Linköping Studies in Science and Technology

Thesis No. 1313

System-on-Chip Test Scheduling with Defect-Probability and Temperature Considerations

by

Zhiyuan He

Linköping University
INSTITUTE OF TECHNOLOGY

Submitted to Linköping Institute of Technology at Linköping University in partial fulfilment of the requirements for the degree of Licentiate of Engineering

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

Linköping 2007

ISBN 978-91-85831-81-4 ISSN 0280-7971
Printed by LiU-Tryck
Linköping, Sweden 2007

Abstract

Electronic systems have become highly complex, which results in a dramatic increase of both design and production cost. Recently a core-based system-on-chip (SoC) design methodology has been employed in order to reduce these costs. However, testing of SoCs has been facing challenges such as long test application time and high temperature during test. In this thesis, we address the problem of minimizing test application time for SoCs and propose three techniques to generate efficient test schedules.

First, a defect-probability driven test scheduling technique is presented for production test, in which an abort-on-first-fail (AOFF) test approach is employed and a hybrid built-in self-test architecture is assumed. Using an AOFF test approach, the test process can be aborted as soon as the first fault is detected. Given the defect probabilities of individual cores, a method is proposed to calculate the expected test application time (ETAT). A heuristic is then proposed to generate test schedules with minimized ETATs.

Second, a power-constrained test scheduling approach using test set partitioning is proposed. It assumes that, during the test, the total amount of power consumed by the cores being tested in parallel has to be lower than a given limit. A heuristic is proposed to minimize the test application time, in which a test set partitioning technique is employed to generate more efficient test schedules.

Third, a thermal-aware test scheduling approach is presented, in which test set partitioning and interleaving are employed. A

constraint logic programming (CLP) approach is deployed to find the optimal solution. Moreover, a heuristic is also developed to generate near-optimal test schedules especially for large designs to which the CLP-based algorithm is inapplicable.

Experiments based on benchmark designs have been carried out to demonstrate the applicability and efficiency of the proposed techniques.

Acknowledgments

It has been a great pleasure for me to work on this thesis. Many people have contributed to it. I appreciate this and I wish to take the opportunity to thank them all.

First of all, I would like to sincerely thank my supervisors Professor Zebo Peng and Professor Petru Eles, for their great support. It is their guidance and encouragement that have been leading me on my study and research throughout all these years. Many of their creative thoughts generated in the enlightening discussions have become the most important essences of this thesis.

Many thanks to my colleagues at the Department of Computer and Information Science in Linköping University, and, in particular, to present and former members of the Embedded Systems Laboratory (ESLAB), for their kind help and the joy shared with me.

I acknowledge the support of the Swedish Foundation for Strategic Research (SSF) via the Strategic Integrated Electronic Systems Research (STRINGENT) program, and I appreciate the feedback and ideas obtained from many well organized workshops.

I am grateful to my parents, who have been great support all the time. Last, but not the least, I would like to express my deepest gratitude to my beloved wife, Huanfang, for her endless love, patience, and encouragement.

Zhiyuan He

Linköping, June 2007

Contents

Abstract ..... **I**

Acknowledgments ..... **III**

Chapter 1 Introduction **1**

1.1 Motivation 1

1.2 Problem Formulation 3

1.3 Contributions 4

1.4 Thesis Overview 5

Chapter 2 Background and Related Work **7**

2.1 Electronic Systems Design 7

2.2 Electronic Systems Test 10

2.3 Core-based SoC Design and Test 12

2.4 Hybrid Built-In Self-Test 15

2.5 Abort-on-First-Fail Test 17

2.6 Power- and Thermal-Aware Test 18

Chapter 3 Defect-Probability Driven SoC Test Scheduling **21**

3.1 Introduction 21

3.2 Definitions and Problem Formulation 22

3.2.1 Basic Definitions 22

3.2.2 Basic Assumptions 24

3.2.3 Possible Test Termination Moment 25

3.2.4 Expected Test Application Time 26
3.2.5 Problem Formulation 31
3.3 Proposed Heuristic 31
3.4 Experimental Results 36
3.5 Conclusions 37
Chapter 4 Power-Constrained SoC Test Scheduling 39
4.1 Introduction 39
4.2 Motivational Example 40
4.3 Problem Formulation 42
4.4 Test Set Partitioning 43
4.5 Proposed Heuristic 48
4.6 Experimental Results 53
4.7 Conclusions 55
Chapter 5 Thermal-Aware SoC Test Scheduling 57
5.1 Introduction 57
5.2 Motivational Example 58
5.3 Problem Formulation 61
5.4 Overall Solution Strategy 63
5.5 CLP-based Approach 66
5.5.1 Concepts of CLP 67
5.5.2 CLP Model 67
5.5.3 Experimental Results 70
5.6 Heuristic-based Approach 72
5.6.1 Motivational Example 73
5.6.2 Heuristic 74
5.6.3 Experimental Results 82
5.7 Conclusions 87
Chapter 6 Conclusions and Future Work 89
6.1 Conclusions 89
6.2 Future Work 90
References 93

Appendix A Abbreviations 103
Appendix B Explanations 105

Chapter 1
Introduction

This thesis deals with testing of core-based systems-on-chip (SoCs). The main purpose of this work is to reduce the test application time (TAT) and consequently reduce the testing cost. In this thesis, three techniques for core-based SoC test scheduling are presented. We first propose an SoC test scheduling technique which utilizes the defect probabilities of individual cores to guide the test scheduling. Second, we propose a power constrained SoC test scheduling technique in order to minimize the TAT and at the same time avoid high power consumption during tests. Third, we propose a thermal-aware SoC test scheduling approach which minimizes the TAT as well as avoid high temperature during tests.

In this chapter, we present the motivation of our work and formulate the problems. Thereafter, we summarize the main contributions of our work and give an overview of the thesis structure.

1.1 Motivation

The rapid advances of microelectronic technologies have enabled the design and manufacturing of highly complex systems. However, this evolution potentially leads to a dramatic increase of the system cost

due to high design complexity, long time-to-market, and high production costs.

In recent years, a core-based SoC design methodology has been employed to reduce the design and production costs by integrating pre-designed and pre-verified intellectual property (IP) cores on a single silicon die. Although the cost of designing and manufacturing SoCs is reduced, the testing cost rises because of inefficient test access mechanism (TAM), large amount of test data, and long test application times. Therefore, how to efficiently generate, transport and apply test data for core-based SoCs becomes a major challenge to test engineers.

One solution to reduce the testing cost is to reduce the TAT. With advanced design for test (DFT) techniques such as TAM and wrapper designs, the tests for individual IP cores can be applied concurrently and thus the TAT can be substantially reduced. However, reducing the TAT can be affected by power and temperature related problems.

During test, more power is dissipated than in the normal functional mode because of a substantial increase of switching activity in the circuit. The test concurrency has to be restricted due to a limited power supply. Thus, the trade-off between the TAT and the power consumption has to be taken into account. Further, high power consumption during test can cause a high level of noises occurring in the circuits and this can potentially damage the devices under test (DUTs). Moreover, high power consumption can also result in excessive heat dissipation and high temperature which also potentially damages the chips. The power and thermal issues are even more severe to the design and test of new generations of integrated circuit (ICs) which employ deep sub-micrometer technology.

Thus, advanced test scheduling techniques which reduce the TATs and at the same time take into account the power and thermal issues are strongly required for core-based SoC testing.

1.2 Problem Formulation

In this thesis, we aim to minimize the TAT of core-based SoCs and we address three test scheduling problems concerning different trade-offs and constraints. The formulations of the three problems are as follows.

The first problem that we deal with is how to minimize the TATs for high-volume production tests. More specifically, this problem is discussed in a context of testing core-based SoCs by using an abort-on-first-fail (AOFF) test approach, which means that the test process is terminated as soon as a fault has been detected. Based on the AOFF test approach, the termination of the test process is considered as a random event which happens with a certain probability. Thus, in order to minimize the TAT for a high-volume production test, we need to minimize the expected test application time (ETAT), which is calculated according to a generated test schedule and the given defect probabilities of individual cores. In particular, we employ a hybrid built-in self-test (BIST) which combines both deterministic and pseudorandom tests for an IP core. Thus, the problem is formulated as the following: given the defect probabilities of IP cores and the test sets for the hybrid BISTs, generate a test schedule such that the ETAT is minimized.

As demonstrated in the previous section, in order to shorten the TAT, concurrent testing can be employed, but the aggregate amount of power consumption has to be restricted. Thus, we address the second problem as the SoC test scheduling for the hybrid BIST in order to minimize the ETAT while keeping the aggregate amount of power consumption below a power constraint. In order to generate efficient test schedules, a test set can be partitioned into shorter sub-test sequences. In this thesis, this method is referred to as the test set partitioning (TSP). Thus, the test scheduling problem is formulated as how to generate the test schedule for all test sub-sequences such that the ETAT is minimized and the power constraint is hold.

The third problem that we tackle in this thesis is the test scheduling with limits on the temperatures of the CUTs and a limit on the bandwidth of the test-bus used for transporting test data. In order to avoid overheating the CUTs, an entire test set is partitioned into shorter test sub-sequences and cooling periods are introduced between test sub-sequences. Furthermore, the test sub-sequences partitioned from different test sets are interleaved in order to improve the efficiency of the test schedule. Thus, the test scheduling problem is formulated as how to generate test schedules for the partitioned and interleaved test sub-sequences such that the TAT is minimized while the temperature and bandwidth constraints are not violated.

1.3 Contributions

The main contributions of this thesis are as follows. First, we have proposed a defect probability driven SoC test scheduling technique based on the AOFF test approach. For this technique, we have defined the expected test application time (ETAT) as the cost function and we have proposed a heuristic to generate the test schedule with minimized ETAT. This approach assumes a test architecture designed for hybrid BISTs and the proposed technique is applicable to the testing of both combinational circuits and sequential circuits.

Second, we have proposed a power-constrained SoC test scheduling technique using test set partitioning. In order to minimize the ETAT, we have proposed heuristics for test set partitioning and test scheduling under the power constraint. The proposed technique minimizes the ETAT and also avoids the power and thermal related problems. It is applicable to both BISTs and external tests.

Third, we have proposed a thermal-aware SoC test scheduling technique using test set partitioning and interleaving. This technique assumes that a test bus is employed to transport test data, and the limit on the bandwidth of the test bus and the limits on the

temperatures of individual cores are given as constraints. In order to avoid overheating during tests, a test set is into test sub-sequences and cooling periods are introduced between consecutive test sub-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. One approach is based on the constraint logic programming (CLP) and the other approach employs a heuristic.

1.4 Thesis Overview

The rest of the thesis is constructed as follows. Chapter 2 illustrates the background and related work in the area of core-based SoC testing and design for test. The principles of electronic systems design and test, core-based SoC design and test, hybrid BIST, AOFF test, as well as power- and thermal-aware test are demonstrated.

Chapter 3 presents the first test scheduling technique, which utilizes the defect probabilities of individual cores for production test. The chapter starts with an introduction to the related work on defect-oriented test scheduling. Thereafter, the concept of the ETAT is presented and the approach to calculate the ETAT is illustrated. Based on the definition of the ETAT, a heuristic for test scheduling is presented. The chapter is concluded with experimental results demonstrating the efficiency of the proposed technique.

In Chapter 4, we present the power constrained SoC test scheduling technique. The chapter starts with an short introduction to related work followed by a motivational example which demonstrates the importance of the addressed power-constrained test scheduling problem. Thereafter, the test set partitioning technique is presented and the proposed heuristics for test set partitioning and test scheduling are illustrated. Finally, experimental results are given in order to demonstrate the feasibility and efficiency of the proposed technique.

Chapter 5 presents the thermal-aware SoC test scheduling technique. An introduction to related work is given at the beginning of the chapter and thereafter a motivational example is given to demonstrate the significance of the thermal-thermal test scheduling problem. The proposed CLP-based approach and heuristic-based approach are then illustrated in details and finally the chapter is concluded with experimental results.

The thesis is concluded in Chapter 6 where possible directions of future work are also discussed.

Chapter 2
Background and Related Work

In this chapter, the basic concepts of electronic systems design and test are presented, followed by a discussion on core-based SoC design and test. Thereafter, the background and related work on hybrid BIST, AOFF test, and power- and thermal-aware test are demonstrated.

2.1 Electronic Systems Design

In order to manage the design complexity of modern electronic systems, the electronic systems design has to be organized in a hierarchical approach which covers several levels of abstraction. Usually, the abstraction levels are referred to as system level, register-transfer (RT) level, logic level, circuit level, and physical level, from higher to lower levels respectively. Figure 2.1 illustrates the generic structure of the electronic systems design space, where the five hierarchical abstraction levels are categorized into three domains [Gaj83].

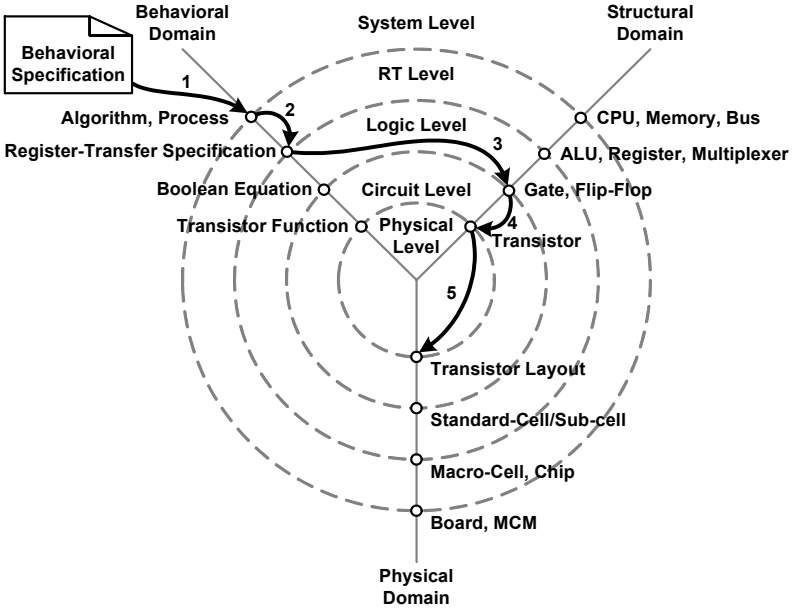


Figure 2.1: Design space of electronic systems [Gaj83]

In principle, the design space can be classified into three different domains, according to the perspective from which different designers look on their design tasks. As depicted in Figure 2.1, the three design domains are the behavioral domain, the structural domain, and the physical domain. In different domains, designers look at their design tasks in different perspective, as listed in Table 2.1. A design flow [Dev94] of electronic systems is also depicted in Figure 2.1 (see the arrows marked with numbers) and it is extended with details in Figure 2.2.

Table 2.1: Design tasks from different perspective

Abs. Level	Behavioral Domain	Structural Domain	Physical Domain
System Level	Algorithm, Process	CPU, Memory, Bus	Board, MCM, SoC
RT level	RT Specification	ALU, Register, MUX	Macro-Cell, Chip
Logic Level	Boolean Equation	Gate, Flip-Flop	Standard-Cell/Sub-Cell
Circuit Level	Transistor Function	Transistor	Transistor Layout

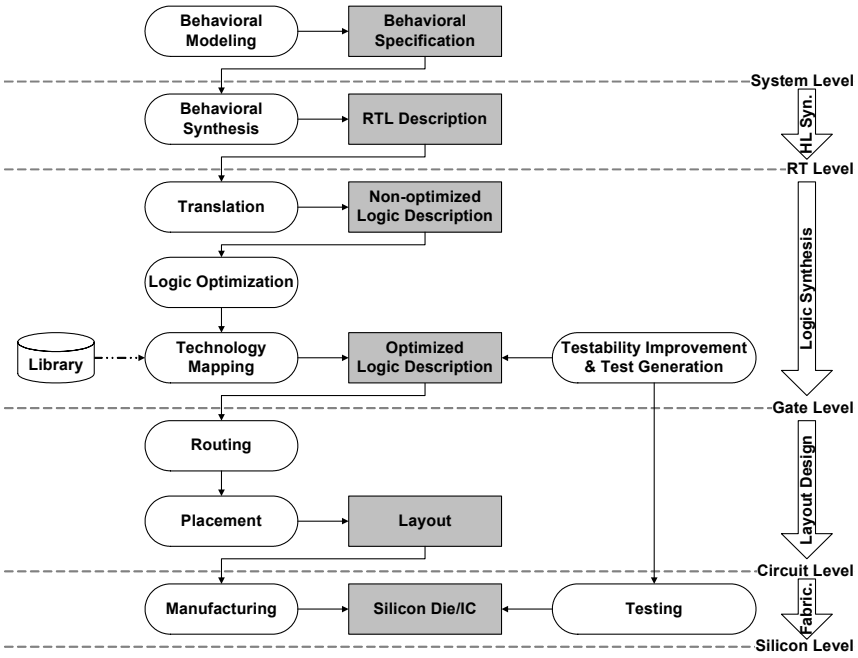


Figure 2.2: Generic design flow of electronic systems [Dev94]

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 the behavioral domain to the structural domain. Each step in the design flow is explained as follows, where the bullet numbers correspond to the numbers marked on the arrows in Figure 2.1 [Gaj83].

(1) Behavioral Modeling: Also called system-level specification. The specification of a system is usually given as a description to the functionality of the system and a set of design constrains. In this phase, the system specification is analyzed and a behavioral description is written in a hardware description language or natural language.

(2) High-Level Synthesis: Also called behavioral synthesis [Ell99]. In this phase the system-level specification is transformed into a

description of RT-level (RTL) components such as ALUs, registers, and multiplexers. The basic components in an RTL design usually correspond to operations in a behavioral specification. In order to obtain the RTL design, the high-level synthesis usually consists of the following steps [Ell99]: derivation of control/data-flow graph (CDFG), operation scheduling, resource allocation and binding, derivation of RTL data-path structure, and description of a controller which can be a finite state machine (FSM).

(3) Logic Synthesis [Dev94]: Also called gate-level synthesis. In this phase, an RTL design is translated into a set of logic functions. Thereafter, the translated RTL design is optimized according to different requirements given by the designer and then mapped into a netlist of logic gates, using a technology library provided by a vendor.

(4) Circuit Design: This step takes the optimized logic description as an input, and generates the transistor implementations of the circuit.

(5) Layout Design: In this phase, the circuits are mapped onto the silicon implementation with a certain layout and placement design.

As illustrated in Figure 2.2 [Dev94], when the logic netlist has been obtained, the testability improvement and test generation are done by a set of tools. However, the testability improvement and test generation at higher abstraction levels can be realized by using the state-of-the-art DFT and test generation (TG) techniques. After the chips are manufactured, they have to be tested by applying the acquired test package. After test, only the qualified products are delivered to customers.

2.2 Electronic Systems Test

Testing of a electronic system is an experiment in which the system is exercised and its resulting response is analyzed to ascertain whether it behaved correctly [Abr94]. In this thesis, an instance of an incorrect operation of the system being tested is referred to as an error [Abr94]. The errors can be further classified as design errors,

fabrication errors, fabrication defects, and physical failures, according the causes of the errors. In this thesis, testing targets fabrication defects. The different types of error are defined as follows [Abr94].

Design errors are usually incomplete or inconsistent specifications, incorrect mapping between different levels of design, violations of design rules, etc. Fabrication errors include wrong components, incorrect writing, shorts caused by improper soldering, etc. Fabrication defects are not directly attributed to human errors, rather, they result from an imperfect manufacturing process. Examples of common fabrication defects are shorts and opens in MOS 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. For example, aluminum connectors inside an IC package thin out with time and may break because of electron migration or corrosion. Environmental factors, such as temperature, humidity, and vibrations, accelerate the aging of components. Cosmic radiation and particles may induce failures in chips containing high-density random-access memories (RAMs). Some physical failures, referred to as “infancy failures”, appear early after fabrication.

Fabrication errors, fabrication defects, and physical failures are collectively referred to as physical faults [Abr94]. According to their stability in time, physical faults can be classified as follows: (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 some environmental factor.

In general, a direct mathematical treatment of testing and diagnosis is not applicable to physical faults [Abr94]. 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 [Abr94]. 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 represents that a single wire being permanently "stuck" at the logic one or logic zero.

2.3 Core-based SoC Design and Test

Design and manufacturing of integrated circuits have moved into the deep submicron technology regime. Scaling of process technology has enabled a dramatic increase of the integration density, which enables more and more functionalities to be integrated into a single chip. With the improving system performance, the design complexity has also been increasing 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 [Mur96], [Zor98], 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 intellectual property (IP) cores that are provided by different vendors. It integrates the IP cores into the system and manufactures the system on a single silicon die. An example of an SoC design is shown in Figure 2.3. It consists of several cores with different functionalities and a user-defined logic (UDL), which are represented by rectangular blocks. The cores are usually processors (Microcontroller, DSP, etc.), memory blocks (ROM, RAM, EEPROM, Flash Memory, etc.), bus structure, peripherals interfaces (USB, FireWire, Ethernet, DMA, etc.), analog circuits (PWM, A/D-D/A, RF, etc.), and so on. The UDL components are used to glue the cores for the intended system.

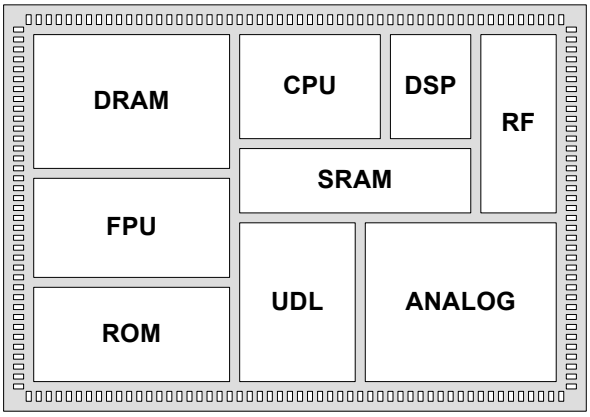


Figure 2.3: An example of core-based SoC design

In order to test individual cores in a SoC [Mur96], [Zor98], a test architecture consisting of certain resources has to be available. The test architecture for SoCs usually includes a test source, a test sink, and a test access mechanism (TAM). Figure 2.4 shows a typical example of an SoC test architecture.

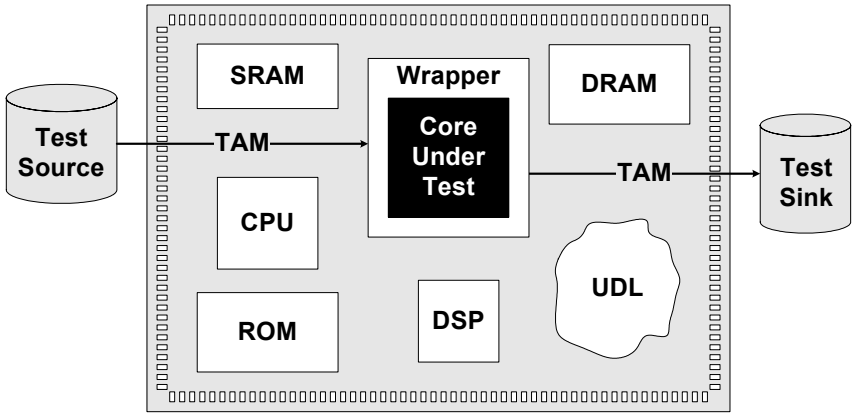


Figure 2.4. An example of SoC test architecture

A test source is a test-pattern provider which can be either external or on-chip. A typical external test source is an automated

test equipment (ATE) which generates test patterns and stores them in its local memory. An on-chip test source can be a linear feedback shift register (LFSR), a counter, or a ROM/RAM which stores already generated test patterns.

A test sink is a test response/signature analyzer that detects faults by comparing test responses/signatures with the correct ones. An ATE can be an external test sink that analyzes the test responses/signatures transported from the cores under test (CUTs). The test sink can also be integrated on the chip so that the test responses/signatures can be analyzed on-the-fly.

A TAM is an infrastructure designed for test data transportation. It is often used to transport test patterns from the test source to CUTs and to transport test responses/signatures from CUTs to the test sink. A common design of the TAM can be a reusable functional bus infrastructure [Har99], such as the advanced microprocessor bus architecture (AMBA) [Fly97], or a dedicated test bus. A wrapper [Mar00] is a thin shell which surrounds a CUT in order to enable the switching between different test modes such as functional, internal, external test modes, etc. The TAM and the wrappers comprise a test access infrastructure for the CUTs of an SoC.

An example of test architecture for external SoC test is depicted in Figure 2.5. In this example, a system of four cores is to be tested. An ATE consisting of a test controller and a local memory serves as an external tester. The generated 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. When test patterns have been activated, the captured test responses are also transported to the ATE through the test bus. The external ATE can be replaced by an embedded tester which is integrated on the chip. The same test architecture is applicable for the system using an embedded tester, as illustrated in Figure 2.6.

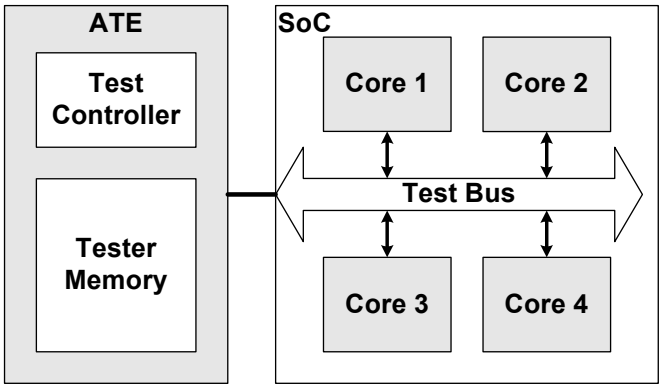


Figure 2.5: An example of test architecture for external test using an ATE

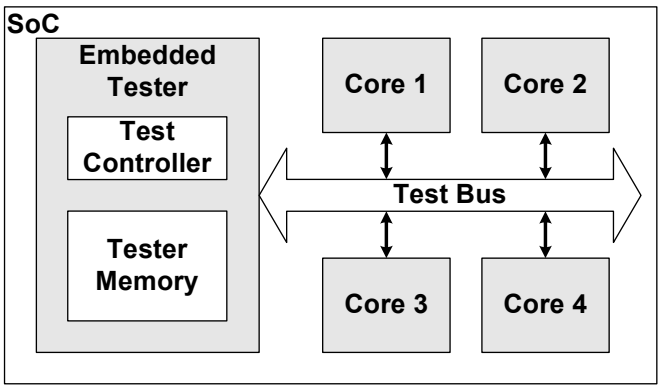


Figure 2.6: An example of test architecture for external test using an embedded tester

2.4 Hybrid Built-In Self-Test

As the number of cores on a chip has been increasing along with the rapid advances of technology, the amount of required test data for SoC testing is growing dramatically. This demands a large quantity of memory to be used in an ATE, if an external test is employed.

Moreover, an external test is usually applied at relatively low speed due to the limited bandwidth of the bus used to transport the test data. Thus, a long test application time is required.

One of the solutions to this problem is to use built-in self-test (BIST), which generates pseudorandom test patterns and compact test responses within the chip. Although BIST can be applied at high speed, it is considered less efficient than external test, regarding the fault coverage and test-sequence length. Due to the existence of random-pattern-resistant faults, BIST usually needs larger amount of test patterns in order to reach a certain level of fault coverage.

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 [Hel92], [Tou95], [Sug00], [Jer00], [Jer03]. 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, while providing higher fault coverage and requires less amount of test data compared to the stand-alone BIST solution.

An example of a test architecture for hybrid BIST is depicted in Figure 2.7. In this example, a system consisting of four cores is to be tested. 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 logic that can generate and apply pseudorandom test patterns on-the-fly. We assume that the test controller is capable of controlling the process of both deterministic and pseudorandom tests according to the test schedule, meaning that it controls the times when the tests should be started, stopped, restarted, and terminated.

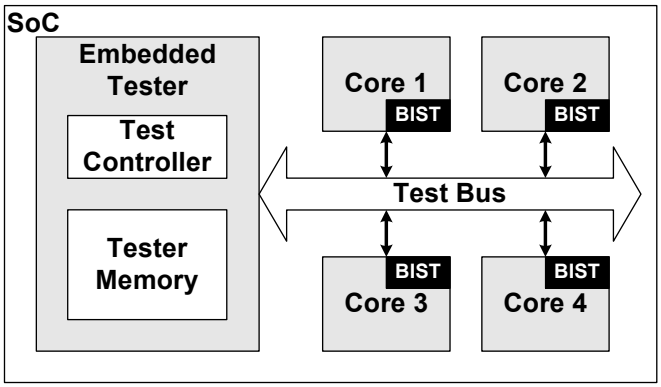


Figure 2.7. An example of test architecture for hybrid BIST

In order to reduce the testing cost, core-based SoC test has received a wide variety of research interests [Mur96], [Cho97], [Aer98], [Var98], [Zor98], [Cha00], [Mur00], [Nic00], [Rav00], [Hua01], [Iye01], [Cot02], [Iye02], [Lar02], [Goe03], [Iye03], [Lar04b], [He06a] concerning advanced test architecture design , test resource allocation, and test scheduling.

2.5 Abort-on-First-Fail Test

Many proposed SoC test scheduling techniques assume that tests are applied to the completion [Hus91], [Mil94], [Kor02]. However, high-volume production testing often employs an AOFF approach in which the test process is aborted as soon as a fault has been detected. The defected devices can be discarded directly or further diagnosed in order to find out the cause of the faults. Using the AOFF approach can lead to a substantial reduction of TAT, since a test does not have to be completed if faults are detected. The test cost can be reduced in terms of the reduced TAT. AOFF test is important to the early-stage of a 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 IP cores can be used for test scheduling in

order to generate efficient test schedules [Jia01], [Lar04a]. The defect probabilities of IP cores can be derived from statistical analysis of production processes or generated from inductive fault analysis.

2.6 Power- and Thermal-Aware Test

Production of integrated circuits has moved into the deep submicron technology regime. Scaling of process technology has enabled dramatically increasing the number of transistors, and therefore improving the performance of electronic chips. However, the rapid growth of integration density has posed critical challenges to the design and test of electronic systems, one of which is the power and thermal issue [Bor99], [Gun01], [Mah02], [Ska04].

It is known that more power is consumed during testing than in normal functional mode [Zor93], [Pou00], [Shi04] and the circuits are therefore more stressed from the power consumption perspective. This is due to a larger amounts of switching activity caused by applying test patterns. High power dissipation results in several critical problems, one of which is the insufficient driving current due to a limited power supply. As a consequence, the circuit can become unreliable. Excessive power dissipation can cause ground noises which can damage the DUT. High power dissipation may also lead to high junction temperature which has large impacts on the integrated circuits [Vas06].

The performance of the integrated circuits is proportional to the driving current of CMOS transistors, which is a function of the carrier mobility. Increasing junction temperature decreases the carrier mobility and the driving current of the CMOS transistors, which consequently degrades the performance of circuits.

In higher junction temperature, the leakage power increases. The increased leakage power in turn contributes to an increase of junction temperature. This positive feedback between leakage power

and junction temperature may result in thermal runaway and destroy the chip due to an excessive heat dissipation.

The long term reliability and lifespan of integrated circuits also strongly depends on junction temperature. Failure mechanisms in CMOS integrated circuits, such as gate oxide breakdown and electro-migration, are accelerated in high junction temperature. This may results in a drop of the long term reliability and lifespan of circuits.

In order to prevent excessive power during test, some techniques have been explored. Low power test synthesis and design for test targeting RTL structures is one of the solutions, for example, low-power scan chain design [Ger99], [Ros04], [Sax01], scan cell and test pattern reordering [Flo99], [Gir98], [Ros02]. Although low power DFT can reduce the power consumption, this technique usually adds extra hardware into the design and therefore it can increase the delay and the cost of every single chip. Power-constrained test scheduling which targets system-level DFT is another approach to tackle the problem [Cho97], [Cha00], [Iye02], [Lar04b], [Mur00], [Nic00], [Rav00]. It reduces the test application time while keeping the power consumption below a given power constraint so that the circuits can work in a common condition.

Advanced cooling system can be one solution to the high temperature problems. However, the cost of the entire system has to face a substantial rise, and the size of the system is inevitably large. In order to test new generations of SoCs safely and efficiently, novel and advanced power and thermal management techniques are required.

Chapter 3
Defect-Probability Driven SoC Test Scheduling

In this chapter, a test scheduling technique based on the AOFF approach is proposed for hybrid BIST. Defect probabilities of individual cores are used to calculate ETAT and a heuristic is proposed to minimize the ETAT.

3.1 Introduction

In [Jia01], a defect-oriented test scheduling approach was proposed to reduce the test times. Based on the defined cost-performance index, a sorting heuristic was developed to obtain the best testing order. In [Lar04a], a more accurate cost function using defect probabilities of individual cores was proposed. Based on the proposed cost function, a heuristic was also proposed to minimize the ETAT.

In this chapter, we propose an approach to calculate the probability of a test process to be aborted at a certain moment when a test pattern has been applied and the test response/signature has been available [He04], [He05]. A heuristic [He04] is also proposed to minimize the ETAT.

3.2 Definitions and Problem Formulation

3.2.1 Basic Definitions

In this chapter, we employ the test architecture (see Figure 2.7) for hybrid BIST, in which all cores have their dedicated BIST logic and a test bus is used to transport deterministic test data from/to the embedded tester. Based on this test architecture, we assume that the pseudorandom test patterns for different cores can be concurrently applied, while the deterministic test patterns can only be applied sequentially. Figure 3.1 depicts a hybrid BIST test schedule for a system consisting of five cores, where TS_i denotes the test set (TS) for core C_i (i = 1, 2, ... , 5). The white and grey rectangles represent the deterministic test sub-sequences (DTSs) and the pseudorandom test sub-sequences (PTSs), respectively. As illustrated in this example, deterministic test patterns are applied sequentially, while pseudorandom test patterns for different cores are applied in parallel. The test application time is 390, which is the longest test time among the five.

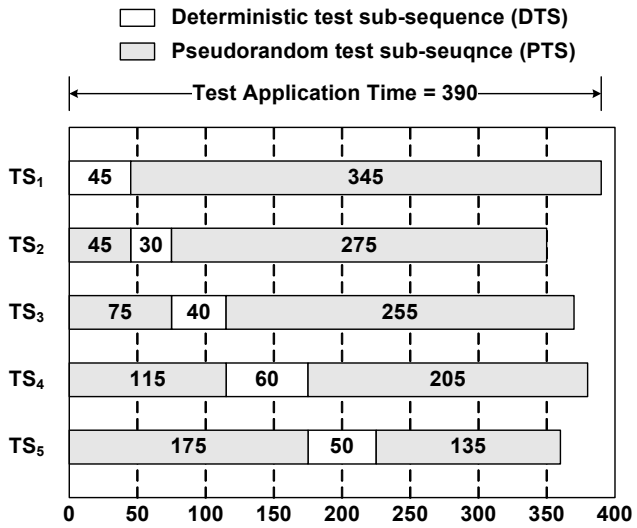


Figure 3.1: A test schedule of a hybrid BIST for five cores

Suppose that a system S, composed of n cores, C_1, C_2, ... , C_n employs a test architecture illustrated in Figure 2.7. In order to test a core, a set of test patterns are generated, usually referred to as test set or test sequence (TS). A test set can consist of deterministic test patterns (DTPs) and pseudorandom test patterns (PTPs). A subset of deterministic test patterns is referred to as a deterministic test sub-sequence (DTS), and a subset of pseudorandom test patterns is referred to as a pseudorandom test sub-sequence (PTS). For each individual core C_i ($1 \le i \le n$), the generated test set/test sequence, the deterministic test sub-sequence, and the pseudorandom test sub-sequence are denoted with TS_i, DTS_i, and PTS_i, respectively. In the cases that more than one deterministic test sub-sequence or pseudorandom test sub-sequence is partitioned from the original test set, DTS_{iv} and PTS_{iw} respectively denotes the v-th deterministic test sub-sequence and the w-th pseudorandom test sub-sequence of TS_i. Suppose that the number of deterministic test patterns and pseudorandom test patterns in TS_i is 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 the 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 a SoC, in short, system defect probability (SDP), is defined as the probability of the SoC having defects, meaning that some cores are defected. We assume that the defect probabilities of different cores in a SoC are independent. Then, the SDP is given by

$$SDP = 1 - \prod_{i=1}^{n} \left(1 - CDP_i\right) \tag{3.1}$$

We suppose that a test process can be terminated with a certain probability. The probability of the test process being aborted at a certain moment depends on the probability of an individual test being aborted due to the detection of faults, referred to as the

individual test failure probability (ITFP), and the probability of an individual test being passed with no faults detected, referred to as the individual test success probability (ITSP).

3.2.2 Basic Assumptions

We assume that the failure probabilities of individual tests (ITFPs) for IP 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 probability of individual tests (ITSPs) for IP 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 is contiguously applied. This means that such a scenario will not appear that a deterministic test is stopped at a certain moment and is restarted after the application of a pseudorandom test sub-sequence for the same core.

On the other hand, we assume that the application of a pseudorandom test can be stopped and restarted later when the deterministic test for the same core has been finished. This is because that pseudorandom tests are usually very long while dividing it into shorter test sub-sequences allows analyzing signatures more frequently. However, frequent switching between deterministic and pseudorandom tests for a core introduces overheads [Goe03]. Since we only stop a pseudorandom test at most once, very few overheads will be introduced and therefore are ignored.

Further more, in this chapter, we schedule the deterministic tests for different cores 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, test effects can be observed at each test application cycle, which provides higher frequency on checking possibilities of test termination and thus can

shorten the test application time. Thus, it does not need to delay any deterministic test in order to insert a pseudorandom test.

3.2.3 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 sub-sequence, the test response is captured as soon as a test pattern has been applied. By analyzing the obtained test response, the test can be aborted immediately, if faults are detected. On the other hand, during the application of a pseudorandom test sub-sequence, the signature is not available until all the pseudorandom test patterns in the sub-sequence have been applied. By analyzing the obtained signature, the test can be aborted, if faults are detected. Therefore, using the AOFF approach, a test is possible to be terminated at every cycle of deterministic test applications, or at the end of contiguous pseudorandom test applications. This analysis leads to the notion of possible termination moment (PTTM).

A PTTM is a time moment when the test process can be terminated due to a detection of faults. As demonstrated previously, a PTTM is the time moment immediately after a deterministic test pattern/pseudorandom test sub-sequence has been applied and the test response/signature has been analyzed.

For a given test schedule, all PTTMs are fixed and easy to obtain. Figure 3.2 gives an example to illustrate PTTMs in a test schedule for a SoC with five cores. In this example, deterministic test patterns are depicted with white rectangles and pseudorandom test sub-sequences are depicted with grey rectangles. The dashed lines in gray indicate the PTTMs when each DTP has been applied, e.g. PTTMs 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10. The dotted lines in black indicate the PTTMs when each 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.

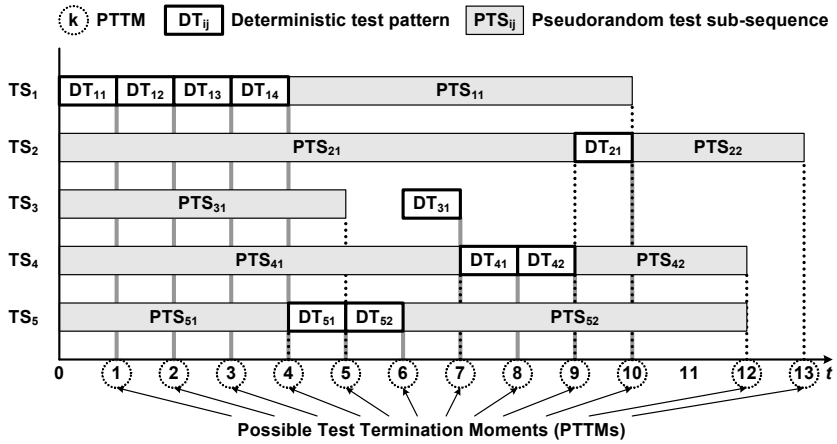


Figure 3.2: Possible test termination moments (PTTMs)

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

3.2.4 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. Therefore, the test application time (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 at completion. Then, the ETAT is given by

$$ETAT = \sum_{\forall x \in X} \left(t_x \times p\left[A_x \right] \right) + L \times p\left[T \right] \tag{3.2}$$

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

In Equation (3.2), the ETAT is presented as a sum of two literals. The first literal corresponds to the situations 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 at completion without detection of any faults. Indeed, Equation (3.2) 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 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 the 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, and (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

$$A_x = \left(\bigcup_{\forall y \in Y_x} F_x(y) \right) \cap \left(\bigcap_{\forall z \in Z_x} P_x(z) \right) \tag{3.3}$$

Figure 3.3 gives a example to explain the situation when the test process is aborted at PTTM 7. This means that, at the 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.

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

$$T = \bigcap_{\forall e \in E} P(e) \tag{3.4}$$

According to the definition of PTTM, at PTTM x, Y_x should not be empty and at least one element in 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 detect any faults so far, otherwise the test process would have been aborted earlier and would not have reached PTTM x. Table 3.1 lists the sets Y_x and Z_x at every PTTM x with respect to the example depicted in Figure 3.2.

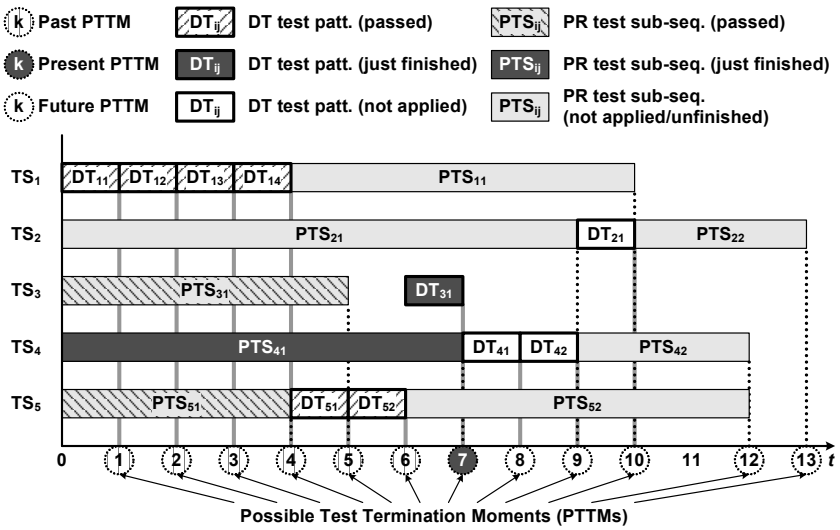


Figure 3.3: An example illustrating the situation when the test process is aborted at PTTM 7

Table 3.1: Y_x and Z_x at each PTTM x w.r.t. Figure 3.2

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	$\{TS_2\}$	$\{TS_1, TS_3, TS_4, TS_5\}$

The set E includes all the individual tests. For the example depicted in Figure 3.2, $E = \{TS_1, TS_2, TS_3, TS_4, TS_5\}$.

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

$$\begin{aligned} p[A_x] &= p\left[\bigcup_{\forall y \in Y_x} F_x(y)\right] \times \prod_{\forall z \in Z_x} p[P_x(z)] \\ &= \left(1 - \prod_{\forall y \in Y_x} (1 - p[F_x(y)])\right) \times \prod_{\forall z \in Z_x} p[P_x(z)] \end{aligned} \tag{3.5}$$

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

$$p[T] = p\left[\bigcap_{\forall e \in E} P(e)\right] = \prod_{i=1}^{n} (1 - CDP_i) \tag{3.6}$$

Thus, the ETAT is represented as

$$\begin{aligned} ETAT = &\sum_{\forall x \in X} \left(t_x \times \left(1 - \prod_{\forall y \in Y_x} (1 - p[F_x(y)])\right) \times \prod_{\forall z \in Z_x} p[P_x(z)] \right) \\ &+ L \times \prod_{i=1}^{n} (1 - CDP_i) \end{aligned} \tag{3.7}$$

where x is a PTTM, X is the set of all PTTMs, t_x is the test application time by the moment x, L is the test application time 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

$$p[F_x(y)] = IFC(v) \times CDP_i \tag{3.8}$$

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

$$p[P_x(z)] = 1 - CDP_i \times \sum_{j=1}^{m} IFC(v_j) \tag{3.9}$$

More details on how Equation (3.8) and Equation (3.9) are obtained can be found in Appendix B.

3.2.5 Problem Formulation

Thus, the ETAT has been completely formulated. Out objective is to generate an efficient test schedule with the minimized ETAT. We have proposed a heuristic that employs ETAT as the cost function to find a near-optimal solution, as presented in the following section.

3.3 Proposed Heuristic

The proposed heuristic is an iterative algorithm that generates a test schedule with a minimized ETAT. As demonstrated earlier, the test scheduling problem in the hybrid BIST and the AOFF context is essentially to schedule deterministic test sub-sequences efficiently,

as they are more efficient from both the test termination and the fault coverage perspectives.

By changing the schedule of deterministic test sub-sequences, the incremental fault coverage of test patterns, the failed sets and the passed sets are also changed, and therefore the failing probabilities, the passing probabilities, and ultimately the ETAT alternate.

It is natural to give an order to the deterministic test sub-sequences such that the cores with higher defect probabilities are scheduled for deterministic test earlier. However, such a solution does not necessarily lead to the minimal ETAT. In addition to the defect probabilities of cores, more factors such as the efficiency of test patterns and the length of individual test sub-sequences have to be taken into account. We address the ETAT minimization problem as a combinatorial problem. Due to the problem complexity, we propose a heuristic in order to solve it efficiently.

The proposed heuristic is an iterative algorithm. We construct two sets of deterministic test sub-sequences (DTSs) in the heuristic, namely the scheduled set S and the unscheduled set U. The scheduled set S is an ordered set which is supposed to include all DTSs when the algorithm is terminated. The DTSs in S are associated with a particular order O according to which the DTSs should be scheduled so that the ETAT of the generated test schedule is the minimum. The unscheduled set U is a complement set of S, with regard to the complete set of all DTSs, meaning that U always include the still unscheduled DTSs during any iteration of the heuristic.

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

Suppose that at one iteration step, S consists of m $(0 \leq m < n)$ scheduled DTSs. The objective at this iteration step is to schedule

one more DTS from U to S, so that is S enlarged to (m + 1) DTSs. Since there are (n – m) candidate DTSs in U for selection and there are (m + 1) alternative position in S for insertion, there are in total (n – m) × (m + 1) different solutions for exploration.

In order to illustrate how to explore and decide on alternative solutions, an example is given in Figure 3.4. In this example, we assume that there are five hybrid test sets in total (n = 5) and two have been temporarily scheduled through previous iteration steps (m = 2). From the depicted partial test schedule at this iteration step, we can see 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 $INSPOS_3$, indicated by the three short arrows. The heuristic explores all the nine alternative solutions each of which is identified by the pair (DTS_i, $INSPOS_j$). With each solution, the currently unscheduled DTS selected from U is 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 time. If a PTS is longer than the period reserved before the scheduled DTS for the same core starts, this PTS has to be stopped right before the DTS starts and restarted right after the DTS has been 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.

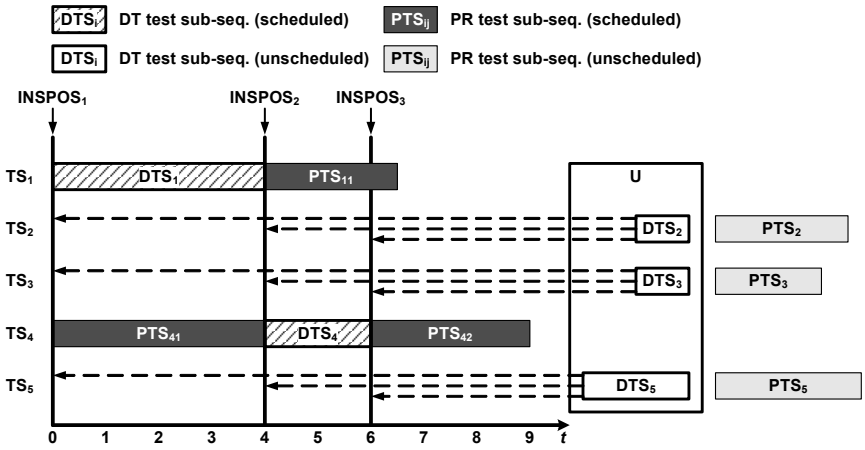


Figure 3.4: Alternative solutions

Figure 3.5 shows a test schedule assuming (DTS_3, $INSPOS_2$) has been 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.

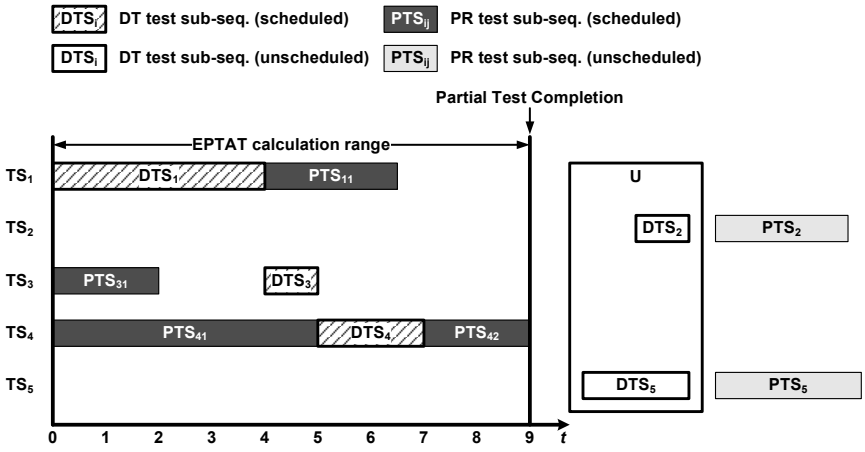


Figure 3.5: Partial test schedule for the best solution

The pseudo-code of the heuristic is given in Figure 3.6. Line 1 initializes S with an empty set and line 2 initializes U with the

complete test set. Lines 3 to 19 are three nested loops that generate the test schedule. The outer loop (lines 3 to 19) moves one unscheduled DTS from U and inserts it into S (lines 17 to 18). The DTS to be moved from U is decided within the middle loop (lines 6 to 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 to 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 to 14). The algorithm returns the generated test schedule with the minimal ETAT (line 20), when all the DTS in U have been moved into S.

```
1:  S := ∅;
2:  U := {DTS1, DTS2, ... , DTSn};
3:  while (U ≠ ∅) loop    /* outer loop */
4:     Reset(EPTATmin);
5:     IPS := GetInsPosSet(S);
6:     for (∀ DTS ∈ U) loop    /* middle loop */
7:        for (∀ InsPos ∈ IPS) loop    /* inner loop */
8:           PartSchedcur := GenPartSched(S, DTS, InsPos);
9:           EPTATcur := CalcETAT(PartSchedcur);
10:          if (EPTATcur < EPTATmin) then
11:             EPTATmin := EPTATcur;
12:             DTSsel := DTS;
13:             InsPossel := InsPos;
14:          end if
15:       end for
16:    end for
17:    Insert(S, DTSsel, InsPossel);
18:    Remove(U, DTSsel);
19: end while
20: Return( GenFullSched(S) );
```

Figure 3.6: Pseudo-code of the proposed heuristic

The proposed heuristic has a polynomial time complexity of $O(kn^4)$, where n is the number of cores and k is the average number of deterministic test patterns generated for a core.

3.4 Experimental Results

We have done experiments for different designs of various numbers of cores. Designs with 5, 7, 10, 12, 15, 17, 20, 30, and 50 cores selected from the ISCAS'85 benchmark are used for our experiments. For each design of a particular number of cores, five different hybrid test sets are generated in order to test the chip. The hybrid test sets for a design are different in terms of different numbers of generated deterministic test patterns and pseudorandom test patterns consisting of the test sets. The defect probabilities of individual cores are randomly generated and allocated such that the defect probability of the SoC equals 0.6 (40% system yield). The experimental results are listed in Table 3.2, which lists the average values of five different experiments for each design.

Table 3.2: Experimental results

#cores	Random Scheduling		Our Heuristic		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 demonstrate the efficiency of the proposed test scheduling technique on the reduction of the ETAT, we have done another set of experiments for comparison, in which a random scheduling algorithm is employed. The ETATs of the generated test schedules by using the random scheduling and our heuristic are listed in columns 2 and 4, respectively. It can be seen that the ETATs of the test schedules generated by our heuristic are 5% to 15% shorter than those produced by the random scheduling.

As our heuristic can produce only a near optimal solution, experiments for estimating the accuracy of our solutions have also been performed. For this purpose, scheduling algorithms based on a simulated annealing strategy and exhaustive search are employed for comparison, if available. The ETATs of the generated test schedules by using the simulated annealing algorithm and exhaustive search are listed in columns 6 and 8, respectively. The CPU times of all the four approaches are listed in columns 3, 5, 7, and 9. By comparing the experimental results, it is clear that our heuristic is able to generate test schedules with ETATs very close to those produced by the simulated annealing algorithm and exhaustive search, while at the same time our heuristic has substantially shorter computation times. These comparisons on ETATs and CPU times are also illustrated in Figure 3.7 and Figure 3.8.

3.5 Conclusions

In this chapter, a test scheduling technique in the context of a hybrid BIST has been presented. In this technique, the AOFF approach is employed and the defect probabilities of individual cores are taken into account. A method to estimating the test application time is proposed, and based on the proposed method a heuristic is developed to minimize the expected test application times by generating efficient test schedules. Experimental results have shown the efficiency of the proposed technique.

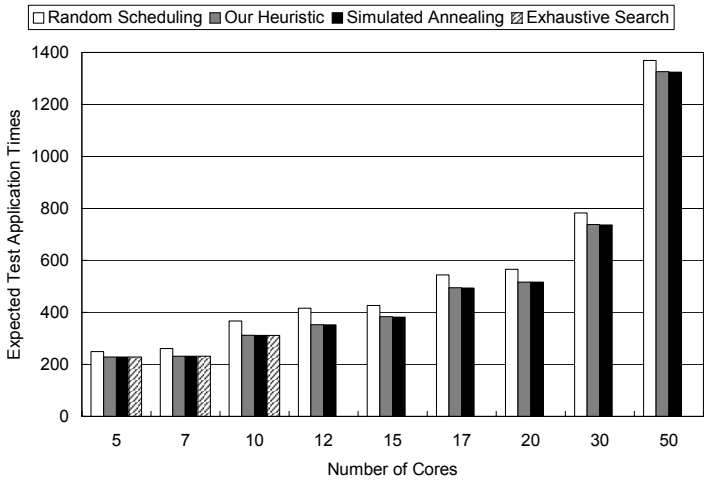


Figure 3.7: Comparison of ETATs of different approaches

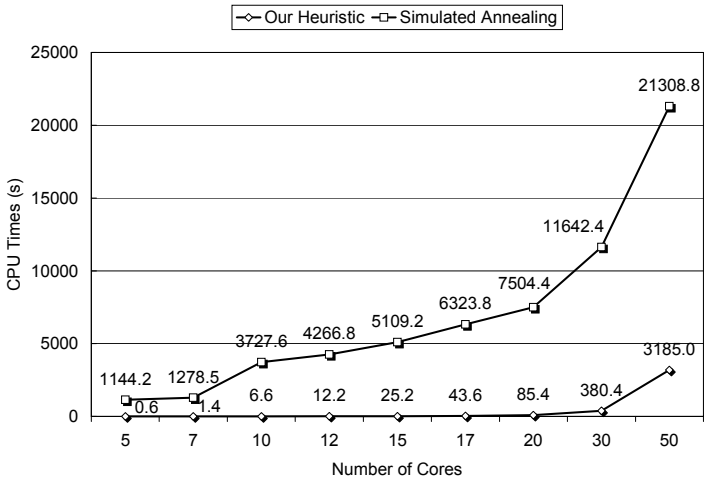


Figure 3.8: Comparison of CPU times of different approaches

Chapter 4
Power-Constrained SoC Test Scheduling

In the previous chapter, we employ the AOFF-based defect-probability driven test scheduling approach to minimize the ETAT. In that approach, the BISTs of all the IP cores can be applied concurrently. However, testing large number of core in parallel can result in power and thermal related problems. Thus, in this chapter, we present a power constrained SoC test scheduling approach which employs the test set partitioning to minimize the ETAT.

4.1 Introduction

The power-constrained test scheduling problem can be considered as a two-dimensional rectangular packing (RP) problem [Bak80], [Dyc90], [Dyc97], [Les04], [Les05], [Kor03], [Kor04] which is NP-complete. In this thesis, a test sub-sequence composed of test patterns is considered as a rectangle, with the height corresponding to the maximum power consumption of the test patterns and the width the time duration of the test sub-sequence.

In this chapter, we present a power constrained and defect-probability driven test scheduling approach to minimize the TAT. The probability of a test termination is calculated according to the results of fault simulation and the given defect probabilities of the

CUTs. A test set partitioning technique which divides a test set into shorter test sub-sequences integrated in the test scheduling heuristic, in order to improve the efficiency of the test scheduling algorithm.

The definitions in Chapter 3.2 and test architecture depicted in Figure 2.7 are also used for this chapter. We suppose that a deterministic test set is partitioned into a_i $(0 \le a_i \le d_i,\ 1 \le i \le n)$ deterministic test sub-sequences, and a pseudo-random test set is partitioned into b_i $(0 \le b_i \le r_i,\ 1 \le i \le n)$ pseudorandom test sub-sequences, where $a_i + b_i > 0$. Note that a test set is also a test sub-sequence which originally has one partition ($a_i = d_i = 1$ and/or $b_i = r_i = 1$). Thus, the term "test sub-sequence" is used to represent a test set as well, if not mentioned otherwise.

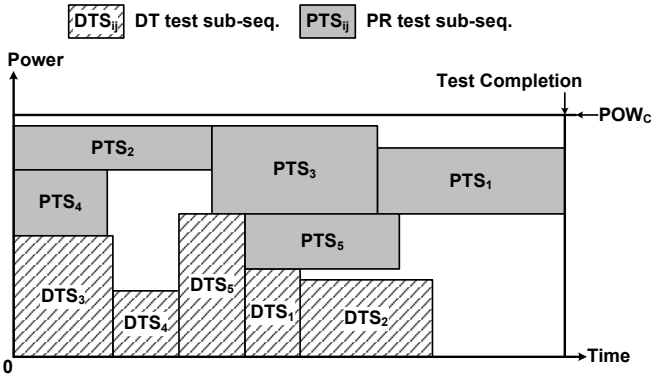
We also employ a test pattern reordering technique proposed in [Ros02] as a pre-processing step for the test set partitioning. Test pattern reordering can reduce the power consumption of test patterns and make the power profiles smoother. For the test set partitioning, we have developed a heuristic to find an appropriate number of partitions such that the sum of the area sizes (the peak-power consumption multiplied by the time duration) of all the partitions is as small as possible.

4.2 Motivational Example

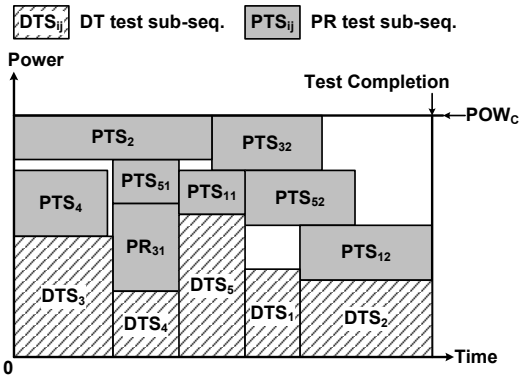
We assume that the power consumption in the circuit by applying a test pattern is proportional to the total number of transitions between this test pattern and the preceding test pattern, occurring at all the primary inputs, primary outputs, and internal nodes. The peak-power consumption by applying a test sub-sequence is defined as the maximum power consumed by applying each of the test patterns belonging to the test sub-sequence (see Figure 4.4).

Different test schedules can have a variety of test application times. Figure 4.1(a) shows an example of a power-constrained test schedule for five deterministic test sub-sequences (DTS) and five

pseudorandom test sub-sequences (PTS), which are illustrated with white and grey rectangles, respectively. Each test sub-sequence is depicted as a rectangle with a height and a width corresponding to the peak-power consumption and the time duration of the test sub-sequence, respectively. The area size of a test sub-sequence is then equal to its 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 sub-sequences belonging to the same core, such as DTS_1 and PTS_1, cannot be scheduled in parallel due to the test conflict.



(a) A test schedule without test set partitioning



(b) A test schedule with test set portioning

Figure 4.1: Power-constrained test schedule w/o vs. with TSP

Comparing the size of the effective scheduled area occupied by all test sub-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 that the efficiency of the test schedule in Figure 4.1(a) is low since much of the space is wasted. One solution to improve the efficiency of the test schedule is to employ test set partitioning to decrease the granularities of test sub-sequences. As shown in Figure 4.1(b), PTS_1, PTS_3, and PTS_5 are partitioned into PTS_{11} and PTS_{12}, PTS_{31} and PTS_{32}, and PTS_{51} and PTS_{52}, respectively. The partitioned test sub-sequences have a shorter time duration and/or a smaller peak-power consumption than the non-partitioned ones, thus can be scheduled at time moments which were not possible for the non-partitioned test sub-sequence due to its large area size. From this example, it can be observed that using test set partitioning can significantly improve the efficiency of the test schedule and shorten the test application time.

4.3 Problem Formulation

We use the same definitions as given in chapter 3.2. Figure 4.2 depicts all the PTTMs in a power-constrained test schedule, where the dotted lines indicate the ending moments of single deterministic test patterns, and the dashed lines indicate the ending moments of pseudorandom test sub-sequences. Overlapped time moments are treated as identical PTTMs.

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

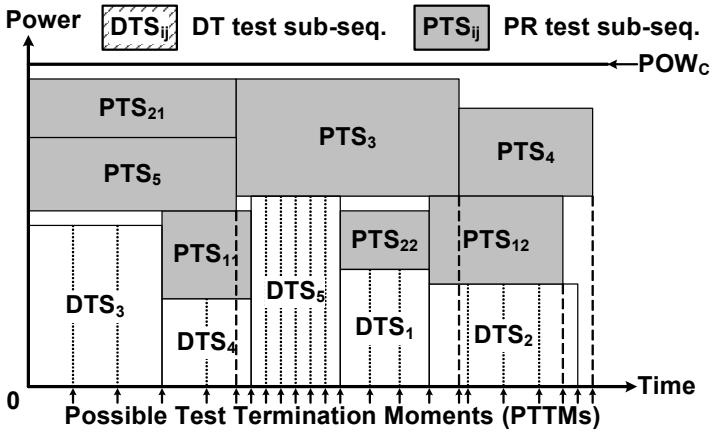


Figure 4.2: PTTMs in a power-constrained test schedule

4.4 Test Set Partitioning

As in a rectangular packing problem, the sizes of test sub-sequences have a large impact on the final schedule. To divide test sub-sequences into smaller partitions with shorter time duration and lower individual peak-power consumptions will lead to more efficient test scheduling, since the partitioned test sub-sequences have smaller granularities in terms of their area sizes and can be packed more tightly. Figure 4.3(a) shows a non-partitioned deterministic test sub-sequence for core C_i and Figure 4.3(b) shows its three partitions (DTS_{i1}, DTS_{i2}, and DTS_{i3}). In Figure 4.3(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 sub-sequence in Figure 4.3(a). The grey rectangles with dashed line edges illustrate the reduced area sizes due to the partitioning.

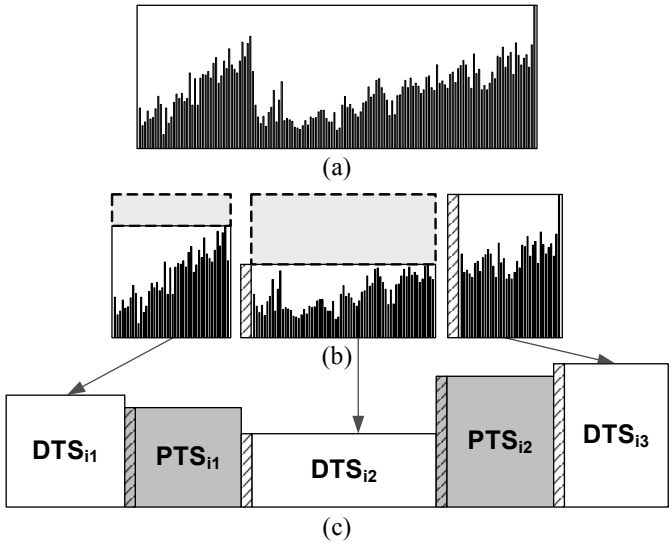
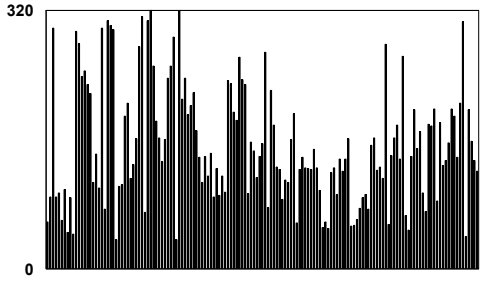
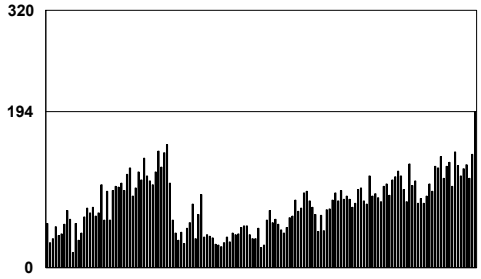


Figure 4.3: Test set partitioning and the time overheads

Reordering test patterns is useful in order to reduce power consumption and can make the power profile of a test sub-sequence relatively smooth and easy to manipulate [Ros02]. Thus, for all deterministic tests, we have used test pattern reordering as a pre-processing step for the test set partitioning. In Figure 4.4(a), the original power profile of a deterministic test sub-sequence is given. As a comparison, the power profile after the test pattern reordering is shown in Figure 4.4(b). It can be seen that, through reordering of the test patterns, the power profile is much smoother and the peak-power consumption is reduced (39% lower for this example).



(a) Power profile before reordering test patterns



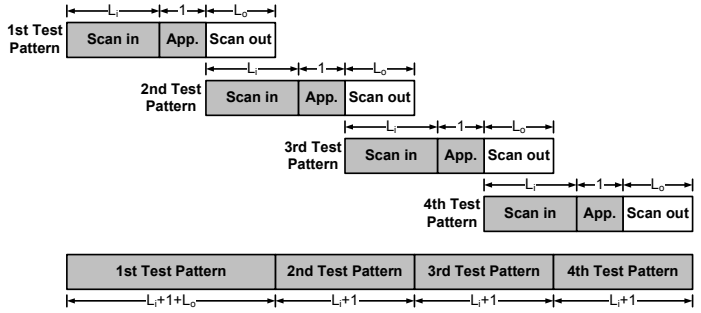
(b) Power profile after reordering test patterns

Figure 4.4: Power profile before/after reordering test patterns

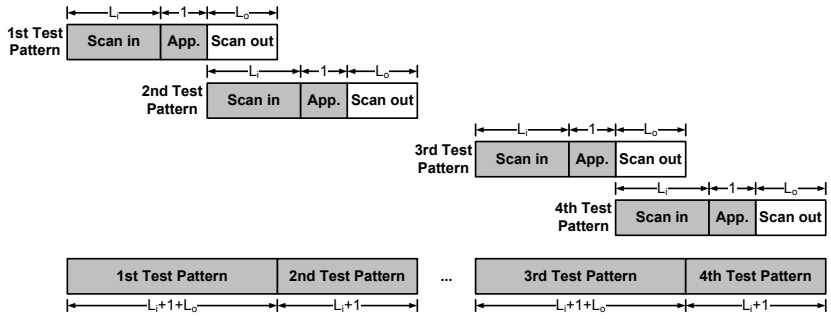
Although test set partitioning can lead to smaller partitions, it introduces time overheads for the partitioned test sub-sequences when a test-per-scan approach is employed. This phenomenon occurs when deterministic test sub-sequences and pseudorandom test sub-sequences belonging to the same core are interleaved, as in the example depicted in Figure 4.3. There, the three partitioned deterministic test sub-sequences (DTS_{i1}, DTS_{i2}, and DTS_{i3}) are interleaved with two partitioned pseudorandom test sub-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 time overheads are due to the following fact. When a deterministic (pseudorandom) test is stopped and resumed later after a pseudorandom (deterministic) test has been applied, the pipeline consisting of three operations (scan-in, application, and scan-out, see Figure 4.5(a)) is interrupted and has to be refilled at the beginning of the latter partition (see Figure 4.5(b)). Thus, the time overhead added to the latter partition is equal to the time duration of the scan-out operation, denoted with L_o in Figure 4.5.

In Figure 4.3(b), the grey rectangles are the areas reduced from the non-partitioned test sub-sequence, while the rectangles filled with slashed lines are the areas added. Thus, we proposed a heuristic to find an appropriate number of partitions for a deterministic test set, such that the sum of the area sizes of all partitions is minimized.



(a)



(b)

Figure 4.5: Pipeline in a test-per-scan approach

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

At every iteration step, we have to decide which existing partition should be selected to be split into two test sub-sequences, and at which position (test pattern) the selected partition should be divided. With an exhaustive search among all possible solutions within this iteration step, the local optimal partitioning scheme with the lowest cost is obtained and one more partition is added. In the global range, among all the local optimal partitioning schemes with different number of partitions, the one with the lowest cost is acquired and accepted as the best solution. Figure 4.6 illustrates how the sum of the area sizes of all partitions distributed with different numbers of partitions. Usually the best partitioning scheme has a relatively small number of partitions in relation to the total number of test patterns in the test set. For example, in Figure 4.6, 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 pseudorandom test sub-sequence is divided into two partitions, two signatures are needed in order to obtain the test results at the end of both partitions, which means that an additional signature should be generated. Thus, extra memory is also needed to store this additional fault-free signature, and an extra time slot is needed to analyze the additionally generated signature. In this chapter, we have assumed that there exists sufficient memory to store the signatures and we ignore the extra time slots for the analysis of the additional signatures, since this time is very short, compared to the time duration of the pseudorandom test sub-

sequence. We do not consider the impact of the increased complexity of the test controller either in this chapter.

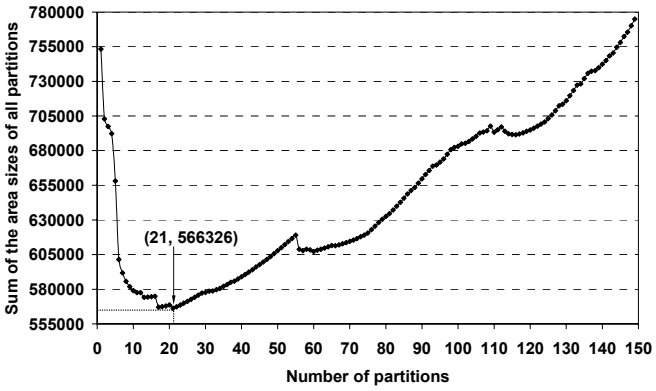


Figure 4.6: Sum of area sizes w.r.t. number of partitions

4.5 Proposed Heuristic

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

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

(2) Deterministic test sub-sequences are scheduled sequentially since a single test bus is used, while pseudorandom test sub-sequences are scheduled in parallel under the peak-power constraint.

(3) The scheduling of deterministic test sub-sequences is performed before the scheduling of pseudorandom test sub-sequences, which means that deterministic test sub-sequences have higher scheduling priorities. This is because deterministic tests can be stopped after every test pattern, while pseudorandom tests can only be terminated at the end of the test sub-sequences, when the signatures are ready.

Additionally, deterministic test patterns are usually more efficient in detecting faults than pseudorandom test patterns.

(4) Pseudorandom test sub-sequences are first sorted in a decreasing order of certain parameters like the defect probability of a core, the peak-power consumption and time duration of a test sub-sequence. Thereafter, they are scheduled to the earliest possible time moment. Deterministic test sub-sequences, however, are scheduled in the order obtained by a defect-probability driven heuristic.

Test set partitioning is integrated into the test scheduling approach in the following way. Deterministic test sets are partitioned statically, meaning that they are partitioned before being scheduled, according to the principles presented in the previous section. Pseudorandom test sets, on the other hand, are partitioned during the test scheduling. When it is impossible to schedule a pseudorandom test sub-sequence to the earliest time moment due to its large area size, the test sub-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 has been developed to find an efficient test schedule for all test sub-sequences in an iterative way. One iteration step of the heuristic is illustrated with the example in Figure 4.7. Suppose that we have five deterministic test sub-sequences DTS_1, DTS_{21}, DTS_{22}, DTS_{31}, and DTS_{32}, and three pseudorandom test sub-sequences PTS_1, PTS_2, and PTS_3. Two deterministic test sub-sequences DTS_{31} and DTS_1 have already been scheduled. In this iteration step, we have to decide which one out of the three unscheduled deterministic test sub-sequences DTS_{21}, DTS_{22}, and DTS_{32} should be scheduled to which time moment among A, B, and C, as illustrated in Figure 4.7. After a deterministic test sub-sequence is scheduled to a time moment, the three pseudorandom test sub-sequences PTS_1, PTS_2, and PTS_3 are scheduled to the rest of the space. Test set partitioning may be needed at this step. Thereafter, the expected partial test application time (EPTAT) is calculated within the range of the scheduled

deterministic test sub-sequences (see Figure 4.8). When all the possible nine solutions within the current iteration step have been explored, the solution with the smallest EPTAT is accepted and the three scheduled deterministic test sub-sequences are taken as a base for the next iteration step. The heuristic stops when no more unscheduled deterministic test sub-sequences are left, and the final test schedule is then obtained. Note that when a test sub-sequence is scheduled, the order of the already scheduled test sub-sequences should remain unchanged.

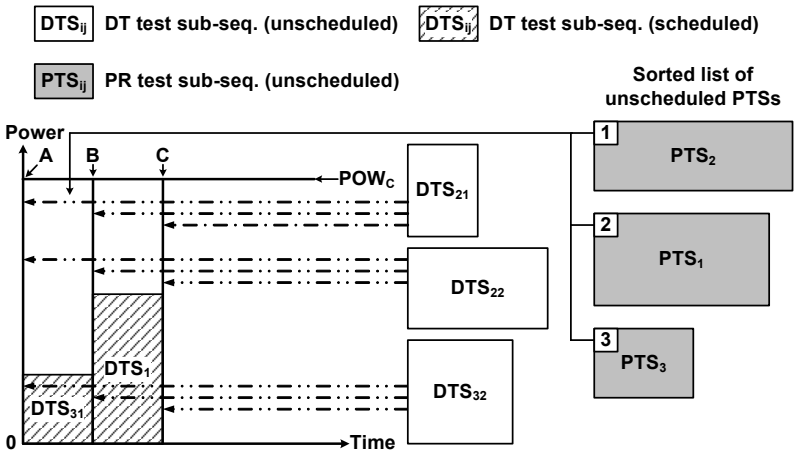


Figure 4.7: Illustration of one iteration step of the heuristic

Figure 4.8 shows a solution in which DTS_{22} is scheduled to time moment *B*. During the scheduling of pseudorandom test sub-sequences, PTS_2 is partitioned into two test sub-sequences PTS_{21} and PTS_{22}. The EPTAT calculation range is from the beginning of DTS_{31} till 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.

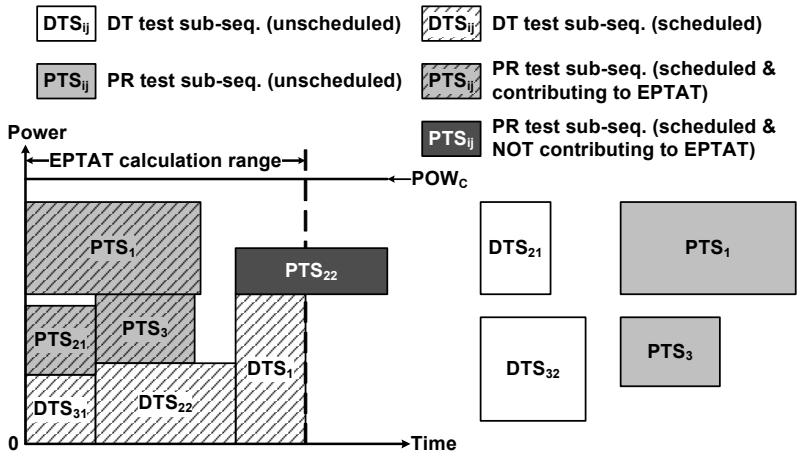


Figure 4.8: Illustration of one solution at the iteration step

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

The pseudo-code of the heuristic, given in Figure 4.9, has three major embedded loops. The outer loop (lines 1 to 19) increments the number of scheduled deterministic test sub-sequences, the middle loop (lines 4 to 17) selects every unscheduled deterministic test sub-sequence, and the inner loop (lines 5 to 16) explores every possible time moment for scheduling. Inside the inner loop, the selected deterministic test sub-sequence is scheduled (line 6), thereafter pseudorandom test sets are partitioned if needed and then scheduled (lines 7 to 10). The EPTAT of the present schedule is then calculated (line 11) and compared to the minimum EPTAT for an acceptance decision (lines 12 to 15). The final test schedule is returned in the end (line 20).

```
1:  for (#SchedDTS := 0 to N-1) loop
2:      Reset(EPTAT_min);
3:      m := #SchedDTS;
4:      for (∀UnschedDTS_ij) loop
5:          for (∀PTTM T_x) loop
6:              Schedule(UnschedDTS_ij, T_x);
7:              for (∀ pseudorandom test set PTS_k ) loop
8:                  Partition(PTS_k) if needed;
9:                  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 4.9: Pseudo-code of the heuristic for test scheduling

4.6 Experimental Results

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

In the first set of experiments, the proposed test set partitioning and test scheduling technique was employed. We did experiments for 5 groups of designs. Each group had five different designs which had the same number of cores of different types, but the cores were assigned with different defect probabilities. The numbers of cores were 5, 10, 20, 30, and 50 for each group, respectively. For each design we used three different levels of peak-power constraints. The experimental results in Table 4.1 represent the average values from 15 experiments (5 different designs with the same number of cores multiplied by 3 different peak-power constraints). The defect probabilities of individual cores were generated randomly, while keeping the system defect probability at the value 0.6, i.e. 40% system yield.

Table 4.1: Different approaches using test set partitioning

# 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

In order to show the efficiency of our heuristic, a classical bottom-left-decreasing (BLD) scheduling algorithm [Les05] is taken for comparison. It sorts deterministic and pseudorandom test sub-sequences decreasingly by their area sizes (the peak-power consumption multiplied by the time duration), and then schedules

them using the bottom-left strategy. As shown in Table 4.1, by employing our heuristic, the ETAT can be reduced around 20% to 29% compared to the BLD scheduling algorithm, with an acceptable increase of execution time. On the other hand, in order to show the accuracy of our heuristic to find a near-optimal test schedule, we also compared our heuristic with a simulated annealing (SA) algorithm. For small designs with 5 and 10 cores, the SA algorithm reached the imposed termination condition in an acceptable time and is supposed to return a solution close to the optimal solution. For large designs with 20, 30, and 50 cores, the SA algorithm took unacceptably long time to reach the termination condition. Thus, for these experiments, we let the SA algorithm run for a time equal to that needed by our heuristic. From Table 4.1, one can see that in small designs, the SA algorithm works just slightly better than our heuristic (2% to 3% lower ETAT), but has up to two orders of magnitude longer execution time than our heuristic. For the large designs, our heuristic found better solutions with 4% to 7% lower ETAT values, than the SA algorithm produced in the same amount of time.

In the second set of experiments where the same designs were used, we intended to show the effect of test set partitioning. As a comparison, we used a defect-probability driven test scheduling heuristic which did not allow test set partitioning. For the sake of fairness, both the partitioned and non-partitioned heuristic used test pattern reordering, thus the advantage of the peak-power reduction by reordering test patterns did not play any role in this comparison. The experimental results are given in Table 4.2. As shown in the table, using test set partitioning can reduce the ETAT with amounts between 16% and 30%. The results are also illustrated in Figure 4.10.

Table 4.2: Our heuristic (without/with test set partitioning)

# Cores	Without Test Set Partitioning		With Test Set Partitioning	
	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

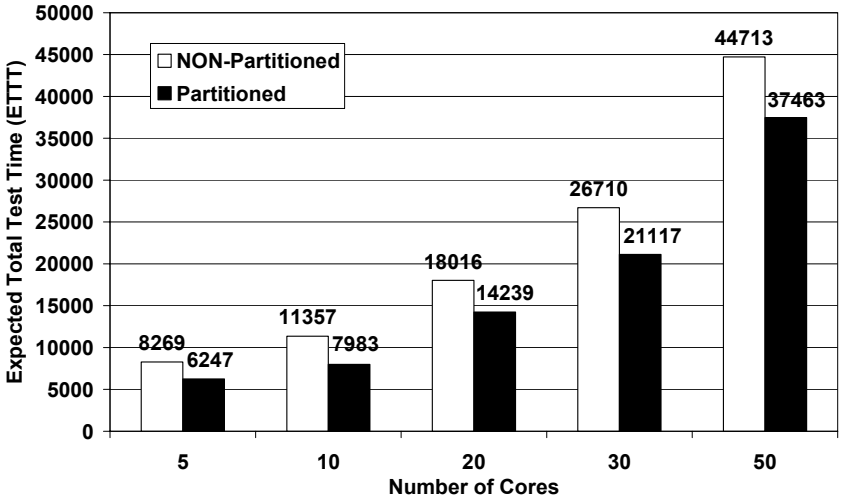


Figure 4.10: Our heuristic (without/with test set partitioning)

4.7 Conclusions

In this chapter, a power-constrained SoC test scheduling approach is presented in a production test environment. Different from other approaches, the defect probabilities of individual cores are utilized to drive the test scheduling and a test set partitioning approach is

employed. Based on the calculation of the ETAT, a heuristic for test set partitioning and test time minimization is used to generate efficient test schedules. Experimental results have shown that the proposed method is effective to shorten the test application time.

Chapter 5
Thermal-Aware SoC Test Scheduling

In this chapter, we assume that a continuous test will increase the temperature of a core towards a limit beyond which the core may be damaged. We also assume a test architecture which employs a test bus to transport test data. Thus, we address the TAT minimization problem as how to generate the shortest test schedule such that the temperature limits of individual cores and the limit on the test-bus bandwidth are satisfied.

5.1 Introduction

Recently, thermal-aware testing [Tad00] has attracted many research interests. Liu et al. proposed a technique [Liu05] to evenly distribute the generated heat across the chip during tests, and therefore avoid high temperature. Rosinger et al. proposed an approach [Ros06] to generate thermal-safe test schedules with minimized test time by utilizing the core adjacency information to drive the test scheduling and reduce the temperature stress between cores. In this chapter, we propose a thermal-aware test scheduling technique [He06b], [He07] which employs test set partitioning and interleaving.

In order to avoid overheating during tests, we partition the entire test set into shorter test sub-sequences and introduce a cooling period between two consecutive test sub-sequences, such that continuously applying a test sub-sequence will not drive the core temperature going beyond the limit. As the TAT substantially increases when long and/or many cooling periods are introduced, we interleave the test sub-sequences from different test sets in such a manner that a cooling period reserved for one core is utilized for the test transportation and application of another core.

We provide two approaches to generate thermal-safe test schedules with minimized TAT: a CLP-based approach [He06b] and a heuristic-based approach [He07]. The CLP-based approach generates the optimal test schedules in terms of the TAT, assuming regular partition length as well as regular cooling periods. The heuristic-based approach explores alternative test set partitioning and interleaving schemes in which partitions and cooling periods have arbitrary length.

5.2 Motivational Example

When a long sequence of test patterns is continuously applied to a core, the temperature of this core may increase towards a certain limit beyond which the core will be damaged. Therefore, a test has to be stopped when the core temperature reaches the limit, and the test can be restarted later when the core has been cooled down. Thus, by partitioning a test set into shorter test sub-sequences and introducing cooling periods between them, we can avoid the overheating during test. Figure 5.1 illustrates a situation in which the entire test set is partitioned into four test sub-sequences, TS_1, TS_2, TS_3, and TS_4, and cooling periods are introduced between them. In this way, the temperature of the core remains under the imposed temperature limit.

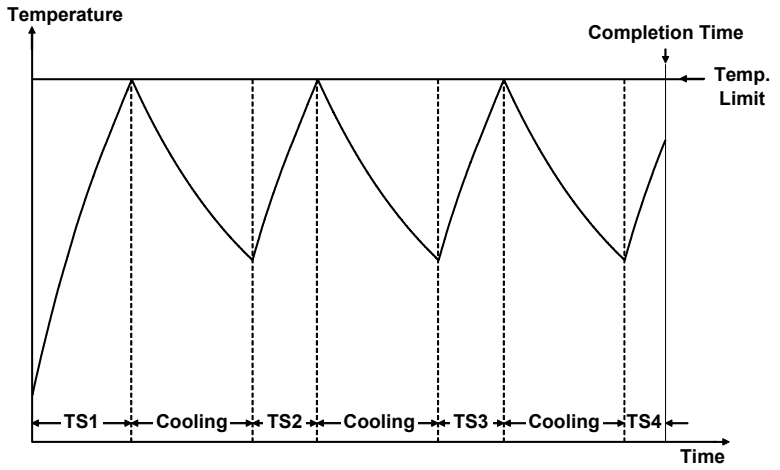


Figure 5.1: An example of test set partitioning

As we have assumed that a test bus is employed in the test architecture, the limit on the test-bus bandwidth becomes a constraint to the scheduling of the test sub-sequences. It is obvious that introducing long cooling periods between test sub-sequences can substantially increase the TAT. Intuitively, we can reduce the TAT by interleaving the partitioned test sets such that the cooling periods reserved for a core C_i are utilized to transport test data for another core C_j ($j \neq i$), and thereafter to test the core C_j. By interleaving the partitioned test sets belonging to different cores, the test-bus bandwidth is more efficiently utilized. Figure 5.2 gives an example where two partitioned test sets are interleaved so that the test time is reduced with no need for extra bus bandwidth.

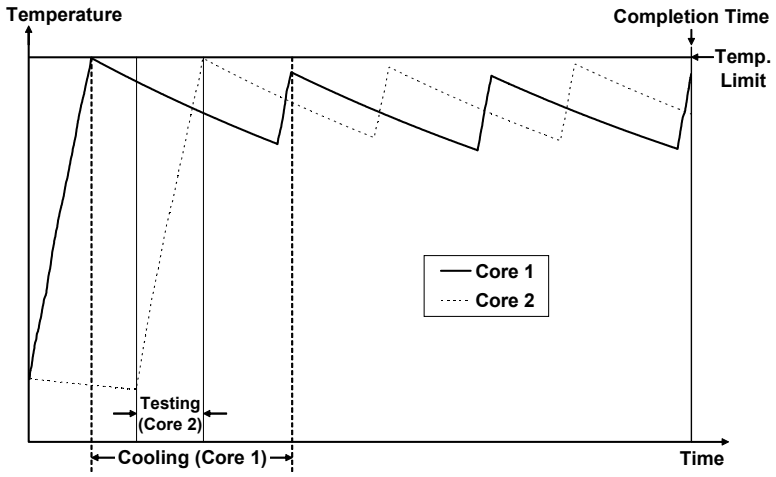
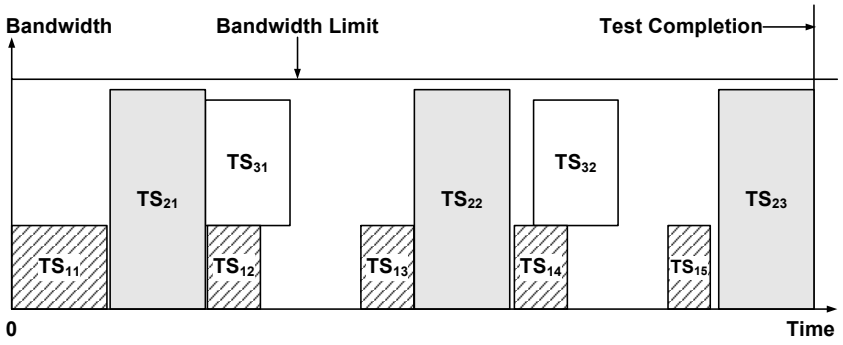
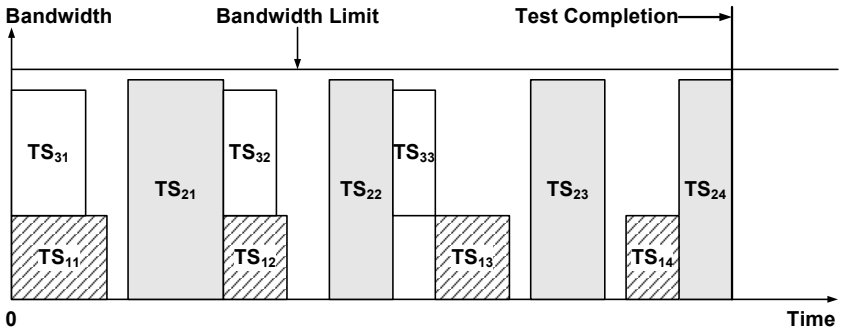


Figure 5.2: An example of test set interleaving

In this chapter, we aim to minimize the TAT by generating an efficient test schedule which avoids violating the temperature limits of individual cores, and at the same time satisfies the test-bus bandwidth constraint. We consider each test sub-sequence as a rectangle, with its height representing the required test-bus bandwidth and its width representing the test time. Figure 5.3 gives a motivational example for our test time minimization problem. Suppose that three test sets, TS_1, TS_2, and TS_3, are partitioned into 5, 3, and 2 test sub-sequences, respectively. Note that the partitioning scheme which determines the length of test sub-sequences and cooling periods has ensured that the temperature of each core will not violate the temperature limit, by using a temperature simulation. Figure 5.3(a) shows a feasible test schedule while Figure 5.3(b) depicts an alternative test schedule where a different partitioning and interleaving scheme is adopted. This example shows the possibility to find a shorter test schedule by exploring alternative solutions, where the number and length of test sub-sequences, the length of cooling periods, and the way that the test sub-sequences are interleaved are different.



(a) A feasible test schedule with regular partitioning scheme



(b) An alternative test schedule with irregular partitioning scheme

Figure 5.3: A motivational example

5.3 Problem Formulation

Suppose that a system S, consisting of n cores C_1, C_2, ... , C_n, employs the test architecture for external test using an ATE, as depicted in Figure 2.5. In order to test core C_i, a test set TS_i consisting of l_i generated test patterns is transported through the test bus and the dedicated TAM wires to/from core C_i, utilizing a bus bandwidth W_i. The test bus is designed to allow transporting several test sets in parallel but has a bandwidth limit BL ($\forall I$, $BL \geq W_i$). We assume that

continuously applying test patterns belonging to TS_i may cause the temperature of core C_i go beyond a certain limit TL_i so that the core can be damaged. In order to prevent overheating during tests, we allow partitioning a test set into a number of test sub-sequences and introducing a cooling period between two partitioned test sub-sequences, such that no test sub-sequence drives the core temperature higher than the limit and the core temperature is kept within a safe rage. The problem that we address is how to generate a test schedule for system S such that the TAT is minimized while the bus bandwidth constraint is satisfied and the temperatures of all cores during tests remains below the corresponding temperature limits. The formal problem formulation is given in Figure 5.4.

Input:
$\{TS_i \mid i = 1, 2, \ldots, n\}$,
$\{W_i \mid i = 1, 2, \ldots, n\}$,
Test-bus bandwidth limit BL,
Temperature limit of each individual core $\{TL_i \mid i = 1, 2, \ldots, n\}$

Output:
Test schedule with the minimized test application time (TAT)

Subject to the following two constraints:
1. At any time moment t before the test completion, the total amount of utilized test-bus bandwidth $BW(t)$ is less than or equal to the bandwidth limit BL, i.e. $BW(t) := \Sigma BW_i(t) \leq BL$.
2. At any time moment t before the test completion, the temperature $TM_i(t)$ of every individual core C_i is less than or equal to the corresponding temperature limit TL_i, i.e. $\forall i$, $TM_i(t) \leq TL_i$.

Figure 5.4: Problem formulation

The formulated problem has high complexity. In order to reduce the optimization time so that optimal solutions can be found in an reasonable time even for large designs, we can restrict the exploration space. Thus we restrict the length of the test sub-sequences belonging to the same test set to be identical. Moreover, we restrict the length of the cooling periods between the test sub-

sequences from the same test set to be identical. By adding these restrictions, the complexity of the test controller can be reduced since less states of the test schedule are needed to store in a memory. Thus, the test scheduling problem can be modeled and solved by using CLP.

Although formulating the test scheduling problem as a CLP model by restricting the exploration space have the advantages demonstrated above, it also leads to trade-offs. The restrictions on the regularity of test sub-sequence and cooling periods can result in less efficient test schedules and, as a consequence, longer TATs. We provide an alternative way to reduce the long optimization time due to the high problem complexity. We assumes that the test sub-sequences and cooling periods can have arbitrary length and we propose a low-complexity heuristic to find near-optimal solutions under the less restricted and more realistic assumptions. Due to the high complexity of the formulated problem, the CLP-based approach is not applicable.

5.4 Overall Solution Strategy

We propose a strategy to solve the formulated problem in two major steps, as illustrated in Figure 5.5. First, we generate an initial partitioning scheme for every test set by using temperature simulation and the given temperature limits. Second, the test scheduling algorithm explores test schedules by selecting alternative partitioning schemes, interleaving test sub-sequences, and squeezing them into a two-dimensional space, constrained by the test-bus bandwidth.

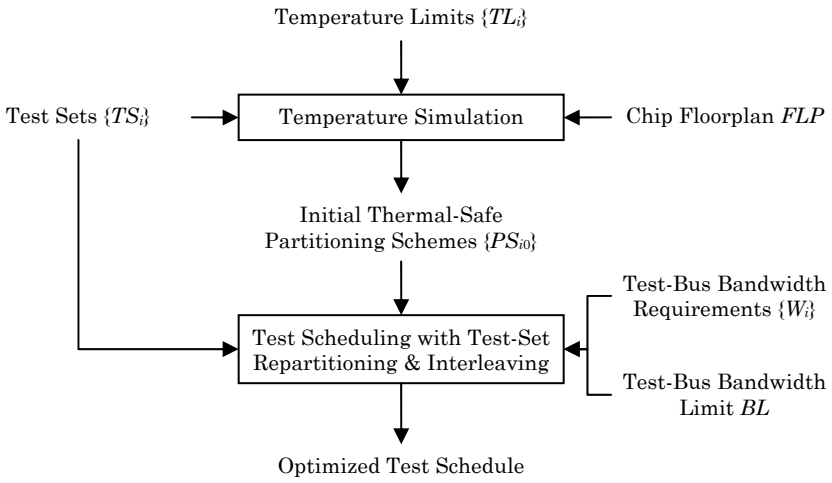


Figure 5.5: The overall solution strategy

In order to generate thermal-safe partitioning schemes, we have used a temperature simulator HotSpot [Ska04], [Hua04], [Hua07] to simulate instantaneous temperatures of individual cores during tests. HotSpot assumes a circuit packaging configuration widely used in modern IC designs, and it computes a compact thermal model [Hua04] based on the analysis of three major heat flow paths existing in the assumed packaging configuration [Hua04]. Given the floorplan of the chip and the power consumption profiles of the cores, HotSpot calculates the instantaneous temperatures and estimates the steady-state temperatures for each unit. In this thesis, we assume that the temperature influences between cores are negligible since the heat transfer in the vertical direction dominates the transferring of dissipated heat. This assumption has been validated by examining simulations with HotSpot.

When generating the initial thermal-safe partitioning scheme, we have assumed that a test set TS_i is started when the core is at the ambient temperature TM_{amb}. Then we start the temperature simulation, and record the time moment t_{h1} when the temperature of core C_i reaches the given temperature limit TL_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 sub-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. It is obvious that a relatively long time is needed in order to cool down a core to the ambient temperature, as the temperature decreases slowly at a lower temperature level (see the dashed curve in Figure 5.6). Thus, we let the temperature of core C_i go down only until the slope of the temperature curve reaches a given value k, at time moment t_{c1}. Note that the value of k can be experimentally set by the designer. At this moment, we have obtained the duration of the first cooling period $d_{i1} = t_{c1} - t_{h1}$. Restarting the test process from time moment t_{c1}, we repeat this heating-and-cooling procedure 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 sub-sequences $\{TS_{ij} \mid j = 1, 2, \ldots, m\}$ and between every two consecutive test sub-sequences, the duration of the cooling period is $\{d_{ij} \mid j = 1, 2, \ldots, m\text{-}1\}$, respectively. Figure 5.6 depicts an example of partitioning a test set into four thermal-safe test sub-sequences with three cooling periods added in between.

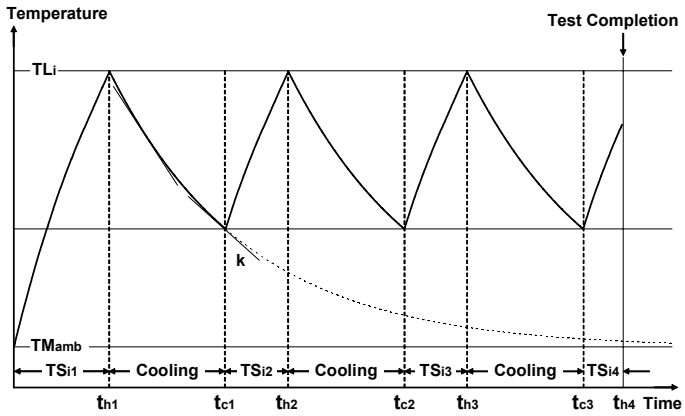


Figure 5.6: An example of initial partitioning scheme

Once the initial thermal-safe partitioning scheme is obtained, we focuses on the problem of how to schedule all the test sub-sequences such that the TAT is minimized under the constraint on the test-bus bandwidth. Since we consider each test sub-sequence as a rectangle, the problem of generating a test schedule with minimized TAT while satisfying the constraint on the test-bus bandwidth can be formulated as a rectangular packing problem. However, our test scheduling problem is not a classical RP problem, due to the fact that the number of test sub-sequences, the length of the sub-sequences, and the cooling periods are not constant. This makes our problem even more difficult to be solved.

Interleaving test sub-sequences belonging to different test sets can introduce time overheads [Goe03], [He06a] when the test controller stops one test and switches to another test. Therefore, partitioning a test set into more test sub-sequences may lead to a longer TAT, since more time overheads and more cooling periods are introduced into the test schedule. On the other hand, partitioning a test set into more test sub-sequences results in a shorter average length of the individual test sub-sequences, which in principle can be packed in a more compact way and thus lead to shorter TATs. Thus, we need a global optimization algorithm, in which different numbers and lengths of test sub-sequences as well as different cooling periods are explored. We have proposed a heuristic to generate optimized test schedules by scheduling test sub-sequences with test set repartitioning and interleaving.

Based on the overall solution strategy, we propose two approaches to solve the test scheduling, a CLP-based approach and a heuristic-based approach.

5.5 CLP-based Approach

As demonstrated previously, in order to restrict the exploration space as well as to reduce the complexity of the test controller, we assume that the test partitions in the same test set have identical time

durations except the first and the last test sequence. Why first test sub-sequence is usually longer than the other in the same test set is because that we assumed that the CUT is at the ambient temperature when the first test sequence starts, and that the first test sequence is partitioned such that it can be continuously applied until the core temperature reaches the temperature limit. For the same reason, the cooling spans between two consecutive test sequences from the same test set also have identical length.

5.5.1 Concepts of CLP

CLP is a programming framework which merges two declarative programming paradigms, constraint solving and logic programming [Jaf87]. CLP defines the relationships between entities as constraints, and incorporates constraint solving methods into a logic based 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 focusing 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 large variety of applications. Some CLP tools also provide solvers to find the optimal solution using branch and bound or exhaustive search. We use CHIP [Hen91] for our CLP-based approach to solve the thermal-safe test scheduling problem.

5.5.2 CLP Model

Each partitioning scheme has three parameters, the number of partitions m_i, the time duration of the first test sequence l_{i1}, and the cooling span d_i between two consecutive test sequences. Each test

starts at time t_i which is equal to the starting time of the first partition in the same test set.

$$t_i = t_{i1} \quad (1 \le i \le n) \tag{5.1}$$

The number of partitions and the test start time is decided during the optimization. The start time t_{ij} and finishing time e_{ij} of the test sub-sequence TS_{ij} can be calculated as follows. Note that o_i is the time overhead.

$$t_{ij} = t_{i,j-1} + l_{i,j-1} + d_i + o_i \quad (2 \le j \le m_i, 1 \le i \le n) \tag{5.2}$$

$$e_{ij} = t_{ij} + l_{ij} \quad (1 \le j \le m_i, 1 \le i \le n) \tag{5.3}$$

The last test sub-sequence in each test set is special since its finishing time is the end of the whole test set. Thus the finishing time e_i of the test set TS_i is

$$e_i = e_{i,m_i} \tag{5.4}$$

and the TAT for testing all cores is the maximum finishing time of all single tests.

$$TAT = \max_{1 \le i \le n} \{ e_i \} \tag{5.5}$$

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 the TAT is minimized, subject to the following two constraints:

(1) at any time moment x, the total amount of test bus bandwidth used by the concurrent test sequences is less than the bandwidth limit, i.e.

$$\forall x \le TAT, \sum_{k=1}^{p_x} W_k \le BL \tag{5.6}$$

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

(2) at any time moment x, the temperature of each core should be less than the temperature limit, i.e.

$$TM_{i,x} \leq TL_i \tag{5.7}$$

where $TM_{i,x}$ is the temperature of core C_i at the time moment x, and TL_i is the temperature limit of core C_i.

We assume that when a test starts, the CUT is at the ambient temperature, denoted with TA. The test set has to be partitioned into a number of test sequences if the CUT reaches its temperature limit before the entire test is completed. When partitioning a test set into test sequences, the length of each test sequence and the number of test sub-sequences depend on the length of the cooling span between two consecutive test sequences. This is because of the following facts. A longer cooling span leads to a lower temperature at which the succeeding test sequence will be started. Thus, with the partitioning schemes that have longer cooling spans, 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 demonstrated in the previous section, we use the temperature simulation to find the interval I_i ($1 \leq i \leq n$) for each test set. We define the number of partitions obtained by using this approach as the minimum value of the exploration interval I_i, denoted with $I_{i,min}$. In order to find the maximum value of the exploration interval 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 consider the maximum value $I_{i,max} = K + I_{i,min}$, where K is a constant fixed by the designer. Thereafter, the obtained exploration interval

$I_i = [I_{i,min}, I_{i,max}]$ (i = 1, 2, ... , n) for TS_i is taken as an input to the optimization algorithm.

For each test set TS_i ($1 \le i \le n$), 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 starting time of the test, denoted with t_i. The two sets of variables for all test sets compromise the decision variables of our optimization problem. The finishing time of a test is equal to its starting time plus the durations of all its test sequences and all the cooling spans between two consecutive test sequences, given as follows.

$$e_i = t_i + \sum_{j=1}^{m_i} l_i + d_i \times (m_i - 1) \tag{5.8}$$

During the optimization, the decision variables are instantiated and test schedules that satisfy the constraints are explored. The solver finds the optimal solution which has the minimal total test application time. The minimal TAT formulated in the CLP model is

$$TAT_{\min} = \min_{1 \le i \le n} \{ \max_{0 \le t_i \le L,\ I_{i,\min} \le m_i \le I_{i,\max}} \{ e_i \} \} \tag{5.9}$$

where L is a constant configured in the CLP model.

5.5.3 Experimental Results

We have used the ISCAS'89 benchmarks as cores for the SoC designs to our experiments. Table 5.1 shows the experimental results for five different SoC designs. The number of cores composing each generated SoC is listed in the first column of Table 5.1. For each SoC design, test patterns are generated for all cores in the design, and the switching activities are calculated for each test pattern. We used the approach in [Sam06] to obtain the power consumption values, taking the switching activities of test patterns as inputs. HotSpot is used to find the total number of partitioning schemes for each SoC design, which are listed in the second column of Table 5.1. The imposed temperature limit is 90°C.

We used the developed CLP formulation to generate the optimal test schedule by selecting the number of partitions and the start time for each test. The third column of Table 5.1 is the problem size of each design, which is the number of partitioning schemes multiplied by the number of cores. The total test time of the optimal solution for each design is shown in the fourth column and the optimization time is listed in the fifth column.

When the test schedule for a design has been generated, we run the HotSpot temperature simulator for the generated test schedules to check if the temperatures of the cores go over the upper limit. The simulation results confirm that the temperatures of cores are below the upper limit.

Table 5.1: Experimental results for five different designs

# Cores	# Partitioning Schemes	Problem Size	TAT (# 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 also did experiments to see how the optimization result is impacted by the given total number of partitioning schemes. In Table 5.2, four different number of partitioning schemes have been given to the optimization algorithm for the same design consisting of 6 cores. The optimal solution is the same in the three cases where the total number of partitioning schemes is 7, 10, and 15, respectively. In the case that the number of partitioning schemes is only 5, the total test time of the obtained solution is larger than those in the other three cases. This experiment shows that, for this design, the best solution does not correspond to the partitioning scheme found among the five alternative partitioning schemes, as indicated in the first line in

Table 5.2. If we introduce 2 additional alternative partitioning schemes (see the second row of Table 5.2), a better solution is found. However, more additional alternatives, up to 15, do not lead to better solutions. The reason why a larger TAT can occur with a partitioning scheme that has less number of partitions is interpreted as follows. When a test set is partitioned into more test sub-sequences, the test sub-sequences as well as the cooling periods are shorter. Although the time overhead in this case is larger, the entire TAT a test can be still shorter. During test scheduling, various partitioning schemes are explored and the optimal solution does not necessarily correspond to the partitioning schemes with minimal number of partitions.

Table 5.2: Experimental results for one design with different number of partitioning schemes

# Cores	# Partitioning Schemes	Problem Size	TAT (# 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

5.6 Heuristic-based Approach

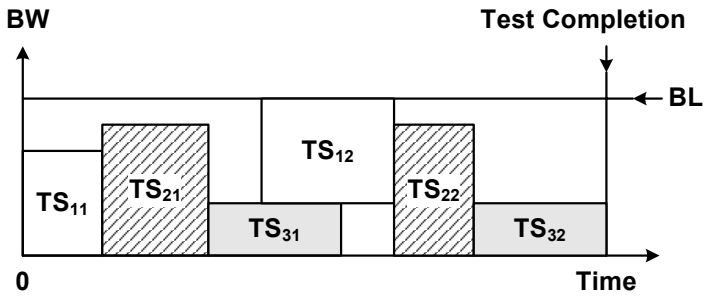
As demonstrated previously, although using the CLP-based approach can provide the optimal solution and also can reduce the complexity of the test controller, the efficiency of the test schedules are limited due to the restrictions on the regularity of test sub-sequences and cooling periods. Alternatively, we assume the test sub-sequences and cooling periods can have arbitrary length and propose a heuristic-based approach to solve the thermal-aware test schedule problem.

5.6.1 Motivational Example

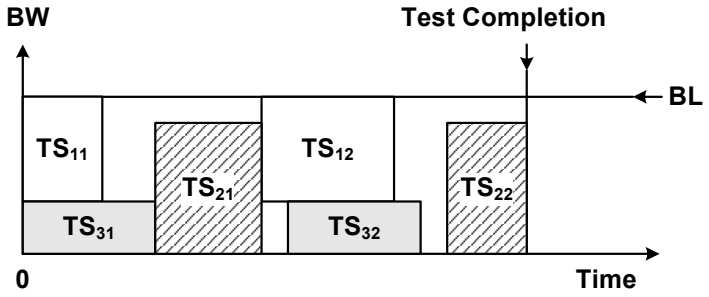
We have proposed a heuristic to do the test scheduling with test set repartitioning and interleaving. Since the order in which the test sets are considered for test scheduling has a large impact on the final test schedule, we construct an iterative algorithm to obtain a good scheduling consideration order (SCO) for all partitioned test sets, and thereafter schedule the test sub-sequences according to the obtained SCO.

Figure 5.7 shows a simple example illustrating the impact of the scheduling consideration order on the test schedule of three test sets, TS_1, TS_2, and TS_3, each of which is partitioned into two test sub-sequences. Figure 5.7(a) and Figure 5.7(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 is obvious that using the second SCO results in a shorter test schedule. Note that in this example the test sets are scheduled to the earliest available time moments.

It should also be noted that the scheduling consideration order refers to the precedence of partitioned test sets to be considered for scheduling. However, when a test set is taken into account for scheduling, we do not schedule all the test sub-sequences of this test set at one time. Instead, we always take the first unscheduled test sub-sequence of the currently considered test set for scheduling, and thereafter take the first unscheduled test sub-sequence of the next test set into account. Thus, in this example, the overall scheduling consideration order (OSCO) for all test sub-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 of Figure 5.7(a) and Figure 5.7(b) respectively. The main concern of not scheduling all test sub-sequences of one test set at one time is to avoid generating low efficient test schedules due to unnecessarily long cooling periods, inappropriate partition length, and inefficient test-set interleaving.



(a) Test schedule with the SCO {TS_1, TS_2, TS_3}



(b) Test schedule with the SCO {TS_3, TS_2, TS_1}

Figure 5.7: Illustration of SCO affecting test schedule length

5.6.2 Heuristic

The basic idea of the proposed heuristic is to iteratively construct a queue that finally consists of all partitioned test sets in a particular order. The pseudo-code of the proposed heuristic (ALG. 5.1) is depicted in Figure 5.8, denoted with. Note that, inside the heuristic, a scheduling algorithm (ALG. 5.2) is invoked, and its pseudo-code is given in Figure 5.11.

```
ALG. 5.1: HEURISTIC for test scheduling
1:  Set of test sets :: U := {TSi | i = 1, 2, ... , n};
2:  Queue of test sets :: Q := ∅;
3:  Queue of test sets :: Q_best := ∅;
4:  for (∀TS ∈ U) loop    /* outer loop */
5:  η_max := 0;
6:  Q := Q_best;
7:  for (∀POS in Q) loop    /* inner loop */
8:      Insert(TS, Q, POS);
9:      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 5.8: Pseudo-code of the heuristic for test scheduling

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

The heuristic starts with an empty queue $Q = Ø$ (line 2). At each iteration step (lines 5 to 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 algorithm terminates when all test sets in U have been moved into Q, and thereafter it schedules the partitioned test sets according to the SCO obtained 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, at 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 5.9 illustrates a situation in which 3 test sets have been inserted in Q (TS_3, TS_8, and TS_6) and 5 test sets remain in U (TS_1, TS_2, TS_4, TS_5, and TS_7). For each test set in U, there are 4 positions for insertion, which the arrows point to. In this example, there are 20 alternative solutions for consideration. Note that each test set in the example has already been partitioned into a number of test sub-sequences, and the scheduling algorithm takes every individual test sub-sequence for scheduling (see ALG. 5.2).

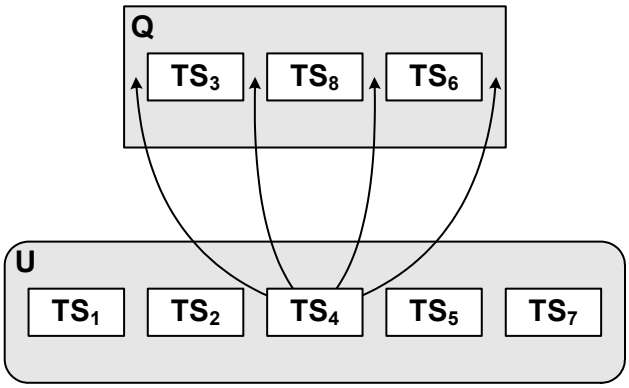


Figure 5.9: An example illustrating alternative solutions

We evaluate the obtained scheduling consideration order 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 the scheduling algorithm ALG. 5.2. Based on the test-schedule

efficiency defined below, we explore different solutions and make decisions according to the efficiency of the generated partial test schedules.

We define the efficiency of a test schedule, denoted with η, as follows. Suppose x is the size of the area covered by all scheduled test sub-sequences, and y is the total area size constrained by the bus bandwidth limit and the completion time moment of the test schedule. The efficiency of the test schedule is the value of x / y. A larger value of η indicates a better test schedule.

Figure 5.10 illustrates how the efficiency of a test schedule is calculated. In the example, a test schedule is given as the area covered by slashed lines. x is the size of the area covered by the actual test schedule, and y is the size of the area covered by the large rectangle surrounded by thick lines.

By calculating and comparing the efficiencies of the alternative partial test schedules (line 10), the best solution that obtains the maximum efficiency is chosen. The maximum efficiency, the chosen test set, and the entire queue, are recorded in η_{max}, TS_{best}, Q_{best}, respectively (lines 12 to 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).

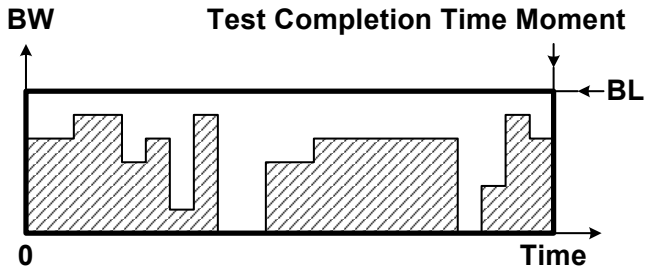


Figure 5.10: Calculation of test schedule efficiency

The algorithm (ALG. 5.2) that schedules a queue of test sets is depicted in Figure 5.11, from lines 21 to 34. Given a queue Q of test

sets, the scheduling algorithm takes the first unscheduled test sub-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 at a time for scheduling. When considering each test set, the scheduler only schedules the first unscheduled test sub-sequence, and thereafter turns to consider the next test set. When one round is finished for all the test sets in Q, the scheduler takes the next round for consideration of scheduling test sub-sequences of all the test sets, in the same SCO. This procedure repeats until all test sub-sequences are scheduled.

```
ALG. 5.2: SCHEDULE(Queue of test sets :: Q)
21: for (j = 1 to max{GetNumOfPar(∀TS ∈ Q)}) loop    /* outer loop */
22:     for (q = 1 to |Q|) loop    /* inner loop */
23:         Choose the q-th test set TSq in Q for scheduling;
24:         if (TSq = ∅) then
25:             Skip TSq and continue with the next test set;
26:         else
27:             Schedule the first unscheduled test sub-sequence TSq,j
                to the earliest available time moment
                    tq,j := GetFinishingTime(TSq,j-1) + dq,j
                where dq := InitialCoolingSpan(TSq);
28:             if (Failed to schedule TSq,j to tq,j) then
29:                 Estimate the completion time te of the entire test set TSq by
                    either postponing TSq,j or repartitioning all the unscheduled
                    test sub-sequences in TSq;
30:                 Choose the solution that has a smaller te and schedule the
                    first unscheduled test sub-sequence;
31:             end if
32:         end if-then-else
33:     end for
34: end for
```

Figure 5.11: Pseudo-code of the scheduling algorithm

Figure 5.12 illustrates how the scheduling algorithm works with an example of three test sets, TS_2, TS_1, and TS_3, sorted with the SCO of $\{TS_2, TS_1, TS_3\}$ in Q. The test set TS_2 has been initially partitioned

into three test sub-sequences, TS_{21}, TS_{22}, and TS_{23}. The rest two test sets, TS_1 and TS_3, are both partitioned into four test sub-sequences. The OSCO of all test sub-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.

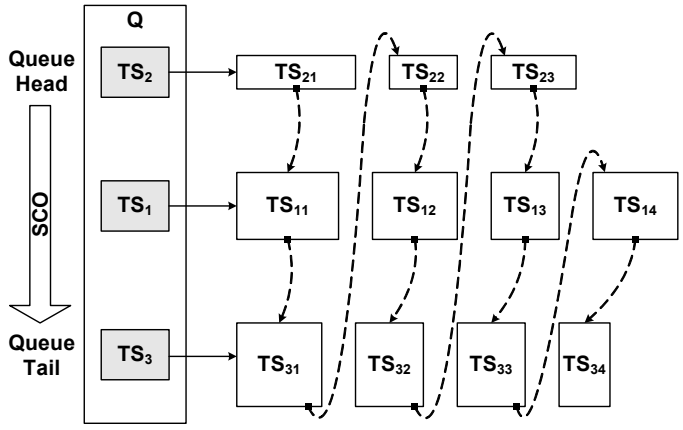


Figure 5.12: Illustration of the scheduling algorithm ALG. 5.2

In the given pseudo-code depicted in Figure 5.11, the scheduling algorithm is constructed with two nested loops. The outer loop (lines 21 to 34) selects the first unscheduled test sub-sequence for the current test set, while the inner loop (lines 22 to 33) selects a test set for scheduling according the its position in Q. The algorithm terminates when all the test sub-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 sub-sequences that the test set has been partitioned into.

When scheduling a test sub-sequence $TS_{q,j}$ (the j-th test sub-sequence of the q-th test set in Q, see line 23 to 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 sub-sequence can be scheduled to is the time moment when the required minimum cooling span succeeding the precedent test sub-sequence has finished. The

minimum cooling span $d_{q,j}$ is given by the initial partitioning scheme for the test set TS_q (line 27).

Although we would like to schedule a test sub-sequence to the earliest available time moment, there can be constraints that make this impossible. Such a constraint is the availability of test-bus bandwidth to be allocated for the required time duration in order to complete the entire test sub-sequence. In Figure 5.13, for example, it is impossible to schedule the test sub-sequence $TS_{q,j}$ at time moment $t_{q,j}$, due to the insufficient space between the bandwidth limit BL and the area occupied by scheduled test sub-sequences (depicted with slashed lines). Actually, in this example, the earliest available time moment that $TS_{q,j}$ can be scheduled at is t_p.

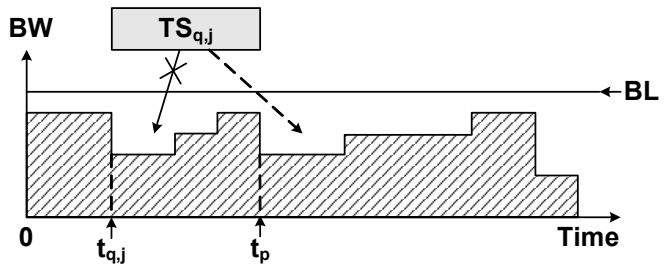
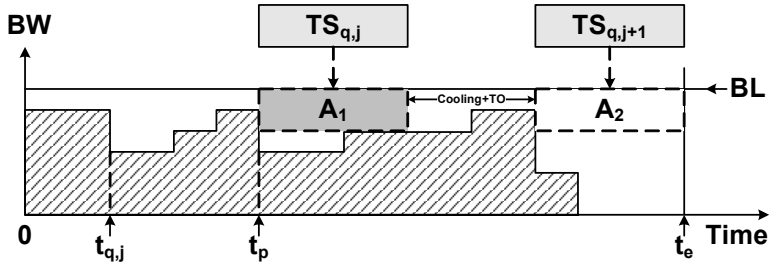
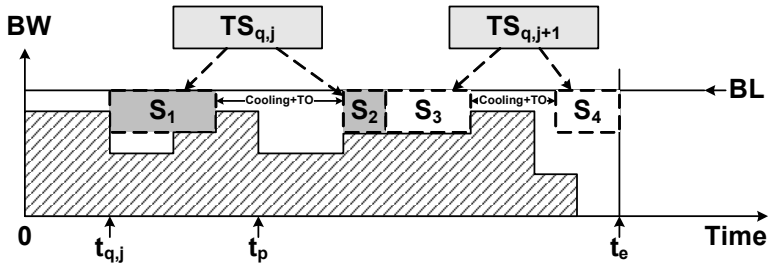


Figure 5.13: An example of scheduling constraints

When encountering such scheduling constraints, two alternatives can be considered. One is to postpone the entire test sub-sequence to a time moment that it can be successfully scheduled to. The other alternative is to split the test sub-sequence into smaller pieces such that the first piece can be squeezed into the available area. Figure 5.14 illustrates both solutions for the same example given in Figure 5.13, where the entire test sub-sequence $TS_{q,j}$ cannot be scheduled at time moment $t_{q,j}$. In Figure 5.14(a), the solution is to postpone the entire test sub-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 5.14(b) illustrates the alternative solution, where $TS_{q,j}$ is split into two pieces which can fit the dark grey rectangular areas S_1 and S_2, respectively.



(a) Postponing the entire test sub-sequence



(b) Splitting the test sub-sequence into smaller pieces

Figure 5.14: Two solutions to schedule a test sub-sequence

Both solutions can result in long test schedules. The first solution, which postpones the entire test sub-sequence, also delays the succeeding test sub-sequences. This can results in delaying the completion of the entire test set. As illustrated in Figure 5.14(a), the succeeding test sub-sequence $TS_{q,j+1}$ is delayed and finishes at time moment t_e. The second solution, which splits the test sub-sequence into smaller pieces, also generates more partitions and introduces more time overheads (TO). In order to avoid these drawbacks, we repartition all the unscheduled test sub-sequences from the same test set, such that the total number of test sub-sequences will not increase dramatically due to the splitting. This is explained in Figure 5.14(b). After splitting $TS_{q,j}$ into two pieces which fits in S_1 and S_2 respectively, we also repartition the succeeding test sub-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 $TS_{q,j+1}$, time overheads (denoted with *TO*) are added between the repartitioned test sub-sequences.

As demonstrated above, both solutions can be adopted when scheduling a test sub-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 sub-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 5.14, the second solution should be chosen, since it leads to a smaller t_e. The scheduling algorithm terminates when all test sub-sequences of all test sets in Q have been scheduled (line 34).

It should be noted that by scheduling test sub-sequences in the demonstrated manner, the test sets have been interleaved and the temperatures of cores under test will not be higher than the temperature limit. This is because that the test sub-sequences are not longer than those in the initial partitioning schemes, and the cooling periods are not shorter than those in the initial partitioning schemes.

5.6.3 Experimental Results

We have done experiments using SoC designs with randomly selected cores in the ISCAS'89 benchmarks. The designs for our experiments have 12 to 78 cores. With the first group of experiments, we demonstrate the impact on the TAT due to various degrees of flexibility of test set partitioning schemes.

We compare our heuristic with other two scheduling algorithms. The first algorithm employs a fixed order 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 sub-sequences of a test set, it keeps the regularity of the partitions and cooling periods given by the initial

partitioning scheme. For the sake of convenience, we call the first algorithm “equal-length scheduling algorithm”.

The second algorithm also employs the fixed order according to the lengths of partitioned test sets (longest first). However, different from the equal-length scheduling algorithm, 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 sub-sequence is usually much longer than the other ones of the same test set in the initial partitioning scheme (see Figure 5.6). Then, in the second phase, it schedules all the remaining test sub-sequences of every test set, according to the same SCO. Similar to the first algorithm, it schedules test sets to the earliest available time moment. When scheduling the test sub-sequences in the second phase, it keeps the regularity of all test partitions and cooling periods given by the initial partitioning scheme, and the first cooling period after the first test sub-sequence may not be shorter than that in the initial partitioning scheme. It can be seen that by separating the scheduling of a test set into two phases, the restriction on partitioning regularity is slightly relaxed, thus this algorithm has higher flexibility on test set partitioning schemes than the equal-length partitioning algorithm. We call the second scheduling algorithm “two-phase scheduling algorithm”.

Compared to the equal-length scheduling and two-phase scheduling algorithm, our heuristic has the highest flexibility on test set partitioning schemes, since it allows repartitioning test sets and allows arbitrarily increasing cooling periods during the scheduling.

Experimental results regarding the first group of experiments are shown in Table 5.3. The first column in the table lists the number of cores used in the designs. Columns 2, 4, and 6 show the test application times of the generated test schedules for the corresponding designs, by using the equal-length scheduling algorithm, the two-phase scheduling algorithm, and our heuristic, respectively. Columns 3, 5, and 7 list the CPU times for executing the corresponding algorithms. Columns 8 and 9 show the percentage

of TAT reduction by using our heuristic, against using the equal-length scheduling algorithm and the two-phase scheduling algorithm, 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 of the equal-length scheduling algorithm and the two-phase scheduling algorithm, respectively.

Table 5.3: Our heuristic vs. equal-length scheduling algorithm, and vs. two-phase scheduling algorithm (to demonstrate the impact of eliminating the regularity restrictions)

# Cores	Equal-Length		Two-Phase		Our Heuristic		TAT Gain (%)	
	TAT	CPU Time (s)	TAT	CPU Time (s)	TAT	CPU Time (s)	from Equal-Length	from Two-Phase
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	n/a	n/a	n/a	n/a	n/a	n/a	30.6%	20.5%

The second group of experiments has been set up in order to see how efficient the test schedules are, which are generated by our heuristic. We compare our heuristic with other two algorithms, a straight forward algorithm (SF) and the simulated annealing based algorithm (SA). In 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 length of cooling periods.

All the three algorithms employ the same scheduling algorithm (ALG. 5.2). The only difference between them is how they generate the SCO for all test sets. The straight forward algorithm 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 sub-sequences to the earliest available time moment, until all test sub-sequences of every test set are scheduled.

The simulated annealing algorithm employs the same scheduling algorithm ALG. 2 to schedule the test sub-sequences, while the SCO of 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 sub-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 experimental results are listed in Table 5.4. Column 1 lists the number of cores used in the designs for experiments. Column 2 shows the TAT of the generated test schedule when the straight forward algorithm is employed, and column 3 lists the corresponding CPU times to obtain the test schedules. Similarly, columns 4 and 5 are the TAT and CPU times for our heuristic, respectively (which are the same as the columns 6 and 7 in Table 5.3). Columns 6 and 7 list the TAT and execution times for the simulated annealing algorithm. In columns 7 and 8, the percentages of reduced TAT of the test schedules generated by our heuristic are listed, compared to those

generated by the straight forward algorithm and the simulated annealing algorithm, respectively.

Table 5.4: Our heuristic vs. straight-forward algorithm, and vs. simulated annealing algorithm (to demonstrate the efficiency of test schedules generated by our heuristic)

# Cores	SF		Our Heuristic		SA		TAT Gain (%)	
	TAT	CPU Time (s)	TAT	CPU Time (s)	TAT	CPU Time (s)	from SF	from SA
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	n/a	n/a	n/a	n/a	n/a	n/a	13.4%	-2.9%

The comparison between our heuristic and the straight forward algorithm aims to show how much TAT can be reduced by a more advanced test scheduling technique. On the other hand, the comparison between our heuristic and the simulated annealing algorithm is to find out how close the generated test schedule is to a solution which is assumed to be close to the optimal one. In order to

generate a close-to-optimal solution, the SA algorithm has been run for very long optimization times.

It can be seen that, when using our heuristic, the TAT is in average 13.4% shorter than those using the straight forward algorithm. The TAT is in average 2.9% longer than those using the simulated annealing algorithm which however needs much longer optimization times.

5.7 Conclusions

In this chapter, we have presented optimization approaches to minimize the total test time for core-based systems which have a temperature upper limit and a bus bandwidth limit. Based on the proposed test set partitioning and interleaving technique, we used constraint logic programming to solve the optimization problem and obtained the optimal solution. Nevertheless, the optimization times for large designs are excessively long. Therefore, a heuristic approach is also proposed.

The proposed heuristic generates thermal-safe test schedules and minimizes the test application time. Based on the initial partitioning scheme generated by a temperature simulation guided procedure, the heuristic utilizes the flexibility of changing the length of test sub-sequences and the cooling periods between test sub-sequences, and interleaves them to generate efficient test schedules. Experimental results have shown the efficiency of the presented approaches.

Chapter 6
Conclusions and Future Work

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

6.1 Conclusions

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

The first approach we have proposed is the defect-probability driven test scheduling technique. It employs the AOFF approach and solves the test time minimization problem in a high-volume production test environment. A hybrid BIST technique and its corresponding test architecture have also been assumed. We have considered the TAT of such a test process as a random variable and have defined the ETAT as the mathematical expectation of the TAT. The proposed heuristic to minimize the ETAT.

The second technique proposed in this thesis is a power-constrained test scheduling approach. The main purpose of

introducing a power constraint at test scheduling is to prevent power and thermal related problems during test. We have considered the test scheduling problem under a given power constraint as a rectangle packing problem. In order to improve the efficiency of test schedules, we have introduced test set partitioning in the test scheduling approach.

Finally we have prepared a test scheduling approach with temperature considerations. The presented approach has assumed that limits on temperature of individual cores and a limit on the test-bus bandwidth are given. A test partitioning technique has been proposed in order to prevent overheating the cores by keeping tests periodically on and off with cooling periods inserted in between. Further, a test set interleaving technique has been proposed to improve the efficiency of the test schedules. Based on the test set partitioning and interleaving techniques, a CLP-based solution and a heuristic algorithm have been proposed to solve the test time minimization problem under the given temperature and bandwidth constraints.

6.2 Future Work

In the third approach, we have assumed that the heat transfer between the adjacent cores is negligible and therefore we ignore the temperature influences between the adjacent cores. However, this assumption has strongly dependency on the packaging configuration and the technology used in the manufacturing process. New process technologies and packaging configuration can change the situation of lateral heat flow in new generations of integrated circuits. Thus, we can extend our work on the thermal-aware test scheduling by considering the variation of the temperature dependencies between cores.

In a long term, the test problems related to the process variation can be a possible direction for future work. When the CMOS process moves into sub-40 nanometer regime, the unreliability of the circuits

becomes large and unavoidable, and the testing of such circuits becomes very difficult since many fault models are not applicable any more. Power- and temperature-aware testing concerning the effects of process variation are very interesting topics for future work.

References

[Abr94] M. Abramovici, M. A. Breuer, and A. D. Friedman, "Digital systems testing and testable design," *IEEE Press*, Sep. 1994.

[Aer98] J. Aerts and E. J. Marinissen, "Scan chain design for test time reduction in core-based ICs," in *Proc. IEEE International Test Conference*, 1998, pp. 448–457.

[Bak80] B. S. Baker, E. G. Coffman Jr., and R. L. Rivest, "Orthogonal packings in two dimensions," *SIAM Journal of Computing*, vol. 9, no 4, pp. 846–855, Nov. 1980.

[Bor99] S. Borkar, "Design challenges of technology scaling," *IEEE Micro*, vol. 19, no. 4, pp. 23–29, Jul. 1999.

[Bus00] M. L. Bushnell and V. D. Agrawal, "Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits," *Kluwer Academic Publishers*, Jan. 2000.

[Cha00] K. Chakrabarty, "Design of system-on-a-chip test access architectures under place-and-route and power constraints," in *Proc. IEEE/ACM Design Automation Conference*, 2000, pp. 432–437.

[Cho97] R. Chou, K. Saluja, and V. Agrawal, "Scheduling tests for VLSI systems under power constraints," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 5, no. 2, pp. 175–184, Jun. 1997.

[Cot02] E. Cota, L. Carro, A. Orailoglu, and M. Lubaszewski, "Test planning and design space exploration in a core-based environment," in *Proc. IEEE Design, Automation and Test in Europe*, 2002, pp. 478–485.

[Dev94] S. Devadas, A. Ghosh, and K. Keutzer, "Logic synthesis," *McGraw-Hill*, Jun. 1994.

[Dyc90] H. Dyckhoff, "A typology of cutting and packing problems," *European Journal of Operational Research*, vol. 44, no. 2, pp. 145–159. 1990.

[Dyc97] H. Dyckhoff, G. Scheithauer, and J. Terno, "Cutting and packing", in: M. Dell'Amico, F. Maffioli, S. Martello (Eds.), *Annotated Bibliographies in Combinatorial Optimization*, Wiley, Chichester, 1997, pp. 393–412.

[Ell99] J. P. Elliott, "Understanding behavioral synthesis: A practical guide to high-level design," *Kluwer Academic Publishers*, Dec. 1999.

[Flo99] P. Flores, J. Costa, H. Neto, J. Monteiro, and J. Marques-Silva, "Assignment and reordering of incompletely specified pattern sequences targeting minimum power dissipation," in *Proc. International Conference on VLSI Design*, 1999, pp. 37–41.

[Fly97] D. Flynn, "AMBA: Enabling reusable on-chip designs," *IEEE Micro*, vol. 17, no. 4, pp. 20-27, Jul./Aug. 1997

[Gaj83] D. D. Gajski and R. H. Kuhn, "Guest editor's introduction: New VLSI tools," *IEEE Computer*, vol. 16, no. 12, pp. 11–16, Dec. 1983.

[Ger99] S. Gerstendörfer and H. J. Wunderlich. "Minimized power consumption for scan-based BIST," in *Proc. IEEE International Test Conference*, 1999, pp. 77–84.

[Ger99] S. Gerstendorfer, and H. J. Wunderlich. "Minimized power consumption for scan-based BIST," in *Proc. IEEE International Test Conference*, 1999, pp. 77–84.

[Gir98] P. Girard, C. Landrault, S. Pravossoudovitch, and D. Severac, "Reducing power consumption during test application by test vector ordering," in *Proc. IEEE International Symposium on Circuits and Systems*, 1998, pp. 296–299.

[Goe03] S. K. Goel and E. J. Marinissen, "Control-aware test architecture design for modular SoC testing," in *Proc. IEEE European Test Workshop*, 2003, pp. 57–62.

[Gun01] S. H. Gunther, F. Binns, D. M. Carmean, and J. C. Hall, "Managing the impact of increasing microprocessor power consumption," *Intel Technology Journal*, vol. 5, no. 1, Feb. 2001. [Online]. Available: ftp://download.intel.com/technology/itj/q12001/pdf/art_4.pdf

[Har99] P. Harrod, "Testing reusable IP – a case study," in *Proc. IEEE International Test Conference*, 1999, pp. 493–498.

[He04] Z. He, G. Jervan, Z. Peng, and P. Eles, "Hybrid BIST test scheduling based on defect probabilities," in *Proc. IEEE Asian Test Symposium*, 2004, pp. 230–235.

[He05] Z. He, G. Jervan, Z. Peng, and P. Eles, "Power-constrained hybrid BIST test scheduling in an abort-on-first-fail test environment," in *Proc. Euromicro Conference on Digital System Design*, 2005, pp. 83–86.

[He06a] Z. He, Z. Peng, and P. Eles, "Power constrained and defect-probability driven SoC test scheduling with test set partitioning," in *Proc. IEEE Design, Automation and Test in Europe*, 2006, pp. 291–296.

[He06b] Z. He, Z. Peng, P. Eles, P. Rosinger, and B. M. Al-Hashimi, "Thermal-aware SoC test scheduling with test set partitioning and interleaving," in *Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2006, pp. 477–485.

[He07] Z. He, Z. Peng, and P. Eles, "A heuristic for thermal-safe SoC test scheduling," submitted to *IEEE International Test Conference (2007)* for review.

[Hel92] S. Hellebrand, S. Tarnick, J. Rajski, and B. Courtois, "Generation of vector patterns through reseeding of multiple-polynomial linear feedback shift registers," in *Proc. IEEE International Test Conference*, 1992, pp. 120–129.

[Hen91] P. Van Hentenryck, "The CLP language CHIP: constraint solving and applications," in *Proc. IEEE Computer Society International Conference*, 1991, pp. 382–387.

[Hua01] Y. Huang, W.-T. Cheng, C.-C. Tsai, N. Mukherjee, O. Samman, Y. Zaidan, and S.M. Reddy, "Resource allocation and test scheduling for concurrent test of core-based SoC design," in *Proc. IEEE Asian Test Symposium*, 2001, pp. 265–270.

[Hua04] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusamy, "Compact thermal modeling for temperature-aware design," in *Proc. ACM/IEEE Design Automation Conference*, 2004, pp. 878–883.

[Hua07] W. Huang, S. Ghosh, K. Sankaranarayanan, K. Skadron, and M. R. Stan. "HotSpot: Thermal modeling for CMOS VLSI systems," submitted to *IEEE Trans. Very Large Scale Integration (VLSI) Systems* for review.

[Hus91] S. D. Huss and R. S. Gyurcsik, "Optimal ordering of analog integrated circuit tests to minimize test time," in *Proc. ACM/IEEE Design Automation Conference*, 1991, pp. 494–499.

[Iye01] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "Test wrapper and test access mechanism co-optimization for system-on-chip," in *Proc. IEEE International Test Conference*, 2001, pp. 1023–1032.

[Iye02] V. Iyengar and K. Chakrabarty, "System-on-a-chip test with precedence relationships, preemption and power constraints," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 9, pp. 1088–1094, Sep. 2002.

[Iye03] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "Test access mechanism optimization, test scheduling, and test data volume reduction for system-on-chip," *IEEE Trans. Computers*, vol. 52, no. 12, pp. 1619–1632, Dec. 2003.

[Jaf87] J. Jaffar and J.-L. Lassez, "Constraint logic programming," in *Proc. ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, 1987, pp. 111–119.

[Jer00] G. Jervan, Z. Peng, and R. Ubar, "Test cost minimization for hybrid BIST," in *Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2000, pp. 283–291.

[Jer03] G. Jervan, P. Eles, Z. Peng, R. Ubar, and M. Jenihhin, "Test time minimization for hybrid BIST of core-based systems," in *Proc. IEEE Asian Test Symposium*, 2003, pp. 318–323.

[Jia01] W. Jiang and B. Vinnakota, "Defect-oriented test scheduling," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 3, pp. 427–438, Jun. 2001.

[Kor02] S. Koranne, "On test scheduling for core-based SoCs", in *Proc. IEEE International Conference on VLSI Design*, 2002, pp. 505–510.

[Kor03] R. E. Korf, "Optimal rectangle packing: Initial results," in *Proc. International Conference on Automated Planning and Scheduling*, 2003, pp. 287–295.

[Kor04] R. E. Korf, "Optimal rectangle packing: New results," in *Proc. International Conference on Automated Planning and Scheduling*, 2004, pp. 142–149.

[Lar02] E. Larsson and Z. Peng, "An integrated framework for the design and optimization of SoC test solutions," *Journal of Electronic Testing: Theory and Applications*, vol. 18, no. 4/5, pp. 385–400, Aug. 2002.

[Lar04a] E. Larsson, J. Pouget, and Z. Peng, "Defect-aware SoC test scheduling," in *Proc. IEEE VLSI Test Symposium*, 2004, pp. 359–364.

[Lar04b] E. Larsson, K. Arvidsson, H. Fujiwara, and Z. Peng, "Efficient test solutions for core-based designs," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 5, pp. 758–775, May 2004.

[Les04] N. Lesh, J. Marks, A. McMahon, and M. Mitzenmacher. "Exhaustive approaches to 2D rectangular perfect packings," *Information Processing Letters*, vol. 90, no. 1, pp. 7–14, Apr. 2004.

[Les05] N. Lesh, J. Marks, A. McMahon, and M. Mitzenmacher, "New heuristic and interactive approaches to 2D rectangular strip packing," *Journal of Experimental Algorithmics*, vol. 10, pp. (1.2)1–18, Dec. 2005.

[Liu05] C. Liu, K. Veeraraghavant, and V. Iyengar, "Thermal-aware test scheduling and hot spot temperature minimization for core-based systems," in *Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2005, pp. 552–560.

[Mah02] R. Mahajan, "Thermal management of CPUs: A perspective on trends, needs and opportunities," Keynote presentation in *the 8th International Workshop on Thermal Investigations of ICs and Systems*, Oct. 2002. [Online]. Available: http://tima.imag.fr/Conferences/therminic/Therminic02/Posters/Rmahajan.pdf

[Mar00] E. J. Marinissen, S. K. Goel, and M. Lousberg, "Wrapper design for embedded core test," in *Proc. IEEE International Test Conference*, 2000, pp. 911–920

[Mil94] L. Milor and A. L. Sangiovanni-Vincentelli, "Minimizing production test time to detect faults in analog circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 6, pp. 796-813, Jun. 1994.

[Mur00] V. Muresan, X. Wang, V. Muresan, and M. Vladutiu, "A comparison of classical scheduling approaches in power-constrained block-test scheduling," in *Proc. IEEE International Test Conference*, 2000, pp. 882–891.

[Mur96] B. T. Murray and J. P. Hayes, "Testing ICs: Getting to the core of the problem," *IEEE Computer*, vol. 29, no. 11, pp. 32–38, Nov. 1996.

[Nic00] N. Nicolici and B. M. Al-Hashimi, "Power conscious test synthesis and scheduling for BIST RTL data paths," in *Proc. IEEE International Test Conference*, 2000, pp. 662–671.

[Pou00] B. Pouya and A. Crouch. "Optimization trade-offs for vector volume and test power," in *Proc. IEEE International Test Conference*, 2000, pp. 873–881.

[Rav00] C. P. Ravikumar, G. Chandra, and A. Verma, "Simultaneous module selection and scheduling for power-constrained testing of core based systems," in *Proc. International Conference on VLSI Design*, 2000, pp. 462–467.

[Ros02] P. Rosinger, B. M. Al-Hashimi, and N. Nicolici, "Power profile manipulation: A new approach for reducing test application time under power constraints," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 10, pp. 1217-1225, Oct. 2002.

[Ros04] P. Rosinger, B. Al-Hashimi, and N. Nicolici, "Scan architecture with mutually exclusive scan segment activation for shift and capture power reduction," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 7, pp. 1142–1154, Jul. 2004.

[Ros06] P. Rosinger, B. M. Al-Hashimi, and K. Chakrabarty, "Thermal-safe test scheduling for core-based System-on-Chip integrated circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*. Vol. 25, no. 11, pp. 2502–2512, Nov. 2006.

[Sam06] S. Samii, E. Larsson, K. Chakrabarty, and Z. Peng. "Cycle-accurate test power modeling and its application to SoC test scheduling," in *Proc. IEEE International Test Conference*, 2006, pp. 1–10.

[Sax01] J. Saxena, K. M. Butler, and L. Whetsel, "An analysis of power reduction techniques in scan testing," in *Proc. IEEE International Test Conference*, 2001, pp. 670–677.

[Shi04] C. Shi and R. Kapur, "How power-aware test improves reliability and yield," *EETimes*, Sep. 15, 2004. [Online]. Available: http://www.eetimes.com/showArticle.jhtml?articleID=47208594

[Ska04] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Architecture and Code Optimization*, vol. 1, no. 1, pp. 94–125, Mar. 2004.

[Sug00] M. Sugihara, H. Date, and H. Yasuura, "Analysis and minimization of test time in a combined BIST and external test approach," in *Proc. IEEE Design, Automation and Test in Europe*, 2000, pp. 134–140.

[Tad00] P. Tadayon, "Thermal challenges during microprocessor testing," *Intel Technology Journal*, vol. 4, no. 3, Aug. 2000. [Online]. Available: ftp://download.intel.com/technology/itj/q32000/pdf/thermal.pdf

[Tou95] N. A. Touba, and E. J. McCluskey, "Synthesis of mapping logic for generating transformed pseudo-random patterns for BIST," in *Proc. IEEE International Test Conference*, 1995, pp. 674–682.

[Var98] P. Varma and B. Bhatia, "A structured test re-use methodology for core-based system chips," in *Proc. IEEE International Test Conference*, 1998, pp. 294–302.

[Vas06] A. Vassighi and M. Sachdev, "Thermal and power management of integrated circuits," *Springer*, Jan. 4, 2006.

[Wan98] S. Wang and S. K. Gupta, "ATPG for heat dissipation minimization during test application," *IEEE Trans. Computers*, vol. 47, no. 2, pp. 256–262, Feb. 1998.

[Zor93] Y. Zorian, "A distributed BIST control scheme for complex VLSI devices," in *Proc. IEEE VLSI Test Symposium*, 1993, pp. 4–9.

[Zor98] Y. Zorian, E. J. Marinissen, and S. Dey, "Testing embedded-core based system chips," in *Proc. IEEE International Test Conference*, 1998, pp. 130–143.

Appendix A
Abbreviations

AMBA	Advanced Microprocessor Bus Architecture
AOFF	Abort-on-First-Fail
ATE	Automated Test Equipment
BIST	Built-In Self-Test
CDP	Core Defect Probability
CLP	Constraint Logic Programming
CUT	Core Under Test
DFT	Design for Test
DSP	Digital Signal Processor
DTP	Deterministic Test Pattern
DTS	Deterministic Test Sub-sequence
DUT	Device Under Test
EPATA	Expected Partial Test Application Time
ETAT	Expected Test Application Time
IFC	Incremental Fault Coverage
IP	Intellectual Property
ITFP	Individual Test Failure Probability
ITSP	Individual Test Success Probability

LFSR	Linear Feedback Shift Register
MCM	Multi-Chip Module
MUX	Multiplexer
OSCO	Overall Scheduling Consideration Order
PTAT	Partial Test Application Time
PTP	Pseudorandom Test Pattern
PTS	Pseudorandom Test Sub-sequence
PTTM	Possible Test Termination Moment
RP	Rectangle Packing
RT	Register-Transfer
SA	Simulated Annealing
SCO	Scheduling Consideration Order
SDP	System Defect Probability
SoC	System-on-Chip
STUMPS	Self-Testing Using MISR and Parallel SRSG
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/Test Sequence
TSP	Test Set Partitioning
UDL	User Defined Logic

Appendix B
Explanations

This appendix gives detailed explanations to Equation (3.8) and Equation (3.9).

Definition 1: Test set and test patterns.

Suppose that a test has m test patterns to be applied in total, 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.

$$TS = \{v_j | 1 \le j \le m\} = \{v_1, v_2, \cdots, v_j, \cdots, v_m\}$$

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

$$IFC(v_j) = \frac{n_j}{N} \quad (0 \le n_j < N, 1 \le j \le m)$$

Definition 3: Failure and success of a test pattern.

We define two random events regarding a test pattern: the failure of a test pattern and the success of a test pattern. The failure of the j-th ($1 \le j \le m$) test pattern, denoted with F_j, is defined as the test pattern detecting at least one fault. This implies that the test is aborted immediately. The success of the j-th ($1 \le j \le m$) test pattern, denoted with $\neg F_j$, is defined as the test pattern detecting no faults. This implies that the test is continued and the next test pattern is going to be applied. Obviously failure and success of a test pattern are complement events. Let D be the random event that a core under test is defected, then defect probability of a core is $DP = p[D]$.

Definition 4: Conditional probability of the failure of the currently applied test pattern.

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 fault and the core is indeed defected.

$$p\left[F_j \mid \neg F_{j-1} \cap \neg F_{j-2} \cap \cdots \cap \neg F_1 \cap D\right] = \frac{n_j}{N - n_{j-1} - n_{j-2} - \cdots - n_1}$$

$$i.e. \quad p\left[F_j \mid \bigcap_{k=1}^{j-1} \neg F_k \cap D\right] = \frac{n_j}{N - \sum_{k=1}^{j-1} n_k}$$

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

formula, those faults should have been detected by preceding test patterns before they were detected by v_j.

Definition 5: Termination and success of a test.

The termination of a test at the j-th test pattern, denoted with A_j, is defined as the random event that the test is terminated immediately after the j-th test pattern v_j has detected at least one fault. The success of a test at the j-th test pattern, denoted with P_j, is defined as the random event that the test is continued after the j-th test pattern v_j is applied without detecting any faults.

According to the definitions, A_j is equivalent to the intersection of the following events: the j-th test pattern detects at least one fault, the preceding test patterns did not detect any faults, and the core is actually defected. Similarly, P_j is equivalent to the intersection of two events: one is an conjunction of such events that all the j applied test patterns did not detect any faults and the core is actually defected, and the other is that the core is actually not defected. A_j and P_j are given by

$$A_j = F_j \cap \neg F_{j-1} \cap \neg F_{j-2} \cap \cdots \cap \neg F_1 \cap D$$

$$i.e.\ \ A_j = F_j \cap \bigcap_{k=1}^{j-1} \neg F_k \cap D$$

$$P_j = \left(\neg F_j \cap \neg F_{j-1} \cap \neg F_{j-2} \cap \cdots \cap \neg F_1 \cap D\right) \cup \neg D$$

$$i.e.\ \ P_j = \left(\bigcap_{k=1}^{j} \neg F_k \cap D\right) \cup \neg D$$

Definition 6: Suppose that a test employs the abort-on-first-fail (AOFF) 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.

$$p[A_j] = IFC(v_j) \times DP$$

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

The above two equations are obtained by using a mathematical induction, 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

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

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

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

Step j: According to the assumptions given in Step ($j-1$), we have

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

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

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

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

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

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

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

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

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

□