# Power-Aware Test Planning in the Early System-on-Chip Design Exploration Process

Erik Larsson, Member, IEEE, and Zebo Peng, Senior Member, IEEE

Abstract—Test application and test design, performed to ensure the production of fault-free chips, are becoming complicated and very expensive, especially in the case of SoCs (System-on-Chip), as the number of possible faults in a chip is increasing dramatically due to the technology development. It is therefore important to take test design into consideration as early as possible in the SoC design-flow in order to develop an efficient test solution. We propose a technique for modular core-based SoCs where test design is integrated in the early design exploration process. The technique can, in contrast to previous approaches, already be used in the core selection process to evaluate the impact on the system's final test solution imposed by different design decisions. The proposed technique considers the interdependent problems of core selection, test scheduling, TAM (test access mechanism) design, test set selection, and test resource floorplanning, and minimizes a weighted cost-function based on test time and TAM routing cost, while considering test conflicts and test power limitations. Concurrent scheduling of tests is used to minimize the test application time; however, concurrent test application leads to higher activity during the testing and, hence, higher power consumption. The power consumed during testing is, in general, higher than that during normal operation since it is desirable with hyperactivity in order to maximize the number of tested faults in a minimal time. A system under test can actually be damaged during testing and, therefore, power constraints must be considered. However, power consumption is complicated to model and, often, simplistic models that focus on the global system power limit only have been proposed and used. We therefore include a novel three-level power model: system, power-grid, and core. The advantage is that the system-level power budget is met and hot-spots can be avoided both at a specific core and at certain hot-spot areas in the chip. We have implemented and compared the proposed technique with a technique that assumes already fixed cores and tests, an estimation-based approach, and a computationally expensive pseudoexhaustive method. The results from the experiments show that, by exploring different design and test alternatives, the total test cost can be reduced, the pseudoexhaustive technique cannot produce results within reasonable computational time, and the estimation-based technique cannot produce solutions with high quality. The proposed technique produces results that are near the ones produced by the pseudoexhaustive technique at computational costs that are near the costs of the estimation-based technique, i.e., it produces high-quality solutions at low computational cost.

Index Terms—Test scheduling, test set selection, design exploration, TAM design, power consumption, hot-spots.

## **1** INTRODUCTION

T ECHNOLOGY development has made it possible to design a chip where the complete system is placed on a single die, a so-called system chip or SoC (System-on-Chip). The production of these systems may lead to faulty chips and it is therefore important that the produced chips are tested. The growing complexity of chips, device size miniaturization, increasing transistor count, and high clock frequencies have led to a dramatic increase in the number of possible fault sites and fault types and, therefore, a high test data volume is needed for high-quality testing. However, the high test data volume leads to long testing times and, therefore, the planning and organization of the testing becomes a challenge that has to be tackled.

EDA (Electronic Design Automation) tools are developed to reduce the design productivity gap, i.e., the gap between what technology allows to be designed and what a design team can produce within a reasonable time. A way to handle the increasing complexity of systems is to model

For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TCSI-0397-1204.

at higher abstraction levels means that fewer implementation specific details are visible. The problem is that device size miniaturization has made implementation specific details highly important. A modular core-based design approach has therefore been proposed to allow the design of complex systems in reasonable time and, at the same time, handle implementation specific details [15], [17]. The basic idea is that predesigned and preverified blocks of logic, as well as newly designed blocks of logic, called cores, are integrated by the core integrator to an SoC. The cores, provided by core vendors, may each have a different origin, such as from various companies, reuse from previous designs, or the cores can be completely new in-house designs. The test designer is responsible for the design of the system's test solution, which includes decisions on the organization and the application of test data (test stimuli and test responses) for each core in the system. Test application time minimization is often one of the main objectives since it is highly related to the cost of test, but it is also important to minimize the added overhead, such as additional wiring, while constraints and conflicts should be considered.

the systems at higher abstraction levels. However, modeling

A core-based SoC design methodology consists usually of two major steps: a core selection step, where the core

<sup>•</sup> The authors are with the Embedded Systems Laboratory, Department of Computer and Information Science, Linköping University, SE-581 83 Linköping, Sweden. E-mail: {erila, zpe}@ida.liu.se.

Manuscript received 5 Dec. 2004; revised 22 Apr. 2005; accepted 14 July 2005; published online 21 Dec. 2005.



Fig. 1. Design flow in a core-based design environment: (a) traditional and (b) proposed.

integrator selects the appropriate cores for the system, and the core test design step, where the test solution for the system is created, which includes test scheduling and the design of the infrastructure for test data transportation, the TAM (Test Access Mechanism). These two steps are traditionally performed in sequence, one after the other (see Fig. 1a). For such an SoC design-flow, it is important to note that the core integrator can, in the initial design step (core selection), select among several different cores, often from several core vendors, to implement a certain functionality in the system. The core integrator selects, based on each core's design characteristics given in its specification, the cores that fit the system best. Each possible core may not only have different design characteristics, but may also have different test characteristics (for instance, test sets and test power consumption). For example, one core may require a large ATE (Automatic Test Equipment) stored test set, while another core, implementing the same functionality, requires a combination of a limited ATE test set and a BIST (Built-In Self-Test) test set. The decision on which core to select therefore has an impact on the global test solution. Selecting the optimal core based only on its functionality will lead to local optimum, which is not necessarily the global optimum when the total cost of the system, including test cost, is considered. In other words, the selection (of cores and/or tests) must be considered with a system perspective in order to find a globally optimized solution. This means that there is a need for a test solution design tool that can be used in the early core selection process to explore and optimize the system's test solution (see Fig. 1b). Such a tool could help the test designer to answer the following question from a core integrator: "For this SoC-design, which of these cores are the most suitable cores for the system's test solution?"

We have previously proposed a technique for integrated test scheduling and TAM design where a weighted costfunction based on test time and TAM wiring cost is minimized while considering test conflicts and test power consumption [11]. We assumed that the tests for each testable unit were fixed and the main objective was, for a given system, to define a test solution. In this paper, on the other hand, we assume that, for each testable unit, several alternatives may exist. We propose a technique to integrate *core selection, test set selection, test resource floorplanning, TAM design,* and *test scheduling* in a single procedure. Core selection, test set selection, test resource floorplanning, TAM design, and test scheduling are highly interdependent. The test time can be minimized by scheduling the tests as concurrently as possible; however, the possibility of concurrent testing depends on the size of the TAM connecting the test resources (test sources and test sinks). The placement of the test resources has a direct impact on the length of the TAM wires. And, finally, the selected test sets for each testable unit are partitioned over the test resources and have a large impact on the TAM design and the test schedule. Therefore, these problems must be considered in an integrated manner.

Test power consumption is becoming a severe problem. In order to reduce testing times, concurrent execution of tests is explored. However, this may lead to more power than the given power budget of the system being consumed and that can damage the system. The proposed technique includes an improved power model that considers 1) global system-level limitations, 2) local limitations on power-grid level (hot-spots), as well as 3) core-level limitations. The motivation for the more elaborate power model is that the system is designed to operate in normal mode; however, during testing mode, the testable units are activated in a way that would not usually occur during normal operation. It can lead to 1) the systems power budget being exceeded or 2) hot-spots appearing and damaging a certain part in the system or 3) a core being activated in such a way that the core is damaged.

The proposed technique can be used to explore alternative cores for an SoC, different test alternatives for each testable unit, as well as the placement of test resources. As the design alternatives increase, we make use of Gantt charts to limit the search space. We have implemented the proposed technique, an estimation-based technique, and a pseudoexhaustive technique. In the experiments, we have compared with our previously proposed technique, where the cores are fixed and the tests are fixed. The experiments show that allowing design and test selection can reduce the final test cost and the experiments using the estimationbased technique show that it is difficult to produce highquality solutions and the experiments with the pseudoexhaustive technique demonstrate that the search space is enormous. The proposed technique, on the other hand, produces solutions with a total cost that is near the cost produced by the pseudoexhaustive technique but at a computational cost that is near the estimation-based technique.

The rest of the paper is organized as follows: Background and an overview of prior work are given in Section 2 and the problem formulation is in Section 3. The test problems and their modeling are in Section 4 and the algorithm and an illustrative example are in Section 5. The experimental results are in Section 6 and the conclusions are in Section 7.

## **2** BACKGROUND AND RELATED WORK

The technology development has, as discussed above, enforced the introduction of the core-based design environment where reusable logic blocks (cores) are combined to form a system that is placed on a single die [17]. A corebased design and production flow is typically a sequential process that starts with core selection, followed by test



Fig. 2. System design with different alternatives.

solution design, and, after production, the system is tested (Fig. 1a). In the core selection stage, the core integrator selects appropriate cores to implement the intended functionality of the system. For each function, there are often a number of possible cores to select from and each candidate core has its specification. The specification includes, for instance, data on performance, power consumption, area, and test characteristics. The core integrator explores the design space (search and combines cores) in order to optimize the performance of the SoC. Once the system is fixed (the cores are selected), the test designer designs the TAM and schedules the tests based on the test specification for each core. In such a design flow (illustrated in Fig. 1a), the test solution design is a consecutive step to core selection. This means that, even if each core's design is highly optimized, when integrated as a system, the system's global test solution is not optimized.

A design flow such as the one in Fig. 1b, on the other hand, integrates the core selection step and the test solution design step. The advantage is that it is possible to consider the impact of core selection when designing the test solution. In such a design flow (Fig. 1b), the global system impact on core selection is considered and it is possible to develop a more optimized test solution. The impact of the design flow in Fig. 1b can be illustrated as in Fig. 2, where the core type is floorplanned in the system, but there is not yet a design decision on which core to select. For each core type (the floorplan position), several alternative cores can be used. For instance, for the cpu core, there are three alternative processor cores (cpu1, cpu2, and cpu3), as illustrated in Fig. 2. And, for each core, several tests can be given.

In this paper, we make use of the test concepts introduced by Zorian et al. [15], which are illustrated with an example in Fig. 3. The example consists of three main blocks of logic, core A (CPU core), core B (DSP core), and core C (UDL, user-defined logic, block). A test source is where test stimulus is created or stored and a test sink is where the test response is stored or analyzed. The test resources (test source and test sink) can be placed on-chip or off-chip. In Fig. 3, the ATE serves as an off-chip test source and off-chip test sink, while TG1, for instance, is an on-chip test source. The TAM is the infrastructure 1) for test stimulus transportation from a test source to the testable unit and 2) for test response transportation from a testable unit to a test sink. A wrapper is the interface between a core and the TAM and a core with a wrapper is said to be wrapped while a core without a wrapper is said to be unwrapped. Core A is a wrapped core, while Core C is



Fig. 3. A system and the illustration of some test concepts.

unwrapped. The wrapper cells at each wrapper can be in one of the following modes at a time: *internal mode, external mode,* and *normal operation mode*. In addition to the definitions by Zorian et al. [15], we assume that a testable unit is not a core, but a block at a core, and that a core can consist of a set of blocks. For example, core A (Fig. 3) consists of two blocks (A.1 and A.2).

For a fixed system where cores are selected and floorplanned and, for each testable unit, the tests are fixed, the main tasks are to organize the testing and the transportation of test stimuli and test responses (as the example design in Fig. 3). Several techniques have been proposed to solve different important problems under the assumption that the cores are already selected (design flow as in Fig. 1a).

Zorian [14] proposed a test scheduling technique for a fully BISTed system where each testable unit is tested by one test with a fixed test time and each testable unit has its dedicated on-chip test source and its dedicated on-chip test sink. A fixed test power value is attached to each test and the aim is to organize the tests into sessions in such a way that the summation of the power consumed in a session is not above the system's power budget, while the test application time is minimized. In a system where the testable units share test sources and test sinks, the test conflicts must be taken into account. Chou et al. proposed a test scheduling technique that uses a conflict graph to handle general conflicts and minimizes the test time for systems where both the test time and power consumption for each test are fixed [1].

The approaches by Zorian and Chou et al. assume fixed testing times for each testable unit. The test time for a core can be fixed by the core provider, which may be due to the core providers having optimized their cores in order to protect the IP-blocks, for instance. However, the test time at a core is not always fixed. For scan-tested cores, the scanned elements can be connected to any number of wrapper chains. If the scanned elements (scan-chains, inputs, and outputs) at a core are connected to a small number of wrapper chains, the testing time is longer compared to when the scan elements are connected into a larger number



 $t_i$ : test i  $\{t_I, ..., t_n\}$ : a set of tests  $r_j$ : test source j  $s_k$ : test sink k  $b_l$ : block l  $c_m$ : core m

Fig. 4. Illustration of design alternatives.

of wrapper chains. Iyengar et al. proposed a scheduling technique for systems where the testing time for all cores is flexible and the objective is to form a set of wrapper chains for each core in such a way that the testing time for the system is minimized [7].

In order to minimize the test times, as many fault locations as possible are activated concurrently, which leads to high power consumption. Zorian [14] and Chou et al. [1] assign a fixed power value to each test and make sure that the scheduling does not activate the tests in such a way that the system's power budget is exceeded at any time. Bonhomme et al. [2] and Saxena et al. [12] proposed a clock-gating scheme intended to reduce the test power consumed during the scan-shift process. The advantage is that the test power can be reduced at a core with such a scheme and, hence, a higher number of cores can be scheduled for test concurrently. The basic idea is if n scanchains at a core are to be connected to *m* wrapper-chains (n > m), only *m* scan-chains can be loaded at a time, which means that not all *n* chains are active at the same time, hence, lower power consumption.

There has been research to find the most suitable ATE/ BIST partition for each testable unit. Sugihara et al. investigated the partitioning of test sets where one part is on-chip test (BIST) and the other part is off-chip test using an ATE [13]. A similar approach was proposed by Jervan et al. [8], which later was extended to not only locally optimize the test set for a core but to consider the complete system by using an estimation technique to reduce the test analysis complexity [9].

Hetherington et al. discussed several important test limitations such as ATE bandwidth and memory limitations [5]. These problems, as well as the problems described above, are important to consider in the search of a final test solution for the system.

The problems addressed above are all individually important to consider when designing the test solution for a SoC. However, it is important to consider them all simultaneously and from a system test perspective. We have previously proposed an integrated technique for test scheduling and TAM design where the test application time and the TAM design are minimized while considering test conflicts and power consumption [11]. The technique handles unwrapped as well as wrapped cores and also systems where some cores have a fixed testing time and some cores have a flexible testing time. The technique is also general in the test source and test sink usage. Each test can be defined to use any test source and any test sink. It is not necessary for a test to use a test source and a test sink where both are placed on-chip or both are placed off-chip. Furthermore, the technique allows an arbitrary number of tests per testable unit, which is an important feature in order to handle testing for timing faults and delay faults and not only stuck-at faults. However, that technique assumes that the tests for each testable unit are fixed and defined.

## **3 PROBLEM FORMULATION**

Fig. 4 illustrates the problem we address. We assume that a floorplan is given of an SoC where the core types are defined but the particular core is not yet selected. For example, at position  $c_x$ ,  $c_1$  and  $c_2$  are the alternatives. Each of the alternative cores may consist of a set of blocks (testable unit) where each block has multiple test alternatives. For instance,  $b_1$  at  $c_1$  can be tested by test  $t_1$  or by tests  $t_2$  and  $t_3$ . Each test is attached to one block and each test has its combination of test source and test sink. For instance,  $t_1$  makes use of  $r_1$  and  $s_1$ . Since no other test in the system makes use of  $r_1$  and  $s_1$ ,  $r_1$  and  $s_1$  will most likely not limit the test time. On the other hand, since  $s_1$  and  $r_1$  are not used by any other test, the added TAM has a low utilization, which leads to the waste of resources.

An example of an input specification of the example system in Fig. 3, the starting point in our approach, is given in

[Global Options] MaxPower = 100[Power Grid] #name power\_limit p\_grid1 50 p\_grid2 60 [Cores] #name x y block\_list coreA 20 10 { blockA1 blockA2 } coreB 40 10 { blockB1 blockB2 } coreC 20  $20 \{ blockC1 \}$ [Generators] #name x y max\_bw memory 10 0 ATE 4 200TG1 30 0 1 50 // the rest of the generators TG2 30 10 1 100 [Evaluators]#name x y max\_bw 50 0 4 ATE TRE1 30 0 1 // the rest of the evaluators TRE2 30 10 1 [Tests]#name pwr time tpg tre min\_bw max\_bw mem ict flexible\_pwr tA1.1 60 60 TG1 TE1 1 1 10 no no // more tests for coreA TG1 TE1 1 1 10 72 tB1.1 60 no no // more tests for coreB 80 TG1 TE2 1 4 10 tC1.1 70 coreB no // more tests for coreC [Blocks]#name idle\_pwr pwr\_grid test\_sets {}{} { tA1.1 }{ tA1.2 tA1.3 } blockA1 0 p\_grid1 blockA2 0 { tA2.1 }{ tA2.2 } p\_grid1 blockB1 5 p\_grid2 { tB1.1 tB1.2 } { tB1.3 } 0 blockC1 p\_grid1 { tC1.1 } [Constraint] #name{block1, block2, ..., blockn} tA1.1 {blockA1} // constraints for each test tC1.1 {blockC1 blockA1 blockA2 blockB1 blockB2}

Fig. 5. Input specification for the example system in Fig. 3.

Fig. 5. The structure of the input specification is based on the specification we made use of in [11]. The major extensions are 1) for each block (testable unit), several alternative lists of tests can be specified, instead of as before where it was only possible to assign one dedicated list of tests per block, and 2) the improved power-grid model makes it possible to more accurately model power dissipation.

The advantage of the possibility of specifying several lists of tests for each block (testable unit), where each test in a list makes use of its specified resources (test source and test sink) and each test has its test characteristics, is that it makes it possible to explore different design alternatives. The test problems that are considered in our technique and their modeling are discussed in Section 4. The input specification is explained in the following text, where the notations are defined and illustrated with the example given in Fig. 5.

The cores are floorplanned, i.e., given (x, y) coordinates and each core consisting of a set of blocks (testable units):

[Cores] #name x y block\_list coreA 20 10 { blockA1, blockA2 }

For each block, several sets of tests are available, where each set of tests is sufficient for the testing of the block. For instance, to test block blockA1, three possible test sets are given:

[Blocks] #name	test_sets {} {}
blockA1	{tA1.2 tA1.3}

{tA1.1} or {tA1.2 tA1.3} should be selected where each test has its resources and characteristics.

The proposed technique will select cores and corresponding blocks and, for each block, the set of tests to use in order to produce an optimized test solution for the system. The cost of a test solution is given by the test application time and the amount of routed TAM wires:

$$cost = \alpha \times \tau_{total} + \beta \times TAM, \tag{1}$$

where  $\tau_{total}$  is the total test application time (the end time of the test with the latest test time), *TAM* is the routing length of all TAM wires, and  $\alpha$  and  $\beta$  are two user-defined constants used to determine the importance of test time in relation to TAM cost. The selection of the user-defined constants  $\alpha$  and  $\beta$  is based on the characteristics of the particular SoC; hence, it is therefore not possible to define universal values on  $\alpha$  and  $\beta$ .

The produced output from our technique is a test schedule where the cores are selected and, for each block (testable unit) at the selected cores, the tests are selected and given a start time and an end time in such a way that all conflicts and constraints are not violated, and a corresponding TAM layout, where the cost (1) is minimized.



Fig. 6. Test time analysis for core 11 in design P93791, where the cost  $w \times \tau$  is plotted for the cores at TAM widths (*w*) 1 to 16.

## 4 TEST PROBLEMS AND THEIR MODELING

In this section, we discuss the test problems that have to be considered in our approach and the modeling of the problems.

## 4.1 Test Time

The testing time for a testable unit can be *fixed* or *flexible* prior to the design of the test solution. A core provider might protect the core and, therefore, optimize the core and its core wrapper prior to delivery, hence, having the testing time fixed. On the other hand, the testing time for a core can be flexible, such as for a scan-tested core where the scanned elements (scan-chains and wrapper cells) can be connected into one or several wrapper-chains. The testing time for a test with flexible test time depends on the number of wrapper-chains. It is important to note that tests with fixed and flexible testing times can be mixed in the system. The test time model must therefore handle systems where some cores have fixed test time while other cores have flexible testing time.

A higher number of wrapper-chains at a core results in lower testing time compared to if fewer wrapper-chains are used. The scan-chains at a core can be few and unbalanced (of unequal length), and the testing time might not be linearly dependent on the number of wrapper chains. Therefore, we analyzed the linearity of the testing time  $(\tau)$ versus the number of wrapper-chains (w), that is, if  $\tau \times w = constant$ . We selected the scan-tested cores in one of the largest ITC '02 designs, namely, the P93791 design [11]. We observed that the testing time for core 11 was the most nonlinear (as shown by the curve labeled as Core 11original in Fig. 6). We noted that the 576 scanned elements were partitioned into 11 scan-chains (with the length 82 82 82 81 81 81 18 18 17 17 17). We redesigned core 11 into four new cores with 11, 22, 44, and 88 balanced scan-chains, respectively. We plotted the  $\tau \times w$  curves for all these cores in Fig. 6. As the number of scan-chains increases, the value  $\tau \times w$  becomes more or less constant at any TAM width. The testing time at a single wrapper chain times TAM width

one is 149,381 (marked in Fig. 6). For core 11 with 44 balanced scan-chains, the value  $\tau \times w$  at any TAM width is always less than 5 percent from the constant theoretical value. It is important to note that, for all cores, the value  $\tau \times w$  is almost constant within a certain range. We assume that the core test designer optimizes the cores, hence, the number of scan-chains at a core is relatively high and of nearly equal length.

In our model, we specify the testing time for a testable unit at a single TAM wire and the bandwidth limitations. For instance, a test tA1.1 has a test time of 60 time units at a single wrapper chain and where the scanned elements can be arranged into wrapper-chains in the range 1 to 4 (only test time and bandwidth are mentioned):

[Tests] #name test time minbw maxbw tA1.1 60 1 4

We assume, based on our experiments, that the test time is linear to the number of TAM wires within the bandwidth range. It means that, given the test time at a single TAM wire ( $\tau_1$ ), the test time  $t_i$  can be computed by:

$$\tau_i = \frac{\tau_1}{i},\tag{2}$$

where *i* is in the range [minbw, maxbw]. If the testing time is fixed, minbw = maxbw.

## 4.2 Test Power Consumption

The test application time is reduced if a high number of cores are activated and tested concurrently; however, it leads to higher switching activity and higher power dissipation. The system-level power budget can be exceeded and high power consumption can damage the system. Furthermore, if cores that are floorplanned close to each other are activated concurrently, a hot-spot can be created and it also can damage the system. For instance, assume a memory organized as four banks where, in normal operation, only one bank is activated at a time. However, during testing, in order to shorten the test time, all banks are activated concurrently. The system's total power limit might not be exceeded, however, a local hotspot is created in the memory subsystem and the system may be damaged.

The switching activity and power consumption are higher during testing than during normal operation. It means that a core during testing can dissipate power above its specified limit due to the nature of the test stimuli and/ or the test clock frequency.

We therefore make use of a three-level power model: *system-level, power-grid-level (local hot-spot),* and *core-level.* For the system-level, we make use of the power model defined by Chou et al. where the summation of power values of the concurrently executed tests is below the power budget of the system [1].

As an example, we can specify the system budget as:

#### MaxPower = 100

and, for each test, we specify the power consumed when the test is activated:



Fig. 7. A memory organized as a bank of four blocks powered by a common grid.

[Tests]	#name	pwr	time	tpg	tre	min_bw	max_bw
	tA1.1	60	60	TG1	TE1	1	4
	mem	ict	flexik	ole_pv	vr		
	10	no	no				

Additionally, the idle power, the power consumed when a block is in stand-by mode and not active, is also specified for each block:

[Blocks] #name idle pwr test sets {} {} blockA1 0 {tA1.1} {tA1.2 tA1.3}

For local hot-spots, we introduce a power grid model, which has similarities to the approach proposed by Chou et al. [1], but, instead of having a single maximal power constraint for the whole system, we have local power constraints for subparts of the system. We assume that each block (testable unit) is assigned to a power grid, where the power grid has its power budget. The system can contain an arbitrary number of power grids. Blocks assigned to a power grid cannot be tested in such a way that the power grid budget is exceeded at any time; the scheduling algorithm prevents such a situation from occurring by selecting alternative tests or scheduling tests later.

An example to illustrate the need for power grids is as follows: A memory can be organized as a bank of memory blocks (see Fig. 7). Assume that the memory, during normal operation, never accesses more than a single memory block at a time and the power grid is designed accordingly.

A single grid is specified as:

[PowerGrid] #name power\_limit p\_grid1 50

For each block the power grid usage is given as:

[Blocks] #name	idle pwr	pwr_grid	test sets {} {}
blockkA1	0	p_grid1 {t	A1.1} {tA1.2 tA1.3}

As discussed above, some tests have a fixed testing time while other tests allow flexible testing times. Regarding test power consumption, we have some tests where the power is fixed regardless of the number of assigned TAM wires, while other tests allow the power to be adjusted by clockgating [12]. Clock-gating can be used to reduce the power consumption so that a higher number of tests can be executed concurrently, but it also can be used for the units under test where its own power dissipation is higher than its allowed power consumption due to, for instance, a too high test clock frequency.

The motivation behind core-level adjustments is twofold. First, by lowering the power consumption at a core, a higher number of cores can be activated concurrently without violating the total power budget. Second, since test power consumption often is higher than that during normal operation, the power dissipation during test at a specific core can be higher than its own power budget.

The power consumption for a test is given as a single value, which corresponds to the power consumption when a single TAM wire is used, for instance, as in the following example (interconnection test flag, test source, and test sink usage are omitted):

[Tests]	#name	pwr	time	minbw	maxbw	flexible_pwr
	tA1.1	60	60	1	4	yes
	tC1.1	70	80	1	4	no

Note that we include the possibility of specifying if clockgating can be used by setting *flexible\_pwr* to yes or no. If power can be modified, we assume a linear dependency [12]:

$$p_i = p_1 \times tam,\tag{3}$$

where  $p_1$  is the power at a single TAM wire,  $p_i$  is the power consumed when *i* number of TAM wires are used; *i* has to be in the specified range [*minbw* : *maxbw*].

### 4.3 Test Conflicts

During the test solution design, there are a number of conflicts that have to be considered and modeled. Each test may have its defined constraints depending on the type of test; stuck-at, functional, delay, timing, etc. For general conflicts, we make use of the following notation [11]:

[Constraints] #name {block1, block2, ..., block n} tA1.1 {blockA1}

The notation means that, when applying test tA1.1, blockA1 must be available and no testing can be performed on it since it is used by test tA1.1 or tA1.1 might interfere with blockA1. This modeling supports general conflicts, which can be due to hierarchy where cores are embedded in cores or interference during testing. The model can also be used for designs where an existing functional bus is used as the TAM. A functional bus can be modeled as a dummy block, where, usually, only one test can be active at a time.

A test source ([Generators]) may have limited bandwidth and memory. The bandwidth limitation and the memory limitation are especially critical for ATEs, but are also important if on-chip resources such as memories are used for test data storage. We denote bandwidth limitation as an integer stating the highest allowed bandwidth for the test source. For memory limitations, an integer is used as the maximal memory capacity. A test sink ([Evaluators]) can also have a limited bandwidth and, in a similar way as with test sources, we denote it with an integer. For simplicity, we only assign memory constraint at the test source. For each test, we give an integer value as its memory requirement. An example with testA1.1 using test source ATE and test sink ATE with memory requirement 10 is given below (for the test, only name, source, sink, and memory limitation are given):

[Genera	itors] #na	me	х	у	maxbw	memory
	ATE	3	10	0	4	100
[Evaluators] #name x y					maxbw	
	ATE	3	50	0	4	
[Tests]	#name	tpg	tre	е	mem	
	tA1.1	AT	ΕA	ΤE	10	







#### Fig. 9. The algorithm.

The wrapper conflicts are slightly different compared to general conflicts because of the TAM routing. The testing of a wrapped core is different from the testing of an unwrapped one. The testing of the wrapped core A (Fig. 3), for example, is performed by placing the wrapper in internal test mode and test stimuli are transported from the required test source using a set of TAM wires to the core and the produced test responses are transported from the core using a set of TAM wires to the test sink. In the case of an unwrapped testable unit such as the UDL block, the wrappers at cores A and B are placed in external test mode. The test stimuli are transported from the required test source on the TAM via core A to the UDL block and the test responses are transported via core B to the TAM and to the test sink. It means that, for the TAM design, the TAM should be routed to core A and B (and not to the UDL block).

We model the wrapper conflict as in the following example, with two blocks (bA and bB) and one test per block (tA and tB):

[Blocks] #name {test1, test2,..., test m} {test1, ..., test n} bA {tA1.1} bB {tB} [Tests] name tg tre ict tΑ s1bB r1 tΒ r1 s1no

The difference between these tests is illustrated in Fig. 8. Test tB is not an interconnection test, hence, ict (interconnection test) is marked as no. It means that there will be a connection between r1 to bB and from bB to s1, marked as TAM for tB in Fig. 8. Test tA, on the other hand, is an interconnection test with bB. It means that r1 is connected to bA and bB is connected to s1. The required TAM is marked as TAM for tA in Fig. 8.

## 5 TEST DESIGN ALGORITHM

In this section, we describe the proposed test design algorithm (outlined in Fig. 9, and detailed in Fig. 10 and Fig. 11).

In order to evaluate the cost of a test solution, we make use of (1). At a design modification, the cost change before and after modification is given by:

$$(\Delta \tau \times \alpha + \Delta T A M \times \beta), \tag{4}$$

where  $\Delta \tau$  ( $\Delta TAM$ ) is the difference in test time (TAM cost) before and after the modification.

The TAM cost is given by the length l and its width w ( $TAM = l \times w$ ) and by combining the cost function (1) considering only one testable unit and the test time versus TAM cost (2), the optimal TAM bandwidth is given by [11]:

$$w = \sqrt{(\alpha \times \tau)/(\beta \times l)}.$$
 (5)

A detailed description of the algorithm (Fig. 9) is in Fig. 10 (test set selection algorithm) and Fig. 11 (test scheduling and TAM design). The algorithm starts by the part given in Fig. 10, where the list of test sets for each testable unit is sorted based on the cost function (1). The cost for each testable unit is locally optimized; however, there is, at this point, no global consideration on the sharing of TAM wires or conflict avoidance. For each testable unit, the first set of tests for each testable unit is selected and the set is scheduled and the TAM is designed (Fig. 11). From the test schedule, the test application time is given and, from the TAM layout, the TAM cost for the solution is given. The algorithm checks the use of resources from a Gantt-chart for the solution (explained below in Section 5.1). For example, assume that a test solution generates a Gantt chart as in Fig. 12, where TG:r1 is the critical resource. For all tests that use the critical (limiting) resource, the algorithm tries to find alternative tests. Equation (4) is used to evaluate the change in cost for each possible alternative (at the critical resource). Instead of trying all possible alternatives, a limited number of design modifications (given from the Gantt chart) are explored. And, to reduce the TAM cost, existing TAMs are reused as much as possible (a test may be delayed and applied later).

1. fo	r each block (testable unit) {
2.	for each test set at a block {
3.	compute optimal bandwidth for each test (Eq. 5);
4.	compute cost for the full test set (Eq. 1);
5.	}
6.	place the test sets sorted descending on the cost (step 4.):
7.	select the first test set in the list as the active test set
8. }	
9. re	neat until no modification can be performed {
10.	create test schedule and TAM layout (see Figure 11)
11.	if the total cost for schedule and TAM layout is best so far{
12.	remember this test schedule and the TAM layout
13.	} else {
14.	if last modification was a TAM width modification {
15.	undo the TAM width modification
16	}
17	if last modification was a test set modification {
18	remove the test set from the blocks list of test sets
19	}
20	}
21	for each block (
22	if the active test set has a test resource limiting the solution {
23	compute cost for increasing the TAM width with 1
24	for every other test set for the block {
25	compute the cost of changing this test set based on Fa
26	if the cost is lower than lowest cost {
20.	remember this test set
27	}
28	, ,
29	if lowest cost for the block is lower than the total cost {
30	remember block TAM width and test set modification
31	}
32	
33	if any alternatives exists {
34	perform TAM width modification or test set modification
35	} else {
36	for each block {
37	for each test set after the active test set for the block?
38	compute the cost of selecting it $(Fa, 4)$
39	if cost is lowest then remember this test set
40	}
41	) if lowest cost for the block is lower than lowest total
71.	cost then {
42	remember block change and test set change
43	}
44	}
45	; if lowest cost difference <0 {
46	do the test set change
47	}
48.	}
49.	·
····)	

Fig. 10. Test set selection algorithm.

## 5.1 Resource Utilization

We make use of a machine-oriented Gantt chart to track bottlenecks (the resource that limits the solution) [3]. We let the resources be the machines, and the tests be the jobs to show the allocation of jobs on machines. For example, a Gantt chart is given in Fig. 12, where test B2 needs TG:r2 and TRE:s2. An inspection of Fig. 12 shows that TG:r2 and TRE:s2 are not critical to the solution. On the other hand, test source TG:r1 is the most critical one. It means that testA, testB1, and testC are the obvious candidates for modification. The Gantt chart pinpoints bottlenecks and therefore reduces the search for candidates for modification. Note that the Gantt chart does not show a valid schedule, only the usage of resources in the system.

1. se	ort the list of tests descending according to the cost function.
2. re	epeat until the list is empty {
3.	select the first test in the list
4.	repeat until a test is scheduled or at end of list {
5.	repeat until selected test is scheduled or
	bandwidth cannot be decreased {
6.	try to schedule the test at current time
7.	if fail to schedule {
8.	<i>if the bandwidth&gt;1 then</i>
9.	reduce bandwidth with 1
10.	}
11.	}
12.	if the selected test could not be scheduled {
13.	select the following test in the list
14.	}
15.	}
16.	if the test was scheduled {
17.	allocate TAM and remove the selected from the list
18.	} else {
19.	update current time to the nearest time in the future
	where there is available power to schedule the first test
	in the list
20.	}

Fig. 11. Test scheduling and TAM design algorithm.

## 5.2 Illustrative Example

We use the design example in Fig. 13 to illustrate the algorithm described above. The example (Fig. 13), simplified by removing power grids, memory limitations, and the list of general constraints, consists of two cores, each with a single block (testable unit), where each block can be tested in two ways; there are two alternative test sets for each block. For instance, blockA can be tested by testA1 or testA2. Each of the tests is defined with its test time, combination of test sink and test source, etc.

The algorithm proceeds as follows: Initial step: For each block, the test sets are ordered ascending according to the cost function ((1) assuming  $\alpha = \beta = 1$ ):

test	time	TAM	total cost
testA1:	60	40	100
testA2:	100	20	120
testB1:	72	40	112
testB2:	120	20	140

The evaluation results in the following sorted lists per block (first in the list is the best candidate):

blockA: {{testA1}, {testA2}}
blockB: {{testB1}, {testB2}}



Fig. 12. A machine-oriented Gantt chart [3].



Fig. 13. An illustrative example with a simplified specification where power grids, memory limitations, and general constraint list are not considered.

The first set of tests are selected as active, that is, for blockA {testA1} and for blockB {testB1}. The test scheduling algorithm sorts the tests based on test time and starts with the longest test, making the test schedule: testB starting at time 0 followed by testA starting at time 72. The resulting total test application time is 132. The TAM design algorithm connects TG1, coreB, coreA, and TA1, and the Manhattan length is 20 + 20 + 20 = 60. The total cost (at  $\alpha = \beta = 1$ ) for the test solution is then: 132 (test time) + 60 (TAM cost) = 192.

From the Gantt chart for this test solution, we observe that TG1 and TA1 both are used for 132 time units, while TG2 and TA2 are not used at all and we note that TG1 and TA1 limit the solution. Based on the Gantt-chart, the algorithm tries to find an alternative that does not use TG1 and TA1. For each test that uses the limiting resources in the Gantt chart, in our example TG1 and TA1, the algorithm computes the alternative cost of using other resources. It is important to note that, in order to limit the number of possible options, we only try with the tests that depend on the resources critical to the solution.

As the first alternative modification, we try to use testA2 to test BlockA instead of using testA1. It means that testA1 will not be executed (only one of the set of tests for each block is required and each list contains only one test). We evaluate the impact of the test modification on the TAM layout and we observe that we do not have to include coreA in the TAM layout. Taking coreA out of the bus layout means that TAM corresponding to 20 units can be removed (testA2 makes use of different test resources compared to testA1). However, in order to execute testA2, we have to include wires from TG2 to coreA and from coreA to TA2. The additional required wiring corresponds to 20 units.

The difference in test time between testA1 and testA2 is (100 - 60 =) 40. It means that the total cost difference is

estimated to be: -20 (gain by not including coreA for testA1) + 20 (what we have to add to include TAM for TG2->coreA->TA2) + 40 = 40.

For the second alternative modification, we try testB2 instead of testB1. It means that a TAM (length and width) corresponding to 20 units can be removed. The additional TAM cost of adding testB2 (its resources) is 20 and the difference in test time between testB2 and testB1 is 48 (120 - 72). The cost difference for this alternative is -20 + 20 + 48 = 48.

In this example, we have two tests using the resources that are critical to the solution and we also had only one possible alternative per test. Since the first alternative is better than the second, the first one is selected. A new test schedule and a TAM layout are created where both testA1 and testB1 are

TABLE 1 Design Data

Design	Cores	Alternatives per core	Total number of tests
Design_1	2	3	8
Design_2	4	6	24
Design_3	8	4	48
Design_4	16	4	96
Design_5	4	2	8
Design_6	13	5	65
Design_7	26	9	234
Design_8	2	6	16
Design_9	4	7	40

Experiment	Design	Technique	Options	Test time	TAM	Total	CPU
		Test scheduling and TAM design [11]	Fixed test sets per core and only ATE tests	120	180	300	<1
		Test scheduling and TAM design [11]	Fixed test sets per core and only BIST tests	240	20	260	<1
1	Design_1	Estimation-based technique [10]	Selection allowed	100	110	210	<1
		Pseudo-exhaustive technique	Selection allowed	90	100	190	<1
		Proposed technique	Selection allowed	90	100	190	<1
		Test scheduling and TAM design [11]	Fixed ATE test sets per core position	3200	1500	4700	<1
		Test scheduling and TAM design [11]	Fixed BIST test per core	6800	400	7200	<1
2	Design_2	Estimation-based technique [10]	Selection allowed	4000	1600	5600	<1
		Pseudo-exhaustive technique	Selection allowed	1800	1800	3600	2
		Proposed technique	Selection allowed	2300	1700	4000	<1
		Test scheduling and TAM design [11]	Fixed ATE test sets per core position	2200	4700	6900	<1
		Test scheduling and TAM design [11]	Fixed BIST test sets per position	7900	1300	8900	<1
3	Design_3	Estimation-based technique [10]	Selection allowed	7000	2700	9700	<1
		Pseudo-exhaustive technique	Selection allowed	2500	2700	5200	241
		Proposed technique	Selection allowed	3100	3500	7400	<1
		Test scheduling and TAM design [11]	Fixed ATE test sets per core position	4500	8900	13400	<1
		Test scheduling and TAM design [11]	Fixed BIST test sets per core position	15350	2800	18150	<1
4	Design_4	Estimation-based technique [10]	Selection allowed	13850	4700	18550	<1
		Pseudo-exhaustive technique	Selection allowed	-	-	(11500)	N.A (12 hours)
		Proposed technique	Selection allowed	5600	5900	11500	2
		Estimation-based technique [10]	Selection allowed	400	140	540	<1
5	Design_5	Pseudo-exhaustive technique	Selection allowed	320	120	440	<1
		Proposed technique	Selection allowed	320	120	440	<1
		Estimation-based technique [10]	Selection allowed	240	810	1050	<1
6	Design_6	Pseudo-exhaustive technique	Selection allowed	-	-	(883)	N.A.(12 hours)
		Proposed technique	Selection allowed	193	690	883	<1
		Estimation-based technique [10]	Selection allowed	215	1072	1287	<1
7	Design_7	Pseudo-exhaustive technique	Selection allowed	-		(1232)	N.A.(12 hours)
		Proposed technique	Selection allowed	220	692	1182	4
		TAM design and scheduling	Best core (test time) per position	1200	90	1290	<1
		TAM design and scheduling	Best core (power) per position	2000	180	2180	<1
8	Design_8	Estimation-based technique [10]	Selection allowed	660	100	760	<1
		Pseudo-exhaustive technique	Selection allowed	700	20	720	<1
		Proposed technique	Selection allowed	700	20	720	<1
		TAM design and scheduling	Best core (test time) per position	2500	4400	6900	<1
		TAM design and scheduling	Best core (power) per position	4600	1500	6100	<1
9	Design_9	Estimation-based technique [10]	Selection allowed	4600	1500	6100	<1
		Pseudo-exhaustive technique	Selection allowed	1500	1900	3400	5
		Proposed technique	Selection allowed	3600	1500	5200	<1

TABLE 2 Results

scheduled to start at time 0 and there are two TAMs, one connecting TG2->coreA->TA2 with a length 20 and one connecting TG1->coreB->TA1 with length 40. The total cost is 60 + 72 = 132 (an improvement from 192 to 132).

# 6 **EXPERIMENTAL RESULTS**

The objective with the experiments is to check that the proposed technique produces high quality solutions at a reasonable computational cost (CPU time). For comparison purposes, we compare the proposed technique with the technique [11] that does not support core selection and two techniques that do support selection, an estimation-based technique [10] and a pseudoexhaustive algorithm. The technique [11] that does not support core selection is used as a reference point to show that the test solution can be improved by allowing selection of tests and cores; the estimation-based technique, which tries to predict the cost at a low computational cost, is used to demonstrate that finding a high quality test solution is not trivial; and the pseudoexhaustive algorithm, which basically tries all possible solutions, is used to demonstrate that the search space is enormous in size.

We have created a set of nine designs with data as in Table 1. The experimental results are collected in Table 2, where we have used  $\alpha = \beta = 1$  for the cost function, and the total cost is therefore the sum of the two. Table 2 reports test application time, the TAM cost, and the CPU time. Note that the total cost is optimized, which means that the combination of test time and TAM cost are optimized. It can lead to cases where one of the optimization parameters (test time or TAM cost) is not minimal since the algorithm can find a solution where the most optimization can be gained from the other parameter and, most importantly, the combination of the two, which is optimized, is minimal. In the experiments, we compared the test scheduling and TAM design technique [11], the estimation-based technique, and the pseudoexhaustive technique with the proposed technique for test set selection. For the test scheduling and TAM design technique [11] that does not allow core selection and test set selection, we assumed preselected cores and test per testable unit for the experiments. We explored different strategies for the preselection. In experiments 1, 2, and 3, we selected the cores with the lowest test time of an ATE test as well as BIST test. In experiments 8 and 9, we selected the cores that consumed the lowest test time and, also, the lowest power. The results show that, by allowing selection, it is possible to select cores and tests in a such a way that the total cost of the test solution is reduced. The results show that the proposed technique produces results that are better than the results produced by the estimation-based technique and near the results produced by the pseudoexhaustive technique. The computational costs (CPU time) for each of the experiments are all reasonable, but, for the pseudoexhaustive technique on experiments 4, 6, and 7, we terminated after 12 hours; hence, the total cost for each one is the best found until abortion time.

## 7 CONCLUSIONS

Test design is traditionally considered as a final step in the system design-flow. However, as test design is becoming a significant part in terms of cost in the design-flow, it is important to consider test design as early as possible in the design-flow. Technology development has made it possible to design high-speed system chips that are shrinking in size but include an increasing number of transistors. In these system chips, the number of fault sites increases drastically and, therefore, a high test data volume is required. It is becoming important to consider test planning as early as possible in the design-flow. In this paper, we propose a technique where system test design is included in the core selection phase. The advantage is that the technique makes it possible to explore the impact of test design already taking a system's global perspective when deciding on which cores to be used to implement the system. The proposed technique can be used to explore the impact of 1) the core selection on the test solution, 2) the test set partitioning (BIST size versus ATE size) on the test solution, and/or 3) the placement of test resources (test source and test sink) on the test solution.

Prior works assume a system where cores, tests, and placement of test resources are already fixed when test planning is to be performed. It means that test scheduling and TAM design are the main problems. Our approach includes the interdependent problems of test scheduling, TAM design, test set selection, and test resource placement, together with core selection. Our technique defines a test solution where the test time and the TAM routing cost are minimized while test conflicts and power limitations are considered.

Test power consumption is becoming an important aspect to be considered; however, previously proposed power models have all been rather simplistic and have only focused on the global power budget. We have improved test power modeling by introducing a three level power budget model: system-level, power-grid (local hot-spot) level, and core-level. The advantage is that, with such a model, it is possible to have more elaborate power constraints on where the power is consumed in the system, at cores, at certain hot-spot areas, and at the global level.

The design space is enormous when integrating core selection, test set selection, test resource placement, and TAM design and, in order to limit it, we make use of Gantt charts to find the limiting resources (bottlenecks). For validation of the proposed technique, we have implemented the proposed technique, an estimation-based technique, and a pseudoexhaustive technique. We have compared the three techniques with a technique where cores and tests are selected prior to scheduling and TAM design and the results show that the total cost can be reduced by including core selection and test selection. Further, the experimental results show that the pseudoexhaustive technique cannot produce solutions within a reasonable CPU time and the estimation-based technique does not produce high quality solutions. The proposed technique can, on the other hand, produce high quality solutions at a reasonable CPU time.

#### ACKNOWLEDGMENTS

This research is partially supported by the Swedish National Program STRINGENT. A preliminary version of this paper was presented at the International Test Conference in 2004 [16].

#### REFERENCES

- R.M. Chou, K.K. Saluja, and V.D. Agrawal, "Scheduling Tests for VLSI Systems under Power Constraints," *IEEE Trans. VLSI Systems*, vol. 5, no. 2, pp. 175-185, June 1997.
- [2] Y. Bonhomme, P. Girard, L. Guiller, C. Landrault, and S. Pravossoudovitch, "A Gated Clock Scheme for Low Power Scan Testing of Logic ICs or Embedded Cores," *Proc. Asian Test Symp.* (ATS), pp. 253-258, Nov. 2001.
- [3] P. Brucker, Scheduling Algorithms. Springer-Verlag, 1998.
- [4] A.L. Crouch, Design for Test. Prentice Hall PTR, 1999.
- [5] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, and J. Rajski, "Logic BIST for Large Industrial Designs: Real Issues and Case Studies," *Proc. Int'l Test Conf. (ITC)*, pp. 358-367, Sept. 1999.

- [6] V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "Test Access Mechanism Optimization, Test Scheduling, and Tester Data Volume Reduction for System-on-Chip," *IEEE Trans. Computers*, vol. 52, no. 12, pp. 1619-1632, Dec. 2003.
- [7] V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "Co-Optimization of Test Wrapper and Test Access Architecture for Embedded Cores," *J. Electronic Testing; Theory and Applications (JETTA)*, pp. 213-230, Apr. 2002.
  [8] G. Jervan, Z. Peng, R. Ubar, and H. Kruus, "A Hybrid BIST G. Jervan, Z. Peng, R. Ubar, and H. Kruus, "A Hybrid BIST for the second se
- [8] G. Jervan, Z. Peng, R. Ubar, and H. Kruus, "A Hybrid BIST Architecture and Its Optimization for SoC Testing," *Proc. Int'l Symp. Quality Electronic Design (ISQED '02)*, pp. 273-279, Mar. 2002.
- [9] G. Jervan, P. Eles, Z. Peng, R. Ubar, and M. Jenihhin, "Test Time Minimization for Hybrid BIST of Core-Based Systems," *Proc. Asian Test Symp. (ATS '03)*, pp. 318-323, Nov. 2003.
- [10] E. Larsson and H. Fujiwara, "Test Resource Partitioning and Optimization for SoC Designs," Proc. VLSI Test Symp. (VTS '03), pp. 319-324, Apr. 2003.
- [11] E. Larsson, K. Arvidsson, H. Fujiwara, and Z. Peng, "Efficient Test Solutions for Core-Based Designs," *IEEE Trans. CAD of Integrated Circuits and Systems*, pp. 758-775, May 2004.
- [12] J. Saxena, K.M. Butler, and L. Whetsel, "An Analysis of Power Reduction Techniques in Scan Testing," *Proc. Int'l Test Conf. (ITC)*, pp. 670-677, Oct. 2001.
- [13] M. Sugihara, H. Date, and H. Yasuura, "Analysis and Minimization of Test Time in a Combined BIST and External Test Approach," Proc. Design and Test in Europe (DATE), pp. 134-140, Mar. 2000.
- [14] Y. Zorian, "A Distributed BIST Control Scheme for Complex VLSI Devices," Proc. VLSI Test Symp. (VTS), pp. 4-9, Apr. 1993.
- [15] Y. Zorian, E.J. Marinissen, and S. Dey, "Testing Embedded-Core Based System Chips," Proc. Int'l Test Conf. (ITC), pp. 130-143, Oct. 1998.
- [16] E. Larsson, "Integrating Core Selection in the SoC Test Solution Design-Flow," Proc. Int'l Test Conf. (ITC), pp. 1349-1358, 2004.
- [17] R.K. Gupta and Y. Zorian, "Introducing Core-Based System Design," IEEE Design and Test, vol. 14, no. 4, pp. 15-25, 1997.



**Erik Larsson** received the MSc, Tech. Lic, and PhD degrees from Linköping University in 1994, 1998, and 2000, respectively. From October 2001 to December 2002, he held a Japan Society for the Promotion of Science (JSPS) funded postdoctoral position at the Computer Design and Test Laboratory at the Nara Institute of Science and Technology (NAIST), Nara, Japan. Currently, he is an assistant professor and director of Studies of the Division for

Software and Systems in the Department of Computer and Information Science, Linköping Universitet, Sweden. 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. He is the author of the book Introduction to Advanced Systemon-Chip Test Design and Optimization (Springer 2005) and is a coquest editor for the IEE Computers & Digital Techniques special issue on resource-constrained testing of system chips. He received the best paper award for the paper "Integrated Test Scheduling, Test Parallelization and TAM Design" at the IEEE Asian Test Symposium (ATS) 2002 and he supervised the thesis which was selected as the best thesis by Föreningen Svenskt Näringsliv, 2002, and the thesis which was selected as the best thesis in the Department of Computer Science, 2004. He is a member of the program committee of Design and Test Automation in Europe (DATE) 2004, 2005, 2006, the IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS) 2004, 2005, 2006, and the Workshop on RTL ATPG & DFT (WRTLT) 2005, 2006. He is a member of the IEEE.



Zebo Peng (M'91-SM'02) received the BSc degree in computer engineering from the South China Institute of Technology, China, in 1982 and the Licentiate of Engineering and PhD degrees in computer science from Linköping University, Sweden, in 1985 and 1987, respectively. Currently, he is a professor of computer systems, director of the Embedded Systems Laboratory, and chairman of the Division for Software and Systems in the Department of

Computer Science, Linköping University. His research interests include design and test of embedded systems, electronic design automation, SoC testing, design for testability, hardware/software codesign, and real-time systems. He has published more than 180 technical papers in these areas and coauthored the books System Synthesis with VHDL (Kluwer, 1997), Analysis and Synthesis of Distributed Real-Time Embedded Systems (Kluwer, 2004), and System-Level Test and Validation of Hardware/Software Systems (Springer, 2005). Professor Peng was a corecipient of two best paper awards at the European Design Automation Conferences (1992 and 1994), a best paper award at the IEEE Asian Test Symposium (2002), a best paper award at the Design Automation and Test in Europe Conference (2005), and a best presentation award at the IEEE/ACM/IFIP International Conference on . Hardware/Software Codesign and System Synthesis (2003). He has served on the program committees of a dozen international conferences and workshops, including ATS, ASP-DAC, DATE, DDECS, DFT, ETS, ITSW, MEMOCDE, and VLSI-SOC. He was the general chair of the Sixth IEEE European Test Workshop (ETW '01), the program chair of the Seventh IEEE Design & Diagnostics of Electronic Circuits & Systems Workshop (DDECS '04), and the test track chair of the 2006 Design Automation and Test in Europe Conference (DATE '06). He is the vice-chair of the IEEE European Test Technology Technical Council (ETTTC). He is a senior member of the IEEE.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.