For every PTTM, the elapsed time is known, and we can calculate the probability that the test aborts at this particular time moment. Figure 4 illustrates all the possible test termination moments in a test schedule, where the dotted lines stand for the ending moments of deterministic test patterns, and the dashed lines indicate the ending moments of pseudorandom test sequences. Note that some of these moments overlap and therefore are treated as identical PTTMs.

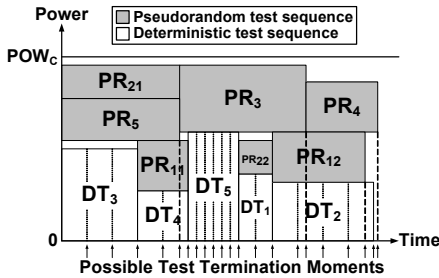


Figure 4. Illustration of possible test termination moments

A generic formula for the ETTT calculation is given in Equation 1 which is presented as a sum of two literals. The first literal corresponds to the test abortion case where tests are terminated because faults are detected. The second literal corresponds to the test completion case where all tests are passed to their completion. For the test abortion case, at every possible test termination moment $x \in X$, we calculate the test abortion probability $p(A_x)$ and the elapsed test time t_x , where A_x denotes the random event that the test has been aborted at PTTM x . Similarly, for the test completion case, we calculate the test completion

probability $p(E)$ and the test completion time l , where E denotes the random event that all tests are carried out to their completion without detecting any fault. More detailed explanations of the ETTT calculation can be found in [11].

$$ETTT = \sum_{x \in X} (t_x \times p(A_x)) + l \times p(E) \quad (1)$$

In order to minimize the test application time in production tests, we need to minimize the ETTT through efficient test scheduling integrated with test set partitioning. Taking into account the peak-power constraint, the test scheduling problem is similar to the classical two-dimensional rectangular packing problem [5, 6]. In this paper, our objective is to develop heuristics to find an efficient partitioning scheme and an efficient test schedule for all partitioned deterministic and pseudorandom test sequences, so that the ETTT is minimized while the power constraint is satisfied.

5. Test Set Partitioning

As in a rectangular packing problem, the sizes of test sequences have a large impact on the final schedule. To divide test sequences into smaller partitions with shorter time duration and lower individual peak-power consumptions will lead to more efficient test scheduling, since the partitioned test sequences have smaller granularities in terms of their area sizes and can be packed more tightly. Figure 5(a) shows a non-partitioned deterministic test sequence for core C_i and Figure 5(b) shows its three partitions (DT_{i1} , DT_{i2} , and DT_{i3}). In Figure 5(b), the individual peak-power consumptions of the first two partitions (DT_{i1} and DT_{i2}) are lower than that of the non-partitioned test sequence in Figure 5(a). The grey rectangles with dashed line edges illustrate the reduced area sizes due to the partitioning.

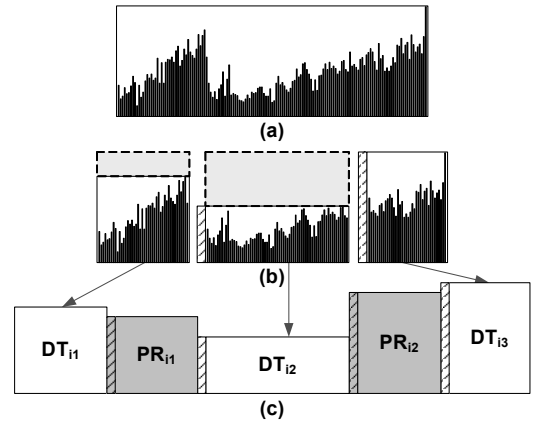


Figure 5. Illustration of test set partitioning and time overheads

Reordering test patterns is useful to reduce power consumption and can make the power profile of a test sequence relatively smooth and easy to manipulate [12]. Thus, for all deterministic tests, we have used test pattern reordering as a pre-processing for the test set partitioning. In Figure 6(a), the original

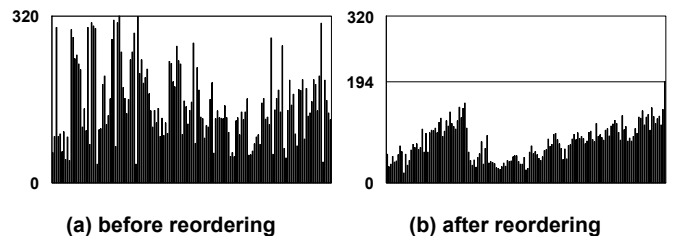


Figure 6. Power profiles before and after test pattern reordering

power profile of a deterministic test sequence is given. As a comparison, the power profile after test pattern reordering is shown in Figure 6(b). It can be seen that, through reordering the test patterns, the power profile is much smoother and the peak-power consumption is reduced (39% lower for this example).

Although test set partitioning can lead to smaller partitions, it can however introduce time overheads for the partitioned test sequences when a test-per-scan approach is employed. This phenomenon occurs when deterministic test sequences and pseudorandom test sequences belonging to the same core are interleaved, as in the example in Figure 5(c). There the three partitioned deterministic test sequences (DT_{i1} , DT_{i2} , and DT_{i3}) are interleaved with two partitioned pseudorandom test sequences (PR_{i1} and PR_{i2}) for the same core C_i . The time overheads are indicated by the rectangles filled with slashed lines and situated at the left of PR_{i1} , DT_{i2} , PR_{i2} , and DT_{i3} .

The time overheads are due to the following fact. When a deterministic (pseudorandom) test is stopped and resumed later after a pseudorandom (deterministic) test has been applied, the pipeline consisting of three operations (scan-in, application, and scan-out, see Figure 7(a)) is interrupted and has to be refilled at the beginning of the latter partition (see Figure 7(b)). Thus, the time overhead added to the latter partition is equal to the time duration of the scan-out operation, denoted with L_o in Figure 7.

In Figure 5(b), the rectangles in grey are the areas reduced from the non-partitioned test sequence, while the rectangles filled with slashed lines are the areas added. Thus, we proposed a heuristic to find an appropriate number of partitions for a deterministic test set, such that the sum of the area sizes of all the partitions is minimized.

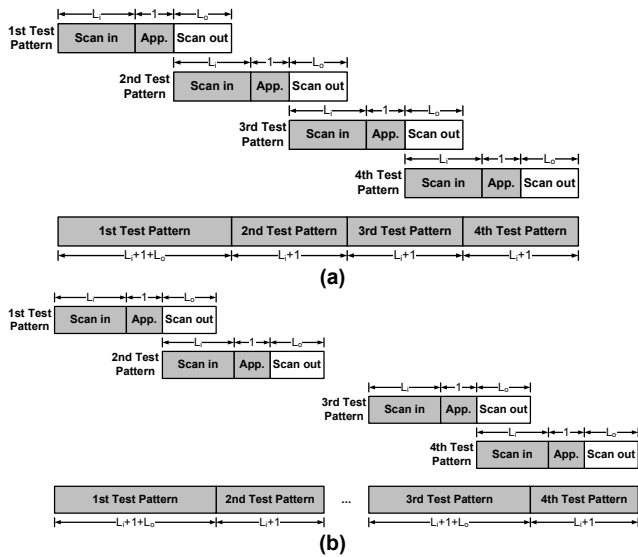


Figure 7. Pipeline in a test-per-scan approach

The heuristic for deterministic test set partitioning starts with the original non-partitioned test sequence. Within each iteration step, one of the existing partitions is divided into two test sequences. The heuristic stops when no more partitions can be added, which means that every partitioned test sequence has one and only one test pattern. Here the cost function is defined as the sum of the area sizes of all the partitioned test sequences, and the objective is to find a partitioning scheme which has the lowest cost among all the explored solutions.

At every iteration step, we have to decide which existing partition should be selected to be split into two test sequences, and at which position (test pattern) the selected partition should

be divided. With an exhaustive search among all possible solutions within this iteration step, the local optimal partitioning scheme with the lowest cost is obtained and one more partition is added. In the global range, among all the local optimal partitioning schemes with different number of partitions, the one with the lowest cost is acquired and accepted as the best solution. Figure 8 illustrates how the sum of the area sizes of all partitions distributed with different numbers of partitions. Usually the best partitioning scheme has a relatively small number of partitions in relation to the total number of test patterns in the test set. For example, in Figure 8, a test set with 149 test patterns should be divided into 21 partitions such that the sum of their area sizes is minimized.

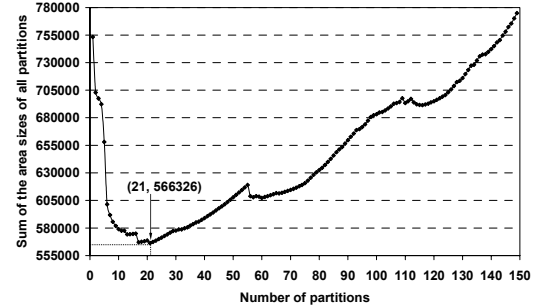


Figure 8. Sum of the area sizes for different number of partitions

When a pseudorandom test sequence is divided into two partitions, two signatures are needed in order to obtain the test results at the end of both partitions, which means that an additional signature should be generated. Thus, extra memory is also needed to store this additional fault-free signature, and an extra time slot is needed to analyze the additionally generated signature. In this paper, we have assumed that there exists sufficient memory to store the signatures and we ignore the extra time slots for the analysis of the additional signatures, since this time is very short, compared to the time duration of the pseudorandom test sequence. We do not consider the impact of the increased complexity of the test controller either in this work, even though this could be an interesting issue to take into account [17].

6. Proposed Heuristic for Test Scheduling

Before the heuristic for test scheduling is presented, some basic partitioning and scheduling principles are summarized as follows: (1) Test sequences belonging to the same core cannot be scheduled in parallel. (2) Deterministic test sequences are scheduled sequentially since a single test bus is used, while pseudorandom test sequences are scheduled in parallel under the peak-power constraint. (3) The scheduling of deterministic test sequences is performed before the scheduling of pseudorandom test sequences, which means that deterministic test sequences have higher scheduling priorities. This is because deterministic tests can be stopped after every test pattern, while pseudo-random tests can only be terminated at the end of the test sequences, when the signatures are ready. Additionally, deterministic test patterns are usually more efficient in detecting faults than pseudorandom test patterns. (4) Pseudorandom test sequences are first sorted in a decreasing order by certain parameters like the defect probability of a core, the peak-power consumption and time duration of a test sequence [11]. Thereafter, they are scheduled to the earliest time moment if available. Deterministic test sequences, however, are scheduled in the order obtained by a defect-probability driven heuristic.

Test set partitioning is integrated into the test scheduling approach in the following way. Deterministic test sets are partitioned statically, meaning that they are partitioned before being scheduled, with the heuristic demonstrated in Section 5. Pseudorandom test sets, on the other hand, are partitioned during the test scheduling. When it is impossible to schedule a pseudorandom test sequence to the earliest time moment due to its large area size, the test sequence is divided into two partitions such that the smaller one can be scheduled as expected, and the scheduling of the other one is performed later.

Based on the basic principles described above, a heuristic has been developed to find an efficient test schedule for all test sequences in an iterative way. One iteration step of the heuristic is illustrated with an example in Figure 9. Suppose we have five deterministic test sequences $DT_1, DT_{21}, DT_{22}, DT_{31},$ and DT_{32} , and three pseudorandom test sequences $PR_1, PR_2,$ and PR_3 . Two deterministic test sequences DT_{31} and DT_1 have already been scheduled. In this iteration step, we have to decide which one out of three unscheduled deterministic test sequences $DT_{21}, DT_{22},$ and DT_{32} should be scheduled to which time moment among $A, B,$ and C , as depicted in Figure 9. After a deterministic test sequence is scheduled to a time moment, the three pseudorandom test sequences $PR_1, PR_2,$ and PR_3 are scheduled to the rest of the space, and test set partitioning may be needed during the scheduling. Thereafter, the partial expected total test time (PETTT) is calculated within the time range of the scheduled deterministic test sequences (see Figure 10). When all the possible 9 solutions within the current iteration step have been explored, the solution with the lowest PETTT value is accepted and the three scheduled deterministic test sequences are taken as a base for the next iteration step. The heuristic stops when no more unscheduled deterministic test sequences are left, and the final test schedule is then obtained. Note that when a test sequence is scheduled, the order of those already scheduled test sequences should remain unchanged.

Figure 10 shows a solution that DT_{22} is scheduled to time moment B . During the scheduling of pseudorandom test sequences, PR_2 is partitioned into two test sequences PR_{21} and PR_{22} . The PETTT calculation range is from the beginning of DT_{31} till the end of DT_1 . The gap between PR_3 and PR_{22} shows that DT_{22} and PR_{22} cannot be scheduled concurrently due to the test conflict.

Formally, suppose that we have N deterministic test sequences all together, and m ($0 \leq m < N$) of them have been scheduled at a certain iteration step. We need to schedule one more deterministic test sequence selected from the set of $N - m$ unscheduled deterministic test sequences to an appropriate time moment, without disturbing the order of the scheduled test sequences. When a selected deterministic test sequence has been scheduled to a time moment, all the pseudorandom test sequences are then scheduled into the rest of the space, with application of dynamic partitioning, if needed. The PETTT of this solution is then calculated within the time range of the $m + 1$ scheduled deterministic test sequences. When all the $(N - m) \times (m + 1)$ possible solutions have been explored, the solution with the minimum PETTT value is accepted. The new list of scheduled deterministic test sequences is then used as a base for the next iteration step. Repeating this procedure from the initial state when $m = 0$ until all the deterministic and pseudorandom test sequences are scheduled when $m = N$, we get the final optimized schedule.

The pseudo-code of the heuristic, given in Figure 11, has three major embedded loops. The outer loop (line 1-19) increments the number of scheduled deterministic test sequences, the middle

loop (line 4-17) selects every unscheduled deterministic test sequence, and the inner loop (line 5-16) explores every possible time moment for scheduling. Inside the inner loop, the selected deterministic test sequence is scheduled (line 6), thereafter pseudorandom test sets are partitioned if needed and then scheduled (line 7-10). The PETTT of the present schedule is then calculated (line 11) and compared to the minimum PETTT for an acceptance decision (line 12-15). The final test schedule is output in the end (line 20).

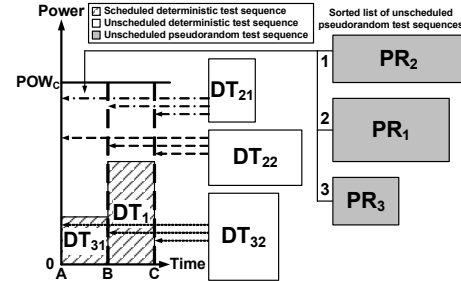


Figure 9. Illustration of one iteration step of the heuristic

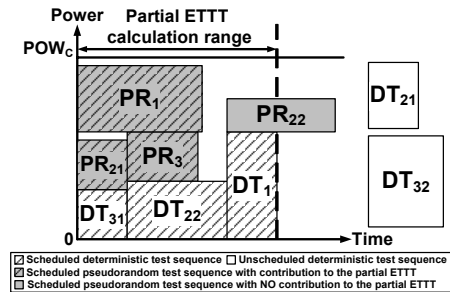


Figure 10. A solution in the iteration step

```

1 for (# of scheduled DT test sequences = 0 to N-1) do // Outer loop
2   Minimum_PETTT = a large value;
3   m = number of scheduled DT test sequences;
4   for (every unscheduled DT test sequence  $DT_{ij}$ ) do // Middle loop
5     for (every possible scheduling time moment  $T_x$ ) do // Inner loop
6       Schedule  $DT_{ij}$  to  $T_x$ ;
7       for (every PR test set  $PR_k$ ) do
8         Partition  $PR_k$  if needed;
9         Schedule  $PR_k$ ;
10      end for;
11      Current_PETTT = PETTT();
12      if (Current_PETTT < Minimum_PETTT) do
13        Minimum_PETTT = Current_PETTT;
14        Record the current solution as the best solution;
15      end if;
16    end for; // Inner loop stops here
17  end for; // Middle loop stops here
18  Accept the best solution for this iteration;
19 end for; // Outer loop stops here
20 Output the final test schedule;

```

Figure 11. Pseudo-code of the heuristic

7. Experimental Results

For the experiments, ISCAS'89 benchmarks were used and the test-per-scan approach was utilized. All cores were redesigned to insert one single scan chain, and the STUMPS architecture is used for BIST.

In the first set of experiments, test set partitioning was employed. We did experiments for 5 groups of designs. Each group had 5 different designs which had the same number of cores of different types, but the cores were assigned with different defect probabilities. The numbers of cores were 5, 10, 20, 30, and 50 for each group, respectively. For each design we used 3 different levels of peak-power constraints. The experimental

results in Table 1 were the average values from 15 experiments (5 different designs with the same number of cores multiplied by 3 different peak-power constraints). The defect probabilities of individual cores were generated randomly, while keeping the system defect probability at the value 0.6 (i.e. 40% system yield).

Table 1. Experimental results of different approaches with test set partitioning

Number of Cores	5		10		20		30		50	
	ETTT	CPU Time (s)	ETTT	CPU Time (s)	ETTT	CPU Time (s)	ETTT	CPU Time (s)	ETTT	CPU Time (s)
BLD Scheduling	7783	0.01	10590	0.02	20081	0.04	28578	0.06	50562	0.11
Our Heuristic	6247	2.5	7983	26.9	14239	293.9	21117	493.4	37463	4372.9
SA	6126	276.0	7732	568.7	14808	301.5	22290	503.9	40074	4409.3

In order to show the efficiency of our heuristic, a classical bottom-left-decreasing (BLD) scheduling algorithm is taken for comparison. It sorts deterministic and pseudorandom test sequences decreasingly by their area sizes (the peak-power consumption multiplied by the time duration), and then schedules them using the bottom-left strategy. As shown in Table 1, by employing our heuristic, the ETTT can be reduced around 20% to 29% compared to the BLD scheduling algorithm, with an acceptable increase of execution time. On the other hand, in order to show the accuracy of our heuristic to find a near-optimal test schedule, we also compared our heuristic with a simulated annealing (SA) algorithm. For small designs with 5 and 10 cores, the SA algorithm reached the imposed termination condition in an acceptable time and is supposed to return a solution close to the optimal solution. For large designs with 20, 30, and 50 cores, the SA algorithm took unacceptably long time to reach the termination condition. Thus, for these experiments, we let the SA algorithm run for a time equal to that needed by our heuristic. From Table 1, one can see that in small designs, the SA algorithm works just slightly better than our heuristic (2% to 3% lower ETTT), but has up to two orders of magnitude longer execution time than our heuristic. For the large designs, our heuristic found better solutions with 4% to 7% lower ETTT values, than the SA algorithm that reached in the same amount of time.

In the second set of experiments where the same designs were used, we intended to show the effect of test set partitioning. As a comparison, we used a defect-probability driven test scheduling heuristic which did not allow test set partitioning. For the sake of fairness, both the partitioned and non-partitioned heuristic used test pattern reordering, thus the advantage of the peak-power reduction by reordering test patterns did not play any role in this comparison. The experimental results are given in Table 2. As shown in the Table, using test set partitioning can reduce the ETTT with amounts between 16% and 30%. The results are also illustrated in Figure 12.

Table 2. Comparison of our heuristic with one without test set partitioning

Number of Cores	5		10		20		30		50	
	ETTT	CPU Time (s)	ETTT	CPU Time (s)	ETTT	CPU Time (s)	ETTT	CPU Time (s)	ETTT	CPU Time (s)
NON-Partitioned	8269	0.09	11357	0.86	18016	14.2	26710	68.6	44713	589.1
Partitioned	6247	2.5	7983	26.9	14239	293.9	21117	493.4	37463	4372.9

8. Conclusions

In this paper, a power-constrained SoC test scheduling approach is presented in a production test environment. Different from other approaches, the defect probabilities of individual cores are utilized to drive the test scheduling and a test set partitioning approach is employed. Based on the calculation of the ETTT, a heuristic for test set partitioning and test time minimization is

used to generate an efficient test schedules. Experimental results have shown that the proposed method is effective to shorten the test application time.

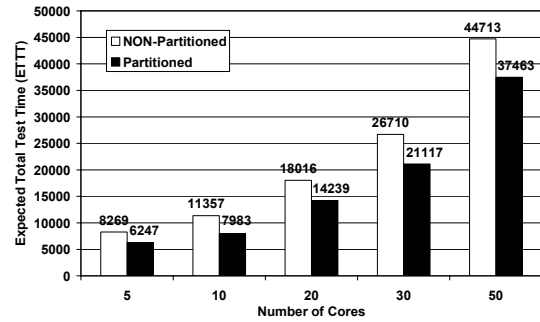


Figure 12. Comparison of our heuristic with one without partitioning

References

- [1] B. T. Murray, and J. P. Hayes. Testing ICs: Getting to the core of the problem. *IEEE Trans. on Computer*, Vol. 29, No. 11, 1996, pp. 32-38.
- [2] Y. Zorian, E. J. Marinissen, and S. Dey. Testing Embedded Core-Based System Chips. *Int. Test Conf.*, 1998, pp. 130-143.
- [3] 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. *Asian Test Symp.*, 2001, pp. 265-270.
- [4] E. Larsson, and Z. Peng. An Integrated Framework for the Design and Optimization of SOC Test Solutions. *J. of Electronic Testing: Theory and Applications*, Vol. 18, No. 4/5, 2002, pp. 385-400.
- [5] B. S. Baker, E. G. Coffman Jr., and R. L. Rivest. Orthogonal Packings in Two Dimensions. *SIAM J. of Computing*, Vol. 9, Issue 4, 1980, pp. 846-855.
- [6] N. Lesh, J. Marks, A. McMahon, and M. Mitzenmacher. Exhaustive Approaches to 2D Rectangular Perfect Packings, *Elsevier Information Processing Letters*, Vol. 90, Issue 1, 2004, pp. 7-14.
- [7] W. J. Jiang, and B. Vinnakota. Defect-Oriented Test Scheduling. *IEEE Trans. on VLSI Systems*, Vol. 9, No. 3, 2001, pp. 427-438.
- [8] E. Larsson, J. Pouget, and Z. Peng. Defect-Aware SOC Test Scheduling. *VLSI Test Symp.*, 2004, pp. 359-364.
- [9] U. Ingelsson, S. K. Goel, E. Larsson, and E. J. Marinissen. Test scheduling for modular SOCs in an abort-on-fail environment. *European Test Symp.*, 2005, pp. 8-13.
- [10] Z. He, G. Jervan, Z. Peng, and P. Eles. Hybrid BIST Test Scheduling Based on Defect Probabilities. *Asian Test Symp.*, 2004, pp. 230-235.
- [11] Z. He, G. Jervan, Z. Peng, and P. Eles. Power-Constrained Hybrid BIST Test Scheduling in an Abort-on-First-Fail Test Environment. *EUROMICRO Conf. on Digital System Design*, 2005, pp. 83-86.
- [12] P. M. Rosinger, B. M. Al-Hashimi, and N. Nicolici, Power Profile Manipulation: A New Approach for Reducing Test Application Time Under Power Constraints, *IEEE Trans. on CAD of Integrated Circuits and Systems*, Vol. 21, No. 10, 2002, pp. 1217-1225.
- [13] M. Sugihara, H. Date, and H. Yasuura. Analysis and Minimization of Test Time in a Combined BIST and External Test Approach. *Design, Automation and Test in Europe*, 2000, pp. 134-140.
- [14] G. Jervan, P. Eles, Z. Peng, R. Ubar, and M. Jenihhin. Test Time Minimization for Hybrid BIST of Core-Based Systems. *Asian Test Symp.*, 2003, pp. 318-323.
- [15] D. Flynn. AMBA: Enabling Reusable On-Chip Designs. *IEEE Micro*, Vol. 17, No. 4, 1997, pp. 20-27.
- [16] S. Gerstendorfer, and H. J. Wunderlich. Minimized power consumption for scan-based BIST. *Int. Test Conf.*, 1999, pp. 77-84.
- [17] S. K. Goel, and E. J. Marinissen. Control-Aware Test Architecture Design for Modular SOC Testing. *European Test Workshop*, 2003, pp. 57-62.