Linköping Studies in Science and Technology

Dissertation No. 1321

Temperature Aware and Defect-Probability Driven Test Scheduling for System-on-Chip

by

Zhiyuan He

Linköping University
INSTITUTE OF TECHNOLOGY

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköping 2010

To Fang,

for her endless love, patience, and support

Abstract

The high complexity of modern electronic systems has resulted in a substantial increase in the time-to-market as well as in the cost of design, production, and testing. Recently, in order to reduce the design cost, many electronic systems have employed a core-based system-on-chip (SoC) implementation technique, which integrates pre-defined and pre-verified intellectual property cores into a single silicon die. Accordingly, the testing of manufactured SoCs adopts a modular approach in which test patterns are generated for individual cores and are applied to the corresponding cores separately. Among many techniques that reduce the cost of modular SoC testing, test scheduling is widely adopted to reduce the test application time. This thesis addresses the problem of minimizing the test application time for modular SoC tests with considerations on three critical issues: high testing temperature, temperature-dependent failures, and defect probabilities.

High temperature occurs in testing modern SoCs and it may cause damages to the cores under test. We address the temperature-aware test scheduling problem aiming to minimize the test application time and to avoid the temperature of the cores under test exceeding a certain limit. We have developed a test set partitioning and interleaving technique and a set of test scheduling algorithms to solve the addressed problem.

Complicated temperature dependences and defect-induced parametric failures are more and more visible in SoCs manufactured

with nanometer technology. In order to detect the temperature-dependent defects, a chip should be tested at different temperature levels. We address the SoC multi-temperature testing issue where tests are applied to a core only when the temperature of that core is within a given temperature interval. We have developed test scheduling algorithms for multi-temperature testing of SoCs.

Volume production tests often employ an abort-on-first-fail (AOFF) approach which terminates the chip test as soon as the first fault is detected. Defect probabilities of individual cores in SoCs can be used to compute the expected test application time of modular SoC tests using the AOFF approach. We address the defect-probability driven SoC test scheduling problem aiming to minimize the expected test application time with a power constraint. We have proposed techniques which utilize the defect probability to generate efficient test schedules.

Extensive experiments based on benchmark designs have been performed to demonstrate the efficiency and applicability of the developed techniques.

Acknowledgments

Over the years, many people have contributed to this thesis, and I appreciate all their support. First and foremost, I would like to sincerely thank my supervisors Professor Zebo Peng and Professor Petru Eles, for their inspiration and guidance on my graduate study and research. Many creative and insightful ideas have been generated during the enlightening discussions. Special thanks to Zebo for the chats about social values and to Petru for introducing me to operas.

I would also like to thank Professor Bashir M. Al-Hashimi for hosting my stay at the University of Southampton, UK, in 2006, and for the fruitful collaboration on the temperature-aware testing issue.

Many thanks to all present and former members of the Embedded Systems Laboratory and colleagues in the Department of Computer and Information Science at Linköping University, for their kind help.

I appreciate the financial support of the Swedish Foundation for Strategic Research (SSF) via the Strategic Integrated Electronic Systems Research (STRINGENT) program.

I am deeply grateful to my father and mother, who have always been giving me their support, encouragement, and advices. Finally, I would like to express my deepest gratitude to my beloved wife, Huanfang, to whom this thesis is dedicated, for her endless love, patience, and sharing my ups and downs all the time.

Zhiyuan He
Linköping, June 2010

Contents

Chapter 1
Introduction

In order to assure correct circuit behavior, integrated circuits (ICs) have to be tested after fabrication. Nowadays, manufacturing test has become an essential part of IC production. Considered as a major contributor to the testing cost, test time needs to be reduced for the sake of cost reduction. Among various techniques, test scheduling is an efficient approach to reduce the test time. This thesis deals with test scheduling problems for systems-on-chip (SoCs) with specific concerns on temperature and power related issues as well as the consideration of defect probabilities. This chapter motivates our work and summarizes the contributions and the organization of the thesis.

1.1 Motivation

The steadily decreasing feature size of electronic devices in ICs has enabled higher integration density. Today's ICs may consist of billions of transistors manufactured with nanometer technology. As a consequence, more functionality is added into the system and higher performance is achieved, which results in substantially increased complexity of the system. Challenges have arisen in design, production and test of such highly complex electronic systems.

ICs manufactured with very-large-scale integration (VLSI) technology may have defects that are process-variation induced flaws or physical imperfections. Defects may lead to faults which can cause malfunction or system failure. Some faults can be detected by test methods, while others may escape all applied tests and cause reliability problems in the field. It is very important to capture as many faults as possible with production tests at the chip level, because escaping chip tests result in huge costs spent for testing, diagnosis and maintenance at the printed-circuit-board (PCB) and system levels, according to the rule of ten [Davis. 1994]. Therefore, effective test methods have to he developed for production tests of modern ICs.

Testing is expensive. It has been reported that testing cost is about 50% to 60% of IC manufacturing cost [Bushnell, et al. 2000]. Although the cost of ICs has been decreasing with the advances in technology, the percentage of the total cost attributed to testing has increased [Bushnell, et al. 2000]. One of the major contributors of testing cost is the test time, which increases along with the system complexity and has a significant impact on the time-to-market of final products.

While the semiconductor industry steadily follows Moore's law [Moore. 1965], the time between technology nodes has been significantly shortened, exacerbating the time-to-market pressure. In order to improve the design productivity of highly complex electronic systems within a shortened time period, a module-based design methodology, referred to as the core-based system-on-chip, has been widely adopted by the industry. The core-based SoC design methodology integrates pre-designed and pre-verified intellectual property (IP) blocks, referred to as cores, into a single silicon die.

Naturally, the testing of modern SoCs inherits the modular design style, making the test of cores to be independent from each other. Nonetheless, the modular SoC test becomes difficult and expensive, due to inefficient test access mechanisms (TAMs), large volume of test data, high power consumption, and high temperature. The long test application time (TAT) is one of the major contributors to the total

testing cost. Several techniques have been proposed to reduce the TAT. Firstly, advanced automatic test-pattern generation (ATPG) tools are used to generate more efficient test patterns. Secondly, efficient test scheduling techniques which schedule tests in parallel are employed to increase the test concurrency and to reduce the TAT. Thirdly, design-for-test (DFT) techniques, such as built-in self-test (BIST), are used to enhance the testability of circuits and reduce the TAT via higher test speed.

Although the proposed techniques reduce the TAT effectively, they increase the power consumption during test. Applying test patterns to the circuits under test cause a substantial increase of switching activity in the circuitry, especially in parallel testing or at-speed testing. This leads to the fact that more power is dissipated in circuits in testing mode than in normal functional mode. The substantially increased power consumption during test poses several problems, such as power supply noise, IR-drop and crosstalk which cause test fails and loss of yield. High power consumption also leads to high temperature which may damage the devices under test (DUTs). Thus, power consumption has to be taken into account for test time reduction and test scheduling methods.

As the process technology goes into the nanometer regime, the power density further increases along with the integration density. In the ICs manufactured with nanometer technology, taking the heat away from the chip becomes more difficult. This makes the high temperature problem more severe for the testing of the latest generation of SoCs. Therefore, test scheduling for SoC should also aim to avoid high operating temperature that may lead to permanent damage to the DUTs. More exactly, the temperature of SoC cores has to be strictly kept below a certain temperature limit and under such a constraint the TAT should be minimized.

Furthermore, testing ICs at different temperatures becomes necessary for current and future technologies. This is because the occurrence of parametric failures arises rapidly due to widely distributed process variations and the wide spectrum of subtle defects

introduced by new manufacturing processes and materials [Segura, et al. 2004].

The existence of complicated temperature dependences and defect-induced parametric failures indicates that we need to test a chip at multiple temperatures. Multi-temperature testing aims to screen the chips having various defects that can only be efficiently sensitized at certain temperatures. Different tests may be needed and applied at different temperatures, and each test targets a particular type of defects that can be detected at a certain temperature interval. Alternatively, the same test can also be applied at different temperature intervals so that outliers can be screened through a comparison of the test results. A multi-temperature test needs substantially long TAT, since a uni-temperature test is already time consuming. The long TAT problem is further exacerbated when multi-temperature testing is combined with modular SoC testing. Therefore, we need efficient test scheduling methods to reduce the TAT of multi-temperature SoC tests.

In volume production tests, an IC is usually discarded as soon as a fault is detected. This test approach is referred to as abort-on-first-fail (AOFF). Using the AOFF test approach leads to a substantial decrease in the TATs of volume production tests. In order to further reduce the TAT, defect probabilities of individual cores can be utilized to generate efficient test schedules for SoC tests using the AOFF approach. The defect probabilities can be derived from the statistical analysis of the production process or generated based on inductive fault analysis.

To summarize, SoC testing is a difficult and challenging problem. Many issues should be considered, such as test application time, temperature, power consumption, and defect probabilities, which are the topics of this thesis.

1.2 Problem Formulation

In this thesis, we aim to minimize the TAT of core-based SoCs. We address three test time minimization problems concerning different trade-offs and constraints, and we use different test scheduling techniques to solve these problems. The formulations of the addressed problems are described as follows.

First, we address the test time minimization problem with constraints on the temperatures of the CUTs and on the width of the test-bus deployed for test-data transportation. In order to prevent the core temperatures from exceeding the temperature limits, an entire test set is divided into shorter test sequences between which cooling periods are introduced. Furthermore, the test sequences for different cores can be interleaved in order to improve the efficiency of the test schedule. Thus, the test time minimization problem is formulated as how to generate test schedules for the partitioned and interleaved test sets such that the TAT is minimized while the temperature and test-bus width constraints are satisfied.

Second, we address the test time minimization problem for multi-temperature testing. In multi-temperature testing, an IC is tested at different temperature levels in order to efficiently sensitize the temperature-dependent defects. We divide the temperature range into multiple intervals, and minimize the TAT within each temperature interval. For each interval, a temperature upper limit and lower limit are imposed. The test scheduling algorithm minimizes the TAT such that test patterns are applied to a CUT only when the temperature of the CUT remains in the temperature interval, and, at the same time, the test-bus width limit is satisfied.

The third problem that we deal with is how to minimize the TAT when an AOFF test approach is employed for core-based SoC testing. Using the AOFF test approach, the test process is terminated as soon as a fault is detected. The termination of the test process is considered as a random event which occurs with a certain probability. Thus, for volume production tests, we minimize the expected test application

time (ETAT), which is the mathematical expectation of the TAT. The ETAT is calculated according to a generated test schedule and the given defect probabilities of individual cores. In particular, we employ a hybrid BIST technique which combines both deterministic and pseudorandom tests for each core in an SoC. The test time minimization problem is formulated as follows. Given the defect probabilities of cores and the test sets for the hybrid BISTs, generate a test schedule such that the ETAT is minimized. A related problem is the minimization of test time for volume production tests with a power constraint. We formulate the problem as how to generate the test schedule with minimal ETAT and the power constraint is satisfied.

1.3 Contributions

The main contributions of this thesis are as follows. First, we propose a test set partitioning and interleaving (TSPI) technique for temperature aware SoC test scheduling. This technique assumes that a test bus is employed to transport test data. The limit of the test-bus width and the limits of the core temperatures are given as constraints. In order to avoid overheating the CUTs during test, a test set is partitioned into multiple test sequences and cooling periods are introduced between consecutive test sequences. The partitioned test sets are further interleaved in order to reduce the TAT and to utilize the test bus efficiently. We have proposed two approaches to solve the constrained test scheduling problem. Both approaches employ the TSPI technique. One approach assumes the lateral heat flow between cores can be ignored. We develop a constraint logic programming (CLP) model and a heuristic algorithm for test scheduling [He, et al. 2006b], [He, et al. 2007], [He. 2007], [He, et al. 2008b], [He, et al. 2010b]. The other approach assumes significant later thermal influence between cores. We propose a thermal-simulation driven test scheduling algorithm which performs thermal simulations to obtain instantaneous temperature values of the CUTs and uses a finite-state

machine (FSM) model to manage the temperatures of the CUTs in test scheduling [He, et al. 2008a].

Second, we propose a SoC test scheduling technique for multi-temperature testing. The proposed technique generates the shortest test schedule for applying SoC tests in different temperature intervals. This means that the test patterns should only be applied when the core temperature is within a certain interval. We use the TSPI technique, a FSM model, and heating sequences to manage the temperature of CUTs in test scheduling. A heuristic algorithm is developed to minimize the TAT [He, et al. 2010a].

Third, we propose a defect-probability driven SoC test scheduling technique based on the AOFF test approach and hybrid BIST architecture. In this technique, we use the ETAT as the cost function and we develop a heuristic algorithm to generate the test schedule with minimized ETAT [He, et al. 2004]. In order to avoid possible damage, test failures, and yield loss caused by the high test power consumption and high temperature, we propose a technique to generate the shortest test schedules with a power constraint [He, et al. 2005], [He, et al. 2006a], [He. 2007], [He, et al. 2009].

The publications that are relevant in the context of this thesis are listed as follows.

HE, Z., JERVAN, G., PENG, Z. AND ELES, P. 2004. Hybrid BIST Test Scheduling Based on Defect Probabilities. In *Proceedings of the 13th IEEE Asian Test Symposium*, Kenting, Taiwan, November 15 - November 17, pp. 230-235.

HE, Z., JERVAN, G., PENG, Z. AND ELES, P. 2005. Power-Constrained Hybrid BIST Test Scheduling in an Abort-on-First-Fail Test Environment. In *Proceedings of the 8th Euromicro Conference on Digital System Design*, Porto, Portugal, August 30 - September 3, pp. 83-86.

HE, Z., PENG, Z. AND ELES, P. 2006a. Power Constrained and Defect-Probability Driven SoC Test Scheduling with Test Set Partitioning. In *Proceedings of the 2006 Design, Automation and Test in Europe Conference*, Munich, Germany, March 6 - March 10, pp. 291-296.

HE, Z., PENG, Z., ELES, P., ROSINGER, P. AND AL-HASHIMI, B.M. 2006b. Thermal-Aware SoC Test Scheduling with Test Set Partitioning and Interleaving. In *Proceedings of the 21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, Arlington, Virginia, USA, October 4 - October 6, pp. 477-485.

HE, Z. 2007. System-on-Chip Test Scheduling with Defect-Probability and Temperature Considerations. Licentiate of Engineering. Thesis No. 1313. Linköping Studies in Science and Technology. Linköping University.

HE, Z., PENG, Z. AND ELES, P. 2007. A Heuristic for Thermal-Safe SoC Test Scheduling. In *Proceedings of the 2007 IEEE International Test Conference*, Santa Clara, California, USA, October 21 - October 26, pp. 1-10.

HE, Z., PENG, Z. AND ELES, P. 2008a. Simulation-Driven Thermal-Safe Test Time Minimization for System-on-Chip. In *Proceedings of the 17th IEEE Asian Test Symposium*, Sapporo, Japan, November 24 - November 27, pp. 283-288.

HE, Z., PENG, Z., ELES, P., ROSINGER, P. AND AL-HASHIMI, B.M. 2008b. Thermal-Aware SoC Test Scheduling with Test Set Partitioning and Interleaving. *Journal of Electronic Testing: Theory and Applications*, 24(1-3), pp. 247-257.

HE, Z., PENG, Z. AND ELES, P. 2009. Thermal-Aware Test Scheduling for Core-based SoC in an Abort-on-First-Fail Test Environment. In *Proceedings of the 12th Euromicro Conference on Digital System Design*, Patras, Greece, August 27 - August 29, pp. 239-246.

HE, Z., PENG, Z. AND ELES, P. 2010a. Multi-Temperature Testing for Core-based System-on-Chip. In *Proceedings of the 2010 Design, Automation and Test in Europe Conference*, Dresden, Germany, March 8 - March 12, pp. 208-213.

HE, Z., PENG, Z. AND ELES, P. 2010b. Thermal-Aware SoC Test Scheduling. (Book Chapter) In *Design and Test Technology for Dependable System-on-Chip,* R. UBAR, J. RAIK AND H.T. VIERHAUS, Eds. IGI Global.

1.4 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 presents the background and related work of core-based SoC testing. The generic design flow of electronic systems and the basic concepts of defects and testing are introduced. The SoC test architecture and test scheduling techniques are described. Power and temperature issues in SoC testing are discussed and related thermal modeling techniques are presented. The multi-temperature testing and AOFF test approach are also discussed.

Chapter 3 and Chapter 4 address the temperature aware SoC test time minimization problem. Different test scheduling techniques are proposed for two types of SoCs where the lateral thermal influence between cores is either negligible or should be considered, respectively.

Chapter 5 addresses the test time minimization problem for multi-temperature testing. A test scheduling technique is proposed to generate the shortest test schedule such that the test patterns are applied only when the temperature of each core is within an interval.

Chapter 6 and Chapter 7 address the test time minimization problem for volume production tests using the AOFF test approach. Defect-probability driven test scheduling techniques are proposed to minimize the ETAT with a power constraint.

Chapter 8 concludes the thesis and discusses possible directions of future work.

Chapter 2
Background and Related Work

This chapter presents the basic concepts of electronic system design and test, followed by a discussion on core-based SoC testing. The background and related work on test scheduling, power and temperature aware testing, multi-temperature testing, as well as the AOFF test approach are described.

2.1 Generic Design Flow

In order to manage the system complexity, the design of electronic systems has to be organized in a hierarchical approach which covers several levels of abstraction. In general, there are four abstraction levels, referred to as the system level, register-transfer (RT) level, logic level, circuit level, in a top-down order. Figure 2.1, often referred to as “Gajski and Kuhn’s Y-chart” [Gajski, et al. 1983], illustrates a structured view on the electronic systems design space, where the four levels of abstraction are categorized into three domains, namely the behavioral, structural and physical (or geometry) domain.

In the different domains, designers have a different perspective on their design tasks, as listed in Table 2.1. A typical design flow is depicted in Figure 2.2 [Devadas, et al. 1994].

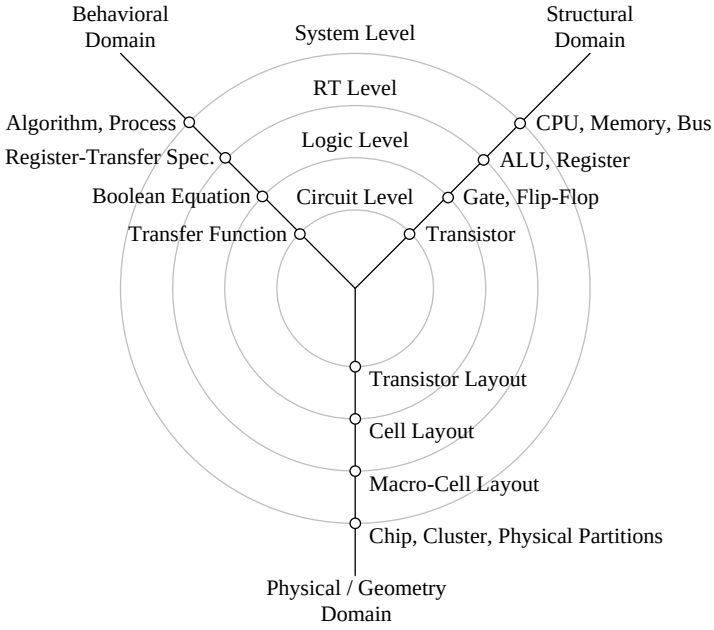


Figure 2.1: Visualization of electronic systems design space

Table 2.1: Design tasks in different domains

Domains / Abs. Levels	Behavioral Domain	Structural Domain	Physical/Geometry Domain
System Level	Algorithm, Process	CPU, Memory, Bus	Chip, Cluster, Physical Partitions
RT level	RT Specification	ALU, Register	Macro-Cell Layout
Logic Level	Boolean Equation	Gate, Flip-Flop	Cell Layout
Circuit Level	Transfer Function	Transistor	Transistor Layout

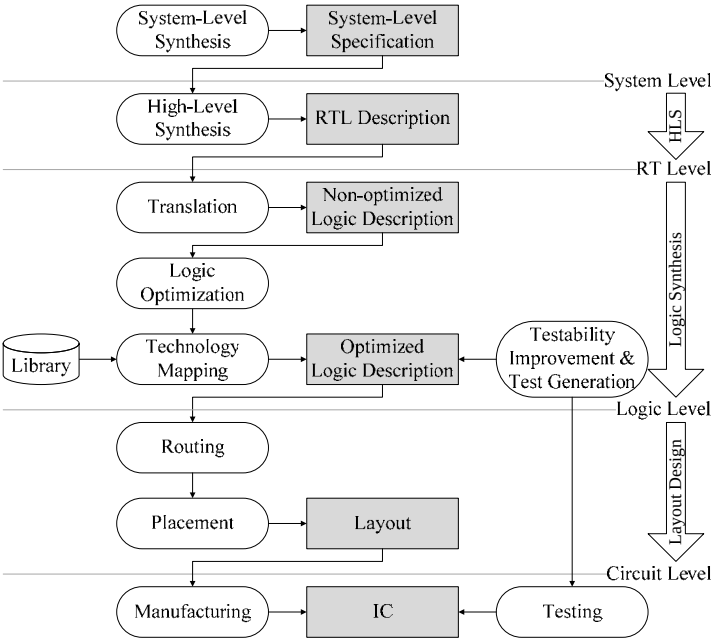


Figure 2.2: A typical electronic systems design flow

Here, a synthesis step is referred to as a transformation of a design from a higher level of abstraction into a lower level of abstraction, or from one domain to another domain. Each step in the design flow is explained as follows.

(1) System-Level Synthesis: The specification of a electronic system is usually given as a description of the system functionality and a set of design constraints. In this step, the system specification is analyzed and a behavioral description is written in a hardware description language or natural language.

(2) High-Level Synthesis: In this step, the system-level specification is transformed into a description of RT-level (RTL) components such as arithmetic logic units (ALUs) and registers. The basic components in the RTL design implement the given system-level specification. In order to obtain the RTL design, the high-level synthesis usually consists of the following steps [Elliott. 1999]:

derivation of a control/data-flow graph (CDFG), operation scheduling, resource allocation and binding, derivation of the RTL data-path structure, and description of a controller such as a FSM.

(3) Logic Synthesis: In this step, a RTL design is first translated into a set of logic functions. Thereafter, the translated RTL design is optimized according to different requirements given by the designer. The optimized design is then mapped to a netlist of logic gates, using a technology library provided by a vendor.

(4) Circuit-Level Synthesis: In this step, the logic netlist is transformed into the transistor implementation of the circuit.

(5) Layout Design: In this step, the circuits are mapped to the silicon implementation with routing and placement design.

As illustrated in Figure 2.2, when the logic netlist has been obtained, the testability improvement and test generation (TG) are performed using design automation tools. After fabrication, each IC is tested using the generated test patterns and the qualified parts are delivered to customers.

2.2 Faults and Testing

In general, testing is a method to assure correct behavior of a system. Usually, a test exercises the system with a set of stimuli and analyzes the system responses to see if they are exactly the same as expected. Electronic testing is an experimental approach in which an electronic system is exercised with test stimuli and the system response is analyzed and compared with the expected response in order to ascertain the correctness of the system behavior.

In this thesis, an instance of incorrect system operation is referred to as an *error*. According to different causes, errors can be further categorized as design errors, fabrication errors, fabrication defects, and physical failures [Abramovici, et al. 1994]. The different types of error are defined as follows.

Design errors can be incomplete or inconsistent specifications, incorrect mapping between different levels of design, or violations of design rules. *Fabrication errors* can be wrong components, incorrect wiring, shorts caused by improper soldering, etc. *Fabrication defects* are not directly attributed to human errors, but rather result from an imperfect manufacturing process. Examples of fabrication defects are shorts and opens in ICs, improper doping profiles, mask alignment errors, and poor encapsulation. *Physical failures* occur during the lifetime of a system due to component wear-out and/or environmental factors. Examples of physical failures are metal connectors thinning out with time, broken metal line due to electron migration or corrosion, etc. Some environmental factors, such as temperature, humidity, and vibrations, accelerate the aging of components. Other environmental factors, such as cosmic radiation and particles, may induce failures in ICs immediately [Abramovici, et al. 1994].

Fabrication errors, fabrication defects, and physical failures are collectively referred to as *physical faults*. In the context of this thesis, testing is referred to as a quality-assurance means that targets physical faults. According to the stability in time, physical faults can be categorized as (1) permanent faults, which are always present after their occurrence; (2) intermittent faults, which only exist during some time intervals; (3) transient faults, which are typically characterized by one-time occurrence and are caused by a temporary change in environmental factors or radiations [Abramovici, et al. 1994].

In general, a direct mathematical treatment of testing and diagnosis is not applicable to physical faults. The solution is to deal with logical faults, which are a convenient representation of the effect of the physical faults on the operation of the system. A logic fault can be detected by observing an error caused by it, which is usually referred to as a fault effect. The basic assumptions regarding the nature of logical faults are referred to as a fault model. Different fault models are proposed and employed to deal with different types of faults, such as static faults, delay faults, bridging faults, etc. A widely used fault model is the stuck-at fault model which assumes that a

single wire is permanently “stuck” at the logic one or logic zero value [Abramovici, et al. 1994].

2.3 Core-based SoC Testing

Scaling of process technology has enabled a dramatic increase of the integration density, which enables more and more functionality to be integrated into a single chip. With the increasing system performance, the design complexity has also been growing steadily. A critical challenge to electronic engineers is that the shorter life cycle of an electronic system has to compete with its longer design cycle. Therefore, more efficient hierarchical design methodologies, such as the core-based SoC design methodology [Murray, et al. 1996], [Zorian, et al. 1999], have to be deployed in order to reduce the time-to-market.

A common approach to modern core-based SoC design reuses pre-designed and pre-verified IP cores that are provided by different vendors. IP cores are integrated into the system which is manufactured on a single silicon die. An abstract example of an SoC design is depicted in Figure 2.3. The SoC consists of several IP cores with different functionality and a user-defined logic (UDL) module. In general, IP cores of SoCs can be processors (e.g. microcontroller, DSP), memory subsystems (e.g. RAM/ROM, Flash Memory), bus infrastructure (e.g. system bus, peripheral bus), I/O subsystems (e.g. USB, FireWire, Ethernet, DMA), analog and mixed-signal subsystems (e.g. PWM, A/D-D/A, RF), and peripheral subsystems (e.g. audio, video, graphic, display, camera). The UDL modules are usually used to “glue” the IP cores for the intended system.

In order to test individual cores in an SoC, a test architecture consisting of certain resources has to be available. The test architecture for SoCs usually includes the test sources, test sinks, and test access mechanisms (TAMs). Figure 2.4 illustrates an example of a generic core-based SoC test architecture.

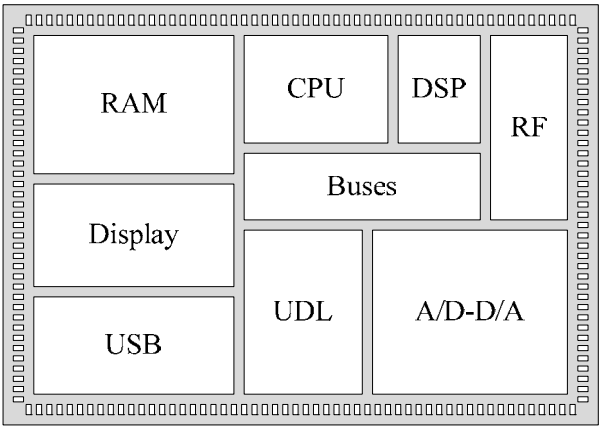


Figure 2.3: An IP core-based SoC example

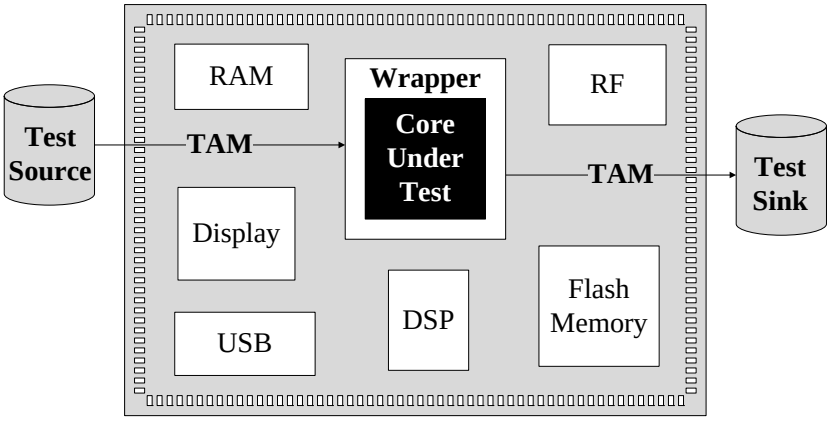


Figure 2.4: Generic core-based SoC test architecture

A test source is a test-pattern provider which can be either external or on chip. A typical external test source is an automatic test equipment (ATE) in which a local memory stores the generated test patterns. An on-chip test source can be a ROM which stores already generated test patterns, a counter, or a linear feedback shift register (LFSR) used for test pattern generation in BIST.

A test sink is a test response/signature analyzer that detects faults by comparing test responses/signatures with the expected ones. An ATE can be an external test sink that analyzes the test responses/signatures transported from the DUTs. A test sink can also be on chip, such as single-input signature register (SISR) or multi-input signature register (MISR) used for signature analysis in BIST.

A TAM is an infrastructure designed for test data transportation. It is often used to transport test patterns from the test source to the CUTs and to transport test responses/signatures from the CUTs to the test sink. A TAM can be a bus infrastructure, such as a reusable functional bus, e.g. advanced microprocessor bus architecture (AMBA) [Flynn. 1997], [Harrod. 1999], reuse of addressable system bus (RASBuS) [Hwang, et al. 2001] etc, or a dedicated test bus, e.g. flexible-width test bus architecture [Iyengar, et al. 2003]. A TAM can also be dedicated wire connections, e.g. direct access test scheme (DATS) [Immaneni, et al. 1990], multiplexing/DaisyChain/distributed test architecture [Aerts, et al. 1998], TestRail [Marinissen, et al. 1998], etc.

In an SoC test architecture, a wrapper, which is a thin shell surrounding a core, is usually designed to switch the CUT between different modes, such as normal functional, internal test, and external test modes [Marinissen, et al. 2000]. The TAM together with the wrappers are usually referred to as test access infrastructure (TAI).

An example of the test architecture for external SoC tests is depicted in Figure 2.5. In this example, an ATE consisting of a test controller and a local memory serves as an external tester. The test patterns and a test schedule are stored in the tester memory. When the test starts, the test patterns are transported to the cores through a test bus. After activating the test patterns, the captured test responses are

transported to the ATE through the test bus. The ATE can be replaced by an embedded tester integrated in the chip. Figure 2.6 depicts an example of the test architecture with an embedded tester for external tests.

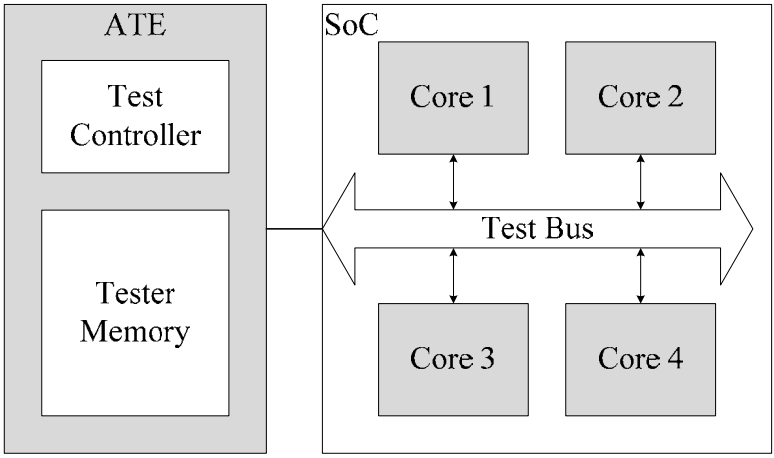


Figure 2.5: Test architecture for external tests using an ATE

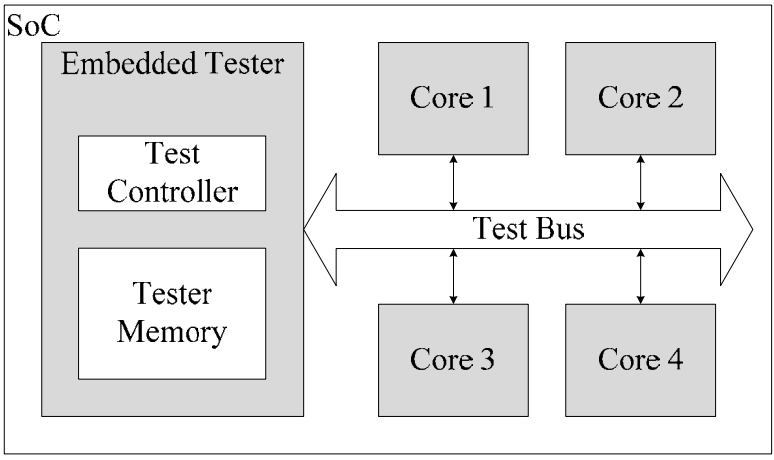


Figure 2.6: Test architecture for external tests using an embedded tester

As the number of cores of an SoC has been increasing along with the rapid advances of technology, the amount of required test data for SoC testing is growing substantially. This demands a large size of tester memory to be used for tests. Moreover, an external test is usually applied at relatively low speed due to the limited TAM width, and therefore results in a long TAT.

One of the solutions to this problem is to use built-in self-test, which generates pseudorandom test patterns and compact test responses into a signature inside the chip. The advantage of BIST is that it can be applied at high speed. However, due to the existence of random-pattern-resistant faults, BIST usually needs much more test patterns in order to achieve the same level of fault coverage as an external test using ATE.

In order to avoid the disadvantages of both external test and BIST, a hybrid approach has been proposed as a complement of the two types of tests, referred to as hybrid BIST [Hellebrand, et al. 1992], [Touba, et al. 1995], [Sugihara, et al. 2000], [Jervan, et al. 2000]. In hybrid BIST, a test set consists of both pseudorandom and deterministic test patterns. Such a hybrid approach reduces the memory requirements compared to the pure deterministic testing, and it provides higher fault coverage and requires less test data compared to the stand-alone BIST solution.

An example of the test architecture for hybrid BIST is depicted in Figure 2.7. In this example, an embedded tester consisting of a test controller and a local memory is integrated in the chip. The generated deterministic test patterns and a test schedule are stored in the local memory of the tester. When the test starts, the deterministic test patterns are transported to the cores through a test bus. Each core has a dedicated BIST circuit that can generate and apply pseudorandom test patterns at speed. The test controller is supposed to control both the deterministic and pseudorandom tests according to the test schedule.

In order to reduce the testing cost, a wide spectra of research has been carried out on several challenging issues, including test scheduling, power aware testing, temperature aware testing, AOFF

test approach. The back ground and related work in these areas are presented in the following sections of this chapter.

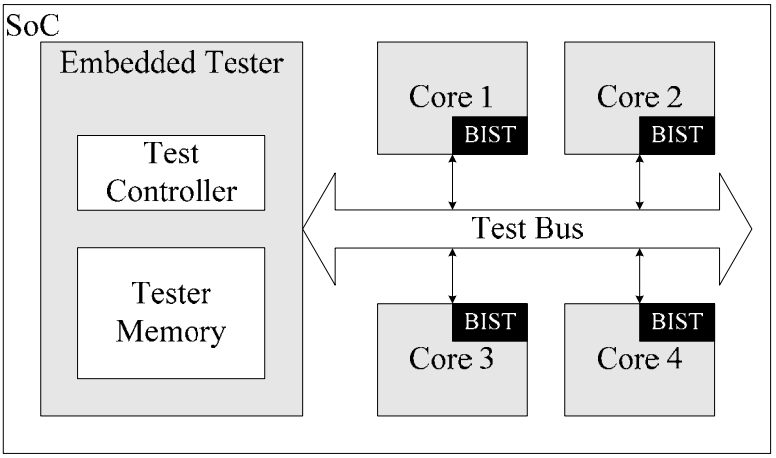


Figure 2.7: Test architecture for hybrid BIST

2.4 Test Scheduling

Test scheduling is a process of deciding the start times and durations of tests as well as the means to utilize the resources for the tests. Usually, test scheduling aims to reduce the TAT through efficiently planning. In recent years, different test scheduling techniques have been proposed.

Non-partitioned test scheduling is proposed in [Zorian. 1993] and [Chou, et al. 1997]. This technique assumes that tests are scheduled into different sessions, which is defined as an uninterrupted period of time spent on testing. Tests have to be applied without interruption and no new test can be started before all the tests scheduled in the same test session are finished. Non-partitioned test scheduling results in long TATs. Recently, partitioned test scheduling techniques have been proposed in order to reduce the TAT.

Partitioned test scheduling is proposed in [Muresan, et al. 2000]. It can substantially improve the efficiency of the test schedules by allowing tests to be started with no need to wait for other tests to finish. This means that the concept of the test session no longer exists in the partitioned test scheduling technique. In order to facilitate this technique, a more complex test controller has to be designed in order to enable a test to start at arbitrary time moments.

A generalized core-based SoC test scheduling problem was addressed in [Chakrabarty. 2000a]. The problem is formulated as follows. Given a set of test resources (TAMs, BIST circuits, etc.), minimize the TAT by determining the start time of each partitioned test. The author shows that the formulated problem is NP-complete and provides a mixed-integer linear programming (MILP) model to obtain the optimal schedule. For large SoC designs, the MILP model needs a substantially long optimization time and may not be feasible to obtain the optimal solution. Therefore, the author develops a heuristic algorithm to generate efficient test schedules with low computational cost.

Preemptive test scheduling is proposed in [Iyengar, et al. 2002]. Similar test scheduling technique is also proposed in [Larsson, et al. 2002]. This technique assumes that a test can be halted for a period of time and then restarted later. The proposed preemptive test scheduling technique generates shorter test schedules than non-preemptive test scheduling. However, preemptive testing needs a complicated test controller and an advanced TAM. Moreover, it cannot be adopted for certain types of tests such as BIST.

2.5 Power and Temperature Issues

Scaling of the complementary metal-oxide-semiconductor (CMOS) technology has enabled the industry to improve the speed and performance of ICs. While all the physical dimensions of a transistor are scaled down, the device area is reduced. At the same time,

designers tend to add more functionality into chips and to build more complex circuits, leading to increasing die area to accommodate more transistors [Vassighi, et al. 2006]. It is shown in [Rabaey, et al. 2003] that the die area sizes of Intel processors increase approximately 7% per year, and the number of transistors are doubled per generation. The latest microprocessors already integrate billions of transistors.

With technology scaling, the power consumption of high-performance chips increases exponentially, especially for the chips manufactured with deep-submicron technology. The main reason is that the scaling of the threshold voltage V_{TH} causes an increase in sub-threshold leakage current [Rabaey, et al. 2003].

With technology scaling, not only the total power consumption but also the power density of chips increases [Borkar. 1999], [Gunther, et al. 2001]. The power density of a chip is defined as the power dissipated by the chip per unit area under nominal frequency and normal operating conditions. The reason for the increasing power density is that the positive supply voltage V_{DD} and the saturated drain current I_{DSAT} are scaling at a lower rate than the device area size [Vassighi, et al. 2006].

The increasing power consumption and power density result in higher junction temperature [Vassighi, et al. 2006], [Mahajan. 2002], [Skadron, et al. 2004], especially in high-performance processors and application-specific integrated circuits (ASICs). Junction temperature is one of the key parameters of CMOS devices, as it affects the performance, power consumption, and reliability of the ICs [Segura, et al. 2004], [Vassighi, et al. 2006].

Carrier mobility decreases as temperature increases, because carriers collide with the Si-crystal lattice more frequently at a higher junction temperature. As a consequence, the driving currents of transistors decrease with reduced carrier mobility, which causes a degradation of the device performance. Similar effects occur in the thin interconnect metal lines using aluminum or copper process. At a higher temperature, the metal resistivity increases, leading to higher interconnect resistance. Thus, circuit performance degradation is often

encountered when operating temperature increases. The performance degradation should be avoided for both normal functional and testing conditions. In the normal functional mode, the performance of an IC directly affects the system efficiency. In the testing mode, the performance degradation due to high junction temperature may fail the test and cause loss of yield.

The elevation of junction temperature results in an increase in leakage current and higher device power consumption. The elevated power consumption in turn increases the junction temperature [Vassighi, et al. 2006]. The positive feedback between the leakage current and junction temperature may lead a chip to thermal runaway in extreme cases. When a chip is in a stress condition, such as a burn-in test where chips are tested with purposely elevated power supply voltage and junction temperature, the chance of thermal runaway is much higher. For ICs manufactured with nanometer technology, the situation of the positive feedback is exacerbated and thermal runaway is more likely to happen.

Another issue related to junction temperature is the long-term reliability of ICs. Many failure mechanisms, such as electron migration, gate oxide breakdown, hot electron effects, negative bias temperature instability, etc., are accelerated when junction temperature is elevated [Segura, et al. 2004]. In order to maintain the device reliability and the lifetime of ICs, it is very important to efficiently and safely manage the transistor junction temperature and operating temperature of other parts in ICs. It is reported that even a small variation of junction temperature (10–15°C) may result in a factor of two times reduction in device lifetime [Vassighi, et al. 2006].

According to the above discussion, one can see that it is critical to develop efficient power and temperature analysis and management techniques for the design and test of modern ICs.

2.6 Power Aware Testing

Compared to the normal functional mode, ICs dissipate more power during test [Zorian. 1993], [Pouya, et al. 2000], [Girard. 2000], [Bushnell, et al. 2000], [Shi, et al. 2004]. It is reported in [Shi, et al. 2004] that the average power dissipated in scan-based testing can be 3 times as the power consumed during normal functional operations, and the peak power consumption can be 30 times as that in normal functional mode.

The high test power is because a larger amount of switching activity occurs when applying test patterns to the circuit under test. There are several explanations to the increase of power consumption in the testing mode [Wang, et al. 2007]. First, ATPG tools tend to generate test patterns with a higher toggle rate in order to reduce the total number of test patterns and the TAT. This results in a much higher switching activity in the testing mode. Second, in order to reduce TATs, SoC tests often employ parallel testing which substantially increase the power dissipation during test. Third, some circuits, e.g. DTF circuitry, only work in the testing mode and only contribute to the test power consumption. Fourth, the correlation between consecutive test patterns is usually much lower than that between successive functional input vectors [Wang, et al. 1997]. There is no definite correlation between successive deterministic test patterns for scan-based tests or pseudorandom test patterns for BISTs [Wang, et al. 2007]. The low correlation between consecutive input vectors results in excessive higher switching activity and consequently extra power dissipation. Last, when scan-based testing is employed, the power dissipation is even higher because of the circuit is excessively stimulated while the test patterns are shifted into the scan cells [Bushnell, et al. 2000].

High power dissipation during test results in several critical problems related to the reliability and safety of the circuit under test. One significant issue is the increase of power supply noise, which is proportional to the inductance of a power line and to the magnitude of

the variation of the current flowing through the power line [Wang, et al. 1997]. The excessive power supply noise can erroneously change the logic state of circuit nodes, resulting in good dies failing the test and consequently loss of yield. A similar type of noise, the voltage glitch, also increases with switching activity and can change the logic states of circuit nodes or flip-flops, leading to yield loss. Another problem caused by high switching activity during test is the IR-drop, which refers to the amount of decrease/increase in the power/ground rail voltage [Wang, et al. 2007]. With high current in the circuit under test, the voltages at gates may be reduced and will cause these gates to exhibit higher delays, leading to fails in speed-related tests and yield loss [Shi, et al. 2004]. A third problem caused by the high test power consumption is the high junction temperature which has large impacts on the ICs [Vassighi, et al. 2006].

In order to prevent high power consumption during test, some techniques have been proposed. Low power test synthesis and DFT targeting RTL structures is one of the solutions, for example, low-power scan chain design [Gerstendörfer, et al. 2000], [Rosinger, et al. 2004], [Saxena, et al. 2001], scan cell and test pattern reordering [Girard, et al. 1998], [Elliott. 1999], [Rosinger, et al. 2002]. Although low power DFT can reduce the power consumption, this technique usually adds extra hardware into the design and therefore it can increase the circuit delay as well as the cost of every single chip. Power-constrained test scheduling is another approach to tackle the high test power consumption problem [Chou, et al. 1997], [Chakrabarty. 2000b], [Muresan, et al. 2000], [Ravikumar, et al. 2000], [Iyengar, et al. 2002], [Larsson, et al. 2006], [He, et al. 2006a]. The proposed techniques minimize the TAT under a fixed power envelope restriction. In general, the power constrained test scheduling problem is related to bin-packing or two-dimensional (2D) rectangle packing (RP) problem [Baker, et al. 1980], [Dyckhoff. 1990], [Dell'Amico, et al. 1997], [Lesh, et al. 2004], [Lesh, et al. 2005], [Korf. 2003], [Korf. 2004], which is NP-complete. Heuristic algorithms are often proposed to solve the power constrained test time minimization problems.

2.7 Temperature Aware Testing

Although the power-aware test techniques are efficient to solve the high power consumption problem, they cannot completely avoid the overheating problem because of the complex thermal phenomenon [Rosinger, et al. 2006] in modern electronic chips. Advanced cooling techniques are effective to solve the high temperature problems. However, they either substantially increase the system cost or usually require large space. Other techniques such as lower frequency and reduced speed do help to avoid unexpectedly high temperature during test, but they result in excessively long TATs and are not applicable to at-speed tests. In order to test new generations of SoCs safely and efficiently, novel and advanced testing techniques are required.

Recently, temperature aware testing [Tadayon. 2000] has attracted many research interests. Liu, Veeraraghavan, and Iyengar address the problem of the high temperature during test, and propose a test scheduling technique that considers temperature constraints [Liu, et al. 2005]. The proposed technique aims to generate thermal-safe test schedules and to reduce the hot-spot temperature such that the heat is more evenly distributed across the die. In this technique, the floor plan of the chip is used to guide test scheduling.

In [Rosinger, et al. 2006], Rosinger, Al-Hashimi, and Chakrabarty indicate that the non-uniform distribution of the heat results in hot spots on the die and therefore the power constrained test scheduling techniques cannot guarantee the thermal safety. The authors proposed a simplified thermal-cost model and an approach using the core adjacency information to guide test scheduling. The proposed technique can generate the minimized thermal-safe test schedules.

Yu, Yoneda, Chakrabarty, and Fujiwara address the temperature aware TAM/wrapper co-optimization problem in [Yu, et al. 2007]. The authors propose a test scheduling approach to generate efficient test schedules which are also thermal safe. The proposed approach uses a thermal-cost model improved from the one proposed in [Rosinger, et al. 2006], and employs a bin-packing algorithm to

minimize the TAT and at the same time to satisfy the temperature constraints.

Although these proposed approaches generate efficient test schedules, they make strong and simplifying assumption that a CUT is never overheated during the application of a single test set. This assumption may not be valid for testing of high performance SoCs in which the temperature of CUTs may exceed the temperature limit before a single test is completed. In this thesis, we assume that before the completion of a single test, the temperature of a CUT may exceed a temperature limit beyond which the core can be damaged.

2.8 Thermal Modeling

In order to obtain the temperature of an IC, thermal modeling techniques are often used. Thermal modeling is a technique that provides mathematical models to predict the temperature of objects. A thermal model usually considers the thermal resistance and thermal capacitance of the object to its surroundings, as well as the heat generated in and removed from the object.

The relationship between the ambient temperature, the average junction temperature, and the power dissipation of an IC is often described as:

$$T_j = T_a + P_{chip} \times R_{ja} \tag{2.1}$$

where T_a is the ambient temperature, P_{chip} is the total power dissipation of the chip, and R_{ja} is the junction-to-ambient thermal resistance. Using a three-dimensional heat flow equation, the junction-to-ambient thermal resistance of a metal-oxide-semiconductor field-effect transistor (MOSFET) can be calculated according to the geometrical parameters of the MOSFET, as shown in Equation (2.2) [Rinaldi. 2000].

$$R_{ja} = \frac{1}{2\pi k}\left[\frac{1}{L}\ln\left(\frac{\sqrt{W^2+L^2}+L}{\sqrt{W^2+L^2}-L}\right)+\frac{1}{W}\ln\left(\frac{\sqrt{W^2+L^2}+W}{\sqrt{W^2+L^2}-W}\right)\right] \quad (2.2)$$

where k is the thermal conductivity of silicon and its typical value is 1.5×10^{-4} W/mm°C [Rinaldi. 2001]. W and L are the channel width and length, respectively.

In an IC, every physical component acts as a heat storage capacitor with a certain thermal capacitance, denoted with C_{th}. At the same time, a physical component also acts as a heat resistor with a certain thermal resistance, denoted with R_{th}, transferring heat through other components towards the ambient. Equation (2.3) models an one-dimensional heat conduction in a homogeneous isotropic material.

$$\frac{\partial^2 T}{\partial x^2} = \frac{c\cdot\rho}{\lambda_{th}}\cdot\frac{\partial y}{\partial x} \quad (2.3)$$

where λ_{th} is the heat conductance, c is the thermal capacitance, ρ is the density of the material, T is the temperature, and x is the direction of the heat flow in the material.

The thermal model described in Equation (2.3) is equivalent to the electrical model, given in Equation (2.4), for the transmission of electric-magnetic wave in a solid line [Vassighi, et al. 2006].

$$\frac{\partial^2 U}{\partial x^2} = C\cdot R\cdot\frac{\partial U}{\partial x} \quad (2.4)$$

where C is the capacitance per unit area, R is the resistance per unit area, and U is the voltage. It can be seen that there is a duality between the electrical and thermal models. Therefore, the heat conduction process can be modeled by a transmission-line-equivalent circuit consisting of only resistors and capacitors, as illustrated in Figure 2.8 [Vassighi, et al. 2006]. Table 2.2 lists the equivalent parameters between the electrical and thermal models [Vassighi, et al. 2006].

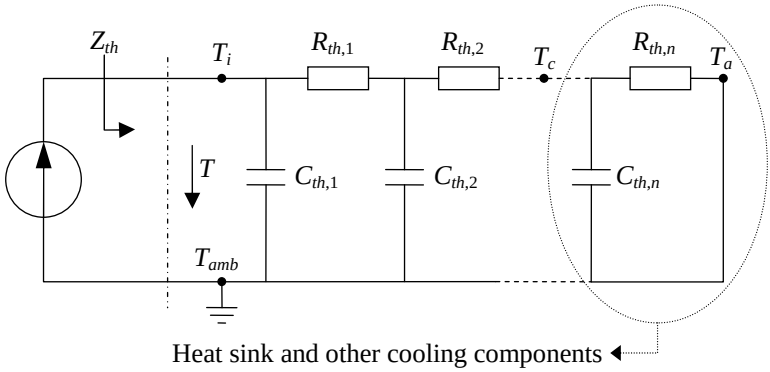


Figure 2.8: An electro-thermal model

Table 2.2: Duality between the electrical and thermal models

Thermal Model		Electrical Model	
Temperature	T (in K)	Voltage	U (in V)
Heat Flow	P (in W)	Current	I (in A)
Thermal Resistance	R_{th} (in K/W)	Electrical Resistance	R (in V/A)
Thermal Capacitance	C_{th} (in Ws/K)	Electrical Capacitance	C (in As/V)

Accurate temperature models are needed at all abstraction levels, since power consumption and performance are strongly dependent on the thermal map of a specific implementation or architecture [Vassighi, et al. 2006]. For the sake of shortening the time-to-market, early design optimization at system level plays a very important role. Compared to thermal models at lower abstraction levels, architectural-level thermal models need less computation recourses in order to be solved. At the same time, such models produce sufficiently accurate results in the context of system-level design optimization [Huang, et al. 2004]. Before the computation of temperature values, architectural-level thermal modeling [Huang, et al. 2006], [Yang, et al. 2007] needs

the following two basic steps: (1) floor plan extraction; (2) thermal resistance-capacitance (RC) modeling.

Skadron et al. have investigated architectural-level electro-thermal modeling and have implemented a thermal simulator, HotSpot [Huang, et al. 2006], to calculate transient as well as steady-state temperatures of functional units at the architecture level. Similar work has also been carried on by Li et al., and a thermal simulator, ISAC [Yang, et al. 2007], has been developed.

In architectural-level thermal modeling, a floor plan is modeled as a set of blocks, each of which is further divided into a matrix of sub-blocks. Every sub-block corresponds to a set of functional units such as ALU, FPU, cache memory, etc. The floor plan is specified by matrices of the adjacency of the sub-blocks. In SoC design and test, it is common practice to consider each core as such a sub-block [Zorian, et al. 1999], [Marinissen, et al. 2000].

When the floor plan is extracted, the thermal resistance R_{th} and thermal capacitance C_{th} are calculated according to the following two simplifying assumptions: (1) the thermal resistance is proportional to the thickness of the material and inversely proportional to the size of the cross-sectional area across which the heat is transferred; (2) the thermal capacitance is proportional to the thickness of the material and proportional to the size of the cross-sectional area. Thus, the thermal resistance and thermal capacitance can be derived according to Equations (2.5) and (2.6), respectively [Vassighi, et al. 2006].

$$R_{th} = t / (k \times A) \tag{2.5}$$

$$C_{th} = c \times t \times A \tag{2.6}$$

where t is the thickness of the material, A is the size of the cross-sectional area of the material, k is the thermal conductivity of the material per unit volume, and c is the thermal capacitance per unit volume. Nominal values of k, at 85°C, are 100 W/m^{3}K for silicon and 400 W/m^{3}K for copper. Nominal values of c are 1.75×10^6 J/m^{3}K for silicon and 3.55×10^6 J/m^{3}K for copper.

Using the area size, thermal resistance and thermal capacitance of each sub-block in the package, an equivalent electrical circuit is derived to model the dynamic heat flows in the chip. The dissipated power in each sub-block is given as an input to the thermal model in every time step. Thereafter, the average temperature of each sub-block over the time interval is calculated using numerical computation methods.

In this thesis, we have used the architecture-level thermal simulators, either HotSpot or ISAC, for temperature aware test scheduling in different contexts. We assume nominal configurations of modern IC dies and packages for thermal simulations. The thermal simulator takes the floor plan of a chip and the power consumption of every core as inputs, and computes the temperature of each core in every simulation cycle.

2.9 Multi-Temperature Testing

Environment-sensitive defects often cause parametric failures that are more and more observed in ICs manufactured with nanometer technologies. These environmental parameters include power supply voltage, clock frequency, temperature, radiation, etc. In recent years, concerns regarding parametric failures increase rapidly due to widely distributed process variations and the wide spectrum of subtle defects introduced by new manufacturing processes and materials [Segura, et al. 2004], [Needham, et al. 1998], [Nigh, et al. 1998], [Montanes, et al. 2002].

Some defects are sensitive to a certain temperature level. For example, metal interconnect defects may pass a delay test at nominal temperature but fail the same test at a high temperature. This indicates that speed tests, such as maximum-frequency test, referred to as F_{max} test, and transition delay test, should usually be applied at a high temperature in order to detect these temperature-dependent defects.

In [Singer, et al. 2009], a closer investigation on the correlation between the maximum frequency and temperature was performed for ICs powered by ultra-low supply voltages. It shows that there exists a turnaround temperature point above which the maximum frequency no longer decreases but rather increases. This means that applying a speed test at a high temperature may not screen the defective chips because of the improper temperature setting for the test. Therefore, for those types of ICs, F_{max} tests or transition delay tests should be applied at a critical temperature which can be obtained by characterization.

Parametric failures induced by subtle defects, such as resistive vias/contacts and weak opens, are hard to detect even when the circuit operates with the lowest performance under the worst environmental condition. In these cases, a speed test needs to be applied at two temperatures (hot/cold) and at a particular frequency [Needham, et al. 1998]. The defective chips can be screened as outliers by comparing the test results at the two different temperatures.

The following sub-sections explain the temperature effects on CMOS circuits as well as the cause of temperature-dependent defects and parametric failures.

2.9.1 Temperature Effects in CMOS Circuits

As one of the environmental parameters, operating temperature has a large impact on the electrical properties of transistors and their interconnects [Segura, et al. 2004]. Carrier mobility usually decreases at high temperature since the carriers collide with the Si-crystal lattice more frequently. Similar effects occur in the thin metal lines connecting the transistors, increasing the interconnect resistance. Thus, performance degradation is often encountered at a high operating temperature, leading design and test efforts to focus on the high-temperature scenarios. In practice, an IC is often tested at high temperatures in order to guarantee the functionality at all temperatures that may appear in the field.

Another temperature-dependent parameter is the transistor threshold voltage, which increases with rising temperature. The increasing threshold voltage results in an elevated drain current, which compensates for the degraded circuit performance due to the reduced carrier mobility and interconnect resistance. The threshold voltage dominates the performance after the operating temperature exceeds a certain point, referred to as the CMOS zero-temperature-coefficient (ZTC) point [Filanovsky, et al. 2001], meaning that the circuit performance increases with further rising temperature. Thus, there exist two temperature dependence regions [Filanovsky, et al. 2001], [Calhoun, et al. 2006], [Wolpert, et al. 2009], a normal dependence region in which the circuit delay increases with rising temperature, and a reverse dependence region in which the circuit delay decreases with rising temperature. Figure 6.1 illustrates circuit delay variation in the normal and reverse dependence regions [Wolpert, et al. 2009]. This phenomenon is usually observed in low-power designs with ultra-low supply voltage. It infers that, for those circuits in which reverse temperature dependence is observed, a delay test should be applied at the temperature point between the normal and reverse regions where the circuit delay is the largest.

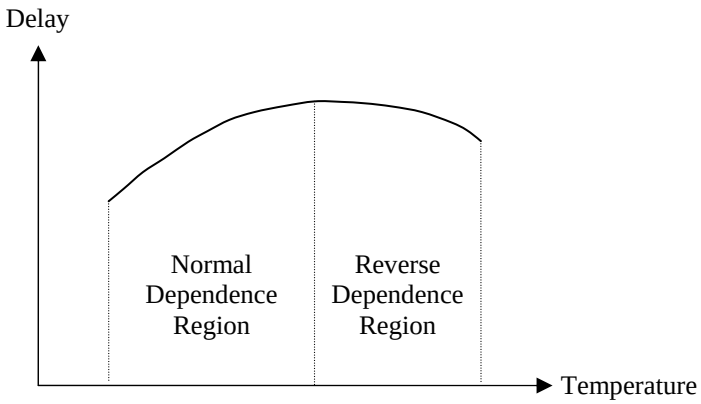


Figure 2.9: Normal and reverse temperature dependence regions

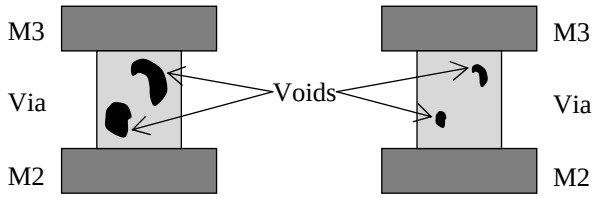
2.9.2 Subtle Defects and Parametric Failures

ICs manufactured with nanometer technology, typically below 45nm, encounter more reliability problems and parametric failures caused by widely distributed variations and a wide spectrum of subtle defects. Defect-induced parametric failure mechanisms include weak interconnect opens, resistive vias and contacts, metal mouse bites and metal slivers, with the first two as major causes [Segura, et al. 2004]. In [Montanes, et al. 2002], examples of a weak interconnect open and a resistive via in a deep-submicron CMOS IC are given.

Although most parametric failures are speed related, some of them are insensitive to a single test method such as I_{DDQ} test, stuck-at test, delay test, and functional test. Simply applying a single type of tests may not be capable to identify the outliers from the normal parts, resulting in either an increased amount of test escapes or unexpected yield loss. In order to effectively screen the chips having subtle defects, multiple parameters may need to be combined for a test making the chip out of specification. Temperature, transition delay, supply voltage, and clock frequency are important parameters to be considered in multi-parameter testing [Segura, et al. 2004], [Needham, et al. 1998], [Nigh, et al. 1998].

Operating at a certain given frequency, a chip with resistive vias may fail a speed test such as F_{max} test and delay test, but pass the test at the same frequency when the operating temperature is elevated [Needham, et al. 1998]. As explained in [Segura, et al. 2004] and [Needham, et al. 1998], the root cause was the voids existing in vias. When the temperature increases, the surrounding metal expands inwardly, forcing the voids to shrink. As a consequence, the metal resistance is reduced and the delay becomes shorter. Figure 2.10 illustrates that the shapes and sizes of two voids in a via vary at different temperatures [Segura, et al. 2004]. This subtle-defect-induced parametric failure infers that a combination of parameters (e.g. frequency and temperature) is needed to sensitize the defects and a

comparison of test results at different temperatures is needed for screening the defective parts.



(a) At room temperature **(b) At a high temperature**

Figure 2.10: Via voids at different temperatures

2.10 AOFF Test Approach

Many proposed SoC test scheduling techniques assume that tests are applied to their completion [Huss, et al. 1991], [Milor, et al. 1994], [Koranne. 2002]. However, volume production tests often employ an AOFF approach in which the test process is terminated as soon as a fault is detected. The defective parts can be either discarded directly or diagnosed in order to find out the cause of the faults. Using the AOFF approach can lead to a substantial reduction in the TAT, since a test needs not to be completed if any faults are detected. The test cost can be reduced as a consequence of the decreased TAT. The AOFF test approach is especially important to the early-stage production in which defects are more likely to appear and the yield is relatively low.

When the AOFF test approach is employed, the defect probability of cores can be used for test scheduling in order to generate efficient test schedules [Jiang, et al. 2001], [Larsson, et al. 2004], [Ingelsson, et al. 2005], [He, et al. 2004], [He, et al. 2005]. The defect probabilities of IP cores can be derived from statistical analysis of production processes or generated from inductive fault analysis.

In [Jiang, et al. 2001], a defect-oriented test scheduling approach was proposed to reduce the TAT. Based on the defined cost-performance index, a heuristic algorithm was developed to obtain the best testing order. In [Larsson, et al. 2004], a more accurate cost function using defect probabilities of individual cores was proposed. Based on the proposed cost function, a heuristic algorithm was also proposed to minimize the expected test time. In this thesis, we propose a method to calculated using the probability of the test process to be terminated at any time moment when the test response/signature is available and develop a heuristic algorithm to minimize the expected test application time using the calculated probability.

Chapter 3
Temperature Aware Test Scheduling

In this chapter, we address the test time minimization problem with temperature concerns for the SoCs in which the lateral thermal influence between cores is negligible. We propose a set of test scheduling techniques to minimize the TAT such that the temperature of each CUT does not exceed an imposed temperature limit and the total amount of test-bus width required for concurrent tests does not exceed the test-bus width limit. We propose a test set partitioning and interleaving technique that avoids overheating the CUTs and keeps high efficiency in utilizing the test bus for concurrent tests. Based on the assumption of negligible lateral heat flow, we propose a CLP model to obtain optimal solution to the test time minimization problem. However, due to the high computational complexity, the CLP model is infeasible to solve the problem for large SoC designs. Therefore, we also propose a heuristic algorithm to find efficient solutions to the temperature aware test time minimization problem.

3.1 Test Set Partitioning and Interleaving

When considering SoC testing in a thermal-safe context, a long test applied to a core may lead to a high temperature even before the test is completed. A CUT may be damaged if a test is not interrupted before the temperature of the CUT exceeds a certain limit. In order to avoid overheating the CUTs, we divide an entire test set into a number of subsets, referred to as test sequences, and introduce a cooling period between the applications of two consecutive test sequences. In this thesis, we refer to cooling as passive cooling which represents a state in which a core is inactive and does not dissipate dynamic power. After a cooling period, the temperature of a CUT is supposed to decrease to a lower level, and then the succeeding test sequence may start. Figure 3.1 illustrates a scenario in which a test set is divided into four test sequences, TS_1, TS_2, TS_3, and TS_4, which are separated by three cooling periods. In this way, an entire test set is partitioned into a number of test sequences separated by cooling periods. This technique is referred to as test set partitioning (TSP) [He, et al. 2008a]. Using the TSP technique, we can effectively keep the temperature of a CUT below an imposed temperature limit.

As we assume that a test bus is employed in the assumed test access infrastructure, the limited width of the test bus becomes a constraint to the test scheduling problem. When test set partitioning is employed to avoid overheating, the efficiency of the test-bus utilization should also be considered for test scheduling. In fact, introducing cooling periods between test sequences increases the TAT for an individual core, though it helps to avoid high temperature. On the other hand, during a cooling period for a core, the test-bus width allocated to this core is not utilized since no test data is required to be transferred to/from the core. Thus, we can release the test-bus width reserved for a core during its cooling periods, and allocate the released test-bus width to other cores for their test-data transportations and test applications. In this way, the test sets for different cores are

interleaved. We refer to this technique as test set interleaving (TSI) [He, et al. 2008a]. With the TSI technique, the test bus is utilized more efficiently and the TAT can be reduced. Figure 3.2 illustrates a scenario where two partitioned test sets are interleaved so that the TAT time is reduced with no need for extra test-bus width.

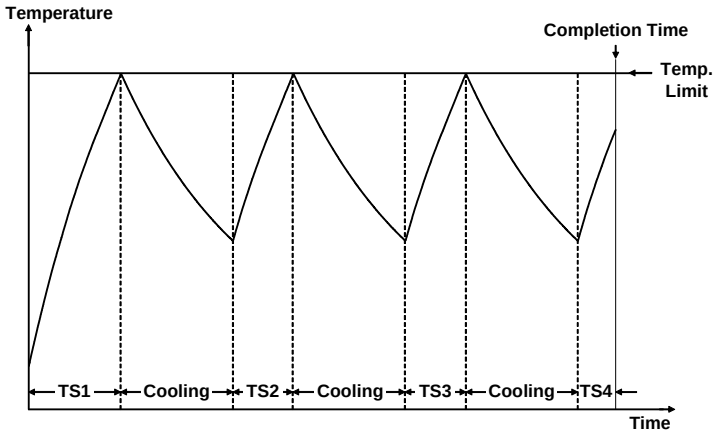


Figure 3.1: Motivational example of test set partitioning

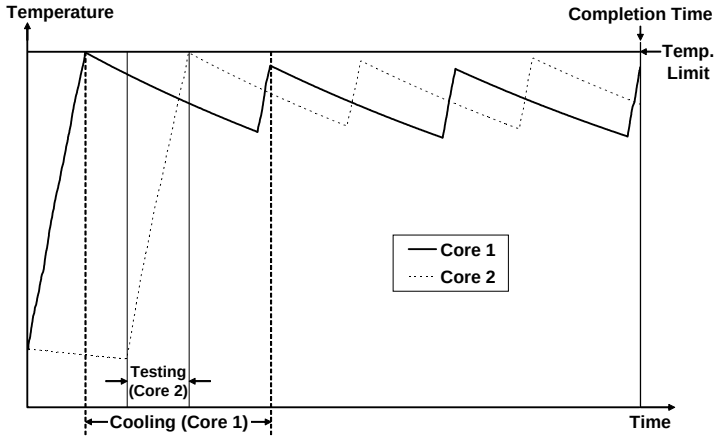


Figure 3.2: Motivational example of test set interleaving

Figure 3.3 depicts plotted temperature profiles of two CUTs in an SoC when the TSPI technique is employed for test scheduling. The temperature values are obtained through a thermal simulation and the imposed temperature limit is 90°C. This experimental result shows that using the TSPI technique can generate an efficient test schedule which satisfies both the test-bus width limit and the temperature limit.

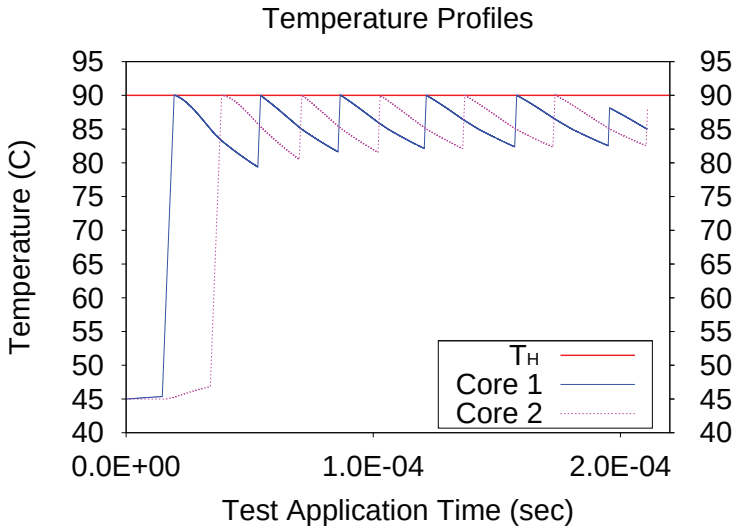


Figure 3.3. Temperature profiles of two CUTs using TSPI

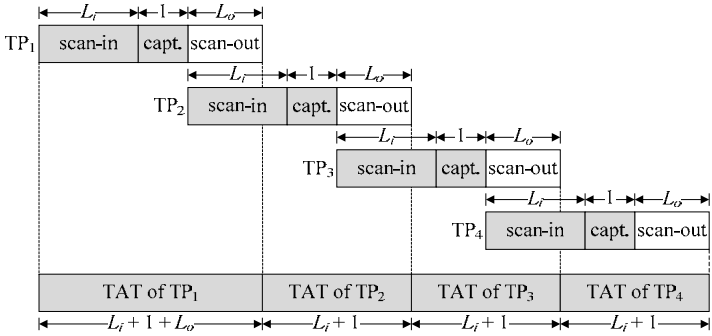
Interleaving test sequences for different cores can introduce time overhead [Goel, et al. 2003], [He, et al. 2006a]. The time overhead occurs when the test controller stops the test for one core and starts the test for another core. The cause of the time overhead is explained as follows.

In scan-based testing, the application of a test pattern includes three consecutive operations: scan-in, capture, and scan-out. During the scan-in phase, a test pattern is shifted into the scan chain. During the scan-out phase, the test response is shifted out to the response analyzer. Normally, the application of test patterns is organized as a

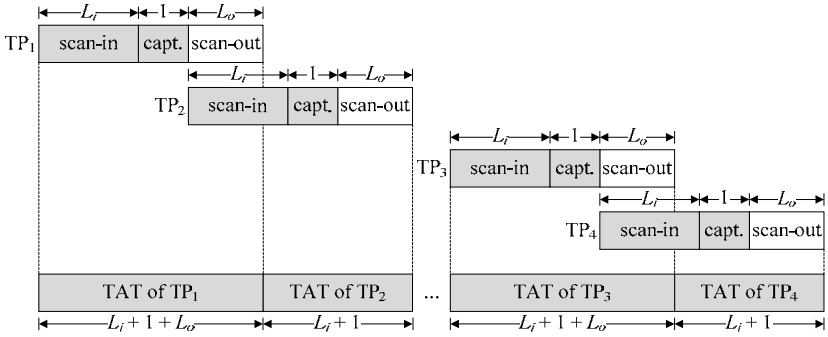
pipeline of three stages corresponding to the three operations, with the scan-out stage for one test pattern overlapping the scan-in stage for the succeeding test pattern. Figure 3.4(a) illustrates the pipeline structure for the application of four test patterns in scan-based testing. The time duration of scan-in, capture, and scan-out is L_i, 1, and L_o, respectively.

When tests are interleaved, two test sequences for the same core are separated by test sequences for other cores. This means that interrupts are introduced to the pipeline of test applications and refilling the pipeline has to be done when a test is resumed. Figure 3.4(b) illustrates that the pipeline of test application is interrupted. The test set is divided into two test sequences, each of which consists of two test patterns. The test is interrupted between the second and third test patterns. The TAT of the third test pattern is increased by L_o due to refilling the pipeline with the scan-in operation. This example shows that a time overhead is added when the test application is interrupted and resumed later.

In general, when the TSPI technique is employed for test scheduling, the time overhead has a large impact on the TAT. Partitioning a test set into more test sequences may lead to a longer TAT, since more time overheads are introduced into the test schedule. In particular, in the context of temperature aware test scheduling, a larger number of partitions leads to an even longer TAT, because more cooling periods are added into the test schedule. However, partitioning a test set into more test sequences results in a smaller average length of the partitioned test sequences. This means that the test sequences can be packed into a more compact test schedule with a shorter TAT. In principle, this trade-off between different TSPI schemes should be considered by the test scheduling algorithm. A global optimization is needed in order to explore different test schedules in which various TSPI schemes are adopted. The number and length of test sequences, as well as the number and length of cooling periods, if applicable, vary in different TSPI schemes, leading to different test schedules with different TATs.



(a) Test application without interruption



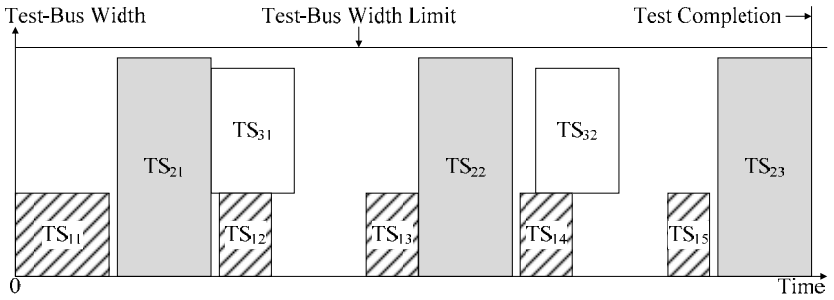
(b) Test application with interruption

Figure 3.4: Pipelined applications of test patterns in scan-based testing

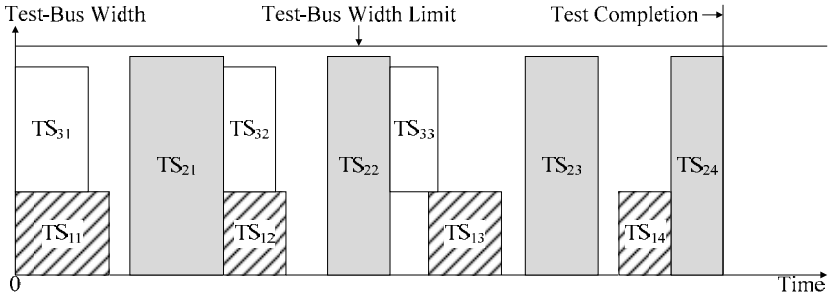
3.2 Motivational Example

In this chapter, we aim to minimize the TAT by generating an efficient test schedule such that the temperatures of the CUTs do not exceed the temperature limits of individual cores and the test-bus width constraint is satisfied. We consider each test sequence as a rectangle, with its height representing the required test-bus width and its width representing the TAT duration. Figure 3.5 shows a motivational example for the test time minimization problem. Suppose

that three test sets, TS_1, TS_2, and TS_3, are partitioned into 5, 3, and 2 test sequences, respectively. Note that, for a partitioning scheme in which the number and length of test sequences and cooling periods are determined, we use HotSpot to perform a thermal simulation to ensure that the temperature of each core does not violate the temperature limit. Figure 3.5(a) depicts a test schedule with the regularity restrictions (which is assumed in the CLP-based approach) on the length of test sequences and cooling periods. Figure 3.5(b) depicts an alternative test schedule where the regularity restrictions are removed. This example shows the possibility to find a shorter test schedule by exploring alternative solutions, which differs from each other in the number, length, and regularity of test sequences and cooling periods, as well as the way how the test sequences are interleaved.



(a) Test schedule with regular partitioning scheme



(b) Test schedule with irregular partitioning scheme

Figure 3.5: Motivational example for temperature aware test scheduling

3.3 Basic Test Architecture

We assume that the tester employed for an SoC test is either an ATE or an embedded tester in the chip. The tester consists of two major components, a test controller and a memory. The memory stores a test schedule and the generated test patterns. The test controller reads the test schedule and controls the transportation of the test data to/from the CUTs according to the test schedule. A test bus is used for the test data transportation between the tester and the CUTs. Each core is connected to the test bus through dedicated TAM wires. Through the test bus and TAM wires, test patterns are sent to the CUTs and test responses are sent back to the tester. The assumed test architectures are depicted in Figure 2.5 and Figure 2.6, corresponding to using an ATE and embedded tester as the test controller, respectively.

3.4 System Model for SoC Testing

We suppose that that a system-on-chip, denoted with S, consists of n cores, denoted with $C_1, C_2, \ldots, C_n$, respectively. A set of physical configurations F of the die and package including the floor plan of the SoC is given. In order to test core C_i $(1 \leq i \leq n)$, l_i test patterns are generated and form a test set TS_i. The test patterns/responses are transported through the test bus to/from core C_i. Transporting the test data for core C_i requires a certain amount of test-bus width W_i in bits. The test bus can concurrently transport test data for different cores under a width limit B ($B \geq W_i$, $i = 1, 2, \ldots, n$) in bits, meaning that the test bus can deliver at most B bits of test data to the CUTs in parallel.

3.5 Problem Formulation

We assume that continuously applying test patterns to a core C_i ($1 \le i \le n$) may cause the temperature of the core increase and exceed a certain limit $T_{H,i}$ and consequently results in damages to the core. We address the temperature aware test time minimization problem as how to generate a test schedule for system S such that the TAT is minimized, the test-bus width constraint is satisfied and the temperature of every CUT remains below the temperature limit $T_{H,i}$. The formal formulation of the problem is given in Figure 3.6.

Problem 3.1: Minimization of TAT for temperature aware testing

Input:
An SoC design together with the physical configuration F of the die and package as well as the floor plan of the SoC;
A set of test set for each core $\{TS_i \mid i = 1, 2, \ldots, n\}$;
A set of required test-bus width for each test $\{W_i \mid i = 1, 2, \ldots, n\}$;
Test-bus width limit B;
A set of temperature limit for each core $\{T_{H,i} \mid i = 1, 2, \ldots, n\}$.

Output:
A test schedule with the minimal test application time.

Constraints:
1. At any time moment t before all tests are completed, the total amount of allocated test-bus width $W(t)$ is less than or equal to the test-bus width limit B, i.e. $\forall t$, $W(t) \le B$, where $W(t) ::= \Sigma_j W_j(t)$;
2. At any time moment u before all tests are completed, the instantaneous temperature $T_i(u)$ of core C_i is less than the temperature limit $T_{H,i}$, i.e. $\forall u$, $\forall i$, $T_i(u) < T_{H,i}$.

Figure 3.6: Problem formulation of temperature aware test scheduling

The formulated problem is highly complex. When considering the first constraint, the test time minimization problem can be mapped to a 2D rectangle packing (RP) problem in which a test sequence is represented by a rectangle. The height of a rectangle represents the test-bus width required by the test sequence and the width of a rectangle represents the TAT of the test sequence (see Figure 3.5). Since a 2D RP problem is NP-complete [Baker, et al. 1980], no polynomial-time algorithm exists to obtain the optimal solution. In this chapter, we provide two approaches to solve the temperature aware test time minimization problem.

The first approach restricts the exploration space by introducing the following two constraints: (1) all test sequences belonging to the same test set, except the first and last one, must have an identical length; (2) all cooling periods between the test sequences belonging to the same test set must have an identical length. By adding these two restrictions, the optimal solution to this restricted test time minimization problem can be obtained by using a CLP model.

The CLP-based approach is infeasible for large SoC designs due to the high computational complexity of the algorithm. Alternatively, we propose a fast heuristic approach to solve the problem with no restriction on the regularity of test sequences and cooling periods, i.e. the test sequences and cooling periods can have flexible length. This means that the test sequences can be repartitioned and the cooling periods can be changed in test scheduling. In order to ensure the thermal safety, we introduce the following two restrictions to the heuristic approach: (1) the length of a repartitioned test sequence must not be to be larger than the regular length of the initially partitioned test sequences; (2) the length of a cooling period must not be smaller than the regular length of the initially fixed cooling periods.

3.6 Overall Solution Strategy

The overall solution strategy to solve the formulated problem is illustrated in Figure 3.7. In the first step, we generate an initial partitioning scheme for every test set through a thermal simulation with the imposed temperature limits. In the second step, we use the proposed test scheduling algorithm to explore alternative test schedules with respect to different partitioning and interleaving schemes for the test sets. The test scheduling algorithm squeeze test sequences into the 2D plane, constrained by the test-bus width limit, such that the TAT of the test schedule is minimized.

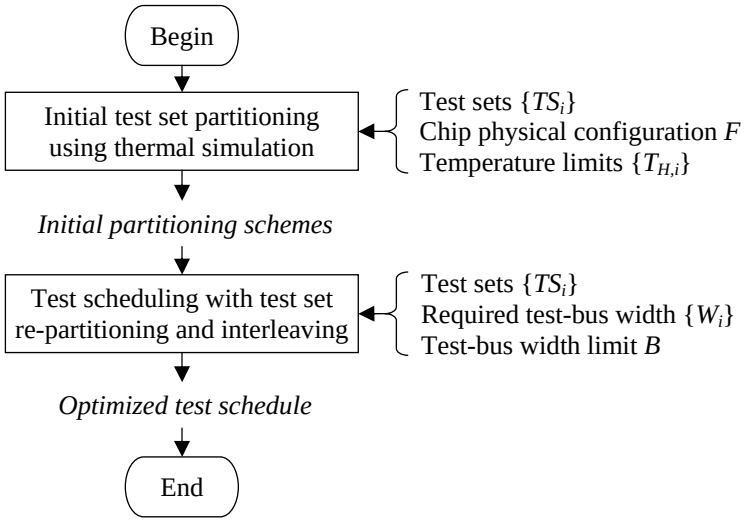


Figure 3.7: Overall solution strategy

In order to generate thermal-safe partitioning schemes, we have used a temperature simulator, HotSpot, to obtain instantaneous temperatures of individual cores. In this chapter, we assume that the lateral thermal influences between cores are negligible for a certain type of SoCs, as the heat transfer in the vertical direction is much larger than that in the lateral direction.

When generating the initial thermal-safe partitioning scheme, we assume that a test set TS_i is started when the core is at the ambient temperature T_A. Thereafter, we start the temperature simulation, and record the time moment t_{h1} when the temperature of core C_i reaches the given temperature limit $T_{H,i}$. Knowing the latest test pattern that has been applied by the time moment t_{h1}, we can easily obtain the length of the first thermal-safe test sequence TS_{i1} that should be partitioned from the test set TS_i. Then the temperature simulation continues while the test process on core C_i has to be stopped until the temperature goes down to a certain degree. Note that a relatively long time is needed in order to cool down a core to a temperature close to T_A, as the temperature decreases slowly at a lower temperature level (see the dashed curve in Figure 3.8). Thus, we let the temperature of core C_i decreases until the slope of the temperature curve reaches a given value k, at time moment t_{c1}. The value of k can be experimentally set by the designer. At this moment, we have obtained the duration of the first cooling period $d_{i1} = t_{c1} - t_{h1}$. Resuming the test process from time moment t_{c1}, we repeat this heating-and-cooling cycle throughout the temperature simulation until all test patterns belonging to TS_i are applied. Thus, we have generated the initial thermal-safe partitioning scheme, where test set TS_i is partitioned into m test sequences $\{TS_{ij} \mid j = 1, 2, \ldots, m\}$ and between every two consecutive test sequences, the duration of the cooling period is $\{d_{ij} \mid j = 1, 2, \ldots, m\text{-}1\}$. Figure 3.8 depicts a motivational example of partitioning a test set into four thermal-safe test sequences separated by three cooling periods.

Once the initial thermal-safe partitioning scheme is obtained, we focuses on the problem of generating the shortest test schedule such that test-bus width constraint is satisfied. As mentioned earlier, the problem can be mapped to a 2D RP problem. However, our test scheduling problem is not a classical RP problem, due to the fact that the number and length of test sequences and cooling periods are not fixed. This makes our problem even more difficult to solve.

Based on the overall solution strategy, we propose two approaches to solve the test time minimization problem, a CLP-based approach and a heuristic approach, which are presented in the following two sections, respectively.

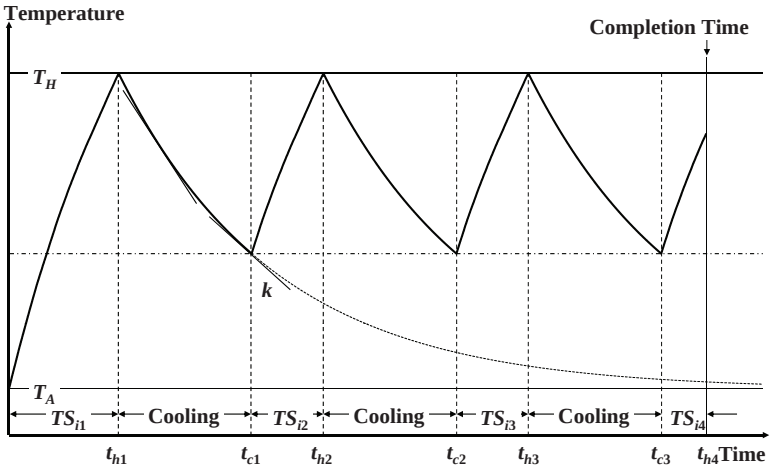


Figure 3.8: Motivational example of the initial partitioning scheme

3.7 CLP-based Approach with Regular TSP

As demonstrated previously, in order to restrict the exploration space, we assume that the test sequences belonging to the same test set have identical length except the first and the last one. The first test sequence is usually longer than the others in the same test set. This is because the CUT is initially at the ambient temperature, and the first test sequence is partitioned such that it is continuously applied until the CUT reaches the temperature limit. Similar to the test partitions, the cooling periods between two test sequences from the same test set also have identical length.

3.7.1 Constraint Logic Programming

Constraint logic programming is a programming framework which merges two declarative programming paradigms, namely constraint solving and logic programming [Jaffar, et al. 1987]. CLP defines the relationships between entities as constraints, and incorporates constraint solving methods into a logic-based programming language. Some key features of CLP include: (1) constraints are employed to describe the queries and answers which are the inputs and outputs of a program; (2) new variables and constraints are dynamically generated during execution of the program; (3) in each state of execution, all constraints are globally tested for satisfiability, and the results of the test are used to control the execution.

As a declarative programming language, CLP is flexible and expressive. It allows programmers to focus on the formulation of problems instead of being stuck in the implementation details. Therefore, it has been widely used in many optimization techniques for a variety of applications. Some CLP tools also provide solvers to find the optimal solution using branch-and-bound or exhaustive search. We use CHIP [Van Hentenryck. 1991] in our CLP-based approach to solve the temperature aware test time minimization problem.

3.7.2 CLP Model

A partitioning scheme has three parameters, the number of partitions, the time duration of the first test sequence, and the time duration of a cooling period between two consecutive test sequences, which are denoted with m_i, l_{i1}, and d_i, respectively. The individual test for each core starts at time moment t_i, which is equal to the start time t_{i1} of the first test sequence in the same test set.

$$t_i = t_{i1} \quad (1 \le i \le n) \tag{3.1}$$

The number of partitions and the start time of every individual test are decided during the optimization. The start time t_{ij} and finish time

e_{ij} of test sequence TS_{ij} can be calculated as follows. Note that o_i is the time overhead.

$$t_{ij} = t_{i,j-1} + l_{i,j-1} + d_i + o_i \quad (2 \le j \le m_i, 1 \le i \le n) \tag{3.2}$$

$$e_{ij} = t_{ij} + l_{ij} \quad (1 \le j \le m_i, 1 \le i \le n) \tag{3.3}$$

The last test sequence in each test set is special since its finish time is equal to the finish time of the individual test for the core. Thus, the finish time e_i of test set TS_i is

$$e_i = e_{i,m_i} \tag{3.4}$$

and the TAT for testing all cores is the maximum finish time of all individual tests, given as follows.

$$TAT = \max_{1 \le i \le n} \{ e_i \} \tag{3.5}$$

TAT is the cost function of our optimization problem, and our objective is to find the optimal solution $\{(m_i^*, t_i^*) \mid i = 1, 2, \ldots, n\}$ such that *TAT* is minimized, subject to the following constraint: at any time moment x before the completion of all individual tests, the total amount of test-bus width used for the concurrent test sequences is less than or equal to the test-bus width limit, i.e.

$$\forall x \le TAT, \sum_{k=1}^{p_x} W_k \le B \tag{3.6}$$

where p_x is the number of concurrent test sequences at the time moment x;

As discussed in previous sections, we assume that when a test starts, the CUT is at the ambient temperature T_A. The test set has to be partitioned into a number of test sequences if the CUT reaches its temperature limit before the entire SoC test is completed. When partitioning a test set into test sequences, the length of each test sequence and the number of test sequences depend on the length of the cooling period between two consecutive test sequences. A longer cooling period leads to a lower temperature at which the succeeding

test sequence will be started. Thus, with the partitioning schemes that have longer cooling periods, a test set can be partitioned into fewer number of test sequences but each test sequence is longer. It is important to find a possible interval of the number of partitions for each test set, since our optimization algorithm explores alternative partitioning schemes in which the number of partitions varies between the minimum and the maximum values in this interval. We denote the interval of the number of partitions for a test set TS_i with I_i ($1 \leq i \leq n$), and $I_i = [I_{i,min}, I_{i,max}]$.

As described in Section 3.6, we perform a thermal simulation to obtain the initial thermal-safe partitioning scheme for each test set. Based on the initial partitioning schemes, we can determine the exploration interval I_i ($1 \leq i \leq n$) for each test set. We define the number of partitions in the initial partitioning scheme as the minimum value of I_i, denoted with $I_{i,min}$. In order to find the maximum value of I_i, denoted with $I_{i,max}$, we have done experiments for different designs and we have found out that the actual numbers of partitions in the optimal solutions are close to the minimum values $I_{i,min}$. Thus, we define the maximum value of the exploration interval as $I_{i,max} = K + I_{i,min}$, where K is a constant value which can be fixed by the designer. The exploration interval $I_i = [I_{i,min}, I_{i,max}]$ ($i = 1, 2, \ldots, n$) for each test set TS_i is taken as an input to the optimization algorithm.

For each test set TS_i ($1 \leq i \leq n$), two variables have to be decided by the CLP solver. One is the number of partitions, denoted with m_i, and the other is the start time of the individual test, denoted with t_i. The finish time of an individual test is equal to its start time plus the time durations of all its test sequences and all the cooling periods added between two consecutive test sequences, given as follows.

$$e_i = t_i + \sum_{j=1}^{m_i} l_i + d_i \times (m_i - 1) \tag{3.7}$$

During optimization, the decision variables are instantiated and test schedules that satisfy the constraints are explored. The CLP solver

finds the optimal solution which has the minimal TAT of the SoC test. The minimal TAT in the CLP model is formulated as:

$$TAT_{\min} = \min_{1 \le i \le n} \{ \max_{0 \le t_i \le L,\ I_{i,\min} \le m_i \le I_{i,\max}} \{ e_i \} \} \tag{3.8}$$

where L is a constant configured in the CLP model. Note that the search of the optimal solution using the CLP model is related to the second step (test scheduling with TSPI) in the overall solution strategy depicted in Figure 3.7.

3.7.3 Experimental Results

We use the ISCAS'89 benchmark circuits as the cores of the SoC designs for our experiments. Table 3.1 shows the experimental results for five different SoC designs with the number of cores listed in column 1. For each SoC design, test patterns are generated for every core, and the switching activities are calculated for each test pattern. The amount of power consumption of each test pattern is calculated using a cycle-accurate power estimation method proposed in [Samii, et al. 2006], which takes the amount of switching activity as an input and calculates the power consumption in Watt. We use HotSpot for the thermal simulation through which we obtain the initial partitioning schemes. The total number of partitioning schemes for each SoC design is listed in column 2. The imposed temperature limit is 90°C.

We used the CLP model to obtain the optimal test schedule by exploring alternative numbers of partitions and start times for each test. Column 3 in Table 3.1 shows the problem size for each SoC design, defined as the product of the number of partitioning schemes and the number of cores in the SoC. The TAT of the optimal test schedule and the optimization time are listed in the columns 4 and 5.

When optimal test schedule is obtained, we perform a thermal simulation according to the generated test schedule in order to check if the temperature of any core exceeds the temperature limit. The thermal simulation results confirm that the temperature of every core is below the temperature limit.

Table 3.1: TATs and execution times using the CLP model

# of Cores	# of Partitioning Schemes	Problem Size	TAT (# of Clock Cycles)	CPU Time (s)
4	7	28	2775	2.141
12	8	96	8306	35.359
24	20	400	9789	47.500
36	20	720	10017	120.219
48	20	960	10941	881.766

We have also performed experiments to evaluate how the obtained optimization result is affected by choosing different number of partitioning schemes. Table 3.2 listed four different numbers of partitioning schemes which are explored by the CLP solver. The experiments are performed for an SoC design with 6 cores. The optimal solution is the same in the three cases of 7, 10, and 15 partitioning schemes, as shown in the last three rows, respectively. When the number of partitioning schemes is 5 (see the 1st row), the TAT of the obtained test schedule is larger than the others, which infers that the best solution does not correspond to any partitioning scheme among the 5 alternative ones. If we introduce 2 more alternative partitioning schemes, a better solution is found (see the 2nd row). However, adding more alternatives partitioning schemes, up to 15, do not lead to a better solution (see the 3rd and 4th rows).

The reason for the shorter TAT with an increased number of alternative partitioning schemes is explained as follows. When a test set is partitioned into more test sequences, more time overheads and cooling periods are added into the test schedule. However, the individual test sequences and the cooling periods are shorter if the test set is partitioned into more test sequences, and the scheduling algorithm can generate more compact test schedules. This trade-off between different partitioning schemes has been discussed in Section 3.1. This experimental result infers that the optimal solution

may not correspond to any partitioning scheme in the minimum set of partitioning schemes.

Table 3.2: TSTs w.r.t. different number of partitioning schemes

# of Cores	# of Partitioning Schemes	Problem Size	TAT (# of Clock Cycles)	CPU Time (s)
6	5	30	9574	10.156
	7	42	9570	26.031
	10	60	9570	31.875
	15	90	9570	39.797

3.8 Heuristic Approach with Irregular TSP

As demonstrated previously, although the CLP-based approach can provide the optimal solution for the restricted problem, it is not feasible to obtain the solutions for large SoC designs due to its high computational cost. Alternatively, we propose a heuristic approach to solve the original test time minimization problem with no restrictions on the regularity of test sequences and cooling periods.

3.8.1 Motivational Example

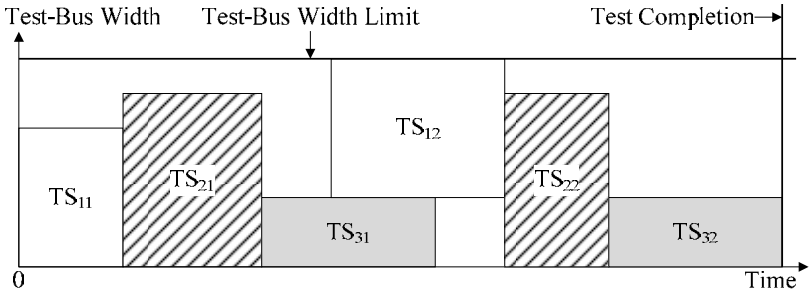
The proposed heuristic algorithm for test scheduling also uses the TSPI technique. Since the order in which the test sets are considered for test scheduling has a large impact on the final test schedule, we construct an iterative algorithm which obtains a good scheduling consideration order (SCO) for all partitioned test sets. Thereafter, the test sequences are scheduled according to the obtained SCO.

Figure 3.9 shows a motivational example to illustrate the impact of the SCO on the test schedule. In this example, each of the 3 test sets, denoted with TS_1, TS_2, and TS_3, is partitioned into 2 test sequences. Figure 3.9(a) and Figure 3.9(b) depict the test schedules when the test sets are considered for scheduling in the order of $\{TS_1, TS_2, TS_3\}$ and $\{TS_3, TS_2, TS_1\}$, respectively. It can be seen that using the second SCO results in a shorter test schedule depicted in Figure 3.9(b). Note that, in this example, the test sets are scheduled to the earliest available time moments (EATM).

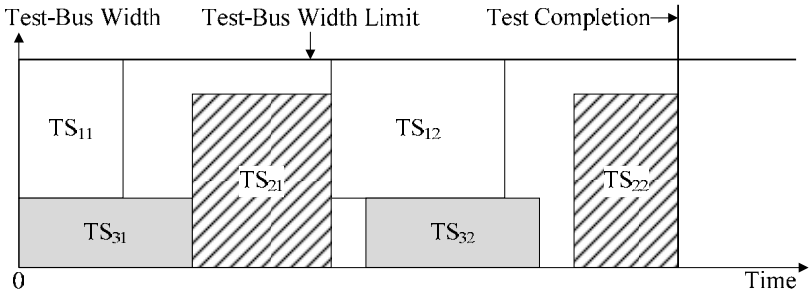
In fact, the SCO reflects the precedence of the partitioned test sets to be considered for scheduling. However, when algorithm considers a test set for scheduling, it does not all the test sequences from the same test set at the same time. Instead, it always take the first unscheduled test sequence of the currently considered test set for scheduling, and thereafter take the first unscheduled test sequence of the next test set into account. Thus, in this example, the overall scheduling consideration order (OSCO) for all test sequences of all test sets is $\{TS_{11}, TS_{21}, TS_{31}, TS_{12}, TS_{22}, TS_{32}\}$ and $\{TS_{31}, TS_{21}, TS_{11}, TS_{32}, TS_{22}, TS_{12}\}$, for the case in Figure 3.9(a) and Figure 3.9(b), respectively. The main concern of not scheduling all test sequences of one test set at the same time is to avoid generating inefficient test schedules due to unnecessarily long cooling periods, inappropriate partition length, and inefficient test set interleaving.

3.8.2 Heuristic Algorithm for Test Scheduling

The basic idea of the proposed heuristic algorithm for test scheduling is to iteratively construct a queue that finally consists of all partitioned test sets in a particular order. The heuristic algorithm is denoted with Algorithm 3.1 and its pseudo-code is depicted in Figure 3.10. Note that Algorithm 3.1 invokes a scheduler Algorithm 3.2.



(a) Test schedule w.r.t. SCO {TS_1, TS_2, TS_3}



(b) Test schedule w.r.t. SCO {TS_3, TS_2, TS_1}

Figure 3.9: Motivational example of test schedules affected by the SCO

Algorithm 3.1: Heuristic algorithm for temperature aware test scheduling

```
01: Set of test sets :: U := {TS_i | i = 1, 2, … , n};
02: Queue of test sets :: Q := ∅;
03: Queue of test sets sorted in the best SCO :: Q_best := ∅;
04: for (∀TS ∈ U) loop    /* outer loop */
05:     η_max := 0;
06:     Q := Q_best;
07:     for (∀POS in Q) loop    /* inner loop */
08:         Insert(TS, Q, POS);
09:         Sched_cur = Schedule(Q);
10:         η = CalcEfficiency(Sched_cur);
11:         if (η > η_max) then
12:             η_max := η;
13:             TS_best := TS;
14:             Q_best := Q;
15:         end if
16:         Remove(TS, Q);
17:     end for
18:     Remove(TS_best, U);
19: end for
20: Schedule(Q_best);
```

Figure 3.10: Pseudo-code of the heuristic algorithm for test scheduling

Given the set of all test sets $U = \{TS_i \mid i = 1, 2, \ldots, n\}$ (line 1), the heuristic algorithm iteratively selects test sets and inserts them into a queue Q (lines 2 through 19). The positions of the test sets in Q represent the order in which the test sets are considered for test scheduling (SCO). The precedence of the positions in Q is defined as follows. A test set positioned closer to the queue head will be considered earlier for test scheduling than those test sets positioned further to the queue head.

The heuristic algorithm starts with an empty queue $Q = \emptyset$ (line 2). In each iteration step (lines 5 through 18), the objective is to select one test set TS_k from U, and insert it into Q at a certain position POS, such that the $(|Q| + 1)$ test sets are put in a good order while the precedence between test sets excluding the newly inserted one remains unchanged. The outer loop terminates when all test sets in U have been moved into Q, and thereafter, the heuristic algorithm invokes the scheduler to schedule the partitioned test sets according to the SCO presented in Q_{best} (line 20).

For each iteration step, there are $|U|$ alternative test sets for selection, where $|U|$ is the current number of test sets remaining in U. For each selected test set, there are $(|Q| + 1)$ alternative positions which the selected test set can be inserted to, where $|Q|$ is the current number of test sets that have already been inserted into Q throughout previous iteration steps. Thus, in one iteration step, there are $|U| \times (|Q| + 1)$ alternative solutions, in which a selected test set is associated with an insertion position in Q.

The example depicted in Figure 3.11 illustrates a scenario where 3 test sets (TS_3, TS_8, and TS_6) have been inserted in Q and 5 other test sets (TS_1, TS_2, TS_4, TS_5, and TS_7) remain in U. For each test set in U, there are 4 insertion positions, which are pointed by the arrows. In this example, there are 20 alternative solutions. Note that each test set in the example has already been partitioned into a number of test sequences, and Algorithm 3.2 takes each test sequence for scheduling.

We evaluate the obtained SCO by the efficiency of the generated partial test schedule, the higher efficiency, the better the SCO. The

partial test schedule is generated (line 9) by Algorithm 3.2. The efficiency of a test schedule (EOTS), denoted with η, is defined as follows. Suppose x is the size of the area covered by all scheduled test sequences, and y is the total area size constrained by the test-bus width limit and the completion time moment of the test schedule. The efficiency of the test schedule is (x / y). A larger value of η indicates a better test schedule. Figure 3.12 depicts an example which illustrates how the EOTS is calculated. In the example, a test schedule is represented as the area covered by slashed lines. The size of the area covered by the actual test schedule is x, and the size of the area covered by the larger rectangle with thick border lines is y. Based on the definition of EOTS, we explore alternative solutions and select the best solution according to the efficiency of the generated partial test schedules.

By calculating and comparing the efficiencies of the alternative partial test schedules (line 10), the best solution that obtains the maximum EOTS is chosen. The maximum TSE, the chosen test set, and the entire queue, are recorded in η_{max}, TS_{best}, Q_{best}, respectively (lines 12 through 14). The iteration terminates when all test sets in U have been moved into Q. The obtained Q_{best} consists of all test sets in the best SCO, in which the test sets will be considered for scheduling (line 20).

Algorithm 3.2 schedules a queue of test sets and its pseudo-code is depicted in Figure 3.14. Given a queue Q of test sets, the scheduler takes the first unscheduled test sequence from every test set for scheduling, in a round-robin fashion. More concretely, the strategy of the scheduling algorithm is explained as follows. According to the SCO given in Q, the scheduler considers one test set for scheduling at a time. When considering each test set, the scheduler only schedules the first unscheduled test sequence, and thereafter turns to consider the next test set in Q. When one round is finished for all the test sets in Q, the scheduler takes the next round to consider scheduling the test sequences of all the test sets in the same SCO. This procedure repeats until all test sequences are scheduled.

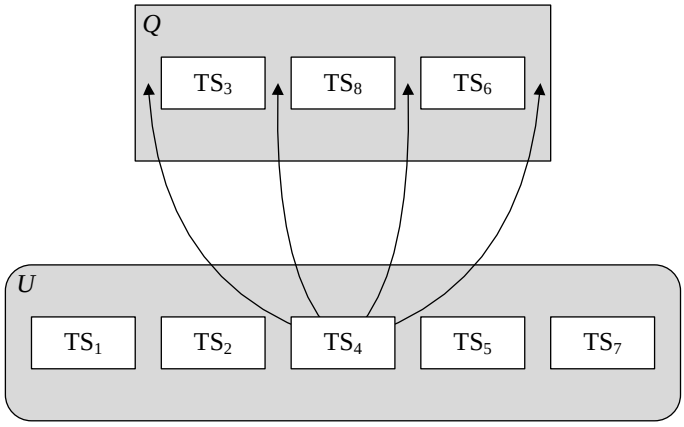


Figure 3.11: Example of alternative solutions

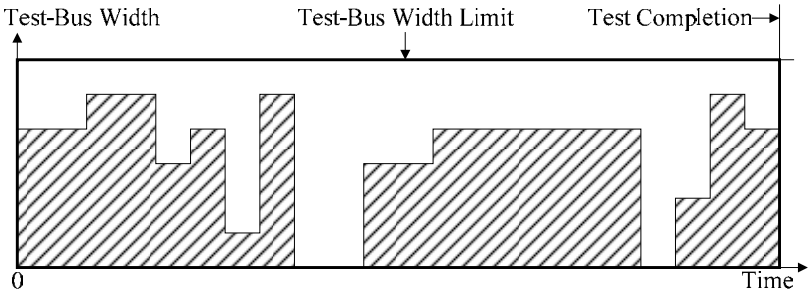


Figure 3.12: Efficiency of a test schedule

Figure 3.13 depicts an example which illustrates how the scheduler works. In the example, three test sets, TS_2, TS_1, and TS_3, are sorted in the SCO of {TS_2, TS_1, TS_3} in Q. The test set TS_2 has been initially partitioned into three test sequences, TS_{21}, TS_{22}, and TS_{23}. The other two test sets, TS_1 and TS_3, are both partitioned into four test sequences. The OSCO of all test sequences is {TS_{21}, TS_{11}, TS_{31}, TS_{22}, TS_{12}, TS_{32}, TS_{23}, TS_{13}, TS_{33}, TS_{14}, TS_{34}}, as indicated by the dashed arrows.

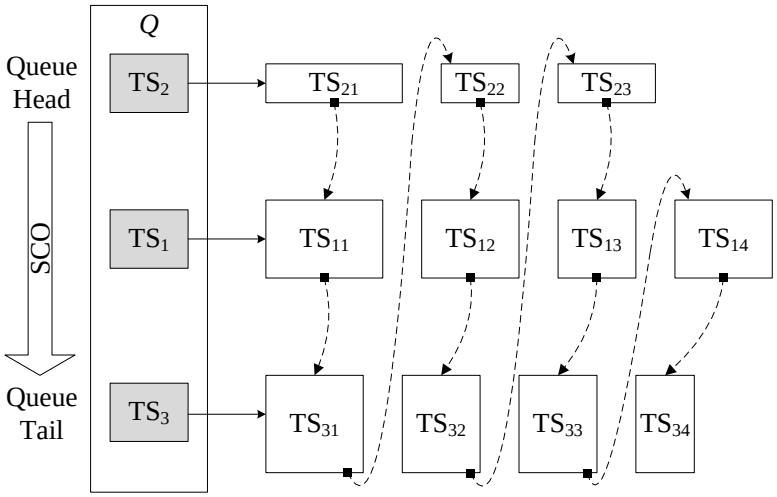


Figure 3.13: Illustration of the scheduling algorithm

In the pseudo-code of Algorithm 3.2 depicted in Figure 3.14, the scheduling algorithm is constructed with two nested loops. The outer loop (lines 21 through 34) selects the first unscheduled test sequence for the current test set, while the inner loop (lines 22 through 33) selects a test set for scheduling according its position in *Q*. The algorithm terminates when all the test sequences have been scheduled. Note that the function *GetNumOfPar*(*TS*) in line 21 takes a test set *TS* as an input, and returns the number of test sequences that the test set has been partitioned into.

When schedules a test sequence $TS_{q,j}$ (the *j*-th test sequence of the *q*-th test set in *Q*, see line 23 through 27), the scheduler tries to schedule it to the earliest available time moment $t_{q,j}$ (line 27). The earliest time moment that a test sequence can be scheduled to is the time moment when the required minimum cooling period succeeding the precedent test sequence has finished. The minimum cooling period $d_{q,j}$ is given by the initial partitioning scheme for the test set TS_q (line 27).

Algorithm 3.2: *Schedule*(Queue of test sets :: *Q*)

21: **for** (j = 1 to $max\{GetNumOfPar(\forall TS \in Q)\}$) **loop** /* outer loop */
22: **for** (q = 1 to $|Q|$) **loop** /* inner loop */
23: Choose the q-th test set TS_q in Q for scheduling;
24: **if** ($TS_q = \varnothing$) **then**
25: Skip TS_q and continue with the next test set;
26: **else**
27: Schedule the first unscheduled test sequence $TS_{q,j}$
to the earliest available time moment
$t_{q,j} := GetFinishingTime(TS_{q,j-1}) + d_{q,j}$
where $d_q := InitialCoolingSpan(TS_q)$;
28: **if** (Failed to schedule $TS_{q,j}$ to $t_{q,j}$) **then**
29: Estimate the completion time t_e of the entire test set TS_q by either postponing $TS_{q,j}$ or repartitioning all the unscheduled test sequences in TS_q;
30: Choose the solution that has a smaller t_e and schedule the first unscheduled test sequence;
31: **end if**
32: **end if-then-else**
33: **end for**
34: **end for**

Figure 3.14: Pseudo-code of the scheduling algorithm

The scheduler tries to schedule every test sequence to the earliest available time moment, but there may not be sufficient space in the 2D plan to squeeze in the test sequence at the desired EATM. Figure 3.15 depicts such an example. It is not possible to squeeze the test sequence $TS_{q,j}$ to the EATM $t_{q,j}$, due to the space between the test-bus width limit B and the area (in slashed lines) occupied by the scheduled test sequences. Actually, in this example, the earliest time moment that $TS_{q,j}$ can be scheduled to is t_p.

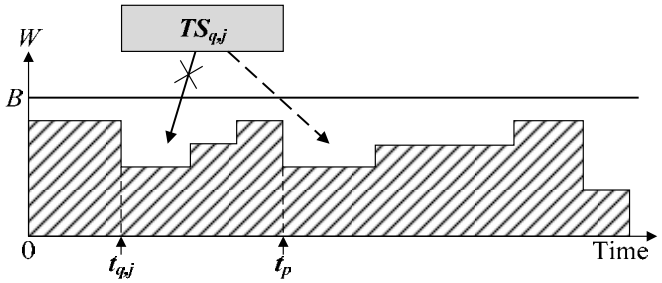


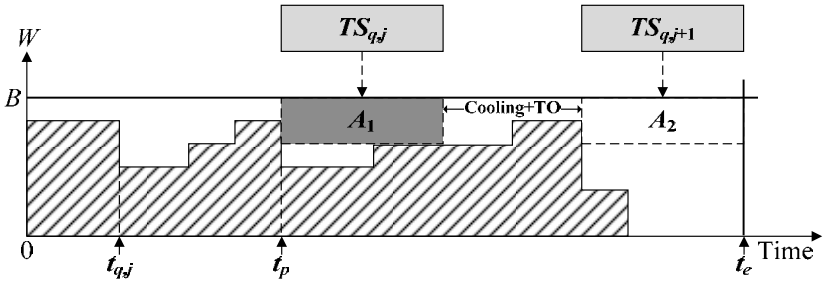
Figure 3.15: A scheduling constraint example

When such scheduling constraints are encountered, two alternative solutions are proposed. One solution is to postpone the entire test sequence to a time moment that it can be successfully scheduled to. The other solution is to split the test sequence into smaller pieces such that the first piece can be squeezed into the available area. Figure 3.16 illustrates both solutions for the same example given in Figure 3.15, where the entire test sequence $TS_{q,j}$ cannot be scheduled to the time moment $t_{q,j}$. In Figure 3.16(a), the solution is to postpone the entire test sequence $TS_{q,j}$ to time moment t_p, which means squeezing $TS_{q,j}$ into the dark grey rectangular area A_1 that the dashed arrow points to. Figure 3.16(b) illustrates the other solution, where $TS_{q,j}$ is split into two pieces which can fit into the dark grey rectangular areas S_1 and S_2, respectively.

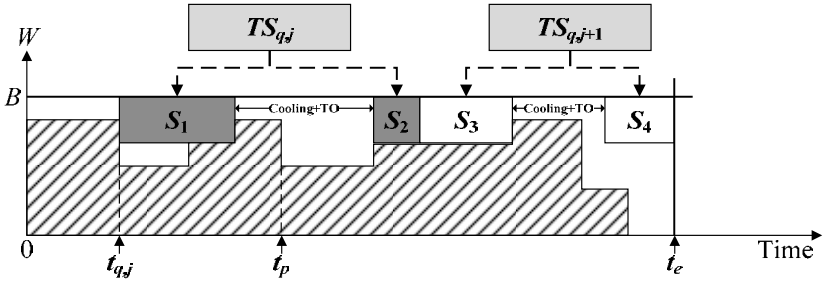
Both solutions can result in long test schedules. The first solution, which postpones the entire test sequence, also delays the succeeding test sequences. This can results in delaying the completion of the entire test set. As illustrated in Figure 3.16(a), the succeeding test sequence $TS_{q,j+1}$ is delayed and finishes at time moment t_e. The second solution, which splits the test sequence into smaller pieces, generates more partitions and introduces more time overheads. In order to avoid these drawbacks, we repartition all the unscheduled test sequences from the same test set, such that the total number of test sequences will not increase dramatically due to the splitting. This is illustrated in Figure 3.16(b). After splitting $TS_{q,j}$ into two pieces which fits in S_1

and S_2 respectively, we also repartition the succeeding test sequence $TS_{q,j+1}$ such that its two pieces fit into S_3 and S_4. Note that due to the splitting of $TS_{q,j}$ and TS_{qj+1}, time overheads (denoted with *TO*) are added between the repartitioned test sequences.

As demonstrated above, both solutions can be adopted when scheduling a test sequence. In order to decide which solution should be employed, we estimate the completion time t_e for the entire test set (line 29), by assuming that all the unscheduled test sequences of this test set can be scheduled to their earliest available time moments. The solution that results in an earlier estimated completion time is chosen (line 30). In the example given in Figure 3.16, the second solution should be chosen, since it leads to a smaller t_e. The scheduling algorithm terminates when all test sequences of all test sets in *Q* have been scheduled (line 34).



(a) Postponing the entire test sequence



(b) Splitting the test sequence into smaller pieces

Figure 3.16: Two alternative solutions to deal with scheduling constraint

3.8.3 Experimental Results

ISCAS'89 benchmark circuits are used as the cores in the SoC designs for our experiments. The number of cores in the SoC designs varies from 12 to 78.

The first group of experiments shows the impact of relaxing the regularity of test sequences and cooling periods on the TAT of the generated test schedules. The results of the first group of experiments are shown in Table 3.3.

Table 3.3: FLSA vs. ESLA and 2PSA

# of Cores	ELSA		2PSA		FLSA		TAT Reduction	
	TAT	CPU Time(s)	TAT	CPU Time(s)	TAT	CPU Time(s)	from ELSA	from 2PSA
12	1502	0.01	1390	0.01	1048	2.74	30.2%	24.6%
18	2761	0.02	2029	0.01	1535	5.41	44.4%	24.3%
24	3975	0.05	3571	0.02	2318	21.88	41.7%	35.1%
30	2831	0.01	2510	0.02	1915	32.41	32.4%	23.7%
36	3587	0.08	3368	0.08	2539	67.52	29.2%	24.6%
42	4845	0.03	4012	0.03	3334	101.39	31.2%	16.9%
48	4878	0.06	4513	0.06	3509	151.33	28.1%	22.2%
54	5696	0.06	5024	0.08	4290	244.36	24.7%	14.6%
60	6303	0.19	5504	0.13	4692	371.73	25.6%	14.8%
66	6868	0.34	5889	0.41	5069	511.88	26.2%	13.9%
72	7903	0.17	6923	0.22	5822	720.53	26.3%	15.9%
78	7900	0.72	6803	0.77	5769	987.75	27.0%	15.2%
AVG	4920.75	0.15	4294.67	0.15	3486.67	268.24	30.6%	20.5%

We compare our heuristic algorithm with two other scheduling algorithms. The first algorithm employs a fixed SCO in which all the test sets are sorted decreasingly according to the length of test sets in

their initial partitioning schemes. Then it schedules the entire test set to the earliest available time moment, according to the obtained SCO. When scheduling the test sequences of a test set, it keeps the regularity of the partitions and cooling periods given by the initial partitioning scheme. For convenience, we call this algorithm "equal-length scheduling algorithm" (ELSA).

The second algorithm also employs the fixed SCO according to the lengths of partitioned test sets (longest first). However, different from the ELSA, it schedules a test set in two phases. In the first phase, it schedules only the first partition of all test sets, according to the obtained SCO. This is due to the fact that the first test sequence is usually much longer than the other ones from the same test set in the initial partitioning scheme (see Figure 3.8). In the second phase, it schedules all the remaining test sequences of every test set, according to the same SCO. Similar to the ELSA, it schedules test sequences to the earliest available time moment. When scheduling the test sequences in the second phase, it keeps the regularity of all test partitions and cooling periods given in the initial partitioning scheme. Moreover, the first cooling period succeeding the first test sequence may not be shorter than that in the initial partitioning scheme. This means that by separating the scheduling of a test set into two phases, the restriction on partitioning regularity is slightly relaxed, thus this algorithm has a higher flexibility on test set partitioning schemes than the ELSA. For convenience, we call the second scheduling algorithm "two-phase scheduling algorithm" (2PSA).

Compared to the ELSA and 2PSA, the proposed heuristic algorithm has the highest flexibility on test set partitioning schemes, since it allows repartitioning of test sets and allows arbitrarily increasing lengths of cooling periods in test scheduling. For convenience, we call the proposed heuristic algorithm "flexible-length scheduling algorithm" (FLSA).

In Table 3.3, column 1 lists the number of cores used in the SoC designs. Columns 2, 4, and 6 list the TATs of the test schedules generated for the corresponding SoC designs, by using the ELSA,

2PSA, and FLSA, respectively. Columns 3, 5, and 7 list the CPU times for executing the corresponding algorithms. Columns 8 and 9 show the percentage of the TAT reduction by using the FLSA versus the ELSA and 2PSA, respectively. It can be seen that by eliminating restrictions on the regularity of partitioning schemes, the TAT is in average 30.6% and 20.5% shorter than that from the ELSA and 2PSA, respectively.

The second group of experiments evaluates the efficiency of test schedules generated by the proposed heuristic algorithm FLSA. In this group of experiments, we compare the FLSA with two other heuristic algorithms, a straight-forward algorithm (SFA) and a simulated-annealing-based algorithm (SABA). For this group of experiments, we assume the same flexibility for all the three algorithms, i.e. all of them employ flexible partitioning of test sets and arbitrary increasing length of cooling periods.

All the three algorithms employ Algorithm 3.2 as the scheduler. The only difference between them is how they generate the SCO for the test sets. The SFA sorts all test sets decreasingly by the lengths of the entire test sets with the initial partitioning schemes. According to the obtained SCO, the scheduler chooses each test set and schedules the first unscheduled test sequences to the earliest available time moment, until all test sequences of every test set are scheduled.

The SABA employs Algorithm 3.2 to schedule the test sequences, while the SCO of the test sets is generated based on a simulated-annealing strategy. When a randomly generated SCO is obtained, the scheduler is invoked to schedule the test sequences according to the current SCO. During iterations, the best SCO that leads to the shortest test schedule is recorded and the algorithm returns this recorded solution when the stopping criterion is met.

The results of the second group of experiments are shown in Table 3.4. Column 1 lists the number of cores used in the SoC designs. Column 2 lists the TAT of the test schedule generated by SFA is employed, and column 3 lists the execution time to obtain the test schedules. Similarly, columns 4 and 5 are the TAT and execution time

for the FLSA, respectively. Columns 6 and 7 list the TAT and execution time for the SABA. In columns 7 and 8, the percentage of TAT reduction by using the FLSA is listed, compared to the TAT by using the SFA and SABA, respectively.

It can be seen that, when using the FLSA, the TAT is in average 13.4% shorter than that from the SFA. The TAT from FLSA is in average 2.9% longer than that from the SABA which is suppose to obtain the solution as close to the optimum as possible but requires much longer optimization times.

Table 3.4: FLSA vs. SFA and SABA

# of Cores	SFA		FLSA		SABA		TAT Reduction	
	TAT	CPU Time(s)	TAT	CPU Time(s)	TAT	CPU Time(s)	from SFA	from SABA
12	1213	0.01	1048	2.74	992	148.31	13.6%	-5.6%
18	1716	0.01	1535	5.41	1513	208.06	10.5%	-1.5%
24	2632	0.01	2318	21.88	2234	229.94	11.9%	-3.8%
30	2274	0.01	1915	32.41	1869	417.08	15.8%	-2.5%
36	3161	0.01	2539	67.52	2494	540.48	19.7%	-1.8%
42	3846	0.01	3334	101.39	3292	631.00	13.3%	-1.3%
48	4328	0.01	3509	151.33	3485	898.77	18.9%	-0.7%
54	4877	0.01	4290	244.36	4051	675.44	12.0%	-5.9%
60	5274	0.01	4692	371.73	4457	2171.73	11.0%	-5.3%
66	5725	0.01	5069	511.88	4917	2321.39	11.5%	-3.1%
72	6538	0.01	5822	720.53	5689	1994.56	11.0%	-2.3%
78	6492	0.01	5769	987.75	5702	3301.45	11.1%	-1.2%
AVG	4006.33	0.01	3486.67	268.24	3391.25	1128.18	13.4%	-2.9%

3.9 Summary

In this chapter, we have presented optimization approaches to minimize the TAT for core-based systems with temperature limits for each CUT and a test-bus width limit. Based on the proposed TSPI technique, we use a CLP model to obtain the optimal solution to the test time minimization problem. Nevertheless, the optimization time of the CLP-based approach is excessively long for large SoC designs. Therefore, we propose a fast heuristic approach to solve the same problem. Based on the initial partitioning scheme, the proposed heuristic algorithm utilizes the flexibility of repartitioning the test sequences and enlarging the cooling periods between test sequences, and generates efficient test schedules. Experimental results have shown the efficiency of the proposed approaches.

Chapter 4
Test Scheduling with Lateral Thermal Influence

This chapter addresses the temperature aware test time minimization problem for the SoCs in which the lateral thermal influence between cores is significant and therefore cannot be ignored. We employ a fast and accurate thermal simulator, ISAC, to obtain instantaneous temperature values of the cores, which are further used to guide the test scheduling algorithm to generated the shortest thermal-safe test schedules.

4.1 Lateral Thermal Influence

In Chapter 3, it is assumed that lateral heat flows between cores can be neglected. This assumption fits a category of SoCs that have relatively large area size and small thickness of the silicon die. However, when the technology scales, the area size decreases while the die thickness is not reduced in the same order of magnitude. This leads to a relatively large contact area between cores. The mismatch of the decreasing rate in geometrical size at the horizontal and vertical

dimensions causes the lateral heat flow taking a higher proportion in the overall heat flow, and therefore cannot be ignored. In this chapter, we take into account the thermal influences between cores and develop a new test scheduling technique in order to guarantee the thermal safety in this new context.

Figure 4.1 depicts a result of thermal simulation performed for an SoC with its die thickness equal to 200 micrometers. The SoC consists of two adjacent cores, both of which have an equal area size. In this experiment, core 1 is tested for a period of 400 microseconds while core 2 remains inactive. It can be seen that core 2 is passively heated by core 1 and the temperature of core 2 increases by 19 degrees. This experimental result illustrates the phenomenon that the temperature of an inactive core is elevated by the active cores in the neighborhood. It indicates that the lateral thermal influence between neighborhood cores should be taken into account when generating a thermal-safe test schedule for this type of SoCs.

As shown in Figure 4.1, due to the significant lateral thermal influence, an inactive core at a lower temperature can be passively heated by those neighbors that have higher temperatures and therefore the temperature of the inactive core is elevated. The degree of the temperature elevation on the inactive core depends on the floor plan, the number of active cores in the neighborhood, and how long the tests last for the active cores in the neighborhood. The temperature elevation on an inactive core is larger, if the active cores are closer to the inactive core, or there are more active cores in the neighborhood, or the tests last longer on the active cores in the neighborhood.

When taking into account the lateral thermal influence and the resulted temperature elevation effect for test scheduling, the spatial distribution of cores and their temperatures, as well as the temporal relations between individual test applications are critically important. They make the thermal-safe test scheduling problem highly complex.

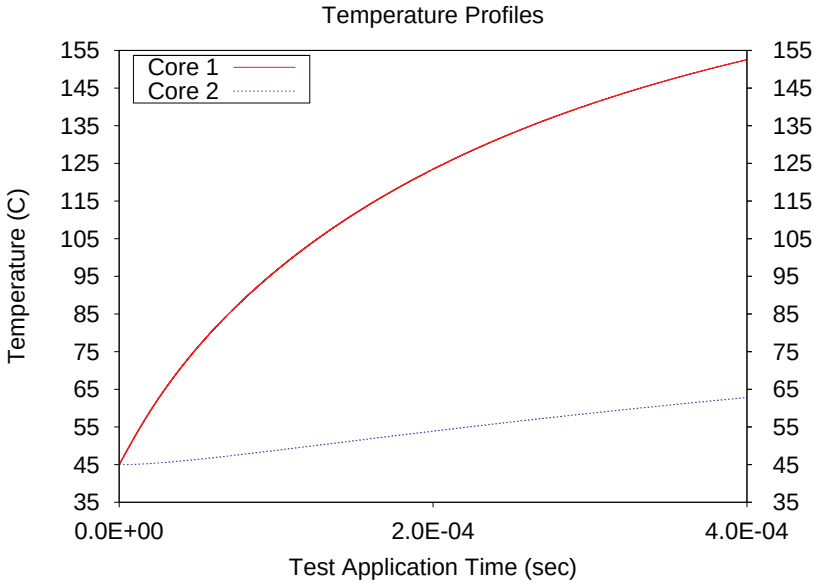


Figure 4.1: Thermal simulation result showing significant lateral thermal influence between two adjacent cores of an SoC design

In Chapter 3, we propose Algorithm 3.1 which determines the initial test set partitioning schemes according to the thermal simulation results of individual cores, and generates the test schedule with minimized TAT. However, Algorithm 3.1 cannot be directly used to solve the temperature aware test time minimization problem when the lateral thermal influence cannot be ignored. Figure 4.2 depicts thermal simulation results for a test schedule generated by Algorithm 3.1. It can be seen that the temperature profiles of the CUTs exceed the temperature limit at several points. This example illustrates that Algorithm 3.1 no longer guarantees the thermal safety in the new context where the lateral thermal influence becomes significant.

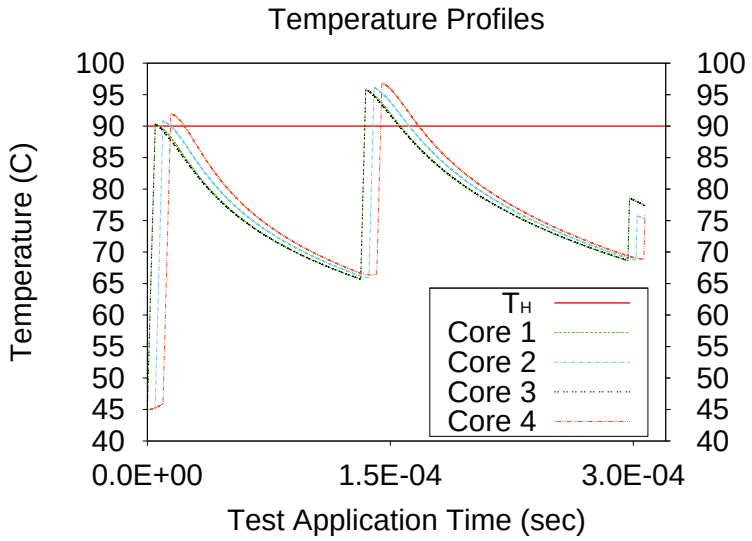


Figure 4.2: Test schedule generated by Algorithm 3.1 leads to violation of the temperature limit due to the significant lateral thermal influence

4.2 Stop-Cooling Temperature

As shown in Figure 3.3, when TSPI technique is used for temperature aware SoC testing, the testing and cooling periods alternate for every core. The testing periods are interrupted at the temperature limit T_H, while the cooling periods are stopped at a relatively lower temperature level. In the thesis, we referred to such a lower temperature level at which the cooling periods are stopped as the stop-cooling temperature (SCT), denoted with T_C. Normally, the temperature curve of a CUT oscillates between T_C and T_H. The gap between T_C and T_H has a large impact on the length of both the cooling periods and the test sequences. Figure 4.3 illustrates a scenario where the test schedule for one of the cores in an SoC varies with respect to different T_C used for test scheduling. Note that, in Chapter 3, we use a different technique rather than the STC to generate the initial partitioning schemes.

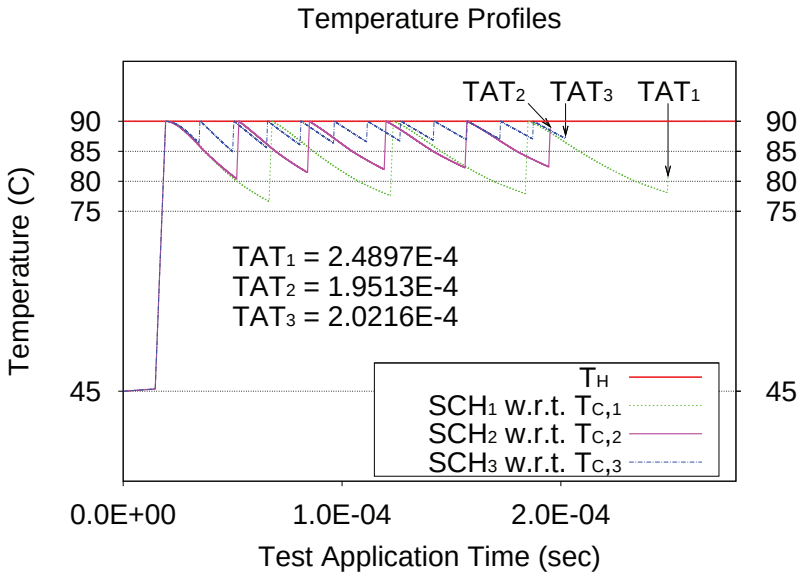


Figure 4.3: Alternative test schedules w.r.t. various SCTs

In Figure 4.3, three different test schedules are depicted, which are denoted with SCH_1, SCH_2, and SCH_3, respectively. The corresponding T_C and TATs are denoted with $T_{C,1}$, $T_{C,2}$, $T_{C,3}$ and TAT_1, TAT_2, TAT_3, respectively. Comparing the temperature profiles of SCH_2 and SCH_3, we find out that SCH_3 uses a higher SCT and has a longer TAT than SCH_2. The main reason why a higher SCT can lead to a longer test schedule is the time overhead introduced when the test controller stops one test and start or resume another test. The detailed reason for the time overhead is explained in Section 3.1. It can be seen that SCH_3 has shorter but more test sequences than SCH_2, indicating that SCH_3 has a larger amount of time overhead.

On the other hand, when comparing the temperature profiles of SCH_1 and SCH_2, we can see that SCH_1 uses a lower SCT and also has a longer TAT than SCH_2. This means that a decreased SCT may not lead to a shorter TAT, although the amount of time overhead is reduced due to the decreased number of test sequences. This is

because the temperature of a core decreases much more slowly at lower temperature levels, and therefore the cooling periods are much longer when a lower SCT is used. If the increase in the length of cooling periods is larger than the decrease in the amount of time overhead, a longer TAT is expected. Thus, in order to generate efficient test schedules, we should use different SCTs to explore alternative TSPI schemes.

4.3 Test Scheduling Approaches

In this chapter, we aim to minimize TATs by generating efficient test schedules with temperature and test-bus width limits. The temperature aware test time minimization problem in the context of this chapter is exactly the same as Problem 3.1 defined in Section 3.2. The test architecture and system model are the same as those presented in Section 3.3 and Section 3.4, respectively. In order to avoid overheating the CUTs during test, the TSPI technique described in Section 3.1 is employed.

We first propose a straight-forward approach (SFA) based on Algorithm 3.1. We also propose a more efficient technique, a thermal-simulation driven test scheduling approach (SDSA), to solve the addressed test time minimization problem. Due to the temporal and spatial thermal interdependencies [Skadron, et al. 2004], [Huang, et al. 2004], coarse grained thermal models cannot provide accurate results for the ICs which has significant lateral thermal influence between cores. In this chapter, in order to obtain accurate instantaneous temperature values with relatively low computational cost, we employ a fast and accurate thermal simulator, ISAC which considers the lateral thermal influences between cores. In the SDSA, a FSM model is developed to control the partitioning and interleaving process, based on which a heuristic algorithm is developed to generate the shortest thermal-safe test schedule. The heuristic algorithm explores alternative test schedules with respect to different SCTs.

4.3.1 Straight-Forward Approach

In order to solve the temperature aware minimization problem with consideration of the lateral thermal influence between cores, we first propose a straight-forward approach (SFA) which is based on Algorithm 3.1. As demonstrated in Section 4.1, Algorithm 3.1 does not generate thermal-safe test schedules for the SoCs with significant lateral thermal influence. However, if we reduce the originally imposed temperature limit to sufficiently lower value and use it in Algorithm 3.1, the generated test schedule can be thermal safe.

We denote the originally imposed temperature limit with $T_{H,orig}$, the new temperature limit with $T_{H,new}$, and the maximum temperature occurred in the thermal simulation result with T_{max}. The difference between the new temperature limit and the originally imposed temperature limit, denoted with d, is defined as

$$d = T_{max} - T_{H,orig} \tag{4.1}$$

and the new temperature limit is given by

$$T_{H,new} = T_{H,orig} - d \tag{4.2}$$

In the SFA, Algorithm 3.1 is invoked with $T_{H,new}$ and a new test schedule is generated. A thermal simulation is performed again to check if the new test schedule is thermal safe. This procedure is repeated until the first thermal-safe test schedule is generated.

The test schedule generated in this way can be excessively long because the adjusted temperature limit may be lower than needed. In order to further reduce the TAT, we use the same procedure to increase the imposed temperature limit until T_{max} is sufficiently close to but smaller than $T_{H,orig}$. The flowchart of the SFA is depicted in Figure 4.4, where D ($D > 0$) denotes a given threshold for d, m denotes the number of iteration steps, and M denotes a given threshold for the total number of iteration steps.

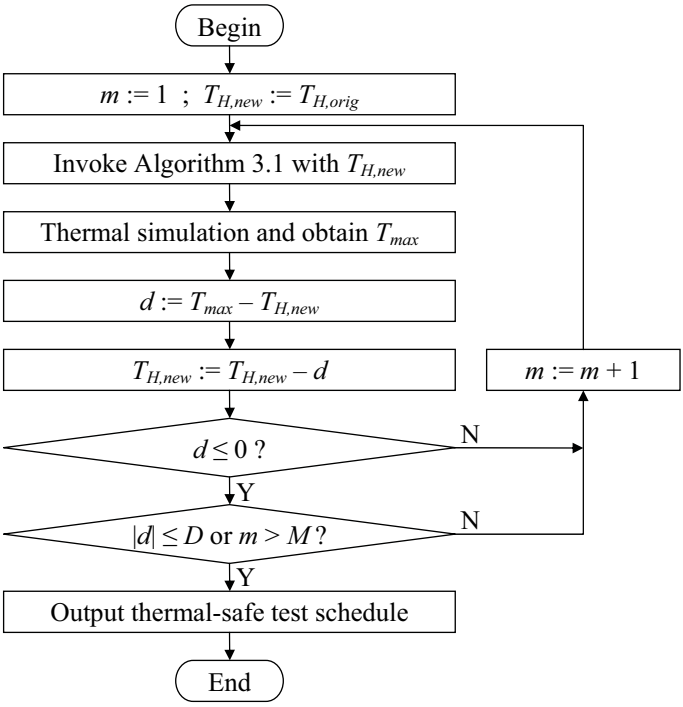


Figure 4.4: Straight-forward approach

4.3.2 Simulation-Driven Scheduling Approach

Although the SFA can generate thermal-safe test schedules, it is not efficient due to the long TAT of the generated test schedules as well as the long execution time of the algorithm. Therefore, we propose a simulation-driven scheduling approach (SDSA) which generates thermal-safe test schedules with short TAT. The SDSA performs thermal simulation to obtain instantaneous temperature values which are further used in a FSM model to guide the TSPI and test scheduling. The developed FSM model is depicted in Figure 4.5.

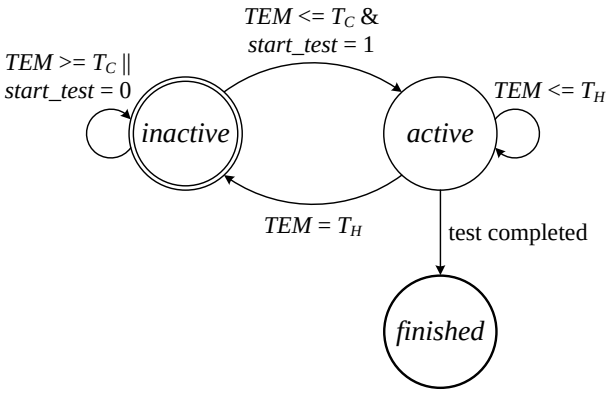


Figure 4.5: FSM model for the SDSA

There are three states defined for a core, namely *inactive*, *active*, and *finished*, corresponding to the status that the core is not tested, the core is tested, and the test for the core is finished, respectively. When test scheduling starts, we assume that all cores are at the *inactive* state and their temperatures are equal to the ambient temperature. When a core is selected (see Algorithm 4.1 in Figure 4.7) for test and the required test-bus width is allocated for the test, a flag *start_test* is set to 1 and the state of the core moves from *inactive* to *active*. While test patterns are applied to the core, the temperature of the core, denoted with *TEM*, increases, and the state of the core remains *active* until the temperature reaches the temperature limit T_H or the test is finished. As soon as the test is finished, the state of the core moves from *active* to *finished*. Otherwise, when the core temperature reaches T_H, the core state moves from *active* to *inactive* and remains unchanged until the core temperature decreases to the stop-cooling temperature T_C, from which the core state moves repeatedly between *active* and *inactive* until the test is finished. The test scheduling algorithm terminates when all cores are at the *finished* state. Figure 4.6 shows a plotted thermal simulation result of a test schedule generated by using the FSM model for an SoC with 4 cores.

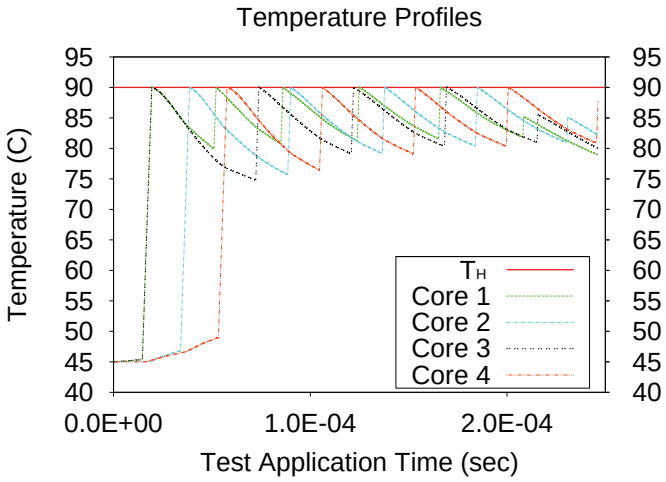


Figure 4.6: Thermal-safe test schedule for an SoC consisting of 4 cores

Using the FSM to guide the test scheduling can guarantee the thermal safety for the generated test schedules. However, the scheduling of test sequences should also take into account the test-bus width limit and the TAT. This is solved by using a heuristic algorithm, denoted with Algorithm 4.1. Its pseudo-code is depicted in Figure 4.7.

Algorithm 4.1 takes a queue of all inactive cores ready for test as an input. It allocates the required test-bus width to some of the cores and changes their states to *active*. The algorithm first sorts the queue decreasingly according to the number of remaining test patterns divided by the current core temperature (line 2). This means that a higher priority is given to a core which has a larger number of remaining test patterns and a lower temperature. In this way, the physical parameters including the sizes of cores and the distances between cores have been taken into account, because the temperature values of the cores are given by the thermal simulator which considers the lateral thermal influence. Then the heuristic algorithm allocates the required test-bus width to the cores according to their priorities until there is no sufficient test-bus width to allocate or all cores have been activated for test. (lines 3 through 13).

Algorithm 4.1: *Activate*(Queue of inactive cores ready for test :: *Q*)

```
01: if (IsNotEmpty(Q)) then
02:     Sort Q decreasingly according to
        (#_of_rem_test_patt × core_defect_prob / curr_tem);
03:     while (GetRemainingBandwidth() > 0 & IsNotEmpty(Q)) loop
04:     CurrentCore = GetFirstElement(Q);
05:         ReqBwd = GetBandwidthRequirement(CurrentCore);
06:         if (ReqBwd <= GetRemainingBandwidth()) then
07:             Move the state of CurrentCore to active;
08:             SubtractBandwidthRemainder(ReqBwd);
09:             Remove(CurrentCore, Q);
10:         else
11:             break loop;
12:         end if-then-else
13:     end while
14: end if
```

Figure 4.7: Pseudo-code of heuristic algorithm activating cores for test

The overall strategy of the SDSA is illustrated in Figure 4.8. The test scheduling algorithm iteratively explores alternative solutions by using different SCTs. In every iteration step, a thermal-safe test schedule is generated by invoking Algorithm 4.1 with a new SCT, denoted with $T_{C,new}$. A counter k is used to count the number of consecutive iteration steps in which the reduction of TAT is not larger than a given threshold ε ($\varepsilon > 0$). If the TAT of the newly generated test schedule is less than the minimal TAT of the best solution obtained throughout the previous iteration steps, the current solution is recorded as the best solution. Further, if the reduction of TAT is greater than ε, counter k is reset to 0. In the cases that the current TAT is larger than the minimal TAT or the reduction of TAT is less than ε, counter k is incremented by 1. This procedure repeats until k is larger than a given threshold K. Thereafter the optimized test schedule is output and the test scheduling process terminates.

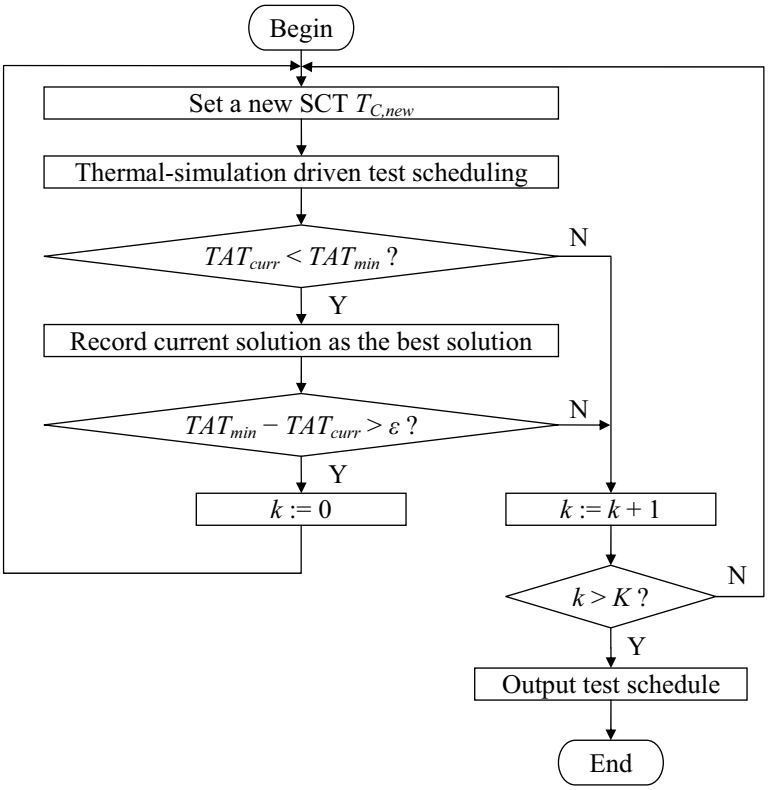


Figure 4.8: Overall solution strategy of the SDSA

By checking the temperature value of each core at every simulation cycle, the test scheduling algorithm restricts the core temperature between T_C and T_H, after the core temperature is raised from the ambient temperature to T_C. With respect to different SCTs, alternative test schedules based on various TSPI schemes are explored. Figure 4.9 depicts the TATs with respect to different SCTs for an SoC consisting of four cores. The best T_C found by the heuristic algorithm is 84.065°C and the corresponding TAT is 2.4629×10^{-4} seconds.

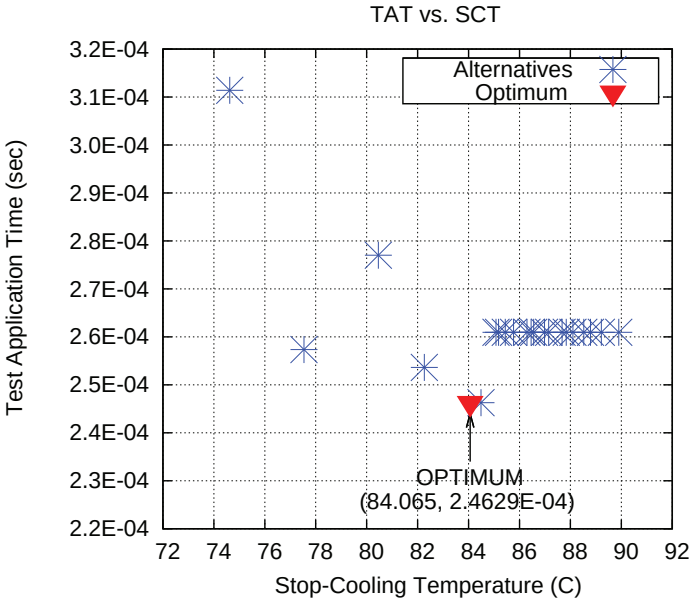


Figure 4.9: TAT vs. SCT

4.4 Experimental Results

ISCAS'89 benchmark circuits are used as cores of the SoC designs for our experiments. The numbers of cores in the SoC designs varies from 4 to 36. The power consumption of a test is obtained through the same method mentioned in Section 3.7.3. With the obtained power consumption values, the thermal simulator ISAC is used to obtain instantaneous temperatures in test scheduling. The imposed temperature limit is 90°C, and the assumed frequency of test application is 100MHz.

We compare the SDSA with the SFA. The experimental results are shown in Table 4.1. Column 1 lists the number of cores used in the SoC designs. Columns 2 and 4 list the TATs of test schedules generated for the corresponding SoC designs, using the SFA and

SDSA, respectively. Columns 3 and 5 list the execution times of corresponding algorithms. Column 6 shows the percentage of the TAT reduction by using the SDSA versus SFA. It can be seen that by using the SDSA, the TAT is reduced by about 25% to 61% for different SoC designs. The execution times of the SDSA are usually shorter than those of the SFA. This is because, in the SFA, each time when Algorithm 3.1 is invoked, a thermal simulation is performed for every core in order to generate the initial partitioning schemes according to the new temperature limit.

Table 4.1: SDSA vs. SFA

# of Cores	SFA		SDSA		TAT Reduction
	TAT (s)	CPU Time (s)	TAT (s)	CPU Time (s)	
6	3.9129E-4	1078	2.1013E-4	1118	46.298%
8	3.2827E-4	4122	2.4474E-4	1222	25.446%
12	4.4911E-4	3118	2.3117E-4	1265	48.527%
18	3.6927E-4	7458	2.0832E-4	1193	43.586%
24	4.5970E-4	6681	2.1004E-4	1259	54.309%
30	5.4901E-4	12705	2.2601E-4	1357	58.833%
36	5.7715E-4	11760	2.2360E-4	1400	61.258%

4.5 Summary

This chapter presents a thermal-safe test scheduling technique to minimize the TAT of SoC with significant lateral thermal influence. The test scheduling algorithm uses a FSM model and the instantaneous temperature values obtained from thermal simulations to partition, interleave, and schedule the test sets. The TAT is minimized such that the temperature limit and the test-bus width limit are satisfied.

Chapter 5
Multi-Temperature Test Scheduling

This chapter addresses the multi-temperature test scheduling issue. We propose a test scheduling technique that generates the shortest test schedules such that the tests are applied only when the temperature of CUT is within a given interval and the test-bus width limit is satisfied. We employ the TSPI technique and the heating patterns in order to ensure the temperature of a CUT is within the given interval whenever the test is applied. In test scheduling, a thermal simulator ISAC is used to obtain instantaneous temperatures of the CUTs and a FSM model is used to manage the temperatures of the cores.

5.1 Problem Formulation

We assume the same test architecture as the one described in Section 3.3. The system model is the same as that in Section 3.4. In order to sensitize temperature-dependent defects, we need to apply tests to an SoC at different temperature spectra. Each temperature spectrum is specified as a given temperature interval $I = (T_L, T_H)$,

where T_L and T_H are the temperature lower limit and upper limit, respectively. In this chapter, it is assumed that a test should be applied only when the temperature of core C_i (i = 1, 2, ... , n), denoted with T_i, is within the temperature interval I.

The problem we address in this chapter is to minimize the TAT by generating an efficient SoC test schedule such that the following two constraints are satisfied: (1) the amount of test-bus width required by the concurrent tests is less than or equal to the test-bus width limit; (2) a test has to be applied when and only when the temperature of the core is within the given temperature interval. The problem formulation is given in Figure 5.1. A multi-temperature testing problem can be further formulated as a set of such test scheduling problems associated with different temperature intervals.

Problem 5.1: Minimization of TAT for a given temperature interval

Input:

An SoC together with the physical configuration F of the die and package as well as the floor plan of the SoC;

A set of test set for each core $\{TS_i \mid i = 1, 2, ... , n\}$;

A set of required test-bus width for each test $\{W_i \mid i = 1, 2, ... , n\}$;

Test-bus width limit B;

Temperature upper limit T_H and temperature lower limit T_L.

Output:

A test schedule with the minimal test application time.

Constraints:

1. At any time moment t before all tests are completed, the total amount of allocated test-bus width $W(t)$ is less than or equal to test-bus width limit B, i.e. $\forall t$, $W(t) \leq B$, where $W(t) ::= \Sigma_j W_j(t)$;

2. At any time moment u when a test is applied to core C_i, the instantaneous temperature $T_i(u)$ of the core C_i is less than the temperature upper limit T_H, and greater than the temperature lower limit T_L, i.e. $T_L < T_i(u) < T_H$.

Figure 5.1: Problem formulation of multi-temperature test scheduling

5.2 Test Scheduling within a Temperature Interval

In order to sensitize faults at a certain temperature level, a test should be applied to the core only when its temperature is within a temperature interval between an upper limit and a lower limit. Whenever the temperature of a core exceeds the upper limit, the test should be stopped and the core is turned into an idle state in which no dynamic power is dissipated and the core temperature decreases. When the temperature of the core decreases to a certain level, the test can be resumed if the test bus has sufficient width to transport the test data for the core. Thus, we use the TSPI technique presented in Section 3.1 for the multi-temperature SoC test scheduling.

5.2.1 Heating Sequence

Ideally, we expect that the temperature of a core is always maintained within the given temperature interval whenever a test is applied. However, this condition does not always hold in reality. Sometimes, the core temperature may decrease below the lower limit of the temperature interval. One reason for the decrease in the temperature of a core is that the test patterns consume insufficient power and the amount of heat generated by applying the test patterns is less than the amount of heat dissipated by the physical cooling system. Another reason is that no sufficient test-bus width is available for a test and it has to be postponed until the test-bus width requirement is satisfied.

If the problem of temperature decrease is not properly addressed in test scheduling, it can cause invalid test schedules where tests may be applied at temperatures below the lower limit and cannot screen the targeted defects. In order to solve this problem, we apply a sequence of dummy patterns that consume sufficiently high power and raise the core temperature towards the lower limit T_L. We refer to such a high-power test pattern as a heating pattern (HP) and a sequence of heating

patterns as a heating sequence (HS). It should be noted that transporting a heating pattern through the test bus requires the same amount of test-bus width as what transporting a test pattern requires.

The minimal length of a heating sequence (denoted with L_{min}) preceding a test sequence is the number of heating patterns needed to raise the core temperature to T_L. If the test sequence following a heating sequence does not consume sufficiently high power and causes the core temperature to decrease, the required length of the heating sequence (denoted with L_{req}) should be larger than L_{min}. The actual value of L_{req} depends on the temperature profile of the succeeding test sequence. Figure 5.2 shows the temperature profiles of a core stimulated with a heating sequence and a test sequence consecutively, depicted with doted and solid curves, respectively. The core temperature is T_s when the HS starts. Three lengths are chosen for the HS, namely L_1, L_2, and L_3, while the length of the succeeding test sequence is M. The HS is too short in Figure 5.2(a) and too long in Figure 5.2(b), causing the core temperature going out of the interval (T_L, T_H) during the test application period. The HS is given a proper length in Figure 5.2(c) such that the core temperature reaches a medium value (denoted with T_M) between T_L and T_H before the test starts and remains inside the interval during the test application period.

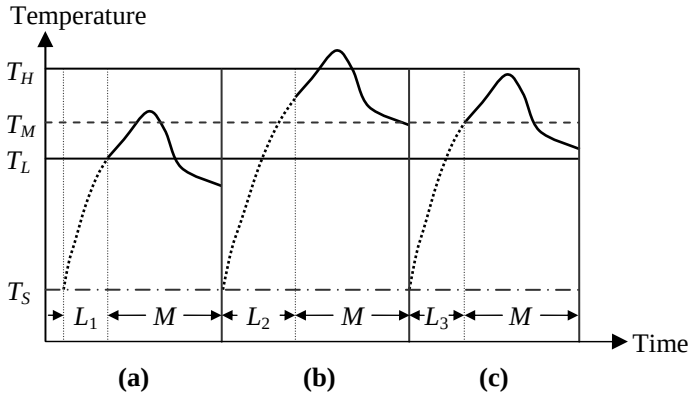


Figure 5.2: The impact of heating sequence length

In order to avoid frequently violating the temperature limits due to improperly determining the lengths of heating sequences, we propose a preprocessing approach for each test set *S*, before we perform the test scheduling algorithm. We define an observation frame (OF) for each test pattern of a test set, and the OF contains *D* consecutive test patterns. For each OF, we calculate the average power consumption P_{OF} of all test patterns in the OF. We categorize an observation frame to be a low-power frame (LPF) if its P_{OF} is smaller than a threshold power value P_{THD}, or a high-power frame (HPF) if otherwise. P_{THD} is defined as a power consumption value that ultimately causes the core temperature to reach a steady-state at T_M. We perform a series of steady-state temperature analysis to find P_{THD}.

Before scheduling a test sequence, we must determine the required length L_{req} of its preceding HS. If the OF associated with the first test pattern of the test sequence is a LPF, L_{req} should be equal to the number of heating patterns that heats the core to T_M. Otherwise, L_{req} should be equal to L_{min}.

5.2.2 FSM for Thermal Management in Test Scheduling

As a part of the test scheduling algorithm, we develop a finite state machine to control the states of cores. A core has the following states: *heating*, *testing*, *cooling*, *waiting*, and *complete*, defined as follows.

Testing: the core is tested within the temperature interval (T_L, T_H).

Cooling: the core is passively cooled down without any test pattern applied, and its temperature is decreasing from T_H towards T_L.

Heating: the core is actively heated by heating patterns and its temperature is increasing.

Waiting: the core is waiting for allocation of sufficient amount of test-bus width for its test and the temperature of the core is usually below T_L.

Complete: the core has finished its test.

Figure 5.3 depicts the temperature profile of a core and illustrates the relation of core state and temperature. When the test scheduling process starts, we assume that all cores are at the *waiting* state and their temperatures are equal to the ambient temperature T_A ($T_A < T_L$). Each core is associated with a dedicated flag *start*, indicating that the core is chosen for test if it is equal to 1, or is not chosen for test if otherwise. A core remains within the *waiting* state until it is selected for test. From the *waiting* state, a core can move to the *heating* state if its temperature T is below T_L, or to the *testing* state if T is already within the imposed temperature interval. In the *heating* state, a core is applied with heating patterns and its temperature increases to T_L or T_M, depending on whether the observation frame of the succeeding test pattern is an HPF or LPF, respectively. As soon as the core temperature T exceeds T_L or T_M, the state of the core is changed to the *testing* state. The core stays in the *testing* state as long as its temperature T remains inside the temperature interval if the test is not finished. Otherwise, the core moves to the *cooling* state when T exceeds T_H, or the *waiting* state when T falls below T_L, or the *complete* state when the test is finished. In the *cooling* state, a core is supposed to be cooled down until T reaches to a stop-cooling temperature T_C ($T_C \geq T_L$), after which the core moves to the *testing* state if it is selected for test, or remains in the *cooling* state until it is moved to the *waiting* state if it is not selected for test. The entire SoC test finishes after all cores reach the *complete* state. Figure 5.4 illustrates the five states and the transitions between the states.

It should be noted that a cooling period ends at the stop-cooling temperature T_C where $T_C \geq T_L$. The purpose of introducing T_C is to further reduce the TAT, especially when cooling a core to T_L needs substantially long time. We have developed a heuristic algorithm, to search for the best T_C between T_L and T_H. The heuristic algorithm is an iterative algorithm that sets a new T_C for each iteration step and invokes the proposed test scheduling algorithm to calculate the TAT with respect to the current T_C. The heuristic algorithm returns the T_C with which the TAT is the shortest among all iterations.

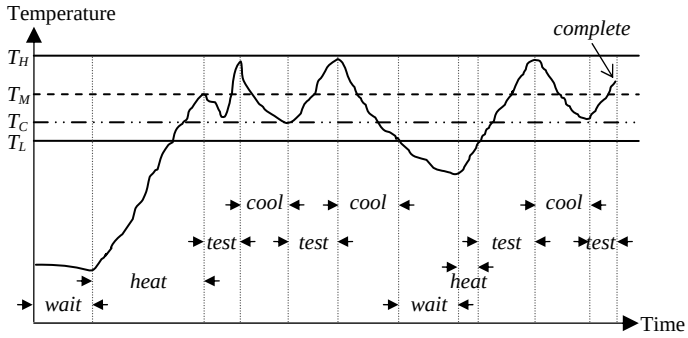


Figure 5.3: Core states w.r.t. changes of temperatures

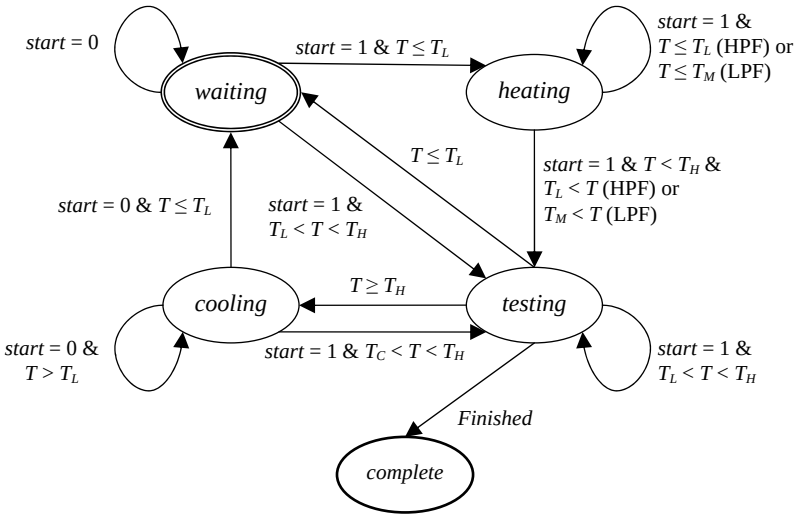


Figure 5.4: FSM model for multi-temperature test scheduling

5.2.3 Test Scheduling Algorithm

We propose a test scheduling algorithm to generate a test schedule that meets the temperature and test-bus width requirements. The test scheduling approach employs the thermal simulator ISAC to obtain instantaneous temperature values which are used by the proposed

finite state machine to control state transitions for every core. The test scheduling algorithm uses the FSM model to update the state of every core at every simulation cycle, and maintains a queue of cores in the *waiting* state in order to activate some cores for test, if available. The test scheduling algorithm stops when all cores reach the *complete* state.

We develop an algorithm, denoted with Algorithm 5.1, to activate cores for test, and its pseudo-code is depicted in Figure 5.5. The algorithm takes the queue (Q) of all cores in the *waiting* state as an input. According to the ratio r of the number of remaining test patterns to the current temperature of each core (line 1), Algorithm 5.1 selects as many cores as possible to start/resume their tests, if their test-bus width requirements can be met. A core that has a relatively larger number of remaining test patterns and is relatively colder gets a higher value of r and hence is given higher priority to be scheduled for test. As such, the scheduling algorithm takes into account the impact of the distance between cores on the temperature, since a core located further away from a hot core is more likely to have a lower temperature than the direct neighbors of the hot core. By allocating the required test-bus width to the selected cores (line 5) and changing their states to *testing* (line 6), the algorithm activates as many cores as possible for test.

Algorithm 5.1: *Activate*(Queue of cores in the *waiting* state :: *Q*)

```
01: Sort Q decreasingly according to r
        where r ::= # of remaining test patterns / core temperature
02: C = GetFrontElement(Q);
03: while (RemainingBusWidth() > 0 & IsNotEmpty(Q)) loop
04:     if (RequiredBusWidth(C) ≤ RemainingBusWidth()) then
05:         AcquireBusWidth(C);
06:         ChangeState(C, testing);
07:         RemoveElement(C, Q);
08:     end if
09:     C = GetNextElement(Q);
10: end while
```

Figure 5.5: Pseudo-code of the algorithm activating cores for test

5.3 Experimental Results

We select ISCAS'89 benchmark circuits as cores of 6 different SoCs. The number of cores in these SoCs varies from 4 to 42. The cores have different physical sizes, depending on their complexity. We use the same power estimation method as the one used for experiments presented in Section 3.7.3 to calculate the power consumption (in Watt) of a core. Taking the floor plan of an SoC and the power consumption profiles of individual cores as inputs, the thermal simulator calculates instantaneous temperatures of all individual cores at every cycle of the test process. The assumed scan frequency is 100MHz.

We employ the proposed test scheduling technique to generate test schedules for the SoCs. Two groups of experiments are performed to generate different test schedules for each SoC with respect to different temperature intervals (at low, medium, and high temperature levels) as well as to different test-bus width limits (low, marginal, and high), respectively.

Table 5.1 shows the impact of the temperature interval on the TAT. The first column lists the numbers of cores in the SoC designs. Columns 2, 4, and 6 list the TATs (in number of cycles) of the generated test schedules with respect to different temperature intervals. Columns 3, 5, and 7 list the CPU times (in seconds) for the generation of the corresponding test schedules. The test-bus width limit for the experiments in this group is 60 bits. The experimental results show that the test schedule length decreases along with increasing temperature level at which the tests should be applied. This is because it takes a longer time to cool down a core when a test is applied at a lower temperature level.

Table 5.2 shows the impact of the test-bus width on the TAT. The first column lists the number of cores in the SoC designs. Columns 2, 4, and 6 list the TATs (in number of cycles) of the generated test schedules with respect to different test-bus width limits. Columns 3, 5, and 7 list the corresponding CPU times (in seconds) the generation of

the corresponding test schedules. The imposed temperature interval for the experiments in this group is 85-100°C. It can be seen that the length of test schedule decreases with increasing test-bus width limit.

Table 5.1: TATs with different temperature intervals (B=60)

# of Cores	T_L=65C, T_H=80C		T_L=85C, T_H=100C		T_L=105C, T_H=120C	
	TAT	CPU Time (s)	TAT	CPU Time (s)	TAT	CPU Time (s)
4	59887	347	29651	171	19562	115
8	61014	404	30256	180	20194	124
16	64658	411	31023	195	21055	138
25	71913	433	35785	214	24798	152
36	74886	477	37249	221	26402	168
42	76102	490	37989	243	27031	174

Table 5.2: TATs with different test-bus width (T_L=85°C, T_H=100°C)

# of Cores	B=40		B=60		B=80	
	TAT	CPU Time (s)	TAT	CPU Time (s)	TAT	CPU Time (s)
4	29821	145	29651	171	29648	177
8	30261	182	30256	180	29752	197
16	31623	210	31023	195	34613	218
25	38391	252	35785	214	35415	230
36	38568	267	37249	221	35936	245
42	39785	264	37989	243	36430	251

The third group of experiments compares the TATs of test schedules generated using different SCTs, either T_C found by the heuristic algorithm or the given lower limit T_L. Table 5.3 shows the impact of the SCT on the TAT. The fist column lists the number of cores in the designs. Columns 2 and 4 list the TATs of test schedules using T_L and T_C as the SCT, respectively. Columns 3 and 5 show the CPU times (in seconds) for test scheduling. The TAT reduction (in percentage) is listed in Column 6. The experiments in this group are performed with a temperature interval 85-100°C and a test-bus width limit of 60 bits. It is seen that using the T_C found by the heuristic algorithm reduce the TAT by up to about 9% rather than using T_L. Similar results are shown in Table 5.4 where the temperature interval is 65-80°C and the test-bus width limit is 60 bits. With this temperature interval, the TAT reduction is up to about 20%.

The third group of experimental results indicates that using T_C rather than T_L for test scheduling leads to a greater reduction on the TAT when the temperature interval is imposed at a lower temperature level. On the other hand, the CPU time for test scheduling becomes substantially longer because of the increased time for determining T_C.

Table 5.3: TATs with/without T_C (B=60, T_L =85°C, T_H =100°C)

# of Cores	Use T_L (85°C) as T_C		Use T_C found by HA		TAT Reduction
	TAT	CPU Time (s)	TAT	CPU Time (s)	
4	29651	171	28711	1265	3.17%
8	30256	180	29142	1327	3.68%
16	31023	195	29779	1402	4.01%
25	35785	214	33654	1511	5.96%
36	37249	221	34372	1776	7.72%
42	37989	243	34627	1843	8.85%

Table 5.4: TATs with/without T_C (B=60, T_L =65°C, T_H =80°C)

# of Cores	Use T_L (65°C) as T_C		Use T_C found by HA		TAT Reduction
	TAT	CPU Time (s)	TAT	CPU Time (s)	
4	59887	347	52691	2340	12.02%
8	61014	404	52746	2366	13.55%
16	64658	411	55376	2587	14.36%
25	71913	433	59162	2830	17.73%
36	74886	477	60701	2865	18.94%
42	76102	490	60935	2884	19.93%

5.4 Summary

In this chapter, we address the problem of long test application time when applying multi-temperature testing to systems-on-chip. We propose a test scheduling approach to minimize the TAT such that a test is applied only when the core temperature is within a given interval and the test-bus width limit is satisfied. The proposed test scheduling technique employs a thermal simulator to partition and interleave test sets on-the-fly and uses a FSM model to manage the state transitions for all cores. Experimental results show that, in general, the TAT is longer when a test is applied at a lower temperature level and/or with a lower test-bus width limit. Moreover, the TAT can be further reduced by stopping the cooling periods at an explored temperature rather than at the imposed temperature lower limit, especially for the tests applied at a low temperature level.

Chapter 6
Defect-Probability Driven Test Scheduling

In this chapter, we address the test time minimization problem for volume-production SoC tests using the AOFF approach. We employ an hybrid BIST, in which a test set is composed of both pseudorandom and deterministic test patterns, for core-based SoCs. In order to minimize the expected test application time, we take into account the defect probabilities of individual cores in test scheduling. A heuristic algorithm is proposed for test scheduling.

6.1 Problem Formulation

6.1.1 Basic Definitions and Assumptions

In this chapter, we employ the test architecture depicted in Figure 2.7 for hybrid BIST. In the test architecture, every core has its dedicated BIST circuit. Moreover, a single test bus is used to transport deterministic test data between the CUTs and the embedded tester. In order to test a core, a set of test patterns are generated. A test set may

consist of both deterministic test patterns (DTPs) and pseudorandom test patterns (PTPs). A subset of the DTPs in the test set is referred to as a deterministic test sequence (DTS), and a subset of PTPs in the test set is referred to as a pseudorandom test sequence (PTS).

Due to the use of BIST circuits for every core and the single test bus for deterministic tests, we assume that PTPs for different cores can be concurrently applied, while the DTPs can only be applied sequentially. Figure 6.1 depicts a hybrid BIST test schedule for a system consisting of 5 cores, where TS_i denotes the test set for core C_i (i = 1, 2, ... ,5). The white and grey rectangles represent the DTSs and the PTSs, respectively. In this example, DTPs are scheduled to be applied sequentially, while PTPs for different cores are scheduled to be applied in parallel.

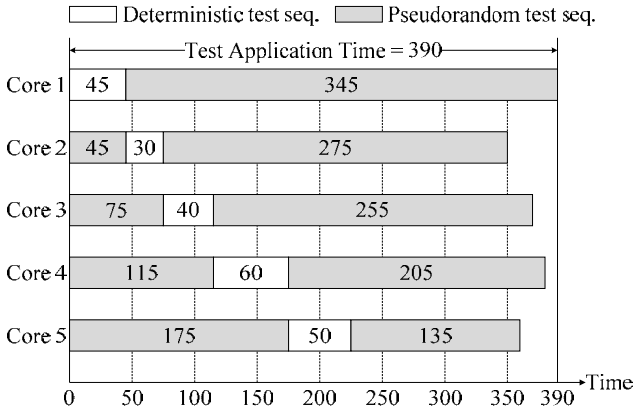


Figure 6.1: A hybrid BIST schedule example

In this chapter, the test set, the deterministic test sequence, and the pseudorandom test sequence for core C_i ($1 \leq i \leq n$) are denoted with TS_i, DTS_i, and PTS_i, respectively. In the cases that more than one deterministic test sequence or pseudorandom test sequence is partitioned from the original test set, DTS_{iv} and PTS_{iw} denote the v-th deterministic test sequence and the w-th pseudorandom test sequence in test set TS_i, respectively. Suppose that the number of deterministic

test patterns and pseudorandom test patterns in test set TS_i are d_i and r_i, respectively. The j-th ($1 \le j \le d_i$) deterministic test pattern of DTS_i is denoted with DT_{ij}. The k-th ($1 \le k \le r_i$) pseudorandom test pattern of PTS_i is denoted with PR_{ik}.

In this thesis, the defect probability of a core, in short, core defect probability (CDP), is defined as the probability of a core having defects. We denote the defect probability of core C_i ($1 \le i \le n$) with CDP_i. Similarly, the defect probability of an SoC, in short, system defect probability (SDP), is defined as the probability of an SoC having defects, meaning that some cores are defective.

We assume that the defect probabilities of different cores in an SoC are independent. Then, the SDP is given by

$$SDP = 1 - \prod_{i=1}^{n} \left(1 - CDP_i\right) \tag{6.1}$$

We suppose that a test process can be terminated with a certain probability. The probability of a test process being terminated at a certain time moment depends on the following two probabilities: (1) the probability of an individual test being terminated due to detection of faults, referred to as individual-test failure-probability (ITFP); (2) the probability of an individual test being passed with no faults detected, referred to as individual-test success-probability (ITSP).

We assume that the failure probabilities of individual tests for the cores in an SoC are independent, meaning that the probability of detecting faults in a core does not depend on that in another core. We also assume that the success probabilities of individual tests for the cores in an SoC are independent, meaning that the probability of detecting no faults in a core does not depend on that in another core.

In this chapter, we assume that a deterministic test cannot be interrupted by the pseudorandom test for the same core. On the other hand, we assume that a pseudorandom test can be interrupted by the deterministic test for the same core. More concretely, this means that the following scenario can occur: a pseudorandom test is stopped at a certain moment, and, after the application of the entire deterministic test set for the same core, the pseudorandom test resumes until the

completion. We make this assumption due to the following concerns. The signature of a pseudorandom test is available only when the test is completed while the TAT of a pseudorandom test is usually very long. If we are allowed to interrupt a pseudorandom test and analyze the signature more frequently, we can terminate the test earlier if faults are detected and hence can shorten the TAT. However, frequently switching deterministic and pseudorandom tests for a core introduces time overheads [Goel, et al. 2003]. Thus, we assume that a pseudorandom test can be interrupted at most once. Under such assumption, the time overhead is substantially small and therefore can be ignored.

Furthermore, in this chapter, we assume the deterministic tests for different cores are scheduled sequentially and consecutively, due to the following concerns. First, deterministic test patterns are considered more efficient since usually a deterministic test pattern can cover more faults than a pseudorandom test pattern. Second, the test responses of deterministic test patterns can be obtained at each test application cycle, and therefore a deterministic test can be terminated at the end of any test application cycle if faults are detected. This also infers that we do not need to delay a deterministic test in order to insert a long pseudorandom test.

6.1.2 Possible Test Termination Moment

When the AOFF approach is employed for a hybrid BIST, there are two possible scenarios regarding the termination of the test process. During the application of a deterministic test sequence, the test response is captured as soon as a test pattern has been applied. By analyzing the test response, the test can be aborted immediately, if faults are detected. On the other hand, during the application of a pseudorandom test sequence, the signature is not available until all the pseudorandom test patterns in the test sequence have been applied. By analyzing the obtained signature, the test can be aborted, if faults are detected. Thus, using the AOFF approach, the test process is possible

to be terminated at the end of every test application cycle of the deterministic test patterns, or at the end of every application period of a pseudorandom test sequence. This discussion leads to the notion of possible test termination moment (PTTM).

A PTTM is a time moment when the test process can be terminated due to detection of faults. According to the discussion on the termination of the test process, a PTTM is the time moment immediately after a deterministic test pattern or a pseudorandom test sequence has been applied and the test response or the signature has been analyzed.

For a given test schedule, all PTTMs are known. Figure 6.2 illustrates the PTTMs in a test schedule for an SoC with 5 cores. In this example, the DTPs are depicted with white rectangles and the PTSs are depicted with grey rectangles. The grey solid lines indicate the PTTMs at which a DTP has been applied, e.g. PTTMs 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10. The black dotted lines indicate the PTTMs at which a PTS has been finished, e.g. PTTMs 4, 5, 7, 9, 10, 12, and 13. Note that some of the PTTMs are considered identical, since they overlap at the same time moment, e.g. PTTMs 4, 5, 7, 9, 10, and 12.

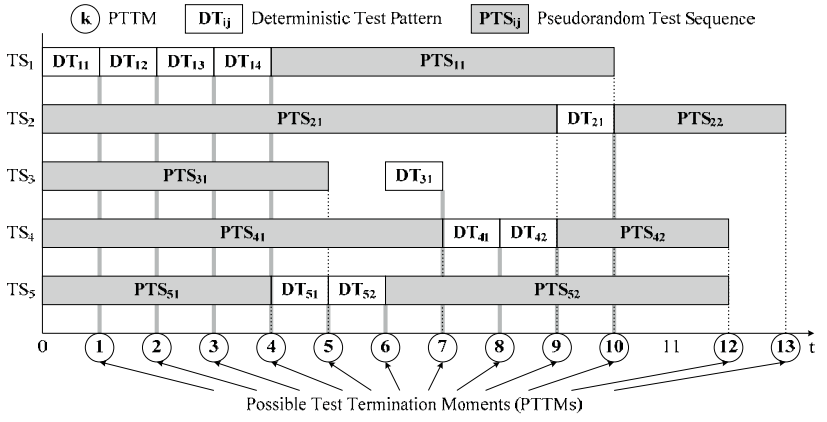


Figure 6.2: Possible test termination moments in a test schedule

From this discussion, we can see that a pseudorandom test sequence can be treated as a single test pattern, since they have the same effect on test termination. It should be noted that a test application cycle of a test pattern differs in combinatorial-circuit testing and the scan-based testing. In combinatorial-circuit testing, applying a test pattern needs one clock cycle, whereas in scan-based testing, a test application cycle of test patterns includes three stages, scan-in, capture, and scan-out, as explained in Section 3.1.

6.1.3 Expected Test Application Time

We consider the termination of the test process at a certain moment as a random event which happens with a certain probability. Thus, the TAT is a random variable, and its mathematical expectation, referred to as the expected test application time (ETAT), is the expected value of the actual TATs.

Let A_x be the random event that the test process is aborted at PTTM x, and let T be the random event that the test process is passed on completion. Then, the ETAT is given by

$$ETAT = \sum_{\forall x \in X} \left(t_x \times p\left[A_x\right] \right) + L \times p\left[T\right] \tag{6.2}$$

where x is a PTTM, X is the set of all PTTMs, t_x is the TAT by the moment x, L is the TAT by the completion moment, $p[A_x]$ is the probability of event A_x, and $p[T]$ is the probability of event T.

In Equation (6.2), the ETAT is presented as a sum of two literals. The first literal corresponds to the case in which the test process can be terminated at different PTTMs because at least one individual test has detected faults. The second literal corresponds to the case in which the test process is passed on completion without detection of any faults. Indeed, Equation (6.2) interprets the ETAT as the sum of the probabilistic TATs at different PTTMs.

It should be noted that two different events A_x and A_y ($x \neq y$) are exclusive, i.e. $\forall x, y \in X, x \neq y, A_x \cap A_y = \varnothing$. Events A_x and T are also

exclusive, i.e. $\forall x \in X, A_x \cap T = \varnothing$. The reason is that, if the test process is terminated at a certain moment x ($x \in X$), it must have passed all the moments earlier than x and it will never go through any moments later than x. In another word, if A_x ($x \in X$) happens, any other event A_y ($\forall y \in X, y \neq x$) as well as T cannot happen.

In order to know whether the test process is aborted or not at any PTTM x, we have to check every individual test to see if they have detected faults by the moment x. The test process is aborted at PTTM x, if and only if both of the following two conditions are satisfied: (1) at least one of the tests that are stopped at PTTM x to analyze test responses/signatures detects faults; (2) all the other tests that are not able to be stopped at PTTM x had not detect any faults until their latest passed PTTMs before x. Therefore, A_x is equivalent to the intersection of the following two events: one event is that at least one of those tests which are just stopped at PTTM x detect faults; and the other event is that those tests which are not able to be stopped at the moment x had not detected any faults until the latest PTTMs when they were stopped for a check.

Let Y_x be the set of all individual tests that are stopped at PTTM x, let Z_x be the set of all individual tests that are not able to be stopped at PTTM x, let $F_x(y)$ be the event that the individual test y detects at least one fault at PTTM x, and let $P_x(z)$ be the event that the individual test z had not detected any faults until the latest PTTM before x when z was stopped to for a check. Then, event A_x is given by

$$A_x = \left(\bigcup_{\forall y \in Y_x} F_x(y) \right) \cap \left(\bigcap_{\forall z \in Z_x} P_x(z) \right) \tag{6.3}$$

Figure 6.3 shows an example where the test process is aborted at PTTM 7. This means that, at PTTM 7, at least one of the two partial tests TS_3 and TS_4 has detected faults, and the other partial tests TS_1, TS_2, and TS_5 had not detect any faults until the latest moments when they were stopped for a check. More specifically, TS_1 had not detected any faults until PTTM 4, TS_2 had not detect any faults since it has

never stopped until the current PTTM, and TS_5 had not detected any faults until PTTM 6.

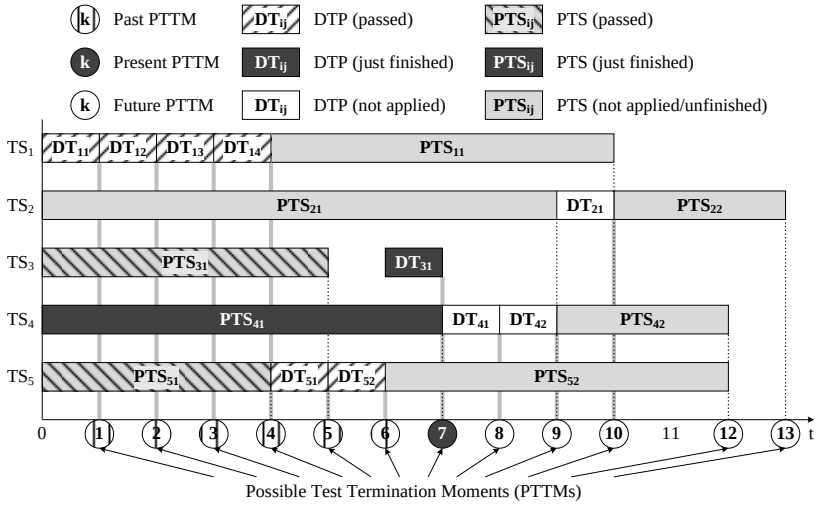


Figure 6.3: Example of the test process aborted at PTTM 7

Let E be the set of all tests that are completed without detection of faults, and let $P(e)$ be the event that the test e has not detected faults until completion. Then, event T is given by

$$T = \bigcap_{\forall e \in E} P(e) \tag{6.4}$$

According to the definition of PTTM, at PTTM x, Y_x should not be empty and at least one test belonging to Y_x should detect faults, otherwise the test process would have not been aborted at PTTM x. Moreover, for a test $y \in Y_x$, it should be the currently checked DTP or PTS that detects the faults, and the DPT(s) and PTS(s) that were finished before x should not detect any faults, otherwise the test had already been aborted earlier. On the other hand, at PTTM x, all the tests in Z_x should have not detected any faults so far, otherwise the test process would have been aborted earlier and would not have reached PTTM x. Table 6.1 lists the sets Y_x and Z_x at every PTTM x in the

example depicted in Figure 6.2. The set E includes all the individual tests. For the example in Figure 6.2, $E = \{TS_1, TS_2, TS_3, TS_4, TS_5\}$.

Table 6.1: Y_x and Z_x at each PTTM x in Figure 6.2

x	Y_x	Z_x
1	$\{TS_1\}$	$\varnothing$
2	$\{TS_1\}$	$\varnothing$
3	$\{TS_1\}$	$\varnothing$
4	$\{TS_1, TS_5\}$	$\varnothing$
5	$\{TS_3, TS_5\}$	$\{TS_1\}$
6	$\{TS_5\}$	$\{TS_1, TS_3\}$
7	$\{TS_3, TS_4\}$	$\{TS_1, TS_5\}$
8	$\{TS_4\}$	$\{TS_1, TS_3, TS_5\}$
9	$\{TS_2, TS_4\}$	$\{TS_1, TS_3, TS_5\}$
10	$\{TS_1, TS_2\}$	$\{TS_3, TS_4, TS_5\}$
12	$\{TS_4, TS_5\}$	$\{TS_1, TS_2, TS_3\}$
13	$\{TS_2\}$	$\{TS_1, TS_3, TS_4, TS_5\}$

We have assumed that the failure probabilities of individual tests are independent, and that the success probabilities of individual tests are independent. Thus, $p[A_x]$, namely the probability of the test process being terminated at a PTTM x, is given by

$$\begin{aligned} p[A_x] &= p\left[\bigcup_{\forall y \in Y_x} F_x(y)\right] \times \prod_{\forall z \in Z_x} p[P_x(z)] \\ &= \left(1 - \prod_{\forall y \in Y_x} (1 - p[F_x(y)])\right) \times \prod_{\forall z \in Z_x} p[P_x(z)] \end{aligned} \tag{6.5}$$

and $p[T]$, namely the probability of the test process being passed on completion without detecting any faults, is given by

$$p[T] = p\left[\bigcap_{\forall e \in E} P(e)\right] = \prod_{i=1}^{n} (1 - CDP_i) \tag{6.6}$$

Substitute $p[A_x]$ and $p[T]$ in Equation (6.2) with Equations (6.5) and (6.6), the ETAT is given by

$$ETAT = \sum_{\forall x \in X} \left(t_x \times \left(1 - \prod_{\forall y \in Y_x} (1 - p[F_x(y)]) \right) \times \prod_{\forall z \in Z_x} p[P_x(z)] \right) + L \times \prod_{i=1}^{n} (1 - CDP_i) \tag{6.7}$$

where x is a PTTM, X is the set of all PTTMs, t_x is the TAT by the moment x, L is the TAT by the completion moment, Y_x is the set of all individual tests that are stopped at PTTM x, Z_x is the set of all individual tests that are not able to be stopped at PTTM x, $p[F_x(y)]$ is the probability of the individual test y detecting at least one fault at PTTM x, $p[P_x(z)]$ is the probability of individual test z detecting no faults until the latest PTTM before x when z was stopped for a check, and CDP_i is the defect probability of core C_i.

In this thesis, we define the incremental fault coverage (IFC) of a DTP/PTS as the percentage of the faults that are only detected by this DTP/PTS and have not been detected by any previously applied test patterns from the same test set.

Let y be individual test which detects faults at PTTM x, let v be the DTP/PTS which belongs to y and is finished exactly at PTTM x, and let $IFC(v)$ be the incremental fault coverage of v. Then, $p[F_x(y)]$ is given by

$$p[F_x(y)] = IFC(v) \times CDP_i \tag{6.8}$$

Let z be the individual test that is not able to be stopped at PTTM x, let CDP_i be the defect probability of core C_i which test z is applied to, let w ($0 < w < x$) be the latest PTTM when test z was checked for test effects, let m ($0 \le m \le d_i + r_i$) be the number of test patterns

(deterministic or pseudorandom) that had been applied by PTTM *w*, and let v_j be the *j*-th test pattern of test *z*. Then, $p[P_x(z)]$ is given by

$$p\left[P_x(z)\right] = 1 - CDP_i \times \sum_{j=1}^{m} IFC\left(v_j\right) \tag{6.9}$$

The deduction of Equations (6.8) and (6.9) are presented in Appendix A. Thus, the ETAT has been completely formulated.

In this chapter, our objective is to minimize the ETAT. We propose a heuristic algorithm that uses the ETAT as the cost function and generates test schedules with the minimized ETATs.

6.2 Test Scheduling Approach

The heuristic algorithm for test scheduling is a defect-probability driven scheduling approach which generate test schedules with minimized ETATs. As demonstrated earlier, in the context of hybrid BIST and AOFF test approach, it is essential to schedule deterministic test sequences efficiently, as they have high fault-coverage and can be terminated at every test application cycle if faults are detected.

The incremental fault coverage of test patterns, the failed sets and the passed sets vary with different schedule of the DTSs. Therefore, the failing probabilities, the passing probabilities, and the ETAT also vary with different test schedules.

It is natural to schedule the DTSs for the cores with higher defect probabilities earlier. However, such a solution does not necessarily lead to the minimal ETAT. In addition to the defect probabilities of cores, other factors such as the efficiency of test patterns and the length of individual test sequences have to be taken into account.

The proposed heuristic algorithm iteratively constructs two sets of DTSs, namely a scheduled set *S* and an unscheduled set *U*. The scheduled set *S* is an ordered set and it is supposed to include all the DTSs when the algorithm is terminated. The DTSs in *S* are associated with a particular order *O* according to which the DTSs should be

considered for scheduling so that the ETAT of the generated test schedule is the minimum. The unscheduled set *U* is a complement set of *S*, related to the complete set of all DTSs. This means that *U* always includes the unscheduled DTSs during any iteration step.

S is initialized with an empty set, while *U* is initialized with the complete set of all DTSs. In each iteration step, all DTSs in *U* are considered as candidates while only one of them is selected and inserted into *S*, at a position between the already scheduled DTSs. Note that the original order of the scheduled DTSs remains unchanged.

Suppose that in one iteration step, *S* consists of m $(0 \leq m < n)$ scheduled DTSs. The objective in this iteration step is to select one DTS from *U* and insert it into *S*. Since there are $(n - m)$ DTSs in *U* for selection and (m + 1) alternative positions in *S* for insertion, there are in total $(n - m) \times (m + 1)$ different solutions to explore.

How to explore and decide on the alternative solutions is illustrated through an example depicted in Figure 6.4. In this example, there are five hybrid test sets ($n = 5$) and two of them have been temporarily scheduled in previous iteration steps ($m = 2$). The depicted partial test schedule shows that $S = [DTS_1, DTS_4]$ and $U = \{DTS_2, DTS_3, DTS_5\}$. There are three different positions for a candidate to be inserted in *S*, namely $INSPOS_1$, $INSPOS_2$, and $INSPOS_3$, pointed by the three short arrows. The heuristic algorithm explores all the nine alternative solutions each of which is identified by the pair $(DTS_i, INSPOS_j)$ which means that DTS_i is selected from U and inserted into S at the position $INSPOS_j$. Thereafter, all the DTSs in *S* are scheduled sequentially according the fixed order, and their corresponding PTSs are scheduled to the earliest available times. If the TAT of a PTS is longer than the reserved period before the start time of the scheduled DTS for the same core, this PTS has to be split into two partitions such that the TAT of the first partition fits the reserved period and the second partition is scheduled right after the DTS is finished. For each explored partial test schedule, the expected partial test application time (EPTAT) is calculated. When all solutions have been explored, the solution with the minimal EPTAT is

selected. Figure 6.5 shows a partial test schedule where solution (DTS_3, $INSPOS_2$) is selected as the best solution. Thus, the updated S is [DTS_1, DTS_3, DTS_4] and the updated U is {DTS_2, DTS_5}. This example also shows the range for calculating the EPTAT of a partial test schedule.

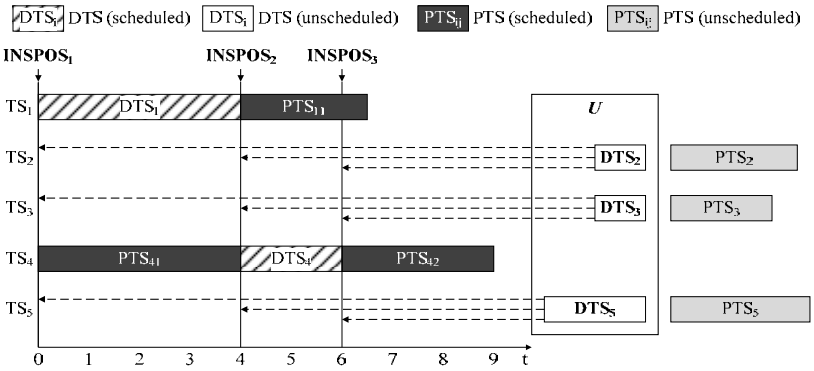


Figure 6.4: Alternative solutions

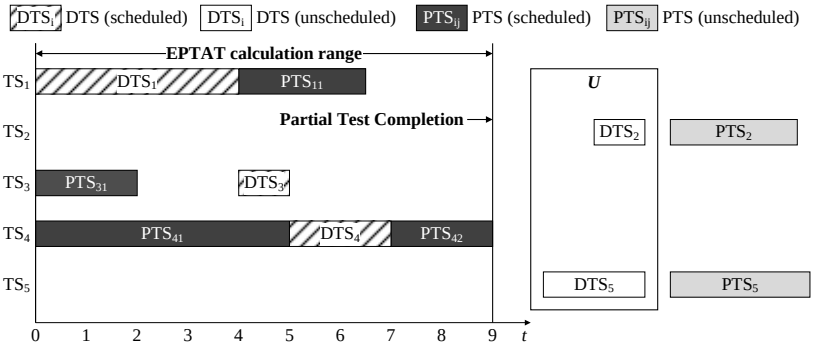


Figure 6.5: Partial test schedule for the best solution

The pseudo-code of the heuristic algorithm is given in Figure 6.6. Line 1 initializes S with an empty set and line 2 initializes U with the complete test set. The outer loop (lines 3 through 19) moves one unscheduled DTS from U and inserts it into S (lines 17 and 18). The DTS to be moved from U is decided within the middle loop (lines 6

through 15) which explores all alternative solutions. For each candidate in U (line 6), each possible position that a candidate in U can be inserted into S is explored within the inner loop (lines 7 through 15). For each alternative solution (line 7), the partial test schedule is generated (line 8) and the EPTAT of the generated partial test schedule is calculated (line 9). Thereafter, the current EPTAT is compared to the minimal EPTAT obtained so far (line 10) and the best solution is updated if the current EPTAT is smaller (lines 11 through 14). The algorithm returns the generated test schedule with the minimal ETAT (line 20), when all the DTSs in U have been moved into S. The computational time complexity of the proposed heuristic algorithm is $O(kn^4)$, where n is the number of cores and k is the average number of deterministic test patterns generated for a core.

Algorithm 6.1: Heuristic algorithm for test scheduling

```
01:  S := ∅;
02:  U := {DTS_1, DTS_2, ... , DTS_n};
03:  while (U ≠ ∅) loop    /* outer loop */
04:      Reset(EPTAT_min);
05:      IPS := GetInsPosSet(S);
06:      for (∀ DTS ∈ U) loop    /* middle loop */
07:          for (∀ InsPos ∈ IPS) loop    /* inner loop */
08:              PartSched_cur := GenPartSched(S, DTS, InsPos);
09:              EPTAT_cur := CalcETAT(PartSched_cur);
10:              if (EPTAT_cur < EPTAT_min) then
11:                  EPTAT_min := EPTAT_cur;
12:                  DTS_sel := DTS;
13:                  InsPos_sel := InsPos;
14:              end if
15:          end for
16:      end for
17:      Insert(S, DTS_sel, InsPos_sel);
18:      Remove(U, DTS_sel);
19:  end while
20:  Return( GenFullSched(S) );
```

Figure 6.6: Pseudo-code of the heuristic algorithm for test scheduling

6.3 Experimental Results

The cores of the SoC designs for our experiments are selected from the ISCAS'85 benchmark circuits. For each SoC design, five different hybrid test sets are generated. Various hybrid test sets have different numbers of DTPs and PTPs. The defect probabilities of individual cores are randomly generated such that the system defect probability is 0.6 (meaning that the yield is 40%). The experimental results are listed in Table 6.2. Each value in the table is the average value of five different experiments for each SoC design.

Table 6.2: Comparison of different scheduling algorithms

# of cores	Random Scheduling		Our Heuristic Algorithm		Simulated Annealing		Exhaustive Search	
	ETAT	CPU Time(s)	ETAT	CPU Time(s)	ETAT	CPU Time(s)	ETAT	CPU Time(s)
5	248.97	1.1	228.85	0.6	228.70	1144.2	228.70	1.2
7	261.38	64.4	232.04	1.4	231.51	1278.5	231.51	80.0
10	366.39	311.8	312.13	6.6	311.68	3727.6	311.68	112592.6
12	415.89	346.8	353.02	12.2	352.10	4266.8	n/a	n/a
15	427.34	371.6	383.40	25.2	381.46	5109.2	n/a	n/a
17	544.37	466.6	494.57	43.6	493.93	6323.8	n/a	n/a
20	566.13	555.4	517.02	85.4	516.89	7504.4	n/a	n/a
30	782.88	822.4	738.74	380.4	736.51	11642.4	n/a	n/a
50	1369.54	1378.0	1326.40	3185.0	1324.44	21308.8	n/a	n/a

In order to evaluate the efficiency of the proposed heuristic algorithm, we compare it with a random scheduling algorithm. The ETATs of the generated test schedules by using the random scheduling and our heuristic algorithm are listed in columns 2 and 4, respectively. It is shown that the ETATs of the test schedules

generated by our heuristic algorithm are 5% to 15% shorter than those produced by the random scheduling algorithm.

In order to evaluate the accuracy of the proposed heuristic algorithm, we compare it with two other scheduling algorithms which are based on a simulated-annealing strategy and an exhaustive search, respectively. The ETATs of the generated test schedules by using the simulated-annealing algorithm and the exhaustive search are listed in columns 6 and 8, respectively. The CPU times of the four approaches are listed in columns 3, 5, 7, and 9. The experimental results show that the ETATs from the proposed heuristic algorithm are very close to those from the algorithms based on simulated-annealing strategy and exhaustive search. Moreover, the execution time of our heuristic algorithm is substantially shorter. These ETATs and CPU times are also plotted in Figure 6.7 and Figure 6.8, respectively.

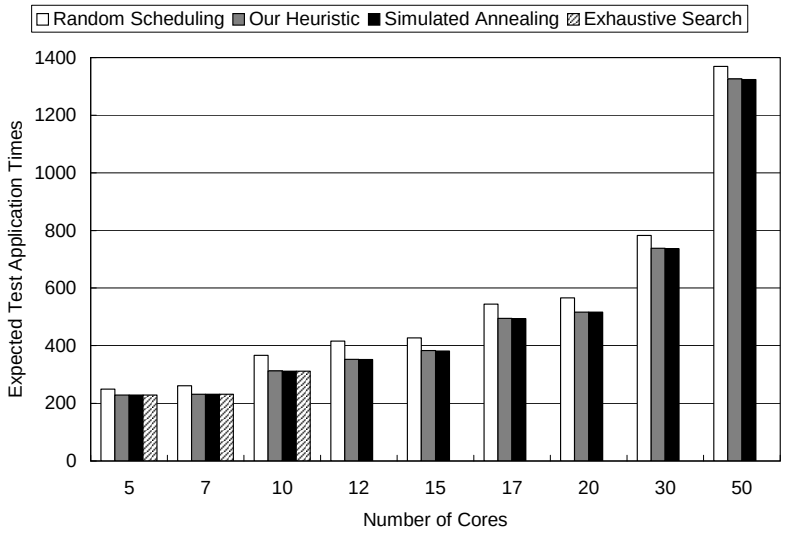


Figure 6.7: ETATs with different approaches

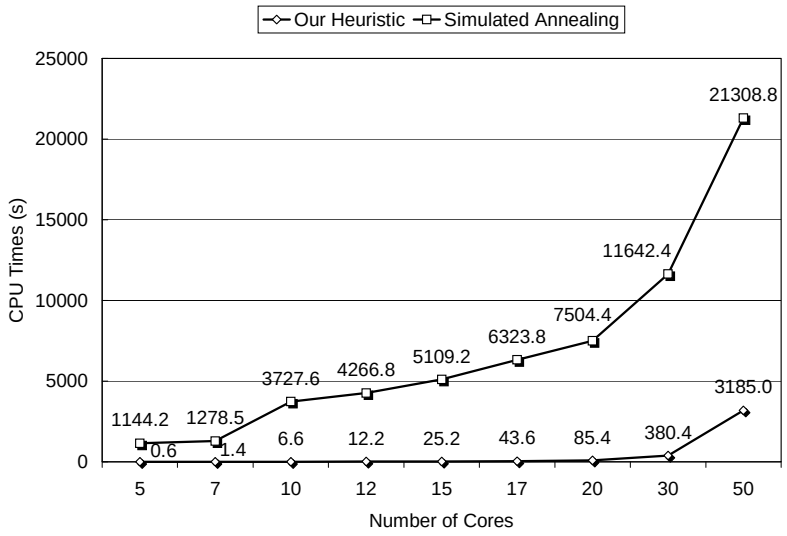


Figure 6.8: Execution times of different scheduling approaches

6.4 Summary

This chapter presents a defect-probability driven test scheduling technique for hybrid BIST using the AOFF test approach. In this technique, the defect probabilities of individual cores are utilized. We propose a method to compute the ETAT which reflects the test application time of volume production tests. We also propose a heuristic algorithm to generate efficient test schedules with minimal ETAT. Experimental results have shown the efficiency of the proposed technique.

Chapter 7
Power Constrained Defect-Probability Driven Test Scheduling

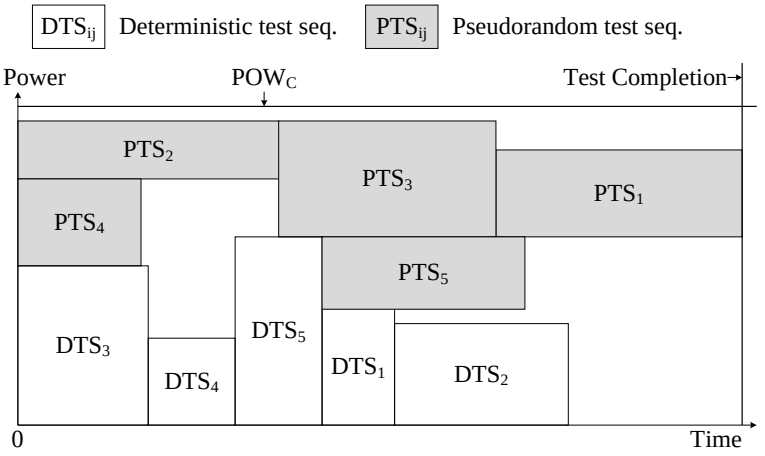
In Chapter 6, we develop a defect-probability driven test scheduling approach which minimizes the ETAT for volume production tests using the AOFF test approach. The proposed technique assumes the BISTs for all cores can be applied concurrently. However, testing large number of core in parallel can result in power and thermal related problems. In this chapter, we present a power constrained defect-probability driven test scheduling technique. In order to improve the efficiency of test schedules, we employ the test set partitioning and test pattern reordering techniques. We develop a heuristic algorithm to find efficient test set partitioning scheme and further to minimize the ETAT of generated test schedules.

7.1 Motivational Example

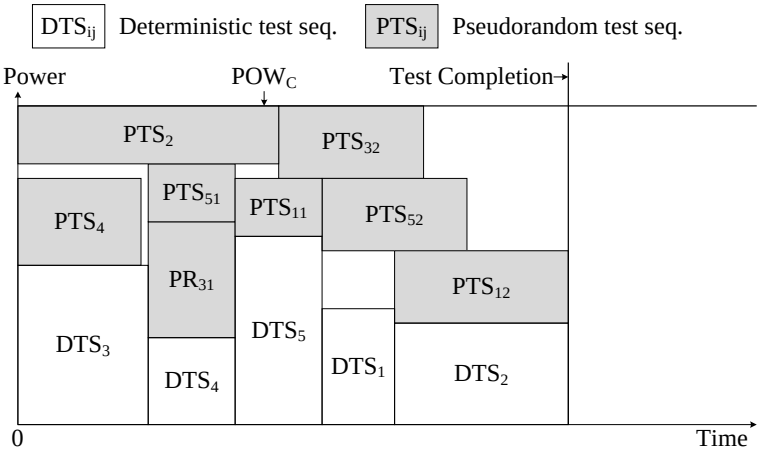
We assume that the power consumption in the circuit by applying a test pattern is proportional to the total number of transitions between this test pattern and the preceding test pattern, occurring at all the primary inputs, primary outputs, and internal nodes. The peak-power consumption by applying a test sequence is defined as the maximum power consumed by applying each of the test patterns belonging to the test sequence (see Figure 7.4).

Figure 7.1 shows an example of a power-constrained test schedule for five DTSs and five PTSs, illustrated with white and grey rectangles, respectively. Each test sequence is represented as a rectangle with its height and width corresponding to the peak-power consumption and the time duration of the test sequence, respectively. The area size of a rectangle is equal to the peak-power consumption multiplied by the time duration. The constraint on the peak-power consumption is denoted with POW_C. It should be noted that test sequences belonging to the same core, such as DTS_1 and PTS_1, cannot be scheduled in parallel due to the test conflict.

Comparing the size of the effective scheduled area occupied by all test sequences to the size of the overall schedulable area confined by the line of peak-power constraint and the line of test completion time, one can find out that the efficiency of the test schedule in Figure 7.1(a) is low since a large area is wasted. One solution to improve the efficiency of the test schedule is to employ test set partitioning TSP to decrease the sizes of test sequences. As shown in Figure 7.1(b), PTS_1, is partitioned into PTS_{11} and PTS_{12}, PTS_3 is partitioned into PTS_{31} and PTS_{32}, and PTS_5 is partitioned into PTS_{51} and PTS_{52}. The partitioned test sequences have a shorter time duration and/or a smaller peak-power consumption than the non-partitioned ones, and therefore can be scheduled at those time moments which are not possible for the non-partitioned test sequences. From this example, it can be observed that using TSP can substantially reduce the TAT.



(a) A test schedule without TSP



(b) A test schedule with TSP

Figure 7.1: Power-constrained test schedule without/with TSP

7.2 Problem Formulation

In this chapter, we use the same definitions given in Section 6.1.1 and the hybrid BIST architecture depicted in Figure 2.7. We assume that a deterministic test set is partitioned into a_i ($0 \le a_i \le d_i$, $1 \le i \le n$) DTSs, and a pseudo-random test set is partitioned into b_i ($0 \le b_i \le r_i$, $1 \le i \le n$) PTSs, where $a_i + b_i > 0$. Figure 7.2 depicts the PTTMs in a power-constrained test schedule, where the dotted lines indicate the finish time moments of DTPs, and the dashed lines indicate the finish time moments of PTSs. Overlapped time moments are treated as identical PTTMs.

In order to minimize the ETAT, we use TSP in test scheduling. The power constrained test time minimization problem is similar to the classical 2D RP problem and is formulated as follows. Given the power constraint, the pseudorandom test sets and deterministic test sets, minimize the ETAT of generated test schedule for all partitioned PTS and DTS. We develop a heuristic approach to explore alternative TSP schemes and generate efficient test schedules.

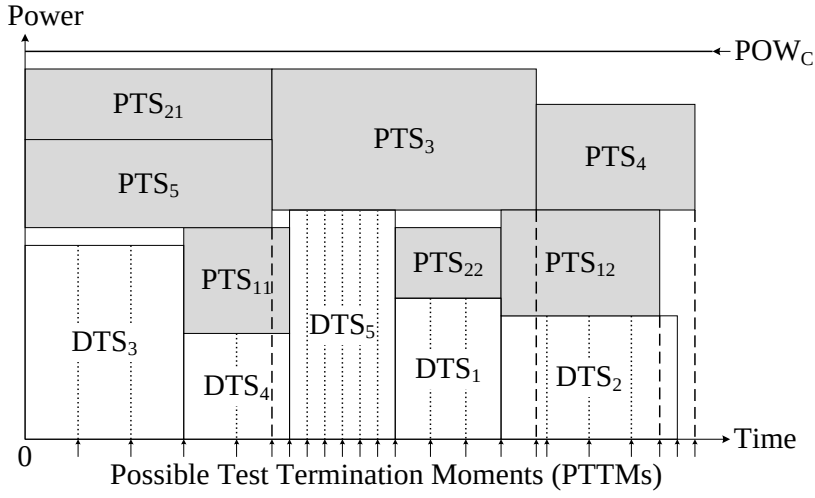


Figure 7.2: PTTMs in a power-constrained test schedule

7.3 Test Scheduling Techniques

7.3.1 Test Set Partitioning

The size of a test sequences has a large impact on the test schedule length. Dividing test sequences into smaller partitions with shorter time duration and lower individual peak-power consumptions leads to more efficient test schedule. This is because the partitioned test sequences have smaller area sizes and can be packed more tightly into the power constrained 2D plane. Figure 7.3(a) shows a non-partitioned deterministic test set for core C_i. Figure 7.3(b) shows three test sequences, DTS_{i1}, DTS_{i2}, and DTS_{i3}, partitioned from the original test set depicted in Figure 7.3(a). In Figure 7.3(b), the individual peak-power consumptions of the first two partitions, DTS_{i1} and DTS_{i2}, are lower than that of the non-partitioned test sequence in Figure 7.3(a). The grey rectangles with dashed boarder lines illustrate the reduced area sizes due to partitioning.

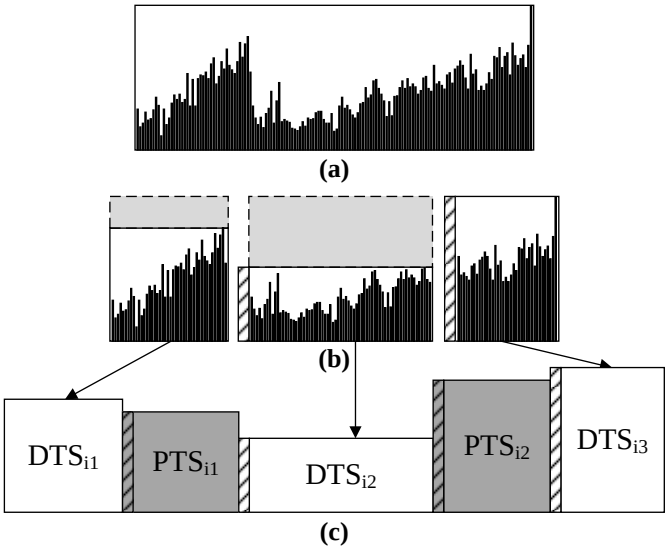
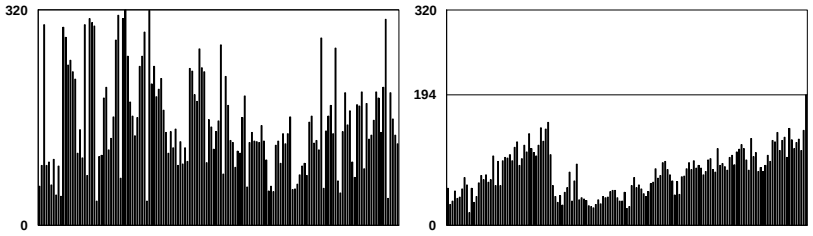


Figure 7.3: Test set partitioning and time overhead

Although test set partitioning can lead to smaller partitions, it introduces time overheads for the partitioned test sequences in scan-based testing. This phenomenon occurs when DTSs and PTSs belonging to the same core are interleaved, as in the example depicted in Figure 7.3. There, the three partitioned deterministic test sequences (DTS_{i1}, DTS_{i2}, and DTS_{i3}) are interleaved with two partitioned pseudorandom test sequences (PTS_{i1} and PTS_{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 PTS_{i1}, DTS_{i2}, PTS_{i2}, and DTS_{i3}. The reason for the time overheads is explained in details in Section 3.1.

7.3.2 Test Pattern Reordering

Reordering test patterns can reduce power consumption and make the power profile of a test sequence relatively smoother and easier to be manipulated in test scheduling [Rosinger, et al. 2002]. Thus, for all deterministic tests, we use test pattern reordering (TSR) as a pre-processing step for test set partitioning. In Figure 7.4(a), the original power profile of a DTS is depicted. As a comparison, the power profile after reordering the test patterns is shown in Figure 7.4(b). It is shown that after reordering the test patterns, the power profile is much smoother and the peak-power consumption is reduced by 39%.



(a) Power profile before TSR **(b) Power profile after TSR**

Figure 7.4: Motivational example of test set reordering

7.3.3 Heuristic Algorithm for Test Set Partitioning

We propose a heuristic algorithm for test set partitioning of deterministic test sets. The algorithm starts with the original test set. In each iteration step, one of the existing partitions is divided into two test sequences. The algorithm stops when partitions cannot be further divided, i.e. every partitioned test sequence consist of 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 explored solutions.

In each 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. Using an exhaustive search among all possible solutions within an iteration step, we obtain the local optimal partitioning scheme which has the lowest cost and add one more partition. Among all the local optimal partitioning schemes with different number of partitions, the one having the minimum cost is acquired and accepted as the best solution. Figure 7.5 illustrates how the sum of the area sizes of all partitions distributes with the 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 7.5, a test set with 149 test patterns should be divided into 21 partitions such that the sum of their area sizes is minimized.

When a PTS 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 expected signature, and an extra time slot is needed to analyze the additional signature. In this chapter, we assumed that there exists sufficient memory in a tester to store the signature. We ignore the extra time slots for analyzing the additional signatures, since it is very short, compared to the time duration of the PTS.

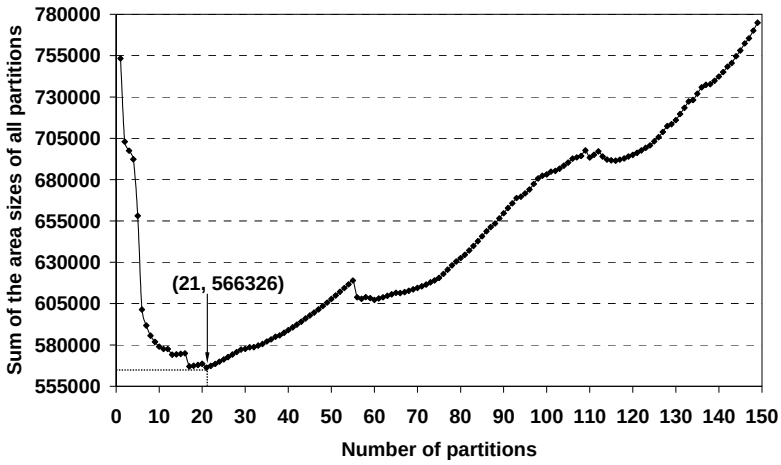


Figure 7.5: Sum of area sizes w.r.t. number of partitions

7.3.4 Heuristic Algorithm for Test Scheduling

Before the heuristic algorithm for test scheduling is presented, some basic principles for test set partitioning and test scheduling are summarized as follows.

(1) Test sequences belonging to the same core cannot be scheduled in parallel.

(2) DTSs are scheduled sequentially since a single test bus is used, while PTSs are scheduled in parallel subject to the peak-power constraint.

(3) The scheduling of DTSs is performed before the scheduling of PTSs, meaning that DTSs have higher scheduling priorities than PTSs. This is because of the assumptions (described in Section 6.1.1) that deterministic tests can be stopped after every test pattern, while pseudorandom tests can only be terminated at the end of the test sequences, when the signatures are available. Moreover, DTPs are usually more efficient in detecting faults than PTPs.

(4) PTSs are first sorted in a decreasing order according to some parameters such as the defect probability of a core, the peak-power consumption, and the time duration of a test sequence. Thereafter, PTSs are scheduled to the earliest available time moment. DTSs, however, are scheduled according to the order obtained by a defect-probability driven heuristic algorithm.

The TSP is integrated into the test scheduling approach in the following way. By using the approach proposed in Section 7.3.3, deterministic test sets are partitioned statically, meaning that they are partitioned before they are scheduled. Pseudorandom test sets, on the other hand, are partitioned during the test scheduling. When it is not possible to schedule a PTS to the earliest time moment due to its large area size, the test sequence is divided into two partitions such that the first one can be scheduled as expected, and the scheduling of the second one is performed later.

Based on the basic principles described above, a heuristic algorithm is developed to find an efficient test schedule for all test sequences in an iterative way. One iteration step of the heuristic algorithm is illustrated through an example in Figure 7.6. Suppose that we have five DTSs (DTS_1, DTS_{21}, DTS_{22}, DTS_{31}, and DTS_{32}), and three PTSs (PTS_1, PTS_2, and PTS_3). Two DTSs (DTS_{31} and DTS_1) have already been scheduled. In this iteration step, we have to decide which one out of the three unscheduled DTSs (DTS_{21}, DTS_{22}, and DTS_{32}) should be scheduled to which time moment among *A*, *B*, and *C*, as illustrated in Figure 7.6. After a DTS is scheduled to a time moment, the three PTSs (PTS_1, PTS_2, and PTS_3) are scheduled to the remaining space. Test set partitioning may be needed at this step. The expected partial test application time is then calculated within the range of the scheduled DTSs (see Figure 7.7). When all the possible nine solutions in the current iteration step have been explored, the solution with the smallest EPTAT is accepted and the three scheduled DTSs are taken as a base for the next iteration step. The heuristic algorithm stops when no more unscheduled DTSs are left, and then the final test schedule is obtained. Note that when a test sequence is scheduled, the

order of the already scheduled test sequences should remain unchanged.

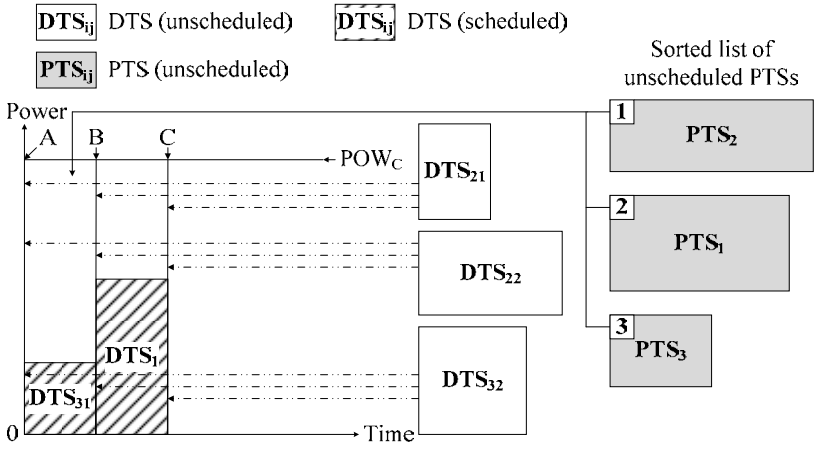


Figure 7.6: One iteration step of the heuristic algorithm

Figure 7.7 shows a solution in which DTS_{22} is scheduled to time moment *B*. During the scheduling of PTSs, PTS_2 is partitioned into two test sequences (PTS_{21} and PTS_{22}). The EPTAT calculation range is from the beginning of DTS_{31} to the end of DTS_1. The gap between PTS_3 and PTS_{22} is due to the fact that DTS_{22} and PTS_{22} cannot be scheduled concurrently due to the test conflict.

A formal description of the heuristic algorithm for test scheduling is presented as follows. Suppose that we have N DTSs altogether, and m $(0 \leq m < N)$ of them have already been scheduled in a certain iteration step. We need to schedule one more DTS selected from the set of $(N - m)$ unscheduled DTSs to an appropriate time moment, without changing the order of the scheduled test sequences. When a selected DTS has been scheduled to a time moment, all the PTSs are then scheduled into the remaining space, with dynamic test set partitioning, if needed. The EPTAT of this solution is then calculated within the time range of the $(m + 1)$ scheduled DTSs. When all the

$(N - m) \times (m + 1)$ possible solutions have been explored, the solution with the minimum EPTAT value is accepted. The new list of scheduled DTSs is then used as a base for the next iteration step. Repeating this procedure from the initial state (where $m = 0$) until all the DTSs and PTSs are scheduled (when $m = N$), we get the final optimized schedule.

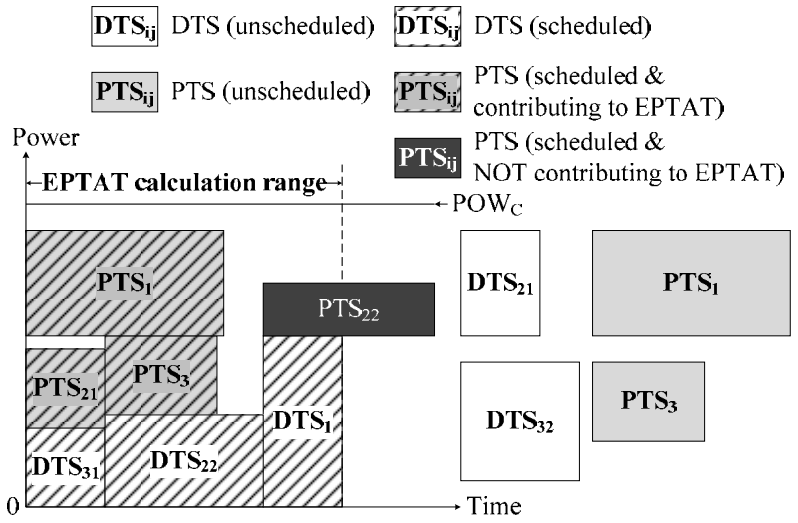


Figure 7.7: Illustration of one solution in the iteration step

The pseudo-code of the heuristic algorithm is depicted in Figure 7.8. The algorithm has three major nested loops. The outer loop (lines 1 through 19) increments the number of scheduled DTSs, the middle loop (lines 4 through 17) selects every unscheduled DTS, and the inner loop (lines 5 through 16) explores every possible time moment for scheduling. Inside the inner loop, after the selected DTS is scheduled (line 6), pseudorandom test sets are partitioned, if needed, and then scheduled (lines 7 through 10). The EPTAT of the present schedule is calculated (line 11) and compared to the minimum EPTAT for a decision (lines 12 through 15). The final test schedule is returned in the end (line 20).

Algorithm 7.1: Heuristic algorithm for test scheduling

```
01: for (#_Sched_DTS := 0 to N-1) loop    /* outer loop */
02:     Reset(EPTAT_min);
03:     m := #_Sched_DTS;
04:     for (∀UnschedDTS_ij) loop    /* middle loop */
05:         for (∀PTTM T_x) loop    /* inner loop */
06:             Schedule(UnschedDTS_ij, T_x);
07:             for (∀ pseudorandom test set PTS_k ) loop
08:                 Partition(PTS_k) if needed;
09:                 Schedule(PTS_k);
10:             end for;
11:             EPTAT_cur := CalcEPTAT();
12:             if (EPTAT_cur < EPTAT_min) then
13:                 EPTAT_min := EPTAT_cur;
14:                 Solution_best := Solution_cur;
15:             end if;
16:         end for;
17:     end for;
18:     Apply(Solution_best);
19: end for;
20: Return(TestSchedule_final);
```

Figure 7.8: Pseudo-code of the heuristic algorithm for test scheduling

7.4 Experimental Results

ISCAS'89 benchmark circuits are used as cores in the SoC designs for our experiments. All cores are redesigned to insert one single scan chain, and the STUMPS architecture is used for BIST.

In the first group of experiments, the proposed test set partitioning and test scheduling technique is employed. We perform experiments

for 5 groups of SoC designs. Each group has 5 different SoC designs which have the same number of cores of different types, and the cores are assigned with different defect probabilities. The numbers of cores in the SoC designs are 5, 10, 20, 30, and 50 for each group, respectively. For each SoC design we impose three different peak-power constraints. The experimental results presented in Table 7.1 are 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 are randomly generated, such that the system defect probability is 0.6, i.e. 40% system yield.

In order to evaluate the efficiency of our heuristic algorithm, a classical bottom-left-decreasing (BLD) scheduling algorithm [Lesh, et al. 2005] is taken for comparison. The BLD algorithm sorts DTSs and PTSs decreasingly according to 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 7.1, by employing our heuristic algorithm, the ETAT is reduced about 20% to 29% compared to the BLD scheduling algorithm, with an acceptable increase in execution time. On the other hand, in order to evaluate the accuracy of our heuristic algorithm to find a near-optimal test schedule, we compared our heuristic algorithm with a simulated annealing (SA) algorithm. For small designs with 5 and 10 cores, the SA algorithm reaches the imposed termination condition in an acceptable time and is supposed to return a solution as close to the optimal solution as possible. For large SoC designs with 20, 30, and 50 cores, the SA algorithm takes 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 algorithm. From Table 7.1, it is shown that in small designs, the SA algorithm works just slightly better than our heuristic algorithm (2% to 3% lower ETAT), but has up to two orders of magnitude longer execution time than our heuristic algorithm. For the large SoC designs, our heuristic algorithm found better solutions with 4% to 7% lower

ETAT than what the SA algorithm produces in the same amount of execution time.

In the second group of experiments where the same SoC designs are used, we evaluate the effect of test set partitioning. As a comparison, we used a defect-probability driven test scheduling heuristic algorithm which does not allow test set partitioning. For the sake of fairness, both the partitioned and non-partitioned heuristic algorithm use test pattern reordering to reduce peak-power consumption. The experimental results are listed in Table 7.2. As shown in the table, using test set partitioning can reduce the ETAT by 16% to 30%. This experimental result is also plotted in Figure 7.9.

Table 7.1: Comparison of different scheduling approaches using TSP

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

Table 7.2: Comparison of scheduling approaches using/not using TSP

# of Cores	Without TSP		With TSP	
	ETAT	CPU Time (s)	ETAT	CPU Time (s)
5	8269	0.09	6247	2.5
10	11357	0.86	7983	26.9
20	18016	14.2	14239	293.9
30	26710	68.6	21117	493.4
50	44713	589.1	37463	4372.9

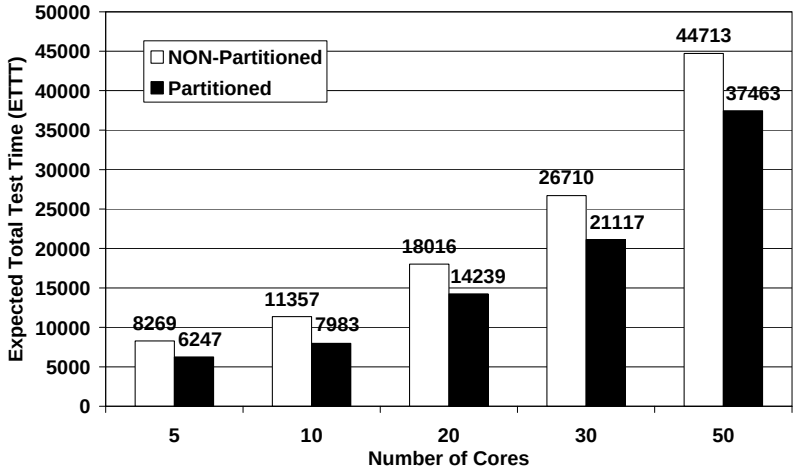


Figure 7.9: Comparison of scheduling approaches using/not using TSP

7.5 Summary

This chapter presents a power constrained defect-probability driven test scheduling approach for volume production test using the AOFF test approach. Defect probabilities of individual cores are utilized to guide test scheduling which employs test set reordering and test set partitioning techniques. Heuristic algorithms for test set partitioning and test scheduling are proposed to generate efficient test schedules. Experimental results have shown that the proposed method is efficient to minimize the ETAT.

Chapter 8
Conclusions and Future Work

This chapter concludes the thesis and discusses possible directions for future work.

8.1 Conclusions

The aim of the work presented in this thesis is to reduce the cost of electronic testing. The major contribution of this thesis is that it proposes a set of test scheduling techniques to minimize the test application time with different considerations, including temperature, power consumption, and defect probability.

The first proposed technique is temperature aware test scheduling based on test set partitioning and interleaving. This technique aims to generate efficient test schedules and avoid high temperature during test. The presented test scheduling technique generates the shortest test schedule for core-based SoCs such that the imposed temperature limit and test-bus width limit are satisfied. The test set partitioning technique aims to avoid overheating the cores under test by dividing a

test set into shorter test sequences and inserting cooling periods between the partitioned test sequences. The test set interleaving technique aims to improve the efficiency of the test schedules by utilizing the cooling periods of one core to test other cores. Based on the test set partitioning and interleaving technique, we propose two different solutions to the test time minimization problem with the temperature and test-bus width constraints. The first solution targets the SoCs with negligible lateral thermal influence. A CLP-based approach and a heuristic approach are proposed for test scheduling. The second solution targets the SoCs with significant lateral thermal influence, and a thermal-simulation driven approach is proposed for test scheduling.

The second proposed technique is multi-temperature testing which aims to test SoCs at different temperature intervals so that the temperature-dependent defects can be efficiently sensitized. A thermal-simulation driven test scheduling approach is proposed to minimize the test application time such that a test is applied to a core under test only when the temperature of the core is within a given interval and the test-bus width constraint is satisfied.

The third proposed technique is defect-probability driven test scheduling based on the AOFF test approach. This technique aims to minimize the expected test application time for volume production tests. In order to avoid the power and temperature related problems during test, we propose a power constrained test scheduling approach for hybrid BIST.

Extensive experiments have been performed and the experimental results have shown the efficiency of the proposed techniques.

8.2 Future Work

Recently, a three-dimensional (3D) integration technique has emerged in IC designs. This technique stacks the silicon die in the vertical dimension and the dies on different layers are connected by the

through-silicon vias (TSVs). The advantages of 3D-stacked ICs include reduced number and length of wires, decreased interconnection delay, increased integration density, and improved performance. Moving from the 2D integration technique, the 3D-stacked ICs encounter a greater challenge in thermal-related issues. High temperature occurs in 3D-stacked ICs as the active silicon layer heat each other while no efficient cooling solutions exist to take away the heat. When testing 3D-stacked ICs, the thermal issues have to be addressed since the testing power dissipation is much higher. Temperature aware testing for 3D-stacked ICs is an interesting topic for our future work. A possible research direction is the 3D temperature aware test scheduling which minimizes the test application time while keeps the temperature of the CUTs below a given limit for 3D-stacked SoCs. New techniques based on the test scheduling approaches proposed in this thesis can be developed for 3D temperature aware test scheduling.

Process variation related testing is another possible direction for future work. When the CMOS process moves into deep-nanometer regime, the reliability of ICs becomes a great challenge due to process variation. Traditional temperature aware testing techniques may not be applicable since variation in physical parameters appears between or within silicon dies. Combining the offline test scheduling techniques using thermal simulation and online test scheduling techniques using temperature sensors for temperature aware testing can be an interesting research direction in the future.

List of Figures

List of Tables

List of Abbreviations

2D	Two-Dimensional
2PSA	Two-Phase Scheduling Algorithm
3D	Three-Dimensional
ALU	Arithmetic Logic Unit
AMBA	Advanced Microprocessor Bus Architecture
AOFF	Abort-on-First-Fail
ASIC	Application-Specific Integrated Circuit
ATE	Automatic Test Equipment
ATPG	Automatic Test Pattern Generation
BIST	Built-In Self-Test
BLD	Bottom-Left Decreasing
CDFG	Control/Data-Flow Graph
CDP	Core Defect Probability
CLP	Constraint Logic Programming
CMOS	Complementary Metal-Oxide-Semiconductor
CPU	Central Processing Unit
CUT	Core Under Test
DATS	Direct Access Test Scheme
DFT	Design for Test
DMA	Direct Memory Access

DSP	Digital Signal Processor
DTP	Deterministic Test Pattern
DTS	Deterministic Test Sequence
DUT	Device Under Test
EATM	Earliest Available Time Moment
ELSA	Equal-Length Scheduling Algorithm
EPATA	Expected Partial Test Application Time
ETAT	Expected Test Application Time
FLSA	Flexible-Length Scheduling Algorithm
FPU	Floating-Point Unit
FSM	Finite-State Machine
HP	Heating Pattern
HPF	High-Power Frame
HS	Heating Sequence
IFC	Incremental Fault Coverage
IC	Integrated Circuit
ILP	Integer Linear Programming
IP	Intellectual Property
ISCAS	International Symposium on Circuits and Systems
ITFP	Individual Test Failure Probability
ITSP	Individual Test Success Probability
LFSR	Linear Feedback Shift Register
LPF	Low-Power Frame
MCM	Multi-Chip Module
MILP	Mixed-Integer Linear Programming
MISR	Multi-Input Signature Register
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
OF	Observation Frame

OSCO	Overall Scheduling Consideration Order
PCB	Printed-Circuit Board
PTAT	Partial Test Application Time
PTP	Pseudorandom Test Pattern
PTS	Pseudorandom Test Sequence
PTTM	Possible Test Termination Moment
PWM	Pulse-Width Modulation
RAM	Random-Access Memory
RASBuS	Reuse of Addressable System Bus
RF	Radio Frequency
ROM	Read-Only Memory
RC	Resistance-Capacitance
RP	Rectangle Packing
RT	Register-Transfer
RTL	Register-Transfer Level
SA	Simulated Annealing
SABA	Simulated-Annealing-Based Algorithm
SCO	Scheduling Consideration Order
SCT	Stop-Cooling Temperature
SDP	System Defect Probability
SDSA	Simulation-Driven Scheduling Algorithm/Approach
SFA	Straight-Forward Algorithm/Approach
SISR	Single-Input Signature Register
SoC	System-on-Chip
STUMPS	Self-Testing Using MISR and Parallel SRSG
TAI	Test Access Infrastructure
TAM	Test Access Mechanism
TAT	Test Application Time

TFP	Test Failing Probability
TG	Test Generation
TP	Test Pattern
TPP	Test Passing Probability
TS	Test Set
TSE	Test Schedule Efficiency
TSI	Test Set Interleaving
TSP	Test Set Partitioning
TSPI	Test Set Partitioning and Interleaving
UDL	User Defined Logic
USB	Universal Serial Bus
VLSI	Very-Large-Scale Integration
ZTC	Zero-Temperature Coefficient

Appendix A
Deduction of Equations (6.8) and (6.9) in Section 6.1.3

This appendix explains how Equations (6.8) and (6.9) in Section 6.1.3 are deducted.

Definition 1: test set and test patterns.

Suppose that a test set consists of m test patterns, which can be deterministic test patterns or pseudorandom test patterns. We denote a test set with *TS*, and the j-th test pattern in *TS* with v_j.

$$TS = \{v_j | 1 \le j \le m\} = \{v_1, v_2, \cdots, v_j, \cdots, v_m\} \quad \text{(A.1)}$$

Definition 2: incremental fault coverage of a test pattern.

The incremental fault coverage of a test pattern v, denoted with $IFC(v)$, is the ratio of the faults that can be detected by the test pattern v but cannot be detected by any preceding test patterns in the same test set, to the total number of faults that can be detected by the entire test set. Suppose that a test set *TS* can detect N faults in total, and the j-th test pattern v_j in *TS* can detect n_j faults that cannot be detected by any of the preceding test patterns $\{v_1, v_2, \ldots, v_{j-1}\}$ in *TS*. Let n_j be the number of faults that can be detected by the j-th test pattern v_j in *TS* but cannot not be detected by any preceding test patterns in *TS*, and let

N be the number of faults that can be detected by the test patterns in *TS*. The IFC of v_j is defined as

$$IFC\left(v_j\right)=\frac{n_j}{N} \quad \left(0 \le n_j < N, 1 \le j \le m\right) \tag{A.2}$$

Definition 3: fail and pass a single pattern test.

We define two random events regarding a test by applying a single test pattern: fail a single pattern test and pass a single pattern test. Fail a single pattern test is an event F_j that a test by applying j-th ($1 \le j \le m$) test pattern v_j is failed due to detection of at least one fault. This infers that the entire test process is aborted immediately. Pass a single pattern test is an event $\neg F_j$ that a test by applying the j-th ($1 \le j \le m$) test pattern v_j is passed due to no detection of any faults. This infers that the entire test process continues and the next test pattern is going to be applied. Fail a single pattern test and pass a single pattern test are complement events. Let D be the random event that a core under test is defective. The defect probability of a core is $DP = p[D]$.

Definition 4: conditional probability of fail a currently applied pattern test.

Suppose that the j-th test pattern can detect n_j incremental faults ($1 \le j \le m$). The following equation shows how to calculate the conditional probability that the j-th test pattern v_j detects at least one fault provided that the preceding test patterns in the same test set did not detect any faults while the core is actually defective.

$$\begin{aligned} & p\left[F_j \mid \neg F_{j-1} \cap \neg F_{j-2} \cap \cdots \cap \neg F_1 \cap D\right] \\ & = \frac{n_j}{N - n_{j-1} - n_{j-2} - \cdots - n_1} \\ & i.e.\ \ p\left[F_j \mid \bigcap_{k=1}^{j-1} \neg F_k \cap D\right] = \frac{n_j}{N - \sum_{k=1}^{j-1} n_k} \end{aligned} \tag{A.3}$$

It should be noted that only incremental faults are counted in this probability calculation. This is because those faults which are covered by both v_j and any preceding test patterns have no chance to be detected by v_j in the actual test process. According to the condition given in the formula, those faults should have been detected by the preceding test patterns before they are detected by v_j.

Definition 5: terminate and pass a partial test.

Terminate a partial test after applying the j-th test pattern is an random event A_j that the test is terminated immediately after the j-th test pattern v_j detects at least one fault. Pass a partial test after applying the j-th test pattern is a random event P_j that the partial test is passed after the j-th test pattern v_j is applied without detecting any faults and that the test process continues to apply the next test pattern.

According to the definitions, A_j is equivalent to the intersection of the following three events: (1) the j-th test pattern detects at least one fault; (2) the preceding test patterns did not detect any faults; (3) the core is actually defective. Similarly, P_j is equivalent to the intersection of the following two events: (1) the conjunction of such events that all the j applied test patterns did not detect any faults and the core is actually defective; (2) the core is actually not defective. Thus, A_j and P_j are given by

$$\begin{aligned} &A_j = F_j \cap \neg F_{j-1} \cap \neg F_{j-2} \cap \cdots \cap \neg F_1 \cap D \\ &i.e.\ \ A_j = F_j \cap \bigcap_{k=1}^{j-1} \neg F_k \cap D \end{aligned} \tag{A.4}$$

$$\begin{aligned} &P_j = \left(\neg F_j \cap \neg F_{j-1} \cap \neg F_{j-2} \cap \cdots \cap \neg F_1 \cap D\right) \cup \neg D \\ &i.e.\ \ P_j = \left(\bigcap_{k=1}^{j} \neg F_k \cap D\right) \cup \neg D \end{aligned} \tag{A.5}$$

Definition 6: suppose that a test employs the AOFF test approach and the test can only be terminated when a test pattern has been applied and the test response or signature has been analyzed. Let $p[A_j]$ be the probability of the test being aborted at a certain test pattern and

let $p[P_j]$ be the probability of the test succeeding at a certain test pattern. Then, $p[A_j]$ and $p[P_j]$ are given by the following two equations respectively.

$$p[A_j] = IFC(v_j) \times DP \tag{A.6}$$

$$p[P_j] = 1 - DP \times \sum_{k=1}^{j} IFC(v_k) \tag{A.7}$$

Equations (A.6) and (A.7) can be proved using a mathematical induction. The proof is given as follows.

Step 1: (observations)

$$p[A_1] = p[F_1 \cap D] = p[F_1 \mid D] \times p[D] = \frac{n_1}{N} \times DP$$
$$= IFC(v_1) \times DP$$

$$p[\neg F_1 \cap D] = p[\neg F_1 \mid D] \times p[D] = (1 - p[F_1 \mid D]) \times p[D]$$
$$= \left(1 - \frac{n_1}{N}\right) \times DP$$

$$p[P_1] = p[(\neg F_1 \cap D) \cup \neg D] = p[\neg F_1 \cap D] + p[\neg D]$$
$$= \left(1 - \frac{n_1}{N}\right) \times DP + (1 - DP) = 1 - \frac{n_1}{N} \times DP$$
$$= 1 - IFC(v_1) \times DP$$

Step 2: (observations)

$$p[A_2] = p[F_2 \cap \neg F_1 \cap D] = p[F_2 \mid \neg F_1 \cap D] \times p[\neg F_1 \cap D]$$
$$= \frac{n_2}{N - n_1} \times \left(1 - \frac{n_1}{N}\right) \times DP = \frac{n_2}{N} \times DP = IFC(v_2) \times DP$$

$$p[\neg F_2 \cap \neg F_1 \cap D] = p[\neg F_2 \mid \neg F_1 \cap D] \times p[\neg F_1 \cap D]$$
$$= (1 - p[F_2 \mid \neg F_1 \cap D]) \times p[\neg F_1 \cap D]$$
$$= \left(1 - \frac{n_2}{N - n_1}\right) \times \left(1 - \frac{n_1}{N}\right) \times DP = \left(1 - \frac{n_2}{N} - \frac{n_1}{N}\right) \times DP$$

$$p[P_2] = p[(\neg F_2 \cap \neg F_1 \cap D) \cup \neg D]$$
$$= p[\neg F_2 \cap \neg F_1 \cap D] + p[\neg D]$$
$$= \left(1 - \frac{n_2}{N} - \frac{n_1}{N}\right) \times DP + (1 - DP) = 1 - \left(\frac{n_2}{N} + \frac{n_1}{N}\right) \times DP$$
$$= 1 - (IFC(v_2) + IFC(v_1)) \times DP$$

Step 3: (observations)

$$p[A_3] = p[F_3 \cap \neg F_2 \cap \neg F_1 \cap D]$$
$$= p[F_3 \mid \neg F_2 \cap \neg F_1 \cap D] \times p[\neg F_2 \cap \neg F_1 \cap D]$$
$$= \frac{n_3}{N - n_2 - n_1} \times \left(1 - \frac{n_2}{N} - \frac{n_1}{N}\right) \times DP = \frac{n_3}{N} \times DP$$
$$= IFC(v_3) \times DP$$

$$p[\neg F_3 \cap \neg F_2 \cap \neg F_1 \cap D]$$
$$= p[\neg F_3 \mid \neg F_2 \cap \neg F_1 \cap D] \times p[\neg F_2 \cap \neg F_1 \cap D]$$
$$= (1 - p[F_3 \mid \neg F_2 \cap \neg F_1 \cap D]) \times p[\neg F_2 \cap \neg F_1 \cap D]$$
$$= \left(1 - \frac{n_3}{N - n_2 - n_1}\right) \times \left(1 - \frac{n_2}{N} - \frac{n_1}{N}\right) \times DP$$
$$= \left(1 - \frac{n_3}{N} - \frac{n_2}{N} - \frac{n_1}{N}\right) \times DP$$

$$p[P_3] = p[(\neg F_3 \cap \neg F_2 \cap \neg F_1 \cap D) \cup \neg D]$$
$$= p[\neg F_3 \cap \neg F_2 \cap \neg F_1 \cap D] + p[\neg D]$$
$$= \left(1 - \frac{n_3}{N} - \frac{n_2}{N} - \frac{n_1}{N}\right) \times DP + (1 - DP)$$
$$= 1 - \left(\frac{n_3}{N} + \frac{n_2}{N} + \frac{n_1}{N}\right) \times DP$$
$$= 1 - (IFC(v_3) + IFC(v_2) + IFC(v_1)) \times DP$$

Step (j – 1): assume that

$$p[A_{j-1}] = \frac{n_{j-1}}{N} \times DP$$

$$p\left[\bigcap_{k=1}^{j-1} \neg F_k \cap D\right] = \left(1 - \sum_{k=1}^{j-1} \frac{n_k}{N}\right) \times DP$$

$$p[P_{j-1}] = 1 - DP \times \sum_{k=1}^{j-1} \frac{n_k}{N}$$

Step j: according to the assumptions given in Step (j – 1), we have

$$p[A_j] = p\left[F_j \cap \bigcap_{k=1}^{j-1} \neg F_k \cap D\right]$$
$$= p\left[F_j \mid \bigcap_{k=1}^{j-1} \neg F_k \cap D\right] \times p\left[\bigcap_{k=1}^{j-1} \neg F_k \cap D\right]$$
$$= \frac{n_j}{N - \sum_{k=1}^{j-1} n_k} \times \left(1 - \sum_{k=1}^{j-1} \frac{n_k}{N}\right) \times DP = \frac{n_j}{N} \times DP = IFC(v_j) \times DP$$

$$p\left[\bigcap_{k=1}^{j}\neg F_k \cap D\right] = p\left[\neg F_j \mid \bigcap_{k=1}^{j-1}\neg F_k \cap D\right] \times p\left[\bigcap_{k=1}^{j-1}\neg F_k \cap D\right]$$

$$= \left(1 - p\left[F_j \mid \bigcap_{k=1}^{j-1}\neg F_k \cap D\right]\right) \times p\left[\bigcap_{k=1}^{j-1}\neg F_k \cap D\right]$$

$$= \left(1 - \frac{n_j}{N - \sum_{k=1}^{j-1} n_k}\right) \times \left(1 - \sum_{k=1}^{j-1}\frac{n_k}{N}\right) \times DP = \left(1 - \sum_{k=1}^{j}\frac{n_k}{N}\right) \times DP$$

$$p[P_j] = p\left[\left(\bigcap_{k=1}^{j}\neg F_k \cap D\right) \cup \neg D\right]$$

$$= p\left[\bigcap_{k=1}^{j}\neg F_k \cap D\right] + p[\neg D] = \left(1 - \sum_{k=1}^{j}\frac{n_k}{N}\right) \times DP + (1 - DP)$$

$$= 1 - DP \times \sum_{k=1}^{j}\frac{n_k}{N} = 1 - DP \times \sum_{k=1}^{j} IFC(v_k)$$

□

References

ABRAMOVICI, M., BREUER, M.A. AND FRIEDMAN, A.D. 1994. Digital Systems Testing and Testable Design. Wiley-IEEE Press.

AERTS, J. AND MARINISSEN, E.J. 1998. Scan Chain Design for Test Time Reduction in Core-Based ICs. In *Proceedings of the 1998 IEEE International Test Conference*, Washington, DC, USA, October 18 - October 23, pp. 448-457.

BAKER, B.S., COFFMAN, E.G. AND RIVEST, R.L. 1980. Orthogonal Packings in Two Dimensions. *SIAM Journal on Computing*, 9(4), pp. 846-855.

BORKAR, S. 1999. Design Challenges of Technology Scaling. *IEEE Micro*, 19(4), pp. 23-29.

BUSHNELL, M.L. AND AGRAWAL, V.D. 2000. Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits. Springer.

CALHOUN, B.H. AND CHANDRAKASAN, A.P. 2006. Ultra-Dynamic Voltage Scaling (UDVS) Using Sub-Threshold Operation and Local Voltage Dithering. *IEEE Journal of Solid-State Circuits*, 41(1), pp. 238-245.

CHAKRABARTY, K. 2000a. Test Scheduling for Core-Based Systems Using Mixed-Integer Linear Programming. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(10), pp. 1163-1174.

CHAKRABARTY, K. 2000b. Design of System-on-a-Chip Test Access Architectures Under Place-and-Route and Power Constraints. In *Proceedings of the 37th Design Automation Conference*, Los Angeles, California, United States, June 5 - June 9, pp. 432-437.

CHOU, R.M., SALUJA, K.K. AND AGRAWAL, V.D. 1997. Scheduling Tests for VLSI Systems Under Power Constraints. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 5(2), pp. 175-185.

DAVIS, B. 1994. The Economics of Automatic Testing. McGraw-Hill.

DELL'AMICO, M., MAFFIOLI, F. AND MARTELLO, S., Eds. 1997. Annotated Bibliographies in Combinatorial Optimization. John Wiley & Sons.

DEVADAS, S., GHOSH, A. AND KEUTZER, K. 1994. Logic Synthesis. McGraw-Hill, Inc.

DYCKHOFF, H. 1990. A Typology of Cutting and Packing Problems. *European Journal of Operational Research*, 44(2), pp. 145-159.

ELLIOTT, J.P. 1999. Understanding Behavioral Synthesis: A Practical Guide to High-Level Design. Kluwer Academic Publishers.

FILANOVSKY, I.M. AND ALLAM, A. 2001. Mutual Compensation of Mobility and Threshold Voltage Temperature Effects with Applications in CMOS Circuits. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 48(7), pp. 876-884.

FLYNN, D. 1997. AMBA: Enabling Reusable On-Chip Designs. *IEEE Micro*, 17(4), pp. 20-27.

GAJSKI, D.D. AND KUHN, R.H. 1983. Guest Editors' Introduction: New VLSI Tools. *IEEE Computer*, 16(12), pp. 11-14.

GERSTENDÖRFER, S. AND WUNDERLICH, H. 2000. Minimized Power Consumption for Scan-Based BIST. *Journal of Electronic Testing: Theory and Applications*, 16(3), pp. 203-212.

GIRARD, P., LANDRAULT, C., PRAVOSSOUDOVITCH, S. AND SEVERAC, D. 1998. Reducing Power Consumption During Test Application by Test Vector Ordering. In *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems*, Monterey, CA, USA, May 31 - June 3, pp. 296-299.

GIRARD, P. 2000. Low Power Testing of VLSI Circuits: Problems and Solutions. In *Proceedings of the 1st IEEE International Symposium on Quality Electronic Design*, San Jose, CA , USA, March 20 - March 22, pp. 173-179.

GOEL, S.K. AND MARINISSEN, E.J. 2003. Control-Aware Test Architecture Design for Modular SOC Testing. In *Proceedings of the 8th IEEE European Test Workshop*, Maastricht, The Netherlands, May 25 - May 28, pp. 57-62.

GUNTHER, S.H., BINNS, F., CARMEAN, D.M. AND HALL, J.C. 2001. Managing the Impact of Increasing Microprocessor Power Consumption. *Intel Technology Journal*, 5(1), pp. 1-9.

HARROD, P. 1999. Testing Reusable IP - A Case Study. In *Proceedings of the 1999 IEEE International Test Conference*, Atlantic City, NJ, USA, September 28 - September 30, pp. 493-498.

HE, Z., JERVAN, G., PENG, Z. AND ELES, P. 2004. Hybrid BIST Test Scheduling Based on Defect Probabilities. In *Proceedings of the 13th IEEE Asian Test Symposium*, Kenting, Taiwan, November 15 - November 17, pp. 230-235.

HE, Z., JERVAN, G., PENG, Z. AND ELES, P. 2005. Power-Constrained Hybrid BIST Test Scheduling in an Abort-on-First-Fail Test Environment. In *Proceedings of the 8th Euromicro Conference on Digital System Design*, Porto, Portugal, August 30 - September 3, pp. 83-86.

HE, Z., PENG, Z. AND ELES, P. 2006a. Power Constrained and Defect-Probability Driven SoC Test Scheduling with Test Set Partitioning. In *Proceedings of the 2006 Design, Automation and Test in Europe Conference*, Munich, Germany, March 6 - March 10, pp. 291-296.

HE, Z., PENG, Z., ELES, P., ROSINGER, P. AND AL-HASHIMI, B.M. 2006b. Thermal-Aware SoC Test Scheduling with Test Set Partitioning and Interleaving. In *Proceedings of the 21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, Arlington, Virginia, USA, October 4 - October 6, pp. 477-485.

HE, Z. 2007. System-on-Chip Test Scheduling with Defect-Probability and Temperature Considerations. Licentiate of Engineering. Thesis No. 1313. Linköping Studies in Science and Technology. Linköping University.

HE, Z., PENG, Z. AND ELES, P. 2007. A Heuristic for Thermal-Safe SoC Test Scheduling. In *Proceedings of the 2007 IEEE International Test Conference*, Santa Clara, California, USA, October 21 - October 26, pp. 1-10.

HE, Z., PENG, Z. AND ELES, P. 2008a. Simulation-Driven Thermal-Safe Test Time Minimization for System-on-Chip. In *Proceedings of the 17th IEEE Asian Test Symposium*, Sapporo, Japan, November 24 - November 27, pp. 283-288.

HE, Z., PENG, Z., ELES, P., ROSINGER, P. AND AL-HASHIMI, B.M. 2008b. Thermal-Aware SoC Test Scheduling with Test Set Partitioning and Interleaving. *Journal of Electronic Testing: Theory and Applications*, 24(1-3), pp. 247-257.

HE, Z., PENG, Z. AND ELES, P. 2009. Thermal-Aware Test Scheduling for Core-based SoC in an Abort-on-First-Fail Test Environment. In *Proceedings of the 12th Euromicro Conference on Digital System Design*, Patras, Greece, August 27 - August 29, pp. 239-246.

HE, Z., PENG, Z. AND ELES, P. 2010a. Multi-Temperature Testing for Core-based System-on-Chip. In *Proceedings of the 2010 Design, Automation and Test in Europe Conference*, Dresden, Germany, March 8 - March 12, pp. 208-213.

HE, Z., PENG, Z. AND ELES, P. 2010b. Thermal-Aware SoC Test Scheduling. (Book Chapter) In *Design and Test Technology for Dependable System-on-Chip*, R. UBAR, J. RAIK AND H.T. VIERHAUS, Eds. IGI Global.

HELLEBRAND, S., TARNICK, S., COURTOIS, B. AND RAJSKI, J. 1992. Generation of Vector Patterns Through Reseeding of Multipe-Polynominal Linear Feedback Shift Registers. In *Proceedings of the 1992 IEEE International Test Conference*, Baltimore, Maryland, USA, September 20 - September 24, pp. 120-129.

HUANG, W., STAN, M.R., SKADRON, K., SANKARANARAYANAN, K., GHOSH, S. AND VELUSAM, S. 2004. Compact Thermal Modeling for Temperature-Aware Design. In *Proceedings of the 41st Design Automation Conference*, San Diego, CA, USA, June 7 - June 11, pp. 878-883.

HUANG, W., GHOSH, S., VELUSAMY, S., SANKARANARAYANAN, K., SKADRON, K. AND STAN, M.R. 2006. HotSpot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(5), pp. 501-513.

HUSS, S.D. AND GYURCSIK, R.S. 1991. Optimal Ordering of Analog Integrated Circuit Tests to Minimize Test Time. In *Proceedings of the 28th ACM/IEEE Design Automation Conference*, San Francisco, CA, USA, June 17 - June 21, pp. 494-499.

HWANG, S. AND ABRAHAM, J.A. 2001. Reuse of Addressable System Bus for SOC Testing. In *Proceedings of the 14th Annual IEEE International ASIC/SOC Conference*, Arlington, VA, USA, September 12 - September 15, pp. 215-219.

IMMANENI, V. AND RAMAN, S. 1990. Direct Access Test Scheme-Design of Block and Core Cells for Embedded ASICs. In *Proceedings of the 1990 IEEE International Test Conference*, Washington, DC, USA, September 10 - September 14, pp. 488-492.

INGELSSON, U., GOEL, S.K., LARSSON, E. AND MARINISSEN, E.J. 2005. Test Scheduling for Modular SOCs in an Abort-on-Fail Environment. In *Proceedings of the 10th IEEE European Test Symposium*pp. 8-13.

IYENGAR, V. AND CHAKRABARTY, K. 2002. System-on-a-Chip Test Scheduling With Precedence Relationships, Preemption, and Power Constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(9), pp. 1088-1094.

IYENGAR, V., CHAKRABARTY, K. AND MARINISSEN, E.J. 2003. Test Access Mechanism Optimization, Test Scheduling, and Tester Data Volume Reduction for System-on-Chip. *IEEE Transactions on Computers*, 52(12), pp. 1619-1632.

JAFFAR, J. AND LASSEZ, J. 1987. Constraint Logic Programming. In *Proceedings of the 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, Munich, West Germany, pp. 111-119.

JERVAN, G., PENG, Z. AND UBAR, R. 2000. Test Cost Minimization for Hybrid BIST. In *Proceedings of the 2000 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, Yamanashi, Japan, October 25 - October 27, pp. 283-291.

JIANG, W. AND VINNAKOTA, B. 2001. Defect-Oriented Test Scheduling. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(3), pp. 427-438.

KORANNE, S. 2002. On Test Scheduling for Core-Based SOCs. In *Proceedings of the 2002 Asia and South Pacific Design Automation Conference*, Bangalore, India, pp. 505-510.

KORF, R.E. 2003. Optimal Rectangle Packing: Initial Results. In *Proceedings of the 13th International Conference on Automated Planning and Scheduling*, Trento, Italy, June 9 - June 13, pp. 287-295.

KORF, R.E. 2004. Optimal Rectangle Packing: New Results. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling*, Whistler, British Columbia, Canada, June 3 - June 7, pp. 142-149.

LARSSON, E. AND PENG, Z. 2002. An Integrated Framework for the Design and Optimization of SOC Test Solutions. *Journal of Electronic Testing: Theory and Applications*, 18(4-5), pp. 385-400.

LARSSON, E., POUGET, J. AND PENG, Z. 2004. Defect-Aware SOC Test Scheduling. In , Napa Valley, CA, USA, April 25 - April 29, pp. 359-364.

LARSSON, E. AND PENG, Z. 2006. Power-Aware Test Planning in the Early System-on-Chip Design Exploration Process. *IEEE Transactions on Computers*, 55(2), pp. 227-239.

LESH, N., MARKS, J., MCMAHON, A. AND MITZENMACHER, M. 2004. Exhaustive Approaches to 2D Rectangular Perfect Packings. *Information Processing Letters*, 90(1), pp. 7-14.

LESH, N., MARKS, J., MCMAHON, A. AND MITZENMACHER, M. 2005. New Heuristic and Interactive Approaches to 2D Rectangular Strip Packing. *Journal of Experimental Algorithmics*, 10(1.2), pp. 1-18.

LIU, C., VEERARAGHAVAN, K. AND IYENGAR, V. 2005. Thermal-Aware Test Scheduling and Hot Spot Temperature Minimization for Core-Based Systems. In *Proceedings of the 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, Monterey, California, USA, October 3 - October 5, pp. 552-560.

MAHAJAN, R. 2002. Thermal Management of CPUs: A Perspective on Trends, Needs and Opportunities. In *Proceedings of the 8th International Workshop on THERMal INvestigations of ICs and Systems (Keynote)*, Madrid, Spain, October 1 - October 4, .

MARINISSEN, E.J., ARENDSEN, R.G.J., BOS, G., DINGEMANSE, H., LOUSBERG, M. AND WOUTERS, C. 1998. A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores. In *Proceedings of International Test Conference (1998)*, Washington, DC, USA, October 18 - October 23, pp. 284-293.

MARINISSEN, E.J., GOEL, S.K. AND LOUSBERG, M. 2000. Wrapper Design for Embedded Core Test. In *Proceedings of International Test Conference (2000)*, Atlantic City, NJ, USA, October 3 - October 5, pp. 911-920.

MILOR, L. AND SANGIOVANNI-VINCENTELLI, A.L. 1994. Minimizing Production Test Time to Detect Faults in Analog Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(6), pp. 796-813.

MONTANES, R.R., VOLF, P. AND GYVEZ, J.P.D. 2002. Resistance Characterization for Weak Open Defects. *IEEE Design and Test of Computers*, 19(5), pp. 18-26.

MOORE, G.E. 1965. Cramming More Components onto Integrated Circuits. *Electronics*, 38(8), pp. 114-117.

MURESAN, V., WANG, X., MURESAN, V. AND VLADUTIU, M. 2000. A Comparison of Classical Scheduling Approaches in Power-Constrained Block-Test Scheduling. In *Proceedings of International Test Conference (2000)*, Atlantic City, NJ, USA, October 3 - October 5, pp. 882-891.

MURRAY, B.T. AND HAYES, J.P. 1996. Testing ICs: Getting to the Core of the Problem. *IEEE Computer*, 29(11), pp. 32-38.

NEEDHAM, W., PRUNTY, C. AND YEOH, E.H. 1998. High Voltage Microprocessor Test Escapes An Analysis of Defects Our Tests are Missing. In *Proceedings of the 1998 IEEE International Test Conference*, Washington, DC, USA, October 18 - October 23, pp. 25-34.

NIGH, P., VALLETT, D.P., PATEL, A. AND WRIGHT, J. 1998. Failure Analysis of Timing and IDDq-only Failures from the SEMATECH Test Methods Experiment. In *Proceedings of the 1998 IEEE International Test Conference*, Washington, DC, USA, October 18 - October 23, pp. 43-52.

POUYA, B. AND CROUCH, A.L. 2000. Optimization Trade-offs for Vector Volume and Test Power. In *Proceedings of the 2000 IEEE International Test Conference*, Atlantic City, NJ, USA, October 3 - October 5, pp. 873-881.

RABAEY, J.M., CHANDRAKASAN, A. AND NIKOLIC, B. 2003. Digital Integrated Circuits. Prentice Hall.

RAVIKUMAR, C.P., CHANDRA, G. AND VERMA, A. 2000. Simultaneous Module Selection and Scheduling for Power-Constrained Testing of Core Based Systems. In *Proceedings of the 13th International Conference on VLSI Design*, Calcutta, India, January 4 - January 7, pp. 462-467.

RINALDI, N.F. 2000. Thermal Analysis of Solid-State Devices and Circuits: An Analytical Approach. *Solid-State Electronics*, 44(10), pp. 1789-1798.

RINALDI, N.F. 2001. On the Modeling of the Transient Thermal Behavior of Semiconductor Devices. *IEEE Transactions on Electron Devices*, 48(12), pp. 279-2802.

ROSINGER, P.M., AL-HASHIMI, B.M. AND NICOLICI, N. 2002. Power Profile Manipulation: A New Approach for Reducing Test Application Time Under Power Constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(10), pp. 1217-1225.

ROSINGER, P.M., AL-HASHIMI, B.M. AND NICOLICI, N. 2004. Scan Architecture With Mutually Exclusive Scan Segment Activation for Shift- and Capture-Power Reduction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(7), pp. 1142-1153.

ROSINGER, P.M., AL-HASHIMI, B.M. AND CHAKRABARTY, K. 2006. Thermal-Safe Test Scheduling for Core-Based System-on-Chip Integrated Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(11), pp. 2502-2512.

SAMII, S., LARSSON, E., CHAKRABARTY, K. AND PENG, Z. 2006. Cycle-Accurate Test Power Modeling and its Application to SoC Test Scheduling. In *Proceedings of the 2006 IEEE International Test Conference*, Santa Clara, CA, USA, October 24 - October 26, pp. 1-10.

SAXENA, J., BUTLER, K.M. AND WHETSEL, L. 2001. An analysis of power reduction techniques in scan testing. In *Proceedings of the 2001 IEEE International Test Conference*, Baltimore, Maryland, USA, October 30 - November 1, pp. 670-677.

SEGURA, J. AND HAWKINS, C.F. 2004. CMOS Electronics: How It Works, How It Fails. Wiley-IEEE Press.

SHI, C. AND KAPUR, R. 2004. How Power-Aware Test Improves Reliability and Yield. 2009, 4. .

SINGER, G., GALIVANCHE, R., PATIL, S. AND TRIPP, M. 2009. The Challenges of Nanotechnology and Gigacomplexity. *IEEE Design and Test of Computers*, 26(1), pp. 88-93.

SKADRON, K., STAN, M.R., SANKARANARAYANAN, K., HUANG, W., VELUSAMY, S. AND TARJAN, D. 2004. Temperature-Aware Microarchitecture: Modeling and Implementation. *ACM Transactions on Architecture and Code Optimization*, 1(1), pp. 94-125.

SUGIHARA, M. AND YASUURA, H. 2000. Analysis and Minimization of Test Time in a Combined BIST and External Test Approach. In *Proceedings of the 2000 Design, Automation and Test in Europe Conference*, Paris, France, March 27 - March 30, pp. 134-140.

TADAYON, P. 2000. Thermal Challenges During Microprocessor Testing. *Intel Technology Journal*, 4(3), pp. 1-8.

TOUBA, N.A. AND MCCLUSKEY, E.J. 1995. Synthesis of Mapping Logic for Generating Transformed Pseudo-Random Patterns for BIST. In *Proceedings of the 1995 IEEE International Test Conference*, Washington, DC, USA, October 21 - October 25, pp. 674-682.

VAN HENTENRYCK, P. 1991. The CLP Language CHIP: Constraint Solving and Applications. In *Compcon Spring '91. Digest of Papers*, San Francisco, CA, USA, February 25 - March 1, pp. 382-387.

VASSIGHI, A. AND SACHDEV, M., Eds. 2006. Thermal and Power Management of Integrated Circuits. Springer.

WANG, L., STROUD, C. AND TOUBA, N.A., Eds. 2007. System-on-Chip Test Architectures: Nanometer Design for Testability. Morgan Kaufmann.

WANG, S. AND GUPTA, S.K. 1997. DS-LFSR: A New BIST TPG for Low Heat Dissipation. In *Proceedings of the 1997 IEEE International Test Conference*, Washington, DC, USA, November 1 - November 6, pp. 848-857.

WOLPERT, D. AND AMPADU, P. 2009. A Sensor to Detect Normal or Reverse Temperature Dependence in Nanoscale CMOS Circuits. In *Proceedings of the 24th IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*, Chicago, IL, USA, October 7 - October 9, pp. 193-201.

YANG, Y., GU, Z., ZHU, C., DICK, R.P. AND SHANG, L. 2007. ISAC: Integrated Space-and-Time-Adaptive Chip-Package Thermal Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(1), pp. 86-99.

YU, T.E., YONEDA, T., CHAKRABARTY, K. AND FUJIWARA, H. 2007. Thermal-Safe Test Access Mechanism and Wrapper Co-optimization for System-on-Chip. In *Proceedings of the 16th Asian Test Symposium*, Beijing, China, October 8 - October 11, pp. 187-192.

ZORIAN, Y. 1993. A Distributed BIST Control Scheme for Complex VLSI Devices. In *Proceedings of the 11th VLSI Test Symposium*, Atlantic City, NJ, USA, April 6 - April 8, pp. 4-9.

ZORIAN, Y., MARINISSEN, E.J. AND DEY, S. 1999. Testing Embedded-Core-Based System Chips. *IEEE Computer*, 32(6), pp. 52-60.

Department of Computer and Information Science
Linköpings universitet

Dissertations

Linköping Studies in Science and Technology

No 14 **Anders Haraldsson:** A Program Manipulation System Based on Partial Evaluation, 1977, ISBN 91-7372-144-1.

No 17 **Bengt Magnhagen:** Probability Based Verification of Time Margins in Digital Designs, 1977, ISBN 91-7372-157-3.

No 18 **Mats Cedwall**: Semantisk analys av process-beskrivningar i naturligt språk, 1977, ISBN 91- 7372-168-9.

No 22 **Jaak Urmi:** A Machine Independent LISP Compiler and its Implications for Ideal Hardware, 1978, ISBN 91-7372-188-3.

No 33 **Tore Risch:** Compilation of Multiple File Queries in a Meta-Database System 1978, ISBN 91- 7372-232-4.

No 51 **Erland Jungert:** Synthesizing Database Structures from a User Oriented Data Model, 1980, ISBN 91-7372-387-8.

No 54 **Sture Hägglund:** Contributions to the Development of Methods and Tools for Interactive Design of Applications Software, 1980, ISBN 91-7372-404-1.

No 55 **Pär Emanuelson:** Performance Enhancement in a Well-Structured Pattern Matcher through Partial Evaluation, 1980, ISBN 91-7372-403-3.

No 58 **Bengt Johnsson, Bertil Andersson:** The Human-Computer Interface in Commercial Systems, 1981, ISBN 91-7372-414-9.

No 69 **H. Jan Komorowski:** A Specification of an Abstract Prolog Machine and its Application to Partial Evaluation, 1981, ISBN 91-7372-479-3.

No 71 **René Reboh:** Knowledge Engineering Techniques and Tools for Expert Systems, 1981, ISBN 91-7372-489-0.

No 77 **Östen Oskarsson:** Mechanisms of Modifiability in large Software Systems, 1982, ISBN 91- 7372-527-7.

No 94 **Hans Lunell:** Code Generator Writing Systems, 1983, ISBN 91-7372-652-4.

No 97 **Andrzej Lingas:** Advances in Minimum Weight Triangulation, 1983, ISBN 91-7372-660-5.

No 109 **Peter Fritzson:** Towards a Distributed Programming Environment based on Incremental Compilation, 1984, ISBN 91-7372-801-2.

No 111 **Erik Tengvald:** The Design of Expert Planning Systems. An Experimental Operations Planning System for Turning, 1984, ISBN 91-7372- 805-5.

No 155 **Christos Levcopoulos:** Heuristics for Minimum Decompositions of Polygons, 1987, ISBN 91-7870-133-3.

No 165 **James W. Goodwin:** A Theory and System for Non-Monotonic Reasoning, 1987, ISBN 91-7870-183-X.

No 170 **Zebo Peng:** A Formal Methodology for Automated Synthesis of VLSI Systems, 1987, ISBN 91-7870-225-9.

No 174 **Johan Fagerström:** A Paradigm and System for Design of Distributed Systems, 1988, ISBN 91-7870-301-8.

No 192 **Dimiter Driankov:** Towards a Many Valued Logic of Quantified Belief, 1988, ISBN 91-7870-374-3.

No 213 **Lin Padgham:** Non-Monotonic Inheritance for an Object Oriented Knowledge Base, 1989, ISBN 91-7870-485-5.

No 214 **Tony Larsson:** A Formal Hardware Description and Verification Method, 1989, ISBN 91-7870-517-7.

No 221 **Michael Reinfrank:** Fundamentals and Logical Foundations of Truth Maintenance, 1989, ISBN 91-7870-546-0.

No 239 **Jonas Löwgren:** Knowledge-Based Design Support and Discourse Management in User Interface Management Systems, 1991, ISBN 91-7870-720-X.

No 244 **Henrik Eriksson:** Meta-Tool Support for Knowledge Acquisition, 1991, ISBN 91-7870-746-3.

No 252 **Peter Eklund:** An Epistemic Approach to Interactive Design in Multiple Inheritance Hierarchies, 1991, ISBN 91-7870-784-6.

No 258 **Patrick Doherty:** NML3 - A Non-Monotonic Formalism with Explicit Defaults, 1991, ISBN 91-7870-816-8.

No 260 **Nahid Shahmehri:** Generalized Algorithmic Debugging, 1991, ISBN 91-7870-828-1.

No 264 **Nils Dahlbäck:** Representation of Discourse-Cognitive and Computational Aspects, 1992, ISBN 91-7870-850-8.

No 265 **Ulf Nilsson:** Abstract Interpretations and Abstract Machines: Contributions to a Methodology for the Implementation of Logic Programs, 1992, ISBN 91-7870-858-3.

No 270 **Ralph Rönnquist:** Theory and Practice of Tense-bound Object References, 1992, ISBN 91-7870-873-7.

No 273 **Björn Fjellborg:** Pipeline Extraction for VLSI Data Path Synthesis, 1992, ISBN 91-7870-880-X.

No 276 **Staffan Bonnier:** A Formal Basis for Horn Clause Logic with External Polymorphic Functions, 1992, ISBN 91-7870-896-6.

No 277 **Kristian Sandahl:** Developing Knowledge Management Systems with an Active Expert Methodology, 1992, ISBN 91-7870-897-4.

No 281 **Christer Bäckström:** Computational Complexity of Reasoning about Plans, 1992, ISBN 91-7870-979-2.

No 292 **Mats Wirén:** Studies in Incremental Natural Language Analysis, 1992, ISBN 91-7871-027-8.

No 297 **Mariam Kamkar:** Interprocedural Dynamic Slicing with Applications to Debugging and Testing, 1993, ISBN 91-7871-065-0.

No 302 **Tingting Zhang:** A Study in Diagnosis Using Classification and Defaults, 1993, ISBN 91-7871-078-2

No 312 **Arne Jönsson:** Dialogue Management for Natural Language Interfaces - An Empirical Approach, 1993, ISBN 91-7871-110-X.

No 338 **Simin Nadjm-Tehrani:** Reactive Systems in Physical Environments: Compositional Modelling and Framework for Verification, 1994, ISBN 91-7871-237-8.

No 371 **Bengt Savén:** Business Models for Decision Support and Learning. A Study of Discrete-Event Manufacturing Simulation at Asea/ABB 1968-1993, 1995, ISBN 91-7871-494-X.

No 375 **Ulf Söderman:** Conceptual Modelling of Mode Switching Physical Systems, 1995, ISBN 91-7871-516-4.

No 383 **Andreas Kågedal:** Exploiting Groundness in Logic Programs, 1995, ISBN 91-7871-538-5.

No 396 **George Fodor:** Ontological Control, Description, Identification and Recovery from Problematic Control Situations, 1995, ISBN 91-7871-603-9.

No 413 **Mikael Pettersson:** Compiling Natural Semantics, 1995, ISBN 91-7871-641-1.

No 414 **Xinli Gu:** RT Level Testability Improvement by Testability Analysis and Transformations, 1996, ISBN 91-7871-654-3.

No 416 **Hua Shu:** Distributed Default Reasoning, 1996, ISBN 91-7871-665-9.

No 429 **Jaime Villegas:** Simulation Supported Industrial Training from an Organisational Learning Perspective - Development and Evaluation of the SSIT Method, 1996, ISBN 91-7871-700-0.

No 431 **Peter Jonsson:** Studies in Action Planning: Algorithms and Complexity, 1996, ISBN 91-7871-704-3.

No 437 **Johan Boye:** Directional Types in Logic Programming, 1996, ISBN 91-7871-725-6.

No 439 **Cecilia Sjöberg:** Activities, Voices and Arenas: Participatory Design in Practice, 1996, ISBN 91-7871-728-0.

No 448 **Patrick Lambrix:** Part-Whole Reasoning in Description Logics, 1996, ISBN 91-7871-820-1.

No 452 **Kjell Orsborn:** On Extensible and Object-Relational Database Technology for Finite Element Analysis Applications, 1996, ISBN 91-7871-827-9.

No 459 **Olof Johansson:** Development Environments for Complex Product Models, 1996, ISBN 91-7871-855-4.

No 461 **Lena Strömbäck:** User-Defined Constructions in Unification-Based Formalisms, 1997, ISBN 91-7871-857-0.

No 462 **Lars Degerstedt:** Tabulation-based Logic Programming: A Multi-Level View of Query Answering, 1996, ISBN 91-7871-858-9.

No 475 **Fredrik Nilsson:** Strategi och ekonomisk styrning - En studie av hur ekonomiska styrsystem utformas och används efter företagsförvärv, 1997, ISBN 91-7871-914-3.

No 480 **Mikael Lindvall:** An Empirical Study of Requirements-Driven Impact Analysis in Object-Oriented Software Evolution, 1997, ISBN 91-7871-927-5.

No 485 **Göran Forslund:** Opinion-Based Systems: The Cooperative Perspective on Knowledge-Based Decision Support, 1997, ISBN 91-7871-938-0.

No 494 **Martin Sköld:** Active Database Management Systems for Monitoring and Control, 1997, ISBN 91-7219-002-7.

No 495 **Hans Olsén:** Automatic Verification of Petri Nets in a CLP framework, 1997, ISBN 91-7219-011-6.

No 498 **Thomas Drakengren:** Algorithms and Complexity for Temporal and Spatial Formalisms, 1997, ISBN 91-7219-019-1.

No 502 **Jakob Axelsson:** Analysis and Synthesis of Heterogeneous Real-Time Systems, 1997, ISBN 91-7219-035-3.

No 503 **Johan Ringström:** Compiler Generation for Data-Parallel Programming Languages from Two-Level Semantics Specifications, 1997, ISBN 91-7219-045-0.

No 512 **Anna Moberg:** Närhet och distans - Studier av kommunikationsmönster i satellitkontor och flexibla kontor, 1997, ISBN 91-7219-119-8.

No 520 **Mikael Ronström:** Design and Modelling of a Parallel Data Server for Telecom Applications, 1998, ISBN 91-7219-169-4.

No 522 **Niclas Ohlsson:** Towards Effective Fault Prevention - An Empirical Study in Software Engineering, 1998, ISBN 91-7219-176-7.

No 526 **Joachim Karlsson:** A Systematic Approach for Prioritizing Software Requirements, 1998, ISBN 91-7219-184-8.

No 530 **Henrik Nilsson:** Declarative Debugging for Lazy Functional Languages, 1998, ISBN 91-7219-197-x.

No 555 **Jonas Hallberg:** Timing Issues in High-Level Synthesis, 1998, ISBN 91-7219-369-7.

No 561 **Ling Lin:** Management of 1-D Sequence Data - From Discrete to Continuous, 1999, ISBN 91-7219-402-2.

No 563 **Eva L Ragnemalm:** Student Modelling based on Collaborative Dialogue with a Learning Companion, 1999, ISBN 91-7219-412-X.

No 567 **Jörgen Lindström:** Does Distance matter? On geographical dispersion in organisations, 1999, ISBN 91-7219-439-1.

No 582 **Vanja Josifovski:** Design, Implementation and Evaluation of a Distributed Mediator System for Data Integration, 1999, ISBN 91-7219-482-0.

No 589 **Rita Kovordányi:** Modeling and Simulating Inhibitory Mechanisms in Mental Image Reinterpretation - Towards Cooperative Human-Computer Creativity, 1999, ISBN 91-7219-506-1.

No 592 **Mikael Ericsson:** Supporting the Use of Design Knowledge - An Assessment of Commenting Agents, 1999, ISBN 91-7219-532-0.

No 593 **Lars Karlsson:** Actions, Interactions and Narratives, 1999, ISBN 91-7219-534-7.

No 594 **C. G. Mikael Johansson:** Social and Organizational Aspects of Requirements Engineering Methods - A practice-oriented approach, 1999, ISBN 91-7219-541-X.

No 595 **Jörgen Hansson:** Value-Driven Multi-Class Overload Management in Real-Time Database Systems, 1999, ISBN 91-7219-542-8.

No 596 **Niklas Hallberg:** Incorporating User Values in the Design of Information Systems and Services in the Public Sector: A Methods Approach, 1999, ISBN 91-7219-543-6.

No 597 **Vivian Vimarlund:** An Economic Perspective on the Analysis of Impacts of Information Technology: From Case Studies in Health-Care towards General Models and Theories, 1999, ISBN 91-7219-544-4.

No 598 **Johan Jenvald:** Methods and Tools in Computer-Supported Taskforce Training, 1999, ISBN 91-7219-547-9.

No 607 **Magnus Merkel:** Understanding and enhancing translation by parallel text processing, 1999, ISBN 91-7219-614-9.

No 611 **Silvia Coradeschi:** Anchoring symbols to sensory data, 1999, ISBN 91-7219-623-8.

No 613 **Man Lin:** Analysis and Synthesis of Reactive Systems: A Generic Layered Architecture Perspective, 1999, ISBN 91-7219-630-0.

No 618 **Jimmy Tjäder:** Systemimplementering i praktiken - En studie av logiker i fyra projekt, 1999, ISBN 91-7219-657-2.

No 627 **Vadim Engelson:** Tools for Design, Interactive Simulation, and Visualization of Object-Oriented Models in Scientific Computing, 2000, ISBN 91-7219-709-9.

No 637 **Esa Falkenroth:** Database Technology for Control and Simulation, 2000, ISBN 91-7219-766-8.

No 639 **Per-Arne Persson:** Bringing Power and Knowledge Together: Information Systems Design for Autonomy and Control in Command Work, 2000, ISBN 91-7219-796-X.

No 660 **Erik Larsson:** An Integrated System-Level Design for Testability Methodology, 2000, ISBN 91-7219-890-7.

No 688 **Marcus Bjäreland:** Model-based Execution Monitoring, 2001, ISBN 91-7373-016-5.

No 689 **Joakim Gustafsson:** Extending Temporal Action Logic, 2001, ISBN 91-7373-017-3.

No 720 **Carl-Johan Petri:** Organizational Information Provision - Managing Mandatory and Discretionary Use of Information Technology, 2001, ISBN-91-7373-126-9.

No 724 **Paul Scerri:** Designing Agents for Systems with Adjustable Autonomy, 2001, ISBN 91 7373 207 9.

No 725 **Tim Heyer:** Semantic Inspection of Software Artifacts: From Theory to Practice, 2001, ISBN 91 7373 208 7.

No 726 **Pär Carlshamre:** A Usability Perspective on Requirements Engineering - From Methodology to Product Development, 2001, ISBN 91 7373 212 5.

No 732 **Juha Takkinen:** From Information Management to Task Management in Electronic Mail, 2002, ISBN 91 7373 258 3.

No 745 **Johan Åberg:** Live Help Systems: An Approach to Intelligent Help for Web Information Systems, 2002, ISBN 91-7373-311-3.

No 746 **Rego Granlund:** Monitoring Distributed Teamwork Training, 2002, ISBN 91-7373-312-1.

No 757 **Henrik André-Jönsson:** Indexing Strategies for Time Series Data, 2002, ISBN 917373-346-6.

No 747 **Anneli Hagdahl:** Development of IT-supported Interorganisational Collaboration - A Case Study in the Swedish Public Sector, 2002, ISBN 91-7373-314-8.

No 749 **Sofie Pilemalm:** Information Technology for Non-Profit Organisations - Extended Participatory Design of an Information System for Trade Union Shop Stewards, 2002, ISBN 91-7373-318-0.

No 765 **Stefan Holmlid:** Adapting users: Towards a theory of use quality, 2002, ISBN 91-7373-397-0.

No 771 **Magnus Morin:** Multimedia Representations of Distributed Tactical Operations, 2002, ISBN 91-7373-421-7.

No 772 **Pawel Pietrzak:** A Type-Based Framework for Locating Errors in Constraint Logic Programs, 2002, ISBN 91-7373-422-5.

No 758 **Erik Berglund:** Library Communication Among Programmers Worldwide, 2002, ISBN 91-7373-349-0.

No 774 **Choong-ho Yi:** Modelling Object-Oriented Dynamic Systems Using a Logic-Based Framework, 2002, ISBN 91-7373-424-1.

No 779 **Mathias Broxvall:** A Study in the Computational Complexity of Temporal Reasoning, 2002, ISBN 91-7373-440-3.

No 793 **Asmus Pandikow:** A Generic Principle for Enabling Interoperability of Structured and Object-Oriented Analysis and Design Tools, 2002, ISBN 91-7373-479-9.

No 785 **Lars Hult:** Publika Informationstjänster. En studie av den Internetbaserade encyklopedins bruksegenskaper, 2003, ISBN 91-7373-461-6.

No 800 **Lars Taxén:** A Framework for the Coordination of Complex Systems´ Development, 2003, ISBN 91-7373-604-X

No 808 **Klas Gäre:** Tre perspektiv på förväntningar och förändringar i samband med införande av informationssystem, 2003, ISBN 91-7373-618-X.

No 821 **Mikael Kindborg:** Concurrent Comics - programming of social agents by children, 2003, ISBN 91-7373-651-1.

No 823 **Christina Ölvingson:** On Development of Information Systems with GIS Functionality in Public Health Informatics: A Requirements Engineering Approach, 2003, ISBN 91-7373-656-2.

No 828 **Tobias Ritzau:** Memory Efficient Hard Real-Time Garbage Collection, 2003, ISBN 91-7373-666-X.

No 833 **Paul Pop:** Analysis and Synthesis of Communication-Intensive Heterogeneous Real-Time Systems, 2003, ISBN 91-7373-683-X.

No 852 **Johan Moe:** Observing the Dynamic Behaviour of Large Distributed Systems to Improve Development and Testing – An Empirical Study in Software Engineering, 2003, ISBN 91-7373-779-8.

No 867 **Erik Herzog:** An Approach to Systems Engineering Tool Data Representation and Exchange, 2004, ISBN 91-7373-929-4.

No 872 **Aseel Berglund:** Augmenting the Remote Control: Studies in Complex Information Navigation for Digital TV, 2004, ISBN 91-7373-940-5.

No 869 **Jo Skåmedal:** Telecommuting's Implications on Travel and Travel Patterns, 2004, ISBN 91-7373-935-9.

No 870 **Linda Askenäs:** The Roles of IT - Studies of Organising when Implementing and Using Enterprise Systems, 2004, ISBN 91-7373-936-7.

No 874 **Annika Flycht-Eriksson:** Design and Use of Ontologies in Information-Providing Dialogue Systems, 2004, ISBN 91-7373-947-2.

No 873 **Peter Bunus:** Debugging Techniques for Equation-Based Languages, 2004, ISBN 91-7373-941-3.

No 876 **Jonas Mellin:** Resource-Predictable and Efficient Monitoring of Events, 2004, ISBN 91-7373-956-1.

No 883 **Magnus Bång:** Computing at the Speed of Paper: Ubiquitous Computing Environments for Healthcare Professionals, 2004, ISBN 91-7373-971-5

No 882 **Robert Eklund:** Disfluency in Swedish human-human and human-machine travel booking dialogues, 2004, ISBN 91-7373-966-9.

No 887 **Anders Lindström:** English and other Foreign Linguistic Elements in Spoken Swedish. Studies of Productive Processes and their Modelling using Finite-State Tools, 2004, ISBN 91-7373-981-2.

No 889 **Zhiping Wang:** Capacity-Constrained Production-inventory systems - Modelling and Analysis in both a traditional and an e-business context, 2004, ISBN 91-85295-08-6.

No 893 **Pernilla Qvarfordt:** Eyes on Multimodal Interaction, 2004, ISBN 91-85295-30-2.

No 910 **Magnus Kald:** In the Borderland between Strategy and Management Control - Theoretical Framework and Empirical Evidence, 2004, ISBN 91-85295-82-5.

No 918 **Jonas Lundberg:** Shaping Electronic News: Genre Perspectives on Interaction Design, 2004, ISBN 91-85297-14-3.

No 900 **Mattias Arvola:** Shades of use: The dynamics of interaction design for sociable use, 2004, ISBN 91-85295-42-6.

No 920 **Luis Alejandro Cortés:** Verification and Scheduling Techniques for Real-Time Embedded Systems, 2004, ISBN 91-85297-21-6.

No 929 **Diana Szentivanyi:** Performance Studies of Fault-Tolerant Middleware, 2005, ISBN 91-85297-58-5.

No 933 **Mikael Cäker:** Management Accounting as Constructing and Opposing Customer Focus: Three Case Studies on Management Accounting and Customer Relations, 2005, ISBN 91-85297-64-X.

No 937 **Jonas Kvarnström:** TALplanner and Other Extensions to Temporal Action Logic, 2005, ISBN 91-85297-75-5.

No 938 **Bourhane Kadmiry:** Fuzzy Gain-Scheduled Visual Servoing for Unmanned Helicopter, 2005, ISBN 91-85297-76-3.

No 945 **Gert Jervan:** Hybrid Built-In Self-Test and Test Generation Techniques for Digital Systems, 2005, ISBN: 91-85297-97-6.

No 946 **Anders Arpteg:** Intelligent Semi-Structured Information Extraction, 2005, ISBN 91-85297-98-4.

No 947 **Ola Angelsmark:** Constructing Algorithms for Constraint Satisfaction and Related Problems - Methods and Applications, 2005, ISBN 91-85297-99-2.

No 963 **Calin Curescu:** Utility-based Optimisation of Resource Allocation for Wireless Networks, 2005, ISBN 91-85457-07-8.

No 972 **Björn Johansson:** Joint Control in Dynamic Situations, 2005, ISBN 91-85457-31-0.

No 974 **Dan Lawesson:** An Approach to Diagnosability Analysis for Interacting Finite State Systems, 2005, ISBN 91-85457-39-6.

No 979 **Claudiu Duma:** Security and Trust Mechanisms for Groups in Distributed Services, 2005, ISBN 91-85457-54-X.

No 983 **Sorin Manolache:** Analysis and Optimisation of Real-Time Systems with Stochastic Behaviour, 2005, ISBN 91-85457-60-4.

No 986 **Yuxiao Zhao:** Standards-Based Application Integration for Business-to-Business Communications, 2005, ISBN 91-85457-66-3.

No 1004 **Patrik Haslum:** Admissible Heuristics for Automated Planning, 2006, ISBN 91-85497-28-2.

No 1005 **Aleksandra Tešanovic:** Developing Reusable and Reconfigurable Real-Time Software using Aspects and Components, 2006, ISBN 91-85497-29-0.

No 1008 **David Dinka:** Role, Identity and Work: Extending the design and development agenda, 2006, ISBN 91-85497-42-8.

No 1009 **Iakov Nakhimovski:** Contributions to the Modeling and Simulation of Mechanical Systems with Detailed Contact Analysis, 2006, ISBN 91-85497-43-X.

No 1013 **Wilhelm Dahllöf:** Exact Algorithms for Exact Satisfiability Problems, 2006, ISBN 91-85523-97-6.

No 1016 **Levon Saldamli:** PDEModelica - A High-Level Language for Modeling with Partial Differential Equations, 2006, ISBN 91-85523-84-4.

No 1017 **Daniel Karlsson:** Verification of Component-based Embedded System Designs, 2006, ISBN 91-85523-79-8

No 1018 **Ioan Chisalita:** Communication and Networking Techniques for Traffic Safety Systems, 2006, ISBN 91-85523-77-1.

No 1019 **Tarja Susi:** The Puzzle of Social Activity - The Significance of Tools in Cognition and Cooperation, 2006, ISBN 91-85523-71-2.

No 1021 **Andrzej Bednarski:** Integrated Optimal Code Generation for Digital Signal Processors, 2006, ISBN 91-85523-69-0.

No 1022 **Peter Aronsson:** Automatic Parallelization of Equation-Based Simulation Programs, 2006, ISBN 91-85523-68-2.

No 1030 **Robert Nilsson:** A Mutation-based Framework for Automated Testing of Timeliness, 2006, ISBN 91-85523-35-6.

No 1034 **Jon Edvardsson:** Techniques for Automatic Generation of Tests from Programs and Specifications, 2006, ISBN 91-85523-31-3.

No 1035 **Vaida Jakoniene:** Integration of Biological Data, 2006, ISBN 91-85523-28-3.

No 1045 **Genevieve Gorrell:** Generalized Hebbian Algorithms for Dimensionality Reduction in Natural Language Processing, 2006, ISBN 91-85643-88-2.

No 1051 **Yu-Hsing Huang:** Having a New Pair of Glasses - Applying Systemic Accident Models on Road Safety, 2006, ISBN 91-85643-64-5.

No 1054 **Åsa Hedenskog:** Perceive those things which cannot be seen - A Cognitive Systems Engineering perspective on requirements management, 2006, ISBN 91-85643-57-2.

No 1061 **Cécile Åberg:** An Evaluation Platform for Semantic Web Technology, 2007, ISBN 91-85643-31-9.

No 1073 **Mats Grindal:** Handling Combinatorial Explosion in Software Testing, 2007, ISBN 978-91-85715-74-9.

No 1075 **Almut Herzog:** Usable Security Policies for Runtime Environments, 2007, ISBN 978-91-85715-65-7.

No 1079 **Magnus Wahlström:** Algorithms, measures, and upper bounds for Satisfiability and related problems, 2007, ISBN 978-91-85715-55-8.

No 1083 **Jesper Andersson:** Dynamic Software Architectures, 2007, ISBN 978-91-85715-46-6.

No 1086 **Ulf Johansson:** Obtaining Accurate and Comprehensible Data Mining Models - An Evolutionary Approach, 2007, ISBN 978-91-85715-34-3.

No 1089 **Traian Pop:** Analysis and Optimisation of Distributed Embedded Systems with Heterogeneous Scheduling Policies, 2007, ISBN 978-91-85715-27-5.

No 1091 **Gustav Nordh:** Complexity Dichotomies for CSP-related Problems, 2007, ISBN 978-91-85715-20-6.

No 1106 **Per Ola Kristensson:** Discrete and Continuous Shape Writing for Text Entry and Control, 2007, ISBN 978-91-85831-77-7.

No 1110 **He Tan:** Aligning Biomedical Ontologies, 2007, ISBN 978-91-85831-56-2.

No 1112 **Jessica Lindblom:** Minding the body - Interacting socially through embodied action, 2007, ISBN 978-91-85831-48-7.

No 1113 **Pontus Wärnestål:** Dialogue Behavior Management in Conversational Recommender Systems, 2007, ISBN 978-91-85831-47-0.

No 1120 **Thomas Gustafsson:** Management of Real-Time Data Consistency and Transient Overloads in Embedded Systems, 2007, ISBN 978-91-85831-33-3.

No 1127 **Alexandru Andrei:** Energy Efficient and Predictable Design of Real-time Embedded Systems, 2007, ISBN 978-91-85831-06-7.

No 1139 **Per Wikberg:** Eliciting Knowledge from Experts in Modeling of Complex Systems: Managing Variation and Interactions, 2007, ISBN 978-91-85895-66-3.

No 1143 **Mehdi Amirijoo:** QoS Control of Real-Time Data Services under Uncertain Workload, 2007, ISBN 978-91-85895-49-6.

No 1150 **Sanny Syberfeldt:** Optimistic Replication with Forward Conflict Resolution in Distributed Real-Time Databases, 2007, ISBN 978-91-85895-27-4.

No 1155 **Beatrice Alenljung:** Envisioning a Future Decision Support System for Requirements Engineering - A Holistic and Human-centred Perspective, 2008, ISBN 978-91-85895-11-3.

No 1156 **Artur Wilk:** Types for XML with Application to Xcerpt, 2008, ISBN 978-91-85895-08-3.

No 1183 **Adrian Pop:** Integrated Model-Driven Development Environments for Equation-Based Object-Oriented Languages, 2008, ISBN 978-91-7393-895-2.

No 1185 **Jörgen Skågeby:** Gifting Technologies - Ethnographic Studies of End-users and Social Media Sharing, 2008, ISBN 978-91-7393-892-1.

No 1187 **Imad-Eldin Ali Abugessaisa:** Analytical tools and information-sharing methods supporting road safety organizations, 2008, ISBN 978-91-7393-887-7.

No 1204 **H. Joe Steinhauer:** A Representation Scheme for Description and Reconstruction of Object Configurations Based on Qualitative Relations, 2008, ISBN 978-91-7393-823-5.

No 1222 **Anders Larsson:** Test Optimization for Core-based System-on-Chip, 2008, ISBN 978-91-7393-768-9.

No 1238 **Andreas Borg:** Processes and Models for Capacity Requirements in Telecommunication Systems, 2009, ISBN 978-91-7393-700-9.

No 1240 **Fredrik Heintz:** DyKnow: A Stream-Based Knowledge Processing Middleware Framework, 2009, ISBN 978-91-7393-696-5.

No 1241 **Birgitta Lindström:** Testability of Dynamic Real-Time Systems, 2009, ISBN 978-91-7393-695-8.

No 1244 **Eva Blomqvist:** Semi-automatic Ontology Construction based on Patterns, 2009, ISBN 978-91-7393-683-5.

No 1249 **Rogier Woltjer:** Functional Modeling of Constraint Management in Aviation Safety and Command and Control, 2009, ISBN 978-91-7393-659-0.

No 1260 **Gianpaolo Conte:** Vision-Based Localization and Guidance for Unmanned Aerial Vehicles, 2009, ISBN 978-91-7393-603-3.

No 1262 **AnnMarie Ericsson:** Enabling Tool Support for Formal Analysis of ECA Rules, 2009, ISBN 978-91-7393-598-2.

No 1266 **Jiri Trnka:** Exploring Tactical Command and Control: A Role-Playing Simulation Approach, 2009, ISBN 978-91-7393-571-5.

No 1268 **Bahlol Rahimi:** Supporting Collaborative Work through ICT - How End-users Think of and Adopt Integrated Health Information Systems, 2009, ISBN 978-91-7393-550-0.

No 1274 **Fredrik Kuivinen:** Algorithms and Hardness Results for Some Valued CSPs, 2009, ISBN 978-91-7393-525-8.

No 1281 **Gunnar Mathiason:** Virtual Full Replication for Scalable Distributed Real-Time Databases, 2009, ISBN 978-91-7393-503-6.

No 1290 **Viacheslav Izosimov:** Scheduling and Optimization of Fault-Tolerant Distributed Embedded Systems, 2009, ISBN 978-91-7393-482-4.

No 1294 **Johan Thapper:** Aspects of a Constraint Optimisation Problem, 2010, ISBN 978-91-7393-464-0.

No 1306 **Susanna Nilsson:** Augmentation in the Wild: User Centered Development and Evaluation of Augmented Reality Applications, 2010, ISBN 978-91-7393-416-9.

No 1313 **Christer Thörn:** On the Quality of Feature Models, 2010, ISBN 978-91-7393-394-0.

No 1321 **Zhiyuan He:** Temperature Aware and Defect-Probability Driven Test Scheduling for System-on-Chip, 2010, ISBN 978-91-7393-378-0.

Linköping Studies in Arts and Sciences

No 504 **Ing-Marie Jonsson:** Social and Emotional Characteristics of Speech-based In-Vehicle Information Systems: Impact on Attitude and Driving Behaviour, 2009, ISBN 978-91-7393-478-7.

Linköping Studies in Statistics

No 9 **Davood Shahsavani:** Computer Experiments Designed to Explore and Approximate Complex Deterministic Models, 2008, ISBN 978-91-7393-976-8.

No 10 **Karl Wahlin:** Roadmap for Trend Detection and Assessment of Data Quality, 2008, ISBN 978-91-7393-792-4.

No 11 **Oleg Sysoev:** Monotonic regression for large multivariate datasets, 2010, ISBN 978-91-7393-412-1.

Linköping Studies in Information Science

No 1 **Karin Axelsson:** Metodisk systemstrukturering- att skapa samstämmighet mellan informationssystemarkitektur och verksamhet, 1998. ISBN-9172-19-296-8.

No 2 **Stefan Cronholm:** Metodverktyg och användbarhet - en studie av datorstödd metodbaserad systemutveckling, 1998, ISBN-9172-19-299-2.

No 3 **Anders Avdic:** Användare och utvecklare - om anveckling med kalkylprogram, 1999. ISBN-91-7219-606-8.

No 4 **Owen Eriksson:** Kommunikationskvalitet hos informationssystem och affärsprocesser, 2000, ISBN 91-7219-811-7.

No 5 **Mikael Lind:** Från system till process - kriterier för processbestämning vid verksamhetsanalys, 2001, ISBN 91-7373-067-X.

No 6 **Ulf Melin:** Koordination och informationssystem i företag och nätverk, 2002, ISBN 91-7373-278-8.

No 7 **Pär J. Ågerfalk:** Information Systems Actability - Understanding Information Technology as a Tool for Business Action and Communication, 2003, ISBN 91-7373-628-7.

No 8 **Ulf Seigerroth:** Att förstå och förändra systemutvecklingsverksamheter - en taxonomi för metautveckling, 2003, ISBN91-7373-736-4.

No 9 **Karin Hedström:** Spår av datoriseringens värden – Effekter av IT i äldreomsorg, 2004, ISBN 91-7373-963-4.

No 10 **Ewa Braf:** Knowledge Demanded for Action - Studies on Knowledge Mediation in Organisations, 2004, ISBN 91-85295-47-7.

No 11 **Fredrik Karlsson:** Method Configuration method and computerized tool support, 2005, ISBN 91-85297-48-8.

No 12 **Malin Nordström:** Styrbar systemförvaltning - Att organisera systemförvaltningsverksamhet med hjälp av effektiva förvaltningsobjekt, 2005, ISBN 91-85297-60-7.

No 13 **Stefan Holgersson:** Yrke: POLIS - Yrkeskunskap, motivation, IT-system och andra förutsättningar för polisarbete, 2005, ISBN 91-85299-43-X.

No 14 **Benneth Christiansson, Marie-Therese Christiansson:** Mötet mellan process och komponent - mot ett ramverk för en verksamhetsnära kravspecifikation vid anskaffning av komponentbaserade informationssystem, 2006, ISBN 91-85643-22-X.