OPTIMAL TEST SOLUTION FOR CORE BASED SYSTEM ON CHIP

Thesis submitted in the partial fulfilment of requirement for the award of degree of

Master of Technology

in

VLSI Design & CAD

Submitted By:

Debbrat Ghosh

Roll No.: 601061007

Under the Guidance of:

Ms. Harpreet Vohra

Assistant Professor

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

(Established under the section 3 of UGC Act, 1956)

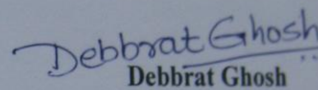
PATIALA – 147004 (PUNJAB)

June-2012

CERTIFICATE

I hereby declare that the work which is being presented here in the thesis entitled, **"OPTIMAL TEST SOLUTION FOR CORE-BASED SYSTEM ON CHIP"** in partial fulfilment of the requirement for the award of degree of M.Tech (VLSI Design and CAD) at Electronics and Communication Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Ms. Harpreet Vohra, Assistant Professor, ECED.

The matter presented in this thesis has not been submitted in any other University/Institute for the award of my degree.

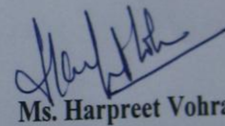


Date: 26/06/2012

Debbrat Ghosh

Roll No.: 601061007

It is certified that the above statement made by the student is correct to the best of my knowledge and belief.

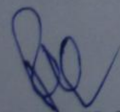


Ms. Harpreet Vohra

Assistant Professor

ECED, Thapar University

Counter Signed By:

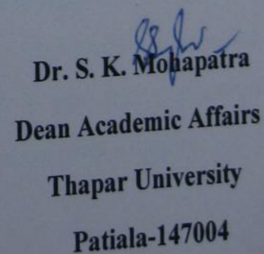


Dr. R. K. Khanna

Professor & Head

ECED, Thapar University

Patiala-147004



Dr. S. K. Mohapatra

Dean Academic Affairs

Thapar University

Patiala-147004

ACKNOWLEDGEMENT

I would like to take the opportunity to express my gratitude to some people who were involved in this thesis work. First, I owe my gratitude to my guide Ms. Harpreet Vohra for doing everything from the inception of the project idea to giving invaluable suggestions at every step. I am also thankful to Dr. R. K. Khanna, Head of the Department and Dr. Kulbir Singh PG Coordinator as well all the faculty members and staff of the Department of Electronics and Communication Engineering for being very supportive to me. I would also like thank all my colleagues for motivating me all the time whenever I needed them and giving me useful tips. I thank all those who have contributed directly or indirectly to this work.

Debbrat Ghosh
601061007

ABSTRACT

The advancement in semiconductor technology has enabled the fabrication of integrated circuits (ICs), which may include billions of transistors and can contain all necessary electronic circuitry for a complete system, so-called System-on-Chip (SOC). A System on Chip (SoC) typically integrates a heterogeneous mix of digital logic, embedded memories, user defined logic (UDL) and analog blocks. The manufacturing process may result in defect chips, for instance due to the base material, and therefore testing chips after production is important in order to ensure fault-free chips. The testing time for a chip will affect its final cost. Thus it is important to minimize the testing time for each chip. To reduce test cost, SoCs are being increasingly tested in modular fashion, i.e. their various design modules are tested separately. For core-based SOCs this can be done by testing several cores at the same time, instead of testing the cores sequentially.

Now-a-days due to functional and performance requirements, modern SoC designs are not limited to only one level of design and test hierarchy. Hierarchy imposes constraints on the manner in which tests must be applied to “parent” cores and their “child” cores.

In this thesis we explore and analyze all the previous work on wrapper design/TAM co-optimization and test scheduling for testing of SoCs taking into consideration multiple constraints like test access time, core testing time, multiple test sets, interconnect testing etc.. It has been found that all the previous work done in wrapper design/TAM co-optimization and test scheduling for testing has treated all the cores in the SoC as if at same level of hierarchy i.e. flat cores. We make use all the analysis done and use the modified wrapper cell design, Multi-level TAM design technique and a efficient test scheduling algorithm based on Integer Linear Programming for handling the testing of SoC with a mix of hierarchical and non-hierarchical cores. A new test architecture is proposed for test access architecture design of megacore based on pareto-optimal points.

TABLE OF CONTENTS

CERTIFICATE (i)
ACKNOWLEDGEMENT (ii)
ABSTRACT (iii)
TABLE OF CONTENTS (iv)
LIST OF FIGURES (vii)
LIST OF TABLES (ix)
ABBREVIATIONS (x)

CHAPTER 1- Introduction 1
1.1 Introduction & Motivation 1
1.2 SOC Test Process 4
1.3 Core-Based SOC Test 5
1.4 Constraints in Test Process 7
1.5 Problem Description 9
1.6 Thesis Overview 10

CHAPTER 2- Preliminaries 11
2.1 System on Chip Model 11
2.2 Wrapper Design 12
2.3 Test Access Mechanism 13
2.4 Test Scheduling 14

CHAPTER 3- Literature Review 18
3.1 Testing Time Minimization 18
3.1.1 Problem Definition 18
3.1.2 Problem of Core Test Time Minimization 19
3.1.2.1 Mathematical Formulation 19
3.1.3 Problem of Core Access Time Minimization 21
3.1.3.1 Mathematical Formulation 21
3.2 Test Access Architecture Optimization 22
3.3 Test Scheduling 23

3.3.1 Open shop Scheduling Problem 24
3.3.2 Simulated Annealing Problem 24
3.3.3 RAIN (RAndom INsertion) formulation 24
3.3.4 Ant Colony Optimization 25
3.3.5 Genetic Algorithm 26
3.3 Testing of Hierarchical Core-Based System-on-Chip 26

CHAPTER 4- Proposed Test Architecture 28
4.1 Wrapper design/TAM optimization and test schedule using Integer Linear Programming 28
4.1.1 Problem Formulation 28
4.1.2 Algorithms 30
4.2 Design & optimization of multi-level TAM architecture for hierarchical SOCs 33
4.2.1 Non-interactive design transfer model 33
4.2.1.1 Problem Formulation 34
4.2.1.2 Algorithms 34
4.2.2 Interactive design transfer model 35
4.2.2.1 Problem Formulation 35
4.2.2.2 Algorithms 36
4.3 IEEE P1500 modified test wrapper design for hierarchical cores 37
4.3.1 Problem Formulation 39
4.3.2 Algorithms 40
4.4 Proposed Test Architecture for hierarchical cores based on pareto-optimal points 42
4.4.1 Pareto-optimal points 42
4.4.2 Algorithms 43

CHAPTER 5- Results 45
5.1 Results of wrapper design/TAM optimization and test schedule using Integer Linear Programming 45
5.2 Results of Design and optimization of multi-level TAM architecture for hierarchical SOCs 49

5.2.1 Non-interactive design transfer model 49
5.2.2 Interactive design transfer model 50
5.3 Results of IEEE P1500 modified test wrapper design for hierarchical cores 52
5.3.1 Non-interactive design transfer model 52
5.3.2 Interactive design transfer model 54
5.4 Results of Proposed test architecture for hierarchical cores based on pareto-optimal points 56

CHAPTER 6- Conclusion & Future Work 63
6.1 Conclusion 63
6.2 Future work 63

REFERENCES 64

APPENDIX A 69

LIST OF FIGURES

Figure 1.1:	Core-based SOC layout	1
Figure 1.2:	Trend in test cost versus fabrication costs per transistor	3
Figure 1.3:	SOC test process	4
Figure 1.4:	Overview of three elements in an embedded-core test approach	6
Figure 1.5:	System on Chip hierarchy	6
Figure 2.1:	SOC model	11
Figure 2.2:	IEEE P1500 TESTSHELL/Wrapper	12
Figure 2.3(a):	Test Architecture Design-Multiplexing architecture	13
Figure 2.3(b):	Test Architecture Design-Daisychain architecture	13
Figure 2.3(c):	Test Architecture Design-Distribution architecture	13
Figure 2.4:	Illustration of test scheduling	14
Figure 2.5:	Dedicated bus-based access to core-based system	15
Figure 2.6(a):	Non-partitioned test scheduling technique	16
Figure 2.6(b):	Partitioned test scheduling technique	16
Figure 2.6(a):	Pre-emptive test scheduling technique	16
Figure 3.1:	2-D Rectangular bin packing problem	22
Figure 3.2:	3-D Rectangular bin packing problem	23
Figure 3.3:	Insertion of a new element, d, into the arbitrary position through sequence pair formation	25
Figure 4.1(a):	Modified Wrapper Architecture configuration for parent core in $INTEST_P$ mode	38
Figure 4.1(b):	Modified Wrapper Architecture configuration for parent core in $INTEST_C$ mode	39
Figure 5.1(a):	Decrease in testing time (in clock cycles) with increase in TAM width for d695	47
Figure 5.1(b):	Decrease in testing time (in clock cycles) with increase in TAM width for d281	47
Figure 5.1(c)	Decrease in testing time (in clock cycles) with increase in TAM width for p22810	47

Figure 5.1(d)	Decrease in testing time (in clock cycles) with increase in TAM width for p34392	48
Figure 5.1(e)	Decrease in testing time (in clock cycles) with increase in TAM width for p93791	48
Figure 5.1(f)	Decrease in testing time (in clock cycles) with increase in TAM width for a586710	48
Figure 5.4(a):	Testing time versus TAM width obtained from all methods and proposed method for p22810 ITC'02 SOC benchmark	59
Figure 5.4(b):	Testing time versus TAM width obtained from all methods and proposed method for p34392 ITC'02 SOC benchmark	60
Figure 5.4(c):	Testing time versus TAM width obtained from all methods and proposed method for p93791 ITC'02 SOC benchmark	61
Figure 5.4(d):	Testing time versus TAM width obtained from all methods and proposed method for a586710 ITC'02 SOC benchmark	62
Figure A:	Decrease in length of longest wrapper scan chain with increase in TAM width for core 6 of p93791 ITC'02 SOC benchmark	70

LIST OF TABLES

Table 5.1:	Lower bounds on testing time for twelve ITC'02 SOC benchmark	46
Table 5.2.1(a):	Results of non-interactive design transfer model for p22810	49
Table 5.2.1(b):	Results of non-interactive design transfer model for p34392	50
Table 5.2.1(c):	Results of non-interactive design transfer model for p93791	50
Table 5.2.1(d):	Results of non-interactive design transfer model for a586710	50
Table 5.2.2(a):	Results of interactive design transfer model for p22810	51
Table 5.2.2(b):	Results of interactive design transfer model for p34392	51
Table 5.2.2(c):	Results of interactive design transfer model for pp93791	52
Table 5.2.2(d):	Results of interactive design transfer model for a586710	52
Table 5.3.1(a):	Results of modified wrapper design in non-interactive design transfer model for p22810	53
Table 5.3.1(b):	Results of modified wrapper design in non-interactive design transfer model for p34392	53
Table 5.3.1(c):	Results of modified wrapper design in non-interactive design transfer model for p93791	54
Table 5.3.1(d):	Results of modified wrapper design in non-interactive design transfer model for a586710	54
Table 5.3.2(a):	Results of modified wrapper design in interactive design transfer model for p22810	55
Table 5.3.2(b):	Results of modified wrapper design in interactive design transfer model for p34392	55
Table 5.3.2(c):	Results of modified wrapper design in interactive design transfer model for p93791	55
Table 5.3.2(d):	Results of modified wrapper design in interactive design transfer model for a586710	56
Table 5.4(a):	Results of proposed test architecture method for p22810	57
Table 5.4(b):	Results of proposed test architecture method for p22810	57
Table 5.4(c):	Results of proposed test architecture method for p22810	58
Table 5.4(d):	Results of proposed test architecture method for p22810	58
Table A:	Results of *design_wrapper* algorithm for core 6 of p93791	69

ABBREVIATIONS

IC	Integrated Circuit
SOC	System on Chip
ATE	Automatic Test Equipment
DFT	Design For Testability
BIST	Built-in Self Test
PCB	Printed Circuit Board
TAM	Test Access Mechanism
UDL	User Defined Logic
HDL	Hardware Description Language
IP	Intellectual Property
VLSI	Very Large Scale Integration
ITRS	International Technology Roadmap of Semiconductor
ILP	Integer Linear Programming
MWR	Modified wrapper design

CHAPTER 1- INTRODUCTION

1.1 INTRODUCTION & MOTIVATION

Integrated circuits (ICs) are embedded now-a-days in a wide range of products and systems, from consumer electronics and medical equipment to automotive and aviation systems, which usually require high availability and where the cost of failures can be immense. There has been an amazing development of ICs. The first IC available commercially was produced by Fairchild Semiconductor Corp. in 1961; it contained one transistor, three resistors and one capacitor. The everlasting improvements in semiconductor fabrication technology have led to ICs with billions of transistors. Such large ICs can contain all necessary electronic circuitry for a complete system and are referred to as SOCs. A typical SOC consists of components such as processors and peripheral devices including data transformation engines, data ports, and controllers [1].

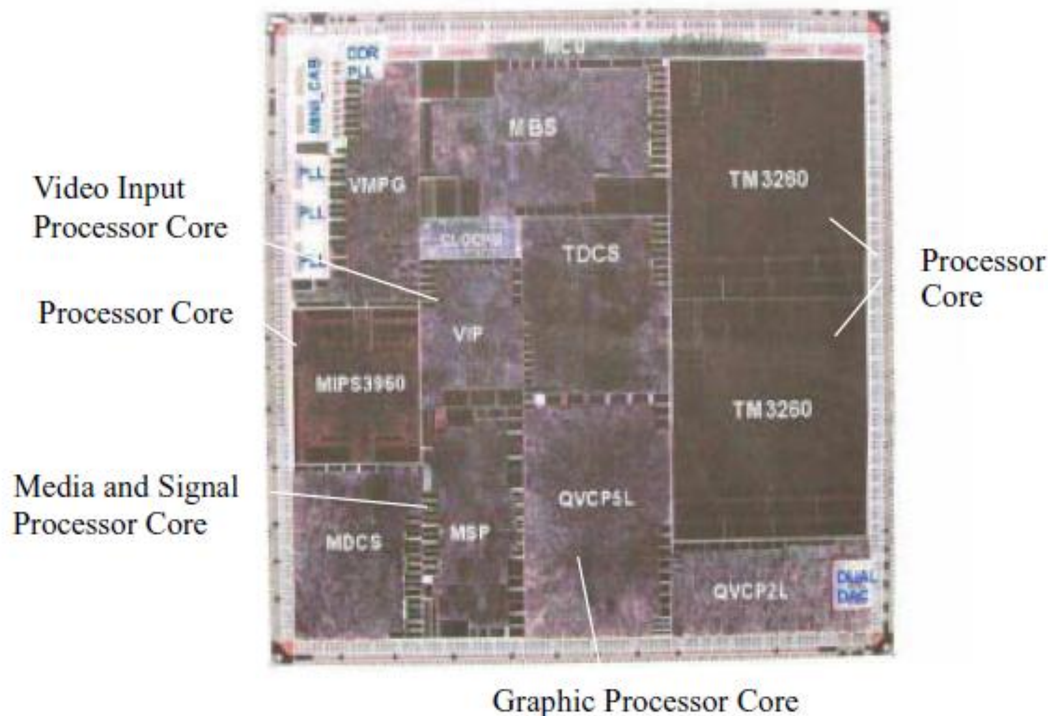


Figure 1.1: Core-based SOC layout, PNX 8550 [4]

ICs can be extremely complex and time-consuming to design. In order to meet short time-to-market requirements, it is therefore common to make use of a modular core-based design approach where a system is composed of pre- designed and pre-verified blocks of

logic, so-called cores. The cores can be designed in-house or bought from core vendors, and it is the task of the system integrator to integrate them into a system. The IC fabrication process is far from perfect and defects such as shorts to power or ground, extra materials, etc., may appear as faults and cause failures. Therefore, each manufactured IC needs to be tested. The aim of fabrication test is to ensure that the fabricated IC is free from manufacturing defects. The general approach to test is to apply test stimuli and compare the produced responses against the expected ones. Due to the complexity of the test process, a design approach, so-called design-for-testability (DFT), aimed at making the IC more easily tested has been proposed. As each fabricated IC is tested, it is important to minimize the test application time. For example, let us assume an IC that has a test application time of 10 seconds and is fabricated in 1 million copies. The total test application time for these ICs will be 116 days. A saving of 1 second per IC leads to a reduction of the total test time with 12 days.

For modular designs, it is possible to perform modular testing where each core is tested as an individual unit. Modular test is an attractive test solution since not only the cores are reused but also their test-data. However, the designers at the core vendor have little or no information about where their cores will be placed on a SOC. It is, therefore, usually assumed that the core is directly accessible and it becomes the task of the system integrator to ensure that the logic surrounding the core allows the test stimuli to be applied and the produced responses to be transported for evaluation. In modular testing, the system integrator is faced by a number of challenges, such as test-architecture design and test scheduling.

The increasing cost for IC testing is in part due to the huge test-data volume (number of bits), test stimuli and expected responses, which can be in the order of tens of gigabits. The huge test-data volume leads to long test application time and requires large tester memory. The 2007 International Technology Roadmap for Semiconductors (ITRS) predicts that the test-data volume for ICs will be as much as 38 times larger in 2015 than it is today [6]. Furthermore, the number of transistors in a single IC is growing faster than the number of I/O pins, i.e., the ratio of transistors per I/O pin is growing. This trend leads to increased test application time since more test data have to be applied through the limited number of I/O pins. The 2007 ITRS predicts that the test application time for ICs will be about 17 times longer in 2015 than it is today [6].

The importance of reducing the cost of test is further motivated by comparing the test cost with the cost of fabrication. Figure 1.2 is adapted from ITRS 1999 [2] and ITRS 2001 [4], and shows how the relative cost of test grows compared to the fabrication cost per transistor. As can be seen in Figure 1.2, the actual cost of test is almost constant while the cost of fabrication has been dramatically reduced over the recent years. Today, the cost of test is a significant part of the overall manufacturing cost (including the cost of fabrication and the cost of test). The high test cost for core-based SOCs can be reduced by adequate test planning, which includes:

- test-architecture design
- test scheduling
- test-data compression

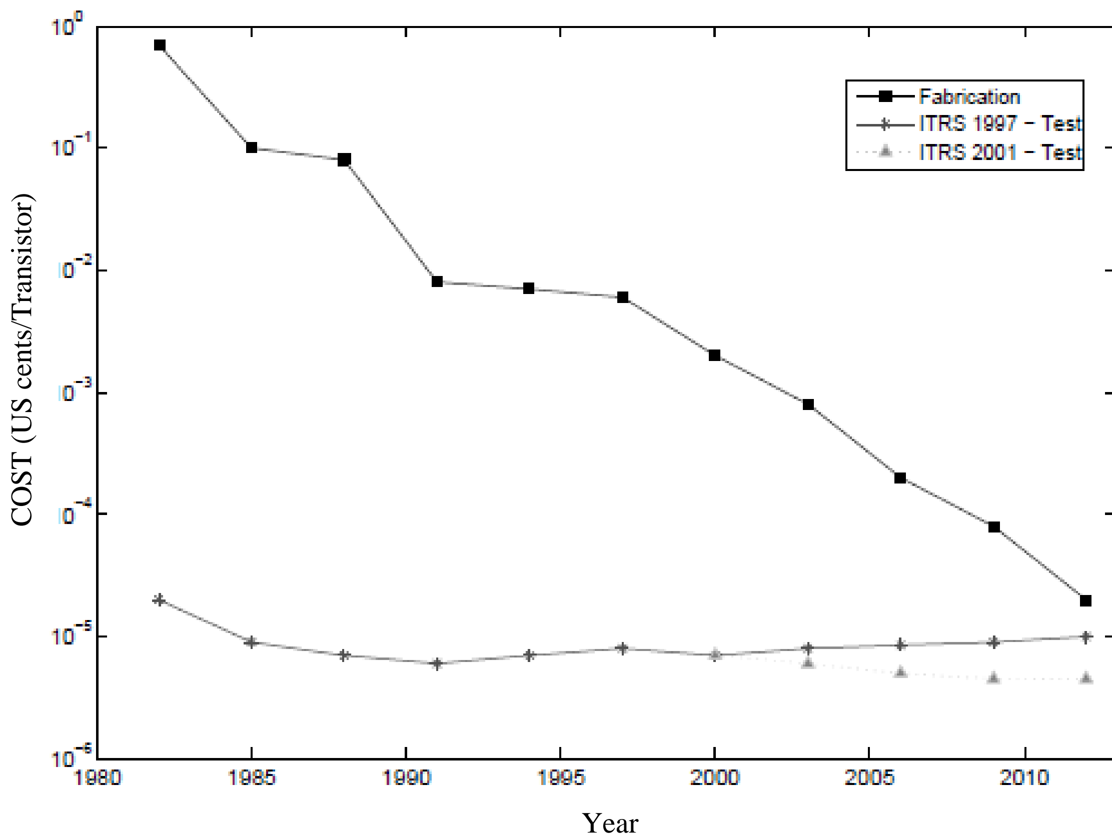


Figure 1.2: Trend in test cost versus fabrication cost per transistor [23]

1.2 SOC TEST PROCESS

An SOC test is essentially a composite test comprised of the individual tests for each core; the users defined logic (UDL) tests, and interconnect tests. Each individual core or UDL test may involve surrounding components and may imply operational constraints (*e.g.*, safe mode, low power mode, bypass mode), which necessitate special isolation modes. Testing is performed by applying test stimuli, consisting of a set of test vectors, to the chip and then comparing the test responses with the expected responses, as illustrated in Figure 1.3.

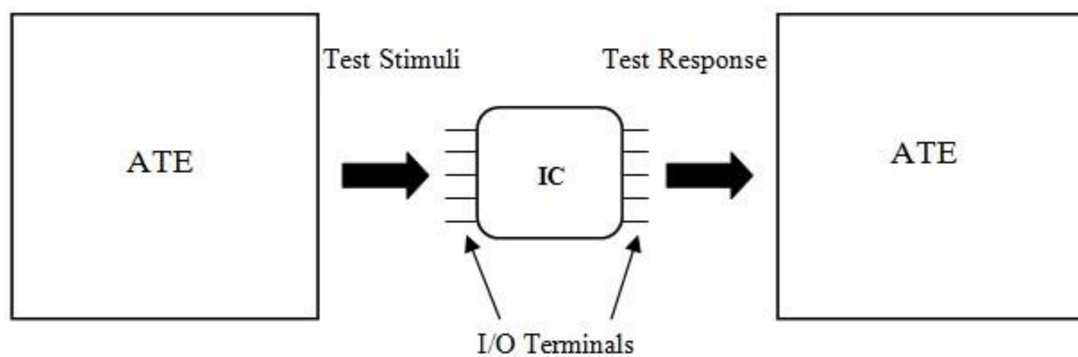


Figure 1.3: SOC test process [23].

In the first approach, the test stimuli are usually applied by using a special machine called automatic test equipment (ATE).The same machine is used to compare the test responses with the expected ones. A difference between the two responses indicates that a fault has been detected on the IC.

In the second approach for testing embedded cores: Built-in self-test (BIST) and applying and observing pre-computed test sets at the core terminals. In the case of BIST, an on-chip engine is responsible for generating the test stimuli and for observing the responses. Although BIST obviously simplifies the test task for the system integrator, there are several limitations: Most cores are not suitable for BIST, i.e. they contain logic that cannot be tested by randomly generated tests. The second approach is to use pre-computed test sets, provided usually by the core vendors, which are applied and observed via the core's terminals. The core vendor has little or no information about where their cores will be placed on the chip. Therefore they assume that all core terminals are directly

accessible. The task of the system integrator is to ensure that the logic surrounding the core allows the tests to be applied, i.e. designing a suitable test access mechanism (TAM).

1.3 CORE-BASED SOC TESTING

The cores in a system can have different origin and they can be classified into three categories:

- Hard Cores
- Soft cores
- Firm Cores

Hard cores are given as layout files that cannot be modified. This type of cores are highly optimized for area and performance, and synthesized to a specific technology. And also the test sets (test stimuli and test responses) are given. Soft cores, on the other hand, are given in a hardware description (HDL) language, hence, technology independent, and the soft cores can easily be modified compared to hard cores. The soft core specification has to be synthesized and optimized, and also it is required to perform test generation. Firm cores are given as technology-dependent netlists using a library with cells that can be changed according to the core integrator’s need.

SOC test development is especially challenging for several reasons. Embedded cores represent intellectual property, and core vendors are reluctant to divulge structural information about their cores to users. Thus, users cannot access core netlists and insert design-for-testability (DFT) hardware that can ease test application from the surrounding logic. Instead, the core vendor provides a set of test patterns that guarantees specific fault coverage. These test patterns must be applied to the cores in a given order, using a specific clocking strategy. Care must often be taken to ensure that undesirable patterns and clock skews are not introduced into these test streams. Furthermore, cores are often embedded in several layers of user-designed or other core-based logic and are not always directly accessible from chip I/Os. Propagating test stimuli to core inputs may therefore require dedicated test transport mechanisms. Moreover, it is necessary to translate test data at the inputs and outputs of the embedded-core into a format or sequence suitable for application to the core.

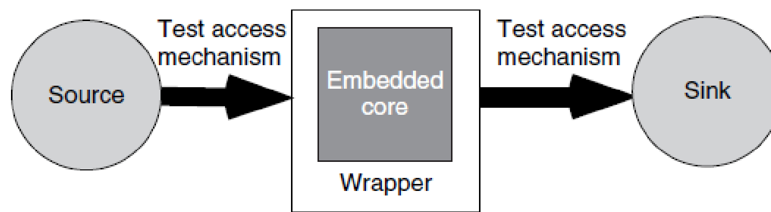


Figure 1.4 : Overview of three elements in an embedded-core test approach [23]

A conceptual architecture for testing embedded core-based SOCs is shown in Figure 1.4 [4]. It consists of three structural elements:

1. Test pattern source and sink: The test pattern source generates the test stimuli for the embedded cores, and the test pattern sink compares the response(s) to the expected response(s).
2. Test access mechanism (TAM): The TAM transports test patterns. It is used for the on-chip transport of test stimuli from the test pattern source to the core under test, and for the transport of test responses from the core under test to a test pattern sink.
3. Core test wrapper: The core test wrapper forms the interface between the embedded core and its environment. It connects the terminals of the embedded core to the rest of the integrated circuit and to the TAM.

The system consists of a set of cores where the cores are all on the same "level"; there is no hierarchy. However, a core can be embedded within a core. A parent core may have several child cores embedded inside of it. Furthermore, each child core can in turn be a parent core with embedded child cores (Figure 1.5). From a modelling perspective, embedding a core in core (parent-child) makes it a bit more complicated.

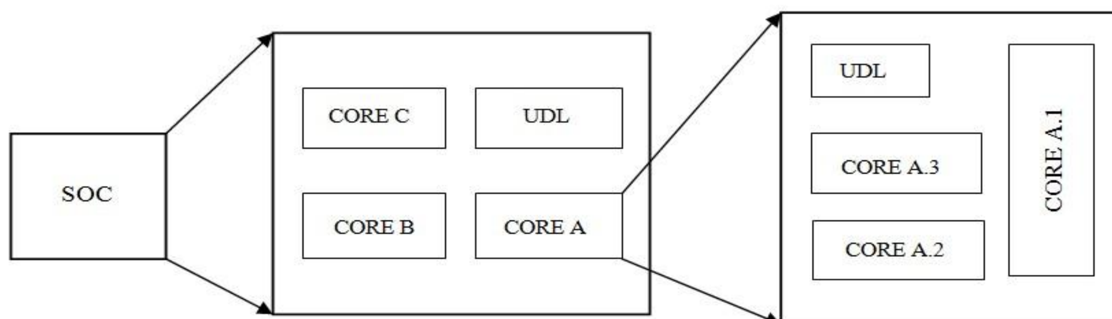


Figure 1.5 : System on Chip Hierarchy [23]

1.7 CONSTRAINTS IN SOC TEST PROCESS

The problem in SOC testing is that the test application time increases as the technology makes it possible to design highly complex chips. These complex chips include a high number of fault sites, which need a high test data volume for testing, and the high test data volume leads to long test application times. For modular core-based SOCs where each module has its distinct tests, concurrent application of the tests can reduce the test application time dramatically, as compared to sequential application. However, when concurrent testing is used, resource conflicts and constraints must be considered. The conflicts described in [7] are:

a) Volume of test data, tester channel capacity and testing time

   The volume of test data is determined by the chip complexity and it grows rapidly as more IP cores are integrated into a single SOC. The easiest way to deal with increased volume of test data is to upgrade the tester memory and use more tester channels to increase test concurrency, however this is infeasible since it will prohibitively increase the ATE cost. A more cost effective approach is to use test data compaction and/or compression. Test data compaction reduces the number of test patterns in the test set (by discarding test patterns that target faults detected by other patterns in the test set) and test data compression decreases the number of bits (that need to be stored for each pattern) and uses dedicated decompression hardware (either off or on-chip) for real-time decompression and application.

b) Heterogeneous IP cores

   Many SOC designs incorporate cores that use different technologies, such as random logic, memory blocks, and analog circuits. For systems assembled on printed circuit boards (PCBs) each of these components was tested using different types of dedicated ATEs (e.g., digital, memory or analog testers). For SOC testing one can use generic high-performance mixed-signal ATEs, however their high production cost brings limited benefits to complex designs, since cores using heterogeneous technologies still need to be tested sequentially, thus lengthening the testing time and ultimately raising the manufacturing test cost. In addition, embedded core controllability and observability issues cannot be addressed without dedicated on-chip DFT hardware, whose necessity justifies a shift toward BIST. The use of different BIST circuitry for the appropriate technologies (logic, memory

or analog BIST), increases both testability and test concurrency of SOCs comprising heterogeneous IP cores.

c) TAM wire assignment

Each test must be assigned a start time and an end time as well as TAM wires in the case of tests stored in the ATE. However, TAM wire are limited due limited number of input, output and bidirectional pins of SOC and the cores within the SOC have higher number of input, output and bidirectional pins which puts a serious constraint over the TAM bandwidth utilization and amount to test data that can be transferred to the core.

d) Power Constraints

Power dissipation is becoming a key challenge for the deep sub-micron CMOS digital integrated circuits. Placing more and more functions on a silicon die has resulted in higher power/heat densities, which imposes stringent constraints on packaging and thermal management in order to preserve performance and reliability. There are two major sources of power consumption in CMOS VLSI circuits: dynamic power dissipation, due to capacitive switching, and static power dissipation, due to leakage and sub-threshold currents. Dynamic power dissipation dominates the chip power consumption for digital CMOS technology. Dynamic power dissipation can be analyzed from two different perspectives. Average power dissipation which stands for the average power utilized over a long period of operation, and peak power dissipation which is the power required in a very short time period such as the power consumed immediately after the rising or falling edge of the system clock. When considering SOC test, to achieve high fault coverage with less test data, the test patterns are usually uncorrelated. This means the switching activity during test can differ from that during functional operation. In most cases, the testing power consumption is the higher one. So test time power dissipation has to be respected in order not to damage the chip.

e) Multiple test sets

A core, for instance, can be tested using one test set generated by an LFSR and another test set stored in the ATE. The selection of the test sets for the cores in the system affects the total test application time and the ATE usage.

f) <u>Interconnection (cross-core) testing</u>

The interconnections between wrapped cores must also be tested. The problem with interconnection testing is that the interconnections and logic between wrapped cores do not have their own dedicated wrapper and hence no direct connection to the TAM. As wrappers are the required interface to the TAM, test stimuli and test responses for interconnection tests have to be transported to and from the TAM via wrappers at wrapped cores.

g) <u>Precedence constraint</u>

A particular order has sometimes to be enforced between some of the tests or core under test.

h) <u>Hierarchical embedded cores</u>

In a core-based environment, cores are embedded in the system. However, a core can also be embedded in a core. One or more child cores can be embedded in a parent core. The embedding of cores in cores results in conflicts at testing. The testing of a parent core can usually not be performed at the same time as the testing of a child core.

1.5 PROBLEM DESCRIPTION

Due to functional and performance requirements, modern SOC designs are not limited to only on level design and test hierarchy (SOC and cores); instead, they contain multi levels of hierarchy. A hierarchical System-on-chip (SOC) for example, [8] and [5] describes SOCs for digital video, for which the design is partitioned into design and test units called *megacores*, which in turn consists of multiple cores. Design units that contain other cores are referred to as hierarchical cores, while cores that do not contain other cores are referred to as flat cores.

Modular testing is generally known to reduce the test length, i.e. the SOC test application time and the vector memory required on the tester. Various wrapper design and wrapper design optimization [10-12], test access infrastructure and optimization [13-20], and test scheduling algorithms along integrated along with wrapper and TAM design [26-29] have been described in the literature. Unfortunately, all these methods unrealistically assume that there is no hierarchy inside the embedded cores. Even if the benchmark SOCs used

contain hierarchical cores, these optimization techniques treat all cores in the SOC as if at same level of hierarchy. Hierarchy imposes constraints on the manner in which tests must be applied to the parent cores and their child cores. Wrapper, TAMs, and test schedules created as if the SOC was non-hierarchical are typically not valid for hierarchical SOCs. Therefore, test solutions proposed by the methods in [10 - 29] are not directly applicable to real life SOCs with hierarchical cores. Hence, there is need for wrapper design/ TAM optimization techniques and test schedule that can handle hierarchical SOCs.

1.6 THESIS OVERVIEW

This chapter has introduced the reader the design paradigm "System-on-Chip" and need to testing electronic chips. Further the organisation of thesis is as follows:

- Chapter 2 presents some preliminary issues related to SOC testing. These preliminaries are specific for this work, such as system-on-chip model, test access architecture and test scheduling, and the concepts presented will be used throughout this thesis work.
- Chapter 3 gives background information and description of all the related previous work done in core based SOC testing, and gaps in these work has eventual led to conceiving of idea behind pursuing this thesis work.
- Chapter 4 presents work related and provides base to this thesis work i.e. (1) wrapper design/TAM optimization and test scheduling based on Integer Linear Programming (ILP) (2) design and optimization of multi-level TAM architecture for hierarchical SOCs (3) new IEEE P1500 modified test wrapper design for hierarchical cores (4) proposed new test architecture utilizing the modified test wrapper design in interactive design flow using ILP.
- Chapter 5 results of our experiments are presented and comparative analysis is done.
- Chapter 6 concludes this thesis and discuss possible directions of future work.

CHAPTER 2- PRELIMINARIES

2.1 SYSTEM ON CHIP (SOC) MODEL

Nowadays, large system-on-a-chip (SOC) designs commonly use IP cores that are deeply embedded in the system chip and direct access is often impossible. Individual cores have to be tested on a system level after manufacturing and therefore special test access mechanisms (TAMs) are required. Choosing and scheduling test solutions for SOC embedded IP cores is a very complex problem. In order to facilitate reuse of test vectors provided by the core vendor, an embedded core must be isolated from the surrounding logic, and test access must be provided from the I/O pins of the SOC. Test wrappers form the interface between TAM and core, while TAMs transport test data between SOC pins and wrappers. Refer to the Figure 2.1.

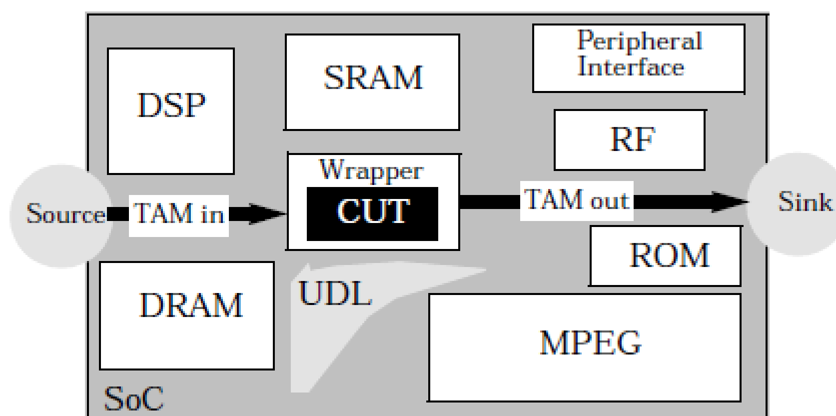


Figure 2.1: SOC model [23]

The arrows inside the SOCs show the TAM which is responsible for test data transport. The three major components of test access architecture are:

1. Test source and sink
2. TAMs
3. Test Wrappers.

The general problem of SOC test integration includes the design of test architecture, optimization of core wrapper, test scheduling wrapper pin assignments.

2.2 WRAPPER DESIGN

Every IP module in SOC is wrapped in a so called TESTSHELL or Wrapper around the core. IEEE P1500 standardises the test wrapper [22] and its interface to one or more TAM. It is developed to help to solve various aspects of core-based testing as:

- Transfer of information about the cores test from the core provider to the core user.
- Access from test stimuli generator and test response evaluator (either ATE or BIST engine) to the core under test.
- Optimisation with respect to multiple criteria (for example: total test time, maximum power dissipation) of the multiple core tests at system chip level.

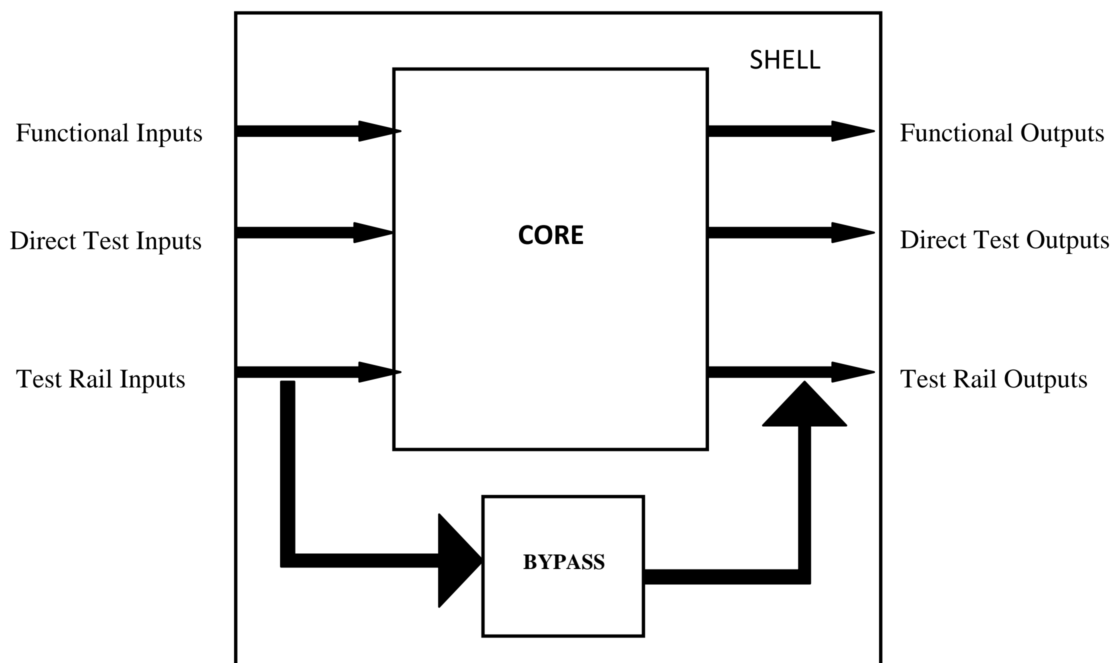


Figure 2.2: IEEE P1500 TESTSHELL/WRAPPER

Wrapper is a small interface layer between IP module and host. Its purpose is to facilitate test expansion as well as core isolation for the core user by creating a standardized interface with bypass. The wrapper allows the core to be run in several modes [22]:

1. Normal mode, in which the wrapper is transparent and does not add any functionality.
2. Internal test mode, in which TAM wires are connected to the core for the transport of test data.
3. External test mode, in which it is possible to test interconnections between cores.

4. Bypass mode, in which the core is bypassed, that is, the core acts as if it does nothing.

2.3 TEST ARCHITECTURE DESIGN

In test architecture design there are two main issues, the *TAM type* and the *architecture type*. In the literature, two different TAM types have been described, the *test bus* and the *test rail* [22]. The test bus is in essence TAM wires connected to the cores on a SOC. Cores connected to a TAM of type test bus can only be tested sequentially. Access to the bus can for instance be implemented by multiplexers or tri-state elements. Figure 2.3 (a) shows the *multiplexing* architecture consisting of one TAM of type test bus.

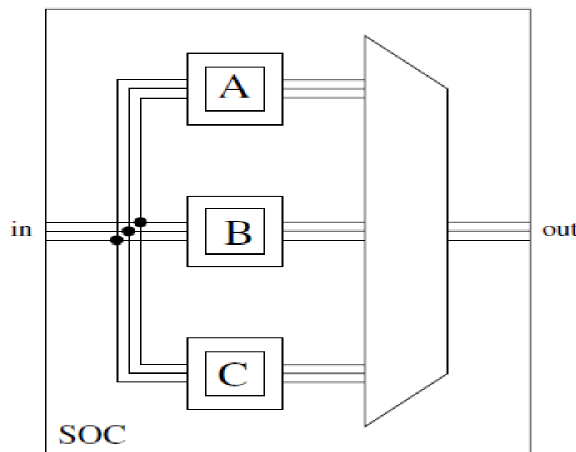


Figure 2.3 (a)

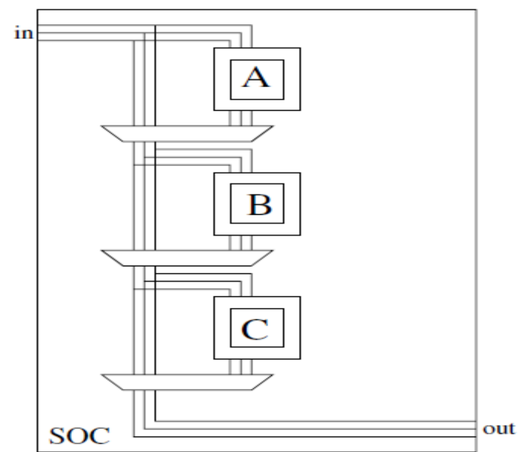


Figure 2.3 (b).

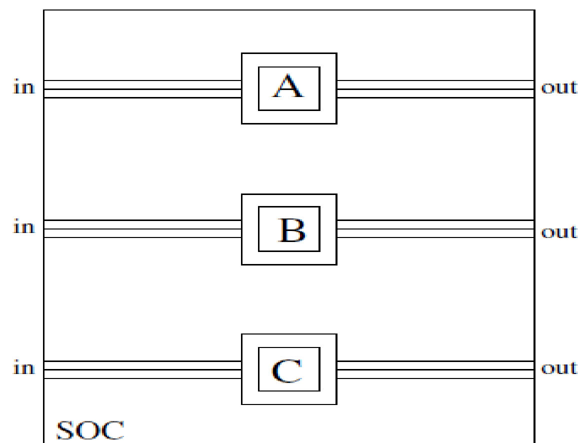


Figure 2.3 (c)

Figure 2.3 Test Architecture Designs (a) Multiplexing Architecture (b) Daisychain Architecture (c) Distribution Architecture

The second TAM type is *test rail*. Cores connected to a TAM of type test rail can be tested simultaneously or sequentially. A simultaneous test of two cores can be done by concatenating the scan chains and wrappers of the cores. Figure 2.3 (b) shows the *daisychain* architecture which consists of one TAM of type test rail.

We distinguish three architecture types. The first type consists of one TAM that connects to all cores. The TAM can be of type test bus or test rail. The multiplexing and daisychain architectures illustrated in Figure 2.3 (a) and Figure 2.3 (b) are two examples. The second type connects each core to its own dedicated TAM. In this case the TAM type is not relevant. An example of this architecture type is the *distribution* architecture, illustrated in Figure 2.3 (c).

The third architecture type is a hybrid combination of the two described types. In this architecture there are one or more TAMs. Each TAM connects to one or more cores in the SOC. With TAMs of type test bus this class of architectures are called *hybrid test bus architectures*. Two common classes of test bus architectures are *fixed-width*- and *flexible-width* test bus architectures. A fixed-width test bus architecture consists of one or several TAMs of type test bus. The cores in an SOC can thus be scheduled sequentially on each TAM. However, more TAMs means that we can test more cores concurrently. Given the total available number of TAM wires we can partition them into several TAMs. In flexible-width test bus architecture we do not distinguish several TAMs. In this case we are able to *fork* and *merge* TAM wires between cores, and thus increase TAM wire utilization.

2.5 TEST SCHEDULING

When the type of TAM is determined the test integrator is faced with another problem, namely in which order the tests for the different cores should be applied. The general problem of test scheduling for SOCs is related to the *NP-Hard*. Test scheduling means that the start time of each test is determined and is usually done in order to minimize some predefined cost function, which often is related to the total test application time. By exploring different start times for each test it is possible to minimize the cost function while ensuring that constraints, such as power consumption and/or hardware overhead,

are not violated, as illustrated in Figure 2.4. Different trade-offs may be considered during the test scheduling. For example, testing several cores in parallel can usually decrease the

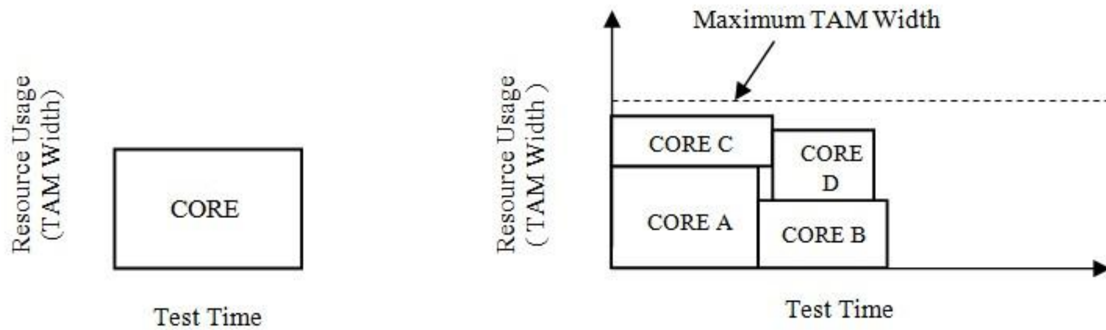


Figure 2.4: Illustration of test scheduling [23]

test time; however it will also increase the power consumption, which potentially will damage the chip. To further explain the scheduling process, a design example consisting of four cores is used (Figure 2.5). The system is tested by applying the tests {$T_{CORE\ A}$, $T_{CORE\ B}$, $T_{CORE\ C}$, $T_{CORE\ D}$} to the cores, {*Core A, Core B, Core C, Core D*}, where $T_{CORE\ A}$ is used to test *Core A*, $T_{CORE\ B}$ to tests *Core B*, and so on. It is assumed that each core has four scan chains, which are connected to a bus-based TAM as illustrated in Figure 2.5. The TAM is designed in such a way that it is possible to apply test stimuli to any two cores in parallel.

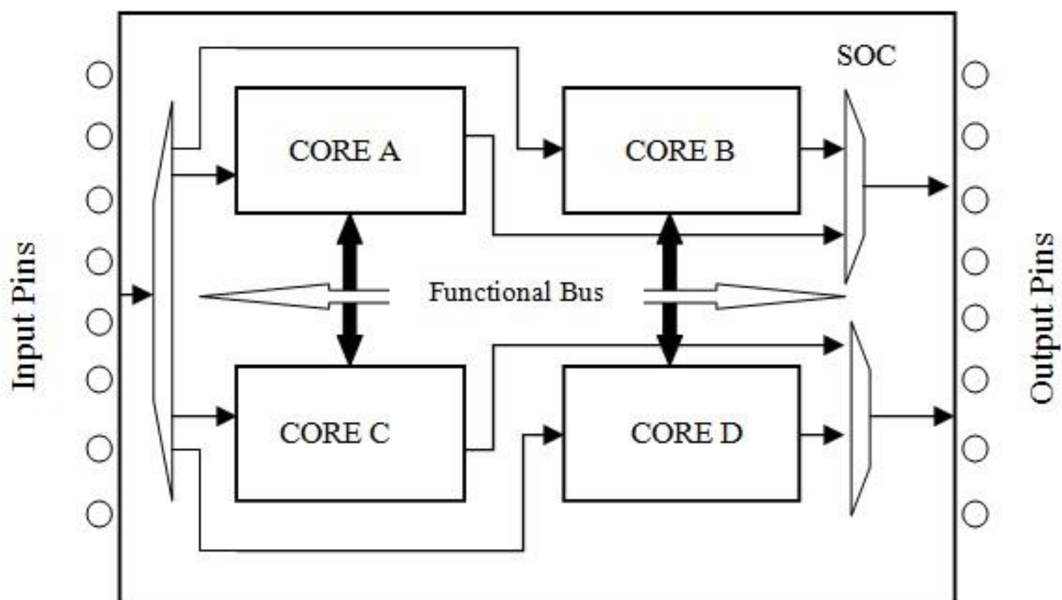


Figure 2.5: Dedicated bus-bases access to core-based system [23]

In this example, it is assumed that the cost function consists of the test application time, which will be minimized without violating the hardware constraint given by the maximum number of TAM wires. Test scheduling techniques can be divided into the following three categories:

- *non-partitioned* testing
- *partitioned* testing with run to completion
- *pre-emptive* testing.

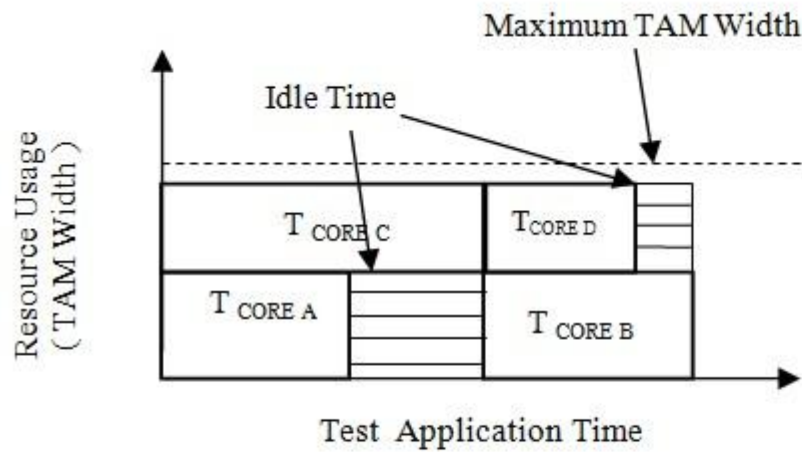


Figure 2.6(a)

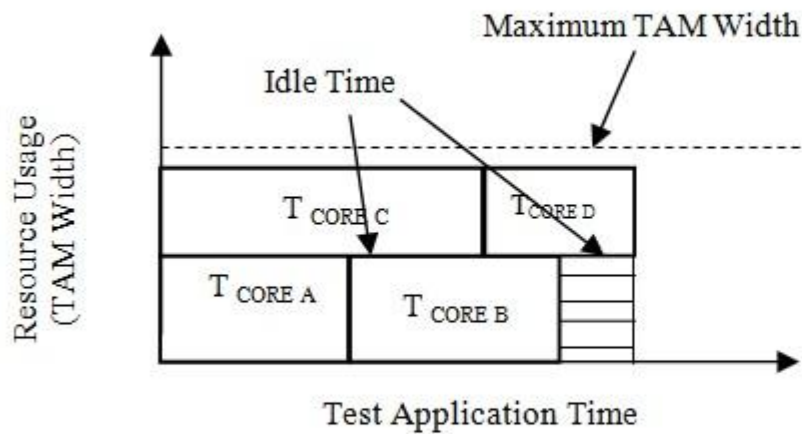


Figure 2.6(b)

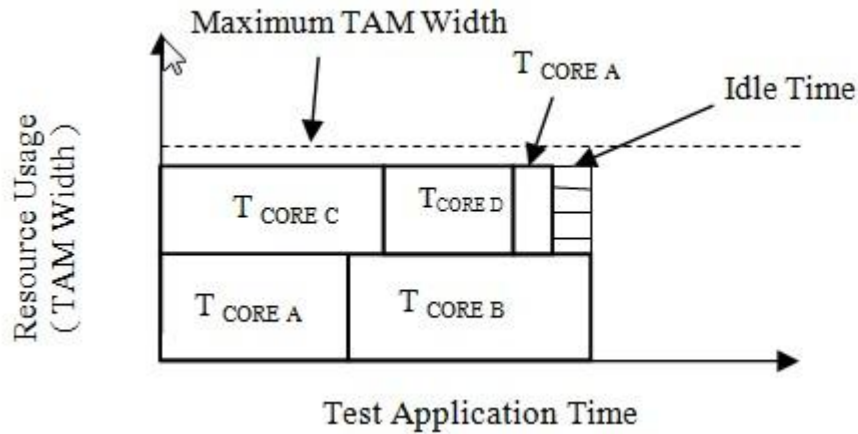


Figure 2.6(c)

Figure 2.6: Three scheduling technique: (a) non-partitioned (b) partitioned (c) pre-emptive [23]

The three test scheduling techniques are illustrated in Figure 2.6. In non-partitioned test scheduling Figure 2.6(a) no new test is allowed to start until all tests in a session are completed. This method produces long test application times due to long periods of time when no core in the system is tested, so-called idle times. In the partitioned technique Figure 2.6(b), tests are allowed to be scheduled as soon as possible, which can decrease the test application time. However, a more advanced test controller is required for the invocation of tests since more possible start times of tests can be used. In order to further optimize the schedule, a pre-emptive test scheduling technique can be used. Such a technique has been proposed in [9]. Pre-emptive test scheduling can be used to reduce the idle time as illustrated in Figure 2.6(c). Pre-emptive test scheduling requires an advanced test controller. Furthermore, it is not applicable to all types of tests. For example, BIST, where the test application is started and then run to completion, is usually not possible to pre-empt.

CHAPTER 3- LITERATURE REVIEW

3.1 TESTING TIME MINIMIZATION PROBLEM

In testing of core-based systems, we have three problems to be solved proposed [24] in are:

1. Protection of IP for cores: A core supplier may give a system designer a set of test vector for a core instead of the detailed information about the core such as internal logic. Several testing methods have been proposed for the protection of IP for cores.
2. Reduction of testing time: Testing time for the core-based system is very long because not only the increasing complexity of the core-based system but the penalty of access to cores makes testing time very long.
3. Standardization of test strategies and test interfaces: Test strategies and test interfaces of each core are different from those of the others. Test strategies and test interfaces of cores should be standardized considering both the protection of IP for cores and the prevention of the increase of testing time for the core-based system.

3.1.1 PROBLEM DEFINITION

There are four reasons for the increase of testing time for core-based system:

1. The increase of transistors on a chip
2. The difference between the increase of transistors and that of external PI/Pos (Primary Inputs/ Primary Outputs). The number of external PI/POs does not increase so rapidly as the number of transistors in the advances of VLSI technology. In other words, the number of external PI/POs is a bottleneck for testing VLSIs.
3. A system designer cannot directly apply test vectors to cores because the core-based system has the glue logic like isolation rings around the cores.
4. Testing method for each core is different from that of the others.

The Problem of test time minimization can be sub-categorized into two problems:

1. Problem of minimizing the core testing time
2. Problem of minimizing core access time

3.1.2 PROBLEM OF CORE TEST TIME MINIMIZATION

The problem of core test time minimization with test scheduling (for concurrent testing of cores) has been mathematically formulated and has been tackled through scheduling algorithms in [16], [20], [21], [25], and [27-29] through optimization of wrapper/TAM [10-13], [15], [17], [19], and [26] and integration of both wrapper/TAM optimization and test scheduling algorithms in [9], [14], [21] and[25-29].

The mathematical formulation of core test testing time minimization problem is defined in [24]. A core-based system can be tested at the system level through three possible ways:

1. Internal testing through BIST
2. External testing through ATE
3. Combination of above two, BIST and ATE

For external testing, the test buses that are used for test access may be shared among multiple cores. If BIST is used, then a core may either be "BIST-ed", in which case it has dedicated BIST logic, or it may simply be "BIST-ready" without containing BIST pattern generators and response monitors. In the latter case, the system integrator may design BIST logic that is shared by multiple cores. In order to minimize the testing time, the test resources in the system (test buses, BIST logic) should be carefully allocated cores, and the tests for the cores should be optimally scheduled.

3.1.2.1 MATHEMATICAL FORMULATION

The test time minimization problem is formulated in [24] is as follows:

Test methodology targets system chips which satisfy the following assumptions:

- A core-based system is constructed by n cores.
- Several sets of test vectors for each core are given. Given sets of test vectors can achieve sufficient fault coverage.
- A set of test vectors is generated for both BIST and external testing. Each set of test vectors is designed to have different ratio of BIST to external testing from others in every core.
- BIST uses a BIST clock with operating frequency and external testing uses test frequency. Operating frequency is usually higher than test frequency.

- Switching testing method in each core (BIST and external testing) does not need overhead time at all.
- External PI/POs are sequentially occupied among cores for external testing. Here, we do not consider the case that more than one core use External PI/POs.
- A set of test vectors for each core is applied directly to the core through a multiplexer. A test vector is applied within a cycle.

For given Vi ***for*** all *i, F and FT* , *find vi which minimizes the testing time T.* Either the total time of external testing for '*n'* cores, or the maximum testing time in '*n'* cores is testing time for core-based. The total time used in external testing, *TET,* is described below:

$$T_{ET} = \sum_{i=0}^{n-1} ETC/FT \quad . . (1)$$

And testing time for core *i*, namely, T_{vi} is below:

$$T_{vi} = \frac{BC(vi)}{F} + \frac{ETC(vi)}{FT} \quad . . (2)$$

T is formulated as below:

$$T = \max(\, T_{ET},\ \max_{i=0}^{n-1} T_{vi} \,)$$
$$= \max\left[\, \sum_{i=0}^{n-1} \frac{ETC(vi)}{FT},\ \max_{i=0}^{n-1} \left\{\frac{BC(vi)}{F} + \frac{ETC(vi)}{FT}\right\}\right] \quad . . (3)$$

Where,

- V_i : A set organized by several sets of test vectors of core *i.*
- v_i : A set of test vectors for core *i,* v_i is an element in V_i, which consists of BIST part and external testing part and satisfies required fault coverage $v_i \varepsilon V_i$
- *BC(wi)* : The number of cycles in BIST for a set v_i.
- *ETC(wi)* : The number of cycles in external testing for a set v_i.
- *F* : Operating frequency. It is used for BIST.
- FT : Test frequency for a core-based system. It is used for external testing.
- *TET* : The total time that external PI/POs are used in testing the core-based system.

- T_{vi} : Testing time for a core *i* by U_{vi} .
- T : Testing time for a core-based system.

The testing time minimization problem is solved by searching determinate variables $v_0, v_1, v_2, \ldots, v_{n-1}$ to minimize the objective function T. This is typically combinatorial optimization problem and can be solves by popular procedures of combinatorial optimization [15-17], [19], [21], and [24-29].

3.1.3 PROBLEM CORE ACCESS TIME MINIMIZATION

A major difficulty concerns accessibility of embedded cores from the I/O terminals of the system. There are three main approaches to deliver test patterns to an individual core and observing core responses. One approach uses extra test circuitry around the core to *isolate* the core during test. For example, an extra multiplex is placed at each core input so that the input is directly accessible from the system input. Alternatively, inserting a partial isolation ring, similar but cheaper than a full scan chain, around the core inputs and outputs so that they can be accessed serially. The second approach uses a *transparency* or *bypass* mode for embedded cores to reduce the problem to one of finding paths from the system inputs to the core inputs and from the core outputs to the system outputs. A similar concept, called *transfer,* was used to transport test data at different levels of circuit hierarchy while a *test protocol* formally specified how this transportation is executed for a core under test. The third approach is based on test bus architectures by which the cores are isolated from each other in test mode using a dedicated bus or a flexible *TESTRAIL* around the cores to transfer test data.

3.1.3.1 MATHEMATICAL FORMULATION

Each core has its own wrapper and cores are independently of wrappers of other cores. The time required to apply the entire test set to a core depends on time to access the core through depends on (i) number of input, output and bidirectional pins available (ii) access elements used in the wrapper design and TAM. Hence, the core access time can be formulated as:

$$T = (1 + \max \{S_i, S_o\}) . p + \min \{S_i, S_o\} \quad . . (4)$$

Here,

p = number of test patterns

$S_i(S_o)$ = length of the longest wrapper scan-in (scan-out) chain for the core

3.2 TEST ACCESS ARCHITECTURE OPTIMIZATION

Most of the earliest work to reduce the testing time of core-based system had focussed on the design of efficient *Test Access Architecture*, i.e. Wrapper design/ TAM optimization as mentioned [25] and was proven equivalent to *NP*-complete. An efficient wrapper design [10], and wrapper design algorithm [11] was developed. It was developed according total the access requirements of various types of core terminals like functional access, as well as for core-internal and external testing. An Integer Linear Programming (ILP) model for optimally assigning cores test bus known as the "*test bus assignment problem"* was developed in [26] by combining the optimal distribution of test buses among individual test buses of cores. It gave designers design options so as to make appropriate choice which includes assignment of test buses, distribution of a given test data bandwidth among multiple test buses, and determining the amount of test data bandwidth required. A new core clustering technique (concurrent testing) based on *"2-D rectangle packing problem"* for wrapper/TAM co-optimization based on ILP modelling with flexible test-bus width through use of *"Pareto-Optimal points"* along with several heuristic to minimize test time was proposed in [16], [17] and [18] shown in Figure 3.2 below.

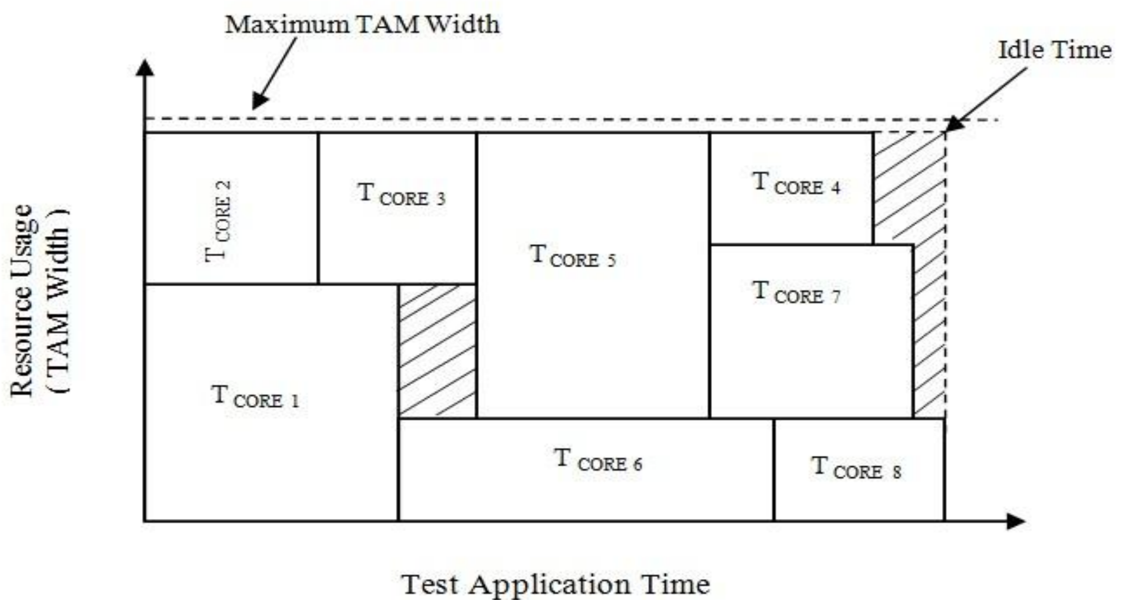


Figure 3.1: 2-D Rectangular bin packing problem [15].

Later, in [19] a core clustering technique for concurrent testing of cores in core-based system was formulated "*3-D rectangle bin packing problem"* simultaneously modelling the optimization of test time, test bus width and power dissipation shown in Figure 3.3.

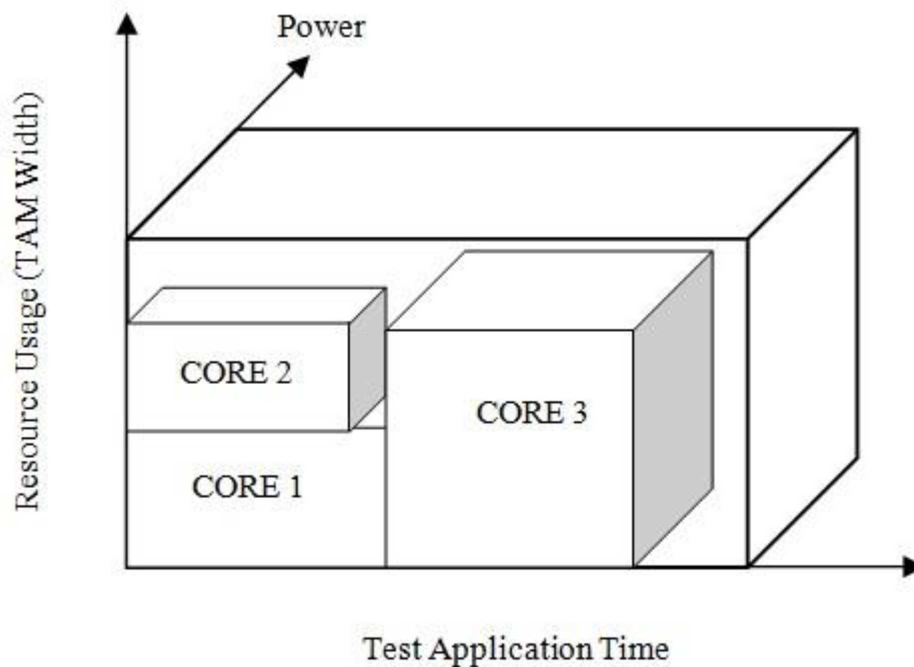


Figure 3.2: 3-D Rectangle bin packing problem [19].

In [21] an integrated wrapper design/TAM co-optimization based on pareto-optimal bus-width architecture for test volume reduction along with resource, precedence and power constraint driven pre-emptive scheduling heuristic was presented modelling test time minimization as 2-D rectangle bin packing problem as presented earlier in [16], [17] and [18]. The method was able to achieve significant reduction in test time along with reducing in area required by the wrapper and test-buses.

3.3 TEST SCHEDULING

The general problem of test scheduling is related to *NP*- complete and solved through various combinatorial optimization scheduling algorithms integrated with wrapper design and TAM optimization algorithms [9], [16], [21], [24] and [26-29]. Optimization algorithms are given below:

3.3.1 OPEN SHOP SCHEDULING ALGORITHM

First, it is shown in [25] that the test scheduling problem is equivalent to open-shop scheduling [26]. In open-shop scheduling, we are given a *shop* consisting of *m* processors,

a set of *J* jobs, each job *j* ε J consisting of m tasks $t_1[j]$, $t_2[j]$,........, $t_m[j]$, and a length $l(t) \geq 0$ for each task. A schedule for an m-shop is a set of *m* processor schedules, one for each processor in the shop. These schedules must be such that no job is processed simultaneously on more than one processor. The Finish *time* of a schedule is the latest completion time of the individual processor schedules. The objective in open-shop scheduling is to minimize the finish time.

In order to establish equivalence between test scheduling for core-based systems and open-shop scheduling, we view the test sets for the cores as jobs. Each job consists of two tasks, corresponding to the external test and BIST components of the test set, respectively. For the problem instance being considered in this section, $m = 2$, i.e. there are two processors in the system, corresponding to the external test bus and the BIST resource, respectively. An optimal schedule, i.e. one with the least finish time, which guarantees the shortest testing time for the core-based system is achieved for a 2-shop scheduling problem with computational time $O(n)$.

3.3.2 SIMULATED ANNEALING ALGORITHM

Simulated Annealing (SA) algorithm [27] is used to solve the 2-D bin packing problem that occurs when optimizing core wrapper design /TAM [15] for test scheduling to achieve optimal SOC test application time. During the process of 2-D packing, the best configuration for every core is selected by SA algorithm to minimize the test application time. To represent test schedules, a data structure is used called a *sequence pair* that is used for floor planning. SA based procedure proposed in [27] leads to better test schedules than the ones found in [20] for a comparable run-time.

3.3.3 RAIN (RAndom INsertion) ALGORITHM

When a core is represented by a rectangle in [16], [17], and [18] the height of the rectangle is the TAM width assigned to the core, and the width is the test application time for that TAM width. To schedule the cores represented by rectangles, sequence pair representation is used. In SOC test scheduling, [27] used sequence pair representation shown in Figure 3.3. The relative position of rectangles is not changed by inserting a new element into arbitrary positions of sequence pair. Therefore it is possible to schedule a new core without breaking down already obtained well-scheduled results [29]. The advantage of sequence pair representation is that only two permutations are sufficient to

represent the placement of rectangles. However, it has disadvantage that as the number of cores increases, the total number of possible sequence pair increases rapidly.

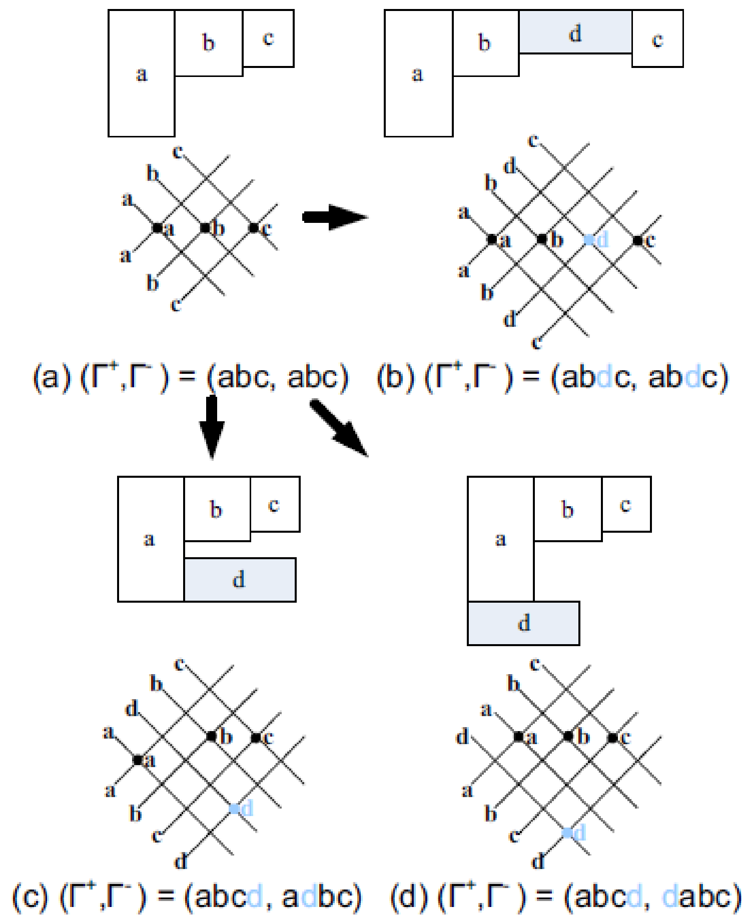


Figure 3.3: Insertion of a new element, d, into the arbitrary position through sequence pair formation [27]

3.3.4 ANT COLONY ALGORITHM

Ant algorithm or Ant Colony Optimization (ACO) is a population-based approach and realizes an adaptive and social behaviour of ants of finding the best route to the food source from the nest by indirect communications between ants using a chemical substance called pheromone. In other words, ants leave a pheromone trail behind while moving, other ants can smell this pheromone, and follow it. In ACO, artificial ants stochastically build new solutions using a combination of heuristic information and an artificial pheromone trail. This pheromone trail is reinforced according to the quality of the solutions built by the ants. ACO has been proved to be very effective in terms of solving many NP problems. ACO-based test scheduling algorithm [30] increases the probability of finding optimal solutions within shorter periods.

3.3.5 GENETIC ALGORITHM

Evolutionary algorithms (EAs) are a class of search, learning, and optimization methods based on analogies to Darwin's theory of natural selection. Unlike simulated annealing (SA) [27] frequently used in VLSl layout synthesis-every EA processes a population of potential solutions in parallel rather than just a single solution. Each individual in the population is a unique solution. Parents are subjected to stochastic "reproduction" operators that produce offspring.

In [27] GA formulation is used to get sequence of rectangles generated by the wrapper design method. For each in the SOC one rectangle has to be selected from the set of rectangles (wrapper configuration) generated for that core so that total testing time will be less. The order in which the rectangle for a core is selected depends on the average area of the rectangles obtained from wrapper design algorithm used. For this purpose, cores are sorted in terms of decreasing average rectangle area. To create population for new generation 20% best fit chromosomes are directly copied and remaining 80% chromosomes are created using crossover and mutation operators.

3.4 TESTING OF HIERARCHICAL CORE BASED SYSTEM ON CHIP

In most of the prior work [14-22] and [24-29] on wrapper design/TAM co-optimization and test scheduling, the SOC design hierarchy is assumed to be flattened for the purpose of simplification. Hierarchical cores are considered as being at the same level in test mode and the TAM assignment for the sub-cores embedded in a hierarchical core is not limited by design hierarchy. Test wrappers, TAMs and test schedules designed for non-hierarchical SOCs are typically not valid for SOCs with hierarchical cores. The hierarchy imposes a number of constraints on the manner in which the tests must be applied to parent cores and their embedded child cores [31]. In [32], problem of test infrastructure design for hierarchical SOCs has been addressed. It was shown that hierarchy aware test planning method can be used for TAM optimisation for hierarchical SOCs in two practical scenarios (i) wrappers and TAM architectures for the child cores are given and fixed (hard), while the wrappers and TAM architectures for the parent cores are to be determined (soft) (ii) wrappers and TAM for both child and parent cores are assumed to be soft. A multi-level TAM architecture is proposed in [32], which describes how known

methods for flattened SOCs can be used for multi-level TAM optimization (based on ILP formulation for flexible bus width) in hierarchical SOCs.

In [33], test schedule is developed for SOCs consisting of hierarchical embedded cores to minimize the test application time of the SOC under the constraint of design hierarchy. The proposed schedule algorithm is based on 2-D rectangle bin packing problem. However, there is no change in the wrapper and TAM architecture used as in case of flat cores. [34] Integrates the test scheduling, wrapper design and TAM assignment for Hierarchical SOC based 3-D rectangle bin packing problem, however fails to consider the simultaneous testing of parent and child core (hierarchical nature of embedded cores) and therefore the whole modelling is flawed. [35] Further researches over the multi-level TAM architecture proposed in [32] and further optimize them for hierarchical SOCs. In [36], a complete new wrapper architecture for parent cores that has two disjoint test modes for testing of parent and child cores is proposed considering the all the constraint imposed by hierarchy and provides full flexibility to test architecture of SOC.

CHAPTER 4 – PROPOSED TEST ARCHITECTURE

The purpose of this chapter is to give a brief overview of some of the research that has been done related to the work presented in this thesis. In the following sections some key concepts, issues and proposed solutions related to reducing testing time of system chip testing will be discussed which has been used for this thesis work.

4.1 WRAPPER DESIGN/TAM OPTIMIZATION AND TEST SCHEDULING USING INTEGER LINEAR PROGRAMMING

The general problem of SOC test integration includes the design of TAM architectures, optimization of the test wrappers, and test scheduling. The goal is to minimize testing time, area costs and power consumption during testing. The wrapper/TAM co-optimization problem is as follows [12] :

4.1.1 PROBLEM FORMULATION

1. **P_W : Design a *wrapper* for a given core, such that (a) the core testing time is minimized, and TAM width required for the core is minimized.**

   Two priority wrapper optimization problem has been addressed and can be formally stated as: Given a core with n functional inputs, m functional outputs, sc internal scan chains of lengths l_1, l_2, . . . , l_{sc}, respectively, and TAM width w, assign the $n + m + sc$ wrapper scan chain elements to $w^{'} \leq w$ wrapper scan chains such that (i) max{ s_i, s_o } is minimized, where s_i (s_o) is the length of the longest wrapper scan-in (scan-out) chain, and (ii) $w^{'}$ is minimized subject to priority (i). Priority (ii) of $\boldsymbol{P_W}$ is based on the earlier observation that max { s_i, s_o } can be minimized even when the number of wrapper scan chains designed is less than w. This reduced the width of TAM required to connect to the wrapper.

2. **P_{AW} : Determine (i) an *assignment* of cores to TAMs of given widths and (ii) a *wrapper* design for each core, such that SOC testing time is minimized.**

   Given N cores and B TAMs of test widths w_1, w_2, ... , w_B, determine the assignment of cores to TAMs and a wrapper design for each core, such that the testing time is minimized. This problem can be shown to be NP-hard using the techniques presented [13]. However, for realistic SOCs the size of the problem instances were found to be small and could be solved exactly using a Integer

Linear Programming Formulation. This problem has been modelled as, if core i is assigned to TAM j, let the time taken to test core i is given *by* T_i (w_j) clock cycles. The testing time T_i (w_j) is calculated by equation (4) given in section 3.1.3.1. A binary variable x_{ij} (where $1 \leq i \leq N$ and $1 \leq j \leq B$) is used to determine the assignment of cores to TAMs in SOC. Let x_{ij} be a 0-1 variable defined as follows:

$$x_{ij} = \begin{cases} 1, & \text{if core } i \text{ is assigned to TAMs } j \\ 0, & \text{otherwise} \end{cases}$$

The time needed to test all cores on TAM j is given by $\sum_{i=1}^{N} T_i(w_j) \,.\, x_{ij}$. Since all the TAMs can be used simultaneously for testing, the system testing time equals $\max_{1 \leq j \leq B} \sum_{i=1}^{N} T_i(w_j) \,.\, x_{ij}$.

3. **P_{PAW} : Determine (i) a *partition* of the total TAM width among the given number of TAMs (ii) an *assignment* of cores to TAMs of given widths and (iii) a *wrapper* design for each core, such that SOC testing time is minimized.**

   The total system TAM width is assumed to be at most W. It is known that the width of each TAM need not exceed the maximum value w_{max} for any core in SOC. For TAM wider than w_{max}, there is no further decrease in testing time. A mathematical programming model for P_{PAW} is shown below [12]:

   Objective: Min $T = \max_j \{ \sum_{i=1}^{N} T_i(w_j) \,.\, x_{ij} \}$, subject to

   i. $\sum_{j=1}^{B} x_{ij} = 1$, $1 \leq i \leq N$, i.e. every core is connected to exactly on TAM
   ii. $\sum_{j=1}^{B} w_j = W$, $1 \leq j \leq B$, i.e. the sum of all TAM widths is W.
   iii. $w_j = w_{max}$, $1 \leq j \leq B$, i.e., each TAM is at most w_{max} bits wide.

4. **P_{NPAW} : Determine (i) the *number* of TAMs for the SOC, (ii) a *partition* of the total TAM width among the TAMs, (iii) an *assignment* of cores to TAMs, and (iv) a *wrapper* design for each core, such that SOC testing time is minimized.**

   Method of restriction is used to prove that P_{NPAW} is *NP*-hard. The problem P_{NPAW} is defined as consisting of only those instances of P_{NPAW} for which (i) W=2, and (ii) all cores on the SOC have a single internal scan chain and no functional terminals. Hence, each core will have the same testing time on a 1-bit TAM as on

a 2-bit TAM. An optimal solution to P_{NPAW} will therefore always result in two TAMs of width 1-bit each. Problem P_{NPAW} now reduces to that of partitioning the set C of cores on the SOC into two subsets C_1 and $C - C_1$, such that each subset is assigned to a separate 1-bit TAM, and the difference between the sum of the testing times of the cores (on 1-bit TAM) and sum of the testing times of the cores (on 2nd 1-bit TAM) is minimized. Hence the problem can modelled mathematically as :

Objective: Minimize $\sum_{c \in C_1} T_c - \sum_{c \in C - C_1} T_c$, where T_c is the testing time of core c on a 1-bit TAM.

4.1.2 ALGORITHMS

The problem P_W is solved efficiently by approximation algorithm based on Best Fit Decreasing (BFD) heuristic. The algorithm has three main parts [12] :

i. Partition the internal scan chains among a minimal number of scan chains to minimize the longest wrapper scan chain length,
ii. Assign the functional inputs to wrapper scan chains created in part (i),
iii. Assign the functional outputs to wrapper scan chains created in part (ii).

Best Fit Decreasing Algorithm [11]

Inputs: Let T be the maximum test length, a set SC of internal scan chains, { sc_1, sc_2, ... , sc_n) are sorted in non-increasing order and a set $WC \in \{wc_1, wc_2, ... , wc_{sc}\}$ of wrapper scan chain so formed.

Output: (WC_{max}) maximum number of scan chains with maximum test length needed to contain all scan chains.

---

design_wrapper (C, W)

for i = 1 to *sc*

do j = 1

while $l(wc_j) + l(sc_i) > T$

do j = j + 1 ***od*** ;

$wc_j = wc_j \cup sc_i$;

od ;

return max { j | $wc_j \neq \emptyset$ } ;

The problem P_{AW} is efficiently solved using heuristic algorithm based on relationship between TAM width and core testing times calculates for P_W. The pseudo code of the algorithm is as follows [12] :

---

core_assign (B, C, T)

Let ***C*** be the set of cores

Let ***B*** be the set of TAMs

Let ***T*** be the best-known testing time for (***C, B***)

for i = 1 to ***C do***

for j = 1 to ***B do***

$T_i(w_j)$ = ***design_wrapper(C, B);***

od ;

od ;

for j = 1 to ***B do***

$T_j = 0$;

od ;

while C ≠ ∅ do

select TAM $j \in$ ***B***, such that T_j is minimum

if there are two or more such TAMs ***then***

select TAM *j*, such that w_j is maximum

fi ;

select core $i \in$ ***C***, such that $T_i(w_j)$ is maximum

if there are two or more such cores ***then***

select TAM $k \in$ ***B***, such that ($w_k < w_j$ AND w_k is maximum

select core *i*, such that $T_i(w_j)$ is maximum

fi ;

Assign core *i* to TAM *j*

Determine TAM $r \in$ ***B***, such that T_r is maximum

If $T_r \geq T$ ***then***

return SOC testing time *T*

fi ;

C = ***C*** – { *i* } ;

od ;

return SOC testing time T_r ;

The problem P_{PAW} is efficiently solved by algorithm *partition_evaluate (W)* presented below. The algorithm employs three levels of solution-space pruning. Firstly, the number of partitions enumerated is significantly limited by the recursive function *increment.* The pseudo code of algorithm is as follows [12] :

---

partition_evaluate (W)

Let $\boldsymbol{W}$ be the total TAM width

Let $\boldsymbol{C}$ be the set of cores

Let $\boldsymbol{B}_{\mathbf{max}}$ be the upper limit of number of TAMs

for $\boldsymbol{B} = 1$ to $\boldsymbol{B}_{\mathbf{max}}$ ***do***

    Let set of TAMs $\boldsymbol{B} = \{ w_1, w_2, \ldots\ldots, w_B \}$

    Set SOC testing time $T = \infty$

    for TAM $j = 1$ to $(\boldsymbol{B} - 1)$, set $w_j = 1$; ***do*** ;

    set $w_{B-1} = 0$; set $flag = 0$;

    while $flag \neq 1$ ***do***

        increment (B,* B-1, *W) ;

        new SOC testing time T_{new} = ***core_assign (C, B)*** ;

        if $T_{new} < T$ ***then***

            set $T = T_{new}$; set $\mathbf{B_{best}} = \mathbf{B}$;

        fi ;

    od ;

od ;

return $\mathbf{B_{best}}$, T ;

---

increment (**B,** *j,* ***W)***

if $w_j < \left\lfloor \frac{W - \sum_{k=0}^{j-1} w_k}{B-(j-1)} \right\rfloor$ ***then***

    set $w_j = w_j + 1$;

    set $w_B = W - \sum_{k=1}^{B-1} w_k$;

    return ;

else

    if $j = 1$ ***then***

    set $flag = 1$;

    return ;

fi;

else increment (* B, *j-1, W);

fi ;

4.2 DESIGN AND OPTIMIZATION OF MULTI-LEVEL TAM ARCHITECTURES FOR HIERARCHICAL SOCs

The top level SOC is composed of embedded cores as well as embedded megacores obtained from core vendors. A megacore may be supplied by core vendors in varying degrees of readiness for test integration. Three design scenarios for megacores considered is as follows [32] & [37]:

1. Not TAM-ed and not wrapped: In this scenario, the system integrator must design a wrapper for the megacore as well as TAMs within the megacore. The megacore is therefore delivered either as soft core or before final netlist and layout optimization, such that TAMs can be inserted within the megacores.
2. TAM-ed but not wrapped: In this scenario, the megacore contains lower-level TAMs, however a wrapper for it is still required to be designed by the core integrator. Knowledge of the number and length of top-level scan chains as well as testing times of lower-level cores are therefore required by the system integrator to design balanced top-level wrapper scan chains for megacore.
3. TAM-ed and wrapped: In this scenario, TAM-ed megacore for which wrappers have been designed by core-vendor are considered. This scenario is especially suitable for a megacore that was an SOC in an earlier generation. The width of TAM that must be supplied to it is pre-specified. It is assumed that megacores are wrapped by the core vendors prior to design transfer and test data cannot be serialized or parallelized by SOC integrator.

Scenario 3 is considered where megacores are wrapped and TAM-ed by the core vendor, either in the non-interactive or interactive design transfer model.

4.2.1 NON-INTERACTIVE DESIGN TRANSFER MODEL

In non-interactive design transfer model, the core vendor designs and implements TAM architectures for use within the megacores. Width optimization for these lower-level TAMs is performed without input from the SOC integrator, and testing times for mega-

cores are specified prior to design transfer. The test parameters supplied by the core vendor to the SOC integrator for non-hierarchical cores include the number of primary (including bidirectional) I/Os, test patterns, scan chains, and scan chain lengths. The parameters supplied for the mega-cores include only the TAM width and testing time.

4.2.1.1 PROBLEM FORMULATION

$P_{\text{non-int}}$ [32]: Given the test set parameters for the top-level cores and the total TAM width W for the SOC, determine a wrapper design for each core, and a partition of W among the cores in the test schedule, such that the SOC testing time is minimized under the constraints that (i) W is not exceeded at any time, (ii) the megacores receive at least their pre-specified TAM widths, and (iii) parent mega-cores are tested only after their embedded child cores.

4.2.1.2 ALGORITHM

The problem $P_{\text{non-int}}$ is solved using the heuristic algorithm based on ILP formulation depicting non-interactive design transfer for multi–level TAM design flow in hierarchical cores. The pseudo code for the design flow and multi-level TAM is as follow [32]:

---

non_interactive_hierarchical (W, C, b)

Let ***C*** be the megacore *i* such that it contains $\boldsymbol{CC_i}$ embedded cores

Let ***b*** be the number of TAMs

Let $\boldsymbol{W_i}$ be total TAM width supplied to megacore *i*

    for all embedded cores in ***C do***

    Partition $\boldsymbol{W_i}$ among the embedded cores *i*

    Design wrapper for each embedded core using ***design_child_wrapper (W,*** $\boldsymbol{CC_i}$ ***)***

    Determine the testing time $\boldsymbol{T_{CC}}$ of each embedded core

    od ;

Determine the assignment of embedded cores among TAM partition ***b*** using ***child_core_assign (*** $\boldsymbol{W_i}$, $\boldsymbol{T_{CC}}$, ***C).***

Determine the total testing time for embedded cores

Design wrapper for parent core in megacore using ***design_parent_wrapper (W, C)*** and testing time $\boldsymbol{T_P}$

Implement TAM architecture for megacore *i* prior to design transfer

Determine total testing time $\boldsymbol{T_i}$ ($\boldsymbol{w_j}$) of megacore *i.*

---

multi_level_TAM_non_interactive (W, C, b)

Let ***C*** be number of top-level cores

Let ***W*** be total SOC TAM width

Let ***b*** number of TAMs

Partition ***W*** among the cores using a TAM partitioning technique illustrated in section 4.1.2 of this thesis.

Let $\boldsymbol{W}_{\mathbf{i}}$ be the width of TAM pre-specified by core vendor for megacore *i*

for each top-level megacore *i* in SOC ***do***

obtain testing time $\boldsymbol{T}_{\mathbf{i}}\ (\boldsymbol{w}_{\mathbf{j}})$ using ***non_interactive_hierarchical (W, C, b);***

set $T_i (w_j) = T_i$, for $w_j \geq W_i$

set $T_i (w_j) = \infty$, for $w_j < W_i$

od ;

Implement system level TAM architecture.

4.2.2 INTERACTIVE DESIGN TRANSFER MODEL

In the interactive design transfer model, the core vendor once again designs and implements TAM architectures for use within the megacores. However, the system integrator is now able to influence the choice of TAM width supplied to megacores by the core vendors based upon system-level TAM width requirements of other cores. The test parameters for each megacore *i* supplied by the core vendor to the SOC integrator prior to system-level TAM design therefore include a set of 2-tuples $\{(W_i , T_i)\}$, where each tuple represents a potential TAM width–testing time choice for the megacore, and the number of tuples depends on the guidelines from the core user to the core vendor.

4.2.2.1 PROBLEM FORMULATION

P_{int} [32]: Given the test set parameters for the top-level cores and W for the SOC, determine a wrapper design for each core, and a partition of W among the cores in the test schedule, such that the SOC testing time is minimized under the constraints that (i) W is not exceeded at any time, (ii) each mega-core receives *one* of its pre-specified TAM widths, and (iii) parent megacores are tested only after their embedded child cores.

4.2.2.2 ALGORITHMS

The problem P_{int} is solved using the heuristic algorithm based ILP formulation depicting interactive design transfer for multi–level TAM design flow in hierarchical cores. The pseudo code for the design flow and multi-level TAM is as follow [32]:

---

interactives_hierarchical (W, C, b)

Let $\boldsymbol{C}$ be the megacore i such that it contains $\boldsymbol{CC_i}$ embedded cores

Let $\boldsymbol{b}$ be the number of TAMs

Let $\boldsymbol{W_{i,max}}$ be total TAM width supplied to megacore i

for i = 1 to $\boldsymbol{W_{i,max}}$ ***do***

    for all embedded cores in ***C do***

    Partition $\boldsymbol{W_{i,max}}$ among the embedded cores i

    Design wrapper for each embedded core using ***design_child_wrapper (W, CC**i)***

    Determine the testing time $\boldsymbol{T_{CC}}$ of each embedded core

    od ;

od ;

Determine the assignment of embedded cores among TAM partition $\boldsymbol{b}$ using ***child_core_assign (W*i*, T*CC*, C).***

Determine the total testing time for embedded cores

Design wrapper for parent core in megacore using ***design_parent_wrapper (W, C)*** and testing time $\boldsymbol{T_P}$

Implement TAM architecture for megacore i prior to design transfer

Determine the total testing time $\boldsymbol{T_i}$ ($\boldsymbol{w_j}$) of megacore i.

---

multi_level_TAM_interactive (W, C, b)

Let $\boldsymbol{C}$ be number of top-level cores

Let $\boldsymbol{W}$ be total SOC TAM width

Let $\boldsymbol{b}$ number of TAMs

Partition $\boldsymbol{W}$ among the cores using a TAM partitioning technique illustrated in section 4.1.2 of this thesis.

Let $\boldsymbol{W_i}$ be the width of TAM pre-specified by core vendor for megacore i

    for each top-level megacore i in SOC ***do***

    obtain testing time $\boldsymbol{T_i}$ ($\boldsymbol{w_j}$) and $\boldsymbol{W_i}$ using ***interactive_hierarchical (W, C, b);***

    set T_i (w_j) = T_i , for $w_j = W_i$

set $T_i (w_j) = \infty$, for $w_j < W_i$

od ;

Implement system level TAM architecture.

4.3 IEEE P1500 MODIFIED TEST WRAPPER DESIGN FOR HIERARCHICAL CORES

Many system-on-chip (SoC) integrated circuits today contain multiple hierarchy levels for both design and test. Hierarchy imposes constraints on the manner in which tests must be applied to "parent" cores and their "child" cores. A modified wrapper design for parent cores that operates in two disjoint modes for testing of parent and child cores has been proposed [31], [36] & [37]. This approach has an impact on the test architecture and corresponding schedule.

To design wrapper around any core, whether hierarchical or non-hierarchical, all and types of terminals around core boundary needs to be identified. Based on the test access requirements, a parent core, in addition to terminals mentioned [31], has terminals that provide test access to its wrapped child cores. These terminals are inputs and outputs at the parent core level, and are referred as CTAM terminals as they are also connected to the TAM of child cores. These terminals also differ for the test data terminals at the parent core, as they operate in both INTEST and EXTEST modes. Since Test Access Mechanism considered throughout this thesis work is fixed-width test architecture, the INTEST and EXTEST modes for a core are time-multiplexed. Figure 4.1 shows wrapper design for parent core in both INTEST and EXTEST mode.

The two INTEST modes that have been identified in a parent core are [31]:

1. **Parent** INTEST **mode** (INTEST_P): In this mode, parent core internal testing is done. Test data are scanned through the parent core's scan chains, the parent core's wrapper cells, and the child core's wrapper cells. In this mode, child cores need be in EXTEST mode, as their wrapper output cells are required to apply test stimuli to the parent core, while their wrapper input cells are required to capture test responses from the parent core. As a result, test data has to been scanned through both the parent core and child cores. Hence, the available TAM wires have to be distributed between both the parent core as well as child core TAM architecture.

2. **Child** INTEST **mode** ($INTEST_C$): In this mode, child core internal testing is done; all the child cores are in INTEST mode. The parent core's wrapper elements can be in any mode of operation since the TAM inputs will be able to transport data to the child core's terminals regardless of the mode of operation of the parent core itself. Thus, in this mode, all the TAM wires can be utilized by the child cores for their INTEST testing.

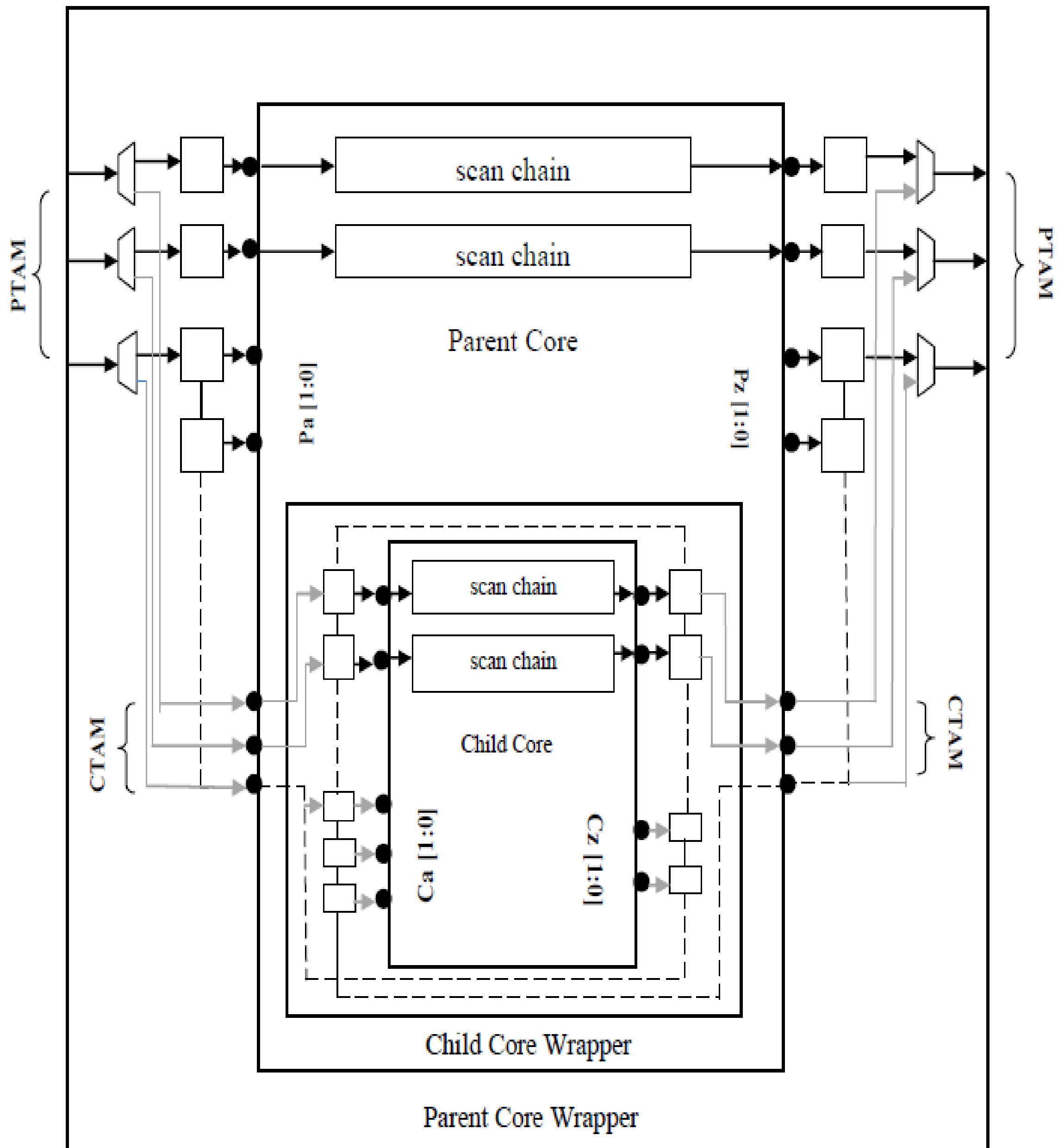


Figure 4.1 Modified Wrapper Architecture configuration for parent core in $INTEST_P$ mode [36]

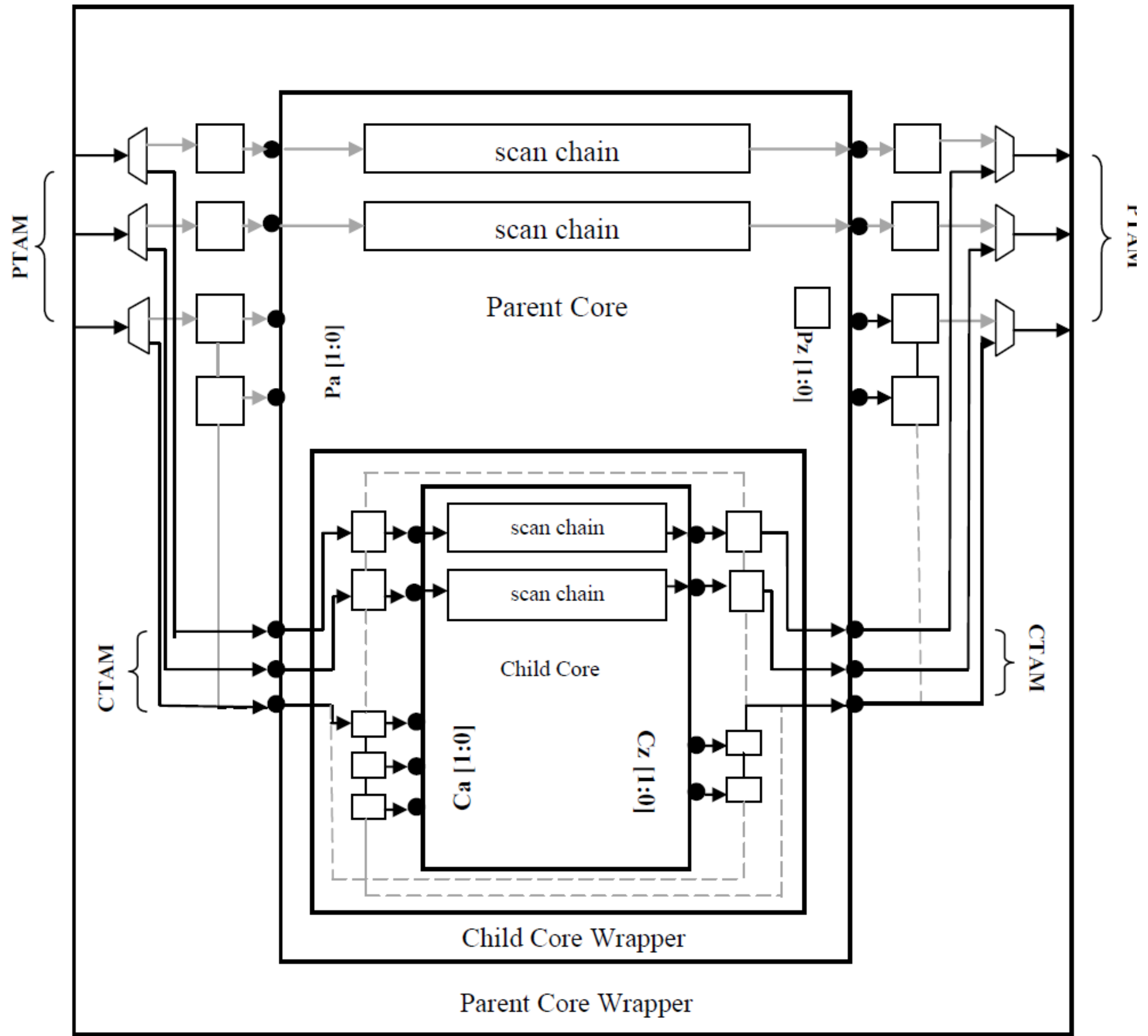


Figure 4.2 Modified Wrapper Architecture configuration for parent core in INTEST$_C$ mode [36]

4.3.1 PROBLEM FORMULATION

Problem 1 (wrapper design in INTEST$_P$ mode) [36]**:** Given a set WI = { WI_1, WI_2, . . . , WI_x } of *wrapper input cells, each wrapper input cell* has a length l (WI_i) = 1. Given a set S = S_1, S_2, ... , S_y of parent core internal scan chains, scan chain S_i has a length l (S_i). Given a set WO = { WO_1, WO_2, ... , WO_z } of wrapper output cells, each wrapper cell has a length l (WO_i) = 1. Given a set S_C = { SC_1, SC_2, ... , $SC_{C,V}$ } of CTAM scan chains, each scan chain has a length l ($S_{C,i}$). Furthermore, given a set of identical w TAM wires, we define for any $X \in WI \cup S \cup WO \cup S_C$, l (X) = $\sum_{x \in X} l(x)$. A TAM partition is a partition P = { P_1, P_2, ... , P_w } of $WI \cup S \cup WO \cup S_C$ into w disjoint sets, one for each TAM wire. We define input set $IN_i = P \setminus WO$. Likewise, we define output set OUT_i =

$P \setminus WI$. The scan-in length for TAM partition P is defined by $si(P) = \max_{1 \geq i \geq w} l(IN_i)$. The scan-out length for TAM partition P is defined $so(P) = \max_{1 \geq i \geq w} l(OUT_i)$.

Objective: To find an optimal TAM partition P^* such that the overall test length of the core is minimized, i.e. P^* satisfies max $(si(P^*)) \leq$ max $(si(P), so(P))$ for all partitions P of $WI \cup S \cup WO \cup S_C$.

Problem 2 (wrapper design in INTEST$_C$ mode) [36]**:** Given a set of CTAM chains M and child cores C. For each child core $c \in C$, the number of test patterns p_C, total scan length $sl_{c,k}$, scan-in time $si_{c,k}$ and scan-out time $so_{c,k}$ on k chain ($k \in M$) are given. Furthermore, we are given a number w that represents the maximum number of parent-core level TAM wires available for testing.

Objective: Determine a wrapper design for parent core such that overall test length (in clock cycles) required to test all child cores is minimized and the number of TAM wires used for child core testing does not exceed w.

> **Case 1:** $w \geq |M|$: If the number of available TAM wires w exceeds the number of CTAM chains in the child core TAM architecture, then every CTAM chain can be connected to a separate TAM wire at the parent level.
>
> **Case 2**: $w < |M|$: If the number of available TAM wire w is less than the number of CTAM chains $|M|$, then the available TAM wires have to be distributed among the CTAM chains, such that the overall test length of the child core is minimized. Two or more TAM chains that share same TAM wire can be daisy-chained to form TAM chain.

4.3.2 ALGORITHMS

The Problem 1 and Problem 2 has been efficiently solved by heuristic algorithm developed by incorporating a new algorithm to meet the objectives mentioned above and modifying the algorithms mentioned in the section 4.1.2 and 4.2.2. The pseudo code for the algorithms is as follows:

---

design_CTAM_architecture (C, W)

Let ***C*** be the megacore *i* and ***CC*** be the embedded cores

Let ***W*** be the total TAM width of megacore

Let ***CTERM*** be the number of CTAM terminals

Let ***C_ASSIGN [ ][ ]*** be used to determine assignment of child core to CTAM chains

Let *T* be the total testing time of megacore *i*

Determine the number of CTAM terminals

either by non-interactive design transfer model (decided by core vendor during Test Architecture design of child cores)

or by interactive design transfer model (maximum allowable TAM width is allocated).

Select the suitable number of CTAM terminals

Design CTAM architecture

Design wrapper for child cores and partition the available TAM width

Assign the child cores to TAMs and determine the test schedule such that total testing time of child cores is minimized

Obtain the best TAM widths for TAM partition

call ***create_CTAM_chains (w_best, C, CTERM, b, C_ASSIGN [ ] [ ])***

determine the total testing time of embedded child cores

Design parent wrapper

for i = 2 to CTERM/2 ***do***

design_parent_wrapper (W, C, CTAM, CTERM)

od ;

Determine the testing time of parent core

return *T* ;

---

create_CTAM_chains (w_best, C, CTERM, b, C_ASSIGN [ ] [ ])

Let ***w_best*** be the maximum CTAM width for embedded cores

Let ***C*** be the megacore *i*

Let ***b*** be number of partitions of CTAMs

Let ***CTAM*** be the CTAM scan chains

Let ***C_ASSIGN [ ] [ ]*** determines the assignment of embedded cores to TAM partitions

for i = 1 to number of embedded cores ***do***

count = count + 1;

od ;

for i = 1 to ***w_best do***

$\boldsymbol{CTAM_i}$ **=** parent core wrapper input cells + child core scan chains + parent core wrapper output cells

od ;

return CTAM scan chains so created ;

4.4 PROPOSED TEST ARCHITECTURE FOR HIERARCHICAL CORES BASED ON PARETO-OPTIMAL POINTS

In this thesis, a new test architecture is proposed for interactive design transfer flow model using modified wrapper cell for hierarchical core. Here the design of the CTAM architecture or design of embedded cores test architecture is done on the basis of pareto-optimal test bus width to each embedded core. Using the wrapper cells proposed in [36] & [38] parallel testing of parent core and its child core becomes possible. So, optimal testing time can be achieved using such wrapper cell design. Moreover, it takes comparatively less area than the wrapper cell design proposed in [36]-[38]. Comparative analysis of the testing time in chapter 5 depicts increase in testing time by 0.85% to 26.5% w.r.t. method in section 4.3. However, there will be considerable saving in area costs i.e. reduction in number of TAM wires and reduced number of wrapper cells required in CTAM architecture design as proposed in [38].

4.4.1 PARETO-OPTIMAL POINTS

Consider the core 6 of p93791, ITC'02 benchmark SOC, which is the largest core. It has 417 functional inputs, 324 functional outputs, 72 bidirectional I/Os, and 46 internal scan chains of lengths: 7 scan chains x 500 bits, 9 scan chains x 521 bits, 30 scan chains x 520 bits respectively. The *design_wrapper* algorithm in section 4,1,2 was used to create wrapper configurations for core 6, for values of TAM width w between 1 to 64 bits. Since the functional inputs outnumber functional outputs, max { s_i, s_o } = s_i. The value of s_i obtained for each value of w is illustrated in Table A of Appendix A. From the Figure A in Appendix A, it is observed that as w increases, s_i decreases in a series of distinct steps. This behaviour cause as w increases, the core internal scan chains are redistributed among a larger number of wrapper scan-in chains, thus s_i decreases only when the increase in w is sufficient to remove an internal scan chain from the longest wrapper scan chain. For example, when the internal scan chains in core 6 are distributed among 24 wrapper scan-in chains, s_i = 1041 bits long. The value s_i remains at 1041 until w reaches 39, when s_i drops to 1021. Hence, for $24 \leq w \leq 38$, only 24 wrapper scan-in chains need to be designed.

4.4.2 ALGORTHMS

The Problem 1 and Problem mentioned in section 4.3.1 of this thesis work has been tackled by modifying the interactive design transfer flow model based on modified test wrapper design using Integer Linear Programming formulation. The CTAM scan chain formation for CTAM architecture design is now been design on basis of pareto-optimal point of each embedded core for maximum allowable TAM width provided to each embedded core of meagacore. The result is considerable reduction in TAM width required to design wrapper and test architecture for embedded cores. In algorithm *pareto_optimal_point,* we initialize the width provided to embedded core to the pareto-optimal TAM width that will provide the closet testing time within *p%* from T_i (W_{max}). The pseudo code for the algorithms is as follows:

---

design_CTAM_architecture (C, W)

Let ***C*** be the megacore *i* and ***CC*** be the embedded cores

Let ***W*** be the total TAM width of megacore

Let ***CTERM*** be the number of CTAM terminals

Let ***C_ASSIGN [ ][ ]*** be used to determine assignment of child core to CTAM chains

Let *T* be the total testing time of megacore *i*

Determine the number of CTAM terminals

interactive design transfer model (pareto-optimal TAM width which is within *p%* of maximum allowable TAM width to megacore *i*).

Select the suitable number of CTAM terminals

Design CTAM architecture

Design wrapper for child cores and partition the available TAM width

Assign the child cores to TAMs and determine the test schedule such that total testing time of child cores is minimized

Obtain the best TAM widths for TAM partition

call ***create_CTAM_chains (w_best, C, CTERM, b, C_ASSIGN [ ] [ ])***

determine the total testing time of embedded child cores

Design parent wrapper

for i = 2 to CTERM/2 ***do***

design_parent_wrapper (W, C, CTAM, CTERM)

od ;

Determine the testing time of parent core

return *T* ;

---

pareto_optimal_point (W, C, p)

Let ***C*** be the megacore *i*

Let ***W*** be maximum allowable TAM width of megacore

for each core $i \in C$

calculate testing time for 1-bit TAM

calculate testing time for w_{max}-bit TAM

calculate $T_{ip} = T_{w,max} + \frac{p}{100} \times (T_i (1) - T_i (w_{max}))$;

calculate highest pareto-optimal width $\boldsymbol{width_H}$;

od ;

return $width_H$;

CHAPTER 5 – RESULTS

Several experiments have been conducted in order to evaluate the methods presented in chapter 4 of this thesis. The goal with which these experiments have been performed show the importance of wrapper/TAM design optimization and test schedule in minimizing testing time taking into consideration the constraints mentioned in chapter 1 of this thesis. In this chapter the results of several experiments conducted on ITC'02 benchmark circuits are presented. To be able to run the experiments, all the methods presented in chapter 4 have been implemented in C++. For the method in section 4.1 experimental results of all the SOCs for ITC'02 SOC benchmarks is presented, because this method considered all the SOC to have flat hierarchy. For the methods of section 4.2 to 4.3 and our proposed method in section 4.4 of this thesis, experimental results for only p22810, p34392, p93791 and a586710 from ITC'02 SOC benchmarks have been presented. These SOCs are appropriate for these experiments because they are hierarchical, containing multiple level of embedded cores. Fixed-width test access architecture i.e. hybrid TESTRAIL as TAM has been used as Test Architecture for all the techniques presented in this thesis work

5.1 RESULTS OF WRAPPER DESIGN/TAM OPTIMIZATION AND TEST SCHEDULING USING INTEGER LINEAR PROGRAMMING

In this section, experimental results on wrapper/TAM co-optimization and test schedule using Integer Linear Programming method for all SOCs of ITC'02 SOC benchmarks has been presented. Table 5.1 shows lower bound values for the set ITC'02 SOC benchmarks. Figure 5.1 (a) d695 (b) d281 (c) p22810 (d) p34392 (e) p97391 (f) a586710 illustrates decrease in testing time (in clock cycles) with increase in TAM width for two flat and four hierarchical SOCs respectively.

ITC’02 SOC BENCHMARKS												
TAM Width (W)	u226	d695	d281	h953	g1023	f2126	q12710	p22810	p34392	p93791	t512505	a586710
16	6319927	11966	67527	30171	11693	91909	782579	143047	328324	400035	1411423	68014038
24	5753714	6322	36243	20267	7622	59029	504116	85713	215212	214569	940826	37226574
32	5602428	4066	24644	15425	5801	36365	382575	65070	156013	131674	707167	26012024
40	5458647	3154	19158	12586	4922	28850	308800	56202	136711	100078	566929	20790508
48	5406955	2756	16249	10646	4427	23947	254365	49394	121838	74495	472922	17869363
56	5370043	2482	14397	9294	4089	20145	219737	45039	114688	65770	406354	16096429
64	5342360	2052	13147	8256	3858	18295	193595	41339	107352	60146	356870	14921281

Table 5.1: Lower Bounds on Testing Time for Twelve ITC’02 Benchmark SOCs

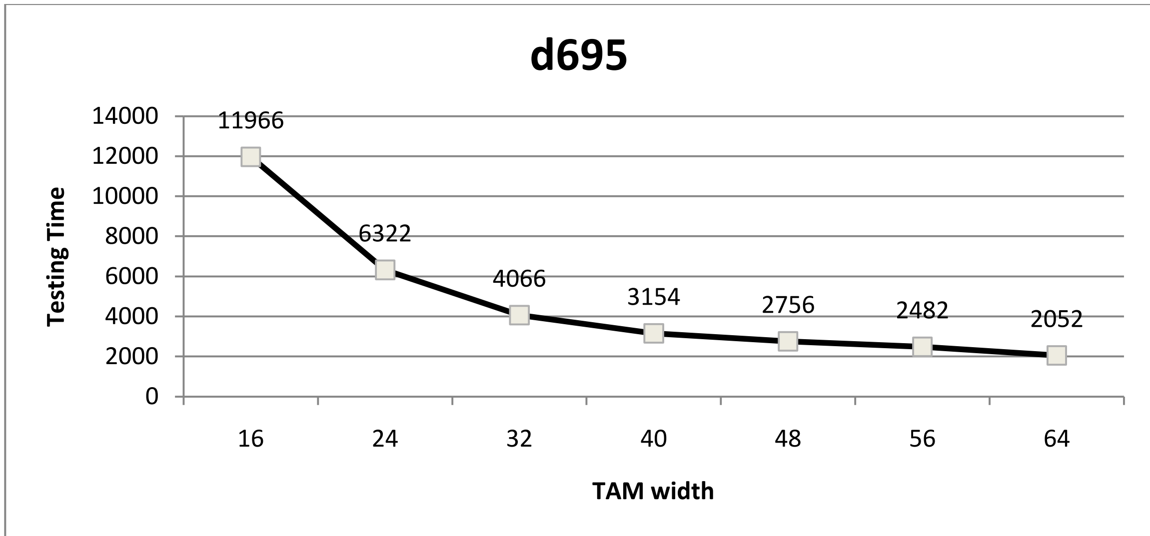


Figure 5.1 (a)

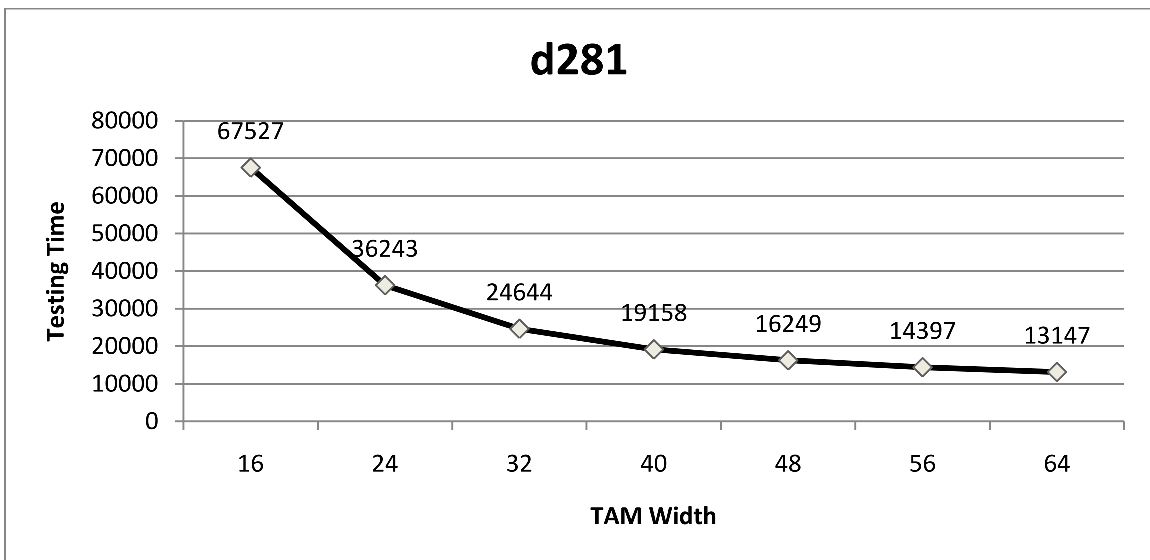


Figure 5.1 (b)

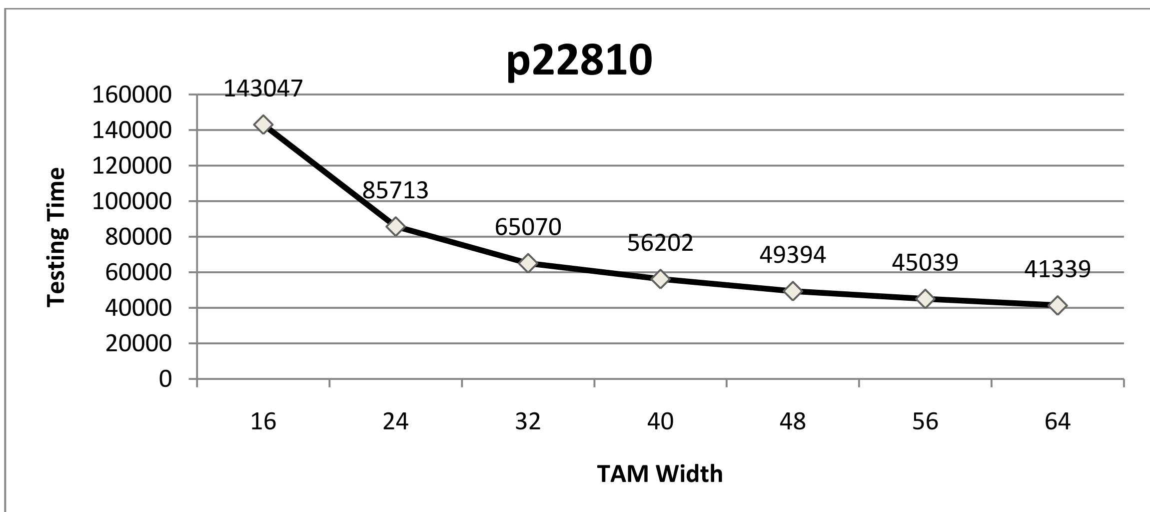


Figure 5.1 (c)

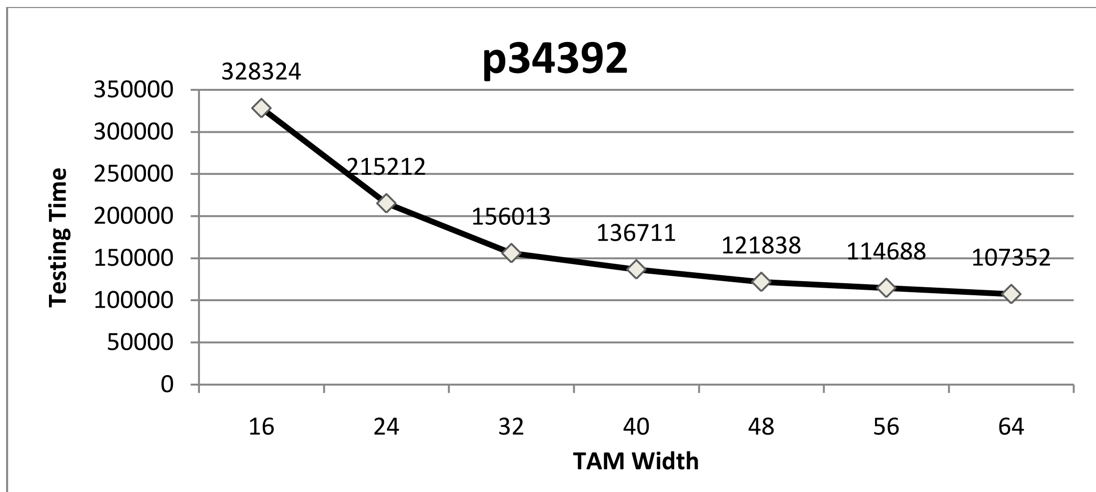


Figure 5.1 (d)

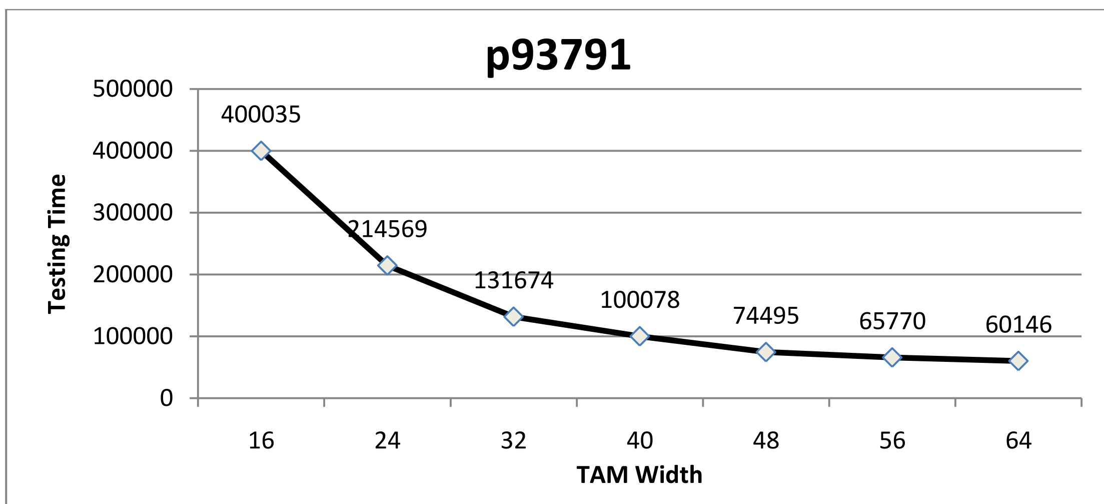


Figure 5.1 (e)

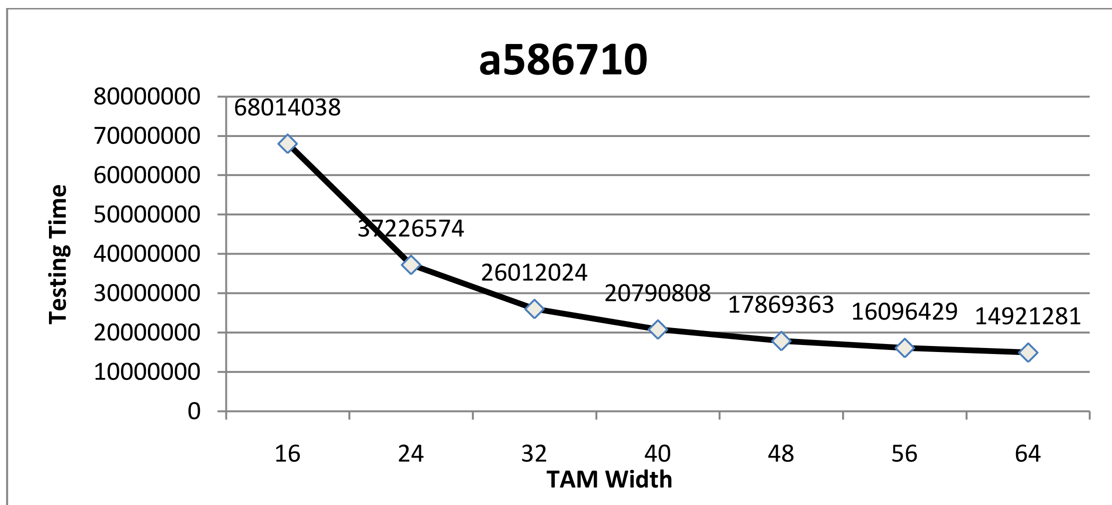


Figure 5.1 (f)

Figure 5.1 (a) d695 (b) d281 (c) p22810 (d) p34392 (e) p97391 (f) a586710 illustrates decrease in testing time (in clock cycles) with increase in TAM width for two flat and four hierarchical SOCs respectively

5.2 RESULTS OF DESIGN AND OPTIMIZATION OF MULTI-LEVEL TAM ARCHITECTURES FOR HIERARCHICAL SOCs

In this section, results of two case studies performed for the method [31] and [37] is presented. Experimental results for four SOCs: p22810, p34392, p93791and a586710 from the ITC'02 SOC benchmarks is presented.

5.2.1 NON-INTERACTIVE DESIGN TRANSFER MODEL

TAM optimization is performed using algorithm in section 4.2.1.2 of thesis. In table 5.2.1 the testing times (in clock cycles) of TAM optimization method is compared with those of the corresponding "flat" methods in section 5.1. The testing times for the "flat" and "hierarchical" method are denoted by T_{flat} and $T_{\text{non-int,hier}}$, respectively. The percentage change in testing time ΔT using hierarchical TAM optimization method is calculated as:

$$\Delta T\,(\%) = \frac{T_{\text{flat}} - T_{\text{non-int,hier}}}{T_{\text{flat}}} \text{ x } 100$$

Table 5.2.2 presents the results of (a) p22810 (b) p34392 (c) p93791 and (d) a586710 ITC'02 SOC benchmarks respectively. TAM width supplied to each megacore of the i.e. pre-specified by core vendor before system level TAM design has been specified in column 2 of each table along with $T_{\text{non-int,hier}}$.

TAM Width	T_{flat}	$T_{\text{non-int,hier}}$ (W=8)	$\Delta T\,(\%)$
16	143047	190553	33.21
24	85713	156279	82.32
32	65070	141895	118.06
40	56202	136239	142.40
48	49394	131826	166.88
56	45039	128758	185.88
64	41339	125928	204.62

Table 5.2.1 (a) Results for non-interactive design transfer model for p22810

TAM Width	T_{flat}	$T_{\text{non-int,hier}}$ (W=16)	ΔT (%)
16	328324	476133	45.01
24	215212	469065	117.95
32	156013	192360	23.29
40	136711	190241	39.15
48	121838	188829	54.98
56	114688	187819	63.76
64	107352	187061	74.25

Table 5.2.1 (b) Results for non-interactive design transfer model for p34392

TAM Width	T_{flat}	$T_{\text{non-int,hier}}$ (W=16)	ΔT (%)
16	400035	505566	26.38
24	214569	455279	112.18
32	131674	147356	11.90
40	100078	141852	41.74
48	74495	135241	81.54
56	65770	132558	101.54
64	60146	131203	118.14

Table 5.2.1 (c) Results for non-interactive design transfer model for p93791

TAM Width	T_{flat}	$T_{\text{non-int,hier}}$ (W=8)	ΔT (%)
16	68014038	207175437	20.46
24	37226574	202928970	44.51
32	26012024	201528861	67.47
40	20790508	200782495	86.57
48	17869363	200366371	102.21
56	16096429	200137512	114.47
64	14921281	199965867	124.01

Table 5.2.1 (d) Results for non-interactive design transfer model for a586710

5.2.2 INTERACTIVE DESIGN TRANSFER MODEL

TAM optimization is performed using algorithm in section 4.2.1.2 of thesis. In table 5.2.2 the testing times (in clock cycles) of TAM optimization method is compared with those of the corresponding “flat” methods in section 5.1. The testing times for the “flat” and “hierarchical” method are denoted by T_{flat} and $T_{int,hier}$, respectively. The percentage change in testing time ΔT using hierarchical TAM optimization method is calculated as:

$$\Delta T\,(\%) = \frac{T_{flat} - T_{int,hier}}{T_{flat}} \times 100$$

Table 5.2.2 presents the results of (a) p22810 (b) p34392 (c) p93791 and (d) a586710 ITC’02 SOC benchmarks respectively.

TAM Width	T_{flat}	$T_{int,hier}$	$\Delta T\,(\%)$
16	143047	190553	33.21
24	85713	106108	23.79
32	65070	76168	17.05
40	56202	62792	11.72
48	49394	54279	9.88
56	45039	48726	8.18
64	41339	44214	6.95

Table 5.2.2 (a) Results for interactive design transfer model for p22810

TAM Width	T_{flat}	$T_{int,hier}$	$\Delta T\,(\%)$
16	328324	476133	42.27
24	215212	278698	29.49
32	156013	192360	23.29
40	136711	168169	23.01
48	121838	145079	19.07
56	114688	135865	18.46
64	107352	128281	19.49

Table 5.2.2 (b) Results for interactive design transfer model for p34392

TAM Width	T_{flat}	$T_{int,hier}$	$\Delta T\,(\%)$
16	400035	505566	26.38
24	214569	238717	11.25
32	131674	147356	10.90
40	100078	114210	14.12
48	74495	78801	5.78
56	65770	69506	5.68
64	60146	62477	3.87

Table 5.2.2 (c) Results for interactive design transfer model for p93791

TAM Width	T_{flat}	$T_{int,hier}$	$\Delta T\,(\%)$
16	68014038	207175437	204.46
24	37226574	97715928	162.48
32	26012024	60428372	132.30
40	20790508	42442941	104.14
48	17869363	33259713	86.12
56	16096429	27068404	68.16
64	14921281	23614994	58.26

Table 5.2.2 (d) Results for interactive design transfer model for a586710

5.3 RESULTS OF IEEE P1500 MODIFIED TEST WRAPPER DESIGN FOR HIERARCHICAL SOCs

In this section, the experimental results are obtained by modifying the algorithms of non-interactive and interactive design flow using Integer Linear Programming method. The test architecture is based on modified wrapper design for simultaneous testing of parent and child cores. Experimental results for four SOCs namely p22810, p34392, p93791 and a586710 is presented. These four SOC are the only ones in benchmark set with multiple level of design hierarchy.

5.3.1 NON-INTERACTIVE DESIGN TRANSFER MODEL

In table 5.3.1 the testing times (in clock cycles) of non-interactive design transfer model using modified wrapper method is compared with those of the corresponding “flat” methods in section 5.1. The testing times for the “flat” and “modified wrapper (mwr)”

method are denoted by T_{flat} and $T_{non\text{-}int,mwr}$, respectively. The percentage change in testing time ΔT using hierarchical TAM optimization method is calculated as:

$$\Delta T_{(\%)} = \frac{T_{flat} - T_{non\text{-}int,mwr}}{T_{flat}} \times 100$$

Table 5.3.1 presents the results of (a) p22810 (b) p34392 (c) p93791 and (d) a586710 ITC'02 SOC benchmarks respectively. TAM width supplied to each megacore of the i.e. pre-specified by core vendor before system level TAM design has been specified in column 2 of each table along with $T_{non\text{-}int,mwr}$.

TAM Width	T_{flat}	$T_{non\text{-}int,mwr}$ (W=8)	$\Delta T_{(\%)}$
16	143047	203645	42.36
24	85713	169371	97.60
32	65070	154987	138.18
40	56202	149331	165.70
48	49394	144918	193.3
56	45039	141850	214.09
64	41339	139020	236.29

Table 5.3.1 (a) Results for modified wrapper design in non-interactive design transfer model for p22810

TAM Width	T_{flat}	$T_{non\text{-}int,mwr}$ (W=16)	$\Delta T_{(\%)}$
16	328324	510418	55.46
24	215212	530350	146.43
32	156013	185546	18.92
40	136711	182015	33.13
48	121838	180247	47.93
56	114688	179188	56.23
64	107352	178482	66.52

Table 5.3.1 (b) Results for modified wrapper design in non-interactive design transfer model for p34392

TAM Width	T_{flat}	$T_{non-int,mwr}$ (W=16)	ΔT (%)
16	400035	606311	51.56
24	214569	558473	160.27
32	131674	165247	25.49
40	100078	158337	58.21
48	74495	153164	105.60
56	65770	151334	130.09
64	60146	150196	149.71

Table 5.3.1 (c) Results for modified wrapper design in non-interactive design transfer model for p93791

TAM Width	T_{flat}	$T_{non-int,mwr}$ (W=8)	ΔT (%)
16	68014038	208170460	20.60
24	37226574	203923993	44.74
32	26012024	202523884	67.85
40	20790508	201777518	87.05
48	17869363	201361394	102.68
56	16096429	201132535	114.95
64	14921281	200960890	124.68

Table 5.3.1 (d) Results for modified wrapper design in non-interactive design transfer model for a586710

5.3.2 INTERACTIVE DESIGN TRANFER MODEL

In table 5.3.2 the testing times (in clock cycles) of interactive design transfer model using modified wrapper method is compared with those of the corresponding "flat" methods in section 5.1. The testing times for the "flat" and "modified wrapper (mwr)" method are denoted by T_{flat} and $T_{int,mwr}$, respectively. The percentage change in testing time ΔT using hierarchical TAM optimization method is calculated as:

$$\Delta T(\%) = \frac{T_{flat} - T_{int,mwr}}{T_{flat}} \times 100$$

able 5.3.2 presents the results of (a) p22810 (b) p34392 (c) p93791 and (d) a586710 ITC'02 SOC benchmarks respectively. According to the heuristic algorithm presented in section 4.3.2, the maximum allowable TAM width supplied to the megacore is ≤ $W/2$.

TAM Width	T_{flat}	$T_{int,mwr}$	ΔT (%)
16	143047	203645	42.36
24	85713	109202	27.40
32	65070	77712	19.42
40	56202	68789	22.39
48	49394	59166	19.78
56	45039	52031	15.52
64	41339	46324	12.05

Table 5.3.2 (a) Results for modified wrapper design in interactive design transfer model for p22810

TAM Width	T_{flat}	$T_{int,mwr}$	ΔT (%)
16	328324	510418	55.46
24	215212	269191	25.08
32	156013	185546	18.92
40	136711	156724	14.63
48	121838	137615	12.94
56	114688	125670	9.57
64	107352	115910	7.97

Table 5.3.2 (b) Results for modified wrapper design in interactive design model for p34392

TAM Width	T_{flat}	$T_{int,mwr}$	ΔT (%)
16	400035	606311	51.56
24	214569	279407	30.21
32	131674	165247	25.49
40	100078	122713	22.61
48	74495	83481	12.06
56	65770	71423	8.59
64	60146	62576	4.04

Table 5.3.2 (c) Results for modified wrapper design in interactive design model for p93791

TAM Width	T_{flat}	$T_{int,mwr}$	$\Delta T\,(\%)$
16	68014038	208170460	206.60
24	37226574	97769652	162.63
32	26012024	60458951	132.42
40	20790508	42478218	104.31
48	17869363	33283764	86.26
56	16096429	27085847	68.27
64	14921281	23628222	58.21

Table 5.2.2 (d) Results for modified wrapper design in interactive design model for a586710

5.4 RESULTS OF PROPOSED TEST ARCHITECTURE FOR HIERARCHICAL CORES BASED ON PARETO-OPTIMAL POINTS

In this section, experimental results of proposed method i.e. test architecture for hierarchical cores based on pareto-optimal points is presented. The algorithm presented in section 4.4.2 is used to obtain results. Again the four SOCs from ITC'02 benchmarks i.e. p22810, p34392, p93791 and a586710, is used for experimental work. The testing times (in clock cycles) so obtained from the proposed method when compared with testing times of "flat" and "modified wrapper". The testing times (in clock cycles) for "flat", "modified wrapper" and "pareto-optimal (po)" method is denoted by T_{flat} , $T_{int,mwr}$ and $T_{int,po}$ respectively. The percentage change in testing time $\Delta T_{flat,po}$ and $\Delta T_{mwr,po}$ using hierarchical TAM optimization method is calculated as:

$$\Delta T_{flat,po}\,(\%) = \frac{T_{flat} - T_{int,po}}{T_{flat}} \text{ x } 100$$

$$\Delta T_{mwr,po}\,(\%) = \frac{T_{int,mwr} - T_{int,po}}{T_{int,mwr}} \text{ x } 100$$

Table 5.4 presents the results of (a) p22810 (b) p34392 (c) p93791 and (d) a586710 ITC'02 SOC benchmarks respectively. Figure 5.4 (a) - (d) presents the comparative analysis of testing time (in clock cycles) versus TAM width obtained from all the methods and the proposed method for all the four SOCs from ITC'02 SOC benchmarks. According to the algorithm presented for the proposed method, the maximum allowable TAM width to the megacore range from 8 to 11 bits (based on pareto-optimal points).

TAM Width	T_{flat}	$T_{int,mwr}$	$T_{int,po}$	$\Delta T_{flat,po}$	$\Delta T_{mwr,po}$
16	143047	203645	233020	62.89	14.42
24	85713	109202	123725	44.34	13.29
32	65070	77712	81601	25.40	5.004
40	56202	68789	70039	24.62	1.81
48	49394	59166	60751	22.92	2.60
56	45039	52031	53027	17.73	1.91
64	41339	46324	46831	13.28	1.09

Table 5.4 (a): Results for proposed test architecture method for p22810

TAM Width	T_{flat}	$T_{int,mwr}$	$T_{int,po}$	$\Delta T_{flat,po}$	$\Delta T_{mwr,po}$
16	328324	510418	584229	77.94	14.46
24	215212	269191	288820	34.20	7.29
32	156013	185546	212747	36.36	14.65
40	136711	156724	160923	17.71	2.67
48	121838	137615	139552	14.53	1.40
56	114688	125670	127081	10.80	1.12
64	107352	115910	117898	9.82	1.71

Table 5.4 (b): Results for proposed test architecture method for p34392

TAM Width	T_{flat}	$T_{int,mwr}$	$T_{int,po}$	$\Delta T_{flat,po}$	$\Delta T_{mwr,po}$
16	400035	606311	741474	85.35	22.29
24	214569	279407	306396	42.79	9.65
32	131674	165247	182595	38.67	10.49
40	100078	122713	128468	28.36	4.69
48	74495	83481	102169	37.14	22.38
56	65770	71423	79846	21.40	11.79
64	60146	62576	63117	4.93	0.86

Table 5.4 (c): Results for proposed test architecture method for p93791

TAM Width	T_{flat}	$T_{int,mwr}$	$T_{int,po}$	$\Delta T_{flat,po}$	$\Delta T_{mwr,po}$
16	68014038	208170460	263345045	287.19	26.50
24	37226574	97769652	115341640	209.83	17.97
32	26012024	60458951	66585785	155.98	10.13
40	20790508	42478218	45746124	120.03	7.69
48	17869363	33283764	35151219	96.71	5.61
56	16096429	27085847	28381276	76.32	4.78
64	14921281	23628222	24317693	62.97	2.91

Table 5.4 (d): Results for proposed test architecture method for a586710

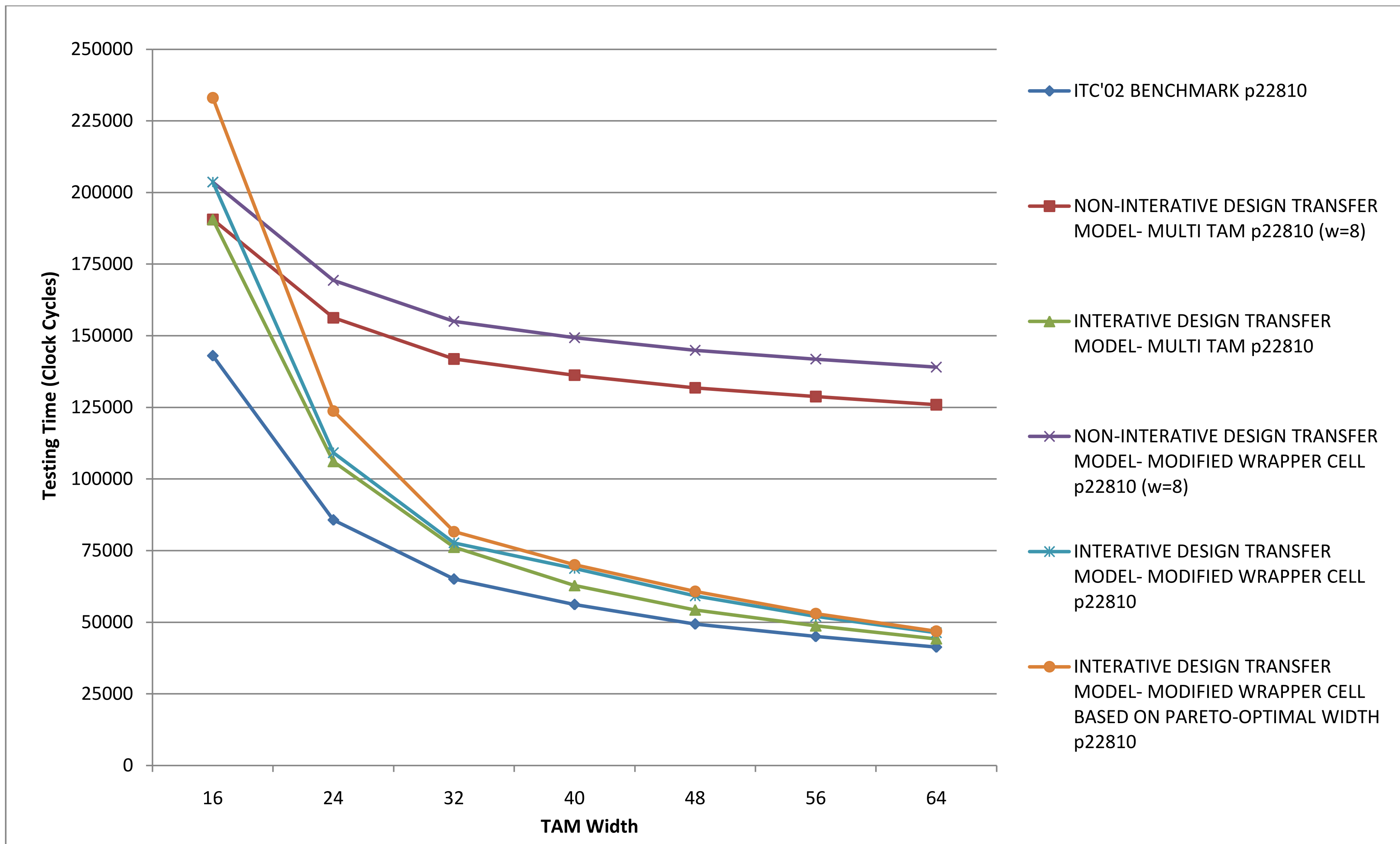


Figure 5.4 (a): Testing time versus TAM width obtained from all the methods and proposed method for p22810

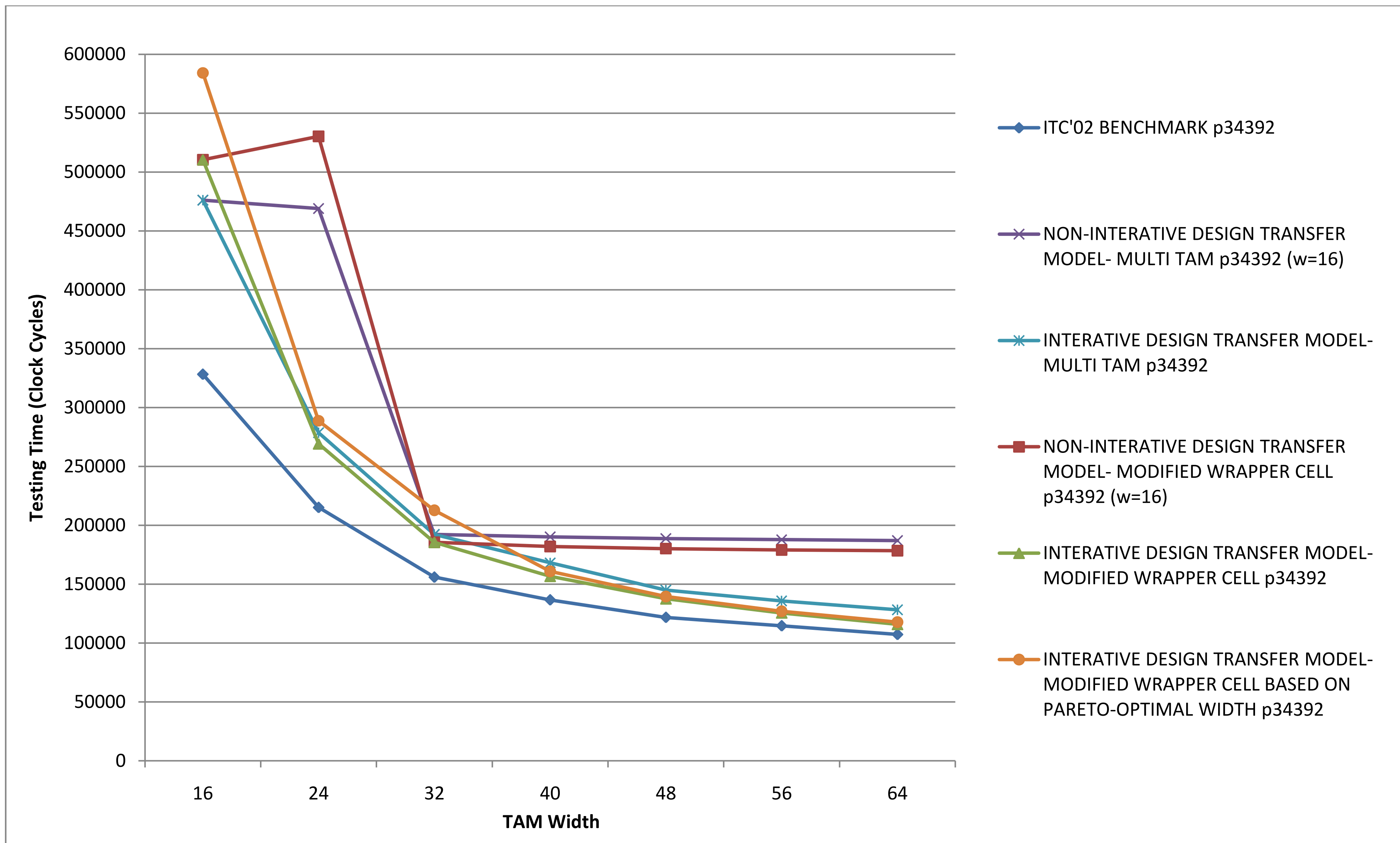


Figure 5.4 (b): Testing time versus TAM width obtained from all the methods and proposed method for p34391

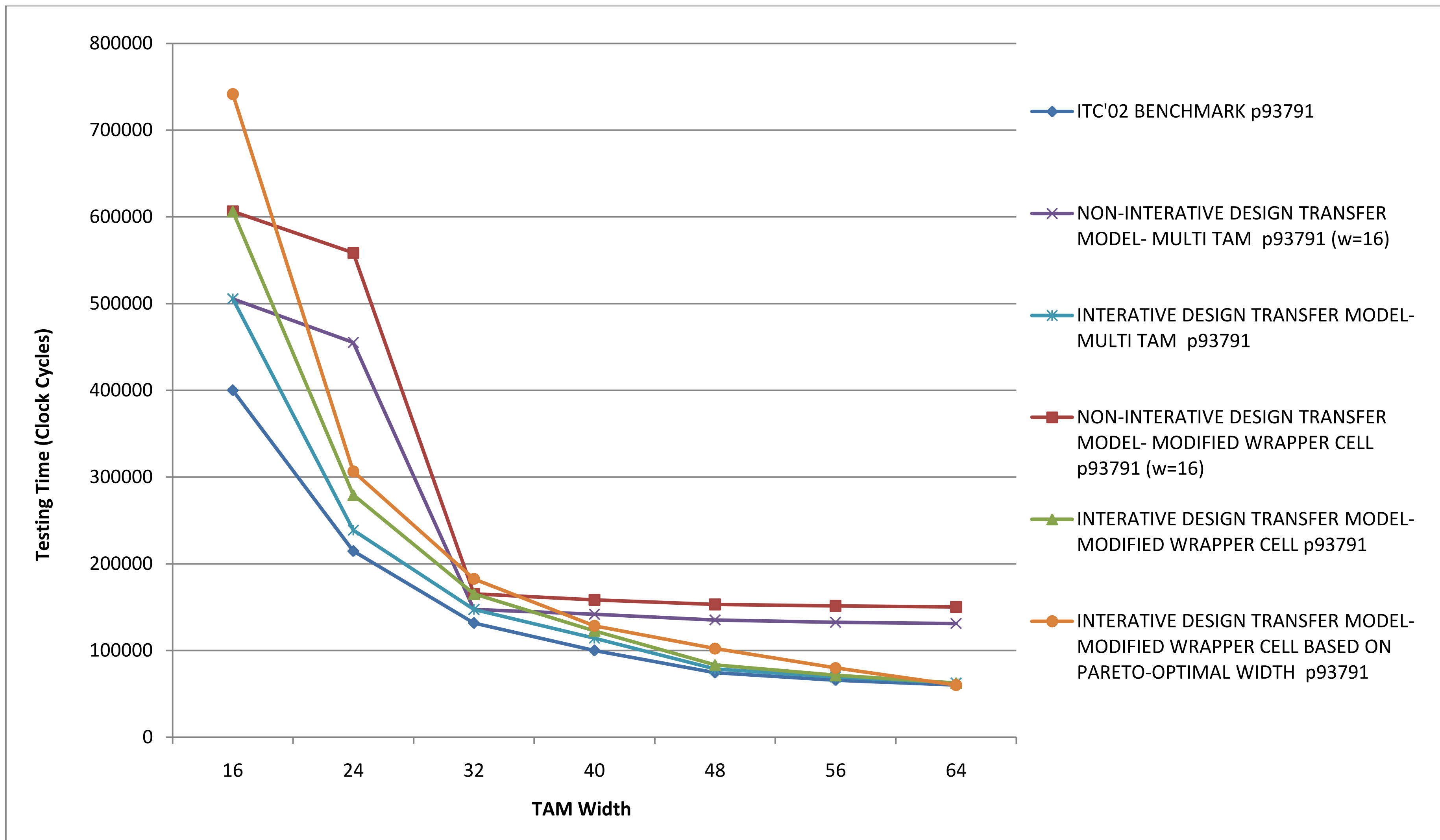


Figure 5.4 (c): Testing time versus TAM width obtained from all the methods and proposed method for p93791

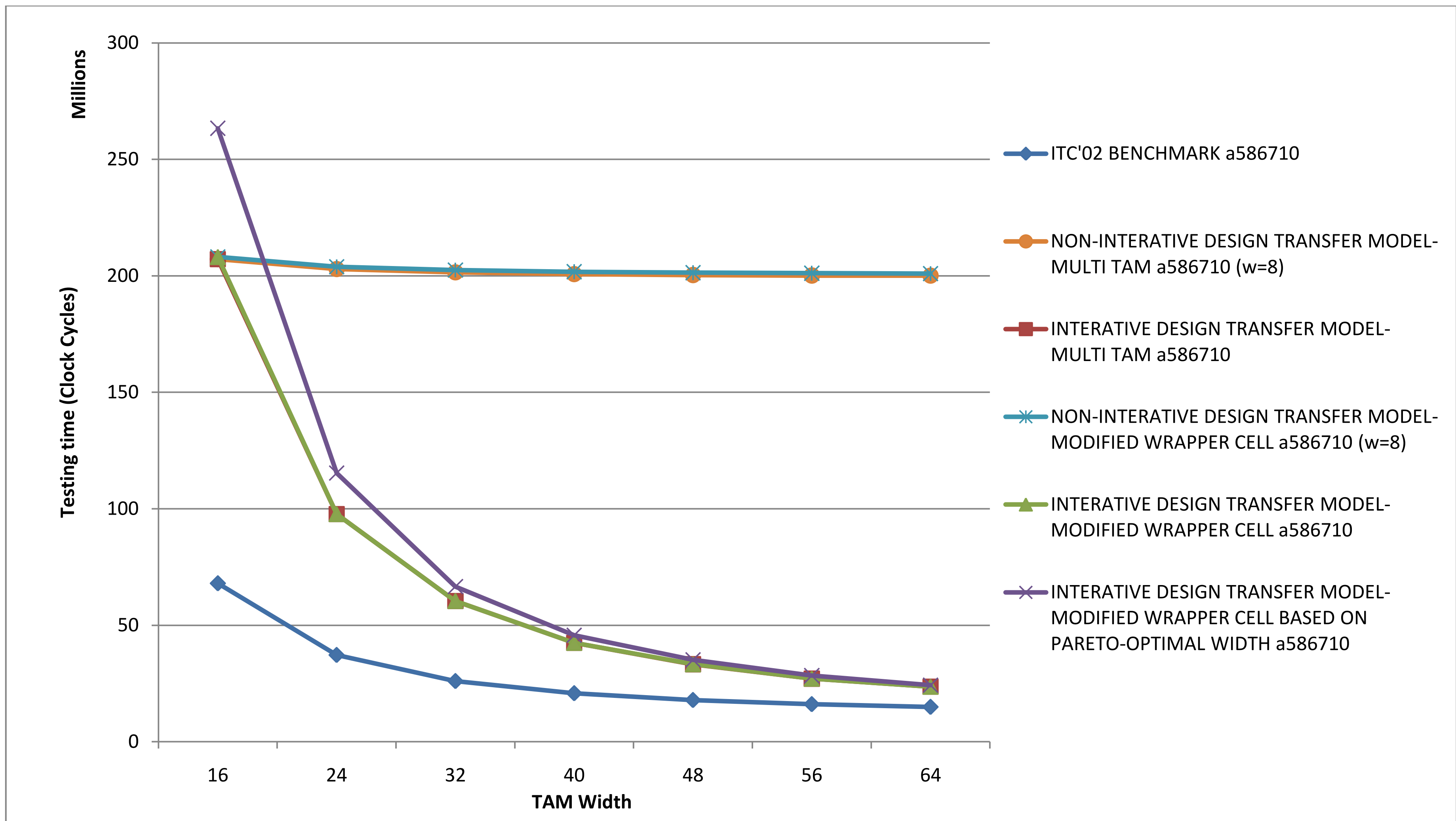


Figure 5.4 (d): Testing time versus TAM width obtained from all the methods and proposed method for a586710

CHAPTER 6- CONCLUSION & FUTURE WORK

6.1 CONCLUSION

SOC designs today are no longer limited to only on level of hierarchy. Instead, they typically consist of multiple levels of design hierarchy. Here in this thesis, the importance of test access architecture design i.e. wrapper/TAM design and optimization procedures and test scheduling to minimize the testing time of the SOCs has been discussed. A new test architecture is proposed in this thesis work which utilizes the modified wrapper design for iterative design model using Integer Linear Programming method. In this proposed method, test architecture of embedded cores in a megacore is designed by core vendor on the basis of highest pareto-optimal TAM width to each embedded core. So this method requires very little modification in architecture optimization procedure.

Experimental results for four hierarchical SOCs taken from ITC'02 SOC test benchmarks have been presented. Comparative analysis of experimental results obtained for the proposed method with "modified wrapper" method shows a minimal increase in testing time (in clock cycles) of 0.86% to 26.50%. However, this increase in testing time is mitigated by decrease in area costs i.e. less number of TAM wires used in designing CTAM architecture in megacores as well as reduced number of modified wrapper cells required, thereby considerable chip area saving. This new proposed architecture extend the existing test architectures in such a way that all the constraints imposed by hierarchy are satisfied and full flexibility is provided design algorithm for SOC with hierarchical cores.

6.2 FUTURE WORK

The method proposed can be extended from fixed-width test bus architecture to flexible width test bus architecture. Further the method can be extended to include constraints such as power dissipation during testing of SOC. Other algorithms such a Genetic Algorithm, Simulated Annealing algorithm which have been used for test resource optimization for "flat" SOC [27]-[30] can be extended with this proposed test architecture method to minimize testing time of Hierarchical core-based SOCs.

REFERENCES

[1] H. Chang, L. R. Cooke, M. Hunt, G. Martin, A. McNelly, and L.Todd, "Surviving the SOC Revolution - A Guide to Platform-based Design," *Kluwer Academic Publishers*, ISBN 0-7923-8679-5, 1999.

[2] Semiconductor Industry Association. "International Technology Roadmap for Semiconductors (ITRS)," *Web site: http://www.itrs.net/reports.html*, 1999.

[3] Y. Zorian, E. J. Marinissen, and S. Dey, "Testing Embedded-Core-Based System Chips," *IEEE Computer*, Vol. 32, No. 6, pp.52–60, 1999.

[4] Semiconductor Industry Association. "International Technology Roadmap for Semiconductors (ITRS)," *Web site: http://www.itrs.net/Links/2001ITRS/ Home.htm*, 2001.

[5] S. K. Goel, C. Kuoshu, E. J. Marinissen, T. Nguyen, and S. Oostdijk, "Test Infrastructure Design for the NexperiaTM Home Platform PNX8550 System Chip," *in Proceedings of IEEE Design, Automation and Test in Europe (DATE)*, pp. 108–113,2004.

[6] Semiconductor Industry Association. "International Technology Roadmap for Semiconductors (ITRS)," *Web site: http://www.itrs.net/Links/2007ITRS/ Home2007.htm*, 2007.

[7] J. Pouget, E. Larsson and Z. Peng, "Multiple-Constraint Driven System-on-Chip Test Time Optimization," *J. Electronic Testing: Theory and Application,* Vol. 21, pp. 599-611, 2005.

[8] S. Dutta, R. Jensen, and A. Rieckmann, "VIPER: A Multiprocessor SOC for Advanced Set-Top Box and Digital TV Systems," *IEEE Design and Test of Computers*, pp. 21-31, 2001.

[9] V. Iyengar and K. Chakrabarty, "Precedence-Based, Preemptive, and Power-Constrained Test Scheduling for System-on-a-Chip," *In Proceedings of IEEE VLSI Test Symposium (VTS)*, pp. 368–374, 2001.

[10] P. Varma and S. Bhatia, " A structured test re-use methodology for core-based system chips", *in Proc. Intl. Test Conf.,* pp. 294-302, 1998.

[11] E.J. Marinissen, S.K. Goel, and M. Lousberg, "Wrapper Design for Embedded Core Test," *Proc. IEEE Int'l Test Conf. (ITC '00)*, pp. 911-920, Oct. 2000.

[12] V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "Test Wrapper and Test Access Mechanism Co-Optimization for System-on-Chip," *Proc. IEEE Int'l Test Conf. (ITC '01)*, pp. 1023-1032, Oct. 2001.

[13] S. Koranne, "Design of Reconfigurable Core Wrappers for Embedded Core Based SOC Test," *Proc. Third Int'l Symp. Quality of Electronic Design (ISQED '02)*, Mar. 2002.

[14] K. Chakrabarty, " Test Scheduling for Core-Based Systems Using Mixed-Integer Linear Programming," *IEEE Trans. on CAD*, vol. 19, pp 1163-1174, October 2000.

[15] M. Nourani and C. Papachristou, "An ILP formulation to optimize test access mechanism in System-on-Chip testing," *in Proc. Intl. Test Conf.*, pp. 902-910, 2000.

[16] Y. Huang et al., "Resource Allocation and Test Scheduling for Concurrent Test of Core-Based SOC Design," *Proc. 10th IEEE Asian Test Symp. (ATS '01)*, pp. 265-270, Nov. 2001.

[17] S.K. Goel and E.J. Marinissen, "Effective and Efficient Test Architecture Design for SOCs," *Proc. IEEE Int'l Test Conf. (ITC '02)*, pp. 529-538, Oct. 2002.

[18] V. Iyengar, K. Chakrabarty and E. J. Marinissen, "On Using Rectangular Packing for SOC Wrapper/TAM Co-Optimization," *Proc. 20th IEEE VLSI Test Symposium*, 2002.

[19] Y. Huang et al., "Optimal Core Wrapper Width Selection and SOC Test Scheduling Based on 3-D Bin Packing Algorithm," *Proc. IEEE Int'l Test Conf. (ITC '02)*, pp. 74-82, Oct. 2002.

[20] S. Koranne and V. Iyengar, "On the Use of k-Tuples for SOC Test Schedule Representation," *Proc. IEEE Int'l Test Conf. (ITC '02)*, pp. 539-548, Oct. 2002.

[21] V. Iyengar, K. Chakrabarty, E. J. Marinissen, "Test Access Mechanism Optimization, Test Scheduling, and Tester Data Volume Reduction for System-on-Chip," *Proc. IEEE Int'l Test Conf.*, vol 12, pp. 319-324, Dec. 2003.

[22] E. J. Marinissen, R. Arendsen and G. Bos, "A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores," *in Proc. Intl. Test Conf.*, pp. 284-293, 1998.

[23] E. Larsson, "Introduction to Advanced System on Chip Test Design and Optimization," Springer, 2005.

[24] M. Sugihara, H. Date and H. Yasuura, " A novel test methodology for core-based system LSIs and is testing time minimization problem," *Proc. Int. Test Conf*, pp. 465472, 1998.

[25] K. Chakrabarty, "Testing Scheduling for Core-Based System," *Proc. Intl. Conf.* CAD, pp. 391-394, November 1999.

[26] K. Chakrabarty, "Design of System-on-Chip Test Access Architecture using Integer Linear Programming," *Proc. Intl. Conf. IEEE VLSI Test Symposium*, 2000.

[27] W. Zou, S. R. Reddy, I. Pomeranz and Y. Huang, "SOC Test Scheduling Using Simulated Annealing," VTS 2003.

[28] J. Im, S. Chun, G. Kim, J. An, S. Kang, "RAIN (RAndom INsertion) Scheduling Algorithm for SOC Test," *in Proc. of Asian Test Symp.*, 2004.

[29] C. Giri, S. Sarkar and S. Chattopadhyay, "A Genetic Algorithm Based Heuristic for Power Constrained Test Scheduling in Core-base SOCs," *Proc. IEEE Int'l Test Conf. (ITC '07)*, pp. 320-323, 2007.

[30] Jin-Ho Ahn and Sungho Kang, "SOC Test Scheduling Using ACO-Based Rectangle Packing," *ICIC,* pp. 655-660, 2006.

[31] A. Seghal, " IEEE P1500-Complaint Test Wrapper Design for Hierarchical Cores in System Chips," in *Digest of papers of ETS,* pp. 147-152, 2004.

[32] V. Iyengar, K. Chakrabarty, M. D. Krasniewski and G. N. Kumar, "Design and Optimization of Multi-Level TAM Architectures for Hierarchical S0Cs," *Proc. IEEE VLSI Test Symposium (VTS'03)*, pp. 299-304, 2003.

[33] T. Wang, C. Tsai, M. Shich, K. Lee, "Efficient Test Scheduling for Hierarchical Core Based Design," *in Proc. IEEE,* 2005.

[34] H. M. Harmanani and R. Farah, " Integrated Test Scheduling, Wrapper Design and TAM Assignment for Hierarchical SOC," *IEEE Trans. of Circuits and Systems,* pp. 1388 -1391, 2007.

[35] Xu Chuan-pei, Dai Kui, "The Optimization of Hierarchical SOC Test Architecture to Reduce Test Time," *Proc. Electronic Packaging Technology & High Density Packaging,* 2008.

[36] S. Goel, E. J. Marinissen, A. Sehgal and K. Chakrabarty, " Testing of SOCs with Hierarchical Cores: Common Fallacies, Test Access Optimization and Test Scheduling," *Proc. IEEE Tran. on computers,* vol. 58, March 2009.

[37] K. Charkrabarty, V. Iyengar and M. D. Kkrasniewski, “Test Planning for Modular Testing of Hierarchical SOCs,” *IEEE Trans. on computer aided designs of integrated circuits and systems*, vol. 24, March 2005.

[38] K. K. Saluja and K. Kim, “Low-Area Wrapper Cell Design for Hierarchical SOC Testing,” *Journal of Electronic Testing*, August 2009.

APPENDIX A

PARETO-OPTIMAL POINTS

TAM Width	Longest wrapper scan chain
1	24206
2	12103
3	8181
4	6203
5	5143
6	4142
7	3622
8	3102
9	3082
10	2582
11	2562
12	2081
13	2062
14	2061
15	2042
16-19	1561
20-21	1541
22	1522
23	1053
24-38	1041
39-42	1021
43-45	1001
46	527
47-64	522

Table A: Results of *design_wrapper* algorithm for core 6 of p93791 SOC

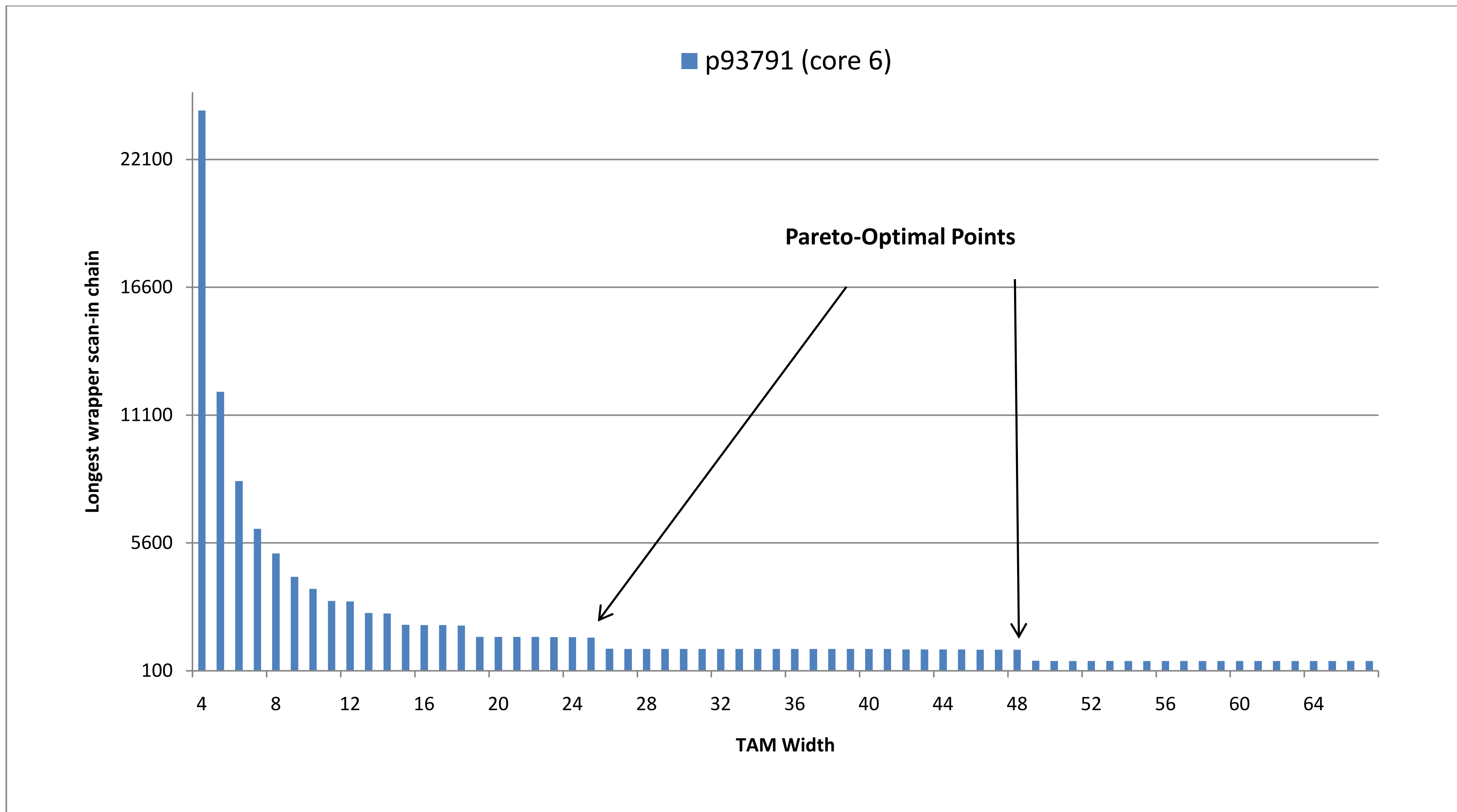


Figure A: Decrease in length of longest wrapper scan chain with increasing TAM width for core 6 of p93791 ITC'02 SOC benchmark