TEST TIME OPTIMIZATION OF CORE BASED SOCs USING HEURISTIC ALGORITHMS

Thesis report submission in the partial fulfillment of the

Requirement for the award of the degree of

MASTER OF TECHNOLOGY

IN

VLSI DESIGN

Submitted by

Ashish Singh

Roll No: 601461005

Under the Guidance of

Ms. Harpreet Vohra

Assistant Professor, ECED

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

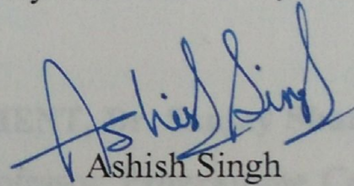
THAPAR UNIVERSITY, PATIALA (PUNJAB) – 147004

JULY-2016

CERTIFICATE

I hearby declare that the work which is being presented in the thesis entitled "**Test Time optimization of Core based SOCs using heuristic algorithms**" in partial fulfillment of the requirement for the award of degree of M.Tech. (VLSI Design) 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.

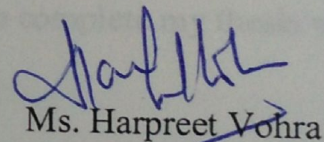


Date: 14-July-16

Ashish Singh

Roll No: 601461005

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

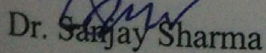


Ms. Harpreet Vohra

Assistant Professor

ECED, Thapar University

Counter signed by:

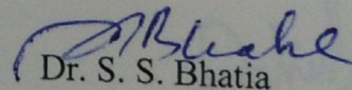


Dr. Sanjay Sharma

Professor & Head

ECED, Thapar University

Patiala-147004



Dr. S. S. Bhatia

Dean of Academic Affairs

Thapar University

Patiala-147004

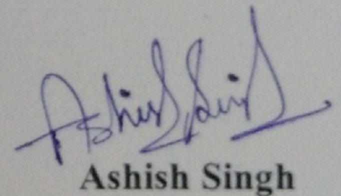
Scanned by CamScanner

ACKNOWLEDGEMENT

I take this opportunity to express my profound sense of gratitude and respect to all those who helped me through the duration of this thesis. I would have never succeeded in completing my task without the cooperation, encouragement and help provided to me by various people. Words are often too less to reveal one's deep regards. I acknowledge with gratitude and humility my indebtedness to **Ms. Harpreet Vohra, Assistant Professor**, Electronics and Communication Engineering Department, Thapar University, Patiala, under whose guidance I had the privilege to complete this thesis. I wish to express my deep gratitude towards her for providing individual guidance and support throughout the thesis work.

I convey my sincere thanks to **HEAD OF THE DEPARTMENT, Dr. Sanjay Sharma** as well as **PG Coordinator, Dr. Amit Kumar Kohli** Associate Professor, **Programme Coordinator, Dr. Anil Arora** Assistant Professor, Electronics and Communication Engineering Department, entire faculty and staff of Electronics and Communication Engineering Department for their encouragement and cooperation.

My greatest thanks to all who wished me success especially my parents and other family members and friends without whom I would not have been able to complete my thesis work.



Ashish Singh

Scanned by CamScanner

ABSTRACT

The cost of testing SOCs (System On Chip) is related to its test application time. With increase in the complexity, the number of on chip test time increases which further increases the possibility of manufacturing imperfections at deep sub micron level. This increase in the complexity increases number of fault sites which need a high test data volume for testing. This application of high test data volume leads to an increase in test application times. SOC consists of digital, analog and various mixed signal components. The urgent time to market requirement poses many challenges for the design and test engineers. Testing cost has made IC testing more difficult. ITRS semiconductor road map represents that there will be a need of hundred of processors for the future generation of SOC designs which will further increase the test cost.

Many techniques have been proposed to reduce the cost by test scheduling, reducing test data volume and optimizing test design mechanisms. In order to reduce the test time, cores needed to be tested simultaneously due to which power consumption increases. The main aim is to schedule all the given cores in the allotted TAM width in most optimum manner. Thus, various combinations have been tried to get optimum results by using the partition based technique which will be explained in detail in the subsequent sections. Test scheduling is proved to be an NP-hard problem.

In this thesis report, an algorithm is proposed for scheduling different cores so as to reduce the test time while taking power and bandwidth constraints into consideration. This problem can be reduced into a rectangle packing problem. Experimental results for ITC’02 benchmark circuits show the optimal results achieved. The proposed schemes are applied on three benchmark circuits from Duke University and Philips.

TABLE OF CONTENTS

CERTIFICATE…………………………………………………………………...i

ACKNOWLEDGEMENT…………………………………………………….....ii

ABSTRACT………………………………………………………………….....iii

LIST OF FIGURES……………………………………………………………vii

LIST OF TABLES…………………………………………………………….x

ABBREVIATIONS……………………………………………………………xii

CHAPTER 1 INTRODUCTION...……………………………………………...1

1.1 MOTIVATION…………………………………………………….....1

1.2 CORE BASED SYSTEM ON CHIP ……………………….……….3

1.3 SOC TEST PROCESS..……………………….……………………...6

1.4 SOC TEST MODEL…………………………………………………8

1.5 SOC TEST CONSTRAINTS……………………………………...14

1.6 THESIS OVERVIEW………………………………………………15

CHAPTER 2 LITERATURE REVIEW..………...…………………………...16

2.1 MINIMIZATION OF TEST TIME………………………..………..…16

2.1.1 CORE TEST TIME MINIMIZATION...……………………...17

2.1.2 CORE ACCESS TIME MINIMIZATION…………………....17

2.1.2.1 BEST FIT DECREASING ALGORITHM..……………18

2.2 TAM OPTIMIZATION…..………...……………………………………..19

2.2.1 FORK AND MERGE APPROACH..........…………………….20

2.2.2 TEST BUS APPROACH………………..…………………...20

2.3 TEST SCHEDULING…...…….………………………………………21

2.3.1 GENETIC ALGORITHM..……………………………………..22

2.3.2 ANT COLONY OPTIMIZATION……………………………..23

2.3.3 SIMULATED ANNEALING ALGORITHM…………....…....23

2.3.4 RANDOM INSERTION ALGORITHM…………………….....24

2.3.5 3-OPT ALGORITHM……………………………….……….24

2.3.6 PARTICLE SWARM OPTIMIZATION ALGORITHM………25

2.4 HIERARCHICAL CORE BASED SOC TESTING…..…………………25

CHAPTER 3 PROPOSED SCHEDULING TECHNIQUE…...…………………27

3.1 BACKGROUND AND PROBLEM FORMULATION ...……………....27

3.2 SCHEDULING ALGORITHM……………………………..……..…...28

3.2.1 EFFICIENT TEST SOLUTION TECHNIQUE (ETT) USING GREEDY ALGORITHM……………………………….……………29

CHAPTER 4 SIMULATION RESULTS...……………………………………….....38

4.1 TEST TIME RESULTS USING GREEDY ALGORITHM……...……...38

4.2 RESULTS COMPARISON..…………………………………………..46

4.2.1 COMPARISON OF GREEDY WITH J.POUGET……………46

4.2.2 COMPARISON OF GREEDY WITH E.J MARINISSEN…......48

CHAPTER 5 CONCLUSION AND FUTURE WORK…………………………..51

REFERENCES………………..….……………………………………………….52

LIST OF FIGURES

Figure No.	Title of Figure	Page No.
Figure 1.1	Moore’s law versus few Intel processors	1
Figure 1.2	Structure of Modular core design	2
Figure 1.3	System on Chip Hierarchy	4
Figure 1.4	Architecture of Core Based SOC	5
Figure 1.5	SOC Test process	7
Figure 1.6	Built in Self Test Process	7
Figure 1.7	(a) TAM Architecture Design- Test Bus Architecture (b) Multiplexing Architecture (c) Daisy Chain Architecture	8
Figure 1.8	Wrapper with Parallel TAM interface	10
Figure 1.9	(a) Unwrapped core A (b) Wrapped core A	11
Figure 1.10	Representation of Rectangular Bin Packing assessing cores	12
Figure 1.11	(a) Non partitioned (b)Preempive (c)Partioned	13
Figure 2.1	Wrapper scan chain arrangement in a core	19
Figure 2.2	Fork and merge technique	20
Figure 2.3	Test buses in double layer SOC	21
Figure 2.4	Two dimensional Rectangle bin packing representation	21

Figure 2.5	Simple genetic algorithm	22
Figure 2.6	Ant colony optimization	23
Figure 2.7	(a) hierarchical core (b) Its test hierarchy tree	26
Figure 3.1	Three Cores scheduled simultaneously initially	30
Figure 3.2	Different Cores scheduled as the testing of any core is completed	31
Figure 3.3	Flowchart representation for proposed test optimization technique with power constraints	33
Figure 3.4	Flowchart representation for proposed test optimization techique	37
Figure 4.1	Wrapper design output file of d695	38
Figure 4.2	Wrapper design output file of p22810	39
Figure 4.3	Wrapper design output file of p93791	39
Figure.4.4	Wrapper design output file for hierarchical cores of p22810	40
Figure 4.5	Wrapper design output file for hierarchical cores of p93791	40
Figure 4.6	Input data of d695 with power constraint	41
Figure 4.7	Input data of p22810 with power constraint	42
Figure 4.8	Input data of p93791 with power constraint	42
Figure 4.9	Graphical representation of Test time vs TAM width for SOC d695	44

Figure 4.10	Graphical representations of Test time vs TAM Width for SOC p222810	45
Figure 4.11	Graphical representations of Test time vs TAM Width for SOC p93791	45
Figure 4.12	Comparison of ETT with [35] graphically for SOC p22810	47
Figure 4.13	Comparison of ETT with [35] graphically for SOC p93791	47
Figure 4.14	Comparison of ETT with [8] graphically for SOC d695	49
Figure 4.15	Comparison of ETT with [8] graphically for SOC p22810	49
Figure 4.16	Comparison of ETT with [8] graphically for SOC p93791	50

LIST OF TABLES

Table No.	Title of table	Page No.
Table I	SOC D695 TEST TIME FOR VARIOUS POWER CONSTRAINT	41
Table II	SOC p22810 TEST TIME FOR VARIOUS POWER CONSTRAINT	41
Table III	SOC p93791 TEST TIME FOR VARIOUS POWER CONSTRAINT	42
Table IV	SOC p22810/p93791 TEST TIME UNDER HIERARCHICAL CONSTRAINTS	46
Table V	TEST TIME COMPARISON BETWEEN GREEDY ETT AND [35] FOR SOC p22810	46
Table VI	TEST TIME COMPARISON BETWEEN ETT AND [35] FOR SOC p93791	47
Table VII	TEST TIME COMPARISON BETWEEN ETT AND [8] FOR SOC d695	48
Table VIII	TEST TIME COMPARISON BETWEEN ETT AND [8] FOR SOC p22810	49

Table IX	TEST TIME COMPARISON BETWEEN ETT AND [8] FOR SOC p93791	49

ABBREVIATIONS

SoC	System on Chip
IP	Intellectual Property
NoC	Network on Chip
GA	Genetic Algorithm
PSO	Particle Swarm Optimization
NP	Non-Polynomial
NI	Network Interface
NN	Nearest Neighbor
MMC	Minimum Maximum Channel Load
MAC	Minimum Average Channel Load
PL	Path Load
MPSoCs	Multi-Purpose System on Chip
SA	Simulated Annealing
SMP	Symmetric Multiprocessor
ILP	Integer Linear Programming
MILP	Mixed Integer Linear Programming
ACO	Ant Colony Optimization
SA	Simulated Annealing

CHAPTER 1
INTRODUCTION

1.1 MOTIVATION

The advancement in the semiconductor technology has led to the level of integration which is being double every 1.5 years suggested by Moore's law [2]. This advancements in IC design have led to the creation of micro-electronic products with complex functionality. Improvements in fabrication technology have integrated the ICs with billions of transistors. Such single IC with all required functionality for a system is referred to as "System on Chip" or SOC. SOC can be defined to be any chip having more than 1 million gates and/or 1000+ package pins [1]. System on chip consists of many cores, memories, analog block, digital block and other functional elements which are placed on a single chip to get the best performance ratio as permitted by the latest technology [1]. Figure 1.1 shows the Moore's law versus the actual no. of transistors/chip for some Intel processors [3].

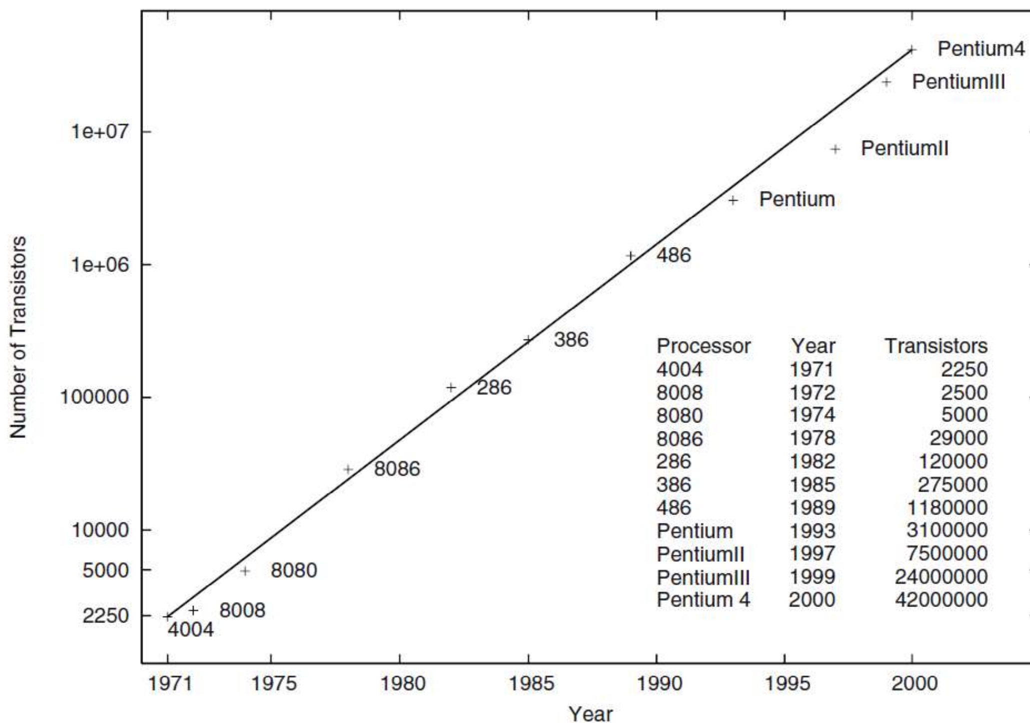


Figure 1.1 Moore's law versus few Intel processors [4].

With increase in the complexity, possibility of manufacturing imperfections at deep sub micron level increases. Due to which the number of on chip test time increases which further which become expensive due to additional circuitry and man power evolved. Also, the controllability and observabilty becomes difficult. Insertion of individual test point and BIST is not feasible as they demand structural changes. Therefore, modular approach is introduced to meet short time requirement. Pre designed and pre verified IP cores are used which is brought from within same company or from external sources. These IP cores are embedded on a single chip, which is termed as SOC. It will reduce the test time as well as the cost. IP cores can be used again and again for minimization purpose. Design for testability is also used for testing the ICs. This approach plays a vital role in reducing the test time. Each and every core in fabricated IC must be tested before releasing to the market. The best approach is to apply test vectors and compare the results at the output with the expected results.

Modular approach consists of different cores in which each and every core can be subjected for an individual test. The cores are tested with the help of test data and then are placed on the single chip. Cores are deep embedded inside SOC. There is no way to directly access the core from SOC pins. So, there is a requirement for test architecture design to handle/access the cores and test it properly. In modular testing[5] various challenges which are being faced frequently are test scheduling and test architecture design. These are major problems which need to be handled properly so that the test cost and test time can be reduced up to certain extend [13]. The structure of modular core based design is shown in Figure 1.2.

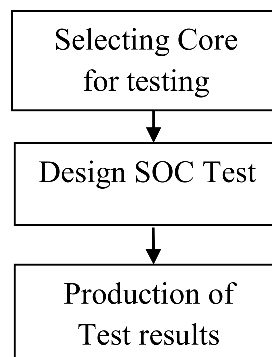


Figure 1.2 Structure of Modular core design

Due to recent advancement in the product, the circuit is becoming complex day by day. Therefore, designing a System on Chip is a major challenge. Every electronic product coming to market needs to be within the reach of customers. Therefore, the cost of the manufacturing should be kept as low as possible. In last few decades, the cost of fabrication has witnessed a sharp decrease but the value of test cost is still on the same level. So going by the facts in few upcoming years definitely the gap will shoot up. Hence it's a major issue regarding the test cost of various cores present in SOC. Also test cost plays a pivotal role in the total manufacturing cost including fabrication cost. Test time of different cores needs to be reduced by using different techniques for proper test planning. With the help of the different planning techniques, the test time of the cores would be reduced up to some extent thereby reducing the test cost hence, also reducing the manufacturing cost of the ICs.

1.2 CORE BASED SYSTEM ON CHIP

In SOC, the cores obtained from core vendors can be of different origin which can be categorized into following categories:

- Soft cores
- Firm cores
- Hard cores

Soft Cores:

Soft cores having highest flexibility are basically modules which are capable of synthesizable architecture. Their performance is highly unpredictable and they have lot of physical design issues to be faced. These cores consist of a technology independent Hardware Description Language (HDL) files, test bench and have necessary information. Soft core poses hindrances for floor planning, placement and routing.

Firm Cores:

Firm cores are ready for the purpose of routing and do not present hindrances for floor planning, routing and placements. These cores comprise of RTL code and net list which is dependent on technology. But their performance can't be predicted.

Hard Cores:

Hard cores give us the physical layout information and thereby their performance can be predicted. They are technology dependent modules and are highly optimized in terms of area. One and only set back is that hard cores can never be customized for a particular design process and hence their modification flexibility is less as compared to soft cores.

SOC comprises of a number of cores that can be possibly on the same level (no hierarchy) or at different abstraction levels (hierarchical). Cores can also be embedded inside other cores. A parent core might have cores embedded within it or it may also happen that a child core can also become a parent core with embedded child cores commonly known as hierarchical cores. Hierarchy in SOC is shown in Figure 1.3.

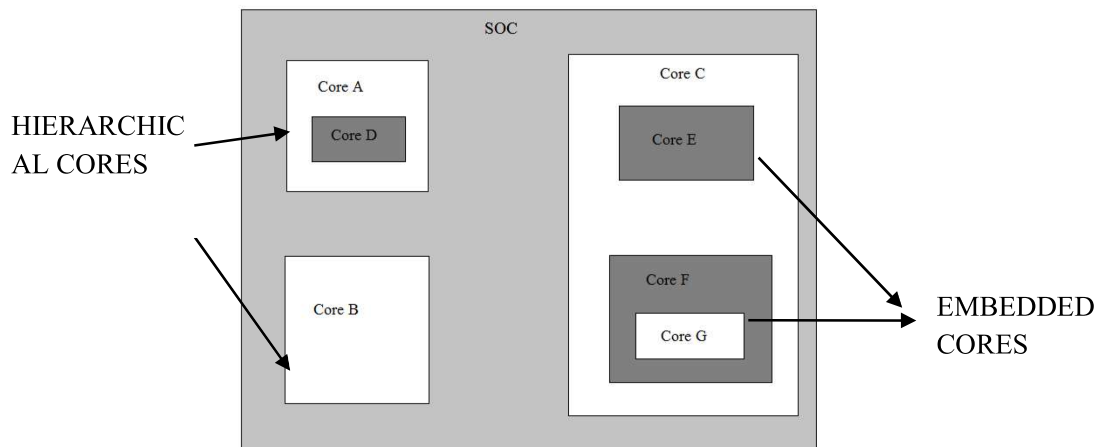


Figure 1.3 System on Chip Hierarchy

Embedded cores depict intellectual property in core based SOC and core vendors are hesitant to reveal the structural information about their cores to the users. So, users aren't able to access the core net lists. Cores are often embedded at different layers hence requiring a dedicated transport system for data. Then test stimuli are applied to embedded

core through that system from the source and we collect the test response from the sink. The core based test approach's architecture is as shown in Figure 1.4. It comprises of following elements:

- Test pattern source and sink
- Test Access Mechanism
- Core Wrapper

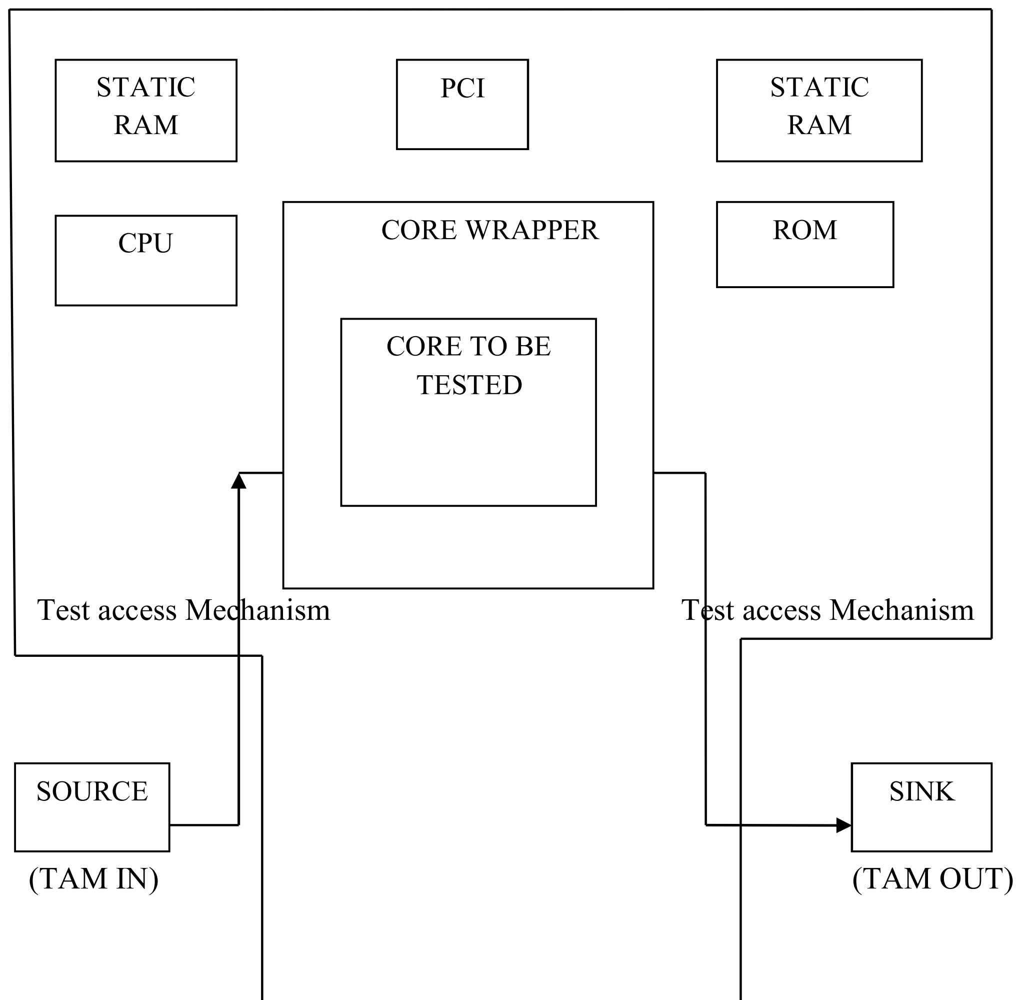


Figure 1.4 Architecture of Core Based SOC

Test pattern source and sink:

The test pattern source is the supplier of test stimuli for the cores and the test response received at the test pattern sink is compared with the expected response.

Test Access Mechanism:

Test Access Mechanism (TAM) is used as a carrier for the test patterns and test responses from test source to the circuit under test and from circuit under test to the test pattern sink. In order to increase the performance we have to optimize the TAM.

Core Wrapper:

Thin shell around the core termed as core wrapper which is acting as an interface between the core under test and its environment. Together with the TAM, core wrapper use to form a test access infrastructure for embedded reusable cores.

1.3 SOC TEST PROCESS

Each individual core in an SOC is surrounded by testing components and may imply certain modes of isolation like safe mode, low power mode and bypass mode. SOC test involves individual testing of cores and interconnect testing. SOC testing involves the test stimuli comprising of test vectors being applied to the chip and the response of the test stimuli is compared with the expected response as shown in Figure 1.3. Application of the test stimuli to the chip is done using following approaches:

- Automatic Test Equipment (ATE)
- Built In Self Test (BIST)

Automatic Test Equipment: ATE is basically inbuilt testing of the SOC without any additional hardware. ATE inherently compares the output response with the expected response obtained through the chip [4]. Any difference found between them leads to a detection of a fault. There are complex and high speeds SOC owing to present technology. So, in order to optimize ATE, we can reduce the test time using scheduling algorithms or test data volume. Automatic test equipment is shown in the figure 1.5

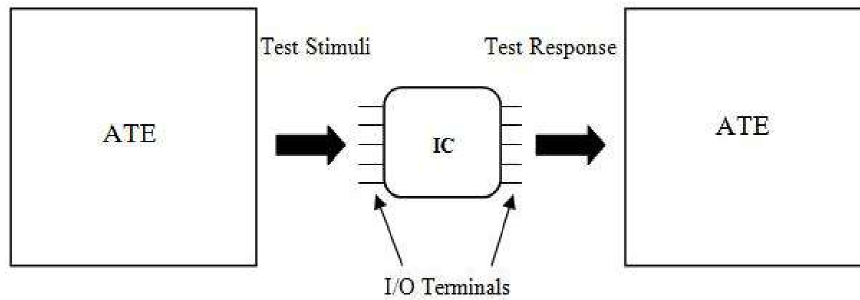


Figure 1.5 SOC Test process [4]

Built in Self Test:

In this approach, the testing of the cores can be carried out using the on-chip engine which generates the test stimuli and check responses by employing LFSR (Linear Feedback shift register) which are basically D Flip flops along with some additional testing hardware. These LFSR's can be made to work as response analyzer as well as a comparator. It simplifies the test task for the system integrator but there are some limitations. Most cores cannot be tested using BIST because they contain the logic of which only around 60% can be tested by employing randomly generated tests. Built in self test process is shown in figure 1.6.

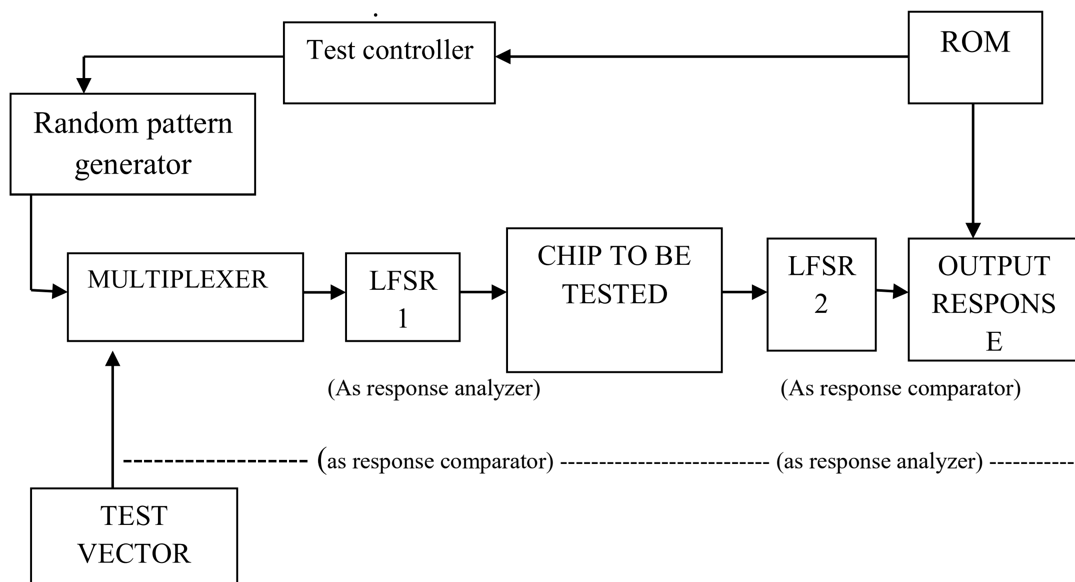


Figure 1.6 Built in Self Test Process

So, pre-computed test set is applied to the cores obtained from the core vendors. The core vendors have no information about the placement of the core on the chip. They simply assume that the terminals of the core have direct access to the test data. It is the responsibility of the system integrator to allow the test which is to be applied, i.e. by designing a suitable test access mechanism (TAM).

1.4 SOC TEST MODEL

The recent advancement in IC design technology has made it easier to create extremely complicated ICs. The test cost of the ICs is dramatically increasing and it's crucial to develop techniques that can minimize the test cost. The test cost is directly dependent on test application time whereas test time for SOCs is increasing due to rising complexity of chips. So, in order to reduce the test time there is a need of SOC test model where pre-defined logic blocks are merged with user defined logic. SOC test model comprises of the following main parts:

- Test Architecture Design.
- Wrapper Design.
- Test Scheduling.

Test Architecture Design:

Test architecture design acts as a carrier for the transportation of the test data in the SOC connecting the cores of the SOC to the test sources and test sink. It can be optimized for the purpose of testing or existing structure can be used by optimizing architecture. In recent years, various architectures have been proposed for architecture design like Multiplexing Architecture [5], Daisy Chain Architecture [5], Test Bus Architecture [6] and Flexible-Width Architecture [7].

In the Multiplexing architecture, each core is assigned to all the TAM wires at an instance and the core is tested one at a time. So, there is sequential testing of cores.

In Daisy chain architecture, a bypass structure is added in order to reduce the access path for each core. Bypass register and 2x1 multiplexer also allows full access to each and every core.

Test Bus is an amalgamation of both Multiplexing and Distribution architecture. Several test buses are created from the given TAM width. All cores are tested in an orderly manner in each test bus and concurrent testing can also be achieved if there are several test buses which are active at a point of time. Figure 1.7 shows the TAM architectures [4] proposed in few papers.

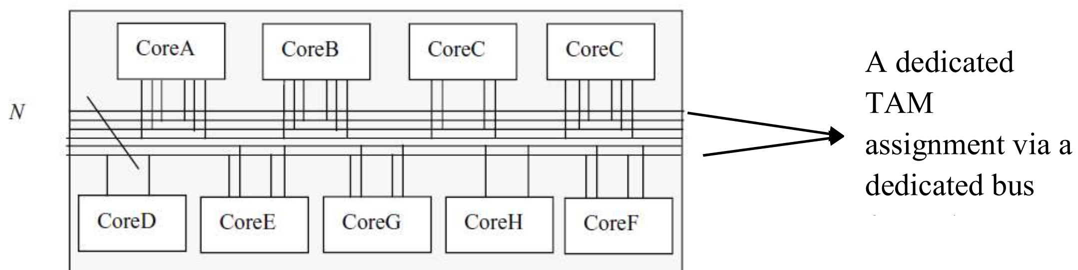


Figure 1.7(a) Test Bus Architecture

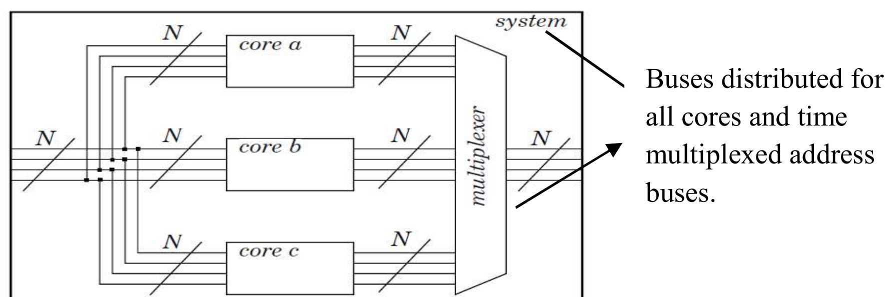


Figure 1.7(b) Multiplexing Architecture

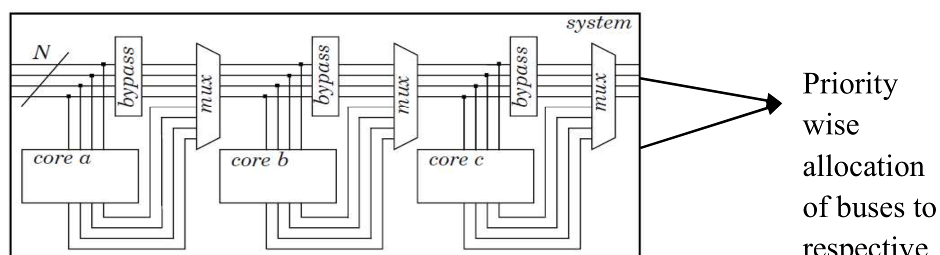


Figure 1.7(c) Daisy Chain Architecture

Wrapper Design:

Core wrapper act as an interface between the core terminals and the TAM wires which is termed as Test shell around the core. A standard IEEE P1500 Wrapper is proposed in [4] which solves the various parameters of core based testing like passing of the information about the cores to the core vendors, access the ATE or BIST engine to the core under test and optimizes with respect to the multiple conflicts. Figure 1.8 shows the IEEE P1500 Wrapper with parallel TAM interface.

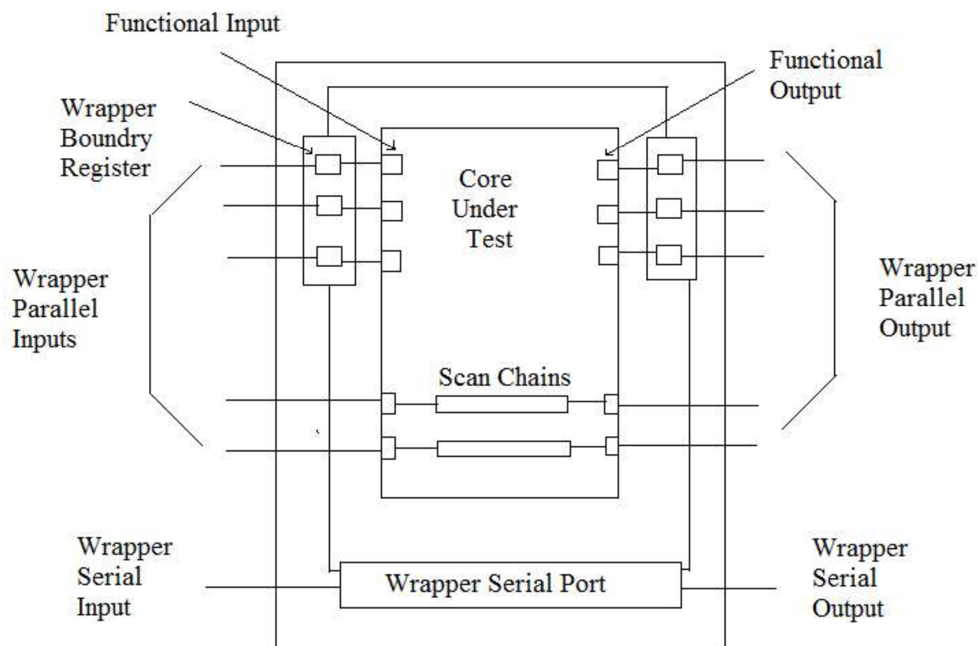


Figure 1.8 Wrapper with Parallel TAM interface

Core Wrapper is used to ensure isolation of the core from the nearby environment. In view of testing, two modes namely internal test mode (Intest) and external test mode (Extest) exist, in which a core wrapper runs. In Intest mode, the each core is tested while in Extest mode the interconnections between the two wrappers are tested. In SOC, there might be wrapped and unwrapped cores in the system.

The internal scan chains and the functional input/outputs of the core under test are merged by a wrapper. Wrapper design algorithms has been proposed such as Best Fit Decreasing [8] so as to arrange the internal elements of the core and thus calculating the

test time of wrapper core. Let us assume there is core A with 5 scan chains. The length of the scan chain is 10,8,5,3 and 2 respectively. Figure 1.9(a) shows the unwrapped core A and Figure 1.9(b) shows the wrapped core A.

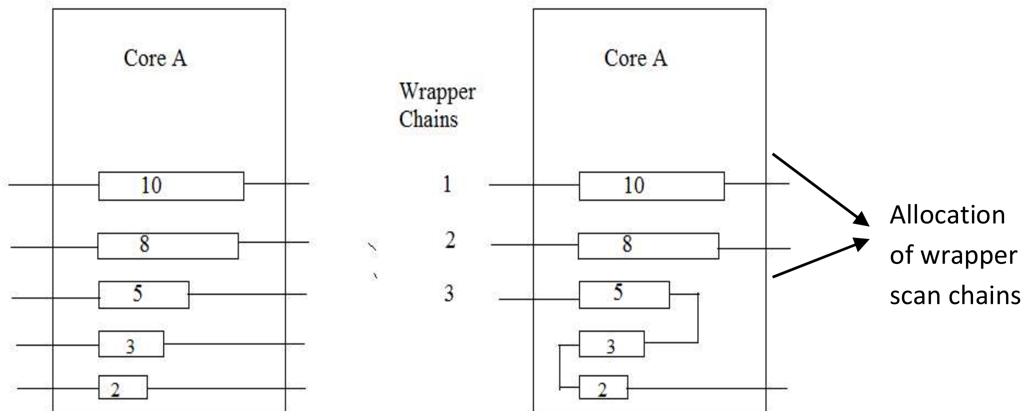


Figure 1.9(a) Figure 1.9(b)

Figure 1.9 Arrangement of scan chains in (a) Unwrapped core A (b) Wrapped core A.

Test Scheduling:

After knowing about the TAM type and wrapper design which are used, now the major challenge is test scheduling. Main operation in test scheduling is to assess the order where the data to be tested are applied so that each and every cores can be tested in lesser time as much as possible. It will definitely prove to be remarkable solution in terms of minimum test time. Test scheduling is the arrangement of the cores in different possible manners beyond the thoughts. Because of this reason it is also called NP Hard Problem. Test scheduling should be performed in such a manner that some of the constraints such as power and TAM width should not exceed its maximum value which is provided. While doing scheduling, multiple parameters are taken in to consideration. Concurrent testing of the cores concept is also taken in to consideration to get optimum test time. Rectangular bin packing [9] is showing the way of scheduling the different cores so as to get optimum value of test time. Bin packing representation is shown in Figure 1.10. There are various scheduling algorithms which are used for scheduling the cores. Some

of the recommended algorithm are genetic [11], Ant colony optimization [12], random insertion algorithm [10] etc.

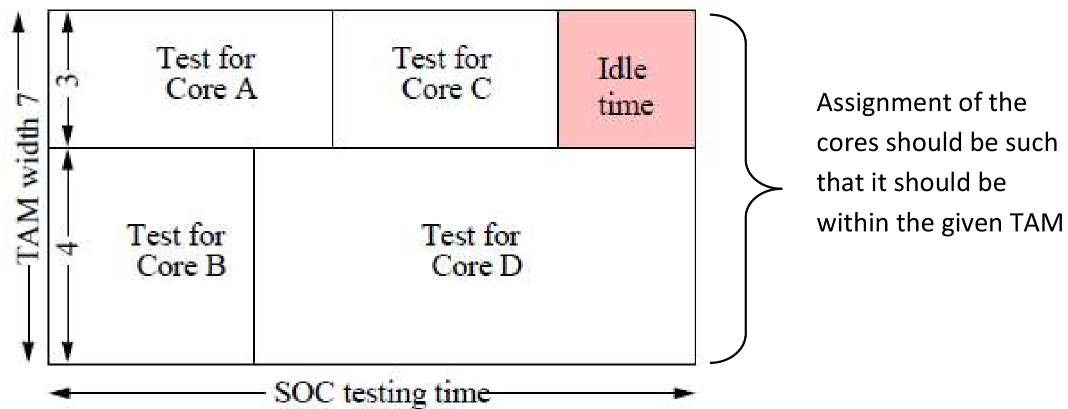


Figure 1.10 Representation of Rectangular Bin Packing assessing cores [9].

There are following scheduling techniques which are used for scheduling the cores:

- Non partitioned testing
- Pre-emptive testing
- Partitioned testing

Non partitioned testing: It is a technique in which no new core is subjected for testing unless and until all the cores which are being tested in that session have been tested successfully. This testing method is shown in figure 1.11(a).

Pre-emptive testing: It is a technique in which a core test can be haulted for some time and later get resumed. The necessary condition is that all the cores should be tested till the end. This testing method is shown in figure 1.11(b).

Partitioned testing: It is a technique in which a core is scheduled as soon as the testing over of the previous one. This technique is shown in figure 1.11(c).

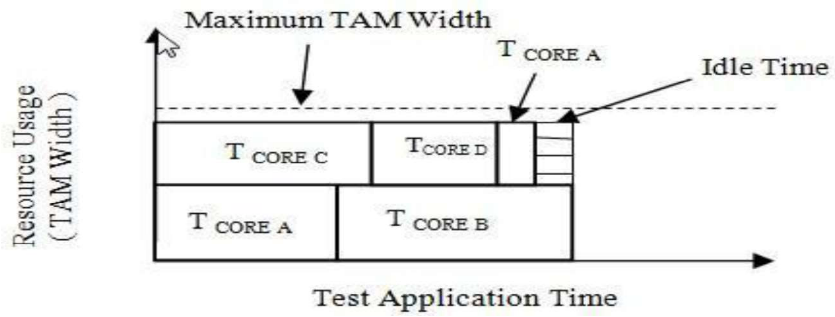


Figure 1.11(a) Non-partitioned testing [4]

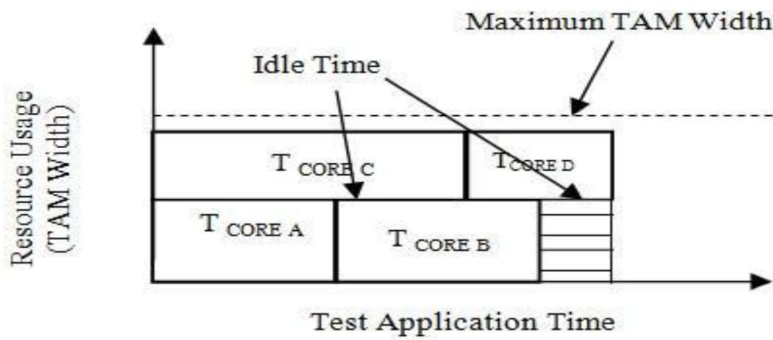


Figure 1.11(b) Pre-emptive testing [4]

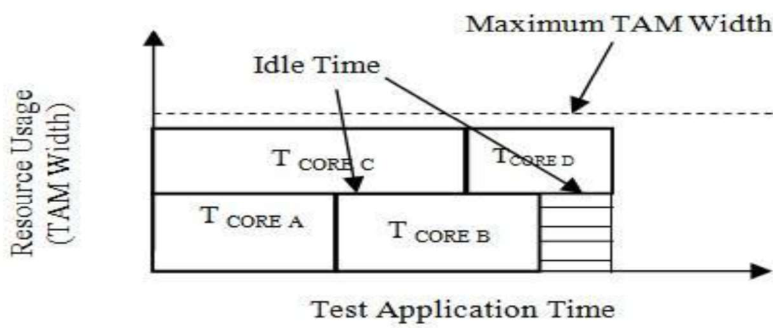


Figure 1.11(c) Partitioned testing [4]

1.5 SOC TEST CONSTRAINTS

In scheduling process, enormous amount of test data volume is one of the major challenge which leads to the large test application time. So to overcome this problem concurrent testing of cores is performed in modular fashion for SOCs. While going for concurrent testing concept, there are few important parameters that should be taken in to care for scheduling. These essential parameters are explained [8] as follows:

Power Constraint:

Power dissipation is one of the major factors to be confronted with, for the submicron integrated circuits. By increasing functionality to the ICs, there is a risk due to increase in temperature and thermal stability of ICs. Due to increase in temperature, performance may get affected and reliability also reduces. Power consumption is basically divided in two parts in VLSI: One is dynamic power dissipation and another one is static power dissipation. Dynamic power dissipation is due to the rapid switching activities and the static power dissipation is due to the increase in the leakage currents.

Dynamic power dissipation lead over in terms of chip power consumption in VLSI SOCs Switching activity performed while doing the testing operation is very rapid as compare to the normal operations performed. On some occasion, the value of power dissipation while performing testing is larger beyond the expectation. It should not damage the chip in any respect. In SOC testing, the total power consumed while doing testing of the cores is constant. So that total power dissipated shouldn't exceed the maximum value.

TAM assignment:

Another parameter is TAM assignment in which test data are being assigned to all the cores via TAM wires. However, TAM wires are not large in number but there are large numbers of inputs wire, outputs wires and bidirectional pins in SOC. Testing of all different cores cannot be done simultaneously. Each and every core must be given the TAM wires for performing the testing. TAM assignment is very important parameter for scheduling purpose and proper utilization of the width should also be done.

Precedence Constraint:

In SOC testing, it is seen that sometimes order of testing the cores is predefined and all the cores must be tested in that particular order. Therefore, scheduling must be done so that the order will be maintained.

Hierarchical Constraint:

In core based SOC, there is a concept of hierarchal cores in which cores are placed in such a manner so that cores are embedded inside the core or in the system. There is a child core which is present inside a parent core. This embedding of cores results in huge structure of the system whose testing will be a major challenge. While going for testing, many times testing of child core is done before testing the parent core by utilizing the same TAM wires.

Test data volume:

The test data volume depends on structure of the chip design. Mostly the chips designed are complex. As the complexity increases, test data volume also increases in huge amount. Thus, a cost effective technique should be used in order to suppress the cost. The mechanisms which can be used to reduce the cost are such as data compression and compaction.

1.6 THESIS OVERVIEW

This chapter defines various System on Chip design model and some preliminary issues related to testing of core based SOCs. The further organization of the thesis is as follows:

- Chapter 2 presents the previous work done in core based SOC testing and gaps in these work that have led to an idea behind pursuing my thesis work.
- Chapter 3 presents my work on SOCs test time optimization using scheduling algorithm.
- Chapter 4 shows the simulation results and the comparative analysis is done.
- Chapter 5 concludes this thesis and discusses the future scope of my work.

CHAPTER 2

LITERATURE REVIEW

In this chapter, previous work analysis is done. Under this section, an analysis of previously proposed test scheduling algorithms is done so as to develop a new and efficient test scheduling algorithm. Wrapper design as well as TAM optimization techniques is also described in brief.

2.1 MINIMIZATION OF TEST TIME

Test time is a major hurdle in the designing of an optimized test solution of SOC. The enormous amount of test data and an ever increasing design intricacy in core based SOCs lead to a prolonged test times. The analysis of Test time takes place in [13] and [14] which were introduced by Larsson and Fujiwara and in that test time is computed by the employment of multiplexing architecture (MA) along with distribution architecture (DA). In multiplex designing, all TAM bandwidth is relinquished to each and every core that means testing takes place in a predefined manner. Whereas in distribution design all the TAM bandwidths are shared among all cores as a result each core gains some exclusive portion of the TAM bandwidth. The result comes out to be that, the MA is not efficient with increasing TAM width and on the other hand DA is less efficient when TAM width diminishes or decreases.

Primary factors responsible for rising of test time:

- A dramatic increase in the transistors being used per chip.
- Test vectors are not to be applied directly onto the core because the logic behind that are isolation rings which are required around core based systems.
- From various core to core, the test method use to differ.

Test time minimization issue can be realized by focusing on one of the issue mentioned below: Core test time minimization and core access time minimization.

2.1.1 CORE TEST TIME MINIMIZATION

Diminution in core test time is associated to the test scheduling of cores which are required for testing. Agenda of algorithms have been incorporated in [8][35][9][10][11] through wrapper as well as TAM optimization[20][23]. Pair of TAM/wrapper optimization can be interspersed through test scheduling algorithms which is done in [9][10][11]. The testing of the core based system can be executed through the following mechanisms:

- Internal testing through BIST.
- External testing through ATE.
- Amalgamation of BIST and ATE.

With the employment of the BIST methodology, each core gets a predefined logic which supplies towards as the BIST pattern. In the technique of external testing, the test patterns are transmitted after assigning test buses to the multiple cores. The core to be chosen and their optimal scheduling needs to be done for achieving minimal test time.

2.1.2 CORE ACCESS TIME MINIMIZATION

One of the primary concerns while doing test scheduling is related with the access mechanism of the cores from the system's input and output terminals. There are various mechanisms which can be employed during the assignment of test patterns to a core. One mechanism is to incorporate an additional test circuit around the core for scheduling the core under test. As an instance, a multiplexer can be employed at each core input so that feed can be straightaway available from system input. A ring which is isolated can be infused to access the cores serially or linearly. Another mechanism employs bypass mode for cores to limit the issues to one of the searching path from the fundamental inputs to the core inputs and thereby from the core inputs to the system outputs. In the last approach, there is an isolation of cores from the surroundings due to the usage of dedicated bus of a test rail around a core for the movement of test data.

Core test time denoted by i usually built up with test bus width that is w to which it is appointed. Cores test time is associated with the longest length of internal scan chain of

each and every core. With the increase in the number of TAM, the value of the longest scan chain diminishes. The internal scan chains are selectively arranged using the wrapper technique .Calculation of test time usually done through the equation given in [20] and [8].

$$T_i(w_n) = (1+max(s_{ii}, s_{oo})).\ t_i + min\ (s_{ii}, s_{oo}). \qquad 2.1$$

where, s_{ii} = input scan chain length obtained using wrapper algorithm.

s_{oo} = output scan chain length calculated obtained using wrapper algorithm.

t_i = total number of test patterns for core *m.*

Further reduction in test time can be achieved if the values of *si* and *so* are taken care of. In [8] authors generally presented two heuristic algorithms to solve and rumble the partitioning of chain algorithm.

2.1.2.1 BEST FIT DECREASING ALGORITHM [8]

Best Fit Decreasing algorithm contains following essential parts are described below:

- Arrange the internal scan chains among a minimum number of scan chains so that longest wrapper scan chain length can be minimized.
- Assign the functional inputs to the internal wrapper scan chains.
- Assign the functional outputs to the internal wrapper scan chains.

This algorithm basically does the arrangement of the different scan chains of a given core in to known number of wrapper chains and then calculated the value of test time. The main requirement is to handle the wrapper scan chains such that the largest wrapper chain can be lower down. Pareto-optimal points concept is also of great use and plays a crucial part in lowering down the test times of each core. It makes use of the decreased wrapper chains for the particular time.

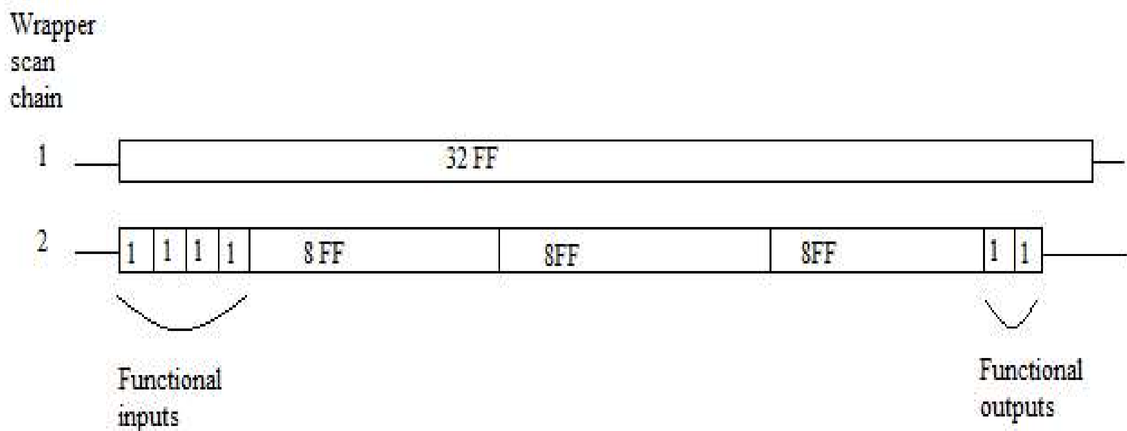


Figure 2.1 Wrapper scan chain arrangement in a core [8]

As an illustration, consider a core with four internal scan chains of length 32, 8, 8 and 8 and 4 functional inputs with 2 functional outputs. The elements which gets scanned can be accredited with at least two wrapped scan chains and *max{si,so}* and it can be calculated as 32. The wrapped chain arrangement is represented in Figure 2.1.

2.2 TAM OPTIMIZATION [21]

The TAM Optimization problem is considered to be a NP-Complete problem as proposed in [21]. Wrapper Design and wrapper design algorithm was built in [22] and [23]. Integer Linear Programming model was explained in [24] known as test bus assignment problem by combining dispensation of test buses among respective test buses of cores.. A new 2D rectangle packing problem is also shown in Figure 2.4 for wrapper and TAM optimization based which is basically upon ILP modeling with fork and merge technique using pareto-optimal points into consideration which is already proposed in [25],[26] and [27]. Also it is seen in [33] that a core clustering technique for concurrent testing of the cores has been illustrated as 3-D rectangle bin packing problem simultaneously.

2.2.1 FORK AND MERGE APPROACH [35]

As the name implies it refers to a technique in which test buses can fork and merge between cores. Core tests are represented as rectangular bins where their height represents TAM width and length represents test times. The fork and merge technique is advantageous towards reducing the test time. The main boon of fork and merge over test bus approach is that there is a very low chance of idle time which implements the wastage of memory and will be lowered as compared to that in the initial cases. In [35], fork and merge technique are employed for test scheduling. Test scheduling is considered with TAM optimization and considering power constraints. This technique can be better understood with Figure 2.2.

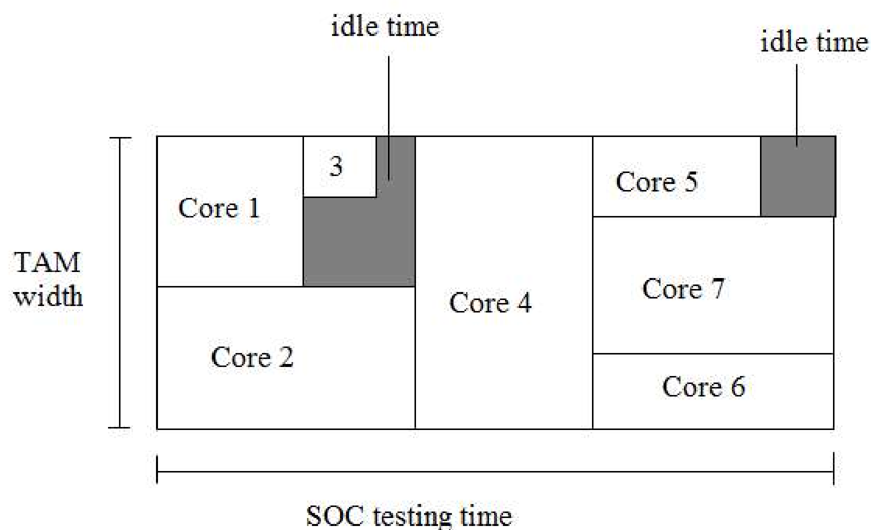


Figure 2.2 Fork and merge technique [35]

2.2.2 TEST BUS APPROACH [35]

In this technique, the whole TAM width gets divided into test buses with varying widths. While scheduling, different cores gets allocated different test buses in correspondence to their TAM widths. If the TAMs are of the same width the core is assigned to the TAM having less test time i.e. first preference is given to TAM width followed by test time. The core assignment in Test buses plays a pivotal role. This approach is highly beneficial in multi level SOCs, as each level can be assigned a particular Test bus and the core at

that level can only be tested through that Test bus [35]. The example of this approach is shown in Figure 2.3.

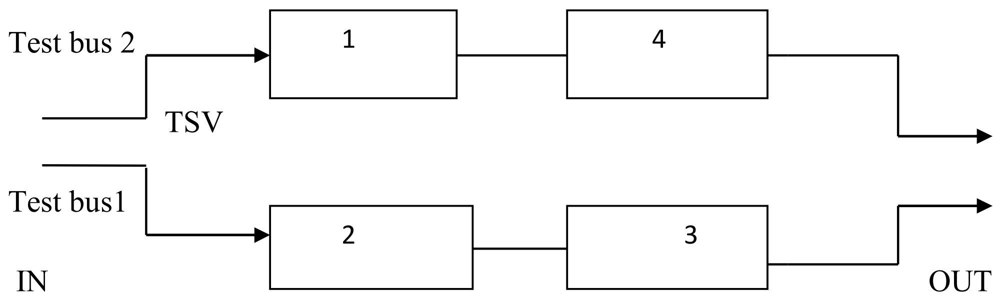


Figure 2.3 Test buses in double layer SOC

2.3 TEST SCHEDULING

Test scheduling is basically a NP-Hard problem which is already described in [4] and it can be elucidated through the combinational optimization algorithm by making use of wrapper design and TAM optimization algorithms in [18] and [25]. Different type of optimization algorithms have been discussed below:

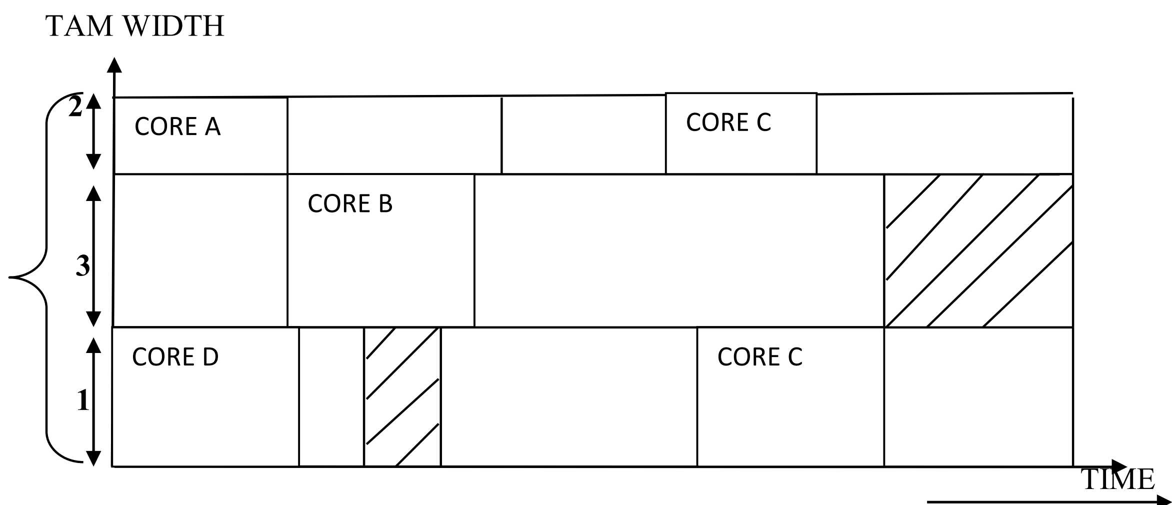


Figure 2.4 Two dimensional Rectangle bin packing representation

2.3.1 GENETIC ALGORITHM [29]

Genetic Algorithm is basically an algorithm which is based on mutative ideas of natural selection and genetics. These are the part of progressive computing, an ever increasing arena of artificial intelligence. GA's are basically motivated by Darwin's theory which is all about evolution – "survival of the fittest". GA represent irregular search which is used to solve optimization problems. In [28] a test scheduling scheme for core-based SOCs based on genetic algorithms is presented. The particular set of tasks are provided, a set of test resources is also given which is generally dedicated test access architecture. Test scheduling is carried out in order to decrease the total test application time. The result is more effective as differential evolution and self-robust mutation are taken into consideration in this algorithm

In [29] parallel elite genetic algorithm is employed with peak power as constraint in order to reduce the test application time. Similar to as offspring are generated using parent A and parent B followed by crossover and then mutation. This algorithm is tested in 2D SOC benchmark circuits and it has proved to be one of the sufficient scheduling algorithms. In [36] an efficient approach for the test scheduling problem of core-based systems is established on genetic algorithm with power as parameter constraint is proposed. In genetic core test scheduling formulation, there is chromosomal portrayal, selection and reproduction, genetic operators (mutation, crossover and fill gap). During each generation, chromosomes are selected to reproduce; as a result new test schedules are formed known as offspring.

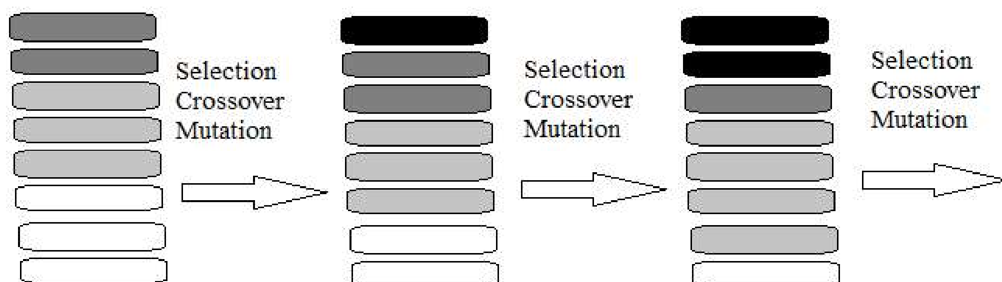


Figure 2.5 Simple genetic algorithm [29]

2.3.2 ANT COLONY OPTIMIZATION [30]

Ant colony optimization is based on real ant colony which is also described in [30]. This is all about the nature of the ant's means what type of mechanism they uses to search the smallest path from their nest to the food. While seeking new tracks for their food, ants use to excrete a hormone called pheromone. Ants smell that pheromone and usually choose the track which has very high concentration levels of pheromone as compare to other paths. During this procedure, ants may find the smallest way due to pheromone trails which was left by the previous ants from their nest to the food and vice versa shown in Figure 2.6.

Ant colony Optimization has further demonstrated the solutions to various NP problems. ACO based test scheduling has enhanced the chances of achieving better solutions in a very small interval of time. In [31], ACO scheduling algorithm is described and the test scheduling issues have been formulated as a rectangle bin packing problem and incorporated ACO to cover more number of solutions space in order further increase the chance of getting optimal solutions.

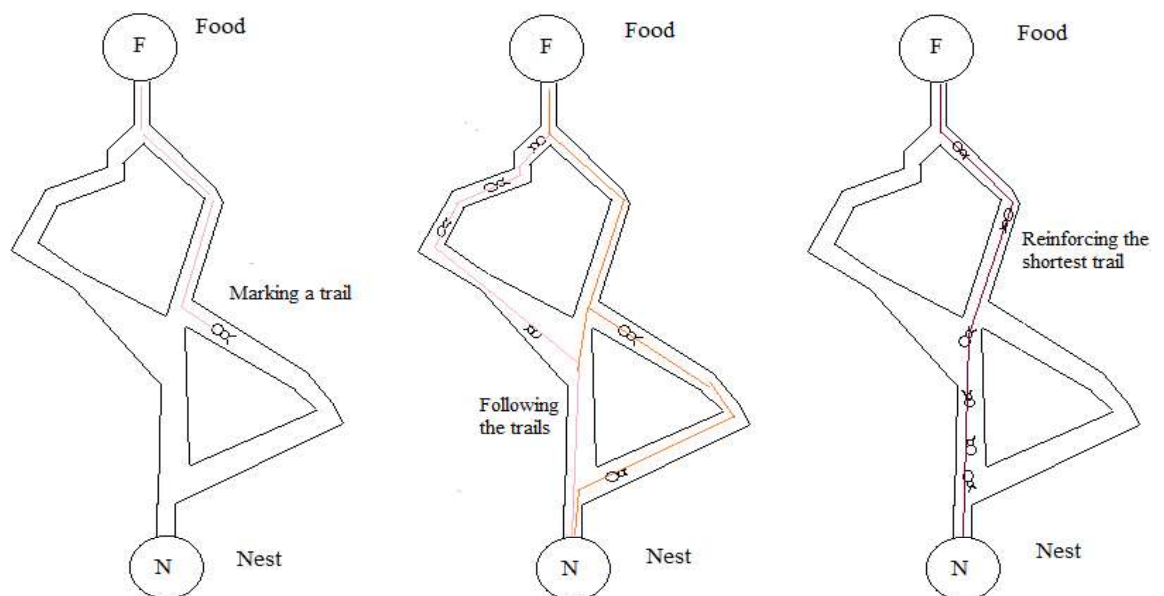


Figure 2.6 Ant colony optimization [30]

2.3.3 SIMULATED ANNEALING ALGORITHM [32]

It is one of the optimization techniques which is also used for scheduling the cores. Simulated annealing (SA) algorithm can be used to solve 2D bin packing problem where we use to optimize the wrapper design and TAM design. SA algorithm minimizes the test time by selecting the best layout among the available layouts of every cores. In this algorithm, while performing scheduling, a data structure known as sequence pair is made of use for floor planning. SA procedure is proposed in [32]. In this algorithm optimal test schedule is found by changing an initial sequence pair and by changing the width of the core wrapper.

2.3.4 RANDOM INSERTION ALGORITHM [32]

In this algorithm, cores are represented in rectangular form in [16] and [26]. The height represents the value of TAM width assigned and the width represents the test application time. For scheduling of the cores, we take into account the test application time for each core as a constraint parameter.

In [32] SOC test scheduling, sequence pair representation is proposed. The relative position of the cores in rectangular form remains unchanged even by inserting a new element into arbitrary positions of sequence pair. So, it is possible to schedule a new core without breaking down the results that are already obtained. Thus, the significance is that only two permutations are more than enough to represent the placement of cores in rectangular form. The main drawback of this algorithm is that a stark increase is observed in the total number of possible sequence pair as the number of core increases.

2.3.5 3-OPT ALGORITHM [43]

The 3-Opt is a simple local search algorithm for solving the TSP in optimization. 3-Opt algorithm is a special case of the k-opt algorithm [45]. In this algorithm, 3-Opt analysis involves deleting three connections (or edges) in a network (or tour), reconnecting the

network in all other possible ways, and then evaluating each reconnection method to find the optimum one. This process is then repeated for a different set of three connections [45]. To reduce the length of the best tour, different algorithms such as GA, ACO are used. When these algorithms try to find the best tour, they fall in local minimum and this leads to a lack of obtain lengths of the best tour. To eliminate local minimum situations, 3-Opt algorithm is used [45].

2.3.5 PARTICLE SWARM OPTIMIZATION (PSO) ALGORITHM [43]

This is one of the algorithms which were used for scheduling the different cores of SOC. PSO is a population-based optimization algorithm developed by Kennedy and Eberhart and inspired by bird flocks' behavior when searching for food [44]. Each individual in the swarm is called a particle and points to a solution in search space. Particles have dimensions similar to the number of parameters whose values are desired to be found in a problem. Particles are randomly distributed in search space at first. Each particle generates a fitness value depending on the objective function of the problem.

2.4 HIERARCHICAL CORE BASED SOC TESTING [35]

Consideration of hierarchical cores can be carried out at same level by employing the test mode whereas TAM assignment for cores embedded inside a hierarchical core is not dependent by design hierarchy. The scheduled test of wrapper design and TAM assignment done for flat cores don't get validation for hierarchical core SOCs. The hierarchical rule is in the manner that the application of the test must takes place to the parent and child core simultaneously.

A multilevel TAM architecture is described in [34] which presently show symmetrically SOC techniques can be worn for multi level TAM augmentation in hierarchical core SOCs with the help of ILP formulation for given bus width.

In [35], optimization of test time takes place for hierarchical cores with two approaches for effective testing of SOC along with hierarchical cores is proposed. In the initial approach, the problem is solved through the extension of a conventional wrapped design and this accession leaves total flexibility for optimization of TAM as well as test scheduling. The secondary approach usually based on unaltered wrapper design for the parent cores that usually operates on two disjoint modes for the testament of the parent and child cores. In [37] optimized hierarchical SOC test architecture model is basically presented. This generally simplifies the multilevel TAM optimization.

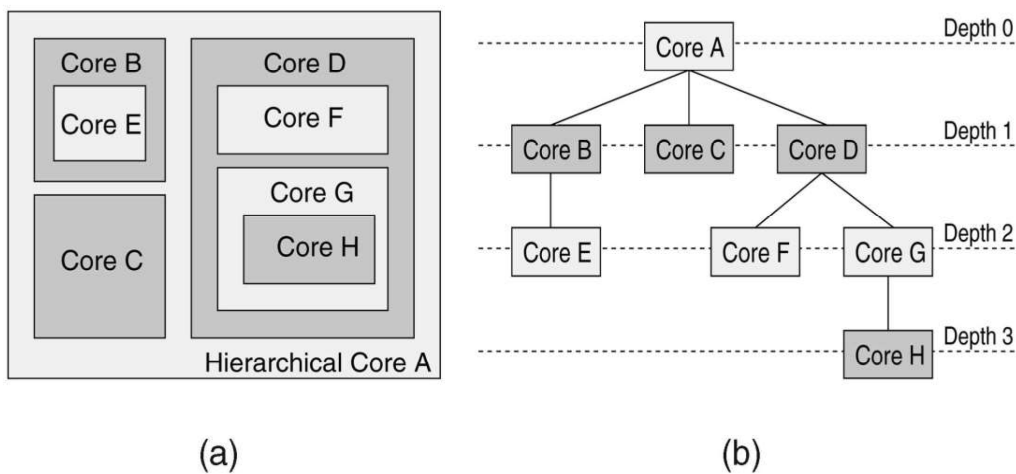


Figure 2.7 Example of (a) hierarchical core (b) Its test hierarchy tree [35].

CHAPTER 3

PROPOSED SCHEDULING TECHNIQUE

This chapter basically defines the technique/algorithm for scheduling the cores of different System on Chip. The main purpose of this is to give brief overview of the technique used for scheduling the cores.

3.1 BACKGROUND AND PROBLEM FORMULATION

Some of the issues or gaps which were found from the prior work are discussed in this section in problem form and several remedies have introduced in order to overcome it.

1. **P_C: Determine :**

   (i) assignment of different cores for the allotted value of TAM and power.

   (ii) test time minimization of SOC after scheduling of different cores.

   Taking a SOC which is containing N cores with their particular TAM widths whose sum will be W TAMs. If bus architecture is introduced for TAM then W TAMs will have widths as $w_1, w_2, \ldots\ldots w_W$. Let us consider that core m is selected for scheduling to TAM n, the test time to test core m can be given as $T_m(w_n)$ clock cycles. The test time of the core is calculated as:

$$T_i(w_n) = (1+max(s_{ii}, s_{oo})).\ t_i + min\ (s_{ii}, s_{oo}). \qquad 3.1$$

Where, s_{ii} = input scan chain length obtained using wrapper algorithm.

s_{oo} = output scan chain length calculated obtained using wrapper algorithm.

t_i = total number of test patterns for core m.

let x_{ii} be a variable of value 0 or 1 (where $1 \leq m \leq N$ and $1 \leq n \leq W$). It can be defined as follows:

$$x_{mn} = \begin{cases} 1, & \text{if core m is scheduled using TAM } n. \\ 0, & \text{if core is not scheduled at all.} \end{cases} \qquad 3.2$$

Therefore, the time needed to test all cores using W TAM is given as

$$\text{Test time of all cores} = \sum_{m=1}^{N} T_m\,(w_n).\,x_m \qquad 3.3$$

All the TAMs can be used for testing therefore the overall test time can be given as

$$\text{Total test time} = max_{\,1 \leq n \leq W}\,\left(\sum_{m=1}^{N} T_m\,(w_n).\,x_m\right)$$

3.2 SCHEDULING ALGORITHM

A major challenge is test scheduling while testing the core based SOC which defines the order in which cores are needed to be checked. Test scheduling of cores even for a simple SOC is basically equivalent to NP-hard problem. An efficient scheduling approach must minimize the overall test time of the cores while considering resource constraints among all the cores due to the use of different TAMs wire, BIST engines and power constraint. The minimum test time can be achieved by taking maximum test time of individual cores. Hence, here in we have worked upon the efficient way by which all cores can be handled by making optimum use of TAM width, thereby saving power which plays a very pivotal role in determining the efficiency of any algorithm. Power dissipation is a function of time and mainly depends upon switching activity of test vectors to the system. SOC in test mode can dissipate around twice the power as they do in normal mode, since all the cores do not operate normally in parallel and can be tested concurrently in order to minimize test time. A divide and conquer based approach is being employed basically to reduce increase in dynamic power dissipation. Total SOC test time means the time taken to test all the cores which are embedded in the chip. The user has no idea of their placement inside the chip, which needs to be minimized to the maximum extent. Therefore, power constrained is required for test scheduling to limit some amount of concurrency. There had already been many approaches defined earlier for test scheduling of core based SOCs. In this chapter, different other but an efficient method is being defined for SOC test scheduling.

After doing the analysis on previous work done, some of the modifications have been done in the algorithm to get optimum test time. Greedy algorithm is proposed for test scheduling problem. Already the test time calculation of each core is being calculated using BFD algorithm which is also described in [8]. Wrapper design algorithm is used for computing test time of each core and explained in [8] whereas scheduling of all the cores are done using greedy algorithm.

3.2.1 EFFICIENT TEST SOLUTION TECHNIQUE (ETT) USING GREEDY ALGORITHM

Greedy algorithm basically solves the problem of choosing local constraints at each and every stage. This algorithm might provide the better optimal solutions that can be used as global optimal solutions for particular time instant. It is one of the important test scheduling technique used for different cores. Greedy algorithm will provide appropriate test time to get minimum test length at last.

Scheduling of different cores of different bench mark circuits are performed using greedy algorithm which is done using test bus architecture. It is based on TAM design architecture. This algorithm will arrange the different cores greedily according to the value of their bandwidth and will not cross the maximum value of power. The test time of each core is provided with the help of Wrapper design techniques, in which various modifications had been done already in order to calculate the minimum test time of each core.

The algorithm greedily arranges the cores with respect to their bandwidth and schedules the cores with total bandwidth and total power. The total test time can be calculated and stored. Then after that, another schedule is formed by re-arranging the cores e.g. with respect to the test time of cores or power of the cores. The new test time is compared with the previous one and the lowest test time is considered to be the best schedule.

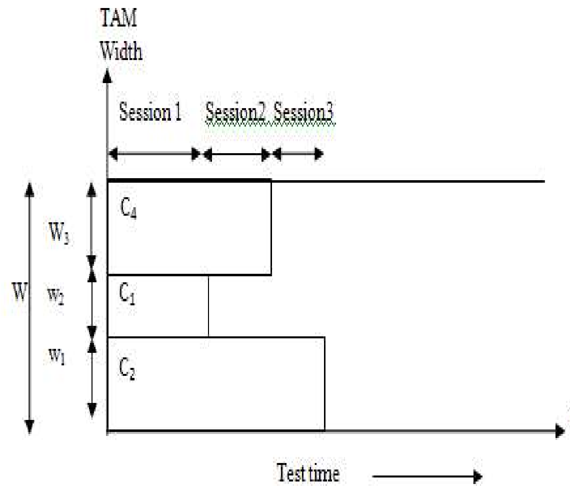


Figure 3.1 Three Cores scheduled simultaneously initially

In this algorithm, three partitions have been done for arrangement of the cores. All the cores are arranged in such a manner that it should not exceeds the maximum power limit and maximum TAM width. For different TAM width, cores are being scheduled for getting overall test time. Total width is divided into various partitions for scheduling of the different cores to calculate the optimum value of test time of the cores.

In the above figure 3.1, three partitions are done and initially three cores are scheduled of different TAM width with different power and test time values. Here all the three cores are being tested simultaneously and as soon as the testing of any of the core is over, other core of that particular TAM width will be added (scheduled) to that particular core. It is called Partitioned scheduling. Scheduling basically means that sequence of core for given TAM. Among different scheduling scheme, partitioned scheduling is preferred for scheduling. This scheme will provide better results when compared with other schemes.

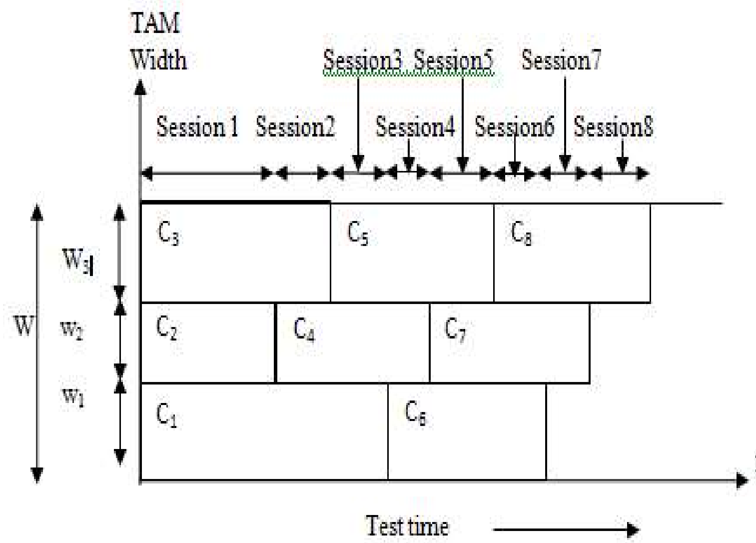


Figure 3.2 Different Cores scheduled as the testing of any core is completed

The representation of eight cores in rectangle bin packing form is shown above in the figure 3.3. Total partition value is divided into three partitions and cores of different width are represented pictorially which are scheduled are in the figure 3.3. The y-axis represents the bandwidth of the particular core and the x-axis represents the value of overall test time of the entire core. The cores should be tightly bounded in order to get the optimum value of test time. The scheduling technique should not cross the maximum power and bandwidth which is being provided while using this algorithm.

i) Greedy Algorithm for scheduling different cores while considering lower value of Power is shown below:

Step1. Initialize the value of TAM width.

Step2. Construct three partitions of TAM width value as tw_1, tw_2 *and* tw_3 and take the test time of different TAM width partitioned for scheduling of the cores.

Step3. Different possible combination is made using *for* loop for the particular TAM width value.

```
for i= 1; i<=n/2          //   outer loop
{l= n-i ;                 // where , n is value of TAM width
for j= 1; j<=l/2          //   inner loop
{k= n-i ;}}               // where  i,j,k,l are integers
```

Minimum test time *tt* is taken among all the possible combinations where, *tt* is sum of the test time of the cores.

tt_j ($\forall$ j $\in$ k) represents test time of each core.

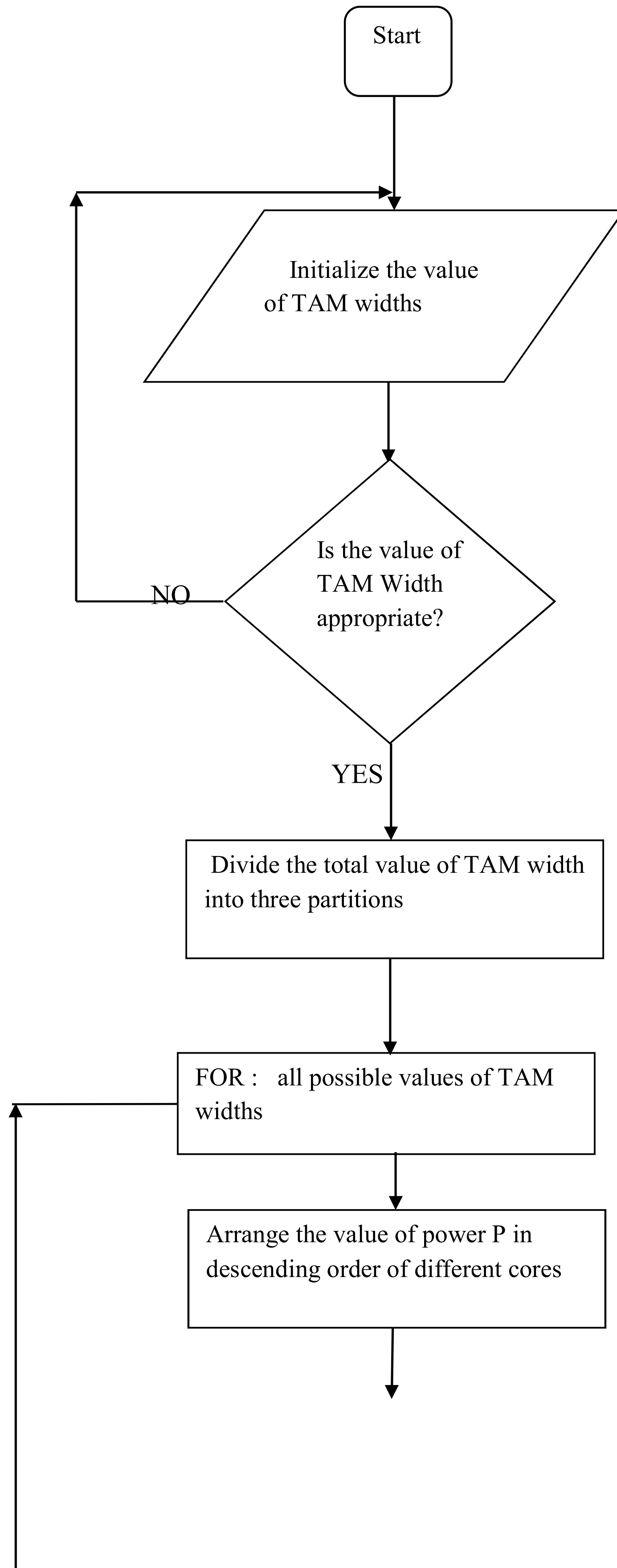
Step4. Sort the value of power P of different cores in descending order using bubble sorting technique and select the two cores of maximum power along with the core of minimum power for scheduling. Update start time as well as end time. As testing of any core is completed the fourth core of minimum power P is added.

Step5. Sort the value of test time *tt* of remaining cores in descending order and schedule the cores from higher to lower test time *tt*. Update the respective cores accordingly and update the test time of each partition.

Step6. Repeat step 7 until all the remaining cores get scheduled.

Step7. Final test time *tt* value is the minimum test time *tt* among all the possible Combinations. Test time *tt* is represented by the partition with maximum value.

Flowchart:



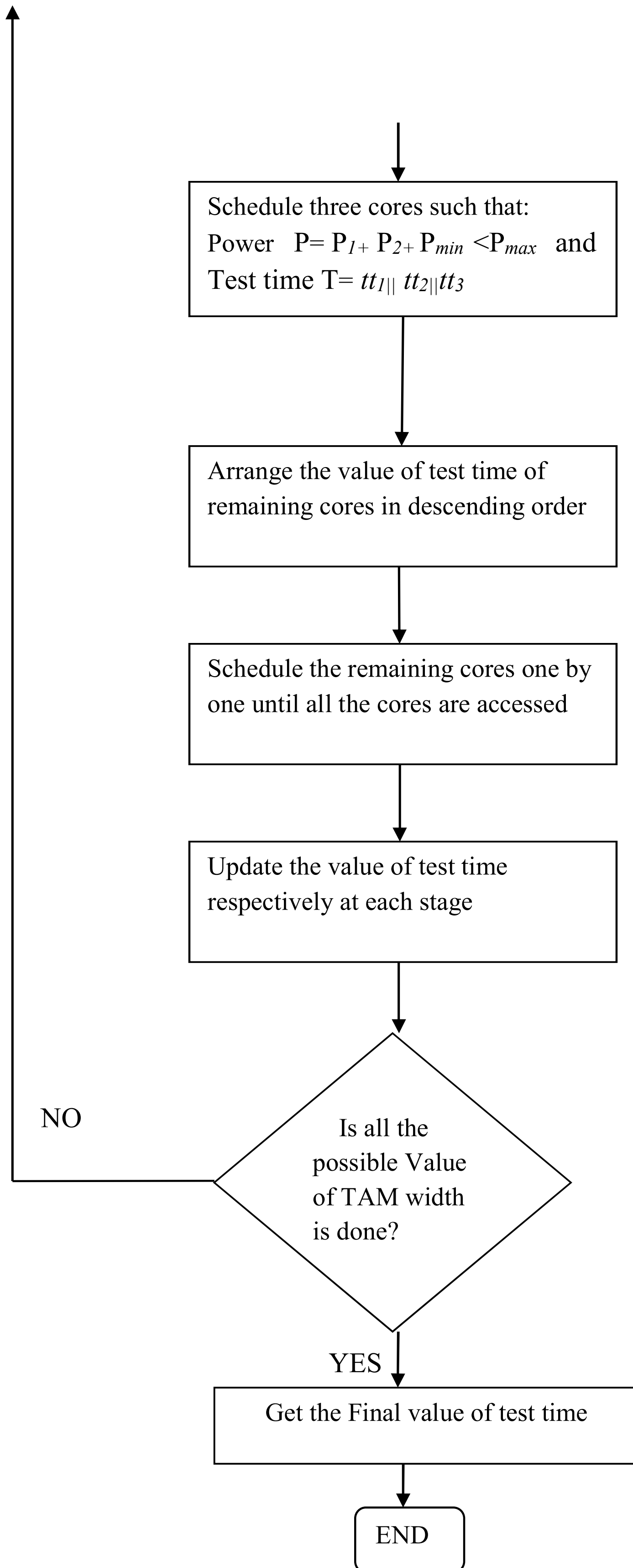


Figure 3.3 Flowchart representation for proposed test optimization technique with power constraints

ii) Greedy Algorithm for scheduling different cores while considering higher value of Power constraint is shown below:

Step1. Initialize the value of TAM width.

Step2. Construct three partitions of TAM width value as tw_1, tw_2 *and* tw_3 and take the test time of different TAM width partitioned for scheduling of the cores.

Step3. Different possible combination is made using *for* loop for the particular TAM *for*

```
i= 1; i<=n/2              //  outer loop
{ l= n-i ;                // where , n is value of TAM width
for j= 1; j<=l/2          //  inner loop
{ k= n-i ;}}              // where  i,j,k,l are integers
```

Minimum test time *tt* value is taken among all the possible combinations

where, *tt* is sum of the test time of the cores.

tt_j ($\forall$j $\in$ k) represents test time of each core.

Step4. Sort the value of test time *tt* of different cores in descending order using bubble sorting technique.

Step5. Select the three cores of maximum test time *tt* of different partition values and schedule them. Update start time as well as end time. Total value of the power P will not be exceeding P_{max}.

Step6. As testing of any core is completed remaining cores are scheduled from higher to lower test time TT. Update the respective cores as well as the test time of each partition.

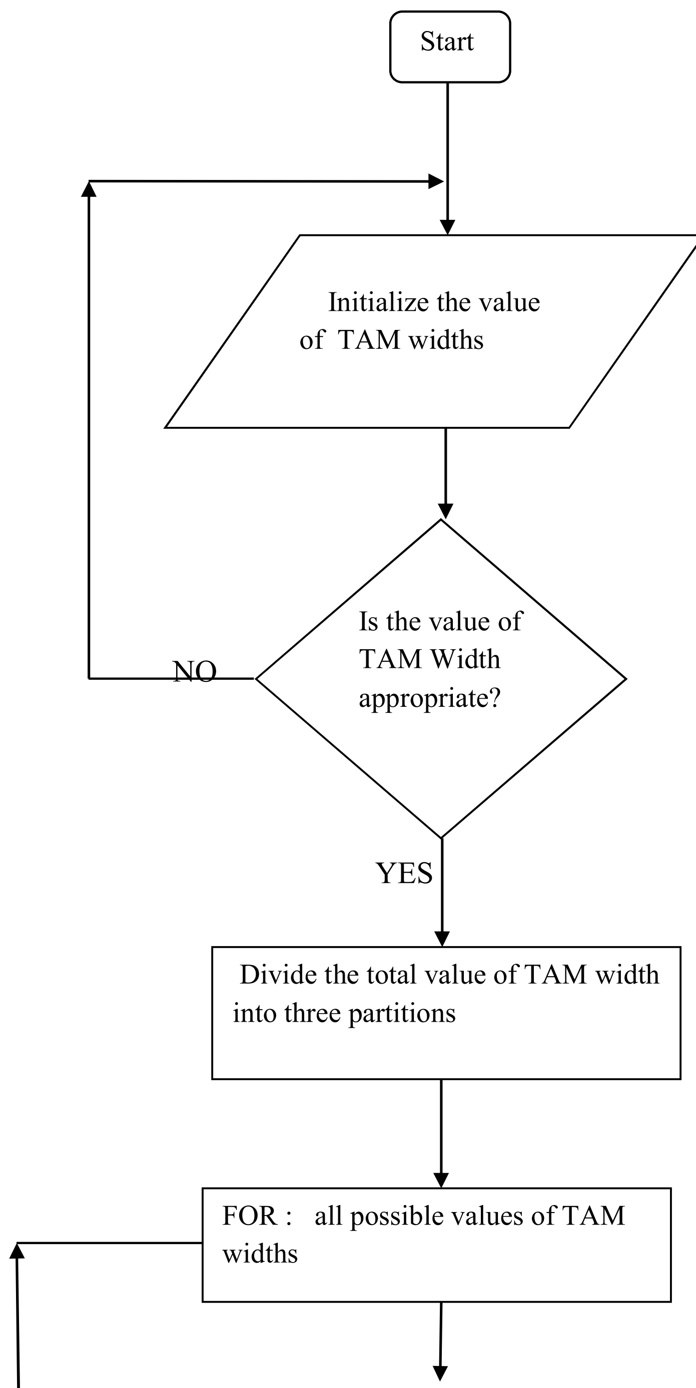
Step7. Repeat step 6 until all the remaining cores get scheduled.

Step8. Final test time *tt* is the minimum test time among all the possible cominations. Test time *tt* is of all scheduled cores are represented by the partition with larger values.

Earlier, it had been seen that in case of two partitions technique, proper utilization of TAM width was not there. Because of which the overall test time used to increment and other constraints also gets affected. Here, the care of this thing has been taken ito

consideration and by using all the TAM width and other parameters, the overall test time for different benchmark circuit have been calculated.

Flowchart:



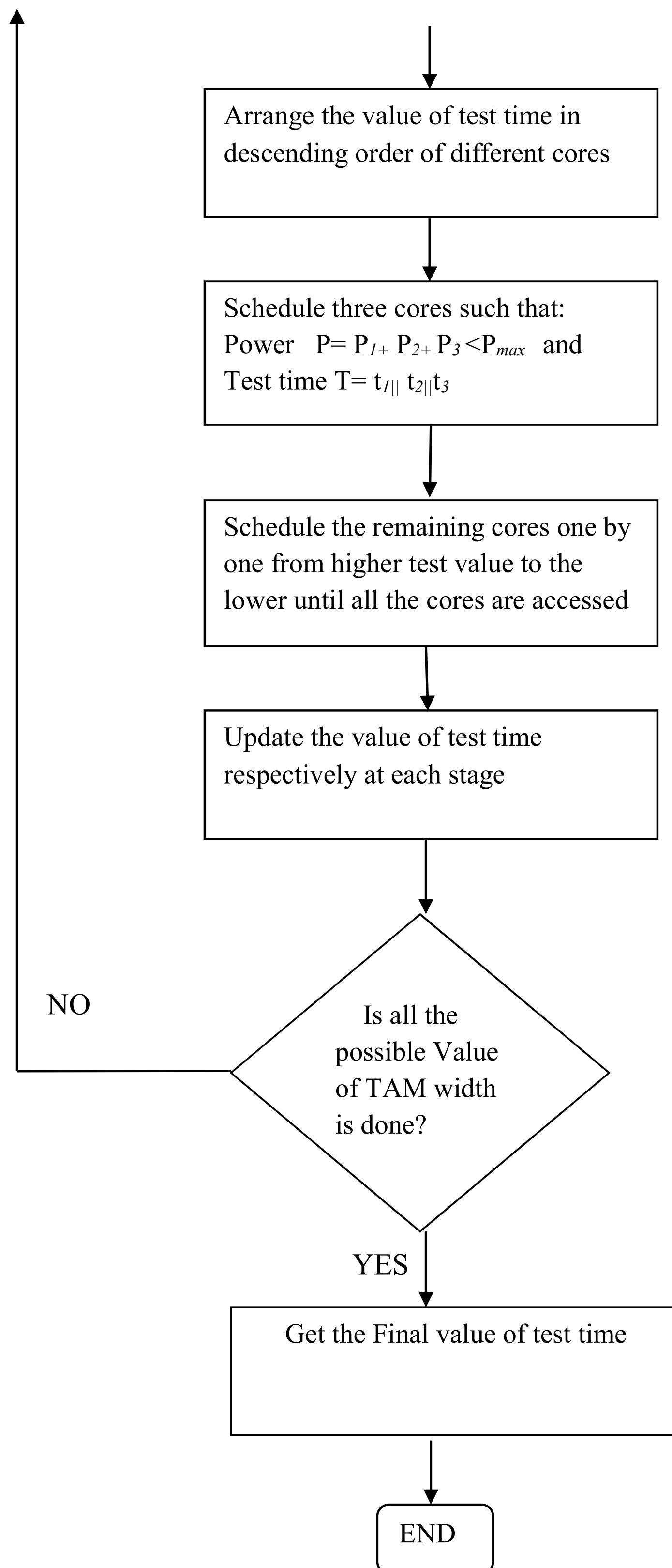


Figure 3.4 Flowchart representation for proposed test optimization technique

CHAPTER 4

SIMULATION RESULTS

Several experiments have been done in order to get the better results for different SOC. Overall test times have been calculated using greedy algorithm which is discussed in previous chapter. Test scheduling algorithms are used to calculate the optimum test time for core based SOC system. While doing this, different parameters are also taken in to consideration such as Power and TAM width. In this, the results are shown which is conducted on ITC'02 test benchmark circuits. While doing calculations flat cores and hierarchical cores are considered for scheduling purpose. Scheduling technique and the algorithm which is mentioned above is used to get minimum possible test time for different SOC. Greedy algorithm is the Efficient test technique (ETT) through which optimum values of test times are obtained. The usage of hard core is done whose test time is known from wrapper design technique. The experimental results for SOC d695, p22810 and p93791 have been shown in this chapter.

4.1 TEST TIME RESULTS USING GREEDY ALGORITHM

In this section, results of test scheduling are shown in the table. While performing calculation wrapper design results for flat cores and hierarchical cores are taken into consideration as an input in test scheduling algorithm and using these results scheduling of the cores is done and overall test time is being calculated for different SOC. The Wrapper design results using enhanced algorithm is shown below in the figure for SOC d695, p22810 and p93971 which is further taken as an input file for scheduling purpose.

1 to 64 data of d695.txt

1	428	220	154	116	102	89	76	64	63	63	50	50	50	50	50	38	37	37	37	37	37	37	37	37
2	15292	7719	5146	3896	3160	2646	2278	1984	1764	1616	1469	1396	1249	1175	1102	1028	1028	955	881	881	808	807	734	734
3	5058	2550	2475	2475	2475	2475	2475	2475	2475	2475	2475	2475	2475	2475	2475	2475	2475	2475	2475	2475	2475	2475	2475	2475
4	26602	13354	11098	6676	5796	5794	5788	5784	5782	5781	5780	5779	5779	5778	5778	5778	5778	5777	5777	5777	5777	5777	5777	5776
5	191874	95992	61210	47996	37931	31727	26733	24053	20748	19752	14973	15424	14870	14859	14852	12082	10095	9988	9986	9983	9975	9970	9966	9964
6	185794	93014	59674	46507	37361	28191	28189	23254	19018	19023	19019	18779	18775	18538	18536	11744	10071	9844	9851	9854	9855	9853	9849	9847
7	65686	32891	20914	16398	12862	9877	9688	8247	6605	6516	6421	6417	6413	6410	6408	4124	3524	3333	3339	3343	3345	3346	3346	3343
8	22427	11165	8702	5583	4576	4583	4575	4571	4568	4567	4566	4565	4564	4563	4563	4563	4562	4562	4562	4562	4561	4561	4561	4561
9	26351	13182	8514	6597	5238	4392	3702	3305	2892	2806	2110	2142	2109	2096	2088	1659	1412	1414	1411	1402	1394	1388	1384	1382
10	120188	59992	38845	29996	24685	21157	17647	15032	14119	14106	10605	10600	10589	10584	10582	7518	7092	7081	7072	7068	7065	7063	7062	7061
11																								
12	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Figure 4.1 Wrapper design output file of d695

1 to 64 data of d695.txt | 1 to 64 data of p22810_corrected.txt

1	690	345	234	173	142	122	102	92	82	71	71	61	61	51	51	51	41	41	41	41	41	41	31
2	6062	3031	2051	1516	1248	1070	892	803	714	624	624	535	535	446	446	446	357	357	357	357	357	357	268
3	926665	456660	347329	260869	189419	174409	173611	173606	173603	102951	102891	102863	102853	102849	102846	102844	102843	102842	102841	102840	102840	102839	102839
4	591585	308116	209519	160220	135570	110921	98596	86272	86271	73947	73947	61622	61622	61622	61622	49298	49297	49297	49297	49297	49297	49297	49297
5	121238	62173	43520	34194	27977	24868	21759	18651	18650	15542	15542	15542	12434	12433	12433	12433	12433	12433	9325	9325	9325	9325	9325
6	14478	7350	4900	3786	2896	2450	2227	2004	1782	1559	1337	1336	1114	1114	1114	1113	891	891	891	891	891	890	890
7	480681	240341	158940	119360	95399	79565	71429	60076	53368	51327	43633	43618	43598	43579	43572	43547	43546	43544	43540	43537	43528	43512	43515
8	57736	29224	19957	14968	12116	10690	9265	7840	7127	6414	6413	5701	5700	4988	4988	4276	4275	4275	4275	3563	3563	3562	3562
9	213256	107944	73717	55288	44756	39490	34225	28960	26327	23694	23693	21061	21060	18428	18428	15796	15795	15795	15795	13163	13163	13162	13162
10	96512	49560	33909	26084	23475	18258	18258	15650	13041	13041	13041	10433	10433	10433	10433	10433	10432	7824	7824	7824	7824	7824	7824
11	415000	206974	138139	103632	86198	69127	68565	52048	51324	51137	40108	34622	34436	34420	34416	34414	33712	33536	33531	33529	33527	22501	21586
12	10119	5094	3842	3817	3810	3808	3806	3805	3804	3804	3803	3803	3803	3802	3802	3802	3802	3802	3801	3801	3801	3801	3801
13	61336	30748	21332	15453	14480	13062	10130	8409	8377	8377	8373	8371	8370	8369	8369	8368	8368	8368	8368	8367	8367	8367	8367
14	70952	35595	23934	18011	15944	12087	11976	11881	11786	9831	7764	7731	7730	7726	7724	7723	7722	7722	7721	7721	7721	7721	7720
15	778	386	246	187	154	146	141	137	132	129	126	123	121	119	118	117	116	115	114	114	113	113	112
16	11001	8012	8001	7997	7995	7995	7994	7994	7993	7993	7993	7993	7993	7993	7993	7993	7993	7992	7992	7992	7992	7992	7992
17	19223	9615	6420	5533	4153	3225	3014	3011	3009	3007	3005	3004	3003	3002	3001	3001	3000	3000	3000	3000	2999	2999	2999
18	1346	923	901	894	890	888	887	886	885	884	884	883	883	883	883	882	882	882	882	882	882	882	881
19	8737	4364	2908	2272	2267	2264	2261	2259	2258	2257	2257	2256	2255	2255	2255	2254	2254	2254	2254	2253	2253	2253	2253
20	237962	119281	80573	72843	48358	44441	44441	44439	44438	44438	44437	44437	44437	44437	44437	44437	44436	44436	44436	44436	44436	44436	44436
21	8191	4118	2752	2566	2562	2560	2559	2557	2556	2556	2555	2555	2554	2554	2554	2554	2554	2553	2553	2553	2553	2553	2553
22	34619	19209	11622	9706	9692	9685	9682	9680	9678	9677	9677	9676	9676	9675	9675	9675	9674	9674	9674	9674	9674	9674	9674
23	545180	273021	182136	136931	130407	92191	88004	87529	87059	87056	86964	86991	86980	86974	86970	86967	86965	86964	86963	86962	86961	86961	86960
24	13245	6646	4634	4622	4615	4612	4610	4609	4608	4607	4607	4606	4606	4605	4605	4605	4604	4604	4604	4604	4604	4604	4604
25	13137	6586	4726	4678	4661	4658	4657	4642	4641	4642	4642	4641	4641	4641	4641	4640	4640	4640	4640	4640	4640	4640	4640
26	7061	3547	2775	2770	2763	2760	2759	2757	2757	2757	2756	2756	2756	2755	2755	2755	2755	2755	2755	2755	2755	2755	2755
27	514283	257243	171710	136427	106863	86177	83067	76797	62604	59717	53474	53252	53249	53248	46768	46774	46772	39244	39176	39153	39145	39141	39139
28	2102430	1058399	716897	535625	438301	375457	329152	278342	256711	232050	205888	205696	193929	183916	159432	145052	145030	145025	145023	145021	145021	133255	133074
29	768	369	247	185	148	125	107	93	84	79	79	78	77	77	76	76	75	75	75	74	74	74	74
30	12576	6304	4775	3231	2684	2647	2645	2640	2636	2634	2633	2632	2631	2630	2630	2629	2629	2629	2629	2628	2628	2628	2628
31																							
32																							
33	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Figure 4.2 Wrapper design output file of p22810

1 to 64 data of d695.txt | 1 to 64 data of p22810_corrected.txt | 1 to 64 data of p93791 - Copy.txt

1	2833432	1423001	945022	740329	567012	493918	431596	370508	315278	308183	305313	247915	246272	244222	189408	186418	186000	185180	181497	180668	179441	177799	126977	126558
2	7906	4049	2891	2120	1734	1541	1348	1156	1155	963	963	962	962	770	770	770	770	769	769	577	577	577	577	577
3	26597	13622	9729	7135	5837	5188	4540	3891	3891	3242	3242	3242	3242	2594	2593	2593	2593	2593	2593	1945	1945	1945	1945	1945
4	1652	810	536	394	311	262	227	201	178	166	165	139	128	128	128	117	117	117	117	105	105	104	82	78
5	631161	318644	214471	165449	134810	110299	98043	85788	79659	73532	67404	61276	55149	55148	49021	49021	42893	42893	42893	42893	36765	36765	36765	36765
6	5301239	2650719	1809477	1358118	1125916	906759	792825	678999	674497	565038	560613	455381	451131	450890	446720	341501	341403	341379	341368	337001	336997	332834	230807	227621
7	5850	3014	1950	1596	1241	1064	887	887	709	709	532	532	532	532	532	532	355	355	355	355	355	355	355	355
8	5850	3014	1950	1596	1241	1064	887	887	709	709	532	532	532	532	532	532	355	355	355	355	355	355	355	355
9	8482	4433	3083	2312	1926	1733	1540	1348	1155	1155	963	962	962	962	770	770	770	769	769	769	769	577	577	577
10	312080	157204	104802	79184	64045	53565	46578	40756	36098	32604	30275	27946	25617	24453	22124	20960	19795	18631	18630	17466	16302	16301	15137	15137
11	135845	68008	45457	34168	30597	22908	19726	17247	15548	15555	15551	15533	15531	15529	15528	15528	15527	15527	15526	15526	15526	15526	15526	15526
12	1785278	892834	595322	446762	363293	297811	255490	223726	217860	181797	181795	149251	145732	145340	145339	112013	109276	109276	109275	108884	108884	108883	77994	74775
13	1879524	939847	639831	480322	395498	320812	278491	242226	235586	199912	193280	165570	159340	153098	152711	123254	123051	123048	116828	116827	110588	110397	85173	85012
14	1879524	939847	639831	480322	395498	320812	278491	242226	235586	199912	193280	165570	159340	153098	152711	123254	123051	123048	116828	116827	110588	110397	85173	85012
15	12994	6641	4619	3464	2886	2597	2308	2020	1731	1731	1443	1442	1442	1442	1154	1154	1154	1153	1153	1153	1153	865	865	865
16	54712	27752	18633	14272	11496	9514	8325	7532	6739	5946	5549	5153	4756	4360	4360	3964	3963	3567	3567	3171	3171	3170	2774	2774
17	1418234	709261	483147	354667	290017	257229	224894	192782	161550	160445	129408	129197	128542	127890	97100	97080	97075	96856	96206	95773	95554	64740	64756	64747
18	3394	1739	1187	890	720	635	550	466	423	381	381	338	338	296	296	254	254	253	253	211	211	211	211	211
19	1016074	508247	338766	254228	203305	169593	145756	127170	112998	104390	92618	84958	83289	83067	67845	63642	62607	62604	62596	62590	62166	46413	44426	42535
20	3163654	1582035	1057385	791422	636629	551561	476927	416463	360968	345978	290614	285907	271345	261326	215850	215435	215423	215421	200861	200444	186682	150491	145371	145368
21	3394	1739	1187	890	720	635	550	466	423	381	381	338	338	296	296	254	254	253	253	211	211	211	211	211
22	1840	941	641	512	426	341	298	298	255	255	213	212	212	170	170	170	170	169	169	169	127	127	127	127
23	1819997	914038	606970	475258	364230	317576	277387	240717	202553	200527	194427	160581	160339	156826	121714	120397	120388	120387	120386	120151	116640	114055	80466	80446
24	55300	30722	21505	18433	15360	12288	12288	12288	9216	9216	9216	9216	9216	9216	9216	9216	6144	6144	6144	6144	6144	6144	6144	6144
25	80656	43016	29573	24196	18819	16130	16130	13442	13441	10753	10753	10753	10753	10753	8065	8065	8064	8064	8064	8064	8064	8064	8064	8064
26	4162	2129	1451	1160	966	773	676	676	579	579	483	482	482	386	386	386	386	385	385	385	289	289	289	289
27	2803245	1402080	954550	713387	588665	480460	418104	365826	341075	303468	279645	248443	248440	231951	217280	187003	186085	186084	186084	179671	170511	154932	125611	125563
28	43610	22201	15064	11496	9118	7928	6739	5946	5549	4757	4360	4360	3963	3567	3567	3171	3170	3170	2774	2774	2774	2378	2378	2378
29	1149163	580444	387030	290323	230073	193616	164338	161058	129087	128880	128704	96909	96704	96699	96525	96351	96350	64731	64530	64524	64521	64347	64347	64345
30	8482	4433	3083	2312	1926	1733	1540	1348	1155	1155	963	962	962	962	770	770	770	769	769	769	769	577	577	577
31	30466	15335	10427	7769	6338	5315	4702	4088	3679	3271	3066	2861	2657	2453	2248	2248	2044	2043	1839	1839	1839	1635	1635	1634
32	829724	416404	280686	209744	169645	141885	123378	107956	95618	86364	80195	74026	67857	64773	58604	55520	52435	49351	49350	46266	43182	43181	40097	40097
33																								
34	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Figure 4.3 Wrapper design output file of p93791

p22810 hierar.txt | 1 to 64 data of p22810.txt | p22810.soc

1	2134	1799	1687	1626	1596	1575	1555	1545	1535	1525	1524	1514	1514	1504	1504	1504	1494	1494	1494	1494	1494
2	12319	7101	5916	4725	3986	3667	3428	3397	3258	3124	3124	3053	3053	2932	2932	2932	2817	2817	2817	2817	2817
3	778	386	246	187	154	146	141	137	132	129	126	123	121	119	118	117	116	115	114	114	113
4	11001	8012	8001	7997	7995	7995	7994	7994	7993	7993	7993	7993	7993	7993	7993	7993	7993	7992	7992	7992	7992
5	19223	9615	6420	5533	4153	3225	3014	3011	3009	3007	3005	3004	3003	3002	3001	3001	3000	3000	3000	3000	2999
6	1346	923	901	894	890	888	887	886	885	884	884	883	883	883	883	882	882	882	882	882	882
7	8737	4364	2908	2272	2267	2264	2261	2259	2258	2257	2257	2256	2255	2255	2255	2254	2254	2254	2254	2253	2253
8	237962	119281	80573	72843	48358	44441	44441	44439	44438	44438	44437	44437	44437	44437	44437	44437	44436	44436	44436	44436	4443
9	8191	4118	2752	2566	2562	2560	2559	2557	2556	2556	2555	2555	2554	2554	2554	2554	2554	2553	2553	2553	2553
10	34619	19209	11622	9706	9692	9685	9682	9680	9678	9677	9677	9676	9676	9675	9675	9675	9674	9674	9674	9674	9674
11	545180	273021	182136	136931	130407	92191	88004	87529	87059	87056	86964	86991	86980	86974	86970	86967	86965	86964	86963	86962	8696
12	13245	6646	4634	4622	4615	4612	4610	4609	4608	4607	4607	4606	4606	4605	4605	4605	4604	4604	4604	4604	4604
13	13137	6586	4726	4678	4661	4658	4657	4642	4641	4642	4642	4641	4641	4641	4641	4640	4640	4640	4640	4640	4640
14	7061	3547	2775	2770	2763	2760	2759	2757	2757	2757	2756	2756	2756	2755	2755	2755	2755	2755	2755	2755	2755
15	514283	257243	171710	136427	106863	86177	83067	76797	62604	59717	53474	53252	53249	53248	46768	46774	46772	39244	39176	39153	3914
16	2102430	1058399	716897	535625	438301	375457	329152	278342	256711	232050	205888	205696	193929	183916	159432	145052	145030	145025	145023	145021	1450
17	768	369	247	185	148	125	107	93	84	79	79	78	77	77	76	76	75	75	75	74	74
18	12576	6304	4775	3231	2684	2647	2645	2640	2636	2634	2633	2632	2631	2630	2630	2629	2629	2629	2629	2628	2628
19																					
20																					
21	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

Figure 4.4 Wrapper design output file for hierarchical cores of p22810

p93791 hierar.txt | p22810 hierar.txt

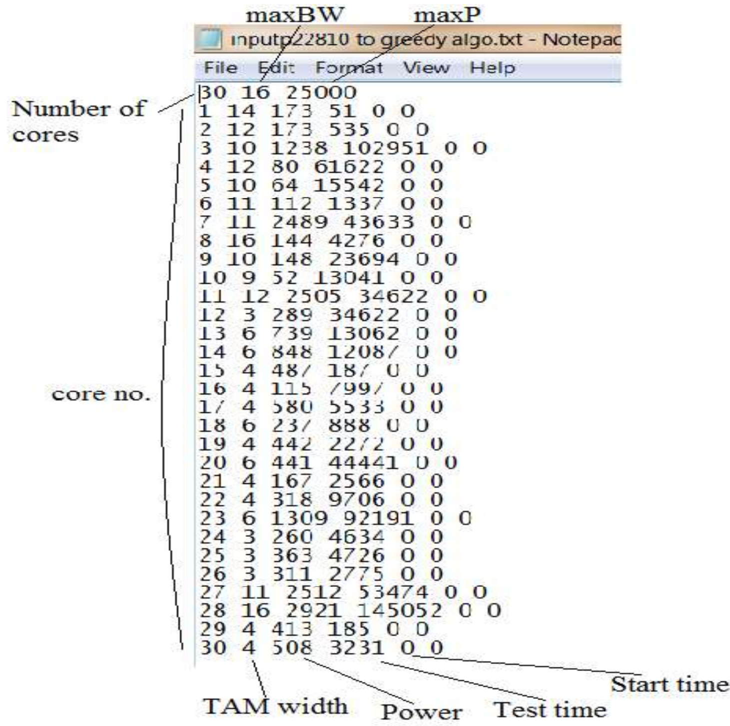
1	1820	1315	1298	1240	1203	1949	1879	1827	1788	1683	1602	1547	1462	1442	1402	1346	1287	1180	1149	1068	954	912	877
2	7906	4049	2891	2120	1734	1541	1348	1156	1155	963	963	962	962	770	770	770	770	769	769	577	577	577	577
3	631161	318644	214471	165449	134810	110299	98043	85788	79659	73532	67404	61276	55149	55148	49021	49021	42893	42893	42893	42893	36765	36765	36765
4	2500145	2255014	2180977	2171822	2169156	2163793	2152825	2151729	2094397	2016508	1975321	1865811	1741211	1659123	1561720	1492021	1440215	1399871	1356712	1249821	1843		
5	135845	68008	45457	34168	30597	22908	19726	17247	15548	15555	15551	15533	15531	15529	15528	15528	15527	15527	15526	15526	15526	15526	15526
6	1785278	892834	595322	446762	363293	297811	255490	223726	217860	181797	181795	149251	145732	145340	145339	112013	109276	109276	109275	108884	108884	108883	77994
7	1879524	939847	639831	480322	395498	320812	278491	242226	235586	199912	193280	165570	159340	153098	152711	123254	123051	123048	116828	116827	110588	110397	85173
8	1879524	939847	639831	480322	395498	320812	278491	242226	235586	199912	193280	165570	159340	153098	152711	123254	123051	123048	116828	116827	110588	110397	85173
9	12994	6641	4619	3464	2886	2597	2308	2020	1731	1731	1443	1442	1442	1442	1154	1154	1154	1153	1153	1153	1153	865	865
10	54712	27752	18633	14272	11496	9514	8325	7532	6739	5946	5549	5153	4756	4360	4360	3964	3963	3567	3567	3171	3171	3170	2774
11	1418234	709261	483147	354667	290017	257229	224894	192782	161550	160445	129408	129197	128542	127890	97100	97080	97075	96856	96206	95773	95554	64740	64756
12	3394	1739	1187	890	720	635	550	466	423	381	381	338	338	296	296	254	254	253	253	211	211	211	211
13	1016074	508247	338766	254228	203305	169593	145756	127170	112998	104390	92618	84958	83289	83067	67845	63642	62607	62604	62596	62590	62166	46413	44426
14	3163654	1582035	1057385	791422	636629	551561	476927	416463	360968	345978	290614	285907	271345	261326	215850	215435	215423	215421	200861	200444	186682	150491	14537
15	3394	1739	1187	890	720	635	550	466	423	381	381	338	338	296	296	254	254	253	253	211	211	211	211
16	1840	941	641	512	426	341	298	298	255	255	213	212	212	170	170	170	170	169	169	169	127	127	127
17	1819997	914038	606970	475258	364230	317576	277387	240717	202553	200527	194427	160581	160339	156826	121714	120397	120388	120387	120386	120151	116640	114055	80466
18	55300	30722	21505	18433	15360	12288	12288	12288	9216	9216	9216	9216	9216	9216	9216	9216	6144	6144	6144	6144	6144	6144	6144
19	80656	43016	29573	24196	18819	16130	16130	13442	13441	10753	10753	10753	10753	10753	8065	8065	8064	8064	8064	8064	8064	8064	8064
20	4162	2129	1451	1160	966	773	676	676	579	579	483	482	482	386	386	386	386	385	385	385	289	289	289
21	2803245	1402080	954550	713387	588665	480460	418104	365826	341075	303468	279645	248443	248440	231951	217280	187003	186085	186084	186084	179671	170511	154932	12561
22	43610	22201	15064	11496	9118	7928	6739	5946	5549	4757	4360	4360	3963	3567	3567	3171	3170	3170	2774	2774	2774	2378	2378
23	1149163	580444	387030	290323	230073	193616	164338	161058	129087	128880	128704	96909	96704	96699	96525	96351	96350	64731	64530	64524	64521	64347	64347
24	8482	4433	3083	2312	1926	1733	1540	1348	1155	1155	963	962	962	962	770	770	770	769	769	769	769	577	577
25	30466	15335	10427	7769	6338	5315	4702	4088	3679	3271	3066	2861	2657	2453	2248	2248	2044	2043	1839	1839	1839	1635	1635
26	829724	416404	280686	209744	169645	141885	123378	107956	95618	86364	80195	74026	67857	64773	58604	55520	52435	49351	49350	46266	43182	43181	40097
27																							
28	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Figure 4.5 Wrapper design output file for hierarchical cores of p93791

Results of test scheduling using greedy algorithm are shown in the table. The overall test times for SOC d695, p22810 and p93791 are described below in the table. Test times have been calculated for different SOC using greedy algorithm. The test times are being calculated by using scheduling technique for the hard cores while taking TAM width and power constraint. Various inputs are taken for different TAM width for getting optimum value of test time. Here inputs for TAM width equal to 16 with power value is shown in the figure 4.4, 4.5 and 4.6 for d695, p22810 and p93791 which are basically flat SOCs. These values are subjected to algorithm for getting the optimum value of test time for the TAM equal to 16. Likewise numbers of combinations have been subjected and best possible result is being calculated for different value of TAM width.

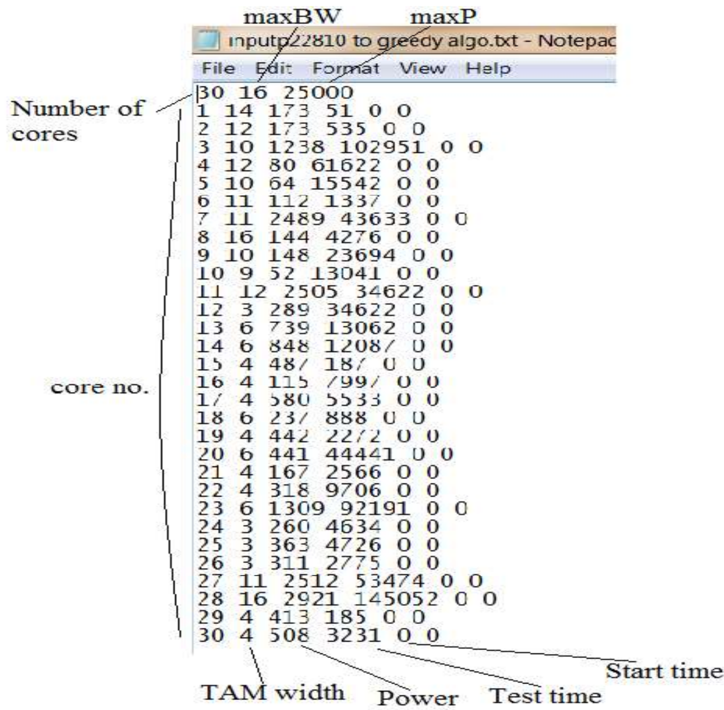
```
10 16 3000
1 11 660 50 0 0
2 14 602 1175 0 0
3 3 823 2475 0 0
4 5 275 5796 0 0
5 11 690 14973 0 0
6 16 354 11744 0 0
7 9 530 6605 0 0
8 6 753 4583 0 0
9 11 641 2110 0 0
10 16 1144 7518 0 0
```

Figure 4.6 Input data of d695 with power constraint



.

Figure 4.7 Input data of p22810 with power constraint



.

Figure 4.8 Input data of p93791 with power constraint

For different TAM width values and for different power constraints, the test time are being calculated is shown below in the table for different SOC like d695, p22810 and p93791. It has been found that the results shows the optimum value of test time as compare to some some prior work done. While doing scheduling, hard cores test times have been considered.

Table I: SOC d695 Test time for various Power constraint

TAM Width	Pmax=1800/1500	Pmax=2000	Pmax=2500
W=64	14462	14462	12972
W=48	19397	19397	16922
W=32	25935	25935	24200
W=24	30837	30837	30608
W=20	38466	38466	34729
W=16	50471	50471	48497
W=12	64334	64334	59896

Table II: SOC p22810 Test time for various Power constraint

TAM Width	Pmax=5000/6000	Pmax=8000	Pmax=10000
W=64	135965	126167	126167
W=48	232578	218620	218620
W=32	278527	267809	267809
W=24	338167	326071	326071
W=20	375704	377257	377257
W=16	456754	438301	438301
W=12	591009	573939	573939

Table III: SOC p93791 Test time for various Power constraint

TAM Width	Pmax=20000	Pmax=25000	Pmax=30000
W=64	443094	443094	360140
W=48	581540	581540	550551
W=32	911832	911832	905535
W=24	1290195	1290195	1199937
W=20	1449474	1449474	1422714
W=16	1841433	1841433	1757802
W=12	2392978	2392978	2324929

Graphical representation of above result is shown below:

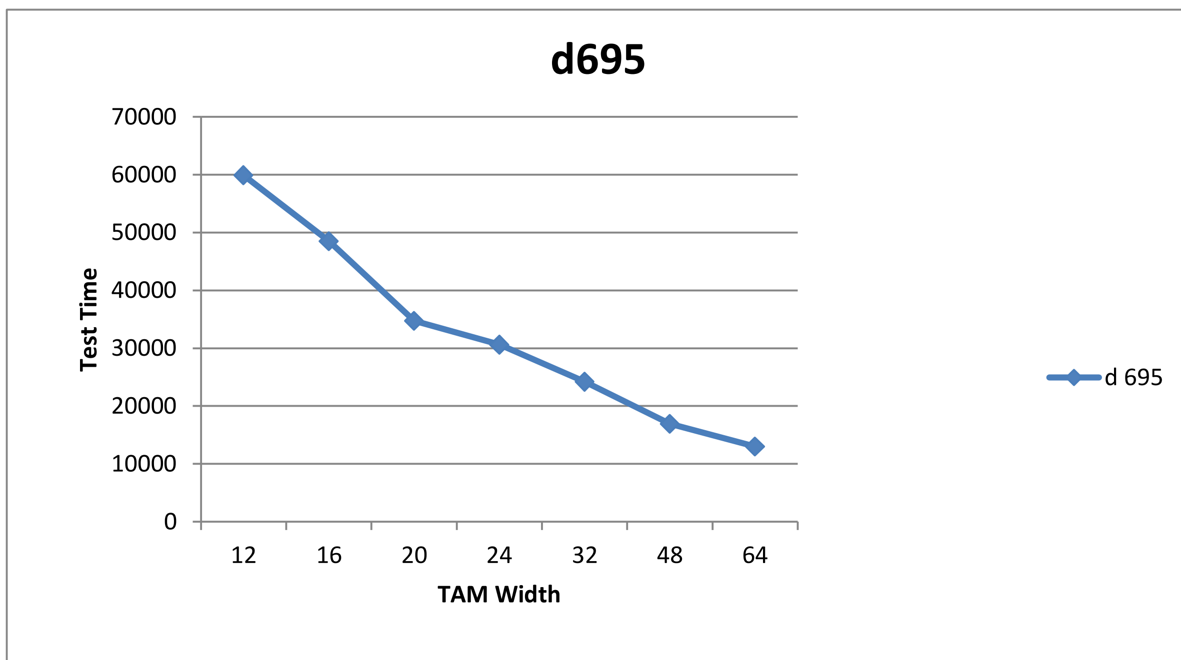


Figure 4.9 Graphical representations of Test time vs TAM width for SOC d695

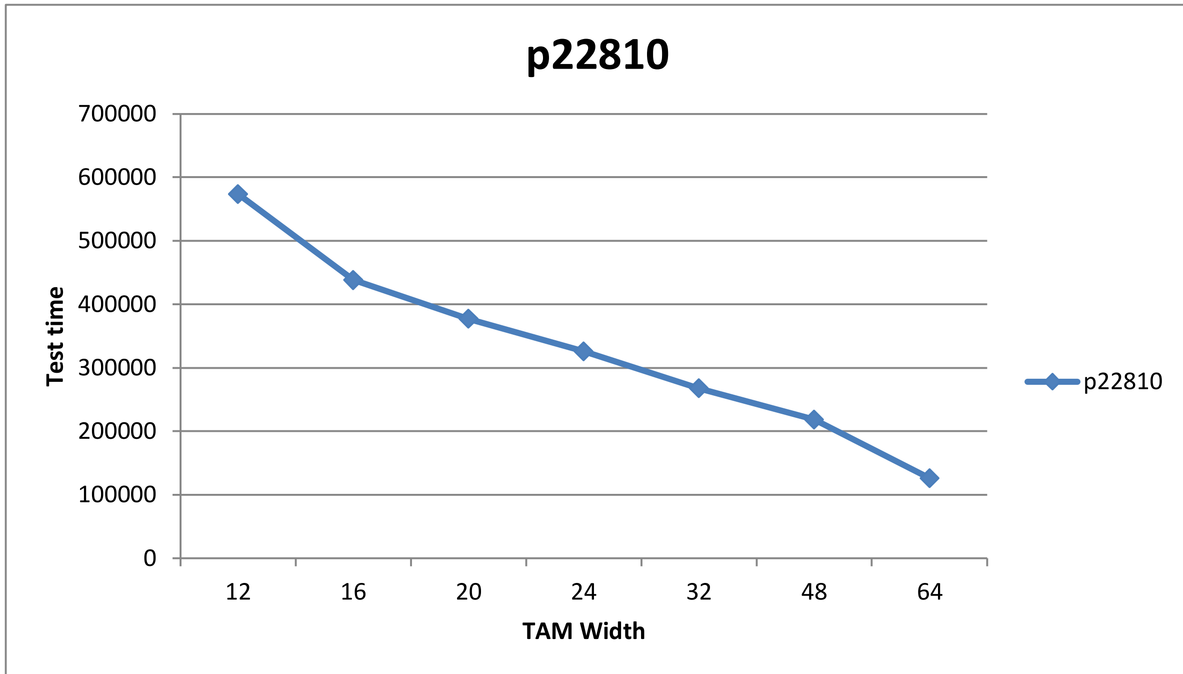


Figure 4.10 Graphical representations of Test time vs TAM Width for SOC p222810

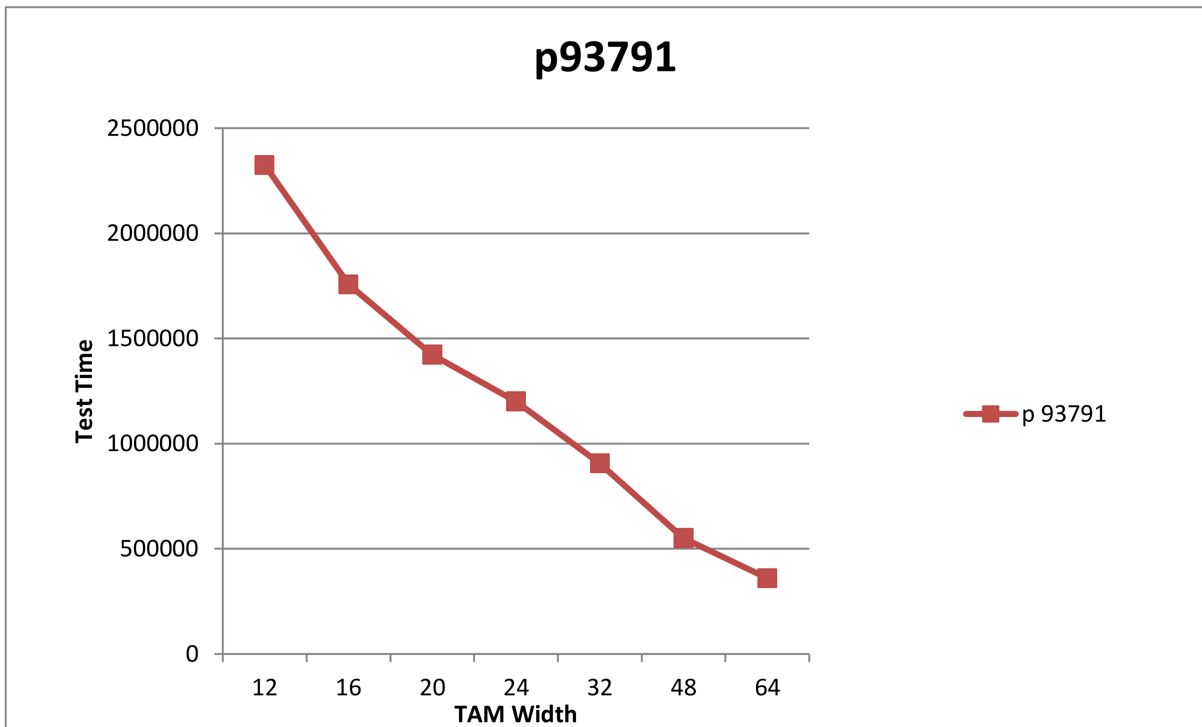


Figure 4.11 Graphical representations of Test time vs TAM Width for SOC p93791

Table IV: SOC p22810/p93791 Test time under hierarchical constraint

TAM Width	p22810	p93791
8	1058399	2257270
12	675625	2185624
16	596445	2155542
24	529890	2093169
32	476721	1868133

4.2 RESULTS COMPARISON

In the following table, results are compared with few papers such as [35] and [8]. It is seen that the test time of various SOCs have been reduced while comparing with [35] and [8]. The test time calculations are shown for three benchmark circuits using hard cores. Table 4.3,4.4 and 4.5 represents the test time of SOCs d695, p22810 and p93791 using hard cores and power constraint.

4.2.1 Comparison of Greedy with E.J Marinssen [35]

For different SOCs such as d695, p22810 and p93791 comparison is done with [35]. It has been observed that the test times obtained from efficient test technique (ETT) are the optimized values. For different SOCs and for different TAM width value, comparison is done with [35]. The results compared are shown below in the following tables:

Table V: Test time comparison between ETT and [35] for SOC p22810

TAM Width	ETT	Modified Schedule [35]	Modified Wrapper Cell [35]
16	438301	621895	466667
32	267809	493290	229899
64	126167	296445	133405

Table VI: Test time comparison between ETT and [35] for SOC p93791

TAM Width	ETT	Modified Schedule [35]	Modified Wrapper Cell [35]
16	1841433	3363819	1792354
32	911832	1504806	917246
64	443094	1908099	458600

Graphical representation of results compared in above table:

Figure 4.12 Comparison of ETT with [35] graphically for SOC p22810

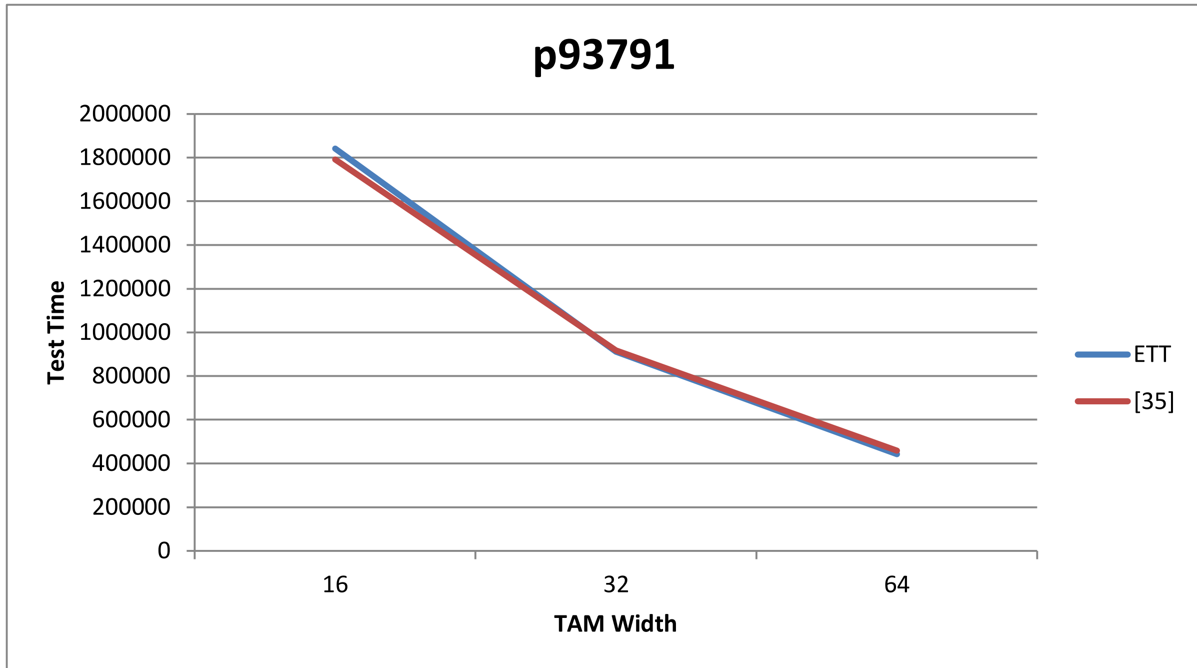


Figure 4.13 Comparison of ETT with [35] graphically for SOC p93791

4.2.2 Comparison of Greedy with J. Pouget[8]

For different SOCs such as d695, p22810 and p93791 comparison is done with [8].]. It has been observed that the test times obtained from efficient test technique (ETT) are the optimized values. For different SOCs and for different TAM width value, comparison is done with [8]. The results compared are shown below in the following tables:

Table VII: Test time comparison between ETT and [8] for SOC d695

Max Power	TAM Width=32		TAM Width=48		TAM Width=64	
	ETT	[8]	ETT	[8]	ETT	[8]
Pmax=1800	25935	42450	19397	32054	14462	23864
Pmax=2000	25935	42450	19397	29106	14462	21942
Pmax=2500	24200	41847	16922	29106	12972	21931

Table VIII: Test time comparison between ETT and [8] for SOC p22810

Max Power	TAM Width=32		TAM Width=48		TAM Width=64	
	ETT	[8]	ETT	[8]	ETT	[8]
Pmax=6000	278527	475951	232578	346461	135965	250487
Pmax=8000	267809	473418	218620	352834	126167	236186
Pmax=10000	267809	473418	218620	352834	126167	236186

Table IX: Test time comparison between ETT and [8] for SOC p93791

Max Power	TAM Width=32		TAM Width=48		TAM Width=64	
	ETT	[18]	ETT	[18]	ETT	[18]
Pmax=20000	911832	1827819	581540	1220469	443094	957921
Pmax=25000	911832	1827819	581540	1220469	443094	965383
Pmax=30000	905535	1827819	550551	1220469	360140	945425

Graphical representation of results compared in above table:

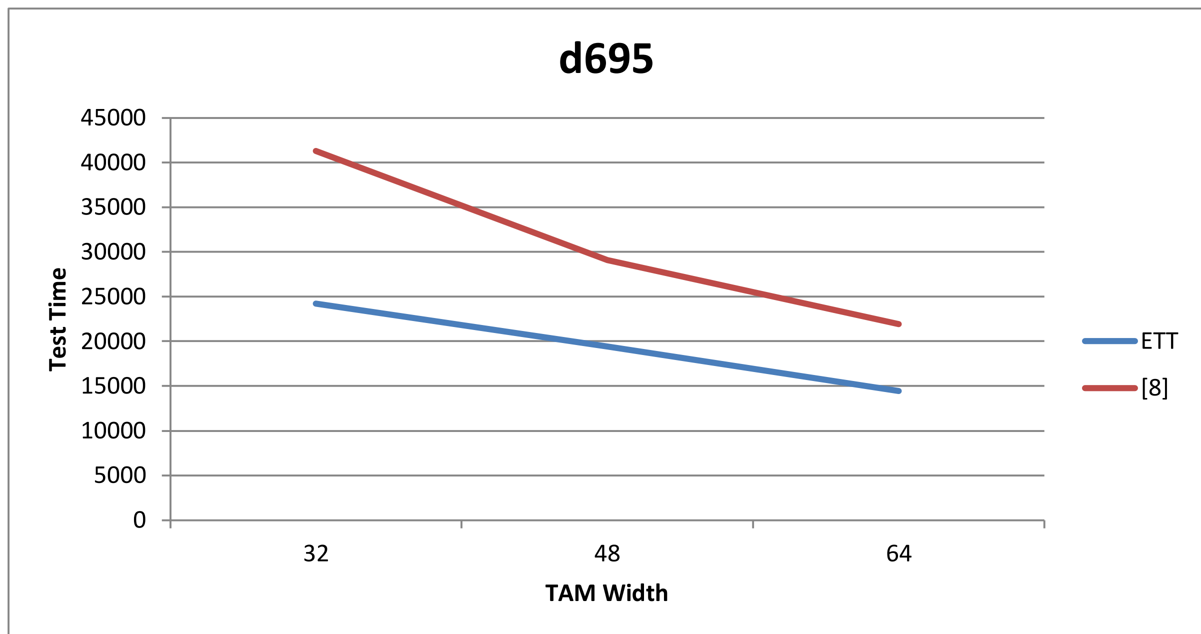


Figure 4.14 Comparison of ETT with [8] graphically for SOC d695

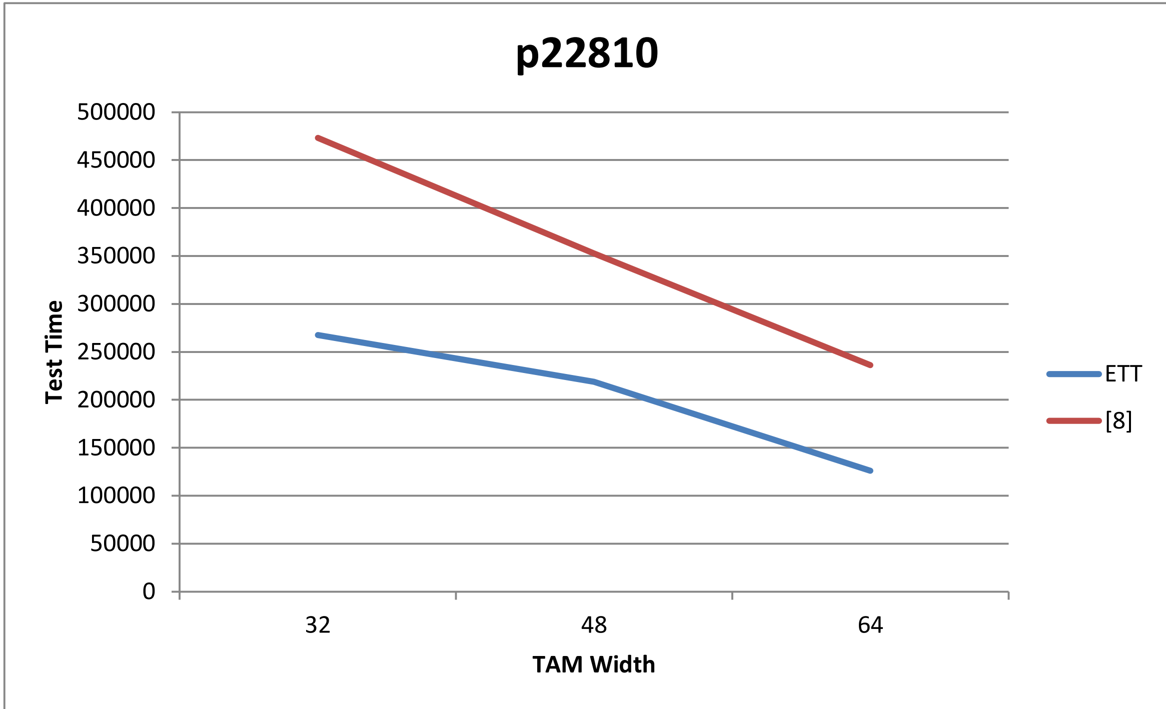


Figure 4.15 Comparison of ETT with [8] graphically for SOC p22810

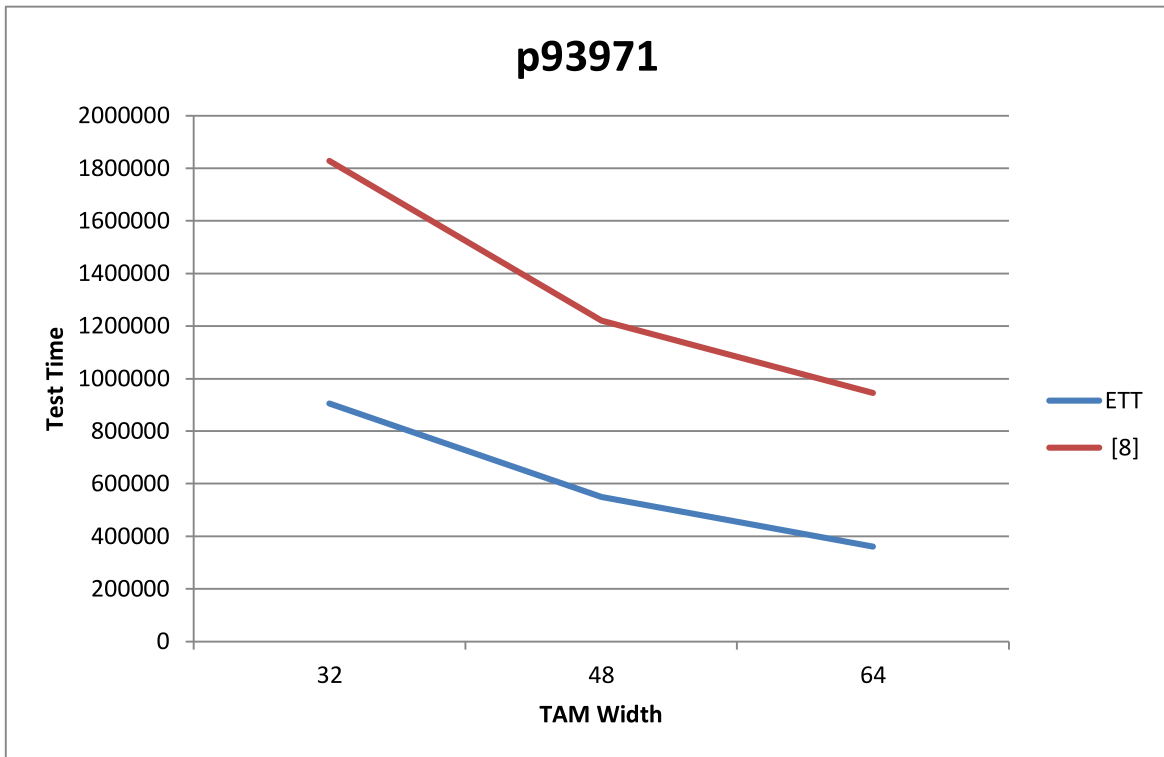


Figure 4.16 Comparison of ETT with [8] graphically for SOC p93791

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this report, scheduling algorithm have been discussed for 2D SOCs, in which various constraints such as power and TAM width have been taken into consideration. The results for three different SOCs have been calculated and also mentioned in the previous chapter. The results show valuable decrease in the test time and after comparing it with previous paper in chapter 4, the results seems to be productive.

Greedy algorithm has been proposed for scheduling purpose which is basically a heuristic algorithm. It can be further improved by implementing some new logic. Using this algorithm, test times have been reduced to an extent while considering other constraints. Simulation results show the optimized value of test times.

The future work in field of test scheduling can be done by considering latest techniques such as PSO, 3 Opt algorithms. Algorithms can be defined for 3D SOCs in which a new constraint is introduced i.e. TSV constraint. By using 3D concept, test times can further be reduced. It will be going to play a crucial role in decreasing the test time of different SOCs.

REFERENCES

[1]. SOC Test: the devil is in the details of integration/implementation.

[2]. G. E. Moore, "Cramming more components onto integrated circuits", Electronics, pp 114-117, 1965.

[3]. Intel, 2003, http://www.intel.com/research/silicon/mooreslaw.htm.

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

[5] J. Aerts and E.J. Marinissen, "Scan Chain Design for Test Time Reduction in Core-Based ICs", Proceedings of International Test Conference (ITC), Washington, DC, USA, pp 448-457, 1998.

[6] P. Varma and S. Bhatia, "A Structures Test Re-Use Methodology for Core-Based System Chips", Proceedings of International Test Conference (ITC), Washington, DC, USA, pp 294-302, 1998,.

[7] V. Iyenger, K Chakrabarty, and E.J. Marinissen, "Recent Advances in test Planning for Modular Testing of Core-Based SOCs", Proceedings of Asian Test Symposium (ATS), 2002.

[8] J. Pouget, E. Larsson and Z. Peng, "Multiple-Constraint Driven System-on-Chip Test time Optimization," Springer Journal of Electronic Testing: Theory and Applications 21, pp 599-611, 2005.

[9] V. Iyenger, K. Chakrabarthy and E.J. Marinissen, "On Using Rectangle Pack-ing for SOC Wrapper/TAM Co-optimization," Proceedings of 20th IEEE VLSI Test Symposium, 253-258, May 2002.

[10] J.M. Im, S. Chun, G. Kim, J. H. An and S. Kang, "RAIN Scheduling Algorithm for SOC Test," Asian Test Symposium 2004.

[11] C. Giri, D.K.R. Tipparthi, S. Chattopadhyay, "A Genetic Algorithm Based Approach for System-on-Chip test Scheduling using Dual Speed TAM with Power Constraint," WSEAS Transactions on Circuits and Systems, 2008.

[12] A. Colorni, V. Maniezzo and M. Dorigo, "Ant System: Optimization by a colony of cooperating agents," IEEE Transactions on Systems, Man and Cybernetics, pp 432-436, 1996.

[13] E. Larsson and H. Fujiwara, "Power Constrained Preemptive TAM Scheduling", Informal Proceedings of European Test Workshop (ETW), Corfu, Greece, pp 119-126, 2002.

[14] E. Larsson and H. Fujiwara, "Preemptive system on chip test scheduling," IEICE Transactions on Information and Systems, pp 620-629, 2004.

[15] M. Sugihara, H. Date and H. Yasuura, "A novel test methodology for core based system LSIs and is testing time minimization problem," Proceedings International Test Conference, pp 465-472, 1998.

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

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

[18]. V. Iyenger, K. Chakrabarthy, E.J. Marinissen, "Test Access mechanism, Test Scheduling and Tester Data Volume Reduction for System on Chip," Proc IEEE Int'l Test Conference, pp 319-324, 2003.

[19]. K. Chakrabarty, "Test scheduling for Core-Based System," proceedings Intl. Conf CAD, pp 391-394, 1999.

[20]. V. Iyenger, K. Chakrabarty and E.J. Marinissen, "Test Wrapper and Test Access Mechanism Co-Optimization for System –on-Chip", ITC International Test Conference, pages 1023-1032,2001.

[21]. K. Chakrabarthy, "Test Scheduling for Core-Based System," Proc. Intl. Conf. Cad, pp. 391-394, 1999.

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

[23]. 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.

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

[25]. Y. Huang et al., "Resource Allocation and Test Scheduling for Concurrent Test of Core-Based SOC Design," Proc. 10th IEEE Asian Test Sym., pp. 265-270, 2001.

[26]. S.K. Goel and E.J. Marinissen, "Effective and Efficient Test Architecture Design for SOCs," Proc. IEEE Int'l Test Conf., pp 529-538, 2002.
[27]. V. Iyenger, K. Chakrabarthy and E.J. Marinissen, "On using Rectangular Packing for SOC Wrapper/TAM co-optimization," Proc. 20th IEEE VLSI Test Symposium, 2002.
[28]. Z. jinyu, L. Xunsheng, G. Bing, X. Guangze and Sang Nan, "A Test Scheduling Scheme for Core-Based SOCs Using Genetic Algorithm", International Conference on Embedded Software and Systems Symposia ICESS, 2008.
[29]. Guangyu Liu, "Parallel Elite Genetic Algorithm for Test Scheduling of SOC", Industrial Electronics and Applications (ICIEA), pp 1985-1988, 2014.
[30]. A.Colorni, V. Maniezzo and M. Dorigo, "Ant system: Optimization by a colony of cooperating agents," IEEE Transactions on Systems, Man and Cybernetics, pp 365-368, 1996.
[31]. Jin-Ho Ahn and Sungho Kang, "SOC Test Scheduling Using ACO-based Rectangle Packing," ICIC, pp 655-660, 2006.
[32]. W. Zou, S. R. Reddy, I Pomeranz and Y. Huang, "SOC Test Scheduling Using Simulated Annealing," VTS 2003.
[33]. Y. Huang et al., "Optimal Core Wrapper Width Selection and SOC Test Scheduling Based on 3-D Bin Packing Algorithm," Proc. Int'l Test Conf (ITC'02), pp 74-82, 2002.
[34]. V. Iyenger, K. Chakrabarthy, M.D. Krasniewski and G.N. Kumar, " Design and Optimization of Multi-Level TAM Architectures for hierarchical SOCs," Proc. IEEE VLSI Test Symposium VTS'03, pp 299-304, 2003.
[35]. S.K. Goel, E.J. Marinissen, Anuja Sehgal and K. Chakrabarthy, "Testing and SOCs with hierarchical Cores: Common Fallacies, Test Access Optimization and Test Scheduling," IEEE Transactions on Computers, 58(3), 2009.
[36]. H.M. Harmanani and H.A.. Salamy, "Power-Constrained System-on-a chip Test Scheduling using a Genetic algorithm," Journal of Circuits, Systems and Computers, World Scientific Publishing Company, pp 331-349, 2006.
[37]. Xu Chuan-pei and Dai Kui, "The optimization of Hierarchical SOC Test Architecture to Reduce test time", International Conference on Electronic Packaging Technology and High Density Packaging ICEPT-HDP, pp 1-4, 2008.

[38]. H. Hu and S. Yihe, “A Scalable Test Mechanism and its Optimization for Test Access to Embedded Cores,” Proceedings 4th International Conference on ASIC, pp 773-776, 2001.
[39]. T. Chen and Y. Chang, “SOC Test Scheduling using a B- Tree based floor planning technique”, Proceedings of the Asia and South Pacific Design Automation Conference, pp 1188-1191, 2005.
[40]. C. Giri, D.K.R. Tipparthi and S. Chattopadhyay, “A Genetic Algorithm Approach for System-on-chip test scheduling using Dual Speed TAM with power constraint”, WSEAS Transactions on circuits and systems, pp 416-427,2008.
[41]. K. Ruparel, "SOC test: the devil is in the details of integration/implementation", proceedings IEEE Conference, pp 881-883, 1998.
[42]. Vijay Sheshadri, Vishwani D. Agrawal and Prathima Agrawal , “Power-Aware SOC Test Optimization through Dynamic Voltage and Frequency Scaling”, 21ST IEEE conference on VLSI, pp 102-107, 2012.
[43]. Mostafa Mahi, Omeer Kaan Baykan , Halife Kodaz, “A new hybrid based on article Swarm optimization, Ant colony Optimization and 3-Opt algorithms for Travelling salesman problem”, Applied Soft computing, Elsevier journal, 2015.
[44]. J.Kennedy, R.Eberhart, “Particle swarm optimization”, in IEEE International Conference on Neural Networks Proceedings, pp 1942–1948, 1995.
[45]. Wikipedia, 3-Opt Algorithm, http://en.wikipedia.org/wiki/3-opt.

3-THESIS_CHAPTER_WISE.pdf

ORIGINALITY REPORT

9%	1%	8%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

#	Source	%
1	Mahi, Mostafa, Ömer Kaan Baykan, and Halife Kodaz. "A new hybrid method based on Particle Swarm Optimization, Ant Colony Optimization and 3-Opt algorithms for Traveling Salesman Problem", Applied Soft Computing, 2015. Publication	2%
2	HAIDAR M. HARMANANI. Journal of Circuits Systems and Computers, 2006 Publication	1%
3	Sungho Kang. "RAIN (RAndom Insertion) Scheduling Algorithm for SoC Test", 13th Asian Test Symposium, 2004 Publication	1%
4	Julien Pouget. "Multiple-Constraint Driven System-on-Chip Test Time Optimization", Journal of Electronic Testing, 12/2005 Publication	1%
5	Submitted to SASTRA University Student Paper	<1%
6	Wu, X.. "Test-access mechanism optimization	