

TEST OPTIMIZATION OF CORE BASED SOCs USING VARIOUS HEURISTIC ALGORITHMS

A

Dissertation

Submitted in the partial fulfillment of the requirements

for the award of the degree of

MASTERS OF TECHNOLOGY

in

VLSI DESIGN

Submitted by

Naveen Dewan

Roll No: 601361013

Under the Guidance of :

Ms. Harpreet Vohra

Assistant Professor, ECED



ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004 (INDIA)

JUNE, 2015

CERTIFICATE

I hereby declare that the work which is being presented in the thesis entitled "**TEST OPTIMIZATION OF CORE BASED SOC_s USING VARIOUS 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 for the award of my degree.

Date: 30th June, 2015


Naveen Dewan

Roll No: 601361013

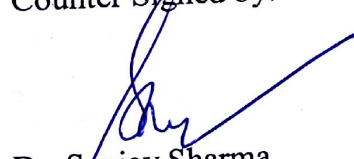
It is certified that the above statement made by the student is correct to the best of my knowledge and belief.


Ms. Harpreet Vohra

Assistant Professor

ECED, Thapar University

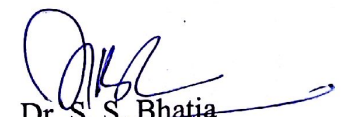
Counter Signed by:-


Dr. Sanjay Sharma

Professor & Head

ECED, Thapar University

Patiala-147004


Dr. S. S. Bhatia

Dean of Academic Affairs

Thapar University

Patiala-147004

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 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.

I thank and own my deepest regards to all of them and all others who have helped me directly or indirectly.

Naveen Dewan

ABSTRACT

The advancement in design methodologies and semiconductor process technologies has led to the development of systems with excessive functionality implemented on a single die, called **system-on-chip**. A set of pre-designed and pre-verified design modules in the form of **soft, hard or firm cores brought from vendors** are integrated into a system using **user-defined logic** (UDL) and interconnects. Complex systems can be implemented having digital, analog and 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 roadmap 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. Testing of SOC is costly due **to large data volume introduced due to increase in the integration** and **interconnection intricacies**, huge power dissipation during test, expensive test generation procedures, heterogeneous mix of cores and their long test application times.

Many techniques have been proposed to reduce the cost by test scheduling, reducing test data volume and optimizing test design mechanisms. Test generation can either be done off-chip by employing ATPG (Automatic test pattern generation) algorithms running on expensive automatic test equipments or **on-chip using a built-in hardware called BIST (Built In Self Test)**. BIST offers the benefits in case if on-chip the availability of TAM is less. However BIST ready cores are not always available, also the multi site testing of SOC for test time reduction makes the ATE more promising. TAM optimization and test scheduling have been the integral part of the research and test optimization for past three decades. Test scheduling is proved to be an NP-hard problem.

This paper proposes a **greedy algorithm based approach and enhanced ant colony optimization algorithm** based approach for test scheduling to reduce the test time subject to test power, bandwidth and hierarchical constraint. We can reduce the problem 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 and they are **d695, p22810, p93791**.

TABLE OF CONTENTS

| | |
|--|-----|
| CERTIFICATE..... | i |
| ACKNOWLEDGEMENT..... | ii |
| ABSTRACT..... | ii |
| i LIST OF FIGURES..... | vii |
| LIST OF TABLES..... | x |
| ABBREVIATIONS..... | xi |
| | |
| CHAPTER 1. INTRODUCTION..... | 1 |
| 1.1 MOTIVATION..... | 1 |
| 1.2 CORE BASED SYSTEM ON CHIP..... | 3 |
| 1.3 SOC TEST PROCESS..... | 5 |
| 1.4 SOC TEST MODEL..... | 6 |
| 1.5 SOC TEST CONSTRAINTS..... | 10 |
| 1.6 THESIS OVERVIEW..... | 12 |
| CHAPTER 2. LITERATURE REVIEW..... | 13 |
| 2.1 TEST TIME MINIMIZATION..... | 13 |
| 2.1.1 CORE TEST TIME MINIMIZATION..... | 14 |
| 2.1.2 CORE ACCESS TIME MINIMIZATION..... | 14 |
| 2.2 TAM OPTIMIZATION..... | 16 |
| 2.2.1 FORK AND MERGE APPROACH..... | 16 |

| | |
|--|----|
| 2.2.2 TEST BUS APPROACH..... | 17 |
| 2.3 TEST SCHEDULING..... | 18 |
| 2.3.1 GENETIC ALGORITHM..... | 18 |
| 2.3.2 ANT COLONY OPTIMIZATION..... | 20 |
| 2.3.3 SIMULATED ANEALING ALGORITHM..... | 20 |
| 2.3.4 RANDOM INSERTION ALGORITHM..... | 21 |
| 2.4 HIERARCHICAL CORE BASED SOC TESTING..... | 21 |
| CHAPTER 3. DEVELOPMENT OF EFFICIENT TEST SCHEDULING TECHNIQUE FOR 2D SOC..... | 23 |
| 3.1 TEST SCHEDULING..... | 23 |
| 3.1.1 PROBLEM FORMULATION..... | 23 |
| 3.1.2 ANALYSIS OF PREVIOUSLY PROPOSED SOLUTIONS..... | 25 |
| 3.1.2.1 ILP ALGORITHM..... | 25 |
| 3.1.2.2 SIMULATED ANEALING..... | 25 |
| 3.1.2.3 GENETIC ALGORITHM..... | 26 |
| 3.1.2.4 ACO ALGORITHM..... | 26 |
| 3.1.3 DEVELOPMENTS IN ALGORITHMS..... | 27 |
| 3.1.3.1 BFD ALGORITHM..... | 27 |
| 3.1.3.2 GREEDY ALGORITHM..... | 30 |
| 3.1.3.3 E-ACO ALGORITHM..... | 37 |

| | |
|---|----|
| CHAPTER 4. SIMULATION RESULTS..... | 42 |
| 4.1 RESULTS OF WRAPPER DESIGN ALGORITHM..... | 42 |
| 4.2 RESULTS OF TEST SCHEDULING ALGORITHM..... | 46 |
| 4.2.1 GREEDY ALGORITHM..... | 46 |
| 4.2.2 E-ACO ALGORITHM..... | 51 |
| CHAPTER 5. CONCLUSION AND FUTURE WORK..... | 54 |
| LIST OF PUBILATIONS..... | 55 |
| REFERENCES..... | 56 |

LIST OF FIGURES

| Figure No. | Title name | Page No. |
|------------|---|----------|
| Figure 1.1 | The number of transistors for Moore's law vs Intel processors [4]. | 1 |
| Figure 1.2 | Design Flow of modular core based design. | 2 |
| Figure 1.3 | SOC Test Process. | 4 |
| Figure 1.4 | System on Chip Hierarchy. | 5 |
| Figure 1.5 | Architecture of Core Based Test | 6 |
| Figure 1.6 | TAM Architecture Designs [4] | 7 |
| Figure 1.7 | Wrapper with parallel TAM interface. | 8 |
| Figure 1.8 | Arrangement of scan chains in a) unwrapped core A b) wrapped core A | 9 |
| Figure 1.9 | Rectangular Bin Packing representation of Test Scheduling. | 10 |
| Figure 2.1 | Wrapper scan chain arrangement in a core. | 15 |
| Figure 2.2 | Fork and merge technique. | 17 |
| Figure 2.3 | Test buses in double layer SOC. | 17 |
| Figure 2.4 | 2D Rectangle Bin packing representation. | 18 |

| | |
|--|----|
| Figure 2.5 3D Rectangle Bin Representation for Scheduling. | 18 |
| Figure 2.6 Simple genetic algorithm. | 19 |
| Figure 2.7 Ant colony optimization. | 20 |
| Figure 3.1 Arrangement if element of parent core. | 29 |
| Figure 3.2 Representation of a schedule in rectangle bin packing consists of eight cores. | 30 |
| Figure 3.3 Flow Chart Diagram of Greedy Algorithm | 34 |
| Figure 3.4 Flow Chart Diagram of E-ACO. | 40 |
| Figure 4.1 a) Wrapper design input file of d695. | 41 |
| b) Wrapper design input file of p22810. | 42 |
| c) Wrapper design input file of p93791. | 42 |
| Figure 4.2 a) Wrapper design output file of d695. | 43 |
| b) Wrapper design output file of p22810. | 44 |
| c) Wrapper design output file of p93791. | 44 |
| Figure 4.3 Bar Graph plot for core 12 of p93791. Figure | 45 |
| 4.4 Input data of d695 with power constraint. Figure 4.5 | 46 |
| Input data of p22810 with power constraint | 46 |
| Figure 4.6 Graphical representation of test time with varying TAM width for d695. | 48 |
| Figure 4.7 Graphical representation of test time with varying TAM width for p22810. | 49 |

| | | |
|-------------|---|----|
| Figure 4.8 | Graphical representation of test time with varying TAM width for p93791. | 49 |
| Figure 4.9 | Graphical representation of test time with varying TAM width for p22810. | 50 |
| Figure 4.10 | Graphical representation of test time with varying TAM width for p93791. | 51 |

LIST OF TABLES

| Table No. | Table Title | Page No. |
|-----------|---|----------|
| Table 4.1 | Test time under power constraint for p93791. | 47 |
| Table 4.2 | Test time under power constraint for p22810. | 47 |
| Table 4.3 | Test time with varying TAM width for hard cores. | 47 |
| Table 4.4 | Greedy Algorithm results for p22810 and p93791. | 48 |
| Table 4.5 | Simulation results for E-ACO and compared with [35]. | 50 |
| Table 4.6 | Test time for d695 using E-ACO under power constraint. | 51 |
| Table 4.7 | Test time for p22810 using E-ACO under power constraint. | 51 |
| Table 4.8 | Test time for p93791 using E-ACO under power constraint. | 52 |
| Table 4.9 | Test time using E-ACO compared with [35] under hierarchical constraint. | 52 |

ABBREVIATIONS

| | |
|-------|----------------------------------|
| IC | Integrated Circuit |
| SOC | System On Chip |
| IP | Intellectual Property |
| DFT | Design For Testability |
| ATE | Automatic Test Equipment |
| BIST | Built In Self Test |
| TAM | Test Access Mechanism |
| HDL | Hardware Description Language |
| MA | Multiplexing Architecture |
| DA | Distribution Architecture |
| ATPG | Automatic Test Pattern Generator |
| GA | Genetic Algorithm |
| ACO | Ant Colony Optimization |
| BFD | Best Fit Decreasing |
| E-ACO | Enhanced Ant Colony Optimization |
| ITC | International Test Conference |
| TSV | Through Silicon Via |

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

The advancements in IC design and semiconductor fabrication technology has led to the creation of micro-electronic products with complex functionality on a single die. Improvements in fabrication technology have led to 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. Usually, such devices are single chip solutions that contain as many cores, memories and functional elements as possible to fit in a single die to get the best price/performance ratio as permitted by the latest technology [1]. The number of transistors/chip has increased enormously over the years. According to Moore’s law, the number of transistors/chip should roughly double every eighteen months [2]. Figure 1.1 shows the Moore’s law versus the actual no. of transistors/chip for some Intel processors [3].

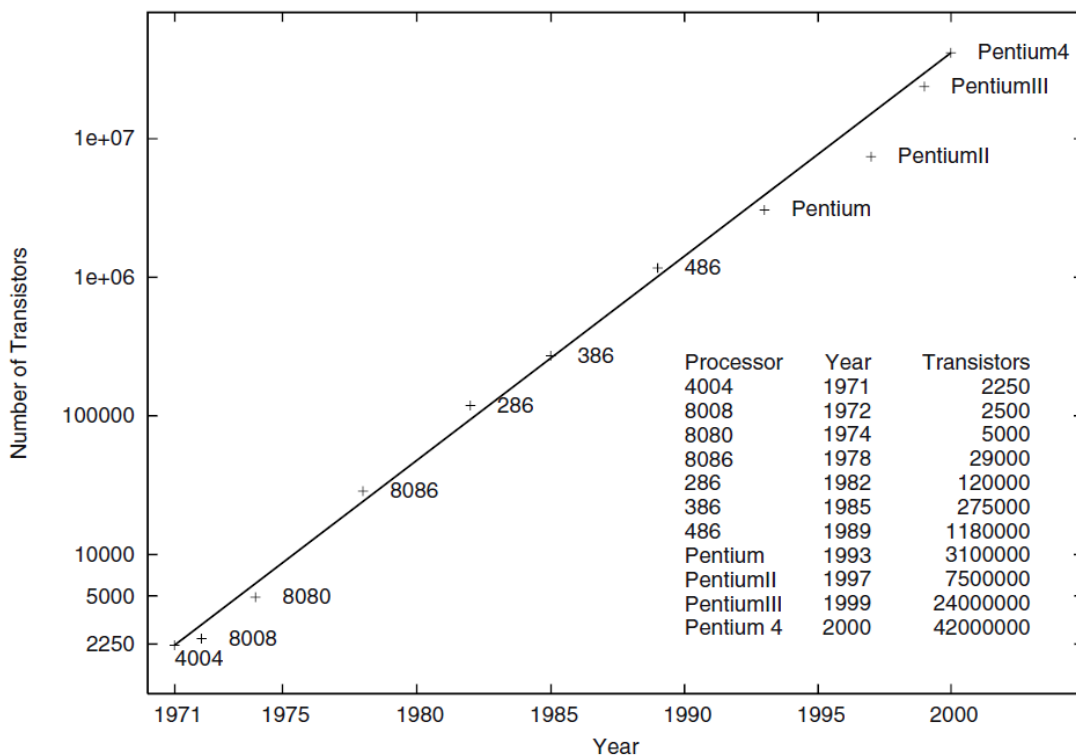


Figure 1.1 The number of transistors for Moore’s law versus some Intel processors [4].

SOC designing is a challenge due to complexity of ICs and the fast time-to-market requirements. Therefore, modular core-based design approach is used. In this approach SOC reuses the pre designed and pre verified intellectual property (IP) cores to shorten the design cycle and reduce the test costs. IP cores are reusable unit of logic, cell or chip layout design owed by one party or can be licensed to other parties for use. System integrator integrates all cores into a system and these cores can be designed in-house or bought from core vendors. Each fabricated IC needs to be tested for defects. The basic test approach is to apply the test stimuli and compare the output response with the expected response. Design for testability (DFT) approach has been proposed so that the complex ICs can be easily tested due to increasing complexity in the test process.

In modular core-based design, each core can be tested as an individual unit. The cores are assumed to be accessible with their test data and it's the job of system integrator to place the cores on the die. Cores are deeply embedded inside the SOC. There is no direct access from the SOC pins to the core. So, there is a need of test architecture design to access the cores and test resources. One of the major research challenges for modular test approach is test scheduling. Design flow of modular core based design is shown in Figure 1.2.

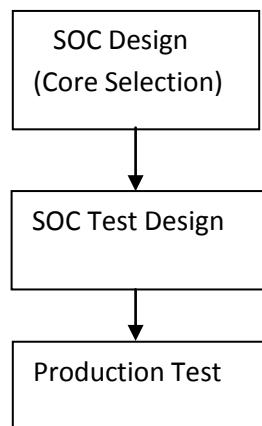


Figure 1.2 Design Flow of modular core based design.

Every electronic product comes to market has to have an affordable price. Therefore, the total manufacturing cost should be reduced as possible. The actual test cost is almost

constant while the fabrication cost has been reduced over the recent years. Soon the cost of testing will surpass the cost of fabrication. Test cost is a significant part of the total manufacturing cost including the test cost and the fabrication cost. Test cost can be reduced by using proper test planning which includes:

- Test Architecture Design.
- Test Scheduling.
- Test Data Compression.

1.2 CORE BASED SYSTEM ON CHIP

The cores in a SOC are obtained from core vendors. Different cores can be of different origin and can be classified into three categories:

- Soft cores.
- Firm cores.
- Hard cores.

Soft Cores: Soft cores are synthesizable architectural modules that have highest modification flexibility. Soft cores are very unpredictable in performance and there are a lot of physical design issues need to be faces. Soft core consists of a technology independent Hardware Description Language (HDL) files, test bench and have necessary information. Soft core presents difficulties for floor planning, placement and routing.

Firm Cores: Firm cores are ready for routing purposes and do not present any challenges for floor planning, placement and routing. Firm cores consist of RTL code and technology dependent net list. The performance of these cores is also unpredictable.

Hard Cores: Hard Cores contain physical layout information and have a predictable performance. These are technology dependent modules and are highly optimized for area. The only disadvantage is that hard cores cannot be customized for a particular design process. Hard cores cannot be easily modified like soft cores.

The SOC consists of number cores that can be on the same level (no hierarchy) or at different levels (hierarchical). Cores can be embedded inside other cores. A parent core may have cores embedded inside it and a child core can also be a parent core with

embedded child cores. These cores are known as hierarchical cores. System on Chip Hierarchy is illustrated in Figure 1.4.

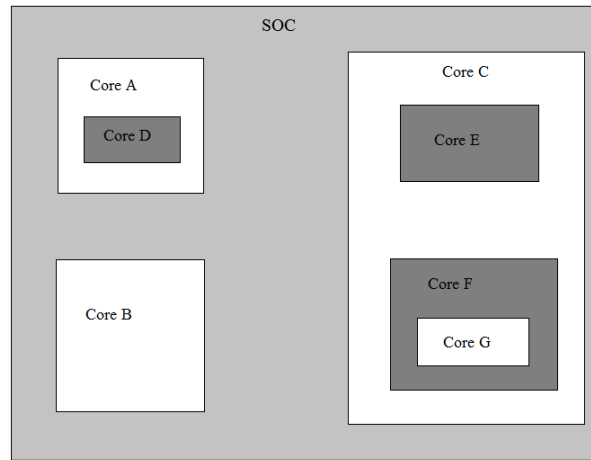


Figure 1.4 System on Chip Hierarchy.

In a core based SOC, embedded cores represent intellectual property and core vendors are reluctant to expose the structural information about their cores to the users. So, users do not get the access of the core net lists. Instead, core vendors provide a set of test patterns that guarantees the maximum fault coverage. Then the test patterns are applied to the cores in a specific clocking strategy. Cores are often embedded in different layers so a dedicated data transport system is required. Through that system test stimuli is applied to embedded core from the source and the test response is collected at the sink.

The architecture of core based test approach is shown in Figure 1.5. It consists of three elements:

1. Test pattern source and sink: The test stimuli are generated through test pattern source for the cores and the test response is compared by the test pattern sink with the expected response.
2. Test Access Mechanism (TAM): TAM is used to transport the test patterns from test pattern source to the core under test and test responses from core under test to the test pattern sink. TAM optimization can be done to increase performance.

3. Core Wrapper: Core Wrapper is a thin shell around the core. It acts as an interface between the **core under test and its environment**. Together with TAM, core wrapper forms a test access infrastructure for embedded reusable cores.

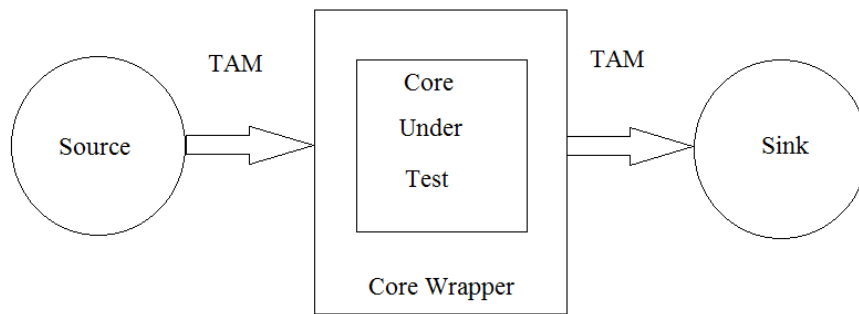


Figure 1.5 Architecture of Core Based Test.

1.3 SOC TEST PROCESS

Each individual core is surrounded by **testing components** and may imply certain isolation modes like **safe mode, low power mode and bypass mode**. SOC test comprised of individual testing of cores and the testing of interconnects. In SOC testing, the test stimuli comprised of **test vectors is applied to the chip** and the response of the test stimuli is compared with the expected response as shown in Figure 1.3. Test Stimuli application to the chip is done using two types of approaches:

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

Automatic Test Equipment: A special machine is used to apply the **test stimuli to the chip called Automatic Test Equipment (ATE)**. ATE also compares the output responses from the chip with the expected responses. Therefore, if there is a difference between both **responses than faults are detected**. In today's technology, there are complex and high speed SOC's but there are limited numbers of pins in ATE through which millions of transistors are needed to be tested. So, in order to reduce the burden over **ATE we can reduce the test time using scheduling algorithms** or test data volume.

Built in Self Test: In this approach, the core can be tested using the on-chip engine which generates the test stimuli and check responses. It simplifies the test task for the system integrator but there are some limitations. Most cores cannot be tested using BIST because they contain logic that cannot be tested using randomly generated tests. So, pre-computed

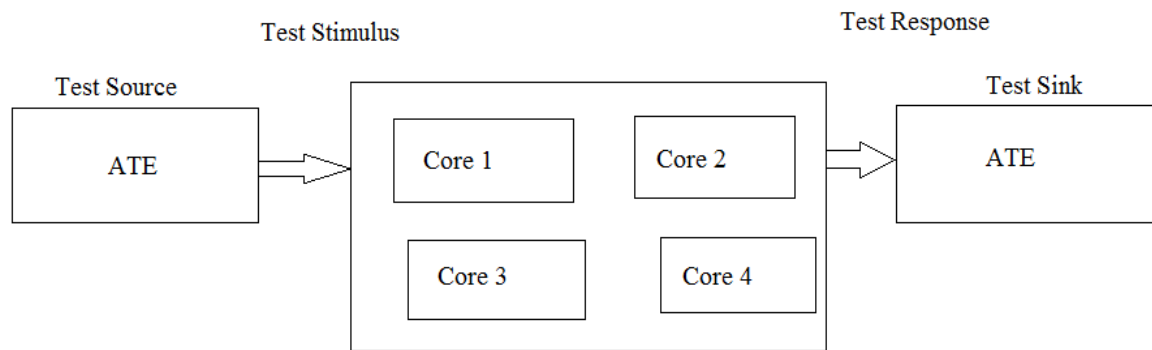


Figure 1.3 SOC Test Process.

test sets can be applied to the cores that can be obtained from the core vendors. The core vendors have no idea whether the core is placed on the chip or not. They assume the core terminals have the direct access to the test data. It is the task of the system integrator that the logic surrounding the core allows the tests to be applied, i.e. designing a suitable test access mechanism (TAM).

1.4 SOC TEST MODEL

The IC design technology has made it possible to create extremely complex ICs. The test cost of these ICs is increasing and it's important to develop techniques that can reduce the test cost. The test cost is dependent on test application time and test time for SOC's are 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 combined with user defined logic. SOC test model consists of three main parts:

- Test Access Mechanism Design.
- Wrapper Design.
- Test Scheduling.

Test Access Mechanism Design: TAM Design is responsible for the transportation of the test data in the SOC. It connects the test sources and test sinks to the cores. The TAM can be optimized for the test purpose or existing structure can be used. In recent years, different architectures has been proposed for TAM design like Multiplexing Architecture [5], Daisy chain Architecture [5], Test Bus Architecture [6] and Flexible-Width Architecture [7]. In Multiplexing architecture, every core is assigned to all the TAM wired and the cores are tested one at a time. So, all the cores have to be tested in sequence. In Daisy chain architecture, a bypass structure is added to shorten the access path for each core. Bypass register and 2x1 multiplexer allows full access to each core. Test Bus is combined form of both Multiplexing and Distribution Architecture. Several Test Buses are created from TAM width. All cores are tested in a sequence in each Test Bus and concurrent testing can be achieved if there are several Test Buses active at a point of time. Figure 1.6 shows the TAM architectures [4] proposed in recent papers.

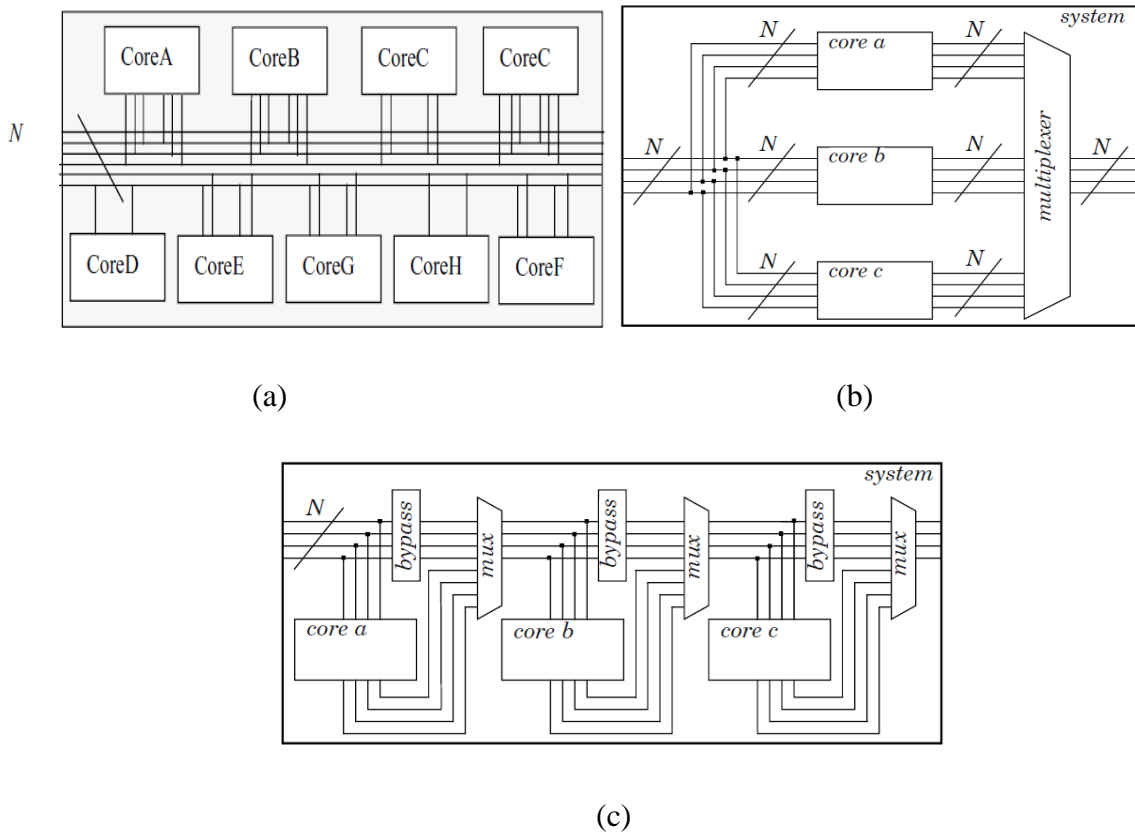


Figure 1.6. TAM Architecture Designs [4] (a) Test Bus Architecture (b) Multiplexing Architecture (c) Daisy Chain Architecture.

Wrapper Design: A core wrapper acts as an interface between the core terminals and the TAM wires. It is also called the **Test shell around the core**. A standard IEEE P1500 Wrapper is proposed in [4] which solves the various aspects of core based testing like transferring the information about the cores to the core vendors, accesses the ATE or BIST engine to the core under test and optimizes with respect to the multiple conflicts. Figure 1.7 shows the IEEE P1500 Wrapper with parallel TAM interface.

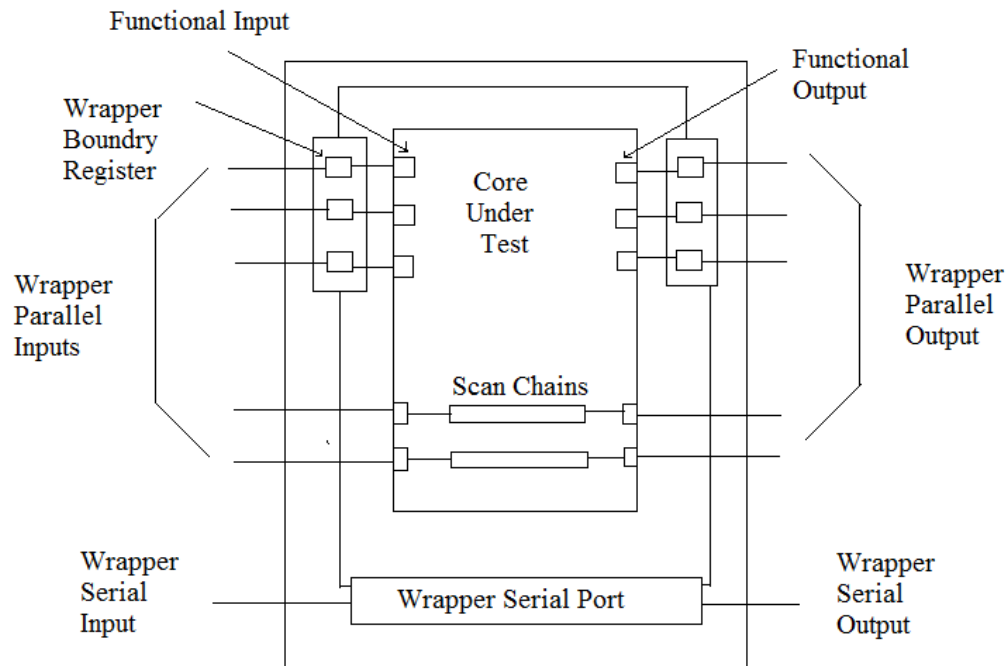


Figure 1.7 Wrapper with Parallel TAM interface.

Core Wrapper is used to facilitate isolation to the core from the surroundings. In case of testing, core wrapper runs in two modes: **internal test mode (INTEST)** and external test mode (EXTEST). In INTEST **mode the core is tested** while in EXTEST mode the interconnection between **the two wrappers is tested**. In an SOC there can be wrapped and unwrapped cores. Wrapped core has to be in INTEST mode during testing and to test unwrapped core the core **has to be in EXTEST mode**. The problem during scheduling is that the core can be in INTEST or EXTEST mode at a time.

Wrapper combines the **internal scan chains** and **the functional input/outputs of the core under test**. Wrapper Design Algorithms has been proposed like **Best fit Decreasing** [8] in

order to arrange the internal elements of the core and calculate the test time of wrapper core. Suppose there is core A with 5 scan chains. The length of the scan chain is 10,8,5,3 and 2 respectively. Figure 1.8(a) shows the unwrapped core A and Figure 1.8(b) shows the wrapped core A.

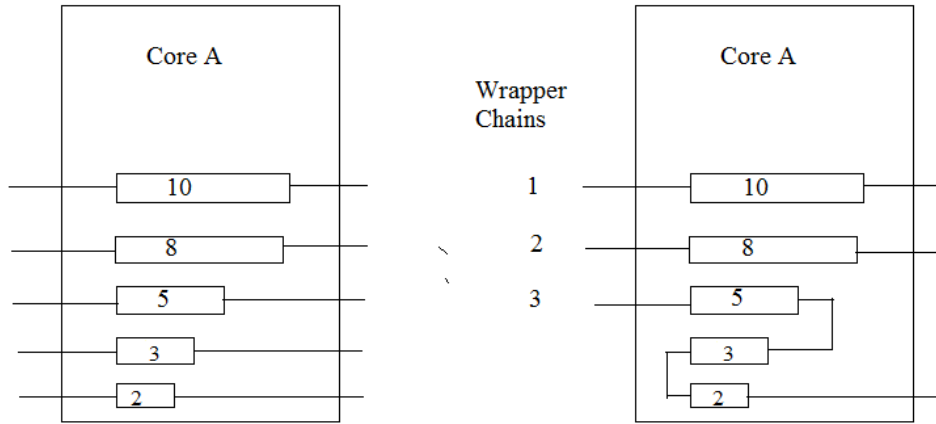


Figure 1.8(a)

Figure 1.8(b)

Figure 8. Arrangement of scan chains in (a) unwrapped core A (b) wrapped core A.

Test Scheduling: After determining the type of TAM to be used and wrapper design test integrator faces the problem of test scheduling. Test scheduling is to determine the order in which the test data is applied so that all cores can be tested in minimum time. Test scheduling is an NP Hard Problem. Test scheduling is done such that some constraints like power consumption and hardware overhead is not violated. Different trade-offs are need to be considered during test scheduling. In order to reduce the test time, concurrent testing of cores is done. The test scheduling of cores can be represented in a rectangular bin packing [9] as shown in Figure 1.9. Various Scheduling algorithms have been proposed like genetic algorithms [11], random insertion algorithm [10] and ant colony optimization algorithm [12] etc.

There are three basic scheduling techniques:

1. **Non partitioned testing**, in which no new core is allowed to test until all cores in that session are tested.
2. Partitioned testing, in which a core is scheduled as soon as possible.
3. Preemptive testing, in which core test can be interrupted and resumed after some time.

The only requirement is that all the cores must be completed at the end of the testing.

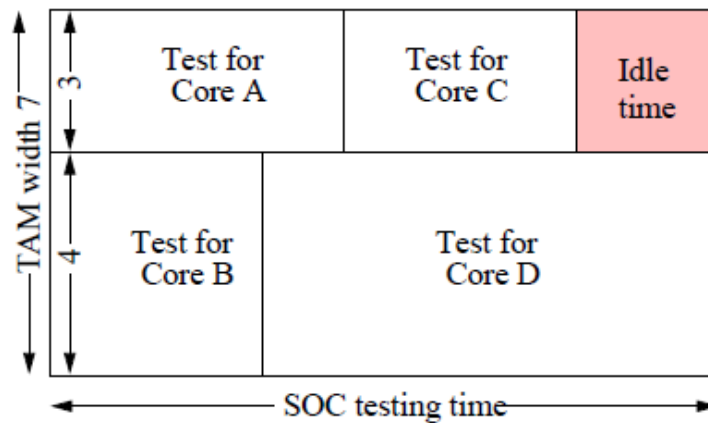


Figure 1.9. Rectangular Bin Packing representation of Test Scheduling[9].

1.5 SOC TEST CONSTRAINTS

In test scheduling, large test data volume leads to the large test application time. In order to reduce the test application time concurrent testing of cores is done in modular core based SOCs. But in case of concurrent testing there are some constraints that must be considered. These constraints can be described [8] as follows:

a).Power Constraint

Power dissipation has become a big challenge for deep submicron integrated circuits. By adding more functionality to the ICs there is a **risk on thermal stability of ICs which affects the performance and reliability**. Power consumption in VLSI circuits is dynamic power dissipation and static power dissipation. Dynamic power dissipation occurs due to

the switching activities and the static power dissipation is due to the leakage currents. Dynamic power dissipation dominates the chip power consumption in VLSI SOC. Switching activities during testing differs from that during the normal operation. Sometimes, power dissipation while testing is higher and it has to be respected so that the chip is not damaged. In case of SOC testing, during test scheduling the total power consumption at a point of time is fixed. So that the total power dissipation do not exceed the limit and the IC is safe from thermal burn out.

b). TAM assignment

The test data is applied to the cores through the TAM wires. However, TAM wires are limited but there are large number of inputs, outputs and bidirectional pins in SOC. All cores cannot be tested at the same time. Each core must be assigned TAM wires for testing. So, TAM assignment is a serious constraint in which full TAM bandwidth utilization is required.

c). Precedence Constraint

In SOC testing, sometimes there is a particular order in which cores must be tested. Therefore, scheduling should be done such that that order can be maintained.

d). Hierarchical Constraint

In a core based SOC, there are cores embedded inside the system. A core can also be embedded inside the core. A child core can also be embedded in a parent core. This embedding of cores is a big conflict in case of testing. Sometimes, testing of child core is done prior to the testing of parent core using the same TAM wires.

e). Volume of test data

The test data volume depends on the chip complexity. As the complexity of the chip grows and there are more IP cores introduced in SOC the volume of test data is increased. This also increases the ATE cost. Therefore, a cost effective approach is used like data compression and compaction. Data compression reduces the number of bits that needs to

be stores for each pattern and data compaction decreases the number of test patterns in the test set.

1.6 THESIS OVERVIEW

This chapter defines the system on chip design model and some preliminary issues related to testing of core based SOC's. The further organization of the thesis is as follows:

- Chapter 2 presents the previous work in core based SOC testing and gaps in these work that have led to the idea behind pursuing my thesis work.
- Chapter 3 presents the analysis of the previous proposed solutions and developments in the algorithms.
- 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

2.1 TEST TIME MINIMIZATION

Test time minimization can be a major problem in developing a SOC test solution. The large test data volumes and high design complexity in core based SOC's cause long test times. Test time analysis is done in [13] and [14] by Larsson and Fujiwara in which test time is calculated using multiplexing architecture (MA) and distribution architecture (DA). In multiplexing design all TAM bandwidth is given to each core which means the testing is done in sequence and in distribution design all TAM bandwidth is distributed to every core which means each core has some dedicated part of the TAM bandwidth. The results show that the MA is not efficient when TAM width increases and DA is less efficient when TAM width decreases.

In [14] M. Sugihara et al. described two test methodologies BIST and ATPG for test time reduction. This paper discusses the preemptive test scheduling method for SOC's to reduce test time. The main contributions of this paper are:

- a) BIST is combined with ATPG to relax the limitation of external primary inputs and primary outputs.
- b) External test of one core and BIST for other core can be combined and tested in parallel to reduce the test time.
- c) Optimal test vector sets are selected from given set of test vectors for each core.

The main reasons of increase in test time:

- a) The increase in number of transistors on a chip.
- b) The test vectors cannot be applied directly to the core because the core based system has logic like isolation rings around the cores.
- c) The test method varies from core to core.

Test time minimization problem can be classified into two problems. Core test time minimization and core access time minimization.

2.1.1 CORE TEST TIME MINIMIZATION

Core test time minimization is related to the test scheduling problem of cores. Scheduling algorithms has been introduced in [8][35][9][10][11] with wrapper and TAM optimization[20][23]. Both TAM/wrapper optimization can be integrated with test scheduling algorithms as done in [9][10][11]. The core based system is tested always using these possible ways:

- a) Internal testing through BIST.
- b) External testing through ATE.
- c) Combination of BIST and ATE.

If BIST methodology is used, core has a dedicated logic which provides the BIST pattern. In external testing, the test patterns are transported using the test buses to the multiple cores. In order to minimize the test time, core selection should be done carefully and the cores should be scheduled optimally.

2.1.2 CORE ACCESS TIME MINIMIZATION

The accessibility of the cores from the I/O terminals of the system is a major difficulty. There are different approaches to deliver test patterns to the core. One approach is to have a extra test circuit around the core so as to isolate the core under test. For example, a multiplex is used at each core input so that input can be directly accessible from system input. An isolation ring can be inserted so that cores can be accessed serially. The Second approach uses a bypass mode for cores to reduce the problem to one of finding paths from the system inputs to the core inputs and from the core inputs to the system outputs. The last approach is based on test architectures by which cores are isolated from each other using a dedicated bus of a test rail around the cores to transfer test data.

Test time for core i depends on the width of the test bus w to which it is assigned. The test time of the core is related to the length of the longest internal scan chain of each core. As the number of TAMs increase the longest scan chain value reduces. The internal scan chains are arranged using the wrapper. Test time calculation is done as equation given in [20] and [8]. The test time required to apply test set to the core is given by equation 1:

$$T = (1 + \max\{si, so\}) \cdot p + \min\{si, so\} \quad 2.1$$

Where T represents the test time of the core i , p is the number of test patterns, si is the input scan chain and so is the output scan chain. This test time can be reduced if si and so values are reduced. In [8] authors have presented two heuristic algorithms to solve scan chain partitioning algorithm. For example, Consider a core has four internal scan chains having length 32, 8, 8 and 8 with 4 functional inputs and 2 functional outputs. The scan elements can be assigned to only two wrapper scan chains and $\max\{si, so\}$ can be calculated as 32. The wrapper scan chain arrangement is shown in Figure 2.1. The best fit algorithm is used in [8] which have three main parts:

- Partitioning the internal scan chains into a minimum number of wrapper scan chains in order to minimize the longest wrapper scan chain length. The internal scan chains are sorted in decreasing order to the wrapper scan chains. The next scan chain is then assigned to the wrapper scan chain whose length after assignment is closest to.
- Assigning the functional inputs to the wrapper scan chains arranged in part (i).
- Assign the functional outputs to the wrapper scan chains arranged in part (i).

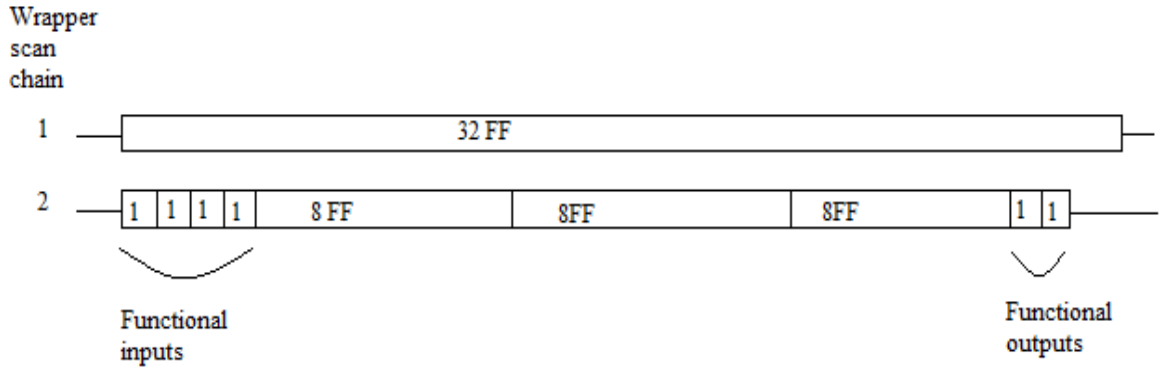


Figure 2.1 Wrapper scan chain arrangement in a core.

In order to calculate the overall test time after scheduling of cores the total test application time is given as equation 2:

$$total\ test\ time = \min(\max_{i=1}^n \sum_{j=1}^{maxtm} t_{ij}x_{ij}) \quad 2.2$$

Where t_{ij} = test time for the core i and tam width j and $x_{ij} = 0$ or 1 which verifies that the core with that tam width is scheduled or not.

2.2 TAM OPTIMIZATION

Test time of the core based SOC can be reduced if Test Access Architecture is efficient. The TAM Optimization problem is considered to be a NP-Complete problem as proposed in [21]. Wrapper Design and wrapper design algorithm was developed in [22] and [23]. Integer Linear Programming model was introduced in [24] known as test bus assignment problem by combining distribution of test buses among individual test buses of cores. Assignment of core buses and determining the TAM width required these are the choices that designers have to make. A new 2D rectangle packing problem as shown in Figure 2.4 for wrapper and TAM optimization based on ILP modeling with fork and merge with the use of pareto-optimal points was proposed in [25],[26] and [27]. Later in [33] a core clustering technique for testing cores concurrently was introduced as 3-D rectangle bin packing problem simultaneously modeling the optimization of test time, test bus width and power dissipation shown in Figure 2.5. There are two basic approaches TAM utilization (i) Fork and merge approach and (ii) Test Bus approach.

2.2.1 FORK AND MERGE APPROACH

Fork and merge technique is a technique in which test buses can fork and merge between cores. Core tests are represented as rectangular bins with their height represents TAM width and length represents test time as shown in Figure 2.3. The fork and merge technique is useful to reduce the test time. Advantage of fork and merge over test bus approach is there is less probability of idle time. The wastage of memory will be less. In [35], fork and merge technique is used for test scheduling. Test scheduling combined with TAM optimization and considering power constraint. This technique can be understood with Figure 2.2.

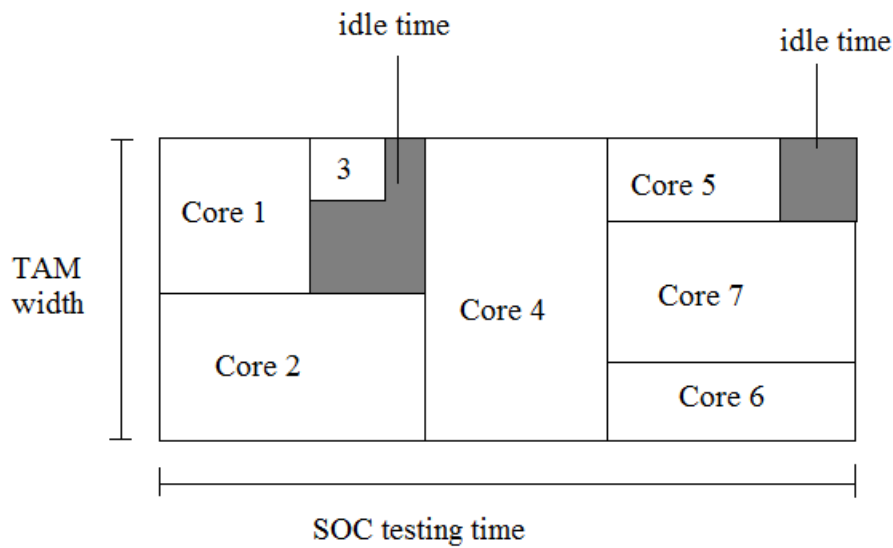


Figure 2.2 Fork and merge technique.

2.2.2 TEST BUS APPROACH

In this technique, total TAM width is partitioned into test Buses with different widths. While scheduling, different cores are assigned different Test Buses according to their TAM widths. If the TAMs are of the same width the core is assigned to the TAM having less test time. The core assignment in test buses is required. This approach is highly helpful in multi level SOC's. As each level can be assigned a particular test bus and the core at that level can only be tested through that test bus. The example of this approach is shown in Figure 2.3.

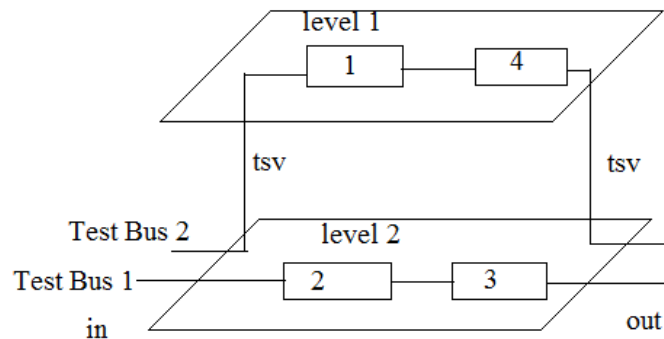


Figure 2.3 Test buses in double layer SOC.

2.3 TEST SCHEDULING.

Test scheduling is **NP – Hard problem as discussed** in [4] and can be solved through combinatorial optimization algorithms using wrapper design and TAM optimization algorithms in [18] and [25]. Optimization algorithms are discussed below:

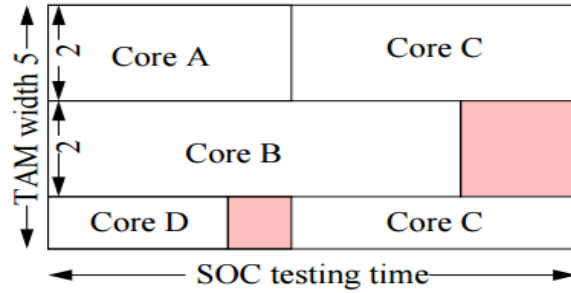


Figure 2.4 2-D Rectangle bin packing representation.

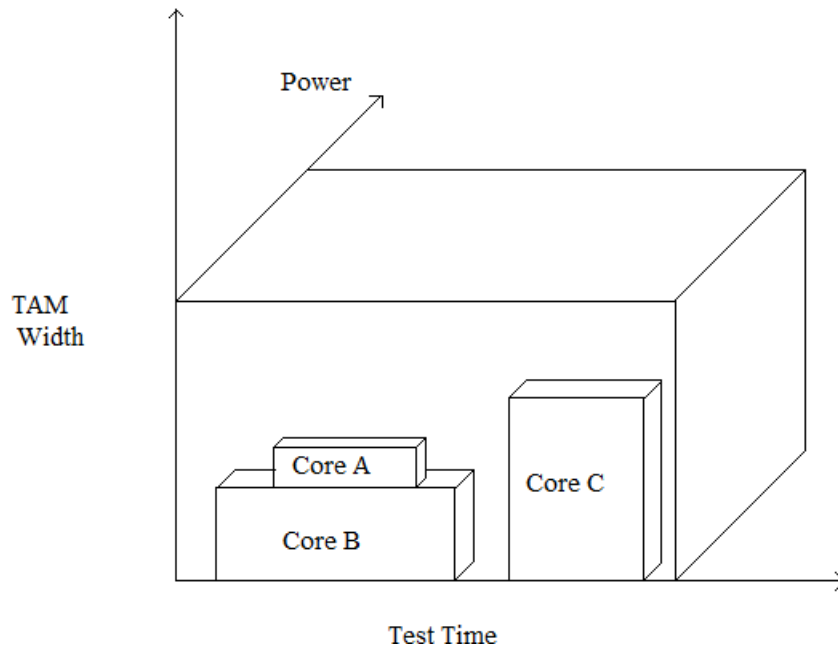


Figure 2.5 3-D Rectangle Bin Representation for Scheduling.

2.3.1 GENETIC ALGORITHM

Genetic Algorithms are adaptive heuristic search algorithm based on evolutionary ideas of natural selection and genetics. These are a part of evolutionary computing, a rapidly growing area of artificial intelligence. GA's are inspired by Darwin's theory about

evolution – “survival of the fittest”. GA represent random search used to solve optimization problems. In [28] a test scheduling scheme for core-based SOC's based on genetic algorithms is presented. A set of tasks is given (test sets for the cores), a set of test resources i.e. test buses and dedicated test access architecture. Test scheduling is done such that total test application time can be minimized. The solution is more optimized as differential evolution and self-adaptive mutation are brought into the traditional genetic algorithm. A simple genetic algorithm uses non-overlapping populations; the entire population is replaced each generation. Individual solutions are represented by shaded ovals as shown in Figure 2.4. In this case, darker shading represents a better solution.

In [29] parallel elite genetic algorithm is introduced to reduce the test application time under peak power constraint. In this offspring are generated using parent A and Parent B and then crossover and mutation is done. This algorithm is tested in 2D SOC benchmark circuits and it has proved to be one of the effective scheduling algorithms. In [36] an power constrained efficient approach for the test scheduling problem of core-based systems based on genetic algorithm is proposed. This method minimizes the test application time through compact test schedule. In genetic core test scheduling formulation there is chromosomal representation, selection and reproduction, genetic operators (mutation, crossover and fill gap). During each generation, chromosomes are elected to reproduce, as a result new test schedules are formed called offsprings.

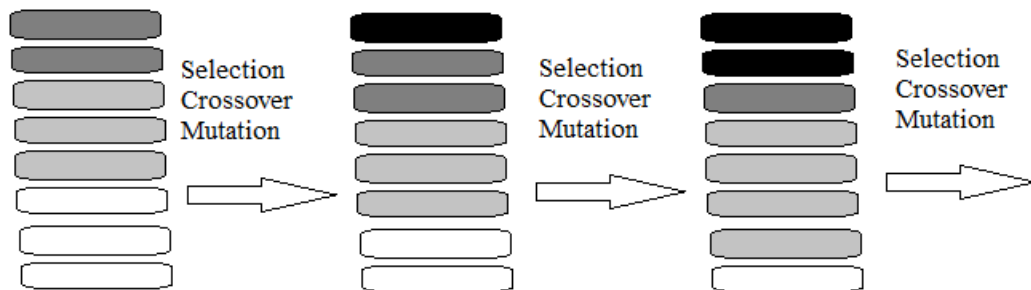


Figure 2.6 Simple genetic algorithm.

2.3.2 ANT COLONY OPTIMIZATION

Ant colony optimization is based on real ant colony which is proposed in [30]. This is about the behavior of the ants, how they find the shortest path from their nest to the food. While searching for food, ants excrete a hormone called pheromone. Ants smell that pheromone and choose their ways that have high pheromone concentrations. During this process ants may find a shortest path due to pheromone trails left by ants from their nest to the food and back as shown in Figure 2.5. Ant colony Optimization has proved to be very effective in solving many NP problems. ACO based test scheduling increases the probability of finding optimized solutions in short time. In [31] ACO scheduling algorithm is proposed. Test scheduling problem is formulated as a rectangle bin packing problem and uses ACO to cover more solution space to increase the probability of finding optimal solutions.

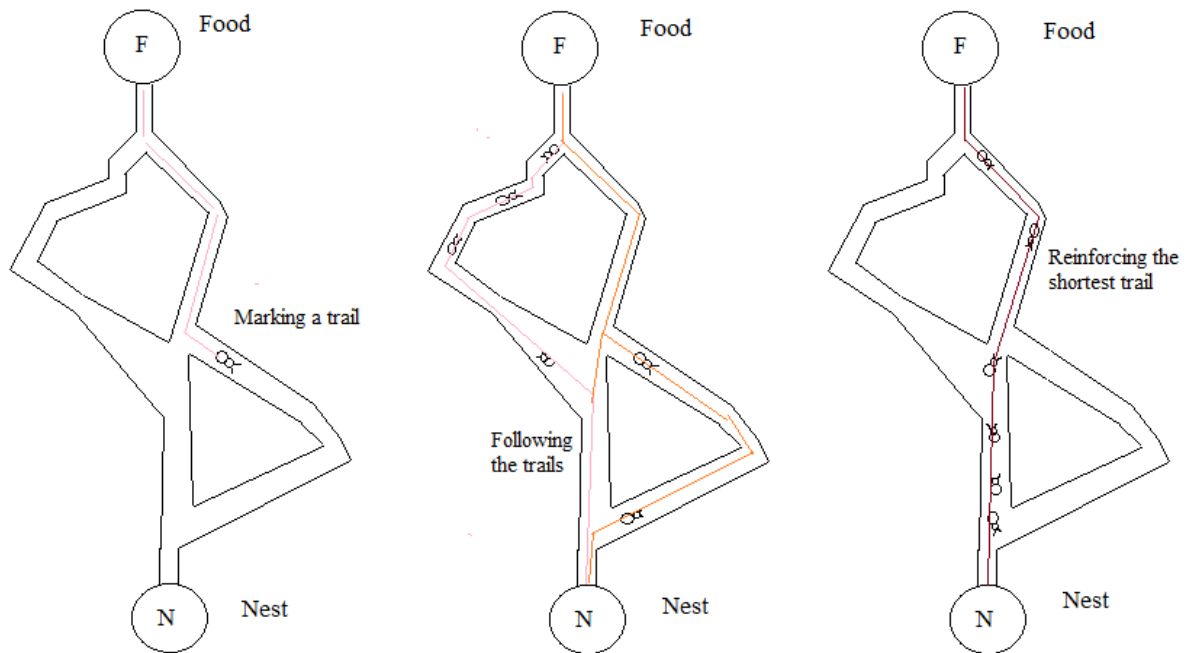


Figure 2.7 Ant colony optimization.

2.3.3 SIMULATED ANNEALING ALGORITHM

Simulated annealing algorithm can solve 2D bin packing problem when we optimize the wrapper design and TAM design. The best configuration among every core is selected by

SA algorithm in order to minimize the test application time. In this while scheduling a data structure called sequence pair is used for floor planning. SA based procedure is proposed in [32]. In this paper optimal test schedule is found by altering an initial sequence pair and changing the width of the core wrapper. A method of wrapper design for cores without internal scan chains is also proposed in this paper.

2.3.4 RANDOM INSERTION ALGORITHM

Cores are represented as rectangles in [16] and [26]. The height is the TAM width assigned and the width is the test application time. To schedule these cores test application time is used. In [32] SOC test scheduling, sequence pair representation is proposed. The relative position of rectangles is not changed by inserting a new element into arbitrary positions of sequence pair. So, it is possible to schedule a new core without breaking down already obtained well scheduled results. The significance of this is only two permutations are sufficient to represent the placement of rectangles. The disadvantage is that as the number of cores increases, the total number of possible sequence pair increases rapidly.

2.4 HIERARCHICAL CORE BASED SOC TESTING

Hierarchical cores can be considered at the same level in test mode and TAM assignment for cores embedded inside a hierarchical core is not limited by design hierarchy. The test scheduling, wrapper design and TAM assignment done for flat cores are not valid for Hierarchical core SOC. The hierarchy constraint is in the manner that the test must be applied to the parent and child core simultaneously. A multilevel TAM architecture is proposed in [34] which shows how flat SOC techniques can be used for multi level TAM optimization in hierarchical core SOC using ILP formulation for flexible bus width.

In [35], test optimization is done for hierarchical cores in which two approaches for efficient testing of SOC with hierarchical cores is proposed. In the first approach, the problem is solved by extending a conventional wrapper design and this approach leaves full flexibility for TAM optimization and test scheduling. The second approach has to do

with improved wrapper design for parent cores that work in two disjoint modes for testing of parent and child cores. In [37] a hierarchical SOC test architecture technique is presented. This model simplifies the multilevel TAM optimization. It applies the evolutionary algorithm to SOC test architecture models.

DEVELOPMENT OF EFFICIENT TEST SCHEDULING TECHNIQUE FOR 2D SOC

3.1 TEST SCHEDULING

A major challenge in testing core based SOC is test scheduling which determines the order in which various cores need to be tested. Test scheduling of cores even for a simple SOC is equivalent to NP-hard problem. An efficient test scheduling approach must minimize the test time while addressing resource constraints among cores due to the use of TAMs, BIST engines and power dissipation constraint. The minimum test time would be achieved by maximizing the test of all individual cores; however, designing constraints may prevent this parallelism. For example, power dissipation is an important factor that may impact the test parallelism. Power dissipation is a function of time and depends on switching activity from application of test vectors to the system. SOC in test mode can dissipate up to twice the amount of power they do in normal mode, since cores that do not normally operate in parallel may be tested concurrently to minimize testing time. Therefore, power constrained test scheduling is needed in order to limit the amount of concurrency while testing so that maximum power dissipation of SOC is not exceeded. There has been many approached for the test scheduling problem in core based SOC. In this chapter, an efficient approach to SOC test scheduling problem based on time, power and hierarchical constraint is presented.

3.1.1 PROBLEM FORMULATION

1. P_T : Determine (i) an assignment of cores to given TAM widths and power. (ii) test optimization of schedule such that SOC test time can be minimized.

Consider a SOC consisting of N cores and W TAMs. If TAM bus architecture is used then W TAMs have widths $w_1, w_2, w_3, \dots, w_W$ or if TAM fork and merge technique is used then B TAMs act as individual wires of width 1. Let core i is selected for scheduling to TAM j , the test time to test core i can be given as $T_i(w_j)$ clock cycles. The test time of the core is calculated as:

$$T_i(w_j) = (1 + \max(s_i, s_o)) \cdot tp_i + \min(s_i, s_o). \quad 3.1$$

Where s_i = input scan chain length calculated using wrapper design.

s_o = output scan chain length calculated using wrapper design.

tp_i = test patterns for core i .

let x_{ij} be a variable with value 0 or 1 (where $1 \leq i \leq N$ and $1 \leq j \leq W$). It can be defined as follows:

$$x_{ij} = \begin{cases} 1, & \text{if core } i \text{ is scheduled using TAM } j. \\ 0, & \text{if core is not scheduled.} \end{cases} \quad 3.2$$

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

$$\text{Test time of all cores} = \sum_{i=1}^N T_i(w_j) \cdot x_{ij} \quad 3.3$$

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

$$\text{Total test time} = \max_{1 \leq j \leq W} \left(\sum_{i=1}^N T_i(w_j) \cdot x_{ij} \right) \quad 3.4$$

In case of fork and merge technique total test time can be given as:

$$\text{Total test time} = \max_{1 \leq j \leq B} \left(\sum_{i=1}^N T_i(w_j) \cdot x_{ij} \right) \quad 3.5$$

2. **P_w : Design a wrapper for each core, such that test time of core can be minimized and the TAM width required for the core is reduced.**

A core with n functional inputs, p functional outputs, scan chains sc of length l_1, l_2, \dots, l_{sc} with TAM width w . Arrange the scan chains into less than or equal to w wrapper scan chains and then arrange the functional inputs and functional outputs such that $\max(s_i, s_o)$ is minimized.

3. **P_H : Design a wrapper for hierarchical cores such that test time of the core can be minimized.**

Given a mega core with fixed TAM width w . Arrange the scan chains under this constraint such (i) w is not exceeded at any time. (ii) each mega core receives one of its pre-specified TAM widths and (iii) parent mega cores are tested only after their child cores.

3.1.2 ANALYSIS OF PREVIOUSLY PROPOSED SOLUTIONS

An analysis of previously proposed test scheduling algorithms is made to develop a new and efficient test scheduling algorithm. The objective is to determine the test architecture at different TAM widths and constraints and reducing the overall SOC test time.

3.1.2.1 INTEGER LINEAR PROGRAMMING (ILP) ALGORITHM

The aim of ILP is to optimize a linear objective function with a set of integer variables, with satisfying set of linear constraints. The typical ILP model is described as follows:

Minimize: Ax

Subject to: $Bx \leq C, x \geq 0$,

A is a cost vector, B is a constraint matrix, C is a column vector of constraints and x is a vector of integer variables. In [38], ILP problem is explained as:

Let there are N cores and W TAM width. Determine the assignment of cores to the testline's signal lines such that test time can be minimized. P in the number of test pins.

$$\text{Min } T \quad 3.6$$

$$T > \sum_{i=1}^N t_i \sum_{j=1}^P x_{ijk} \quad 1 \leq k \leq W \quad 3.7$$

Equation 3.6 means that T maximum test time of testline's signal and equation 3.7 means that every test must connect to the signal line.

3.1.2.2 SIMULATED ANEALING

This algorithm starts with an initial solution. A neighboring solution is then generated using perturbation on the current solution. If the test time of the neighboring solution is lower than the initial solution the neighboring solution is accepted. If the cost of initial

solution is lower than neighboring solution is rejected. The probability of accepting an optimized solution is a function of parameter and temperature. The probability can be defined as:

$$P = e^{\frac{-\Delta C}{T}} \quad 3.8$$

ΔC is the change in cost and T is the temperature. As the test time increases the T decreases. At first the temperature is large therefore the probability of finding the solution is high but as the temperature decreases with time the probability of finding an optimized solution decreases. Therefore, an optimized solution can be obtained as in [39].

3.1.2.3 GENETIC ALGORITHM

The genetic algorithm involves the following choices during its formation:

- a. Encoding of the solutions to form chromosomes.
- b. Decide upon a crossover mutation.
- c. Identify a optimized mutation operator.

Chromosome consists of TAM partition part, core assignment part and the test time of that chromosome. Genetic operators used in optimizing the solution are crossover, mutation and fitness measure. Crossover sorts the whole population according to fitness values. Mutation is used bring more effectiveness in the solutions. A genetic algorithm based approach is proposed in [40] for SOC test scheduling using dual speed TAM considering power constraint.

3.1.2.4 ANT COLONY OPTIMIZATION ALGORITHM

ACO algorithm can be implemented for test scheduling. ACO collects the pheromone data with heuristic favorability to get optimized results. Heuristic parameter can be used to reduce premature stagnation so that ACO can be available to all conditions. Stagnation is the process in which ants follow same path and new solution is generated to find better solutions. In [31], ACO algorithm is implemented for wrapper/TAM width selection to schedule the cores. Testing of the core is represented as rectangle bin packing problem.

3.1.3 DEVELOPMENTS IN ALGORITHMS

Based on previous work analysis some developments have been done. Greedy algorithm and enhanced ant colony optimization algorithm is proposed for **test scheduling P_T problem**. The test time calculation of each core is done using best fit decreasing algorithm is also described in this section. Wrapper algorithm is modified for hierarchical cores and scheduling is done using greedy algorithm.

3.1.2.1 BFD ALGORITHM

The problem P_W mentioned above is solved by Best Fit Decreasing Heuristic Algorithm. The algorithm consists of three main parts [8]:

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

This wrapper design algorithm arranges **the scan chains at a core into a given number of wrapper chains and computes the test time of the core**. The main aim of this wrapper algorithm is to balance the wrapper chains so that the longest wrapper chain length can be minimized. The longest wrapper chain evaluates the test time of the core. The test time is dependent on the wrapper chains. The test time can be reduced if the wrapper chains are increased. In case the test time does not change with wrapper chains then we use Pareto – optimal points. **Pareto-optimal points are of interest because they make use of the lowest possible wrapper chains at a certain test time**. This also effects the test time of the whole system.

Inputs: Let **N be the number of cores**. The number of functional inputs for core i is Fin_i and the number of functional outputs for core i is $Fout_i$. Let k be the number of scan chains for core i . A set of internal scan chains for core i is **$SC_i \in \{sc_1, sc_2, \dots, sc_k\}$** . The wrapper width for core i is W_i .

Outputs: T_i is the test time calculated for the core i .

Algorithm:

$W_i = \text{number of TAMs for core } i$

$Lines = \text{int } W_i / 2$

$\#SC_i = \text{number of scan chains of core } i$

- Sort the scan chains for core i in decreasing order in length
 - Select the longest scan chains as the $(Lines)$ lines
 - While $(\#SC_i > Lines)$
 - Chain the shortest line with shortest scan chain
 - $\#SC_i = \#SC_i - 1$;
 - Update the length of the longest scan length
 - Sort scan chain in decreasing order
 - Add the Fin_i and $Fout_i$ to balance the scan chains
- i. $\max S_i = \text{number of } Fin_i + \text{longest wrapper scan chain}$
 - ii. $\max S_o = \text{number of } Fout_i + \text{longest wrapper scan chain}$
 - iii. $\min S_i = \text{number of } Fin_i + \text{shortest wrapper scan chain}$
 - iv. $\min S_i = \text{number of } Fout_i + \text{shortest wrapper scan chain}$

$$T_i = \{ (1 + \max (S_i, S_o) TP) + \min(S_i, S_o) \}$$

This BFD algorithm calculates the test time of each core in the core based SOC. The test time for a core with its possible wrapper width from 1 to 64 is calculated for three ITC'02 benchmark circuits. For each core pareto-optimal points are selected and are further used for test scheduling.

Pareto-optimal points : Consider a core 12 of p93791 of ITC'02 benchmark circuits. It has 289 functional inputs, 8 functional outputs, 391 bidirectional I/Os and 46 internal scan chains of lengths: 33 scan chains x 93 bits and 13 scan chains x 92 bits. The wrapper design in this section can be used to arrange the core 12 for values of TAM width w between 1 to 64 bits. As functional inputs are more than functional outputs therefore $\max(s_i, s_o) = s_i$. As w increases s_i decreases in a series of distinct steps. This behavior causes as w increases, the redistribution of internal scan chain happens thus s_i can be reduced only by increasing w . For example, when core 12 of p93791 is arranged using

tam width 33 the test time is 72819. Test time is dependent on s_i . The test time remains constant till w reaches 39. Hence, for $33 \leq w \leq 39$, only 33 wrapper scan input chains need to be designed.

Hierarchical Constraint: Wrapper design for a hierarchical core is P_H problem. Each hierarchical core has cores embedded inside it called child cores. These child cores may also have some cores embedded inside it. In order to test the parent core, child cores need to be tested first. In hierarchical cores, the child cores are tested with the same TAM allotted to the parent core. So the wrapper design for these cores is for the same TAM width. The test time for child cores remains same as BFT for flat cores. But wrapper design for parent core is as shown in Figure. In Figure 3.1 n represents the number of child output cells and m represents the number of parent output cells:

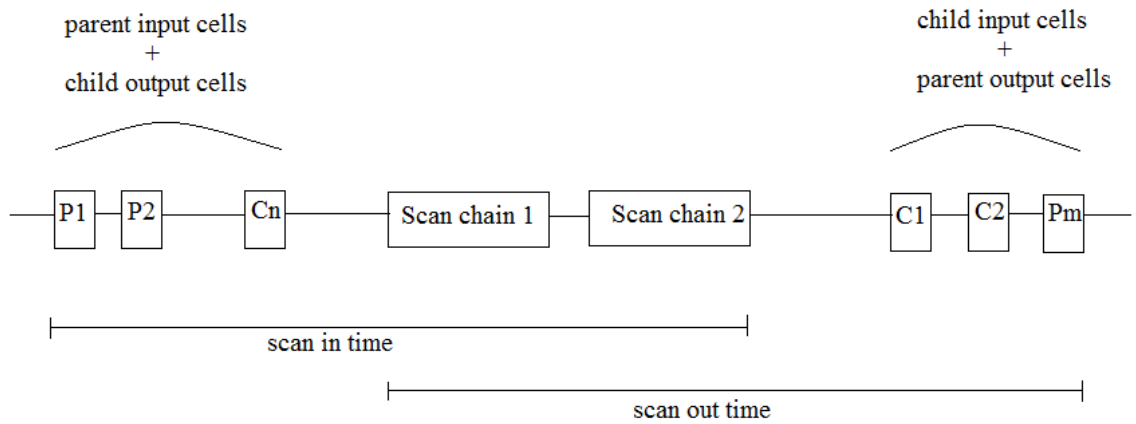


Figure 3.1 Arrangement of elements of parent core.

3.1.2.2 GREEDY ALGORITHM

Greedy algorithm is an algorithm which follows the problem solving heuristic of choosing the local optimal constraints at each stage. This algorithm may provide locally optimal solutions that can be globally optimal solutions at a reasonable time. This is a test scheduling algorithm. Using greedy schedule we may not get the minimum test time so the schedule is heuristically improved for time minimization. The pictorial representation of the schedule is given in figure. The y-axis of each rectangle represent the bandwidth of

the particular core and the x-axis represents the test time for that core. The maximum power and maximum bandwidth in Figure 3.1 is 12 and 10 respectively. At a particular instance the power (p) and bandwidth (bw) should not exceed the maximum value. The cores should be closely bound to get the minimum test time. The cores with bandwidth and power lesser than the total bandwidth and total power can be tested using this algorithm.

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 (TTT) can be calculated and stored. Then 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 TTT is compared and the lowest TTT is considered to be the best schedule.

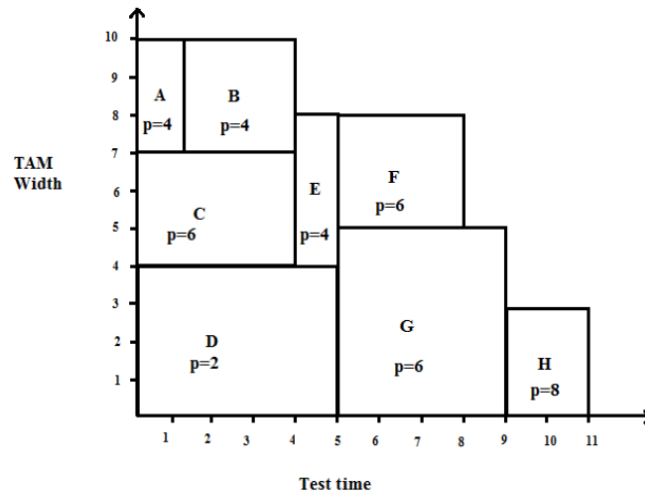


Figure 3.2 Representation of a schedule in rectangle bin packing consists of eight cores.

The scheduling of cores using greedy algorithm is done by two methods based on TAM design architecture: i). Using fork and merge technique and ii). Using test bus architecture.

i.) Algorithm using fork and merge technique.

Input:

1. N: total no of cores.
2. maxBW: total bandwidth available.

3. maxP: total power available.
4. Core set: a set of cores, for each core
 - i. number of core,
 - ii. bandwidth of core,
 - iii. max power consumption of the core,
 - iv. test time of the core,
 - v. integer u: to check whether the core is scheduled or not,
 - vi. start time of the core.

Outputs:

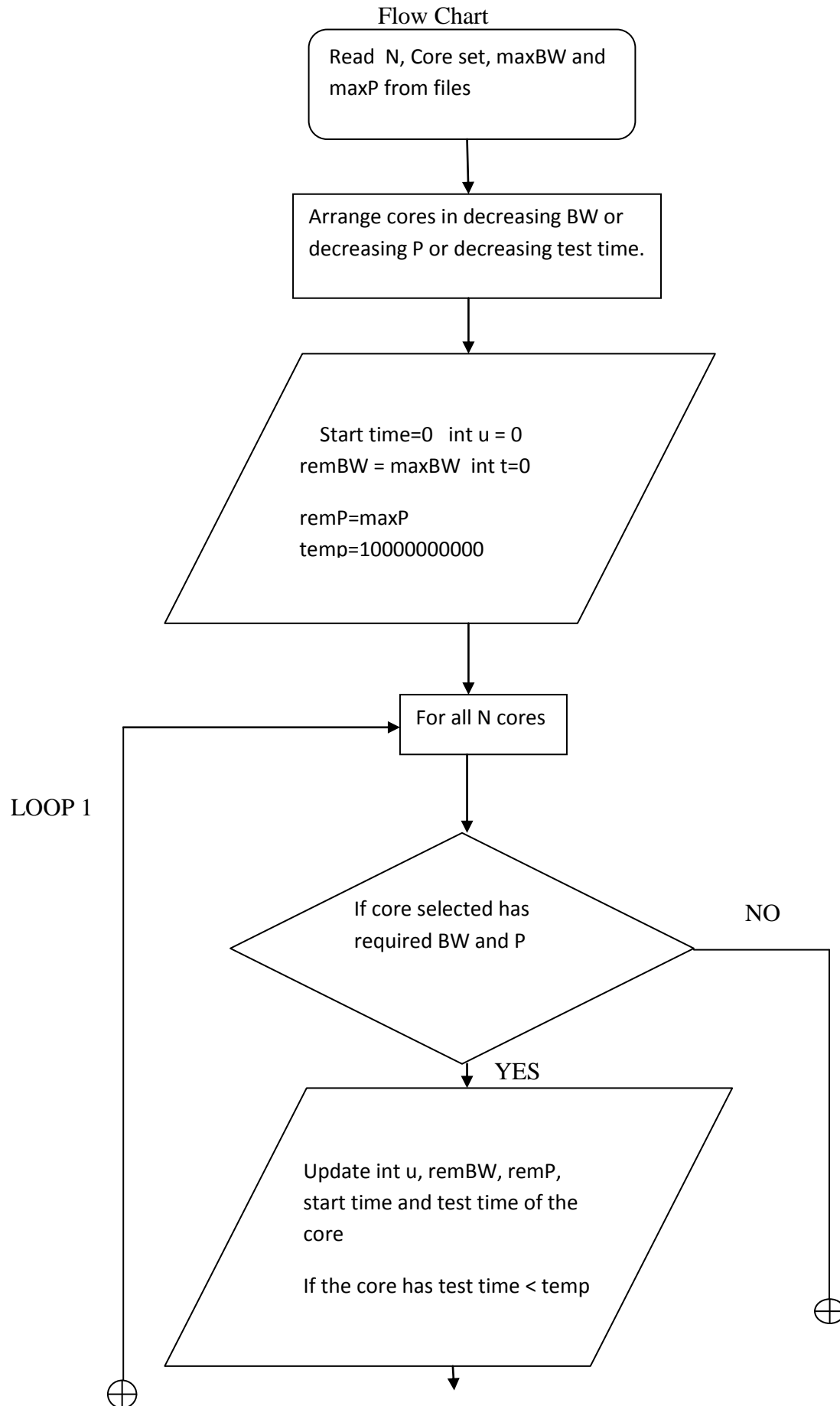
1. Core set: for each core
 - i. the test end time of core.
 - ii. integer u value to check whether every core is tested or not.
2. total test time of the schedule.

Proposed Algorithm:

- Get the inputs N, Core set, maxBW, maxP.
- Arrange the Core set with decreasing BW and decreasing P, decreasing Power and decreasing test time or any other arrangement possible.
- Set the values of start time (ST) = 0; remBW = maxBW; remP = maxP; integer u = 0; integer temp = 1000000000; integer t=0;
- **for** i=0 to N-1 **do**
- **for** i=0 to N-1 **do**
- Select the core with integer u = 0, which has the required bandwidth and power from the arranged cores.
- Update the remBW, remP, integer u=1;
- start time of the core = ST;
- test time of core = ST + test time of core;
- **for** i=0 to N-1 **do**
- **if** integer u = 1 and test time of that core < temp
- then**

- temp = test time of the core;
- integer t = number of core;
- **for** i=0 to N-1 **do**
- **if** the remP and remBW \leq power and bandwidth of the core and if integer u=1 and test time of core = temp
then
 - Update bandwidth and power of the core again
 - Set the integer u = 2;
 - Set integer u = 2;
 - Update the remBW and remP;

In this algorithm, the test time of the last core selected for scheduling will be updated as the total test time for the whole scheduling process. This proposed algorithm is applied to benchmark circuits p93791 and p22810.



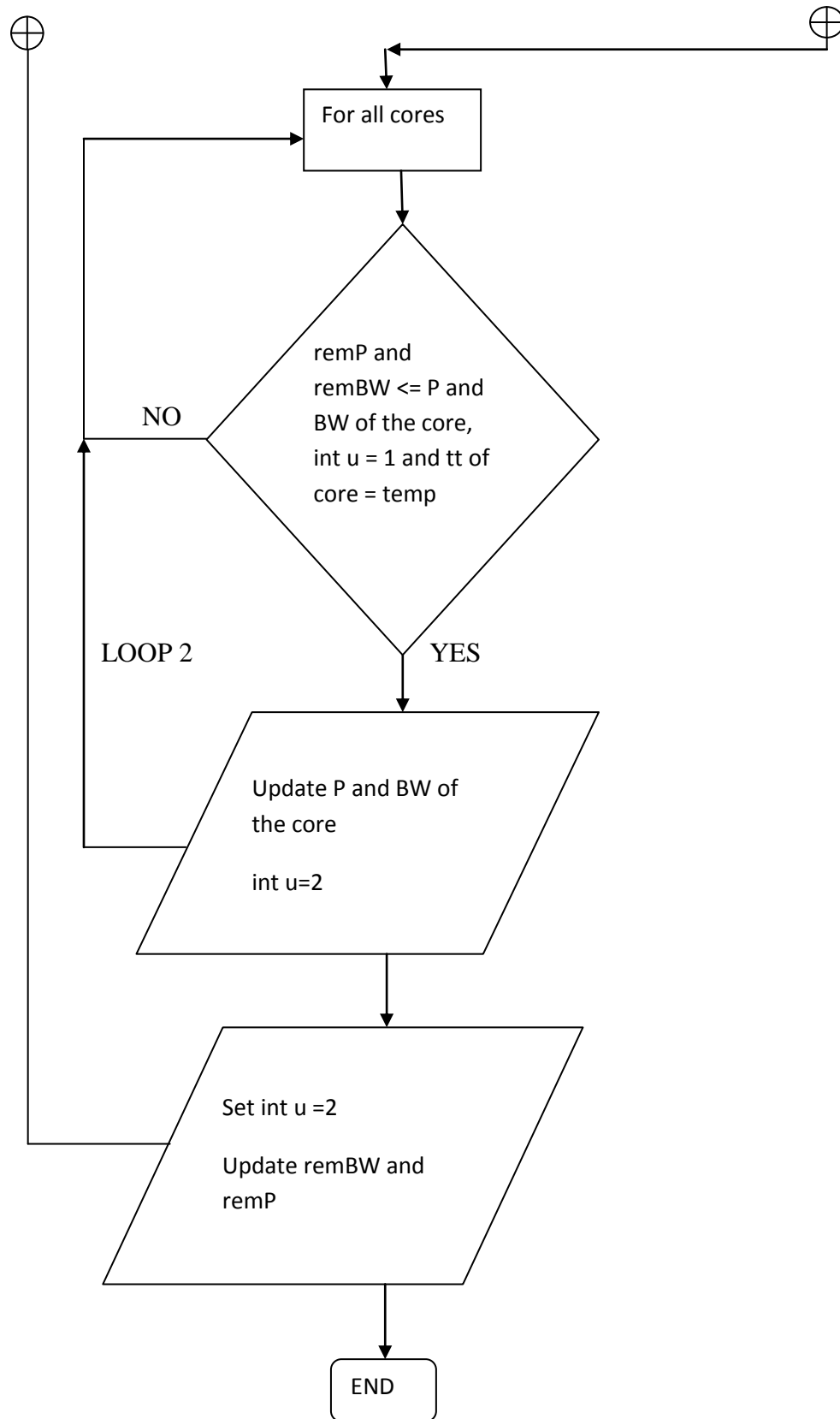


Figure 3.3 Flow Chart Diagram of Greedy Algorithm

ii). Algorithm for test bus architecture.

The greedy algorithm for test scheduling without power constraint using test bus architecture is described below:

- Partition the Total TAM width into p no of partitions.
- Construct the partitions of TAM widths tw_1, tw_2, \dots, tw_p where tw_i ($\forall i \in p$) represents sum of test time of cores
 - $tc_{i(w_j)}$ ($\forall j \in N$) for tam width w_i .
 - $tw_i = (tc_{1(w_i)} + tc_{2(w_i)} + tc_{3(w_i)} + \dots + tc_{N(w_i)})$
- Select the partition with maximum and minimum test times as $\max(tw_i)$ and $\min(tw_i)$.
- Select the core with maximum test time from $\max(tw_i)$.
- Subtract the test time value of that core from every partition except the $\min(tw_i)$.
- Update the test time values of all partitions.
- Repeat the Step 5,6 and 7 untill all the cores are selected and subtracted once.
- After step 8, all partitions contain cores resulting in minimum test time.

The greedy algorithm for test scheduling with power constraint using test bus architecture is described below:

- The cores are divided into number of partitions using greedy partition algorithm. Each partition w_j has cores.
- All cores in each partition is greedily arranged in decreasing order of test time.
- Select $\max w_j$ and schedule the first core. Update the start time and the end time.
- Select the next partition with $\max w_j$.
- Schedule the core with condition such that the sum of powers of cores scheduled at that time should not exceed the $\max P$. If $\max P$ is exceeded then select the next

core for scheduling. Update the arrangement in the partition. Update the start time and end time.

- Repeat the step 4 & 5 for all partitions.
- Select the partition with minimum end time and schedule the next core from that partition as done in step 5.
- Repeat step 7 until every core of every partition is scheduled.
- Partition with the maximum end time represents the total test time of the circuit.

3.1.2.3 E-ACO ALGORITHM

ACO algorithm is about the behavior of the ants, how they find the shortest path from their nest to the food. While searching for food, ants excrete a hormone called pheromone. Ants smell that pheromone and choose their ways that have high pheromone concentrations. During this process ants may find a shortest path due to pheromone trails left by ants from their nest to the food and back.

In enhanced ACO, the core selection for scheduling is done on the basis of $maxp$ and $maxtw$ and is partially dependent on probability. Firstly, a core is selected using probability, then it is tested using the constraints. If the conditions are satisfied, the core is selected. Otherwise, the core with next highest probability is tested. This step is repeated until all conditions are satisfied. Let us assume an ant a starts from a core i and has to reach out to other cores. The first core i is selected randomly and the next core j is selected using probability $prob$. This process is repeated until all cores are scheduled. Ant density model is used in this algorithm. According to Ant density model, the pheromone trail quantity left by ant through their paths is constant.

$$prob_{ij} = \frac{\tau_{ij} \eta_{ij}^{\beta}}{\sum_{i=n}^n \tau_{ij} \eta_{ij}^{\beta}} \quad 3.6$$

$prob_{ij}$ = probability for core j as the next core to be scheduled.

τ_{ij} = pheromone trail value from path i to $j \geq 0$.

η_{ij} = it is the heuristic value which is dependent on the test time of the cores ≥ 0 .

$\eta_{ij} = 1/tt$. Where tt is the test time of the core.

β is a parameter used to enhance the heuristic value. Its value is higher for the hierarchical cores than the flat cores so that hierarchical cores can be tested first. When an ant moves from core i to j then pheromone trail needs to be updated through that path. This process is called trail intensification. The trail intensification can be formulated using equation below:

$$\tau_{(i \text{ to } j)} = \begin{cases} Q & \text{if ant goes from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases} \quad 3.7$$

Q is a constant parameter. It is the quantity of trail left when ant moves from core i to j . Trail intensification is done at the time of core selection. After scheduling, using an ant a , the test time is calculated. Scheduling process is repeated for no. of ants and the best test time of all is selected. The results using ITC'02 benchmark circuits are shown in the next chapter. This algorithm schedules the cores with fork and merge technique and provide optimal solution.

Input: Let there be n number of cores with maximum tan width available is $maxtw$ and maximum power dissipation allowed is $maxp$. Each core has a core number, tan width tw , power dissipation of each core p , test time of each core calculated using wrapper algorithm tt and the int u which is used to check whether the core is scheduled or not.

Output: $best_time$ is the minimum test time obtained after number of iterations. Test time is calculated for each iteration and the $best_test$ time represents the minimum test time of all the test times.

Proposed Algorithm

- Initialize the pheromone matrix $\tau[i,j]$ and probability matrix $prob[i,j]$. i is the current core and j is the next core to be scheduled.
- Before scheduling the value of i need to be initialized.
- **For** all ants **do**

Remtw = *maxtw*;

Remp = *maxp*;

- Exclude the pareto points of the core selected as i .
- Update *remtw*, *rmp*, *core end time*, u and *start_time*;
- **For** 0 to $n-1$ **do**

Calculate the probability for core j and Update the $prob[i,j]$.

Select the core with maximum probability.

- **if** $tw_j < remtw \ \&\& \ p < remp$ **then**

Update *remtw*, *rmp*, u , *core end time* and *start_time*;

- **else**

Initialize an array which has *core end time* of all cores that are scheduled.

Sort the array in ascending order.

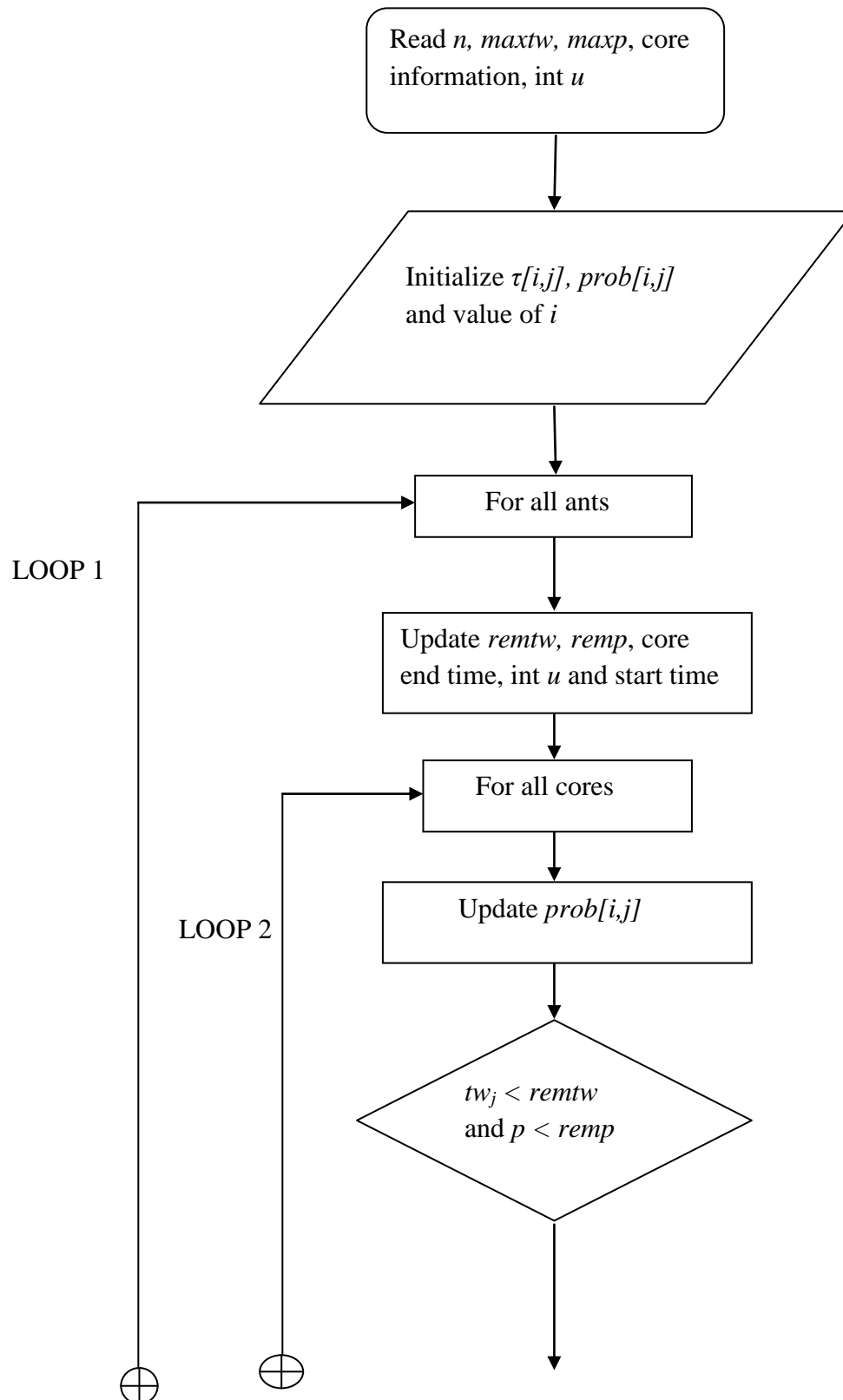
- **For** all array values **do**
- **if** $tw_j < remtw \ \&\& \ p < remp$ **then**

the cores of this *core end time* is selected as start time.

break;

- Update *remtw*, *rmp*, u , *core end time* and *start time*;
- $\tau[i,j]$ is updated when a path is confirmed from i to j ;
- value of j = value of i ;
- Test time is calculated after every core is scheduled.
- Value of i is initialized for scheduling of next ant.
- Calculate the best test time.

Flow Chart



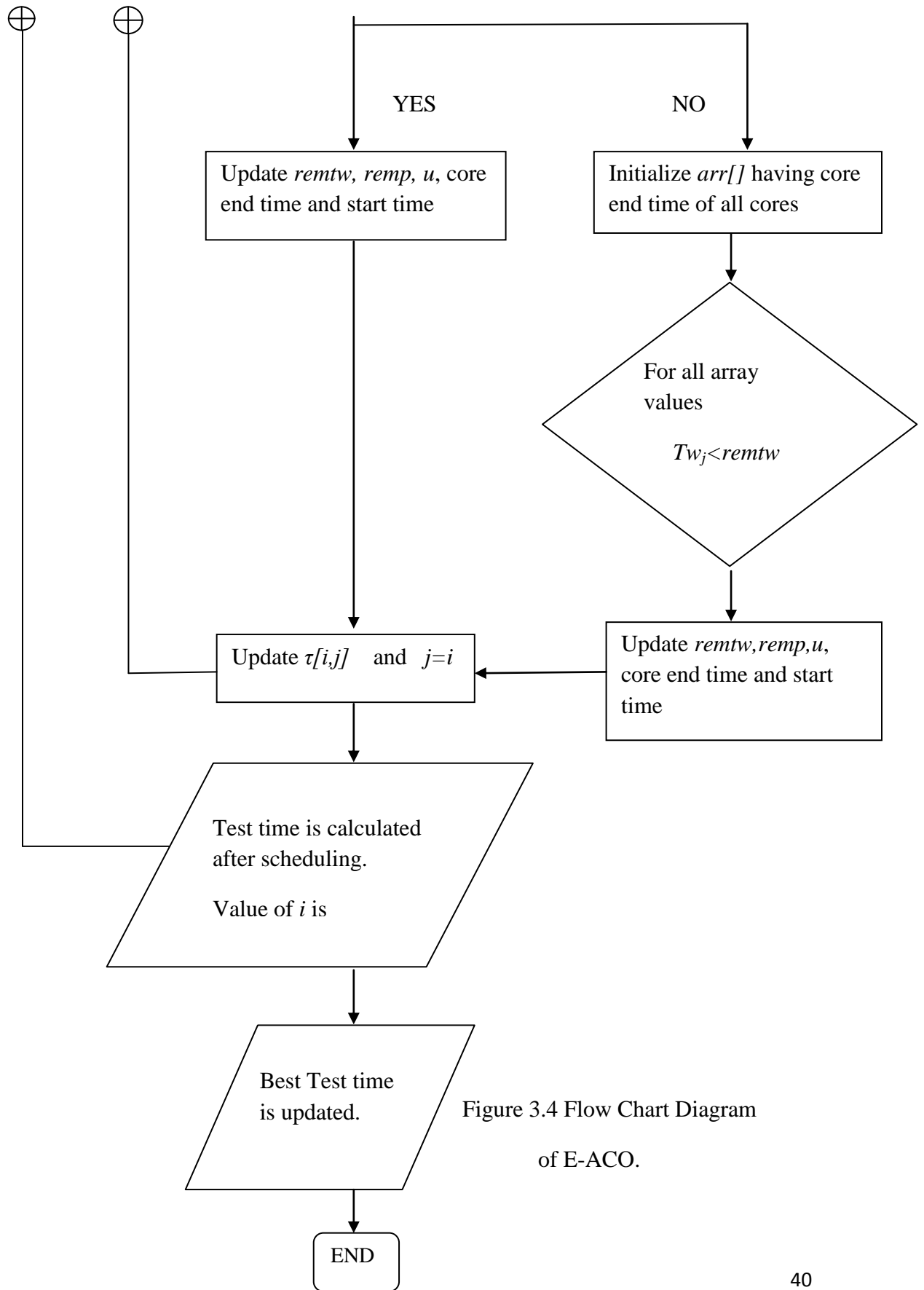


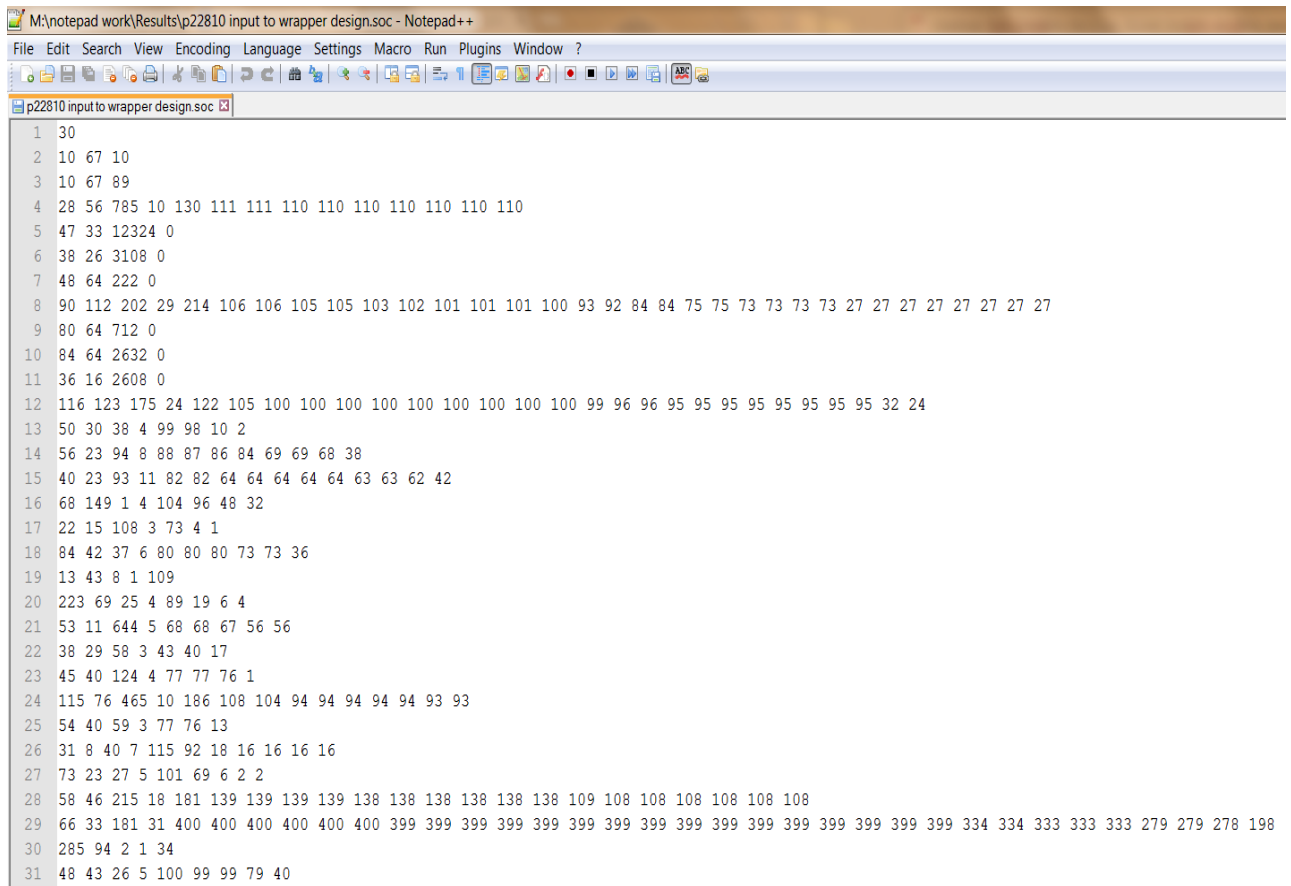
Figure 3.4 Flow Chart Diagram
of E-ACO.

SIMULATION RESULTS

4.1 RESULTS OF WRAPPER DESIGN ALGORITHM

[illegible]

41



In Figure 4.1(a),(b) and (c) the first line defines the number of cores in the benchmark circuit. In the second line the core information of core 1 is defined in the format:

- Number of functional inputs.
- Number of functional outputs.
- Test patterns.
- Number of scan chains.(k)
- Length of each scan chain. (sc_1 sc_2 sc_3 sc_k).

Similarly, next lines in the input file represents rest of the cores in that benchmark circuits.

The output of this wrapper design algorithm is shown in figure 2(a),(b) and (c) for d695, p22810 and p93791 respectively.

| 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 | 4562 | 4562 | 4562 | 4562 | 4561 | 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 4.2(a) Wrapper design output file of d695.

In figure 4.2(a),(b) and (c) each row represents a core of benchmark circuit and each column represents the wrapper TAM width from 1 to 24. Output file contains test time of each core with varying wrapper TAM widths.

[illegible]

Figure 4.2(b) Wrapper design output file of p22810.

[illegible]

Figure 4.2(c) Wrapper design output file of p93791.

4.2 RESULTS OF TEST SCHEDULING ALGORITHMS.

In this section, simulation results of greedy algorithm and enhanced ant colony optimization algorithm are presented. The results are presented for three ITC'02 benchmark circuits d695, p22810 and p93791.

4.2.1 GREEDY ALGORITHM

Test time has been calculated for three benchmark circuits using greedy algorithm. The results are calculated for flat with power constraint. In Figure 4.3 bar graph plot is presented used to find the pareto-optimal points. Input for d695 and p22810 is shown in Figure 4.4 and 4.5 considering them as flat SOCs and output of respective circuits in shown in Table 4.1 and 4.2 considering power constraint.

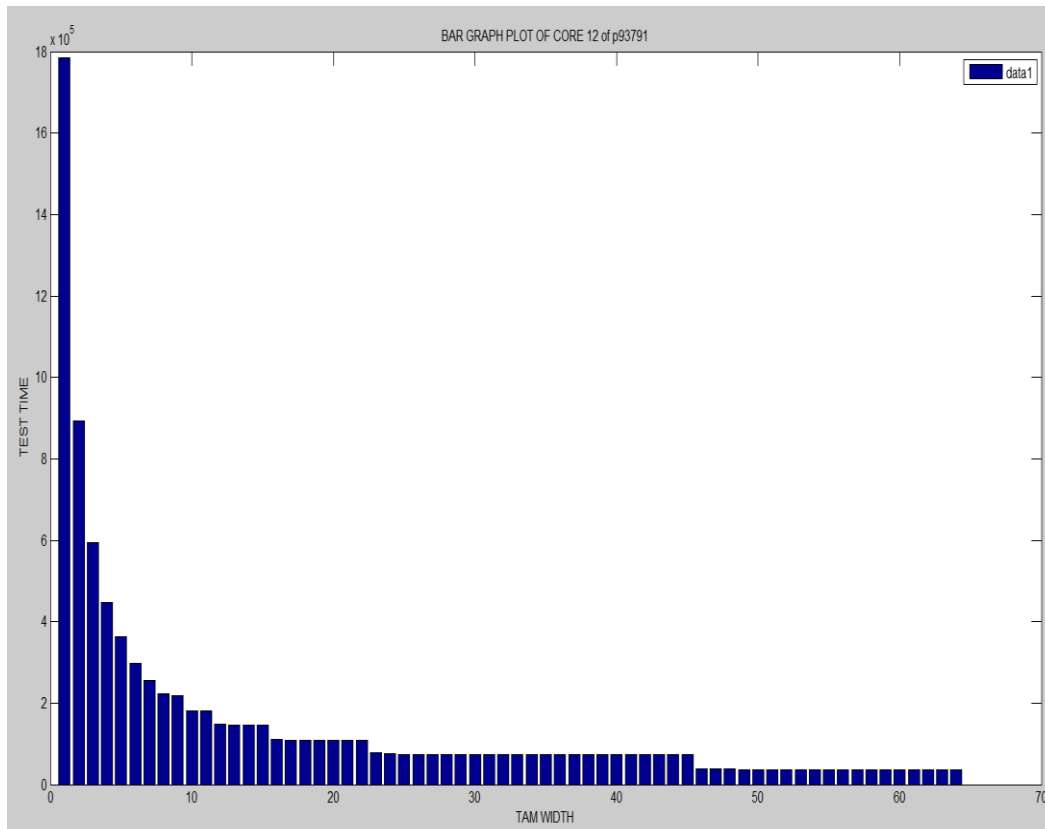


Figure 4.3 Bar Graph plot (test time v/s tam width) for core 12 of p93791.

| core no. | Number of cores | maxBW | maxP | TAM width | Power | Test time | Start time |
|----------|-----------------|-------|------|-----------|-------|-----------|------------|
| 1 | 10 | 16 | 3000 | | | | |
| 2 | 1 | 11 | 660 | 50 | 0 | 0 | |
| 3 | 2 | 14 | 602 | 1175 | 0 | 0 | |
| 4 | 3 | 3 | 823 | 2475 | 0 | 0 | |
| 5 | 4 | 5 | 275 | 5796 | 0 | 0 | |
| 6 | 5 | 11 | 690 | 14973 | 0 | 0 | |
| 7 | 6 | 16 | 354 | 11744 | 0 | 0 | |
| 8 | 7 | 9 | 530 | 6605 | 0 | 0 | |
| 9 | 8 | 6 | 753 | 4583 | 0 | 0 | |
| 10 | 9 | 11 | 641 | 2110 | 0 | 0 | |
| 11 | 10 | 16 | 1144 | 7518 | 0 | 0 | |

Figure 4.4 Input data of d695 with power constraint.

| core no. | Number of cores | maxBW | maxP | TAM width | Power | Test time | Start time |
|----------|-----------------|-------|-------|-----------|-------|-----------|------------|
| 1 | 30 | 16 | 25000 | | | | |
| 2 | 1 | 14 | 173 | 51 | 0 | 0 | |
| 3 | 2 | 12 | 173 | 535 | 0 | 0 | |
| 4 | 3 | 10 | 1238 | 102951 | 0 | 0 | |
| 5 | 4 | 12 | 80 | 61622 | 0 | 0 | |
| 6 | 5 | 10 | 64 | 15542 | 0 | 0 | |
| 7 | 6 | 11 | 112 | 1337 | 0 | 0 | |
| 8 | 7 | 11 | 2489 | 43633 | 0 | 0 | |
| 9 | 8 | 16 | 144 | 4276 | 0 | 0 | |
| 10 | 9 | 10 | 148 | 23694 | 0 | 0 | |
| 11 | 10 | 9 | 52 | 13041 | 0 | 0 | |
| 12 | 11 | 12 | 2505 | 34622 | 0 | 0 | |
| 13 | 12 | 3 | 289 | 34622 | 0 | 0 | |
| 14 | 13 | 6 | 739 | 13062 | 0 | 0 | |
| 15 | 14 | 6 | 848 | 12087 | 0 | 0 | |
| 16 | 15 | 4 | 487 | 187 | 0 | 0 | |
| 17 | 16 | 4 | 115 | 7997 | 0 | 0 | |
| 18 | 17 | 4 | 580 | 5533 | 0 | 0 | |
| 19 | 18 | 6 | 237 | 888 | 0 | 0 | |
| 20 | 19 | 4 | 442 | 2272 | 0 | 0 | |
| 21 | 20 | 6 | 441 | 44441 | 0 | 0 | |
| 22 | 21 | 4 | 167 | 2566 | 0 | 0 | |
| 23 | 22 | 4 | 318 | 9706 | 0 | 0 | |
| 24 | 23 | 6 | 1309 | 92191 | 0 | 0 | |
| 25 | 24 | 3 | 260 | 4634 | 0 | 0 | |
| 26 | 25 | 3 | 363 | 4726 | 0 | 0 | |
| 27 | 26 | 3 | 311 | 2775 | 0 | 0 | |
| 28 | 27 | 11 | 2512 | 53474 | 0 | 0 | |
| 29 | 28 | 16 | 2921 | 145052 | 0 | 0 | |
| 30 | 29 | 4 | 413 | 185 | 0 | 0 | |
| 31 | 30 | 4 | 508 | 3231 | 0 | 0 | |

Figure 4.5 Input data of p22810 with power constraint.

Table 4.1 Test time under power constraint for p93791.

| TAM width | maxP = 30000 | | maxP= 25000 | | maxP= 10000 | |
|-----------|--------------|---------|-------------|---------|-------------|-----------|
| | greedy | [8] | greedy | [8] | greedy | [8] |
| 128 | 228 718 | 457 862 | 228 718 | 493 599 | 432 241 | 568 734 |
| 80 | 449 134 | 787 588 | 449 134 | 821 475 | 493 419 | 1 091210 |
| 64 | 454 711 | 945 425 | 454 711 | 965 383 | 493 416 | 1 117 385 |

Table 4.2 Test time under power constraint for p22810.

| TAM width | maxP = 10000 | | maxP= 6000 | | maxP= 3000 | |
|-----------|--------------|---------|------------|---------|------------|---------|
| | greedy | [8] | greedy | [8] | greedy | [8] |
| 128 | 61 620 | 128 332 | 68 307 | 157 568 | 96 909 | 293 021 |
| 80 | 98 133 | 195 733 | 98 133 | 209 559 | 115 194 | 356 215 |
| 64 | 127 018 | 236 186 | 127 018 | 250 487 | 127 018 | 309 255 |

In this table, results are compared with recent paper [8]. Test time has reduced in comparison to [8] because the cores used are soft cores. In Table 4.3, test time calculations are shown for three benchmark circuits using hard cores. Table 4.4 represents the test time of the p22810 and p93791 using hard cores and power constraint.

Table 4.3 Test time with varying TAM width for hard cores.

| TAM width | d695 | p22810 | p93791 |
|-----------|--------|---------|-----------|
| 16 | 48 758 | 507 271 | 2 347 950 |
| 24 | 33 443 | 364 674 | 1 370 322 |
| 32 | 23 738 | 271 377 | 1 008 396 |
| 40 | 23 442 | 217 118 | 795 185 |
| 48 | 17 129 | 194 672 | 765 803 |
| 56 | 14 973 | 150 589 | 613 119 |
| 64 | 14 973 | 134 777 | 613 356 |

Table 4.4 Greedy algorithm results for p22810 and p93791.

| TAM width | p22810 | | p93791 | |
|-----------|-----------|-----------|------------|------------|
| | maxp=2000 | maxp=1800 | maxp=20000 | maxp=17000 |
| 16 | 44 803 | 44 803 | 1 784 515 | 1 784 515 |
| 32 | 28 441 | 28441 | 913 605 | 934 889 |
| 48 | 19 685 | 20802 | 616 890 | 625 875 |
| 64 | 20 024 | 20024 | 395 286 | 398 654 |

Graph representation is done in figures below:

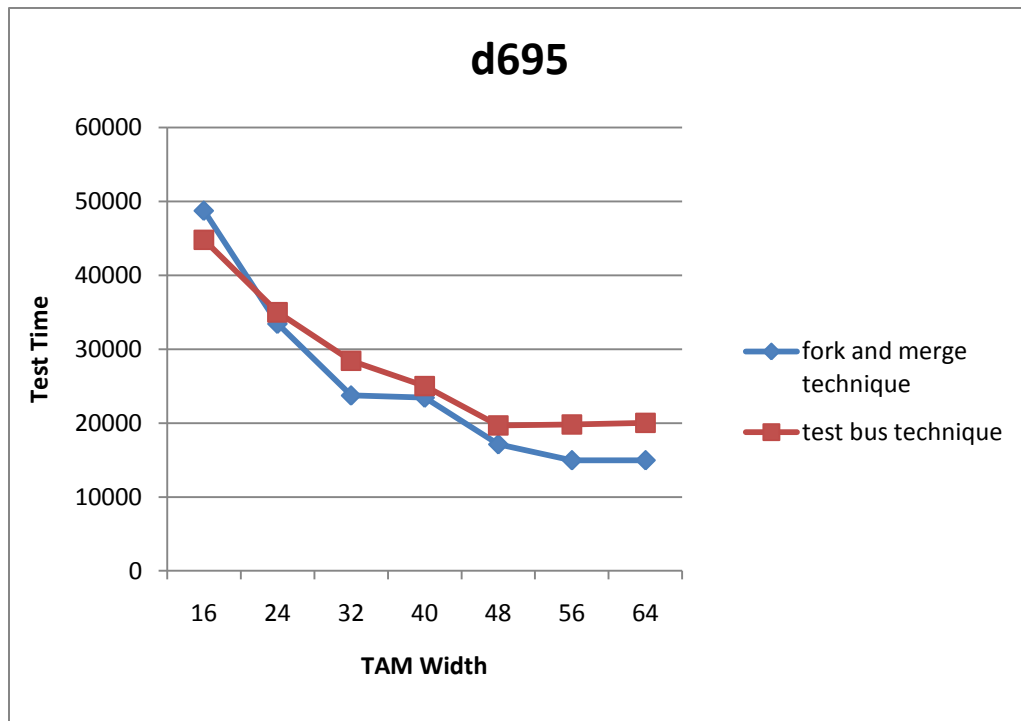


Figure 4.6 Graphical representation of test time for varying TAM width for d695.

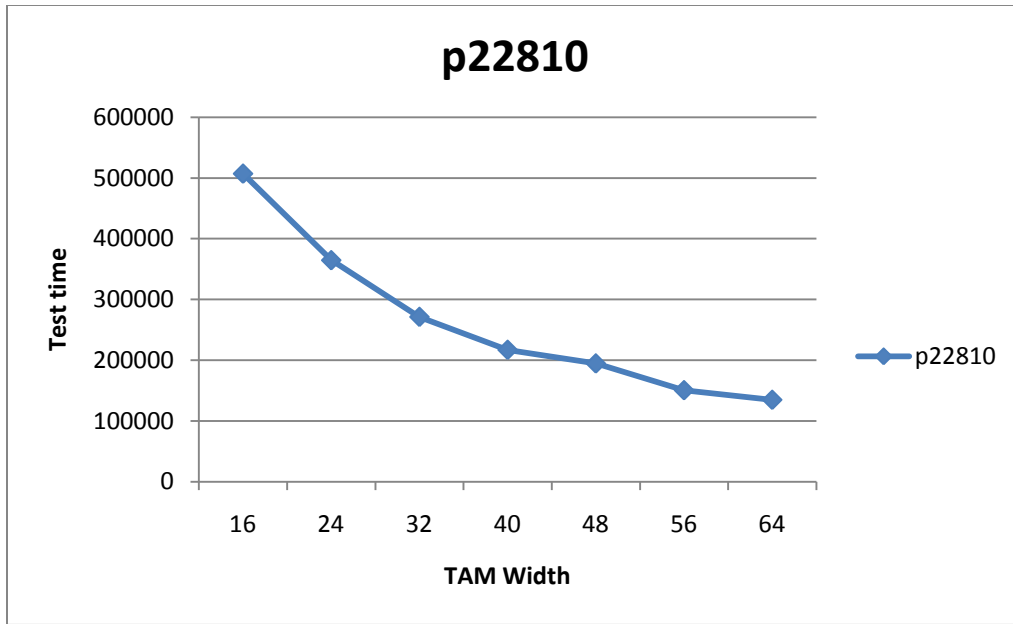


Figure 4.7 Graphical representation of test time for varying TAM width for p22810.

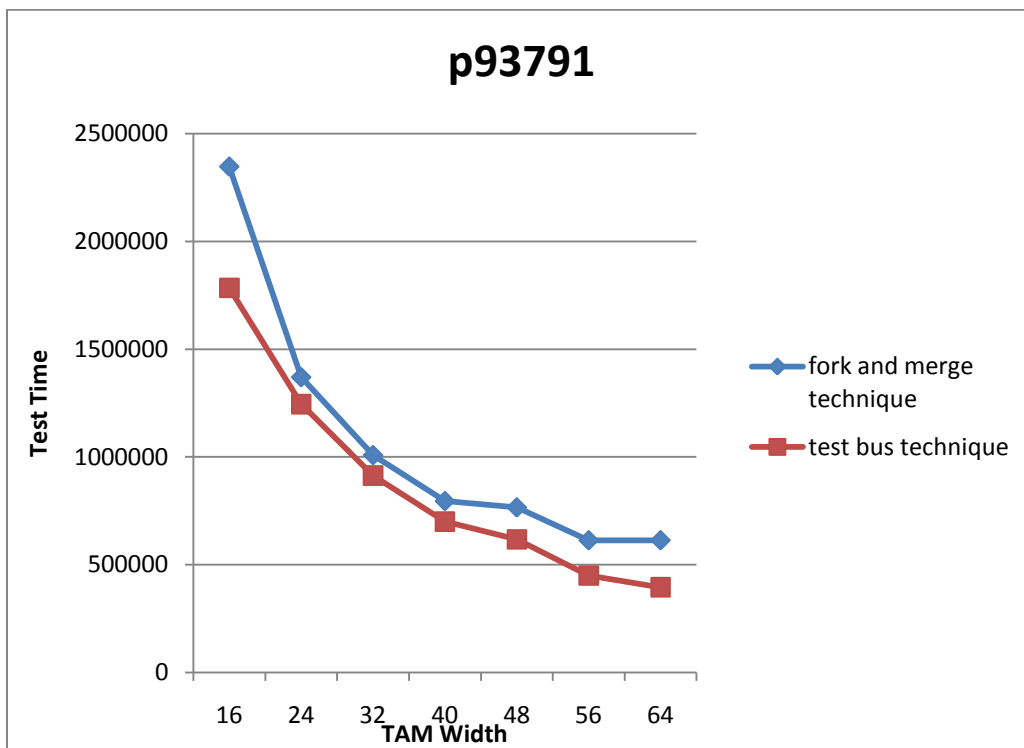


Figure 4.8 Graphical representation of test time with varying TAM width for p93791.

4.2.2 E-ACO ALGORITHM

Using E-ACO simulation results has been calculated for two benchmark circuits p22810 and p93791. The results are also compared with [35]. Table 4.5 compares the results of E-ACO algorithm with [35]. The Table 4.6, 4.7 and 4.8 presents the **test time of three benchmark circuits under power constraint**. Figure 4.9 and 4.10 is the graphical representation of Table 4.5. Table 4.9 shows the test time calculation and comparison with [35] under hierarchical constraint.

Table 4.5 Simulation results using E-ACO and compared with [35].

| SOC | TAM Width | Enhanced ACO | [35] |
|--------|-----------|--------------|-----------|
| p22810 | 16 | 280 725 | 458 068 |
| | 32 | 190 511 | 222 471 |
| | 64 | 130 586 | 133 405 |
| p93791 | 16 | 1 651 355 | 1 791 638 |
| | 32 | 877 738 | 912 233 |
| | 64 | 432 394 | 455 738 |

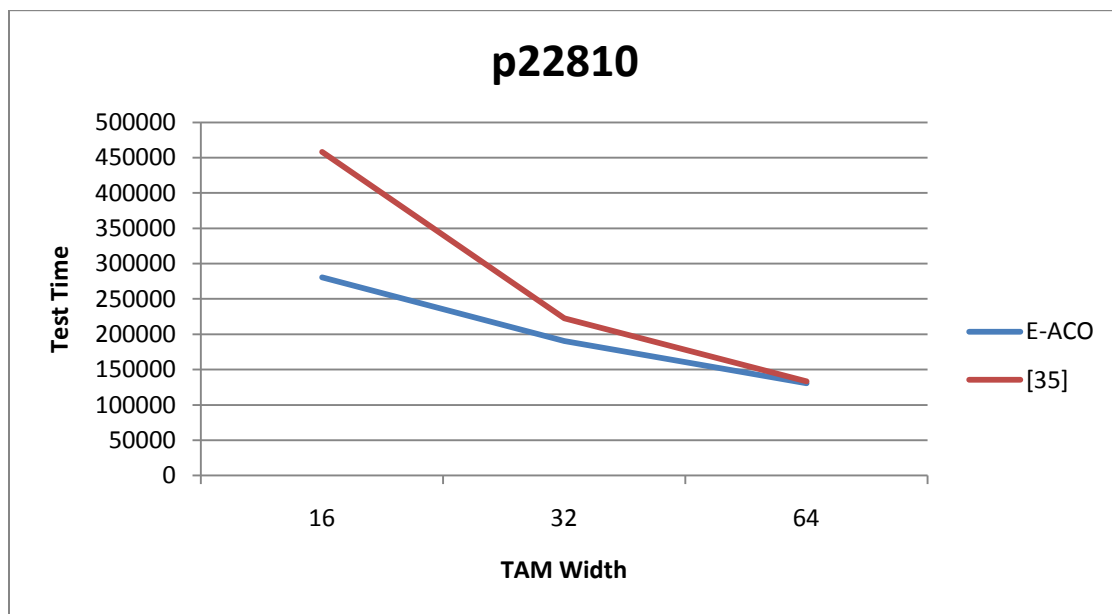


Figure 4.9 Graphical representation of Test time with varying TAM width for p22810.

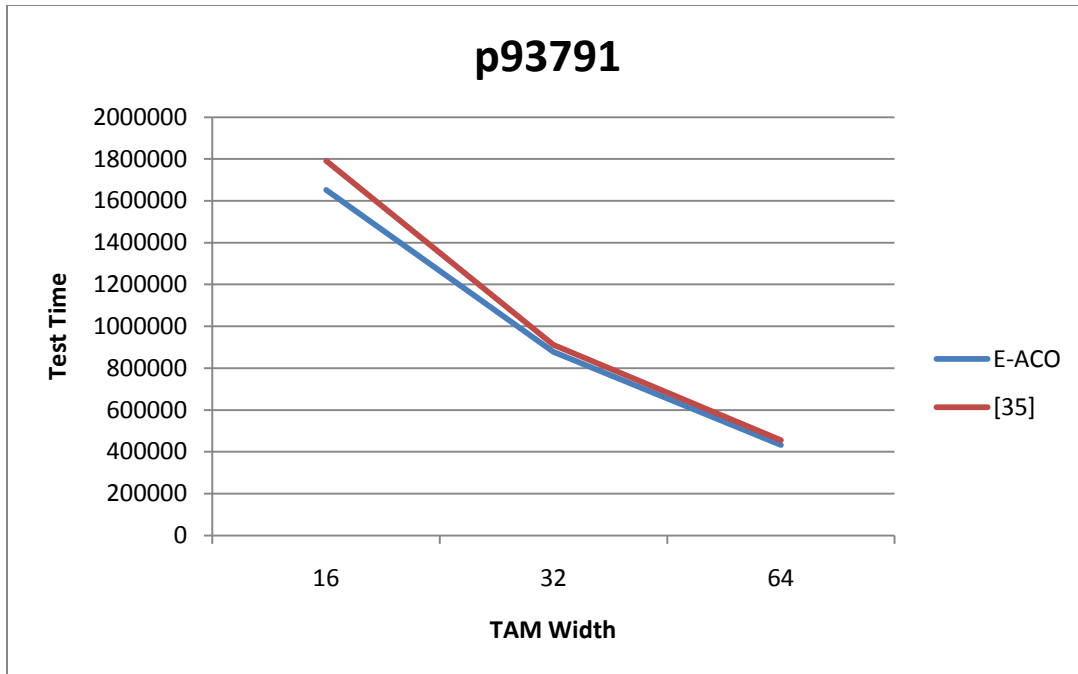


Figure 4.10 Graphical representation of Test time with varying TAM width for p93791.

Table 4.6 Test time for d695 using E-ACO under power constraint.

| TAM Width | Maxp=1500 | Maxp=2000 | Maxp=2500 |
|-----------|-----------|-----------|-----------|
| 16 | 57 869 | 55 777 | 55 777 |
| 32 | 31 947 | 25 791 | 25 791 |
| 64 | 21 755 | 17 540 | 14 605 |

Table 4.7 Test time for p22810 using E-ACO under power constraint.

| TAM Width | Maxp=6000 | Maxp=8000 | Maxp=10000 |
|-----------|-----------|-----------|------------|
| 16 | 280 775 | 280 775 | 280 775 |
| 32 | 290 511 | 290 511 | 190 511 |
| 64 | 176 663 | 173 292 | 130 586 |

Table 4.8 Test time for p93791 using E-ACO under power constraint.

| TAM Width | Maxp=20000 | Maxp=25000 | Maxp=30000 |
|-----------|------------|------------|------------|
| 16 | 1 651 355 | 1 651 355 | 1 651 355 |
| 32 | 1 177 738 | 1 177 738 | 877 738 |
| 64 | 652 885 | 652 885 | 432 394 |

Table 4.9 Test time using E-ACO compared with [35] under hierarchical constraint.

| TAM Width | p22810 | | p93791 | |
|-----------|---------|---------|-----------|-----------|
| | E-ACO | [35] | E-ACO | [35] |
| 16 | 605 164 | 621 895 | 2 452 303 | 3 363 819 |
| 32 | 322 437 | 493 290 | 1 267 400 | 1 504 806 |
| 64 | 139 467 | 296 445 | 716 593 | 1 908 099 |

CONCLUSION AND FUTURE WORK

Work is done on various heuristic algorithms which have been optimized under power and hierarchical constraints for 2D SOC's. The main aim of these algorithms is to minimize the total test time. This is done by creating an efficient schedule. In this report the analysis of previous proposed solutions and development is done. The results for three ITC'02 benchmark circuits have been evaluated. The results show an effective decrease in the test time as compared to previous papers mentioned in chapter 4.

Greedy algorithm for test scheduling is proposed which is a heuristic algorithm and can be improved. Using this algorithm, test schedules are generated and test time is reduced under the power constraint. Simulation results show the optimized results.

Enhanced ACO algorithm is enhanced version of ACO algorithm in which the test schedule is prepared using ACO algorithm but core assignment is done under power constraint and hierarchical constraint. Simulation results show the optimized results.

The future work in field of test scheduling can be done by modifying these algorithms for 3D SOC's. By including 3D SOC's, a new constraint is introduced i.e. TSV constraint. These algorithms can be optimized to reduce the test time of 3D SOC's. The proposed method can be extended using the test rail architecture. Other algorithms can be optimized to calculate minimum test time considering power constraint, hierarchical constraint and TSV constraint.

LIST OF PUBLICATIONS

1. Authors: Naveen Dewan and Harpreet Vohra
Paper title: Test Scheduling of Core Based SOC using Greedy Algorithm.
Published in: International Journal of Engineering and Research Applications (IJERA), 4(9), pp. 80-85, September 2014.

2. Authors: Naveen Dewan and Harpreet Vohra
Paper title: Test Optimization of 2D SOC using Enhanced ACO Algorithm.
Published in: International Journal of Computational Engineering and Management (IJCEM), 18(3), pp. 19-22, May 2015.

3. Authors: Naveen Dewan, Prashit Aggarwal and Harpreet Vohra
Paper title: Test time and Power Optimization of 2D SOC using GA and Greedy Algorithm.
Accepted in: STM Journals: Journal of Power Electronics and Power Systems.

REFERENCES

- [1] K. Ruparel, “SOC test: the devil is in the details of integration/implementation,” in *Proc. ITC*, Washington, DC, 1998, pp 1143.
- [2] G. E. Moore, “Cramming more components onto integrated circuits,” in *Proc. of the IEEE*, 1998, pp 82-85.
- [3] Intel, “Advancing Moore’s Law – The Road to 14 nm,” <http://www.intel.in/content/www/in/en/silicon-innovations/advancing-moores-law-in-2014-presentation.html>.
- [4] E. Larsson (2005). *Introduction to Advanced System on Chip Test Design and Optimization*. [Online]. Available: <http://www.springer.com/in/book/9781402032073>.
- [5] J. Aerts and E. J. Marinissen, “Scan Chain Design for Test Time Reduction in Core-Based ICs,” in *Proc. of ITC*, Washington, DC, 1998, pp 448-457.
- [6] P. Varma and S. Bhatia, “A Structures Test Re-Use Methodology for Core-Based System Chips,” in *Proc. of ITC*, Washington, DC, 1998, pp 294-302.
- [7] V. Iyenger, K. Chakrabarty, and E.J. Marinissen, “Recent Advances in test Planning for Modular Testing of Core-Based SOCs,” in *Proc. ATS*, Guam, USA, 2002, pp 320-325.
- [8] J. Pouget, E. Larsson and Z. Peng(2005). *Multiple-Constraint Driven System-on-Chip Test time Optimization*. *Springer Journal of Electronic Testing: Theory and Applications* 21, pp. 599-611.
- [9] V. Iyenger, K. Chakrabarty and E.J. Marinissen, “On Using Rectangle Pack-ing for SOC Wrapper/TAM Co-optimization,” in *Proc. IEEE VLSI Test Symposium VTS*, 2002, pp. 253-258.
- [10] J.M. Im, S. Chun, G. Kim, J. H. An and S. Kang, “RAIN (Random INsertion) scheduling algorithm for SOC Test,” in *Proc. Asian Test Symposium ATS*, 2004, pp. 242-247.
- [11] C. Giri, D.K.R. Tipparthi and S. Chattopadhyay, “A Genetic Algorithm Based Approach for System-on-Chip test Scheduling using Dual Speed TAM with Power Constraint,” in *WSEAS Transactions on Circuits and Systems*, 7(5), 2008, pp. 416-427.

- [12] A. Colorni, V. Maniezzo and M. Dorigo, "Ant System: Optimization by a colony of cooperating agents," in *IEEE Transactions on Systems, Man and Cybernetics* 26(1), 1996, pp. 29-41.
- [13] E. Larsson and H. Fujiwara, "Power Constrained Preemptive TAM Scheduling," in *Proc. European Test Workshop (ETW)*, Corfu, Greece, 2002, pp. 119-126.
- [14] E. Larsson and H. Fujiwara, "Preemptive system on chip test scheduling," in *IEICE Transactions on Information and Systems*, 87(3), 2004, pp. 620-629.
- [15] M. Sugihara, H. Date and H. Yasuura, "A novel test methodology for core based system LSIs and is testing time minimization problem," in *Proc. International Test Conference ITC*, 1998, pp. 465-472.
- [16] Y. Huang, "Resource Allocation and Test Scheduling for Concurrent Test of Core-Based SOC Design," in *Proc. of 10th IEEE Asian Test Symposium (ATS)*, 2001, pp. 265-270.
- [17] S. Koranne and V. Iyenger, "On the Use of k-Tuples for SOC Test Schedule Representation," in *Proc. International Test Conference (ITC)*, 2002, pp. 539-548.
- [18] V. Iyenger, K. Chakrabarty and E.J. Marinissen, "Test Access mechanism, Test Scheduling and Tester Data Volume Reduction for System on Chip," in *Proc. IEEE ITC*, 12, 2003, pp. 319-324.
- [19] K. Chakrabarty, "Test scheduling for Core-Based System," in *Proc. International Conference CAD*, 1999, pp. 391-394.
- [20] V. Iyenger, K. Chakrabarty and E.J. Marinissen, "Test Wrapper and Test Access Mechanism Co-Optimization for System –on-Chip," in *Proc. ITC International Test Conference*, 2001, pp. 1023-1032.
- [21] K. Chakrabarty, "Test Scheduling for Core-Based System Using Mixed-Integer Linear Programming," in *IEEE Transactions on CAD of ICs and Systems*, 2000, pp. 1163-1174.
- [22] P. Varma and S. Bhatia, "A Structured test re-use methodology for core-based system chips," in *Proc. International Test Conference ITC*, 1998, pp. 294-302.
- [23] E.J. Marinissen, S. K. Goel and M. Lousberg, "Wrapper Design for Embedded Core Test," in *Proc. IEEE International Test Conference ITC*, 2000, pp. 911-920.

- [24] K. Chakrabarthy, "Design of System-on-Chip Test Access Architecture using Integer Linear Programming," in *Proc. International Conference of IEEE VLSI Test Symposium*, 2000.
- [25] Y. Huang et al., "Resource Allocation and Test Scheduling for Concurrent Test of Core-Based SOC Design," in *Proc. 10th IEEE Asian Test Symposium ATS*, 2001, pp. 265-270.
- [26] S.K. Goel and E.J. Marinissen, "Effective and Efficient Test Architecture Design for SOC's," in *Proc. IEEE International Test Conference ITC*, 2002, pp. 529-538.
- [27] V. Iyenger, K. Chakrabarthy and E.J. Marinissen, "On using Rectangular Packing for SOC Wrapper/TAM co-optimization," in *Proc. 20th IEEE VLSI Test Symposium VTS*, 2002.
- [28] Z. jinyu, L. Xunsheng, G. Bing, X. Guangze and Sang Nan, "A Test Scheduling Scheme for Core-Based SoCs Using Genetic Algorithm," in *Proc. International Conference on Embedded Software and Systems Symposia ICESS*, 2008.
- [29] Guangyu Liu, "Parallel Elite Genetic Algorithm for Test Scheduling of SoC," in *Proc. Industrial Electronics and Applications (ICIEA)*, 2014, pp. 1985-1988.
- [30] A.Colormi, V. Maniezzo and M. Dorigo, "Ant system: Optimization by a colony of cooperating agents," in *IEEE Transactions on Systems, Man and Cybernetics*, 26(1), 1996.
- [31] Jin-Ho Ahn and Sungho Kang, "SOC Test Scheduling Using ACO-based Rectangle Packing," in *Proc. Springer ICIC*, Heidelberg, 2006, pp. 655-660.
- [32] W. Zou, S. R. Reddy, I Pomeranz and Y. Huang, "SOC Test Scheduling Using Simulated Annealing," in *Proc. VLSI Test Symposium VTS*, 2003, pp. 325-330.
- [33] Y. Huang et al., "Optimal Core Wrapper Width Selection and SOC Test Scheduling Based on 3-D Bin Packing Algorithm," in *Proc. International Test Conference ITC*, 2002, pp 74-82.
- [34] V. Iyenger, K. Chakrabarthy, M.D. Krasniewski and G.N. Kumar, "Design and Optimization of Multi-Level TAM Architectures for hierarchical SOC's," in *Proc. IEEE VLSI Test Symposium VTS*, 2003, pp. 299-304.

- [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," in *IEEE Transactions on Computers*, 58(3), 2009, pp. 409-423.
- [36] H.M. Harmanani and H.A.. Salamy, "Power-Constrained System-on-a chip Test Scheduling using a Genetic algorithm," in *Journal of Circuits, Systems and Computers, World Scientific Publishing Company*, 15(3), 2006, pp. 331-349.
- [37] Xu Chuan-pei and Dai Kui, "The optimization of Hierarchical SOC Test Architecture to Reduce test time," in *Proc. International Conference on Electronic Packaging Technology and High Density Packaging ICEPT-HDP*, 2008, pp. 1-4.
- [38] H. Hu and S. Yihe, "A Scalable Test Mechanism and its Optimization for Test Access to Embedded Cores," in *Proc. 4th International Conference on ASIC*, 2001, pp. 773-776.
- [39] T. Chen and Y. Chang, "SoC Test Scheduling using a B*- Tree based floorplanning technique," in *Proc. Asia and South Pacific Design Automation Conference*, 2005, pp. 1188-1191.
- [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," in *WSEAS Transactions on circuits and systems*, 2008, pp. 416-427.