DEVELOPMENT OF AN EFFICIENT TESTING METHODOLOGY FOR SYSTEM-ON-CHIP DESIGN

Thesis submitted for the degree of Doctor of Philosophy
(Engineering)

University of North Bengal

By

Gautam Das

Siliguri Institute of Technology
Sukna, Siliguri, W.B.

Dedicated to my parents

Acknowledgements

I wish to thank my supervisor Prof. Haripada Bhaumik for his help and assistance, for his guidance and encouragement throughout this PhD.

I wish to thank my co-supervisor Dr. Santanu Chattopadhyay for his continuous help & support through out my research. I am grateful for his invaluable help during the preparation of this thesis and several papers. Without his encouragement and constant guidance, I could not have finished this thesis. I am very grateful for his patience, motivation, enthusiasm, and immense knowledge in System-on-Chip testing that, taken together, make him a great mentor.

I wish to thank my friends and colleagues for their guidance and constructive comments throughout the period of this PhD.

I would also like to thank the Department of Electronics and Communication Engineering, Siliguri Institute of Technology for providing the required computing facilities during my research.

Last but certainly not least, this work could have not been carried out without the unshakable love, encouragement and tolerance of my parents, father-in-law, mother-in-law and my wife, to whom I am forever thankful.

Gautam Das

Declaration

I hereby declare that neither the thesis nor any part thereof have been submitted for any degree whatsoever.

Gautam Das

Certificate

This is to certify that the thesis entitled "Development of an efficient testing methodology for System-On-Chip design" submitted by Gautam Das who got his name registered on 03/06/2004 for the award of PhD (Engineering) degree of NBU, is absolutely based upon his own work under the supervision of Prof. Haripada Bhaumik and co-supervision of Dr. Santanu Chattopadhyay and neither his thesis nor any part of it has been submitted for any degree/diploma or any other academic award anywhere before.

Prof. Haripada Bhaumik
(Supervisor)

S. Chattopadhyay
Dr. Santanu Chattopadhyay
(Co-supervisor)

Dr. Santanu Chattopadhyay
Associate Professor
Department of Electronics &
Electrical Communication Engineering
Indian Institute of Technology
Kharagpur-721302, India
E-mail: santanu@ece.iitkgp.ernet.in

Abstract

Due to their relative advantages of design reuse and the reduction in time-to-market, core based system on chips are becoming more and more popular today. A major challenge in the core-based System-on-Chips is the adoption of efficient test and diagnostic strategies. The main difficulty in these systems is the poor accessibility of embedded cores from I/O terminals of the system. Tremendous increase in the size of the circuits incorporated onto a single chip has resulted in a corresponding increase in the size of the test data needed to test such systems, which has subsequently placed a heavy demand on the memory requirements of the Automatic Test Equipment. Moreover, high switching activity while testing can cause average and peak power dissipation to be much higher than the normal mode operation. This obviously can cause damage to the chip.

In this thesis, we have first presented a detailed survey report of SOC testing strategies. A number of approaches have been discussed to design an efficient Test Access Architecture. A new Test Data Compression Algorithm has been proposed to alleviate the cost of high-end Automated Test Equipment, used for System-on-Chip testing. We also propose a novel mapping strategy to reduce scan-in power dissipation while testing System-on-Chips.

Table of Contents

List of Figures

List of tables

List of algorithm

List of Abbreviations

AMBA	Advanced Microcontroller Bus Architecture
ANSI	American National Standards Institute
ARM	Advanced RISC Machine
ASIC	Application Specific Integrated Circuit
ATE	Automatic Test Equipment
ATPG	Automatic Test Pattern Generator
BFD	Best Fit Decreasing
BIST	Built-In-Self-Test
BST	Boundary Scan Test
CGU	Code Generator Unit
CPU	Central Processing Unit
CSR	Cylindrical Scan chain Registrar
CTL	Core Test Language
CUT	Core Under Test
cVIHC	Cumulative Variable-length Input Huffman Coding
CWD	Code Word Detector
DFT	Design For Test
DIB	Device Interface Board
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processing
EFDR	Extended Frequency-Directed Run-length
FDR	Frequency Directed Run-length
FFD	First Fit Decreasing
FIFO	First In First Out
FSM	Finite State Machine
GA	Genetic Algorithm
HDL	Hardware Description Language
IC	Integrated Circuit
ILP	Integer Linear Programming
IP	Intellectual Property
JTAG	Joint Test Access Group
MTC	Minimum Transition Count

PCB	Printed Circuit Board
RAM	Random Accesses Memory
RISC	Reduced Instruction Set Computer
RTL	Register-Transfer Level
SA	Simulated Annealing
SC	Selective Coding
SOC	System-On-Chip
TAM	Test Access Mechanism
TAP	Test Access Port
TAT	Test Application Time
TDC	Test Data Compression
TDCE	Test Data Compression Environment
TDI	Test Data In
TDO	Test Data Out
UDL	User Defined Logic
VIHC	Variable-length Input Huffman Coding
WBR	Wrapper Boundary Registrar
WBY	Wrapper Bypass Registrar
WIP	Wrapper Interface Port
WIR	Wrapper Instruction Registrar
WPI	Wrapper Parallel Input
WPO	Wrapper Parallel Output
WSC	Wrapper Scan Chain
WSI	Wrapper Serial Input
WSO	Wrapper Serial Output

Chapter 1

Introduction

The reliability of electronic systems is no longer a concern to the military, aerospace and banking industries, where failure consequences may have catastrophic impact. Reliability and testing techniques have become of increasing interest to all other applications, such as computers, telecommunications, consumer products, and automotive industry, because of the following factors : (1) at present, electronic systems are becoming ubiquitous in the workplace and are now being used in harsher environments; (2) because of the proliferation of their use, users of electronic systems are not necessarily experienced people and may misuse the machines inadvertently; and (3) the continuous decrease in technology feature size, accompanied by an increase in system complexity and speed, has resulted in newer failure modes.

A key requirement for obtaining reliable electronic system is the ability to determine that the systems are error free. The majority of the hardware used today consists of digital circuits. A circuit must be tested to guarantee that it is working and continues to work according to specifications. Such testing detects failures due to the manufacturing defects. It can also detect many field failures due to aging, environmental changes, power supply fluctuations, and so on. Test pattern generation is a complex problem. Often, simulation patterns developed for design verification are augmented with patterns that are generated manually or by an automatic test pattern generator (ATPG) to obtain a complete test set, capable of detecting all faults in the circuit. This test also verifies the functionality of its logic.

The patterns are then applied to the circuit using automatic test equipment (ATE). Because of the complexity of testing processes, a design approach aimed at making digital circuits more easily testable has been formulated. This approach to design is known as *design for test* (DFT) or *design for testability*.

As a result of the continuous decrease in the minimum feature size of transistors, both device density and design complexity have steadily increased. The shift toward submicron technologies has allowed integrated-circuit(IC) designers to increase the complexity of their designs to the extent that an entire system can now be implemented on a chip. This new paradigm of *system on a chip* (SOC) has changed the approach to design and testing. To increase the design productivity, and hence to decrease time-to-market, the reuse of previously designed modules is becoming common practice in SOC design. This is known as *core-based-design.* The reuse approach is not limited to in-house design, but is extended to modules that have been designed by others. Such modules are referred to as *embedded cores.*

An integrated circuit core is a predesigned, preverified silicon circuit block, usually containing at least few thousand (5000) gates, that can be used in building a large or more complex application on a semiconductor chip. A core may be *hard, firm* or *soft.*

Hard cores are optimized for area and performance and they are mapped into specific technology and possibly a specific foundry. They are ready to be dropped into a system. They are provided as layout files that can't be modified by the user. *Firm cores* are usually provided as technology-dependent netlists using library cells whose size, aspect ratio, and pin location can be changed to meet the customer's needs. Soft core consists of a synthesizable *hardware description language* (HDL) description that can be retargeted to different semiconductor processes.

Major problem in the core based system design is the adoption of efficient test and diagnostic strategies. Testing core-based System-on-Chips poses two-fold challenge [13], namely core-level testing and chip-level testing. Core-level testing involves making each core testable inserting the necessary *Design for Testability* (DFT) structures and generating test sequences. When the cores are integrated into a SOC, chip-level testing needs to be addressed by the SOC designer. The main difficulty in chip-level testing is the problem of justifying precomputed test sequences of a core embedded deep in the design from the

chip inputs, and propagating the test responses from the core outputs to the chip outputs. As the cores are deeply embedded in SOC, special access mechanisms are required to test them at system level, known as *Test Access Mechanisms* (TAMs). The efficiency of a TAM depends on to what extent it can reduce the testing time, which is the time to test all cores and interconnects between the cores in a SOC.

The cost of Automated Test Equipment (ATE) grows significantly with the increase in operating frequency, channel capacity and memory requirements of the test data [1]. Reduction in test data volume not only reduces ATE memory requirements but also lowers the testing time as much less amount of data would need to be transferred from ATE to the chip. There exist two potential solutions to alleviate the raising costs of ATE. The first solution is Built-In-Self-Test (BIST) [19] incorporated into the System-on-Chip. However, the application of BIST is limited; first due to pseudo-random resistant faults which limit the fault coverage achievable by BIST and secondly and most importantly due to the fact that currently there are very few cores that include BIST features. To incorporate BIST into the existing cores would require a considerable redesign effort. The second solution is that of compressing the test data stored on the tester (ATE) and utilizing small amount of *on-chip* hardware to decompress the data before being fed into the scan chains driving the *circuit-under-test* (CUT). This method not only reduces the memory requirements on the ATE but also reduces the testing time since the reduced volume of test data can be transferred faster to the chip. Moreover, unlike BIST, this method does not require the redesign of intellectual property (IP) cores. Test Data Compression requires compression techniques that are able to simultaneously satisfy the dual objectives of *loss-less compression* of test data with *minimal decompression architecture* circuit area overhead.

One of the other most important challenges in the field of SOC testing is of increased power dissipation during test. During testing, power consumption is much higher than during normal operational mode [37]. Increased power

consumption results in higher heat dissipation which has the potential to damage the circuit-under-test thereby decreasing the manufacturing yield. The situation is further worsened by the fact that several test scheduling techniques attempt to reduce testing time by testing several cores simultaneously [9].

Contribution of the thesis

In the above background, we have addressed some of the important problems related to SOC testing. These are as follows.

1. Designing a test access mechanism by partitioning the test lines available and schedule the cores for testing.
2. An integrated approach to reduce the overall testing time taking into consideration the core and interconnect testing. We have also made a trade-off between the power consumption and test time.
3. Test data compression to have improved compression ratio, reduced test application time with some extra added hardware.

Organization of the thesis

The thesis is organized as follows.

Chapter 2 presents a detailed survey of the works carried out in the field of SOC testing.

Chapter 3 presents a simulated annealing based approach for the partitioning of the test lines and assignment of cores to one of the partitions ensuring reduced test time. It shows improved results compared to many of the contemporary works.

Chapter 4 presents a genetic algorithm based approach to solve the combined problem of minimizing the test time and test power in SOC testing. The test time minimization not only addresses the core test time reduction, but also takes care of the reduction in the time required to test the interconnects.

Chapter 5 presents another core test scheduling solution that does not partition the test lines. It rather views the problem as a two-dimensional rectangle-packing one. It also takes into consideration the power constraint, in the sense that the overall power required at any point of time during testing should not exceed the specified power limit. The precedence constraints between the cores to be tested have also been addressed.

Chapter 6 presents a test data compression strategy that decomposes an input file into two to achieve higher compression ratio and reduced test application time, though at the cost of slightly higher area overhead.

Finally, Chapter 7 draws the conclusion and enumerates the scopes for future works.

Chapter 2

Literature Survey

Continuous advancement in semiconductor manufacturing technology has facilitated the integration of tens or even hundreds of millions of transistors onto silicon die [1]. The need for faster and smaller products has driven the semiconductor technology recently to introduce a new generation of complex chips. Chips that allow the integration of a wide range of complex functions, which used to comprise a system, into a single die, called System-On-Chip(SOC).

System-level integration is evolving as a new paradigm in system design, allowing an entire system to be built on a single chip, using pre-designed functional blocks called cores. While SOC is proving to be very useful in meeting aggressive time-to-market, performance, and cost requirements of today's electronic products, testing such core-based SOCs poses a two-fold challenge [2]. Core-level testing involves making each core testable, inserting the necessary design for testability (DFT) structures and generating test sequences. Typically, for hard and firm cores, testability is ensured, and pre-computed test sets are provided by the core provider. For soft cores, testability can be addressed and test sets be generated by the user. When the cores are integrated into an SOC, chip-level testing needs to be addressed by the SOC designer. The main difficulty in chip-level testing is the problem of justifying pre-computed test sequences of a core, embedded deep in the design from the chip inputs, and propagating the test responses from the core outputs to the chip outputs, that is, the cores integrated in an SOC need to be tested after the manufacturing process of the device [3]. So testing SOC is more complex than conventional board test.

Today's embedded cores incorporated into system-chips cover a very wide range of functions, while utilizing an unprecedented range of technologies,

from logic to DRAM to analog [4]. They often come in hierarchical compositions. For instance, a complex core may embed one or more simple cores. An example system-chip design is shown in Fig. 2.1, where different cores are shown. These cores typically come in a wide range of hardware description levels. They spread from fully optimized layouts in GDSII format to widely flexible RTL codes. Embedded cores are categorized into three major types based on their hardware description level: *soft*, *firm* and *hard* [5]. Each type of core has different modeling and test requirements. The three types of cores offer trade-off. *Soft* cores leave much of the implementation to the designer, but are flexible and process-independent. *Hard* cores have been optimized for predictable performance, but lack flexibility. *Firm* cores offer a compromise between the two.

The emerging process of plug-and-play with embedded cores from diverse sources faces numerous challenges in the areas of system-on-chip design, integration, and test.

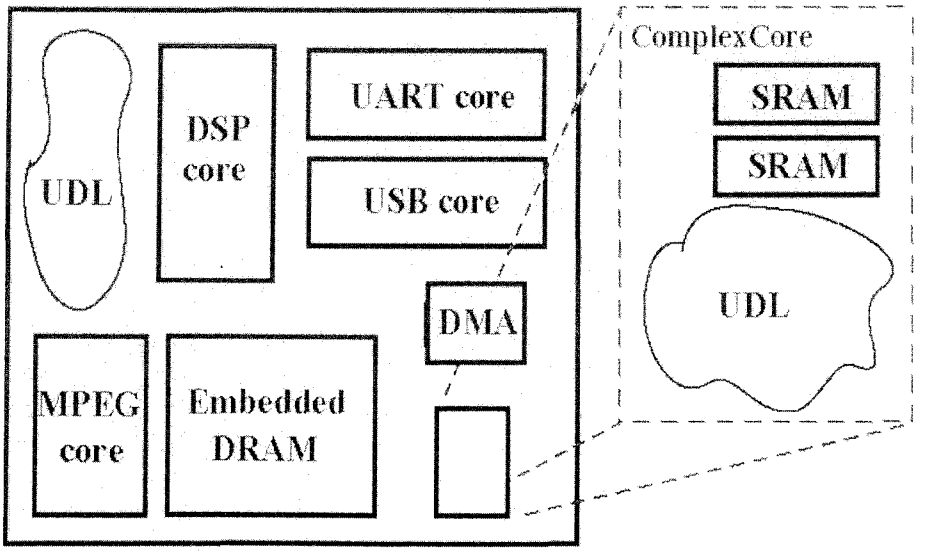


Figure 2.1: System-Chip consisting of hierarchical cores and User Defined Logic (UDL)

2.1 System Chip Test Challenges

In this section, the main challenges of testing system chips are analyzed and compared to the traditional chip test approach. Even though the design process in core-based system chips is conceptually analogous to the one used in traditional chip design, the manufacturing test processes in both cases are quite different [3]. In the traditional system-on-board approach, chip manufacturing and testing are performed by the component provider, prior to PCB assembly and test done by the system integrator, as shown in Fig. 2.2(a). On the other hand, in a core-based system chip, the reusable cores are first designed individually by the component provider. Next, the system integrator designs the *User Defined Logic (UDL)* and assembles the pre-designed cores. Only then, the manufacturing and test steps are performed. This is done for the whole system chip, as shown in Fig. 2.2(b).

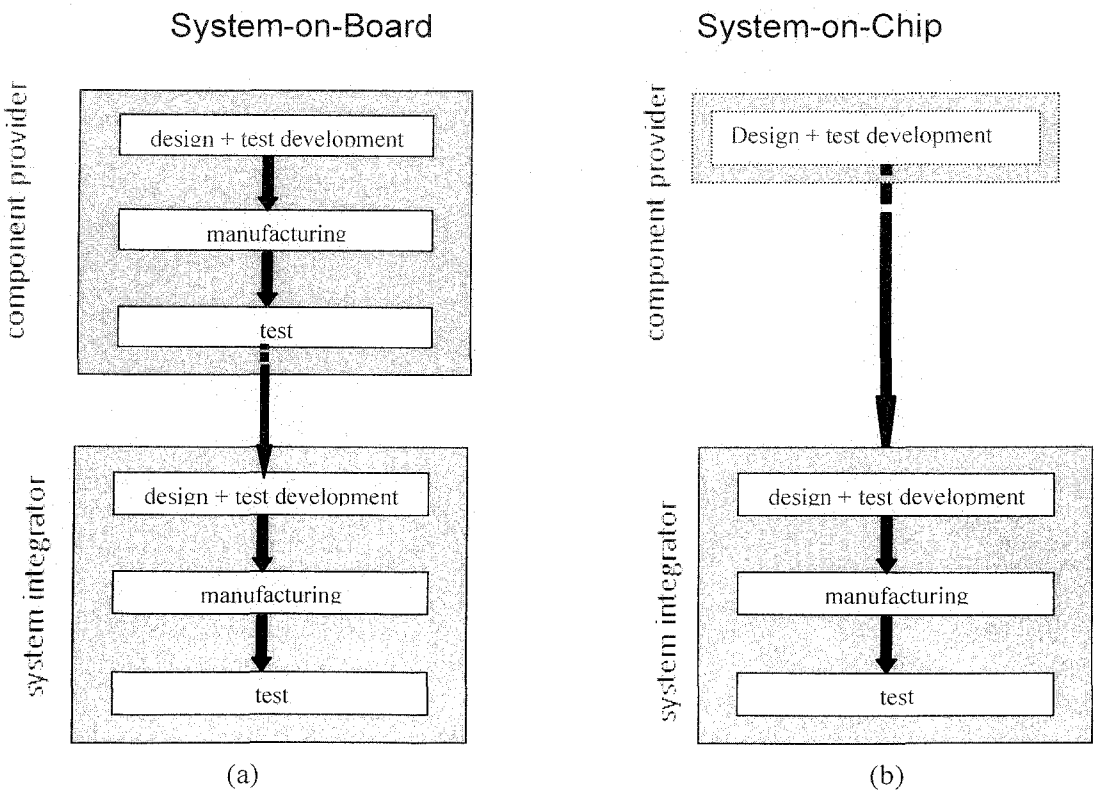


Figure2.2: System-on-Board (a) vs. System-on-Chip (b) trajectory.

2.1.1 Core Level Test

A core is typically the hardware description of today's standard ICs, e.g., DSP, RISC processor, DRAM core, etc. Even though a given core is not tested individually as in standard ICs, and instead is tested as a part of the overall system chip by the system integrator, the preparation of a core internal test, i.e. the test development, is often done by the core provider, as in Fig. 2.2(b). Because, the system integrator in most cases, except for soft cores, has very limited knowledge about the structural content of the adopted core, and hence deals with it as a black box. Therefore, he/she cannot develop the necessary test for it. That is why the core vendor has to provide the model, the DFT (e.g., scan based DFT or built-in self test (BIST) based DFT) [9, 10, 11] structures and the corresponding test vectors. This is especially true if a core is a hard one or is an encrypted Intellectual Property (IP) block. This necessitates that the core provider develops the core test, i.e., the DFT structures and the corresponding test patterns, and delivers it with the core. Another function of core provider is to determine the internal test requirements of the core without knowing the system chip environment and possibly even the target process. For instance, which test method needs to be adopted (e.g., BIST, scan, IDDQ, functional test), what type of fault(s) (e.g., static, dynamic, parametric) to target and what level of fault coverage is desired. In the traditional approach, the overall chip test method and the desired fault coverage are predetermined. Hence, the designer incorporates the requirements during test development. But in system chips, a core provider often does not have enough information about the target applications of his component and their quality requirements. Hence, the provided quality level might or might not be adequate. If the coverage is too low, the quality level of the system chip is put to a risk, and if it is too high, the test cost may become prohibitive (e.g., test time, performance, area, power). Furthermore, different processes have different defect densities and distributions.

The core internal test developed by a core provider needs to be adequately described, ported and should be ready for plug-and-play, i.e., for interoperability, with the system chip test. For an internal test to accompany its corresponding core and be interoperable, it needs to be described in a commonly accepted, i.e., standard, format. Such a standard format has been proposed in the IEEE P1500 [48] and is referred to as standardization of a Core Test description Language (CTL) [49].

2.1.2 Test Access

Another key difference between the traditional approaches and the ones for system chip is the accessibility to component peripheries, i.e., accessing primary input/outputs of chips and cores, respectively. With a system-on-board, direct physical access to chip peripheries, i.e., pins, is typically available for probing during manufacturing test; whereas for cores, which are often deeply embedded in a system chip, direct physical access to its peripheries is not available by default, hence, an electronic access mechanism is needed. This access mechanism requires additional logic, such as a *wrapper* around the core and wiring, such as a test access mechanism to connect core peripheries to the test sources and sinks, defined in the next section. The wrapper performs switching between normal mode and the test mode and the wiring is meant to connect the wrapper surrounding the core to the test source and sink. The wrapper can also be utilized for core isolation. Typically, a core needs to be isolated from its surroundings in certain test modes. Core isolation is often required on the input side, the output side, or both. If it is on the input side, it can put the core into a safe-state by protecting the core under test from external interference (from preceding cores or UDL). On the output side, it protects the inputs of the superseding blocks (cores or UDL) from undesired values (e.g., random patterns applied to tri-state buffers creating bus conflicts).

2.1.3 System Chip Level Test

One of the major challenges in the system chip realization process is the integration and coordination of the on-chip test and diagnosis capabilities. Compared to the conventional scheme, the system chip test requirements are far more complex than the PCB assembly test, which for instance in digital chips consists of interconnect and pin toggling tests. The system chip test is a single composite test. This test is comprised of the individual tests of each core, the UDL test, and the test of their interconnects. As discussed earlier, each individual core or UDL test may involve surrounding components. Certain peripheral constraints (e.g., safe mode, low power mode, bypass mode) are often required. This necessitates access and isolation modes. In addition to the test integration and interdependence issues, the system chip composite test requires adequate test scheduling. This is needed to meet a number of chip-level requirements, such as total test time, power dissipation [6], area overhead, etc. [7]. Also, test scheduling is necessary to run intra-core and inter-core tests [8] in certain order not to impact the initialization and final contents of individual cores. With the above scheduling constraints the schedule of the composite system chip test is created.

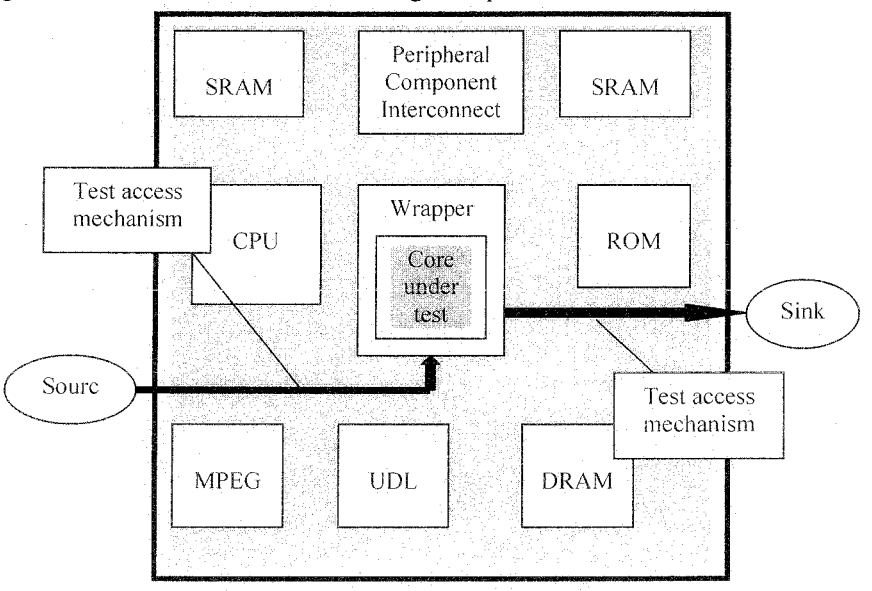
In addition to the above differences between testing traditional chips and system chips, we have to note that system chips do also have the typical testing challenges of the very deep-submicron chips, such as defect/fault coverage, overall test cost, and time-to-market.

Driven by the above challenges a vast body of research has endeavored to provide a better understanding of this research area. Numerous test strategies and algorithms in SOC test architecture design and optimization, test scheduling and test resource partitioning have been proposed in order to reduce test cost of the SOC. Iyenger *et al.* [12] presented an overview of modular SOC test planning techniques that addresses the problems of test architecture design and optimization and constrained test scheduling algorithm.

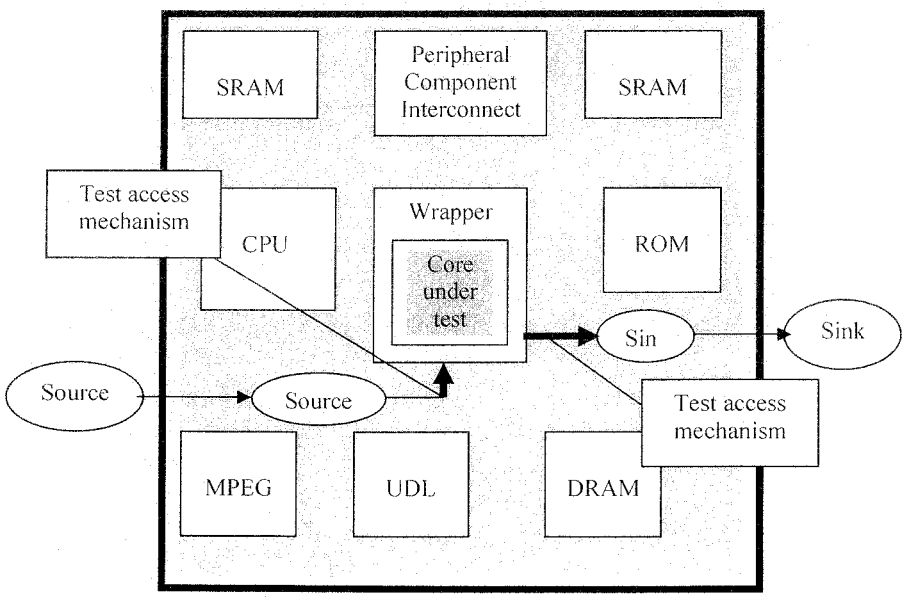
2.2 SOC Test Access

Conceptual Architecture for Core Test

Zorian *et al.* [3] proposed a conceptual infrastructure for SOC test, as shown in Fig. 2.3. It consists of the following components.



(a)



(b)

Figure 2.3. This architecture applies to test patterns generated by automatic test equipment (ATE) from (a) outside the chip. It also applies to built-in self-test (BIST) configurations, which have source and sink (b) inside the chip.

- *Test pattern source and sink.* The source generates the test stimuli for the embedded core, and the sink compares the response(s) to the expected values.
- *Test access mechanism.* It transports test patterns. It provides on-chip transport of test stimuli from a test pattern source to the core under test. It also transports test responses from the core under test to a test pattern sink.
- *Core test wrapper.* The wrapper forms the interface between the embedded core and its environment. It connects the terminals of the embedded core to the rest of the IC and to the test access mechanism.

All three elements can be implemented in various ways, such that a whole palette of possible approaches to testing embedded cores emerges. We review the various alternatives and classify the current approaches.

Test Pattern Source and Sink

Designers can implement test pattern sources and sinks either off-chip, using external automatic test equipment (ATE), or on-chip, using built-in-self-test (BIST), or a combination of both or even an embedded microprocessor [19]. Source and sink need not be of the same type; that is, an embedded core's source can be off-chip, while its sink is on-chip. Three factors influence the choice of a certain type of source or sink:

- The type of circuitry in the core,
- The predefined tests that come with the core,
- Quality, test time, and cost considerations.

Core circuitry

Today, system chips use three main types of circuitry: digital logic, memory, and analog. Simple cores consist of only one type; complex cores combine multiple simple cores, possibly of different circuitry types.

These three types of circuitry exhibit different defect behavior and require different tests [13]. The various tests require different types of sources to

generate the stimuli and sinks to compare the responses. Typically, distinct ATE systems as well as BIST schemes are used for logic, memory, and analog circuitry. System chips, which often incorporate all three types of circuitry into one IC, are encouraging ATE vendors and BIST providers to integrate their traditionally separate solutions for logic, memory, and analog into combined product offerings. Hence, instead of using a separate ATE for the logic part of the system chip, a second ATE for the embedded memory and a third for the analog circuitry, ATE vendors are offering "super" ATE systems to combine the test capabilities of all three types.

Core tests

The variety of core tests is much larger than the three circuitry types. Tests are classified not only by the type of circuit they test, but also by the measurements they require (voltage or current), by the way they are generated (based on the IC's function or structure), by the amount of core-internal adaptation they require (scan or test points), and so on.

Examples of current measurement tests are IDDQ and DDT [14], which measure quiescent and transient currents. Current can be measured both by sinks on-chip (current monitors) [16] as well as off-chip (current monitors in an ATE system).

Not all test patterns can be generated on-chip in a cost-effective manner. The test patterns of cores that come with function tests and/or ATPG-generated tests are often irregular in structure. It is difficult to generate such irregular deterministic test patterns on-chip at acceptable area costs.

Quality and cost

Off-chip sources and sinks often require large capital investment and, because they are built using yesterday's technology, suffer from various problems. Faster ICs require increasing accuracy to detect timing signals at the IC pins. Although tester accuracy has improved by 12% annually, IC speeds have improved by about 30% per year. This growing gap reduces off-chip testers'

ability to properly identify bad chips, leading to yield losses and cost increases. Furthermore, it is becoming increasingly difficult for off-chip ATE to keep up with the very high frequencies needed to sufficiently test performance-related defects in today's ICs.

In addition to these test-quality-related issues, the increased pin count and the mixing of diverse technologies in system chips will cause ATE costs - for running the tests as well as the equipment itself - to rise toward $20 million, according to the *1997 SIA National Technology Roadmap for Semiconductors [16]*. These problems with the quality and cost of external ATE will only become worse for high-speed, high-density, and mixed-technology system chips, rendering external ATE unacceptably inaccurate and prohibitively expensive.

Test Access Mechanism

A test access mechanism takes care of on-chip test pattern support. It can be used to transport

- test stimuli from the test pattern source to the core under test, and
- test responses from the core under test to the test pattern sink.

By definition, the test access mechanism is implemented on-chip. Although one core often uses the same type of test access mechanism for transporting both stimulus and response, such consistency is not required and various combinations may coexist.

Designing a test access mechanism involves making trade-offs between the mechanism's transport capacity (bandwidth) and its test application cost. Bandwidth is limited by the bandwidth of source and sink and the silicon area that we want to spend on the test access mechanism itself. A wider test access mechanism provides more bandwidth, but consumes more wiring area. For example, if the test pattern source is an external ATE, it does not make much sense to provide a mechanism wider than there are IC pins available to connect it to. In this case, the IC pins are the bandwidth bottleneck, and a wide

mechanism costs more silicon area without adding to that bandwidth.

Test time is a result of the test data volume of the individual cores and the bandwidth of the test access mechanism. How expensive test time is per unit of time depends on the type of source and sink. There is a wide range of external ATE with similarly wide-ranging associated test costs, and these again differ from the cost of test application time for BIST.

There are several options for implementing a core test access mechanism, which can

- reuse existing functionality to transport test patterns or be formed by dedicated test access hardware;
- go through other modules on the IC - including other cores - or pass around those other modules;
- provide access for only one core or for multiple cores; or
- be a plain signal transport medium or may contain certain intelligent-test-control functions.

Core Test Wrapper

The core test wrapper is the interface between the embedded core and its system chip environment. It connects the core terminals both to the rest of the IC as well as to the test access mechanism. By definition, the core test wrapper is implemented on-chip and should have the following mandatory modes:

- *Normal operation* (non test). In this mode, the core is connected to its system IC environment, and the wrapper is transparent.
- *Core-internal test*. The test access mechanism is connected to the core such that a source can apply stimuli at the core's inputs, and a sink can observe responses at the core's outputs.
- *Core-external test*. The test access mechanism is connected to the interconnect wiring and logic such that a source can apply stimuli at the core's outputs, and a sink can observe responses at the core's inputs.

Apart from these mandatory modes, a core test wrapper can also have several optional modes. For example, a wrapper can include a detach mode to disconnect the core from its system chip environment and the test access mechanism.

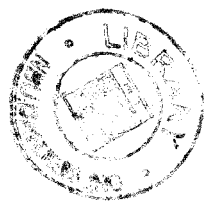
Depending on the test access mechanism's implementation, some of these modes can coincide. For example, if the test access mechanism uses existing functionality, normal and core test modes can coincide.

Predesigned cores have their own internal clock distribution system. Different cores have different clock propagation delays, which might result in clock skew for intercore communication. The system-IC designer should take care of this clock skew in the functional communication between cores. However, clock skew might also corrupt the data transfer over the test access mechanism, especially if multiple cores share this mechanism. The core test wrapper is the best place in the test access paths between cores to implement clock skew prevention.

The *test collar* [17] and the *test shell* [18] are examples of core test wrappers. Details of wrapper design have been given in Section 2.6.

2.3 Testing Strategies

The test access mechanism takes care of on-chip test pattern transport. It can be used (1) to transport test stimuli from the test pattern source to the core-under-test, and (2) to transport test responses from the core-under-test to the test pattern sink [6]. The test access mechanism is, by definition, implemented on-chip. In this section an overview of the various testing strategies are discussed. The embedded cores can be accessed from the chip's I/O pins in four different ways [3, 20]: (i) direct parallel access via chip inputs; (ii) serial access and core isolation through a boundary scan-like architecture (called isolation ring access mechanism); (iii) functional access through functional busses or transparency



of embedded cores; and (iv) access through a combination of core wrappers and dedicated test busses.

2.3.1 Direct Access

An obvious mechanism to make embedded cores testable from the IC pins makes the core-under-test directly and parallely accessible from the IC pins. In this mode, no previously existent connection is reused. A direct connection between the CUT interface and the system interface is established.

This approach is commonly practiced for embedded memories, but also many block based ASICs use this test access strategy [21]. Additional wires are connected to the core's terminals and multiplexed onto existing IC pins. The multiplexer control is coded as a dedicated test mode. In case of multiple embedded cores, this leads to an equal number of test modes. Fig. 2.4 gives a schematic view of the approach.

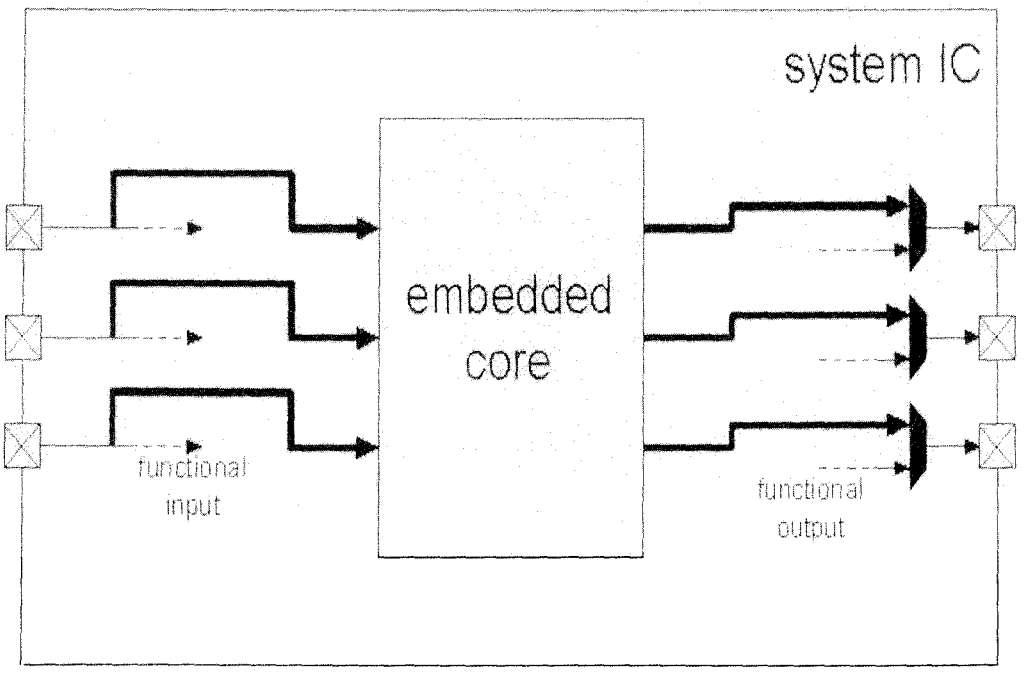


Figure 2.4: Multiplexed direct parallel access.

The advantage of this approach is in its simplicity and in testing the core as if it is the only circuit on the IC *i.e.* the embedded core can be tested as if it were a stand-alone device. This also simplifies silicon debug and diagnosis. The disadvantage of this technique is that it is not very scalable. If a system IC contains many embedded cores, this approach leads to high area costs for connecting and multiplexing all core terminals to IC pins, and the control circuitry for these multiplexers will grow more complex. The bit width of this connection is assumed to be as large as the number of bits that need to be propagated to the interface. This implies an overhead of n in the number of pins, for n the number of test signals being propagated. The area overhead is proportional to n and to the routing distance from CUT to the system interface.
The test access mechanism as proposed by Varma & Bhatia [22, 23] also accesses the core-under-test directly from the IC pins, but provides the option to share one access mechanism with multiple cores. Such a shared access structure is called a *test bus*. Per IC, there are one or more test buses of varying width. Multiple cores can be connected to one tri-stateable test bus; all cores on the same bus are tri-stated, except for the core-under-test. This approach allows the system IC integrator to choose his optimal mix of number and width of test buses and number of cores per test bus and hence provides the opportunity to trade off silicon area for test time. Although presented as a dedicated test access mechanism, it is relatively easy in this approach to reuse existing on-chip buses as test bus implementation, as is done in ARM's AMBA system (Fig. 2.5) [24]. The main disadvantage of this approach is that per test bus, only one core can be connected at a time. This limits the possibilities to test multiple cores at the same time, which is sometimes desired (test time reduction through test scheduling, IDDQ testing) or even required (testing interconnect wiring and glue logic in between cores).

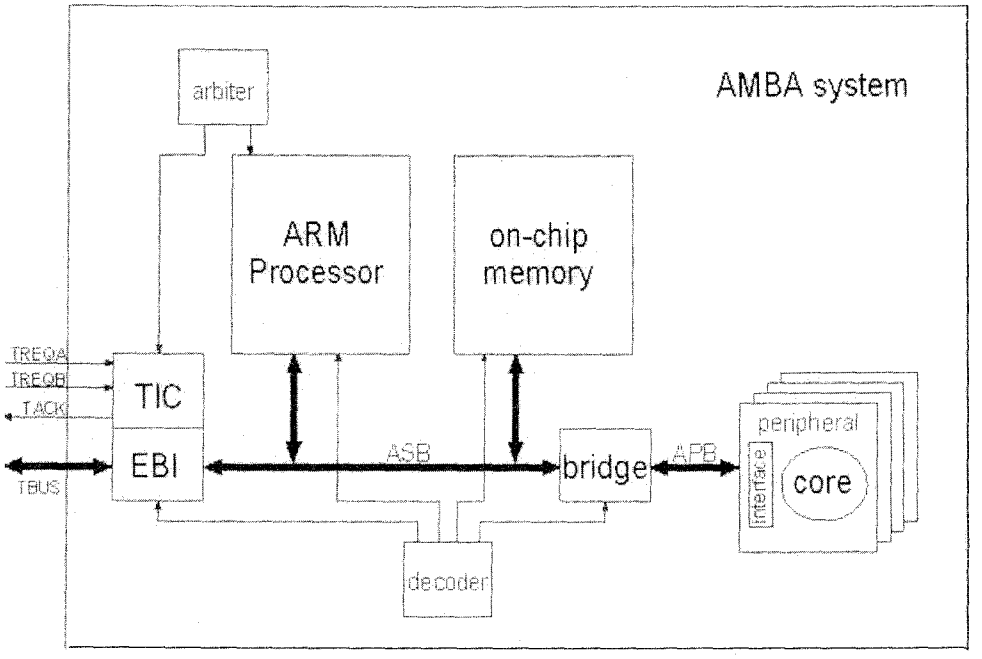


Figure 2.5:ARM's AMBA system

2.3.2 Isolation ring access (Use of Boundary-scan)

The Boundary Scan Test (BST) approach was initially a DFT technique for printed circuit boards (PCBs). This technique requires that ICs include extra hardware to facilitate communication between the board and the various ICs on which they are mounted during testing. Also with the present trend of a system-on-chip, the ICs themselves consist of several embedded complex and diverse modules and start to beg for a DFT technique that facilitates their testing. In 1990, the methodology became IEEE/ANSI Standard 1149.1 [25].

The use of Boundary Scan has been extended beyond boards to circuits [26]. For example, it is useful for internal testing and to run BIST. In addition to testing, Boundary Scan was adopted in debugging and diagnosis using in-circuit emulation protocols [27].

In this approach, a serial scan chain is laid around the terminals of the core. Through this scan chain both the core itself, as well as the interconnect wiring and logic in between cores can be tested. The obvious benefit of this approach is that an existing test standard is reused. Many ICs are equipped with BST, often augmented with multiple private instructions which represent all kinds of test, debug, and emulation modes. If these ICs become cores in large system ICs, it seems logical to reuse the test infrastructure as provided by BST. A technical problem is that a scheme has to be developed to control multiple TAP controllers on one IC; various solution approaches have been proposed [28, 29]. A more serious drawback of this approach is that the BST standard defines a combined test control and test data access path of only one bit wide via TDI and TDO. Therefore, the BST approach does not allow to make a trade-off between bandwidth and test time. For small test pattern sets, the small bandwidth does not pose a problem. This is, for example, the case for board-level interconnect testing, for which BST was developed, or for cores equipped with BIST. However, for large scan-testable cores, providing many scan test patterns via only one serial access wire results in prohibitively long test times.

Touba & Pouya [30] have presented a variation on the Boundary Scan Test approach by using a partial boundary scan ring around the core. The ATPG techniques are used to find out which of a given set of stimuli can be justified from the (assumed) user-defined logic (UDL) in front of the core, such that the corresponding core inputs can be excluded from the partial boundary scan ring. In [31] the same authors even allow modification of the UDL such that the required test patterns can be generated. In addition to its function as test data and control signal transportation medium, the Test Access Mechanism presented in Zorian [7] has an embedded intelligence to execute the system chip test in a predetermined schedule. The test sequencing of all the cores is embedded in a network of distributed modular controllers, which is an integral part of the Test Access Mechanism. This network called, the Universal BIST Scheduler, is either customized to execute a predetermined BIST schedule for

manufacturing test or is programmed on the fly through the JTAG port to execute the test of individual cores during silicon debug and diagnosis.

2.3.3 Functional Access

The third technique for peripheral access is based on using the surrounding logic to propagate and justify the necessary test patterns [33]. This can be either based on existing normal mode logic put in transparency for core test purposes [32], or be based on using existing DFT mode, such as scan chains in the surrounding logic. Even though, the cost of extra hardware is very low, this approach becomes complicated if the surrounding logic has deep functional paths and requires sophisticated protocols for test pattern translation.

The timing impact of a peripheral access mechanism is critical. The path delay created by peripheral access logic and the skews introduced in the access path need to meet the requirements of core test timing constraints.

The ring based technique remains the most realistic option for peripheral access. The ring approach provides access to run internal BIST and internal scan for each core. It also helps testing the surrounding logic using the ring registers.

Ghosh *et. al.* [34] assumed that every core has a transparent mode in which data can be propagated from inputs to outputs in a fixed number of cycles. Their approach is based on the concept of finding functional test path (F-path) [35] through test control /data flow extraction. Although the proposed method successfully lowered the test area and delay overheads, it requires functional description of each core to make it transparent, which in most cases is not available or it is hard to extract. In addition, because test data cannot be pipelined through a core in this method, the potentially large transparency latency of each core (number of cycles needed to propagate test data from inputs to outputs of the core) may incur a relatively large test application time. To address the above issue, in [2] the same authors extended their approach by describing a trade-off between the silicon area consumed by the design-for-

transparency hardware and propagation latency. They suggested that the core providers offer a catalogue of area/latency versions for their cores, from which the system integrator can select one in order to meet the test access needs of its neighboring cores. By associating a user-defined cost function with the transparent test paths in the core connectivity graph, the system integrator is able to select the version of cores that achieve an optimized test solution at the system level. This approach, however, requires a large design effort at the core provider side. Another limitation of [34,2] stems from the difficulty in handling other popular DFT schemes, such as scan or BIST, in the SOC.

Solutions from [34,2] require that all the I/Os of a core be simultaneously, though indirectly, controllable/observable. Ravi *et. al.* [36] showed that this complete controllability and observability is unnecessary and proposed to provide them on an "as needed basis". Their approach allows for complex core transparency modes and is able to transfer test data to and from the CUT in a more aggressive and effective manner.

In [37], Nourani and Papachristou presented a similar technique to core transparency, in which cores are equipped with a bypass mode using multiplexers and registers. They can model the system as a directed weighted graph, in which the accessibility of the core input and output ports is solved as a shortest path problem. While this approach eases the problem of finding paths from the SOC inputs to the CUT, it requires packetisation of test data (to match the bit width of input and output ports), and the help of serialization/de-serialisation bit-matching circuits.

2.3.4 Test Rail

A single test rail provides access to one or more cores and an IC may contain one or more test rails of varying width. Each core is wrapped in a test shell. Per core, there is also a test rail bypass mode. This allows the core user to test each core sequentially or multiple cores in parallel. A test rail provides flexible and scalable test access mechanism and helps to trade-off between test time and area overhead by varying test data width. Fig. 2.6 shows typical test rail

architecture. In the figure there are 5 cores which are embedded in their wrappers and test buses of total width 16. Total bus width is divided into three buses of widths 5, 5 and 6 respectively. Cores 1 and 2 are assigned to first bus, cores 3 and 4 are assigned to second bus and core 5 is assigned to third bus. The testing time for a SOC is determined to a large extent by the design of test wrappers and the TAM.

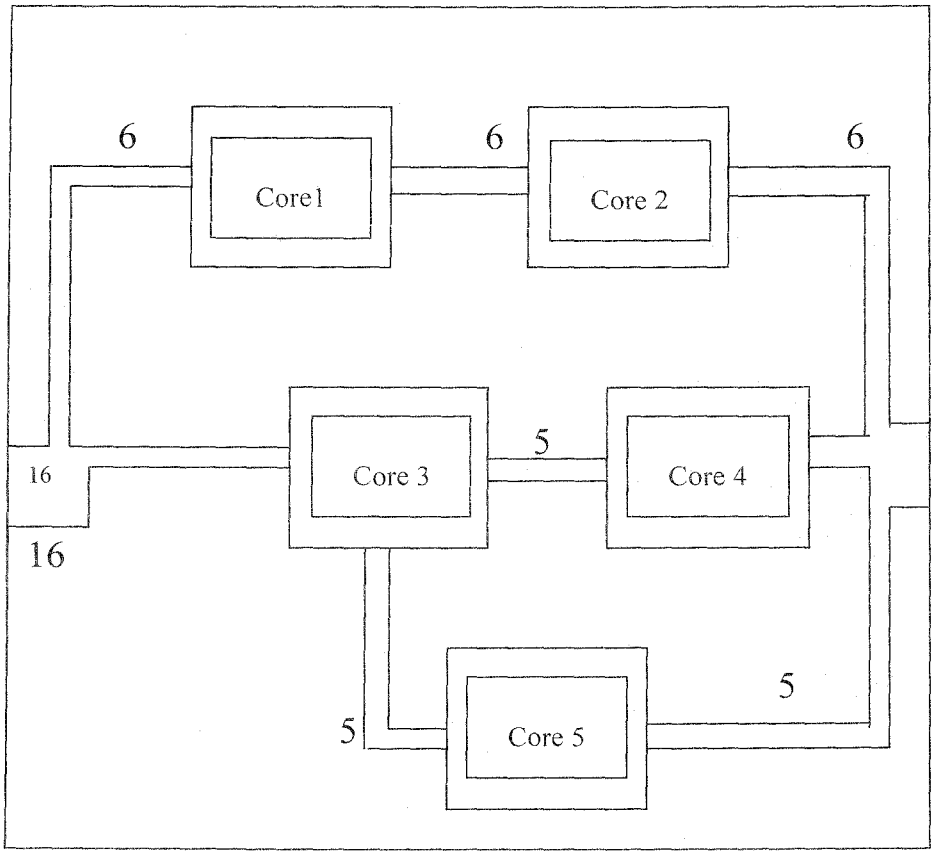


Figure 2.6: Test Rail example

Wrapper/TAM co-optimization is therefore necessary for minimizing SoC testing time. Iyengar *et al.* in [34, 35] proposed an exact technique for co-optimization based on a combination of integer linear programming (ILP) and exhaustive enumeration. However, this approach is computationally expensive for large SOCs, and it is limited to fixed-width test buses. Rectangle packing, also referred to as two-dimensional packing [5] for wrapper/TAM co-

optimization decreases the testing time by reducing the mismatch between a cores test data needs and the width of the TAM to which it is assigned. One of the more recent works on the reduction of test cost for System-on-Chip uses virtual TAMs to match high speed ATE channels to slower scan chains [4]. This method also presented a new TAM optimization framework based on Lagrange multipliers.

2.4 Test Scheduling and Test Architecture Optimization

Test access mechanisms (TAMs) and test wrappers have been proposed as important components of SOC test access architecture [38]. TAMs deliver pre-computed test sequences to cores on the SOC, while test wrappers translate these test sequences into patterns that can be applied directly to the cores. Test wrapper and TAM design are of critical importance in SOC system integration since these directly impact the vector memory depth required on the ATE, as well as testing time, and thereby affect test cost. A TAM and wrapper design that minimizes the idle time spent by TAMs and wrappers during test directly reduces the number of don't-care bits in vectors stored on the tester, thereby reducing vector memory depth. The design of efficient test access architectures has become an important focus of research in core test integration [40, 41, 42, 43, 44, 45, 46, 47]. This is especially timely and relevant, since the proposed IEEE P1500 standard provides a lot of freedom in optimizing its standardized, but scalable wrapper, and leaves TAM optimization entirely to the system integrator [48, 49].

The general problem of SOC test integration includes the design of TAM architectures, optimization of core wrappers, and test scheduling. The goal is to minimize the testing time, area costs, and power consumption during testing. The wrapper/TAM co-optimization problem addressed by Iyengar *et. al.* in [50] is as follows. Given the test set parameters for the cores on the SOC, as well as the total TAM width, determine an optimal number of TAMs for the SOC, an

optimal partition of the total TAM width among the TAMs, an optimal assignment of cores to each TAM, and an optimal wrapper design for each core, such that the overall system testing time is minimized. In order to solve this problem, authors examine a progression of three incremental problems structured in order of increasing complexity, such that they serve as stepping-stones to the more general problem of wrapper/TAM design. The first problem P_W is related to test wrapper design. The next two problems P_{AW} and P_{PAW} are related to wrapper/TAM co-optimization.

1. $\boldsymbol{P_W}$: Design a *wrapper* for a given core, such that (i) the core testing time is minimized, and (ii) the TAM width required for the core is minimized.
2. $\boldsymbol{P_{AW}}$: Determine (i) an *assignment* of cores to TAMs of given widths and (ii) a *wrapper* design for each core, such that SOC testing time is minimized. (Item (ii) equals P_W.)
3. $\boldsymbol{P_{PAW}}$: Determine (i) a *partition* of the total TAM width among the given number of TAMs, (ii) an *assignment* of cores to the TAMs, and (iii) a *wrapper* design for each core, such that SOC testing time is minimized. (Items (ii) and (iii) together equal P_{AW}.)

These three problems lead up to P_{NPAW}, the more general problem of wrapper/TAM co-optimization described as follows.

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

Authors [50] assume the "test bus" model for TAMs and that the TAMs on the SOC operate independently of each other; however, the cores on a single TAM are tested sequentially. This can be implemented either by (i) multiplexing all the cores assigned to a TAM, or (ii) by testing one of the cores on the TAM, while the other cores on the TAM are bypassed.

Test wrappers provide a variety of operation modes, including normal operation, core test, interconnect test, and (optional) bypass [51]. In addition, test wrappers need to be able to perform test width adaptation if the width of the TAM is not equal to the number of core terminals. The IEEE P1500 standard addresses the design of a flexible, scalable wrapper to allow modular testing [48, 49]. This wrapper is flexible and can be optimized to suit the type of TAM and test requirements for the core. A "test collar" was proposed in [23] to be used as a test wrapper for cores. However, test width adaptation and interconnect test were not addressed. The issue of efficient de-serialization of test data by the use of balanced wrapper scan chains was discussed in [44]. Balanced wrapper scan chains, consisting of chains of core I/Os and internal scan chains, are desirable because they minimize the time required to scan in test patterns from the TAM. However, no mention was made of the method to be used to arrive at a balanced assignment of core I/Os and internal scan chains to TAM lines. The TESTSHELL proposed in [51] has provisions for the IEEE P1500 required modes of operation. Furthermore, heuristics for designing balanced wrapper scan chains, based on approximation algorithms for the well-known Bin Design problem [52], were presented in [46]. However the issue of reducing the TAM width required for a wrapper was not addressed. A number of TAM designs have also been proposed which include multiplexed access [53], partial isolation rings [30], core transparency [54], dedicated test bus [55], reuse of the existing system bus [56], and a scalable bus-based architecture called TESTRAIL [51]. Bus-based TAMs, being flexible and scalable, appear to be the most promising. However, their design has largely been ad hoc and previous methods have seldom addressed the problem of minimizing testing time under TAM width constraints. While [57] presents several novel TAM architectures (i.e., multiplexing, daisy chaining and distribution), it does not directly address the problem of optimal sizing of TAMs in the SOC. In particular, only internal scan chains are considered in [57], while wrappers and functional I/Os are ignored. Moreover, the lengths of the internal scan chains

are not considered fixed, and therefore [57] does not directly address the problem of designing test architectures for hard cores.

More recently, integrated TAM design and test scheduling has been attempted in [58, 59]. However, in [58, 59], the problem of optimizing test bus widths and arbitrating contention among cores for test width was not addressed. In [59], the cost estimation for TAMs was based on the number of bridges and multiplexers used; the number of TAM wires was not taken into consideration. Furthermore, in [58] the impact of TAM widths on testing time was not included in the cost function. The relationship between testing time and TAM widths using ILP was examined in [60, 61], and TAM width optimization under power and routing constraints was studied in [62]. However, the problem of effective test width adaptation in the wrapper was not addressed. This led to an overestimation of testing time and TAM width. Iyenger *et. al.* in [50] present a new wrapper/TAM co-optimization methodology that overcomes the limitations of previous TAM design approaches that have addressed TAM optimization and wrapper design as independent problems. The new wrapper design algorithm improves upon previous approaches by minimizing the core testing time, as well as reducing the TAM width required for the core. It uses an approach based on ILP to solve the problems of determining an optimal partition of the total TAM width and determining an optimal core assignment to the TAMs. They also address a new problem, that of determining the optimal number of TAMs for an SOC. This problem gains importance with increasing SOC size.

2.4.1 Constraint-driven test scheduling

Test scheduling is the process that allocates test resources (i.e. TAM wires) to cores at different time. Primary objective of test scheduling is to minimize testing time, while addressing one or more of the following issues: resource conflicts between cores arising from the use of shared TAMs [115], precedence constraints among tests [116], and power dissipation constraints [117]. Furthermore, testing time can be decreased further through the selective use of

test pre-emption [115]. As discussed in [115,116] most problems related to test scheduling for SOCs are also NP-hard.

Test scheduling was formulated as a combinatorial optimization problem. Reordering tests to maximize defect detection early in the schedule was explored in [117]. The authors used a polynomial time algorithm to reorder tests based on the defect data as well as execution time of the tests [117]. A test scheduling technique based on the defect probabilities of the core has been reported in [118]. In [119], a heuristic algorithm based on pair wise composition of test protocol was presented.

SOCs in test mode can dissipate up to twice the amount of power they do in normal mode, since cores, that do not operate in parallel may be tested concurrently [120]. Hence power constrained test scheduling is important. In [121], a method based on approximate vertex cover of a resource constrained test compatibility graph was presented. In [122], the use of a list scheduling and tree growing algorithm for power constrained scheduling was discussed. The authors presented a greedy algorithm to overlay tests such that the power constraint is not violated. The issue of re-organizing scan chains to trade-off testing time with power consumption was investigated in [123]. The authors presented an optimal algorithm to parallelize tests under power constraints. In [115], an integrated approach for test scheduling with precedence constraints was presented. In [124], a new approach wrapper/TAM co-optimization and constraint driven test scheduling using rectangular packing was described. Flexible widths TAMs that are allowed to fork and merge were designed. Rectangle packing was used to develop test schedules that incorporate precedence and power constraints, which allowing the SOC integrator to designate group of tests as pre-emptable. The work reported in [115] was extended in [125] to address the minimization of ATE buffer reload and include multisite testing. The mapping between core I/Os and SOC pins during the test schedule was investigated in [126]. TAM design and test scheduling was modeled as 2D bin packing in which each core test is presented by a rectangle. The authors next formulated constraint driven pin mapping and test

scheduling as the chromatic number problem from the graph theory and as a dependency matrix partitioning problem. In [127], SOC test problem is solved with power constraints as a 3D bin packing problem and provided a heuristic to handle the problem. Zou et al. proposed in [81] a method using simulated annealing (SA) to handle the problem. The problem has also been addressed by Harmanani [177]. In [81], TAM lines are not partitioned; rather, a rectangle fitting problem has been solved. It utilized a special data structure called *sequence-pair*. In this formulation, a TAM line needs to be switched individually, rather than the whole bus at a time between the cores. Thus, the test controller becomes complex. It borrowed concepts from floor-planning problem. In [177], an integrated approach for wrapper design, TAM assignment and test scheduling for SOC has been reported. Xia et al. in [86] presented an algorithm for co-optimizing test scheduling and wrapper design under power constraints by using an evolutionary algorithm and the sequence pair representation. Same authors also presented an algorithm for assigning non-consecutive TAM wires to core tests. In [128], authors proposed test scheduling solution using B-tree based floor-planning and using simulated anneling techniques. Recent work on TAM optimization has focused on the use of ATEs with port scalability features [129, 90,130]. Optimization techniques have been developed to ensure that the high-data-rate tester channels are efficiently used during SOC testing [130]. The availability of dual-speed ATEs was also exploited in [129,90], where a technique was presented to match ATE channels with high data rate to core scan chain frequencies using virtual TAMs. In [130], the hardware overhead is reduced over [129] through the use of a smaller number of on-chip TAM wires; ATE channels with high data rates directly drive SOC TAM wires, without requiring frequency division hardware. Most recently test methodology concerns about the hierarchical SOCs. A hierarchical SOC is designed by integrating heterogeneous technology cores at several layers of hierarchy [1]. The ability to reuse embedded cores in a hierarchical manner implies that 'today's SOC is tomorrow's embedded cores'. Two broad design transfer model: interactive and non-interactive, are emerging

in hierarchical SOC design flow. Recently test design methodologies for hierarchical SOCs were proposed in [131,132]. The problem of designing test wrappers and TAMs of multilevel TAMs for the cores within core's design paradigm [133,134,135,136] has been addressed. Two design flows have been considered for the scenario in which "megacores" are wrapped by the core vendor prior to delivery. In an alternative scenario, the megacores can be delivered to the system integrator appropriately designs the megacore wrappers and SOC-level TAM architecture to minimize overall testing time.

2.5 Test Wrapper Design and Optimization

The IEEE P1500 wrapper as shown in Fig. 2.7, is a thin shell around a core that allows the core and its environment to be tested independently[63].

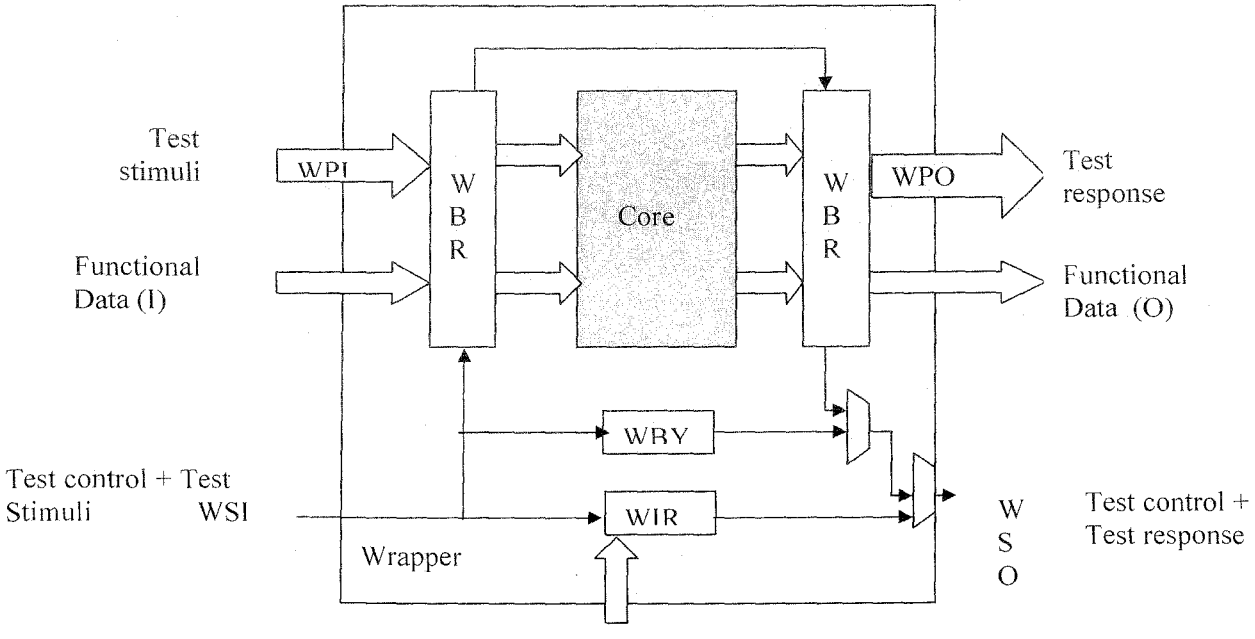


Figure 2.7: IEEE P1500 wrapper architecture [63]

The P1500 core wrapper is a shell which allows various configurations at its inputs and outputs, i.e., various operating modes. An IEEE P1500 core wrapper is illustrated in Fig. 2.7. The wrapper has as inputs the wrapper parallel input

(WPI), the wrapper serial input (WSI) and the functional inputs (I), and the wrapper interface port (WIP).

The outputs are the wrapper parallel output (WPO), the wrapper serial output (WSO) and the functional outputs (O). The core wrapper comprises two wrapper boundary registers (WBR), the wrapper bypass register (WBY) and the wrapper instruction register (WIR). The WBRs provide the interface between the test environment and the core, facilitating the test modes normal operation, core-internal test and core external test. The WBR is a register formed out of wrapper boundary scan cells. The wrapper cell assigned to the inputs are referred to as wrapper boundary input cells, while those assigned to the outputs are referred to as wrapper boundary output cells. The WIP facilitates the load of instructions corresponding to the above modes into the WIR. For each of the above modes there can be different configurations. For example, there can be a serial internal test, when the test stimuli are provided through the WSI (serial access), and the test responses are observed at the WSO; or a parallel internal test when the test stimuli are provided through the WPI (parallel access), and the test responses are observed at the WPO. A standardized, but scalable test wrapper is an integral part of the IEEE P1500 working group proposal [48]. A test wrapper is a layer of DFT logic that connects a TAM to a core for the purpose of test [64]. Test wrappers have four main modes of operation. These are (i) *Normal* operation, (ii) *Intest*: core-internal testing, (iii) *Extest*: core-external testing, i.e., interconnect test, and (iv) *Bypass* mode. Wrappers may need to perform test width adaptation when the TAM width is not equal to the number of core terminals. This will often be required in practice, since large cores typically have hundreds of core terminals, while the total TAM width is limited by the number of SOC pins. Iyenger *et. al.* in [50] address the problem of TAM design for *Intest*. A core usually contains several core I/Os as well as several internal scan chains consisting of flip-flops connected in serial within the core for the purpose of scanning test data in and out of the core. To perform test width adaptation, wrapper scan chains are constructed by connecting core I/Os and internal scan

chains in serial. The number of wrapper scan chains constructed is equal to the TAM width provided to the core; hence each wrapper scan chain is assigned to a single unique TAM line. Thus the test data width (number of core terminals) of the core is adapted to the TAM width. The problem of designing an effective width adaptation mechanism for *Intest* can be broken down into three problems [65]: (i) partitioning the set of wrapper scan chain elements (internal scan chains and wrapper cells) into several wrapper scan chains, which are equal in number to the number of TAM lines, (ii) ordering the scan elements on each wrapper chain, and (iii) providing optional bypass paths across the core. The problems of ordering scan elements on wrapper scan chains and providing bypass paths were shown to be simple in [65], while that of partitioning wrapper scan chain elements was shown to be NP-hard.

Recent research on wrapper design has stressed the need for balanced wrapper scan chains [44, 65]. *Balanced* wrapper scan chains are those that are as equal in length to each other as possible. Balanced wrapper scan chains are important because the number of clock cycles to scan in (out) a test pattern to (from) a core is a function of the length of the longest wrapper scan-in (scan-out) chain. Let s_i (s_o) be the length of the longest wrapper scan-in (scan-out) chain for a core. The time required to apply the entire test set to the core is then given by $T = (1 + \max\{s_i, s_o\}).p + \min\{s_i, s_o\}$, where p is the number of test patterns. This time T decreases as both s_i and s_o are reduced, i.e., as the wrapper scan-in (and scan-out) chains become more equal in length.

Figure 2.8 illustrates the difference between balanced and unbalanced wrapper scan chains; *Bypass* and *Extest* mechanisms are not shown. In Figure 2.8(a), wrapper scan chain 1 consists of two input cells and two output cells, while wrapper scan chain 2 consists of three internal scan chains that contain 14 flip-flops in total. This results in unbalanced wrapper scan-in/out chains and a scan-in and scan-out time per test pattern of 14 clock cycles each. On the other hand, with the same elements and TAM width, the wrapper scan chains in Figure 2.8(b) are balanced. The scan-in and scan-out time per test pattern is now 8 clock cycles. The problem of partitioning wrapper scan chain elements into

balanced wrapper scan chains was shown to be equivalent to the well-known Multiprocessor Scheduling and Bin Design problems in [65]. In this paper, the authors presented two heuristic algorithms for the Bin Design problem to solve the wrapper scan chain element partitioning problem. Given k TAM lines and sc internal scan chains, the authors assigned the scan elements to m wrapper scan chains, such that $\max\{s_i, s_o\}$ was minimized. This approach is effective if the goal is to minimize only $\max(s_i, s_o)$.

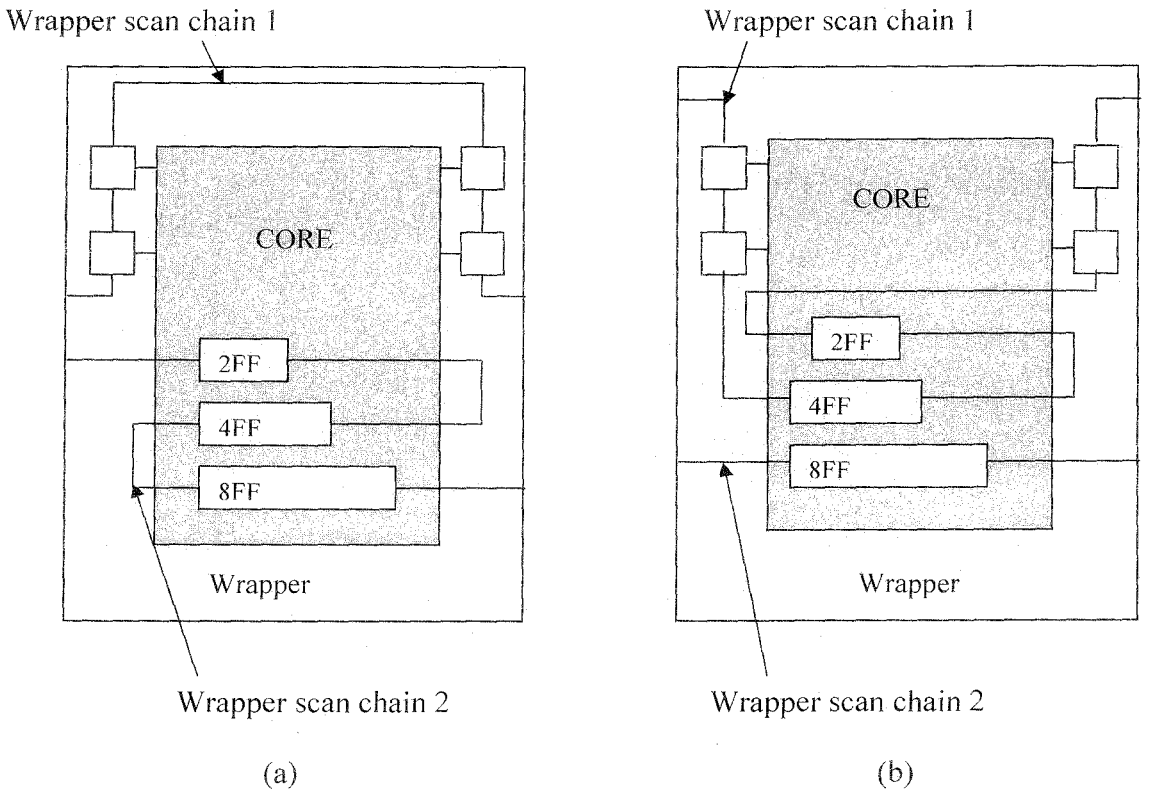


Figure 2.8: Wrapper chains (a) Unbalanced (b) Balanced

This can be explained as follows. Consider a core that has four internal scan chains of lengths 32, 8, 8, and 8, respectively, 4 functional inputs, and 2 functional outputs. Let the number of TAM lines provided be 4. The algorithm in [65] will partition the scan elements among four wrapper scan chains as shown in Fig. 2.9(a), giving $\max\{s_i, s_o\} = 32$. However, the scan elements may also be assigned to only 2 wrapper scan chains as shown in Fig. 2.9(b), which also gives $\max\{s_i, s_o\} = 32$. The second assignment, however, is clearly more

efficient in terms of TAM width utilization, and therefore would be more useful for a wrapper/TAM co-optimization strategy.

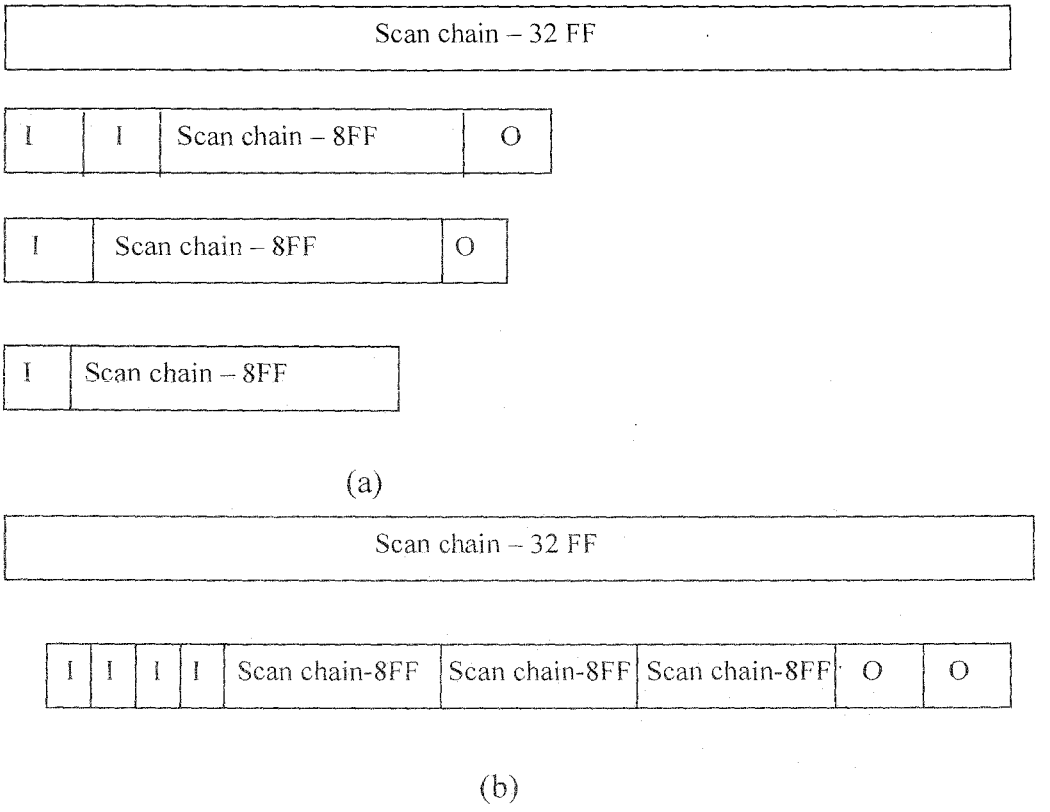


Figure 2.9: Wrapper design example using (a) four wrapper scan chain, and (b) two wrapper scan chains

Koranne [66, 67] addressed a design of reconfigurable core wrapper which allows a dynamic change in the TAM width while executing the core test. This is achieved by placing extra multiplexers at the input and output of each reconfigurable scan chain. He also described a procedure for the automatic derivation of these multiplexers using a graph representation of core wrappers. Instead of connecting the outputs of each scan chain to multiplexers, Larsson and Peng [68] presented a reconfigurable power-conscious core wrapper by connecting the inputs of each scan chain to multiplexers. This approach

combines the reconfigurable wrapper [66,67] with the scan chain clock gating [69] concepts. The reconfigurable core wrapper is useful for cores with multiple tests, where each of the tests has different TAM width requirements.
In [70], Vermaak and Kerkhoff presented a P1500 compatible wrapper design for delay fault testing, based on the digital oscillation test method. To be able to use this method, they introduced extra multiplexers and a cell address register to each wrapper cell. This approach for delay testing is only suitable for combinational cores, because paths between the core's inputs and outputs are tested. Xu and Nicolici [71] proposed a novel core wrapper that addressed the testability problems raised by embedded cores with multiple clock domains. By partitioning core I/Os and scan cells from different clock domains into different virtual cores, they proposed to build wrapper SCs within virtual cores to solve the clock skew problem during the shift phase. To avoid clock skew during the capture phase, a capture window design technique similar to [72] was proposed that supports multi-frequency at-speed testing. The authors also described wrapper optimization algorithms that can tradeoff between the number of tester channels, test application time, area overhead and power dissipation. In [73], Goel described a wrapper architecture for hierarchical cores, which allows for parallel testing of the parent and child cores, at the cost of an expensive wrapper cell design.

2.6 Design-Wrapper Algorithm

V.Iyenger, K. Chakraborty, and E.J. Marinissen [50] have developed an approximation algorithm based on the Best Fit Decreasing (BFD) heuristic [52] to solve P_W efficiently. The algorithm has three main parts, similar to [65]: (i) partition the internal scan chains among a minimal number of wrapper scan chains to minimize the longest wrapper scan chain length, (ii) assign the functional inputs to the wrapper scan chains created in part (i), and (iii) assign the functional outputs to the wrapper scan chains created in part (i). To solve part (i), the internal scan chains are sorted in descending order. Each internal scan chain is then successively assigned to the wrapper scan chain, whose

length after this assignment is closest to, but not exceeding the length of the current longest wrapper scan chain. Intuitively, each internal scan chain is assigned to the wrapper scan chain in which it achieves the *best* fit. If there is no such wrapper scan chain available, then the internal scan chain is assigned to the current *shortest* wrapper scan chain. Next the process is repeated for part (ii) and part (iii), considering the functional inputs and outputs as internal scan chains of length l. The pseudocode for this *Design wrapper* algorithm is as follows.

Procedure *Design- wrapper*

Part (i)

1. **Sort** the internal scan chains in descending order of length
2. **For** each internal scan chain l
3. **Find** wrapper scan chain S_{max} with current maximum length
4. **Find** wrapper scan chain S_{min} with current minimum length
5. **Assign** l to wrapper scan chain S, such that {length(S_{max}) - (length(S)+length(l))} is minimum
6. *If* there is no such wrapper scan chain S *then*
7. Assign l to S_{min}

Part (ii)

8. Repeat steps 1 through 7 to add the primary inputs to the wrapper chains created in part (i)

Part (iii)

9. Repeat steps 1 through 7 to add the primary inputs to the wrapper chains created in part (i)

Algorithm 2.1: Design_wrapper algorithm

	Wrapper scan chains			
	1	2	3	4
Internal scan chains	12+6	12+6	8+6+6	8+8
Wrapper input cells	2	2	0	4
Wrapper output	3	3	0	5
Scan-in length	20	20	20	20
Scan-out length	21	21	20	21

Table 2.1: Wrapper scan chains for Core A.

The algorithm designed based on the BFD heuristic mainly because BFD utilizes a more sophisticated partitioning rule than First Fit Decreasing (FFD), since each scan element is assigned to the wrapper scan chain in which it achieves the *best* fit [52]. FFD was used as a subroutine to the wrapper design algorithm in [65]. In this algorithm, a new wrapper scan chain is created only when it is not possible to fit an internal scan chain into one of the existing wrapper scan chains without exceeding the length of the current longest wrapper scan chain. Thus, while the algorithms presented in [65] always use k wrapper scan chains, *Design wrapper* uses as few wrapper scan chains as possible, without compromising test application time. The worst-case complexity of the *Design wrapper* algorithm is $O(sc.\log sc + sc \cdot k)$, where sc is the number of internal scan chains and k is the limit on the number of wrapper scan chains.

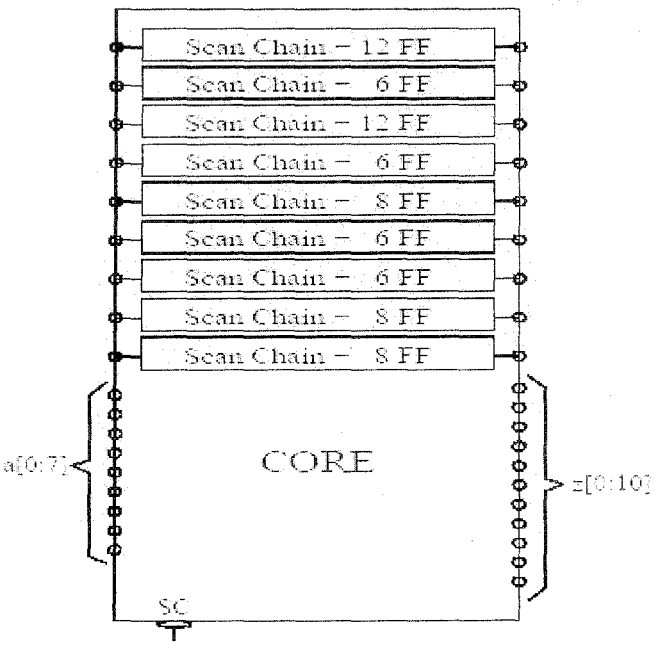


Figure 2.10(a): Example core A[65]

This algorithm is used to design wrapper for the example Core A in [65] shown in figure 2.10(a). Core A has 8 functional inputs `a[0:7]`, 11 functional

outputs z[0:10], 9 internal scan chains of lengths 12, 12, 8, 8, 8, 6, 6, 6, and 6 flip-flops, and a scan enable control *sc*. The test wrapper for A is to be connected to a 1-bit TAM STP (as mandated by the IEEE P1500 standard [48]) as well as to a 4-bit TAM MTP[0:3]. Furthermore, the test wrapper is to contain a bypass from TAM inputs to TAM outputs [65]. The scan elements in the core were partitioned among four wrapper scan chains using the *Design wrapper* algorithm as shown in Table 2.1. This partition yields a longest scan-in chain of length 20, and a longest scan-out chain of length 21, both of which are optimal values for a 4-bit TAM. The wrapper designed for Core A is illustrated in Fig. 2.10(b).

2.7 Test Data Compression

Test Data Compression (TDC) is a test resource partitioning scheme whose main target is the reduction of volume of test data sent from the ATE to the CUT. TDC implies the usage of a compression scheme during test set partitioning and an on-chip decompression unit during test set application. By introducing an on-chip hardware it reduces the load on the ATE, and therefore it simplifies the ATE channel capacity and speed requirements.

There exists wide number of data compression schemes proposed in the literature, some of which could give high compression on test data for SOCs but most of them are too complex for their corresponding decoder to be realized efficiently in hardware. Compression techniques mentioned in literature can broadly categorize in two categories. These are:

- Non-Dictionary based compression scheme.
- Dictionary based compression scheme.

In non-dictionary based compression techniques, total test data is decomposed into either fix length or variable length blocks and a code word, also of either fixed or variable length, is assigned to each block. The basic idea is to assign frequent blocks to a comparably small code word. Iyenger *et al.* [102] proposed a BIST approach for testing non-scanned circuits based on statistical coding. Jas et al. [103] presented a compression technique for scanned circuits by

dividing the test vectors into fixed length blocks and using the Huffman coding technique. In [104], Jas and Touba described another TDC scheme using run-length coding. By exploiting the capabilities of present ATEs to assign groups of inputs to ports and to perform vector repeat per port. Jas *et al.* in [104] employed variable-to-fixed length run-length encoding in which the runs of zeros are encoded by a fixed number of bits.

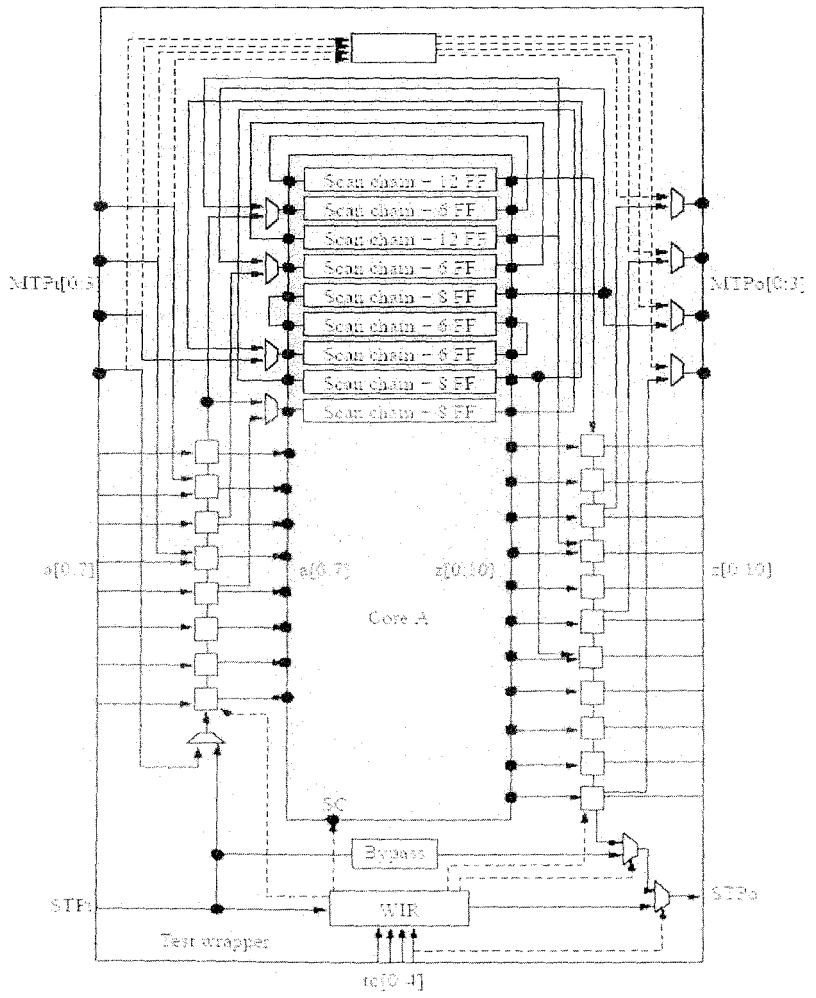


Figure 2.10(b): Wrapper design for example core A from [65].

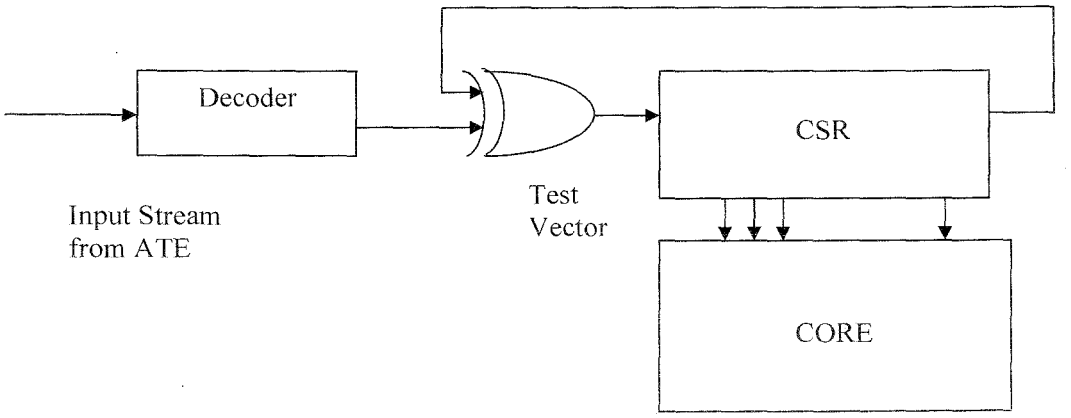


Figure 2.11: Decompression architecture based on a Cylindrical Scan Chain Register (CSR)

This method takes advantage of the fact that the consecutive vectors in the test set tend to be similar as the exercising the faults in the CUT that are structurally related require similar input value assignments. Hence instead of encoding the test vectors, this scheme encodes the difference vectors formed by taking the XOR of the consecutive test vectors. The test set is reordered to maximize the length of zeros in the difference vectors and the decompression hardware utilizes a cyclical scan chain register (CSR), as shown in Fig. 2.11 to obtain back the test vectors from the difference vectors before they are shifted into the scan chain of the CUT. The main drawback of this scheme is additional hardware area overhead that the CSR incurs.

Vranken et al. [105] implemented a similar run-length coding scheme, although the decompression is achieved of-chip on the ATE instead of on-chip. The above coding techniques are based on "variable-to-fixed-length" codes. Chandra and Chakrabarty [106] first proposed a "variable-to-variable-length" test data coding scheme, based on Golomb coding [107]. When using Golomb coding the saving in scan-in power are trade-off against improvement in compression ratio. Rosinger et al. described a Minimum Transition Count (MTC) coding scheme, which can simultaneously reduce test data and power. Frequency-directed run-length coding [108] technique proposed by Chandra and Chakrabarty, further increased the compression ratio. It is designed based

on the observation that the frequency of runs decreases with the increase of their length. While the original FDR coding was based on encoding runs of 0's, El-Maleh and Al-Abaji [109] extended it with EFDR coding, by encoding both runs of 0's and 1's. Gonciari et al. [110] analysed the three main test data compression environment parameters: compression ratio, decoder area overhead and test application time, and introduced a variable-length input Huffman (VIHC) coding scheme to efficiently trade them off. Finally, Tehranipour et al. [111] presented a 9C coding scheme that supports mapping "don't care" bits to random values instead of dedicated long runs of 0's and 1's, which is able to detect more non-modeled defects. On-chip decoders for most of these techniques are parallel in nature. These parallel on-chip decoders will require less amount of area to implement compare to those of serial decoders used for dictionary based methods. But as it is parallel extra synchronization circuitry is needed to synchronize two units with ATE. It is possible to reduce the TAT in parallel decoders by using them at optimum operating frequency.

In dictionary based compression schemes, some selected entries are taken into the dictionary. These entries are encoded using techniques like statistical coding to get a statistical model of the test data and encode blocks according to their frequencies of occurrence. Dictionary based methods select strings of the blocks to establish a dictionary, and then encode them into equal size tokens [107]. The dictionary may be either static or dynamic(adaptive). The static dictionary is permanent, whereas the dynamic dictionary permits additions and deletions of strings as new input is processed. Wolf and Papchristou [112] described a method that employs the well-known LZ77 algorithm, which uses dynamic dictionary. Knieser et al. [113] presented another TDC technique based on LZW algorithm. Li and Chakrabarty [114] proposed a TDC approach using dictionaries with fixed-length indices. One advantage of this method is the elimination of the synchronization problem between the ATE and the SOC, because it does not require multiple clock cycles to determine the end of a compressed data packet. Decoders for the dictionary based compression

techniques are serial in nature, as a result there is no synchronization between ATE and CUT. Usually a RAM or combinational circuitry is used to recover the dictionary encoded patterns and a shift register based circuitry is used to transfer non-dictionary entries. This will lead to a large amount of area overhead of on-chip decoder compare to non-dictionary methods.

2.8 Conclusion

In this chapter a survey report have been presented regarding the methodologies adopted for SOC testing. The conceptual architecture for core test and test access and test wrapper design methodologies have been discussed here. Different method of optimal test architecture, including assigning cores to test busses, distributing a given test data width among multiple test busses and test data compression technique have also been reviewed. From the next chapter onwards, we will enumerate our works. To start with, we will present a simulated annealing based approach for test scheduling

Chapter 3

Test Scheduling using Simulated Annealing based Approach

In this chapter, we present a two-step approach to solve the generalized problem of TAM co-optimization, P_{NPAW} (introduced in Section 2.4) efficiently. In the first step, we use a fast heuristic technique for wrapper / TAM co-optimization. In the final step, simulated annealing has been used to optimize the final assignment of cores to reduce the SOC testing time further. This two-step approach allows us to apply our method to design effective wrapper and TAM architectures for large industrial SOCs. It leads to significantly better optimization than either simulated annealing or other heuristic approaches (as evidenced in the experimental results noted in Section 3.4). It may be noted that simulated annealing based approaches have also been proposed by Zou et al [81] and Harmanani [177]. In [81], TAM lines are not partitioned; rather, a rectangle fitting problem has been solved. It utilized a special data structure called *sequence-pair*. In this formulation, a TAM line needs to be switched individually, rather than the whole bus at a time between the cores. Thus, the test controller becomes complex. It borrowed concepts from floor-planning problem. In [177], an integrated approach for wrapper design, TAM assignment and test scheduling for SOC has been reported. We show that our formulation produces better or competitive results than this and other heuristic approaches presented in the literature, when tested on benchmark SOCs.

The generalized problem of TAM co-optimization, P_{NPAW} has been restated in the following.

Given the total TAM width W and an SOC, determine (i) the number of TAMs for the SOC, (ii) a partition of the total TAM width among the determined

number of TAMs, (iii) an assignment of cores to TAMs of various width, and (iv) a wrapper design for each core, such that SOC testing time is minimized. Section 3.1 discusses the core assignment problem to minimize testing time of individual cores. Section 3.2 illustrates the TAM partitioning strategy that utilizes the algorithm designed in Section 3.1 to evaluate individual partitions. Section 3.3 presents the final optimization step based on simulated annealing to improve the initial solution produced in Section 3.2. Section 3.4 presents experimental results on benchmark SOCs and compares the results with those reported in the literature.

3.1 Core Assignment

The first problem is that of designing an optimal wrapper for the I/O terminals and internal scan chains of a core, such that the core testing time is minimized. To solve this problem, we have used the *Design_wrapper* algorithm [50] detailed in Section 2.6. To calculate the test time, T, for a core with a designated wrapper, we have used the well-known formula [46] given below.

$$T = \{ 1 + max(S_i, S_o) \} \times P + min(S_i, S_o) \quad \ldots\ldots\ldots\ldots\ldots\ldots (3.1)$$

Where, P is the number of test patterns and $S_i(S_o)$ denotes the length of the longest wrapper scan-chain used during scan-in(out) for a core.

The second problem is that of assigning cores to TAMs of given widths. We have used the *Core_assign* algorithm for the assignment of cores to the TAMs. This algorithm takes as input the TAMs with given widths and the cores to be assigned. In each iteration, the algorithm calculates the summed testing time on each TAM by adding up the testing times of all the cores assigned to that TAM. Then the core with the largest testing time (among all unassigned cores) is assigned to the TAM with the shortest current summed testing time. Algorithm 3.1 presents the pseudo code for the heuristic *Core_assign*.

Procedure Core_assign (**B**, B)

1. Let **C** be the set of cores;
2. Let **B** be the set of TAMs and B be number of TAMs;
3. **for all** core $i \varepsilon \mathbf{C}$
4. **for all** TAM j $\varepsilon \mathbf{B}$
5. Let ω_j be the width of TAM j;
6. Perform wrapper design for core i for TAM size ω_j using *Design_wrapper*;
7. Compute $T_i(\omega_j)$ using *Eqn. 3.1*; /* $T_i(\omega_j)$ is the time needed to test core i on TAM j of width ω_j */
8. **for all** TAM j $\varepsilon \mathbf{B}$
9. Set total testing time T_j on TAM j to 0; /* T_j stands for time upto which TAM j is busy in testing cores */
10. **while** $\mathbf{C} \neq \varphi$
11. Select TAM j $\varepsilon \mathbf{B}$, such that T_j is minimum;
12. if there are two or more such TAMs
13. Select TAM j, such that ω_j is maximum;
14. Select Core i $\varepsilon \mathbf{C}$, such that $T_i(\omega_j)$ is maximum;
15. if there are two or more such cores
16. Select TAM k $\varepsilon \mathbf{B}$, such that $\omega_k < \omega_j$ AND ω_k is maximum such value;
17. Select Core i, such that $T_i(\omega_k)$is maximum;
18. Assign Core i to TAM j;
19. $\mathbf{C} = \mathbf{C} - \{i\}$;
20. **end while**
21. Determine TAM r $\varepsilon \mathbf{B}$, such that T_r is maximum;
22. **return** SOC testing time T_r;

Algorithm 3.1: Algorithm for core assignment

We illustrate the *Core_assign* algorithm using an example SOC containing 5 cores and 3 TAMs. The testing times for the 5 cores when assigned to the TAMs of widths 8, 16, and 32 are shown in Table 3.1. Initially, the testing time on all TAMs is 0 cycles. Therefore TAM 1 of width 32, being the widest, is considered first. Core 5 has the highest testing time on TAM 1, therefore Core 5 is assigned to TAM 1. Next, there is a choice between Cores 1 and 3 to be assigned to TAM 2 of width 16. We choose to assign Core 1 to TAM 2 here because the testing time for Core 1 on TAM 3 is higher than the testing time for Core 3 on TAM 3. Next Core 2 is assigned to TAM 3. TAM 2 is now the minimally loaded TAM; therefore, Core 3 is assigned to TAM 2. Finally, Core 4 is assigned to TAM 1. Table 3.1 also presents the final assignment of cores to TAMs. The testing times on TAM 1, 2 and 3 are 180, 200 and 200 clock cycles, respectively. The complexity of *Core_assign* is O (N^2), where N is the number of cores in the SOC.

Cores	Testing time (cycles)			Final assignment	
	TAM 1 32 bits	TAM 2 16 bits	TAM 3 8 bits	TAM	Testing Time (cycles)
1	50	100	200	2	100
2	75	95	200	3	200
3	90	100	150	2	100
4	60	75	80	1	60
5	120	120	125	1	120

Table 3.1: Core testing times for the SOC used to illustrate *Core_assign*

3.2 TAM Partition

In the last section we have seen a technique to assign cores to TAMs of various widths to reduce the overall SOC testing time. This section details the generation of various TAM width partitions and evaluating them using the *Core_assign* routine of Section 3.1. We have used the algorithm *Partition_eval*

(Algorithm 3.2) to develop a fast method to generate partitions and evaluate them using algorithm *Core_assign*. The *Partition_eval* algorithm starts with a set of TAMs **B**. The number of TAMs in the set is B. Let the width of the TAMs in **B** be $\omega_1, \omega_2, \ldots, \omega_B$.To invoke the algorithm, all ω_i's be initialized to one. If the total available width is W, the algorithm, after assigning a width of 1 to ω_1 to ω_{B-1}, assigns remaining width to ω_B. Testing time for this partition is calculated. Next, it gives a call to another recursive routine *Partition* (Algorithm 3.3) to generate further partitions. The number of partitions enumerated is significantly limited by generating only the unique partitions. Here, in the i^{th} level, ω_i is incremented in steps of 1, thereby decreasing ω_B by the same amount to keep total sum W. Each increment in ω_i constitutes a partition provided that it satisfies the condition $\omega_i < \omega_{i+1}$ and $\omega_{B-1} < \omega_B$. This condition restricts ω_i to be in ascending order so that duplicate partitions are eliminated. Without this restriction, enumeration would be as follows: $(\omega_1 + \ldots + \omega_B) = (1+ \ldots +1 +(W - B +1)), (1 + \ldots +2 + (W - B)), \ldots ((W - B +1) +1 + \ldots + 1)$. If both the conditions are satisfied, the partition is accepted and *Partition* algorithm is called recursively to enumerate all the partitions that can be derived from the current partition, in the subsequent levels. Value of *level* varies from B – 1 to 1 and whenever level becomes 0, the control is returned to the calling procedure, indicating that there are no valid partitions in this level. Finally, heuristic algorithm *Core_assign* is used to evaluate these partitions.

Procedure *Partition_ eval*(**B**,B)

1. Let W be the total TAM width;
2. Let set of TAMs **B** = $(\omega_1 + \ldots + \omega_B)$;
3. Let **C** be the set of cores;
4. Set SOC testing time T=200000000;
5. $\omega_B = W - \sum_{i=1}^{B-1} \omega_i$;
6. New SOC testing time T_{new} = *Core_assign*(**B**,B);

7. if (T_{new} < T)

8. Set T = T_{new};

9. Set B_{best}= **B;**

10. *Partition* (**B**,B,B-1);

11. Output B_{best}, T;

Algorithm 3.2: Algorithm for partition evaluation.

This algorithm significantly reduces the computation performed. For example, for W = 8, B = 4, the 5 partitions enumerated are (1+1+1+5), (1+1+2+4), (1+1+3+3), (1+2+2+3), (2+2+2+2). Without the restriction, the repeated partition (1+3+1+3) would also subsequently be enumerated. The best partition produced is remembered in B_{best}.

Procedure *Partition* (**B,** B, level)

1. Let increment = 0;
2. **if** (*level* = 0)
3. **return;**
4. **else if** ((ω_{level}+ 1) > $\omega_{level+1}$)
5. **return;**
6. **else**
7. **while** (true) do
8. **if** (ω_B <= ω_{B-1})
9. **return**;
10. *increment* = *increment* + 1;
11. ω_{level} = ω_{level} + *increment*;
12. ω_B = ω_B – *increment*;
13. New SOC testing time T_{new} = *Core_assign* (**B**,B);
14. **if** T_{new} < T
15. Set T = T_{new} ;
16. Set B_{best} = **B;**
17. *Partition* (**B**,B,*level* – 1);

18. $\omega_{level} = \omega_{level}$ - *increment;*

19. $\omega_B = \omega_B$ + *increment;*

20. **end while**

Algorithm 3.3: Algorithm for partition generation and evaluation.

Now, to generate a solution to the overall problem P_{NPAW}, for a given TAM width W, the *Partition_eval* routine is called, varying the number of partitions from 1 to W. The best result obtained (in terms of test time) is noted as the solution. This is treated as the initial solution which is improved via simulated annealing as noted in the next section.

3.3 Final Optimization Step

The *Partition_eval* procedure provides a fast approximation of the optimal values of TAM width partition and testing time. We further improve upon this result by performing a final optimization step using simulated annealing. This general optimization method was first introduced by Kirkpatrick[175], based on the work of Metropolis[176]. It simulates the softening process ("annealing") of metal. The metal is heated, upto a temperature near its melting point and then slowly cooled down. This allows the particles to move towards an optimum energy state, with a more uniform crystalline structure. The process therefore permits some control over the microstructure. The Simulated Annealing (SA) technique uses a hill-climbing mechanism to avoid getting stuck at local optimum. We now outline the Simulated Annealing (SA) technique and describe how it is adopted to be used for scheduling and TAM design.

3.3.1 The Simulated Annealing Algorithm

The SA algorithm starts with an initial solution and a minor modification of it creates a neighboring solution. The cost of the new solution is evaluated and if

the new solution is better than the previous one, the new solution is kept. A worse solution can be accepted at a certain probability, which is controlled by a parameter referred to as temperature. The temperature is decreased during the optimization process, and probability of accepting a worse solution decreases with the reduction of the temperature value. The optimization terminates, when the temperature value is approximately zero.

3.3.2 Initial Solution and Parameters

We use the *Partition_eval*, *Core_assign* and *Partition* algorithms to generate the initial solution. The parameters, initial temperature TI, the temperature length TL and the temperature reduction factor α ($0 < \alpha < 1$) are determined based on experiments. The initial temperature TI should be high enough to make a large fraction of the neighbors acceptable. The temperature should not be so high that we spend lot of time working with high temperature. On each step, the temperature must be held constant for an appropriate period of time (i.e. for an appropriate number of iterations) in order to allow the algorithm to settle into a "thermal equilibrium", i.e. a balanced state. This time is the temperature length TL. If this time is too short, the algorithm is likely to converge to a local minimum. The combination of temperature steps and cooling times is known as the "annealing schedule", which is usually selected empirically.

Fig 3.1(a) shows an example solution. It corresponds to a case in which a TAM of width 16 has been partitioned into four parts having individual widths of 6, 5, 3, and 2 respectively. Ten cores have been assigned to different partitions – cores 1, 3, 4, 8, 9 have been allotted to partition 2, cores 2 and 5 to partition 1, cores 6, 7 to partition 3 and core 10 to partition 4.

3.3.3 Neighboring Solution

The search process within a neighborhood can be done in a number of ways. A simple way is to select schedules in the neighborhood at random, evaluate these schedules, and decide which one to accept.

Fig 3.1(b) shows an example of generating trial solution from the neighborhood of the current solution (Fig 3.1(a)). Here the TAM widths of the first and the third TAMs are changed from 6 and 3 to 8 and 6 randomly. Similarly the third and the seventh cores are now assigned to the third and fourth TAMs respectively instead of their earlier assignment to second and third TAMs. Here the total TAM width is equal to 16 and number of cores is 10. Now the sum of the TAM width of the solution is greater than the total TAM width 16, hence the TAM widths are decremented at random places by 1 till the sum equals 16. Hence the final TAM widths of the trial solution become 7,4,4,1. The assignment of the cores remains the same. The corrected solution has been shown in Fig 3.1(c).

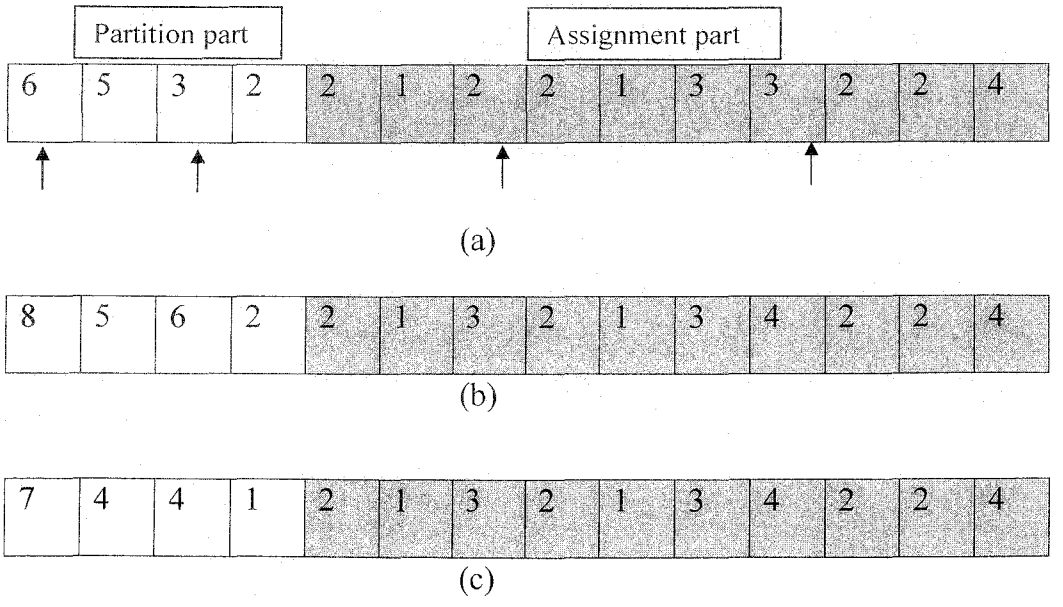


Figure 3.1: (a) A solution, (b) Generating a trial solution (c) Correction

Cost function

Consider an SOC consisting of N cores and B TAMs. If the core i is assigned to TAM j, let the time taken to test Core i be given as $T_i(\omega_j)$ clock cycles. Total core testing time for a TAM is the sum of the testing times for all the cores assigned to it and the total time to test all the cores in the SOC is the maximum of the times taken among the TAMs. The testing time of a core i on a TAM j, $T_i(\omega_j)$ is calculated using Eqn. 3.1 noted earlier.

In order to get the total time to test all the cores in the SOC, we introduce binary variables x_{ij} (where $1 \leq i \leq N$ and $1 \leq j \leq B$) which are used to determine the assignment of cores to TAMs in the SOC. Let x_{ij} be a 0-1 variable defined as follows:

$x_{ij} = 1$, if Core i is assigned to TAM j

$= 0$, otherwise.

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

Algorithm 3.4 presents the pseudo code for the *Simulated_Annealing* algorithm.

Procedure *Simulated_Annealing* (**B**,**C**,W)

1. Construct initial solution, x^{now}; /* As described in Section 3.2 */
2. Initial Temperature: T=TI;
3. **While** stop criteria is not met **do**
4. **for** $i = 1$ **to** TL
5. Generate randomly a neighboring solution x'ε *Neighbourhood* (x^{now});
6. Compute change of cost function $\Delta C = C(x') - C(x^{now})$;
7. /* C(x) is the total test time for a solution x */
8. **if** $\Delta C \leq 0$

9. $x^{now}=x'$

10. **else**

11. Generate q=*random* (0,1);

12. **if** $q<e^{-\Delta C/T}$

13. $x^{now}=x'$

14. Set new temperature $T=\alpha*T$;

15. **return** solution corresponding to the minimum cost function;

Algorithm 3.4: Simulated annealing algorithm

3.4 Experimental Results

We present experimental results for the nine benchmark SOCs from the ITC'02 SOC Test Benchmarks suite [80], namely p34392, p93791, p22810, d695, t512505, g1023, f2126, q12710 and h953. We have implemented our algorithm in C, and the experimental results presented in this work were obtained using a machine with Intel Pentium IV, 1.6 GHz processor and 256 MB memory. All SOC testing times in this section are expressed in clock cycles and the CPU times are in seconds. The coefficient α has been chosen to be 0.97. The number of iterations for one temperature step i.e TL has been chosen to be 3000. The initial temperature TI has been chosen to be 50000. These numbers were determined by experiments. The stopping criteria has been set to be equal to the temperature coming down to be less than or equal to 0.1.

Tables 3.2 through 3.10 present the results for the SOC benchmarks for different TAM widths (W). The tables also note the CPU time needed to run the individual cases.

W	Proposed SA method				
	B	TAM Partition	Core Assignment	Test Time	CPU Time
16	4	6,5,4,1	2,1,1,3,2,4,2,3,4,3, 4,4,4,3,4,1,1,4,3	972352	12.88
24	4	8,8,7,1	3,4,3,2,2,2,4,4,3,2, 3,3,2,4,3,2,2,1,2	663193	14.69
32	4	15,10,5,2	2,3,3,1,3,3,1,2,3,1, 3,3,2,3,3,3,3,4,3	544579	16.81
40	3	18,12,10	3,2,3,3,2,3,2,2,2,3, 3,3,3,3,2,2,2,1,2	544579	18.98
48	3	18,18,12	1,2,2,2,1,2,2,1,1,1, 2,2,1,2,2,1,1,3,2	544579	24.65
56	3	21,18,17	1,1,1,2,2,1,1,1,2,2, 2,2,2,2,2,2,1,3,2	544579	32.88
64	3	25,22,17	1,1,1,2,2,1,2,2,1,2, 2,1,1,2,2,2,1,3,1	544579	43.55

Table 3.2: Results for SOC p34392

W	Proposed SA method				
	B	TAM Partition	Core Assignment	Test Time	CPU Time
16	5	6,3,3,2,2	1,5,5,5,3,5,4,2,3,1,4, 3,4,5,4,1,4,2,3,2,4,1,1, 1,1,1,5,5,2,3,5,2	1728834	19.02
24	5	8,6,5,4,1	1,2,3,3,4,1,4,4,1,4,4, 3,5,3,3,2,2,1,5,2,2,2, 3,4,3,4,5,3,5,3,1,1	1164864	21.99
32	5	12,9,6,3,2	2,5,1,3,1,5,1,5,4,3,4, 4,5,1,4,5,3,1,2,2,1,5, 4,4,4,4,5,1,4,3,4,2	885105	25.42
40	4	15,12,8,5	4,2,2,1,2,1,3,3,3,3,4, 1,3,2,2,1,4,3,1,4,1,2, 4,4,1,3,2,2,3,3,3,4	707785	28.37
48	3	23,23,2	1,2,2,1,2,1,2,3,2,2,1, 1,3,3,3,3,3,2,3,1,3,2, 3,2,2,1,3,3,3,2,2,3	597447	32.44
56	4	23,16,9,8	2,3,3,3,1,3,3,4,1,3,4, 1,1,3,1,3,1,3,3,4,3,4, 1,2,2,4,3,1,4,3,2,2	508415	36.52
64	4	23,23,11,7	2,4,4,2,4,3,3,4,4,2,1, 3,2,2,3,3,1,2,3,1,4,4, 3,4,4,2,2,4,4,3,1,4	438878	40.74

Table 3.3: Results for SOC p93791

W	Proposed SA method				
	B	TAM Partition	Core Assignment	Test Time	CPU Time
16	4	6,5,3,2	2,3,3,3,1,1,4,1,2,3, 3,4,3,3,4,3,1,2,1,1, 1,1,1,1,2,4,4,1	430951	35.72
24	5	8,6,5,3,2	3,1,3,5,4,5,3,1,1,2, 4,3,2,4,1,4,3,2,4,2, 2,2,2,4,1,5,4,3	288485	41.28
32	7	11,5,5,3,3,3,2	5,1,5,5,6,7,7,4,3,7, 7,7,3,1,2,1,7,3,2,6, 4,6,2,4,6,2,3,2	220521	87.03
40	7	15,6,6,4,4,3,2	2,3,7,4,5,1,1,6,6,4, 3,4,2,5,2,4,7,4,6,7, 6,4,5,2,7,1,6,4	183031	114.94
48	6	16,10,7,6,5,4	4,1,1,2,4,3,6,1,2,1, 1,6,6,2,6,1,5,2,2,5, 5,5,6,6,6,3,1,6	154323	173.74
56	5	16,12,11,11,6	4,1,2,4,2,2,1,4,1,1, 1,2,5,5,1,4,2,5,2,4, 5,4,1,2,2,3,1,4	145417	68.63
64	5	31,10,9,8,6	4,5,3,1,1,3,3,3,3,3, 4,4,3,5,1,4,1,2,1,4, 2,5,5,1,1,3,3,5	132111	77.66

Table 3.4: Results for SOC p22810

W	Proposed SA method				
	B	TAM Partition	Core Assignment	Test Time	CPU Time
16	3	8,6,2	3,1,3,3,1,2,2,3,3,1	42268	20.55
24	4	12,2,9,1	1,1,2,2,1,3,3,2,4,1	28289	30.18
32	3	18,9,5	3,3,3,3,2,1,1,3,3,1	21518	40.38
40	3	17,11,8,4	4,2,4,4,1,3,3,4,3,2	17624	48.30
48	5	18,17,9,2,2	1,1,2,3,1,5,5,2,1,4	14310	54.06
56	4	18,16,16,4,2	3,3,3,1,5,4,2,1,5,2	12462	60.02
64	5	19,18,18,5,4	2,3,4,2,5,3,1,2,4,1	10985	65.92

Table 3.5: Results for SOC d695

W	Proposed SA method				
	B	TAM Partition	Core Assignment	Test Time	CPU Time
16	2	14,2	2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1	10530995	17.30
24	2	16,8	2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1	10453470	19.07
32	3	28,3,1	3,3,3,3,3,2,2,2,3,2,3,2,3,3,3,2,3,3,3,2,3,3,3,2,3,3,3,2,2,1	5268868	22.44
40	3	30,5,5	3,2,2,2,3,3,3,3,2,2,3,2,2,3,3,3,2,3,3,3,3,3,2,2,3,3,2,3,3,1	5228420	24.32
48	3	29,13,6	2,2,3,3,2,2,2,3,3,2,2,2,2,3,3,3,2,2,2,2,2,2,3,2,2,2,2,2,3,1	5228420	26.38
56	3	31,23,2	3,2,2,2,2,3,3,2,3,2,3,2,2,2,2,2,3,2,2,2,3,3,3,2,2,2,3,3,3,1	5228420	28.39
64	3	31,20,13	2,3,2,2,2,2,2,2,3,2,2,2,3,3,2,3,3,2,2,2,2,2,3,2,3,3,3,3,3,1	5228420	30.36

Table 3.6: Result for SOC t512505

W	Proposed SA method				
	B	TAM Partition	Core Assignment	Test Time	CPU Time
16	5	5,5,4,1,1	3,2,5,1,1,5,5,4,5,1,1,3,3,4	31414	30.76
24	5	7,7,6,2,2	5,4,2,3,2,4,2,5,4,2,2,3,3,1	21521	35.99
32	5	10,10,7,3,2	5,4,1,2,1,1,1,4,4,2,3,3,5,3	16492	76.42
40	4	21,8,7,4	2,3,1,4,1,1,1,3,1,3,1,3,3,2	14794	33.06
48	4	20,15,14,7	4,1,2,3,1,4,4,4,1,1,2,1,4,2	14794	36.89
56	4	19,18,11,8	1,4,4,2,1,1,4,4,3,3,3,3,1,3	14794	40.47
64	4	30,14,12,8	4,3,2,1,2,2,2,2,3,2,2,2,2,4	14794	43.90

Table 3.7: Result for SOC g1023

W	Proposed SA method				
	B	TAM Partition	Core Assignment	Test Time	CPU Time
16	2	8,8	2,1,1,1	372125	14.10
24	3	13,10,1	1,3,2,3	335334	16.22
32	2	18,14	2,1,1,1	335334	18.29
40	2	28,12	1,2,2,2	335334	20.63
48	2	33,15	1,2,2,2	335334	22.89
56	2	35,21	1,2,2,2	335334	25.22
64	2	39,25	1,2,2,2	335334	27.37

Table 3.8: Result for SOC f2126

W	Proposed SA method				
	B	TAM Partition	Core Assignment	Test Time	CPU Time
16	4	6,4,3,3	3,1,4,2	2222349	21.64
24	4	12,5,4,3	3,2,4,1	2222349	25.27
32	4	15,8,5,4	1,2,3,4	2222349	29.00
40	4	13,12,12,3	3,4,2,1	2222349	32.63
48	4	19,12,10,7	4,3,2,1	2222349	36.34
56	4	22,18,10,6	2,1,4,3	2222349	40.19
64	4	24,19,12,9	1,2,3,4	2222349	43.88

Table 3.9: Result for SOC q12710

W	Proposed SA method				
	B	TAM Partition	Core Assignment	Test Time	CPU Time
16	2	10,6	1,2,2,2,2,2,2,2	119357	6.30
24	2	14,10	2,1,1,1,1,1,1,1	119357	7.22
32	2	19,13	1,2,2,2,2,2,2,2	119357	8.39
40	2	21,19	1,2,2,2,2,2,2,2	119357	9.42
48	2	25,23	1,2,2,2,2,2,2,2	119357	10.40
56	2	31,25	1,2,2,2,2,2,2,2	119357	11.49
64	2	38,26	2,1,1,1,1,1,1,1	119357	14.47

Table 3.10: Result for SOC h953

Table 3.11 compares the performance of our simulated annealing based approach with other existing works reported in the literature. The results produced by our approach have been given in the column marked 'Our'. The cases, in which the result produced by our approach is either better or same as the best result reported in the literature, have been marked in bold letters. It may be noted that out of 42 test cases, in 21 cases, our approach produces the best possible result. This clearly establishes the acceptability of our simulated annealing based formulation. A comparison between the with work [177] shows that our work produces results better than [177] in 17 cases, whereas, [177] produces better results than ours in 8 cases.

SOC	W	[166]	[98]	[178]	[90]	[77]	[85]	[179]	[81]	[180]	[177]	Our
f2126	16			372125			357109	357089	357088		357088	372125
	24			335334			335334	335334	335334		335334	**335334**
	32			335334			335334	335334	335334		335334	**335334**
	40			335334			335334	335334	335334		335334	**335334**
	48			335334			335334	335334	335334		335334	**335334**
	56			335334			335334	335334	335334		335334	**335334**
	64			335334			335334	335334	335334		335334	**335334**
t512505	16			10530995			10531003	11210100	10530995		10530995	**10530995**
	24			10453470			10453470	10525823	10453470		10453470	**10453470**
	32			5268868			5268872	6370809	5268868		5268868	**5268868**
	40			5268420			5268420	5240493	5268420		5268420	**5268420**
	48			5268420			5268420	5239111	5268420		5268420	**5268420**
	56			5268420			5268420	5228474	5268420		5268420	**5268420**
	64			5268420			5268420	5228489	5268420		5268420	**5268420**
p34392	16	998733	1033210	1010821	1021510	1053491	1016640	995739	944768		940626	972352
	24	720858	882182	680441	729864	759427	681745	690425	628602		637263	663193
	32	591027	663193	551778	630934	544579	553713	544579	544579		544579	**544579**
	40	544579	544579	544579	544579	544579	544579	544579	544579		544579	**544579**
	48	544579	544579	544579	544579	544579	544579	544579	544579		544579	**544579**
	56	544579	544579	544579	544579	544579	544579	544579	544579		544579	**544579**
	64	544579	544579	544579	544579	544579	544579	544579	544579		544579	**544579**
g1023	16			34459			31444	32602	31139		31777	31414
	24			22821			21409	22005	21024		20261	21521
	32			16855			16489	17422	15890		16332	16492
	40			14794			14794	14794	14794		14794	**14794**
	48			14794			14794	14794	14794		14794	**14794**
	56			14794			14794	14794	14794		14794	**14794**
	64			14794			14794	14794	14794		14794	**14794**
p93791	16	1771720	1786200	1791638	1775586	1932331	1791860	1767248	1757452	1827819	1777840	**1728834**
	24	1187990	1209420	1185434	1198110	1310841	1200157	1178776	1169945	1220469	1204168	**1164864**
	32	887751	894342	912233	936081	988039	900798	906153	878493	945425	943087	885105
	40	698563	741965	718005	734085	794027	719880	737624	718005	787588	760868	707785
	48	599373	599373	601450	599373	669196	607995	608285	594575	639217	638993	597447
	56	514688	514688	528925	514688	568436	521168	539800	509041		557890	**508415**
	64	460328	473997	455738	472388	517958	549233	485031	447974	457862	489591	**438878**
d695	16	42568	42644	44307		44545	42716	41905	41604	41847	42382	42268
	24	28292	30032	28576		31569	28639	28231	28064	29106	29103	28289
	32	21566	22268	21518		23306	21389	21467	21161	20512	21456	21518
	40	17901	18448	17617		18837	17366	17308	16993	18961	17480	17624
	48	16975	15300	14608		16984	15142	14643	14182	17527	14779	14310
	56	13207	12491	12462		14794	13208	12493	12085		12708	12462
	64	12941	12941	11033		11984	11279	11036	10723	13348	11069	10985
p22810	16	462210	468011	458068	434922	489192	446684	465162	438619	473418	464809	**430951**
	24	361571	313607	299718	313607	330016	300723	317761	289287	353834	310456	**288485**
	32	312659	246332	222471	245622	245718	223462	236796	218855	236186	233101	220521
	40	278359	232049	190995	194193	199558	184951	193696	175946	195733	197556	183031
	48	278359	232049	160221	164755	173705	167858	174491	147944	159994	166169	154323
	56	268472	153990	145417	155417	157159	145087	155730	126947		145417	145417
	64	260638	153990	133405	133628	142342	128512	145417	109591	128332	123543	132111

Table 3.11: Comparison with existing works

3.5 Conclusion

In this chapter we have presented a simulated annealing based approach for TAM partitioning and test scheduling for the cores in an SOC. However, a major component in the SOC testing involves the testing of interconnects. In the following chapter, we introduce a Genetic Algorithm based approach for testing both cores and interconnects. It also attempts to reduce transitions during shifting-in the patterns through the wrapper scan chains.

Chapter 4

Integrated Core and Interconnect Testing

Modern system designs are dominated by the interconnects between the modules. In an SOC based design scenario, the core providers supply the test patterns which are to be applied by the system integrator to test the overall system. In the last chapter, we have discussed about testing such a system using a simulated annealing approach. However, the work does not address the issue of testing interconnects. To test an interconnect between two cores, placed on TAM i and TAM j respectively, the interconnect test data has to reach output of first core through TAM i, and the response should traverse from input of second core to the test sink via TAM j. For an interconnect of width n, $(n+1)$ test patterns are needed to test it completely [99]. This imposes a severe cost on the testing time until and unless, a number of such tests can be carried out simultaneously. To identify the maximum number of interconnects that can be tested in parallel, this chapter presents a clique formulation of the problem and uses the left-edge heuristic to solve it.

Another important issue is that of test power reduction. In this chapter, we consider an worst case scenario in which the wrappers designed around cores do not possess the by-pass registers. This may be noted that the by-pass registers are optional in a wrapper design. In such a situation, if core-2 is fitted next to core-1 on the same TAM, the patterns meant for core-2 will pass through the scan chain of core-1 as well, creating a number of toggles to the inputs of core-1. This leads to unnecessary power consumption in core-1 also. It may be noted that several other works [77, 78, 86, 89, 127, 179, 180, 181] have been reported in the literature to perform test scheduling keeping the power budget of the SOC as a constraint. However, these works do not consider interconnect testing. We will take up the problem of power-

constrained core test scheduling in the next chapter. This chapter will address the following two important issues.

1. To perform assignment of cores to TAMs to reduce core testing time. At the same time, it will identify the potential parallelism possible between the testing of interconnects. The overall test time of the SOC is taken as the sum of core testing time and interconnects testing time. The core assignment will minimize this overall testing time.
2. To reduce the scan ripples through the wrapper scan chains. As discussed earlier, if two cores are placed on a single TAM, the test patterns for the second core will also pass through the scan chains of the first core, while the responses of the first core will also pass through the scan chains of second core. Thus, ordering of cores on a particular TAM affects the number of scan ripples reaching individual cores. Our formulation will attempt to minimize such ripples.

This chapter presents a combined approach to handle both test time and scan power. Section 4.1 discusses the basics. Section 4.2 presents the computation of scan ripples during test. A Genetic Algorithm based formulation to solve the combined problem has been presented in Section 4.3. Section 4.4 presents the experimental results.

4.1 Testing SOCs

SOC testing time is determined by the time needed to test the cores, the interconnects, and the user-defined-loguc (UDL). Since the UDL portion is expected to be very small, the SOC testing time will be dominated by the other two components. As noted in Eqn 3.1 (Chapter 3), the testing time of core i assigned to TAM j of width ω_j is given by,

$$T_i(\omega_j) = (1 + max(S_i, S_0)) * p_i + min(S_i, S_0)$$

Where p_i is the number of test patterns for core i, S_i (S_0) is the length of the longest wrapper scan-in (scan-out) chain for the core.

Different test busses can be used simultaneously for delivering test data to the cores, while the cores assigned to the same test bus are to be tested sequentially. Total core testing time for a test bus is the sum of the testing times of the cores assigned to it. The total time to test all cores in the SOC is the maximum of the times taken among all test buses.

Let x_{ij} be a variable defined as follows:

x_{ij} = 1 if core i is assigned to bus j

= 0 otherwise

Now, the total testing time needed to test all the cores in the system is

$$C = max_j \{ \Sigma T_i (\omega_j) * x_{ij} \}, \; 1 \leq i \leq N_c \; and \; 1 \leq j \leq N_b$$

Where N_c and N_b are the number of cores and number of test buses respectively, in the system.

Interconnect Testing

Interconnect testing of an SOC is the testing of all interconnects present in that SOC. Test patterns for the interconnection-under-test are delivered to the source core by its test bus and the destination core delivers the received responses to the test bus assigned to it. To test an interconnection of width n, while guaranteeing the complete diagnosis of multiple interconnection faults, $(n+1)$ test patterns are required [99]. The width of each test pattern is equal to the width of the interconnection to be tested. Two cycles are required to test each interconnect, one cycle for the source core and the other for destination core. Serialization is required, when the test rail width is less than the test pattern width (i.e. the width of the interconnection to be tested).

Let

T = testing time for the interconnect between core i and core j

Φ = width of test pattern = width of the interconnect t_i

Wt_i = width of the TAM assigned to core i

Wt_j = width of the TAM assigned to core j

$W = \min \{ Wt_i , Wt_j \}$

P = number of test patterns for the interconnect = $\Phi + 1$

Then,

$T = 2*P$ cycles if $W \geq \Phi$

$= \{2+2* (\Phi - W)\}* P$ cycles otherwise .....(4.1)

This is because, when $W < \Phi$, first W bits of the test patterns can be delivered in parallel while the rest ($\Phi - W$) bits are delivered serially.

Schedule-Interconnect Testing

Once the assignment of cores to TAMs has been finalized, the interconnects testing time for the whole SOC can be evaluated as follows.

Let the tuples $< C_1 , C_2 >$ and $< C_3 , C_4 >$ denote two interconnects, one from core C_1 to C_2 and the other from core C_3 to C_4 . These two interconnects can be tested in parallel if, TAM (C_1) $\neq$ TAM (C_3) and TAM (C_2) $\neq$ TAM (C_4). Now, we construct a graph where each interconnect is represented by a vertex and there is an edge between two vertices if the corresponding two interconnects can be tested in parallel. If the resultant graph is a clique, all the interconnections can be tested in parallel. But, in general, this will not be the case. We try to minimize the testing time by finding a *minimum clique cover* (partitioning the vertex set of a graph into minimum number of subsets, so that each subset is a clique).

Interconnects are now partitioned into subsets (corresponding to the cliques). All interconnects in a subset can be tested simultaneously. The time required can be calculated using Eqn. 4.1. Testing time of a subset is the maximum of the testing times for the interconnections in it. All such subsets are tested sequentially. Thus the total interconnect testing time is equal to the sum of the times to test each subset. The *minimum clique cover* problem is known to be *NP-complete*. Let G (V, E) denote a graph, where V is the set of vertices and E is the set of edges. The problem is to partition the vertex set into minimal number of sets such that each set forms a clique and each node belongs to

exactly one set. We have used the left-edge algorithm [100] to solve this clique partitioning problem in a heuristic approach.

4.2 Scan Ripples in TAMs

Excessive switching activity during testing can cause average power dissipation and peak power during test to be much higher than during normal operation. This obviously can cause damage to the SOC. As shown in Figure 2.6, a single test rail provides access to one or more cores. Testing the cores in a rail must be done sequentially. To test Core2, in Figure 2.6, all the test data for Core2 must pass through the wrapper scan chain of Core1. In general, if we have n cores on a TAM, to test the n^{th} core, all the test data for this core must be passed through the wrapper scan chains of the cores 1, 2..., n-1. So, the switching activity, while testing, can be reduced by reordering the position of the cores on the TAM.

To estimate switching activity, *weighted transition* metric has been introduced [101]. *Weighted transition* metric models the fact that the measure of switching activity depends not only on the number of transitions in it, but also on their relative positions. For example, consider the test vector $b_1b_2b_3b_4$= 1001 where b_4 is scanned in first and b_1 last. This vector has two transitions - first between b_1 and b_2, and the second between b_3 and b_4. During the scan-in of this vector, the first transition will occur only once but the second transition will be propagated throughout the scan chain. Hence, the first transition results in less switching activity and thus has less weight than the second. More formally, the number of weighted transitions in a test vector or an output response is given by,

$$\text{Weighted\_Transitions} = \Sigma\,(\text{Size\_of\_Scan\_Chain} - \text{Position\_of\_Transition}) \quad \ldots\ldots\ldots(4.2)$$

The intrinsic value of position changes depending upon whether one considers a test vector or an output response. For example, consider the scan vector $b_1b_2b_3b_4$ = 0001. In the case where this is a test vector, the position of the transition between b_3 and b_4 is 1 and weight is (4-1) = 3, whereas, when this is

an output response, position of the given transition is 3 and weight is (4-3) = 1.This is so because during scan-in, b_4 is scanned in first, while during scan-out, b_4 is scanned out first.

Apart from the scan-in transitions and scan-out transitions there is one more kind of transition. This transition occurs when the first bit of the test vector differs from the last bit of the previous output response. In this case, the transition propagates through the entire scan chain, and the weight assigned to it is equal to the size of the scan chain.

For example, consider the test sequence shown in Figure 4.1, which is composed of three test vectors and the corresponding output responses. The scan chain has four flip-flops and hence scan vectors are four bit long. The number of weighted transitions for test vector V1 is (4-3) + (4-1) = 4, and that for output response R1 is (4-1) + (4-2) = 5. For V2, R2 and V3, R3 these are 6, 3 and 1, 5 respectively. In addition, a transition will propagate through the entire scan chain when the first bit of test vector V2 will be scanned in. The weight associated to this transition is 4, which corresponds to the size of the scan chain. Assuming an initial state of 0000 for the scan flip-flops, the total number of weighted transitions produced by the test sequence is (4+5+6+3+1+5) + (4) =28.

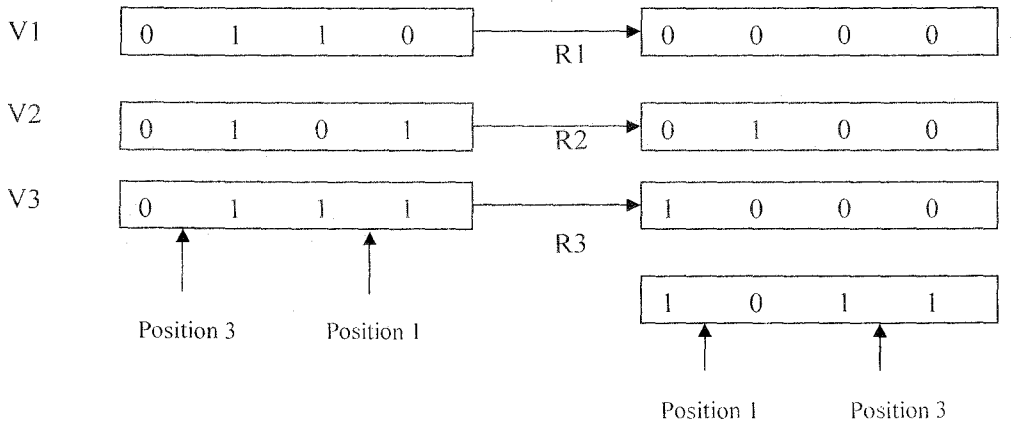


Figure 4.1: Weighted Transition Metric

When a number of cores are connected to a TAM, the individual TAM lines act as scan-chains. The scan-flipflops of different cores get connected to each other in series, forming long scan chains. Eqn. 4.2 is applied on these long scan chains to obtain the number of scan ripples in the entire testing process. Next section discusses about Genetic Algorithmic (GA) approach to the test access architecture design and minimization of scan-in, scan-out ripples.

4.3 GA Formulation

In this section, we present the Genetic Algorithm based formulation of the combined problem of SOC test time minimization and scan ripple reduction. It can formally be stated as follows.

Problem: *Given total TAM width W, an SOC with cores and the interconnects between them, determine*

(i) the number of TAMs for the SOC, (ii) a partition of the total TAM width, (iii) an assignment of cores to TAMs, (iv) a wrapper design for each core, (v) an ordering of cores assigned to a TAM,

such that, (i) total test time is minimized,

(ii) switching activity during testing is minimized.

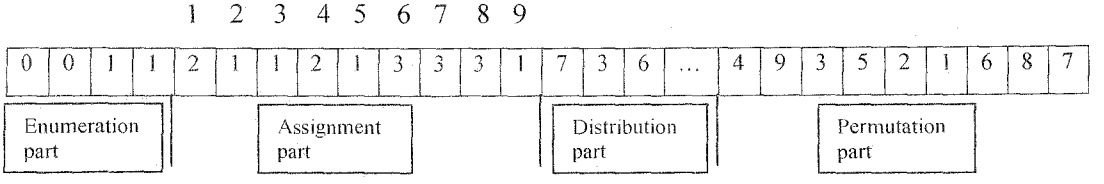


Figure 4.2: Example Chromosome

4.3.1 Solution representation

A chromosome in our approach consists of four parts namely *enumeration part, assignment part, distribution part,* and *permutation (ordering) part. Enumeration part* represents the number of TAMs for the SOC, in binary

format. *Assignment part* is the assignment of cores to test buses, which is an array of integers, with i^{th} element representing the bus number to which core i is assigned. *Distribution part* is the distribution of total bus width, an array of integers, with j^{th} element representing the width of bus j. Since each chromosome may have different number of TAMs, to make the length of distribution part uniform, we have used W (total TAM width) integer entries. If the enumeration part of the chromosome has a value e, only the first e entries of the distribution part are significant. The sum of these e widths is equal to the total TAM width W. The *permutation part*, also an array of integers, represents an ordering of the cores assigned to test buses. These four parts of the chromosome form a solution to the given problem. Figure 4.2 shows an example chromosome for the above problem. Here W = 16 and number of cores N_c = 9. The number of TAMs is 3 (binary 0011), as identified by the enumeration part. Cores 2, 3, 5 and 9 are assigned to TAM 1, cores 1 and 4 are assigned to TAM 2 and finally cores 6, 7 and 8 are assigned to 3^{rd} TAM. Bus width W(16) is divided into 7, 3 and 6. In the permutation part, 4 comes earlier than 1, thus, on TAM 2, core 4 will be placed first, and then core 1. Similarly, for TAM 1, the ordering of cores is as follows: core 3, followed by core 5, followed by core 2.

4.3.2 Crossover Operator

Two-point crossover is applied to the *enumeration part, assignment part* and *permutation part,* while single point crossover is applied to the *distribution part* of the chromosome. The crossover works on individual parts as follows.

1. For the enumeration part, portion beginning to the first point of parent1 is copied to child1, while the similar portion from parent2 is copied to child2. Portions between two points are generated randomly. Portion of parent1 from second point to the end is copied to child2, whereas, the similar portion from parent2 is copied into child1.
2. For the assignment part, portion from the beginning to first crossover point of parent1 is copied to child1, while that portion from parent2 is

copied to child2. Similarly, portion from second cross point to end of parent1 is copied to child2, and the portion from parent2 is copied into child1. The portion between the two cross points of the assignment part of child1 and child2 are generated randomly.

3. For the distribution part, from the beginning to cross point of parent1 is copied to child1 and that from parent2 is copied to child2. Portion from crosspoint to end of parent1 is copied to child2, while the similar portion of parent2 is copied into child1.

4. For the permutation part, crossover is performed similar to the assignment part.

4.3.3 Mutation Operator

Mutation on the *enumeration part, assignment part* and *permutation part* selects some random number of positions and changes their value. Mutation on the *distribution part*, selects a random number of elements and tries to modify them so that sum of the values on this part remains equals to W.

4.3.4 Fitness measure

Measure of fitness of chromosomes is useful to rank the obtained solutions at a particular generation. Since our objective is to integrate both the testing time and scan ripples, we proceed as follows.

A chromosome gives the TAM distribution, allocation of cores on them and the ordering of cores. The core testing time can now be evaluated as shown in Section 4.1. The interconnect testing time is next computed using Eqn. 4.1 and Left Edge algorithm [100]. Sum of these two gives the total testing time. The number of scan ripples can be calculated using Eqn. 4.2 (Section 4.2). The testing time (T) and weighted transitions (W), computed thus, are next

combined together. For this purpose, the values are first normalized by dividing them by the maximum testing time and the maximum weighted transitions among all chromosomes in the initial population of the GA. Let, the normalized values for a particular chromosome are *TNORM* and *WNORM* respectively. We compute the fitness of the chromosome as,

$$F = \sqrt{(TNORM^2 + WNORM^2)}$$

That is, if we plot the solutions on a graph with the axes T and W, a better fit chromosome will have lesser distance from the origin (corresponding to the hypothetical optimum solution with zero test time and zero scan transitions).

4.3.5 Genetic Algorithm

The overall genetic algorithm for the test time and transition minimization has been shown in Algorithm 4.1. It starts with a randomly created initial population and evolves over generations to improve upon the solution quality. The fitness of each chromosome is evaluated as in Section 4.3.4. To create the next generation, 20% best fit chromosomes of current generation are directly copied to the next generation, 30% are generated via mutation and the remaining 50% via crossover.

Algorithm GA

1. Create a random initial population of size POP_SIZE.
2. MAX_TIME = BIG_NUM; MAX_TRANS = BIG_NUM;
3. For each chromosome *i* do
    - 3.1 Let T_i = Test time for chromosome *i*
    - 3.2 Let W_i = Transitions for chromosome *i*
    - 3.3 If MAX_TIME < T_i then MAX_TIME = T_i
    - 3.4 If MAX_TRANS < W_i then MAX_TRANS = W_i

4. For each chromosome *i* do

    4.1 $Fitness_i = \sqrt{((T_i / MAX\_TIME)^2 + (W_i / MAX\_TRANS)^2)}$

5. Sort population

6. $No\_of\_gen\_without\_improv = 0$

7. While ($No\_of\_gen\_without\_improv < 50$) do

    7.1 Copy 20% best chromosomes to next generation

    7.2 Generate 30% new chromosomes via mutation

    7.3 Generate 50% new chromosomes via crossover

    7.4 Evaluate fitness of each chromosome using Steps 3.1, 3.2 and 4.1

    7.5 Sort population

    7.6 If (no improvement in fitness of best chromosome) then

        $No\_of\_gen\_without\_improv$ ++

        Else

        $No\_of\_gen\_without\_improv = 0$

Algorithm 4.1: Genetic algorithm

4.4 Experimental results

In this section, we present our experimentation with the ITC'02 benchmark SOCs. The benchmark files, in their current format, do not contain the interconnect information. For the sake of our experimentation, we have generated different number of interconnects. For an SOC with *n* cores, we assume that at the most *n(n-1)* possible interconnects can be there. We generate different test cases with different percentages of interconnects. For a particular interconnect density, the actual connections are generated randomly. Important information needed to compute scan chain ripples is the test patterns applied to

a core and the corresponding responses. Though the benchmarks are having information about the total number of test patterns, the actual patterns are not specified. Hence, we have generated the specified number of input patterns and output responses randomly.

Fig 4.3, 4.4, and 4.5 present the results for SOC d695 with 40%, 75%, and 100% interconnects respectively. All chromosomes in the final generation have been plotted on a graph with X-axis representing the switching activity and Y-axis representing the testing time. A solution is said to be a *dominated* one, if there exists some other solution with both higher test time and switching activity. Further, a dominated solution should not dominate any other solution. Thus a dominated solution is a good one, as there does not exist any other solution with both lesser testing time and switching activity. The dominated solutions have been marked in the figures. The numbers alongside a dominated solution, marks the number of TAMs for which the solution has been obtained. With 40% Interconnect density and TAM width of 56, minimum testing time obtained is 15764 cycles and the minimum switching activity is 9508414 weighted transitions. Fig 4.6, 4.7, 4.8 present the results for SOC g1023. Fig 4.9, 4.10 present the results for SOC p93791. Similarly, other figures (Fig. 4.11 through 4.25) present the results for different other SOCs with varying interconnect densities. The population size of GA for smaller circuits *(q12710)* is 100 and for the rest is 1000. It may be noted that from Fig 4.12 onwards (that is for SOCs d281, f2126, h953, u226, q12710) only the dominated points have been plotted.

4.5 Conclusion

In this chapter we have discussed about test scheduling that takes care of both core and interconnects testing time. It also performs a trade-off between test power and test time. As a solution, it comes up with a range of solution points from which the test engineer can choose a proper combination. In the next chapter we will discuss about another test scheduling technique that does not perform TAM partitioning, rather it assigns TAM width independently to the

cores subject to the constraint that no TAM line is used by more than one core simultaneously.

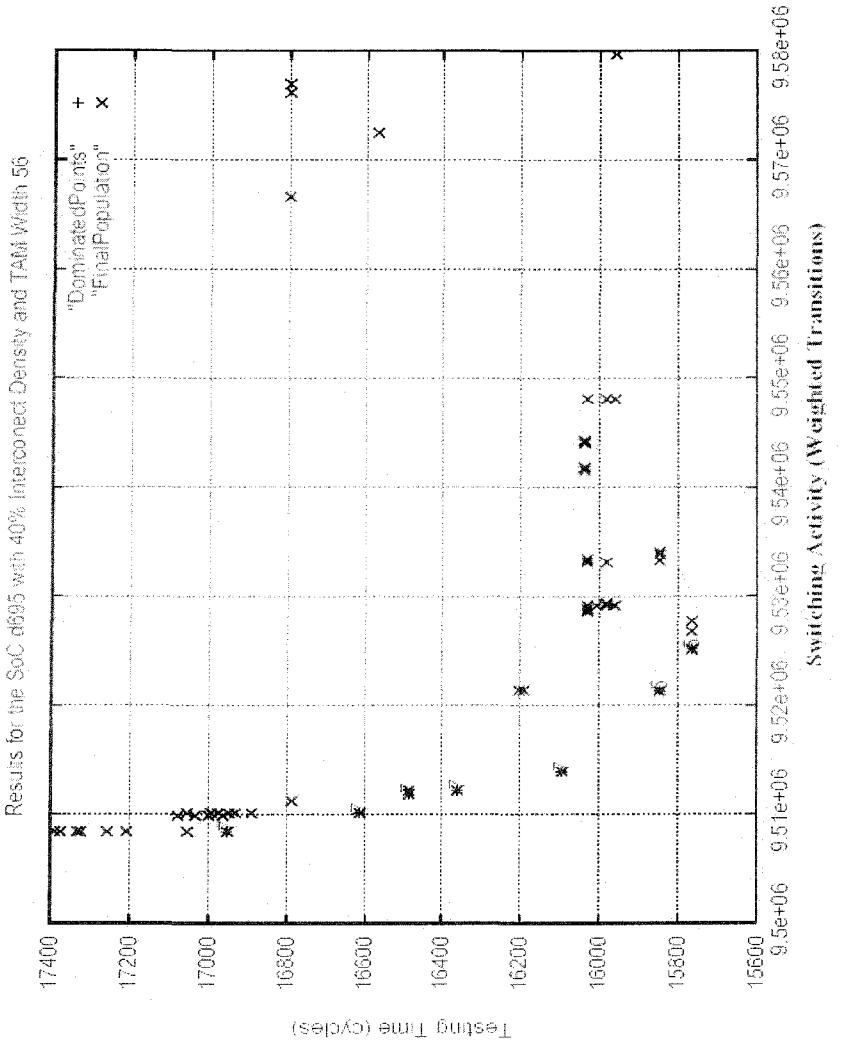


Figure: 4.3: Results for SOC d695

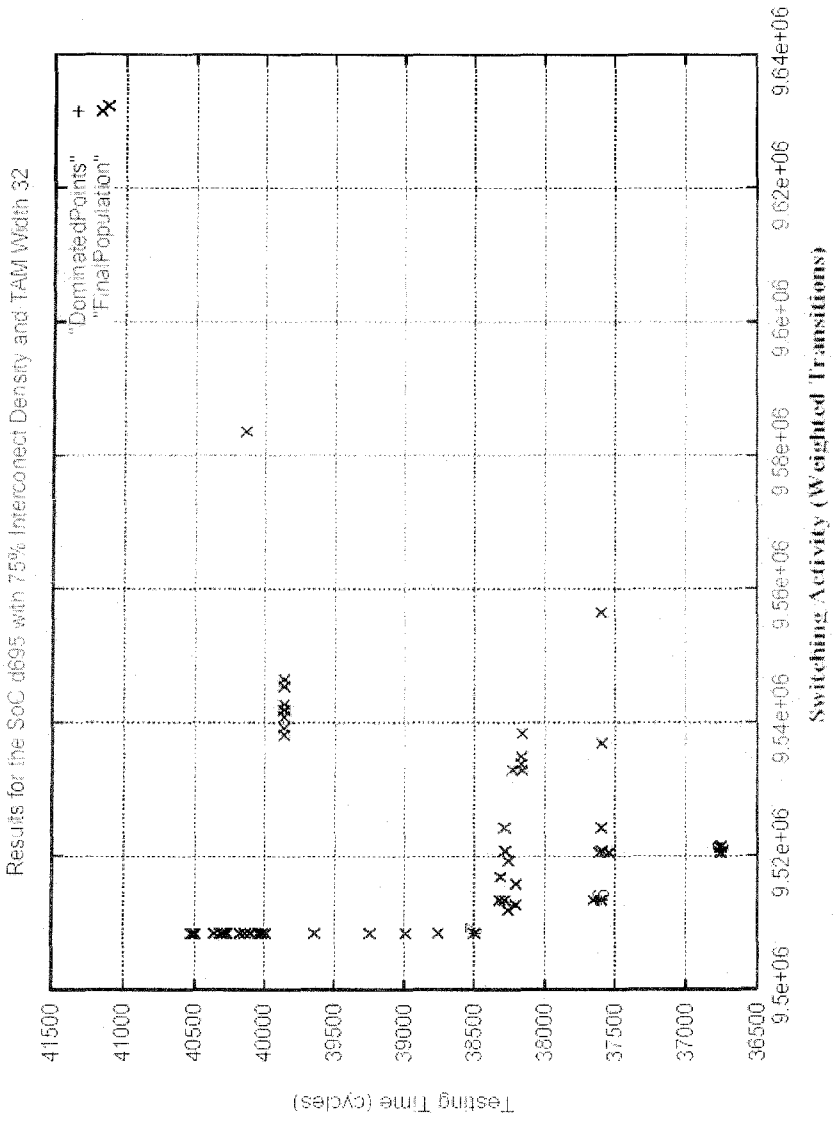


Fig.4.4: Results for SOC d695

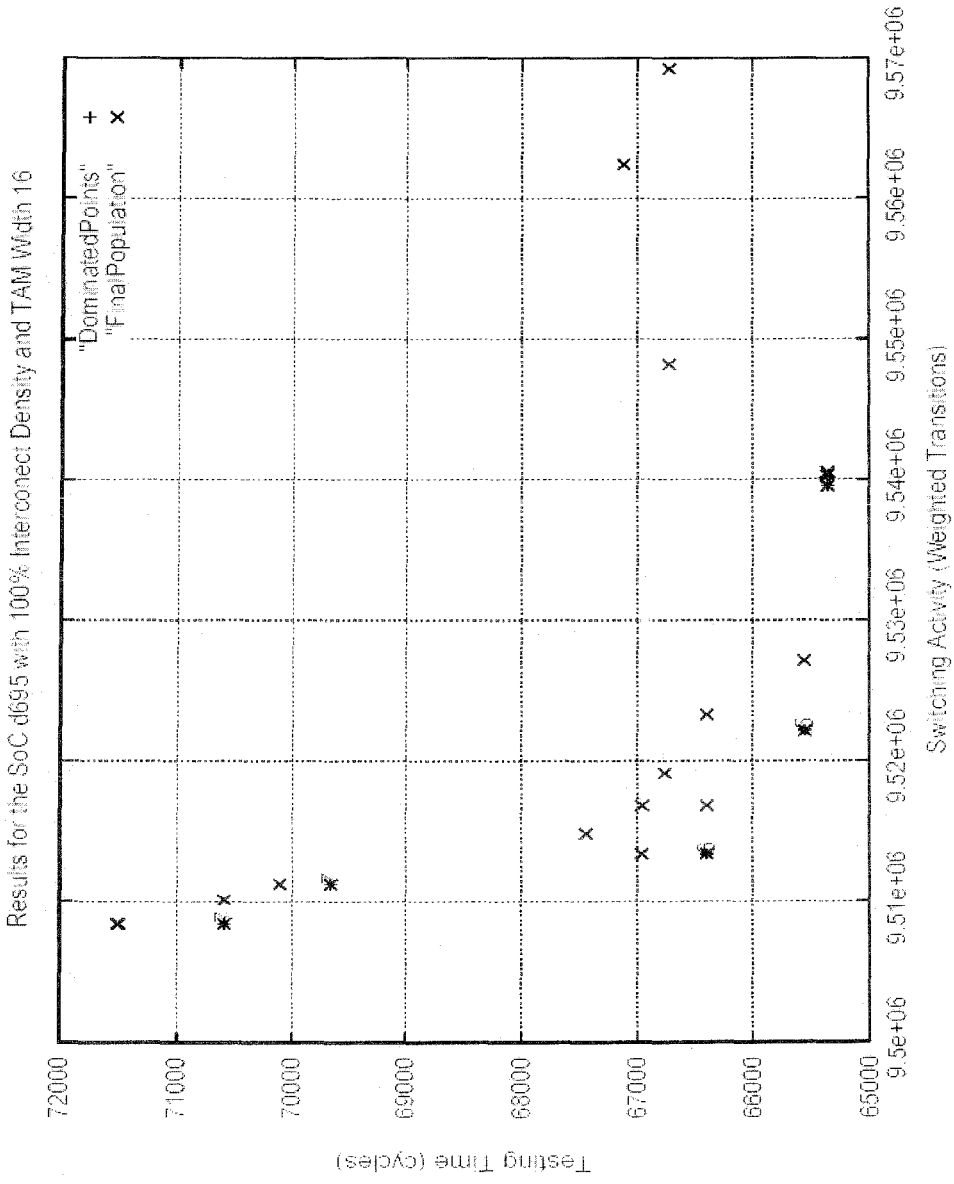


Fig.4.5: Results for SOC d695

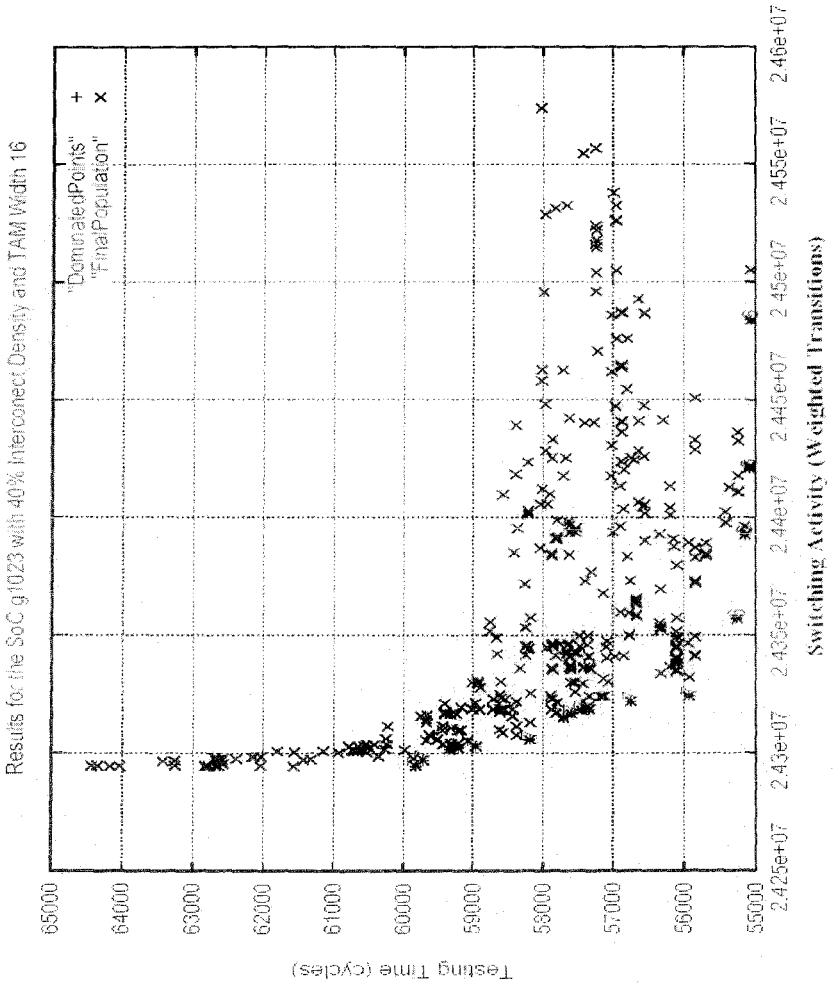


Fig.4.6: Results for SOC g1023

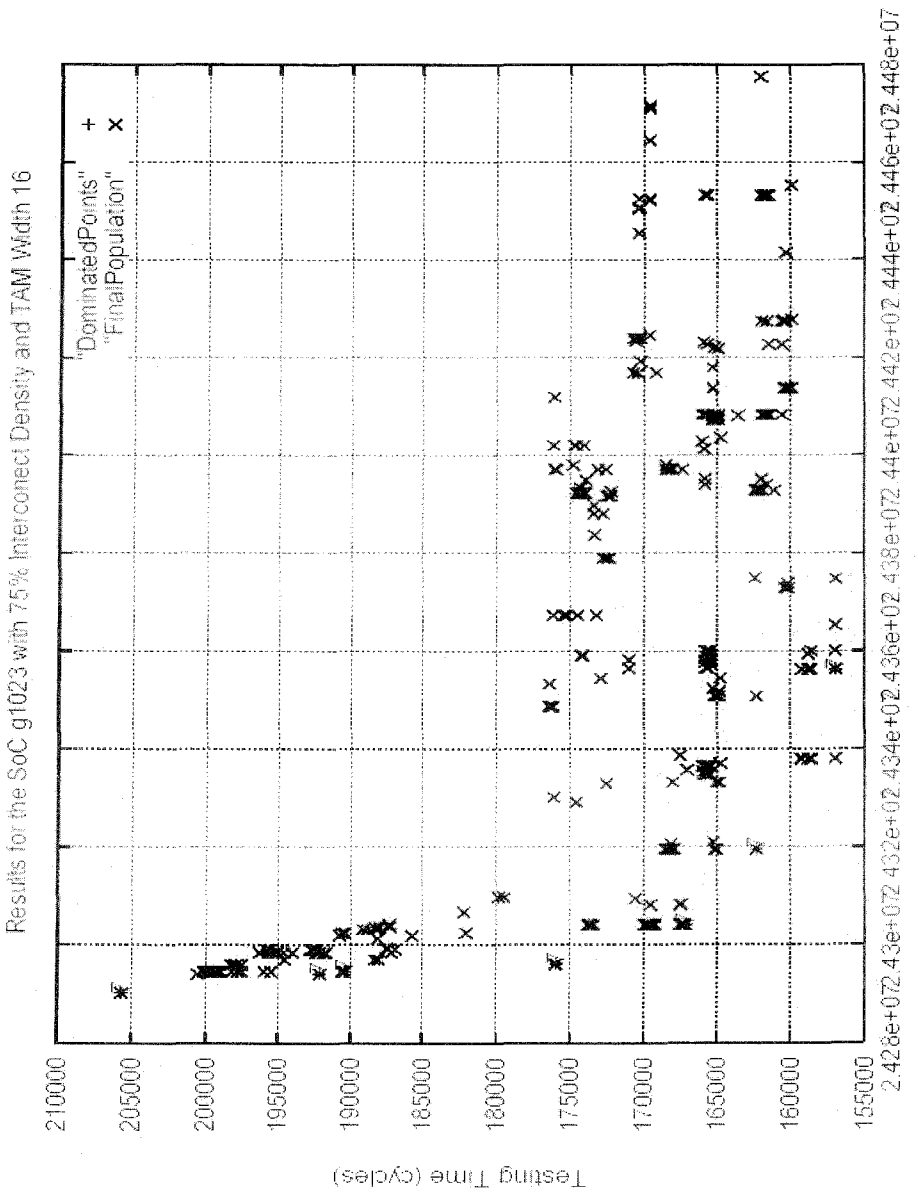


Fig.4.7: Results for SOC g1023

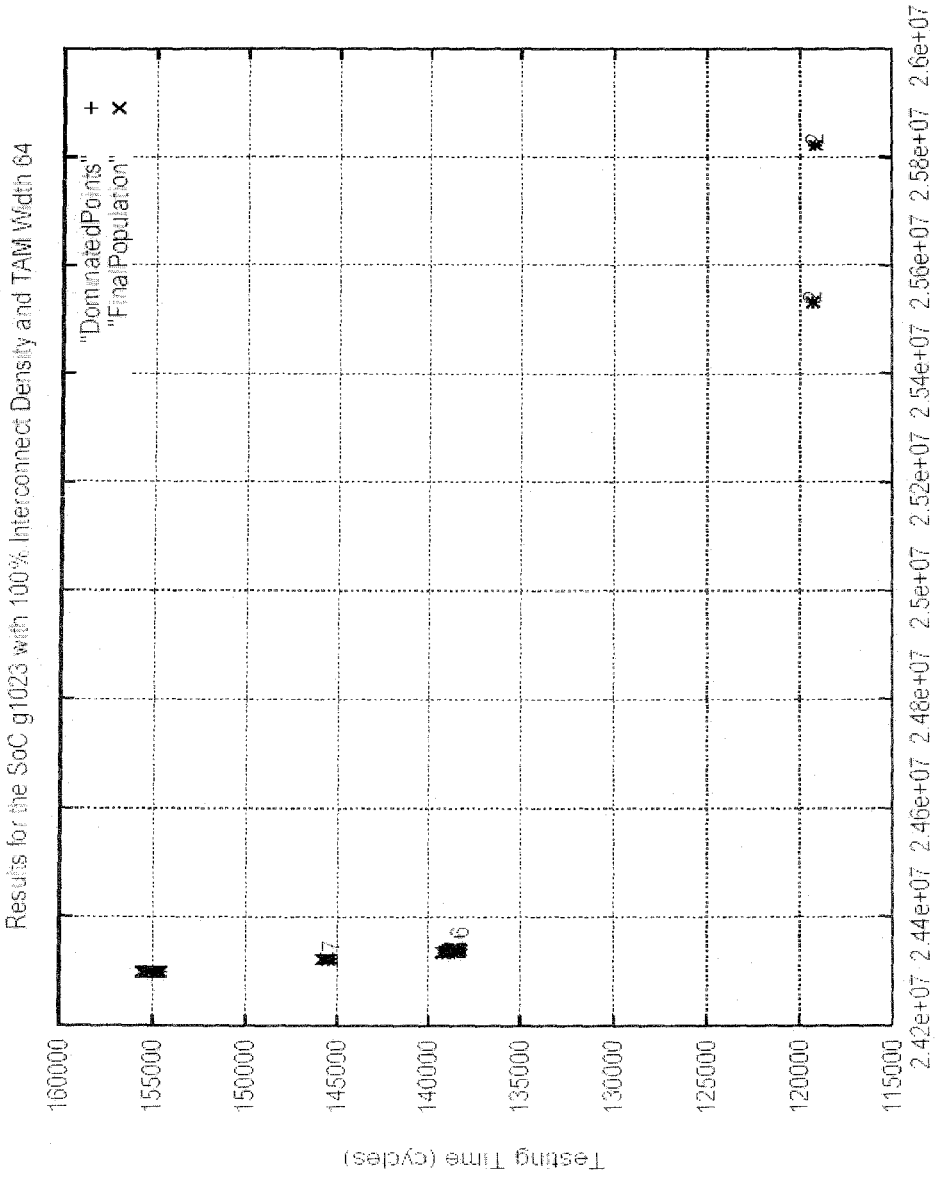


Fig.4.8: Results for SOC g1023

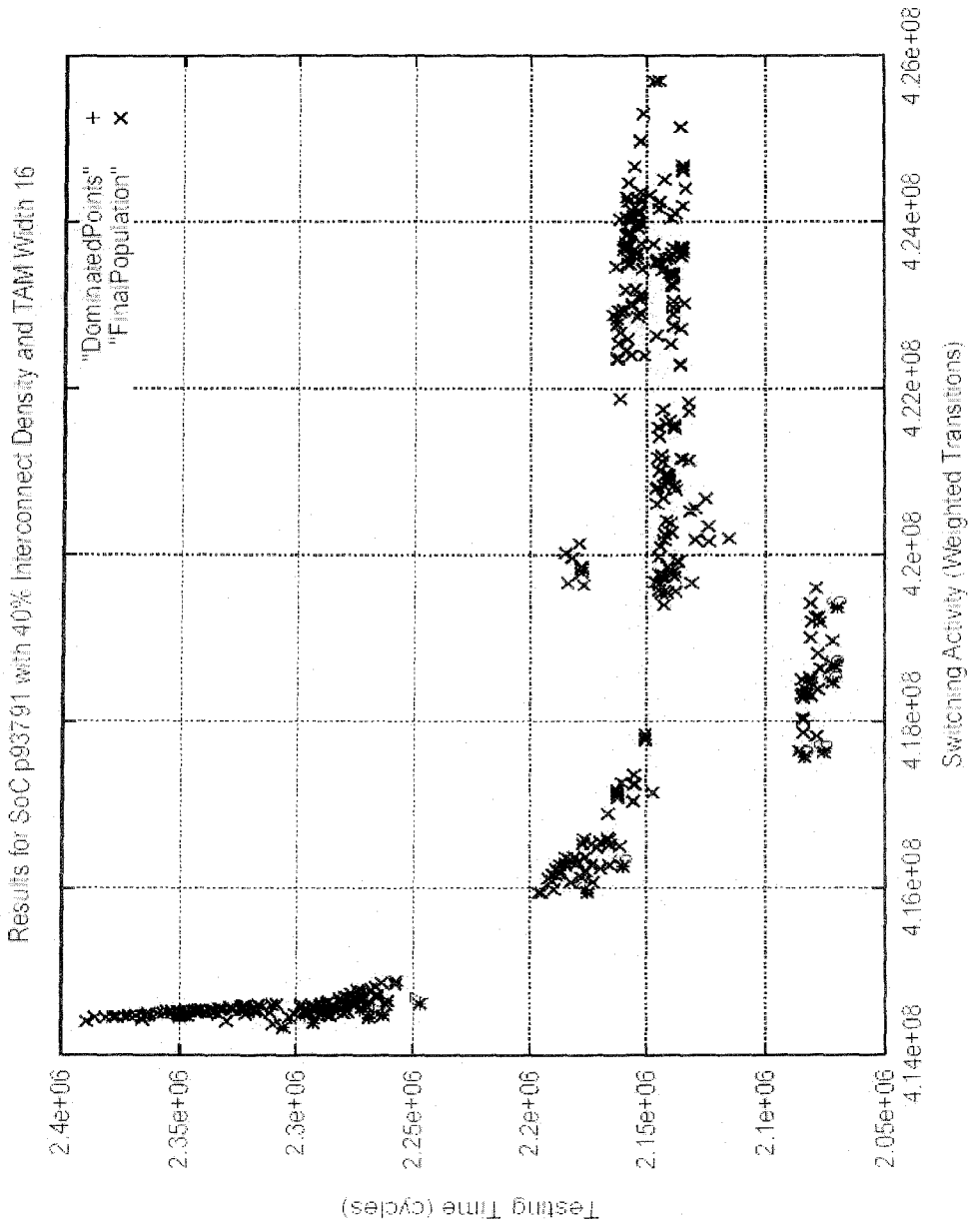


Fig.4.9: Results for SOC p93791

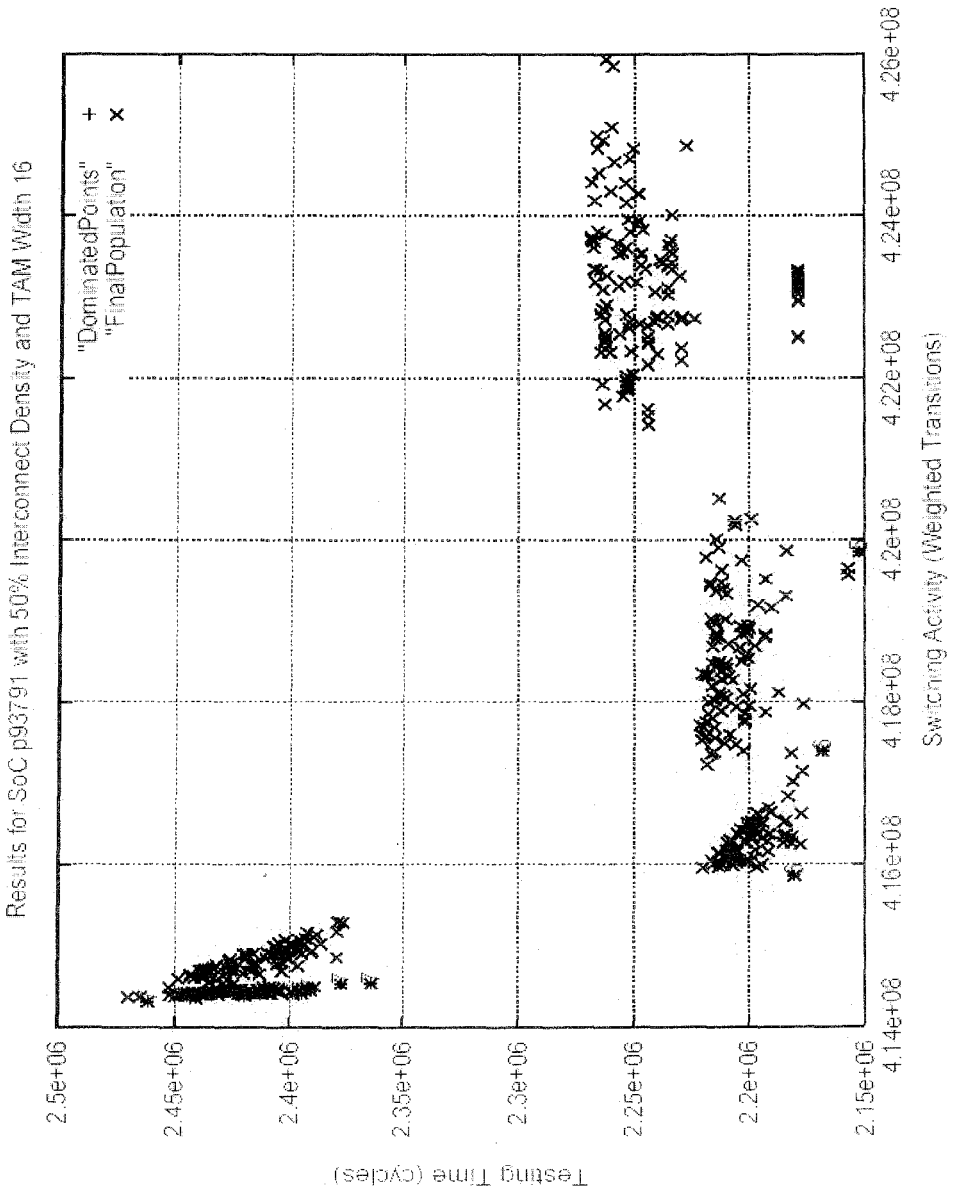


Fig.4.10: Results for SOC p93791

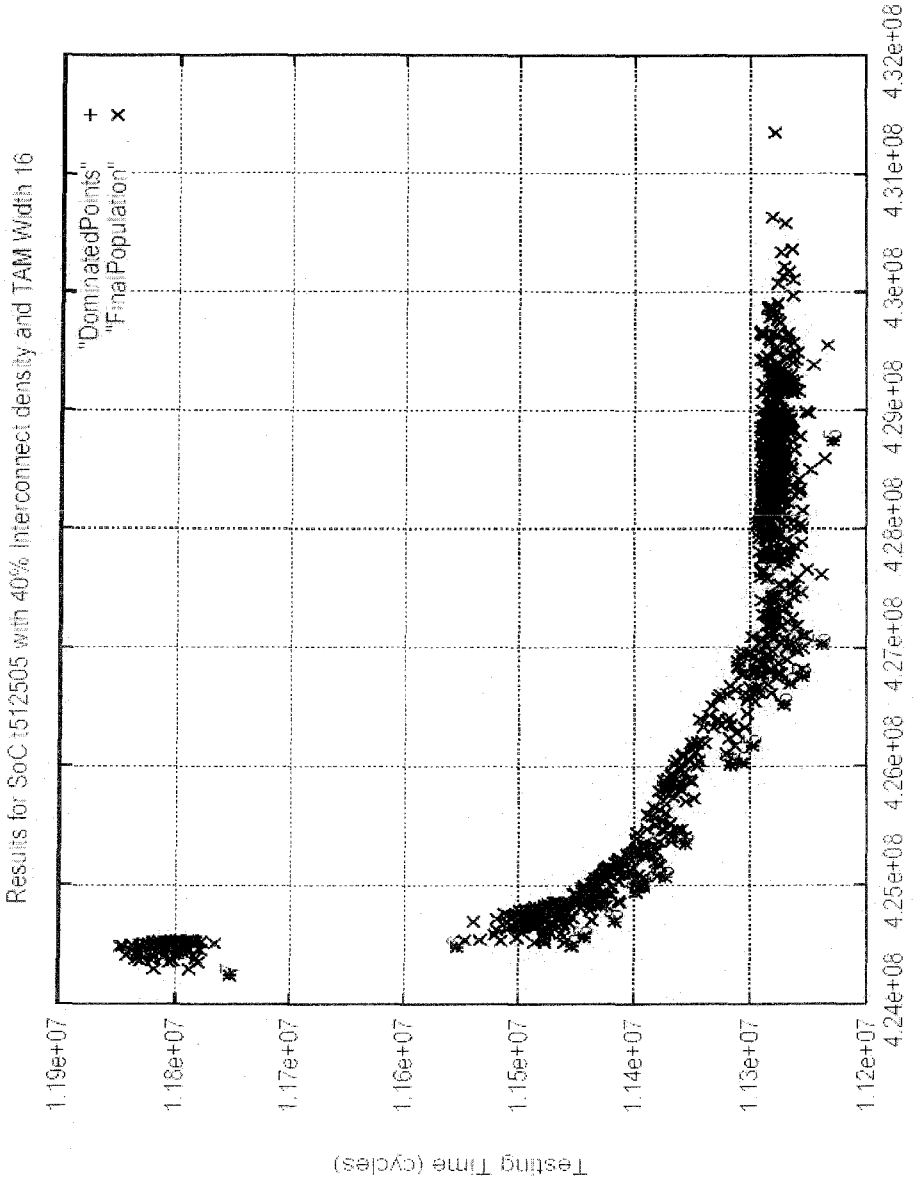


Fig.4.11: Results for SOC t512505

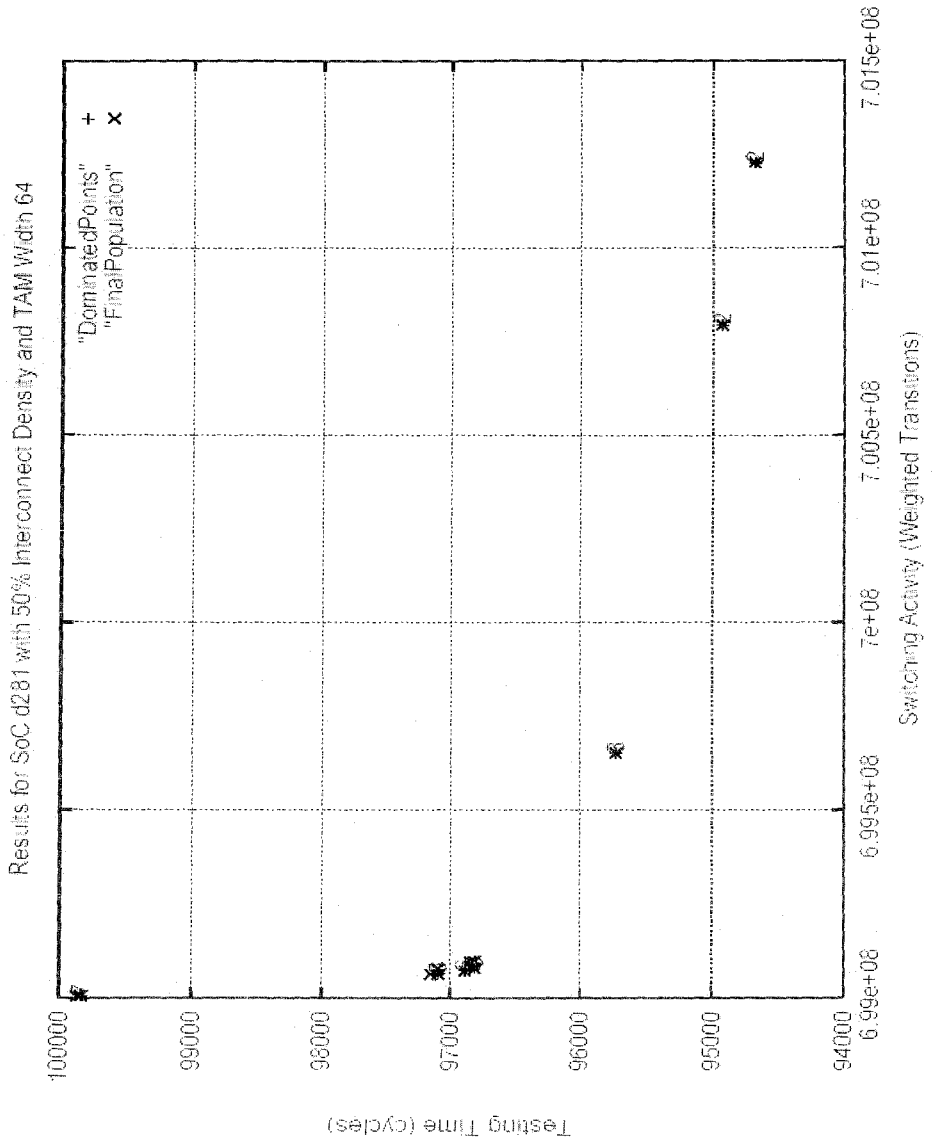


Fig.4.12: Results for SOC d281

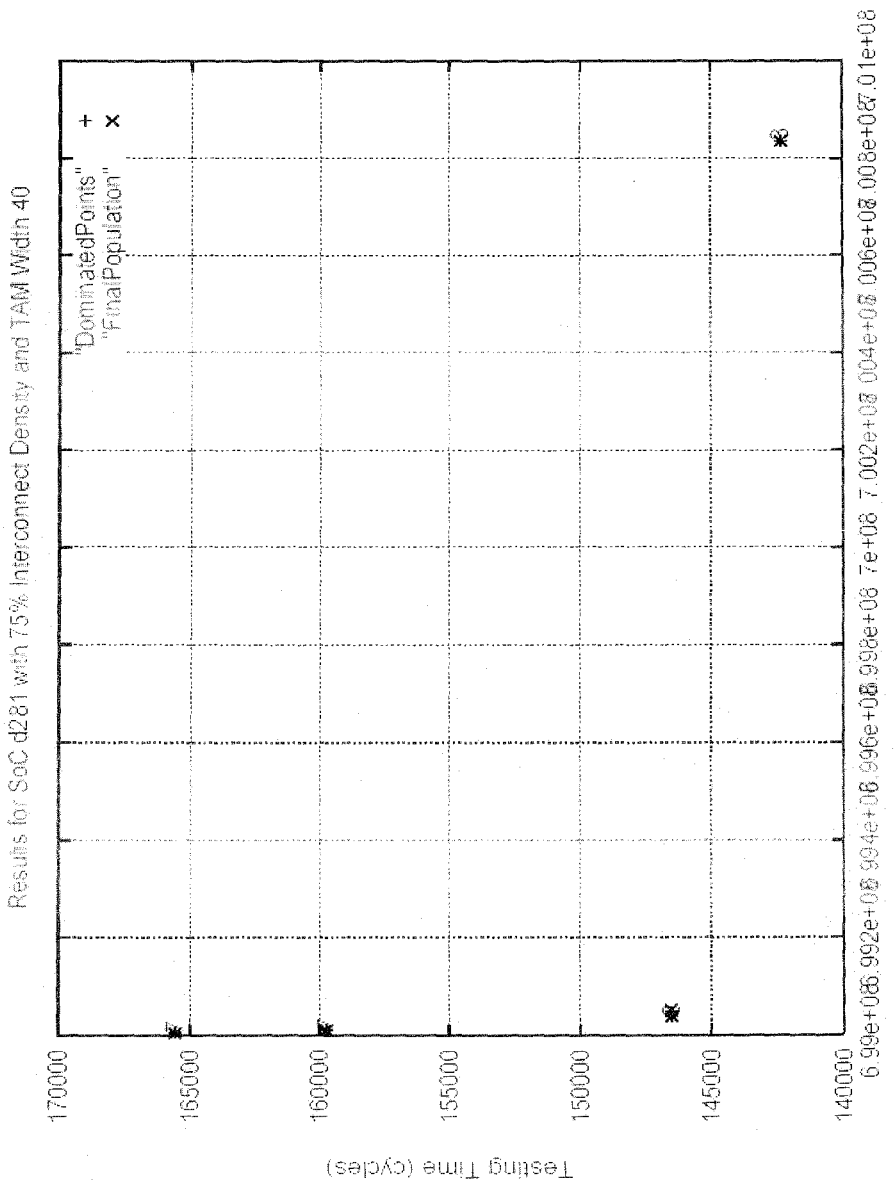


Fig.4.13: Results for SOC d281

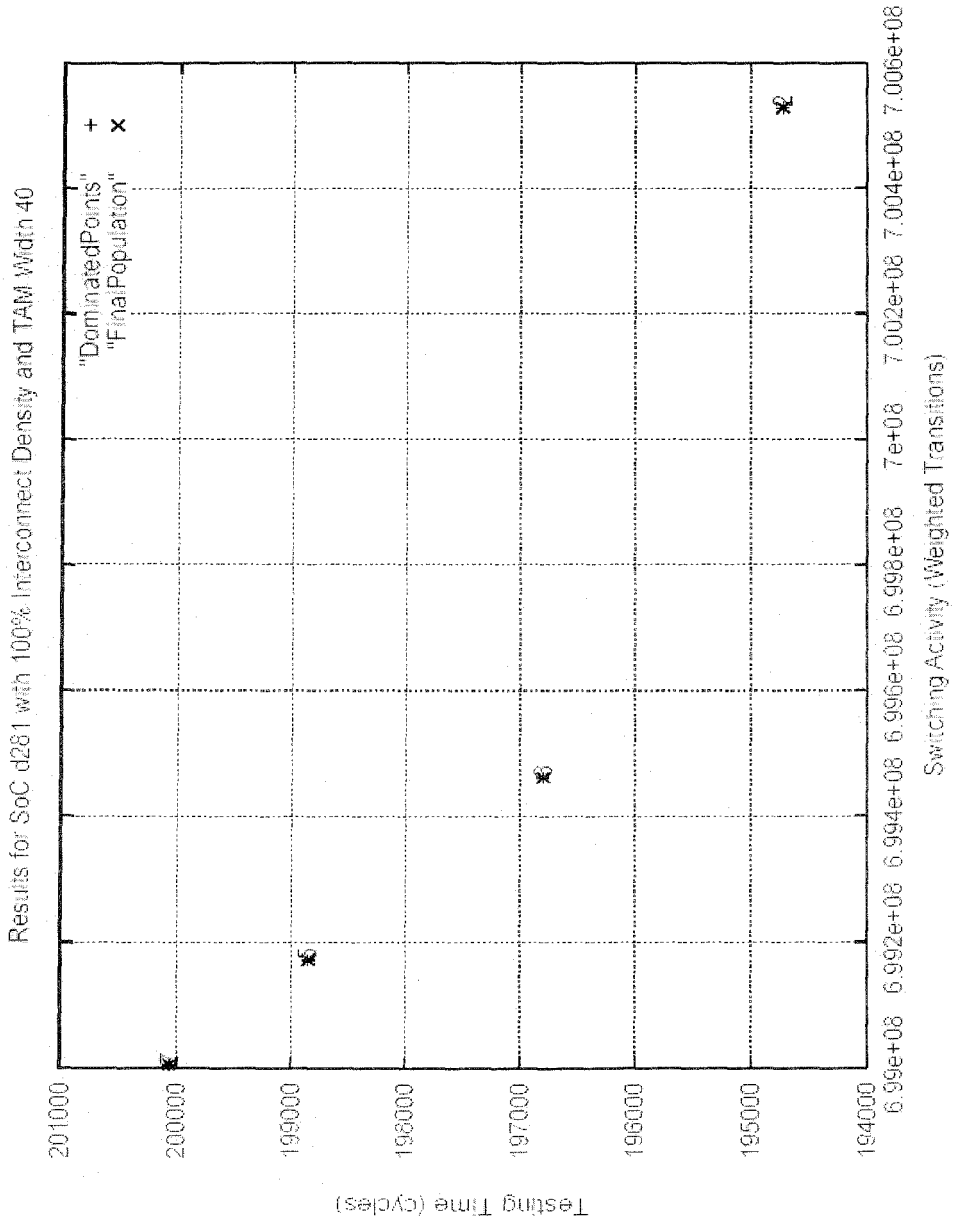


Fig.4.14: Results for SOC d281

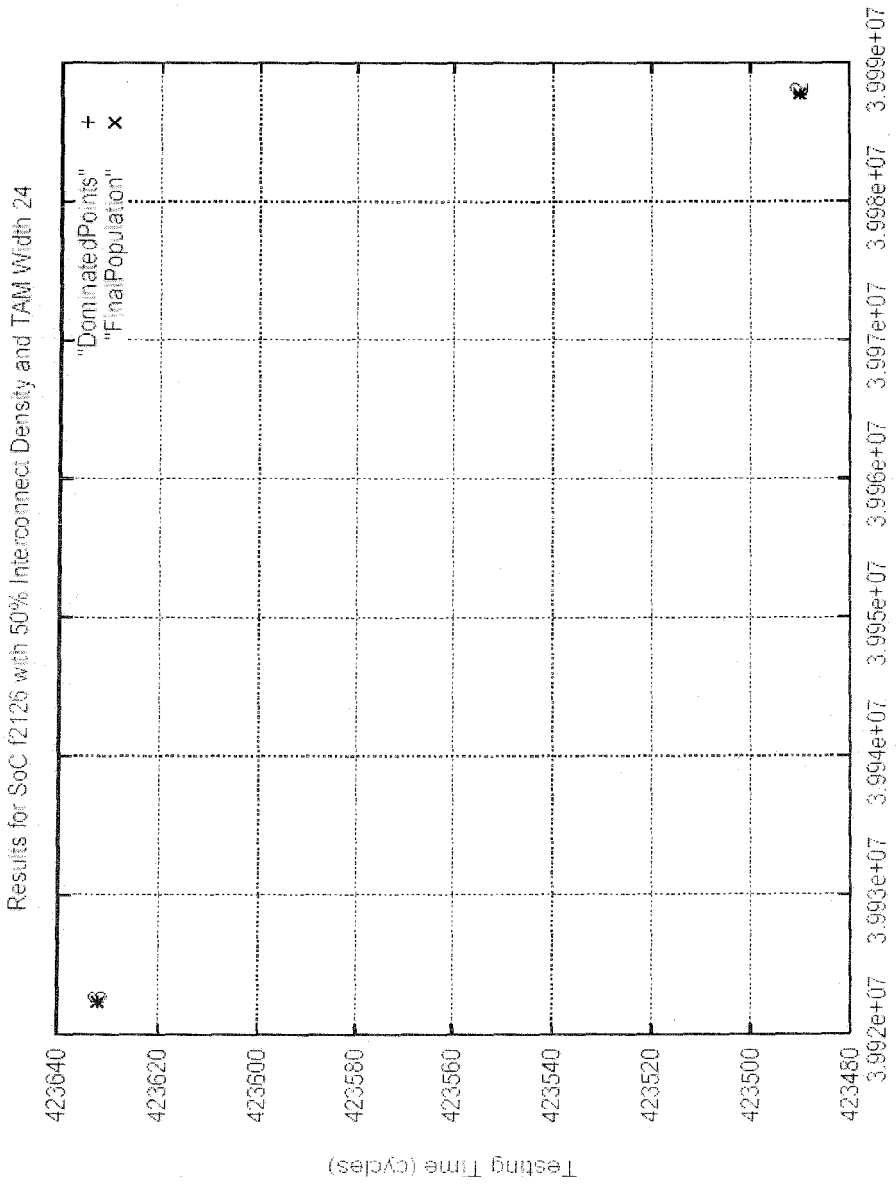


Fig.4.15: Results for SOC f2126

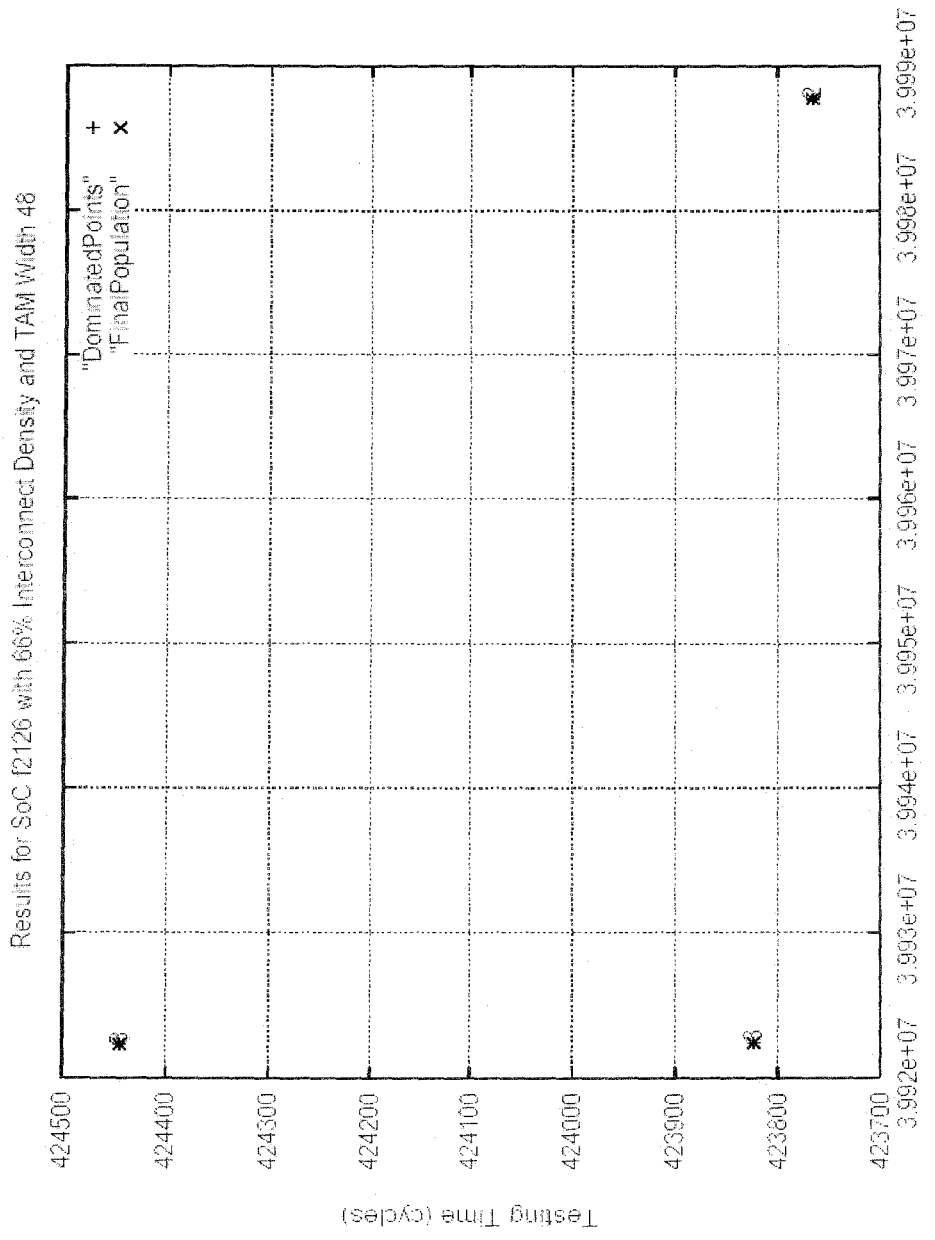


Fig.4.16: Results for SOC f2126

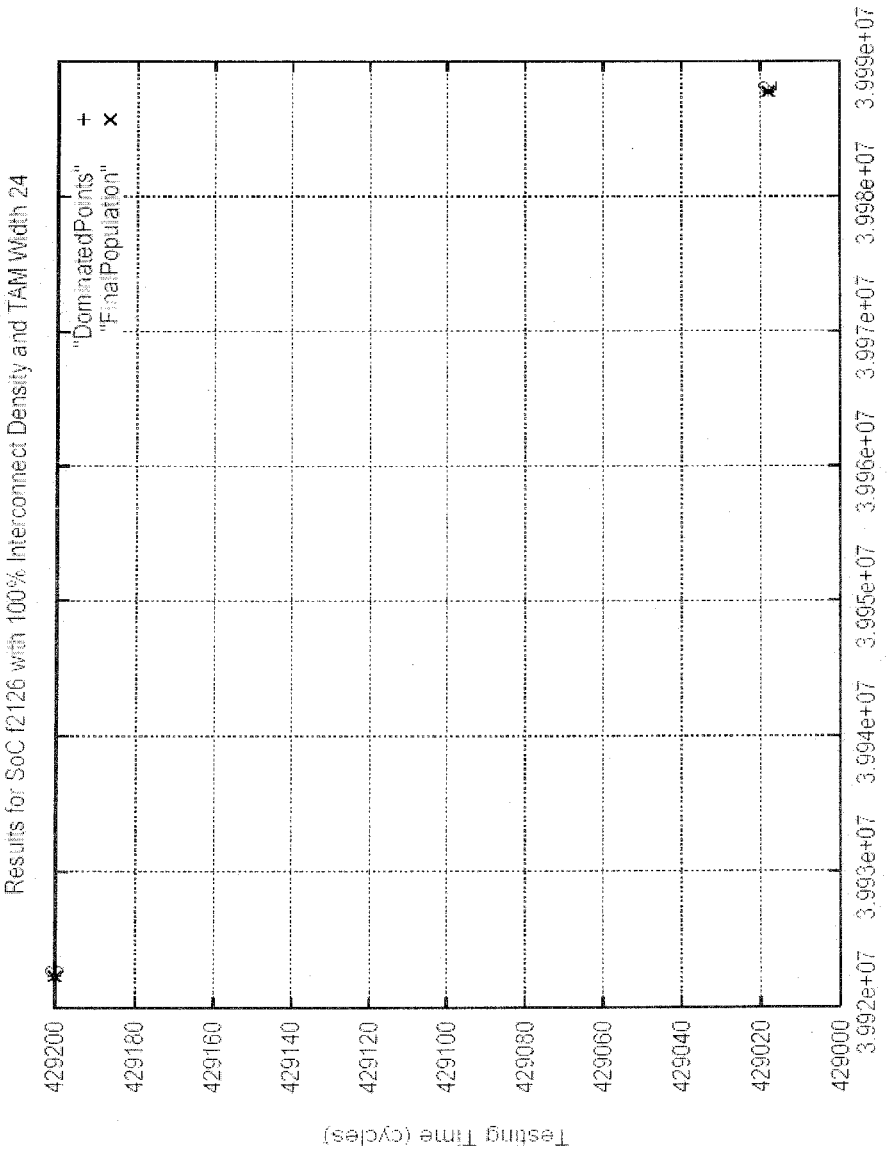


Fig.4.17: Results for SOC f2126

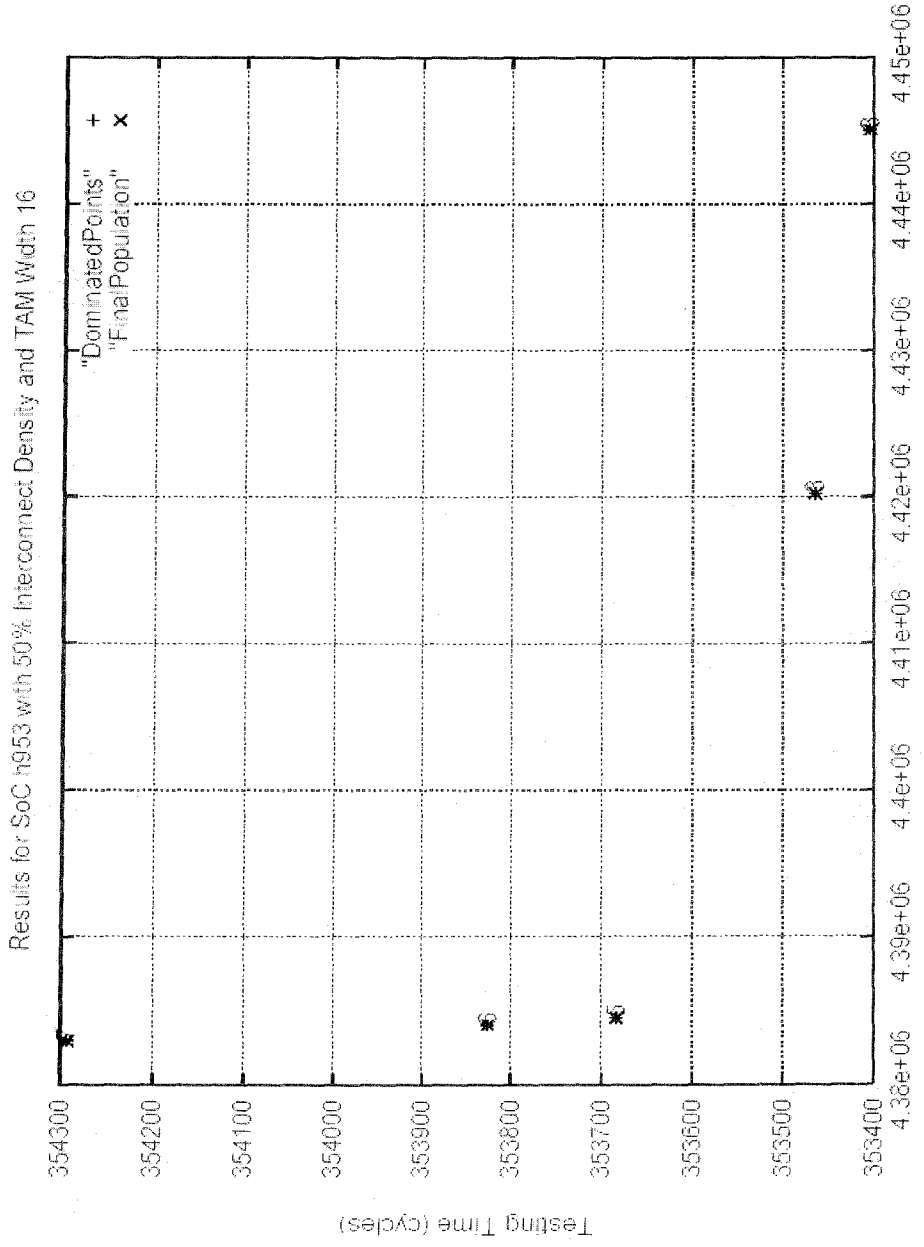


Fig.4.18: Results for SOC h953

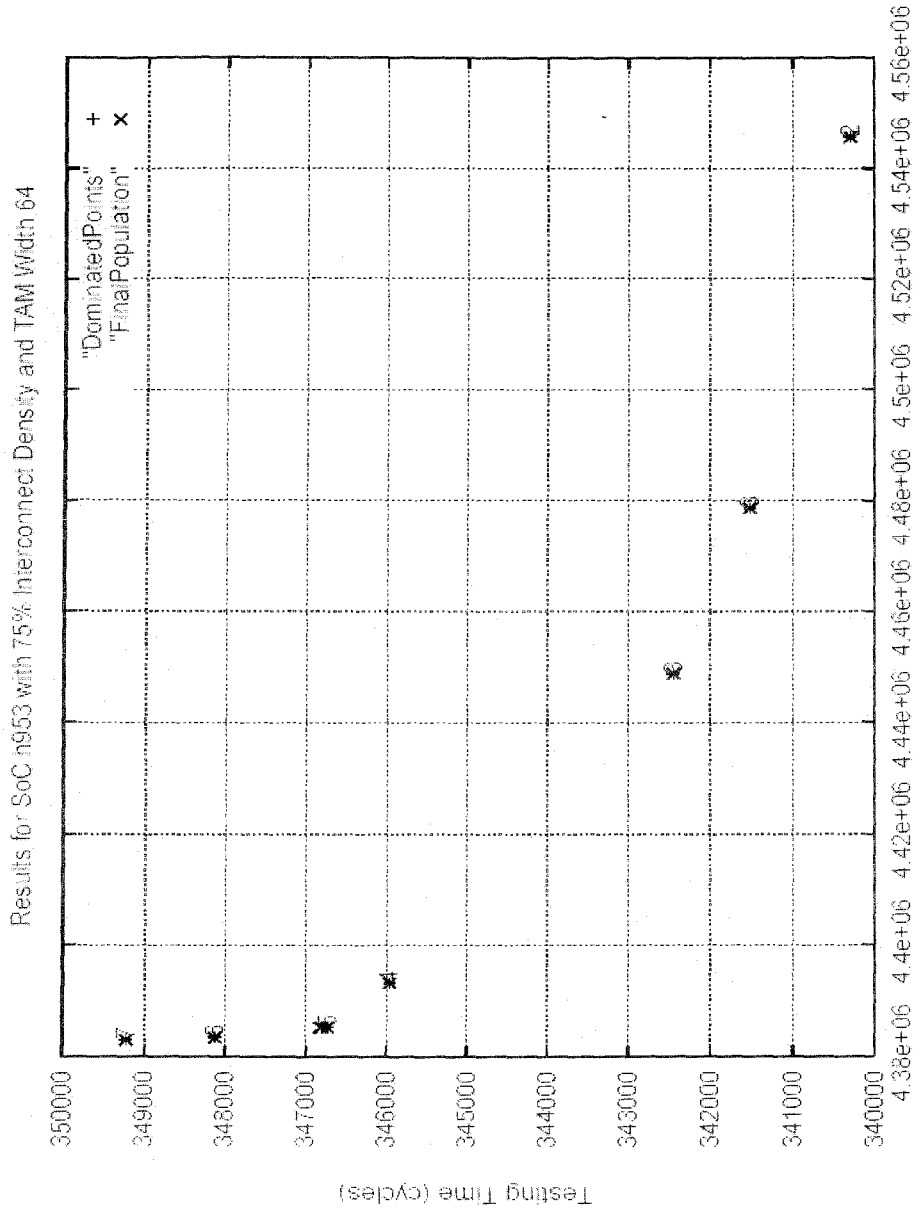


Fig.4.19: Results for SOC h953

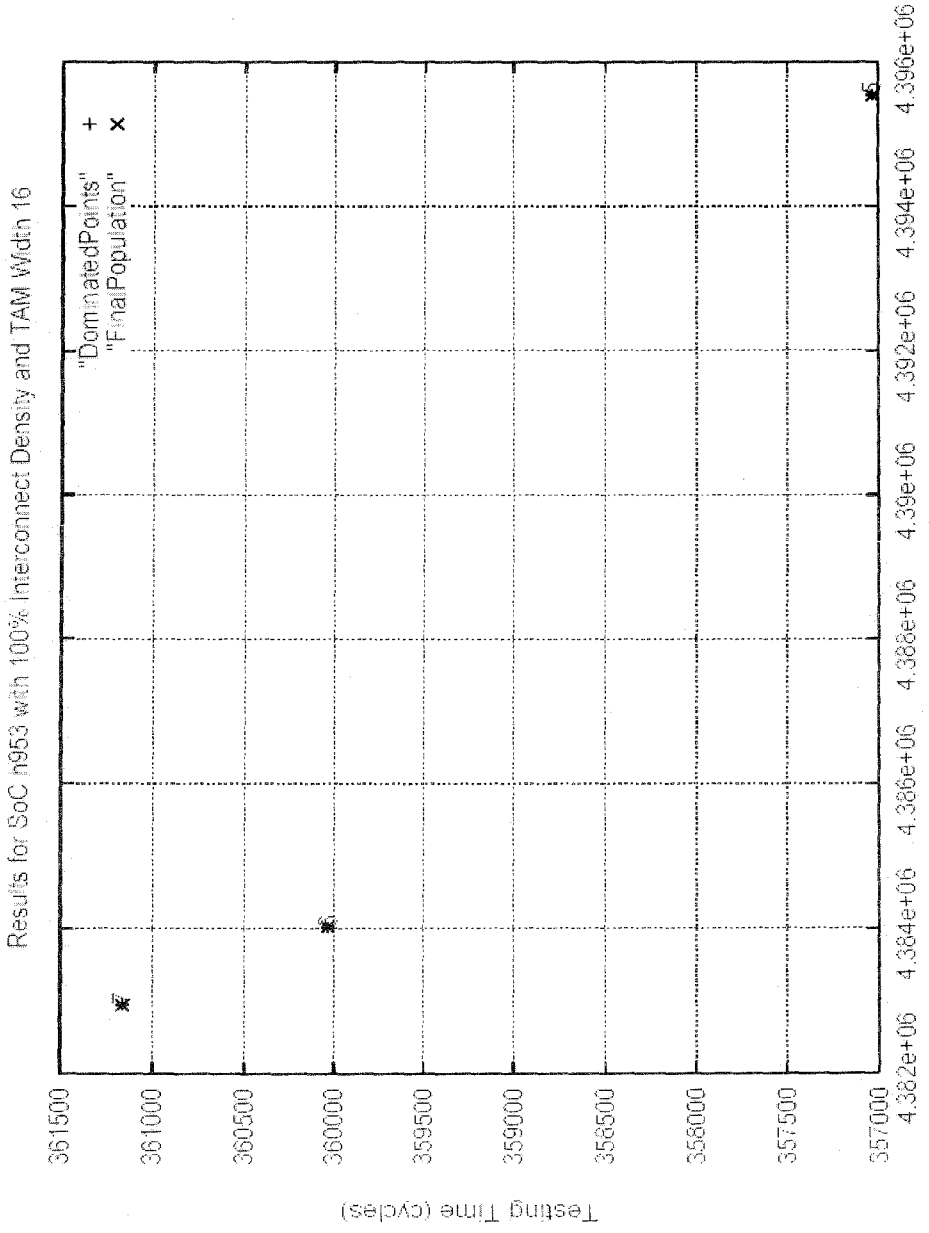


Fig.4.20: Results for SOC h953

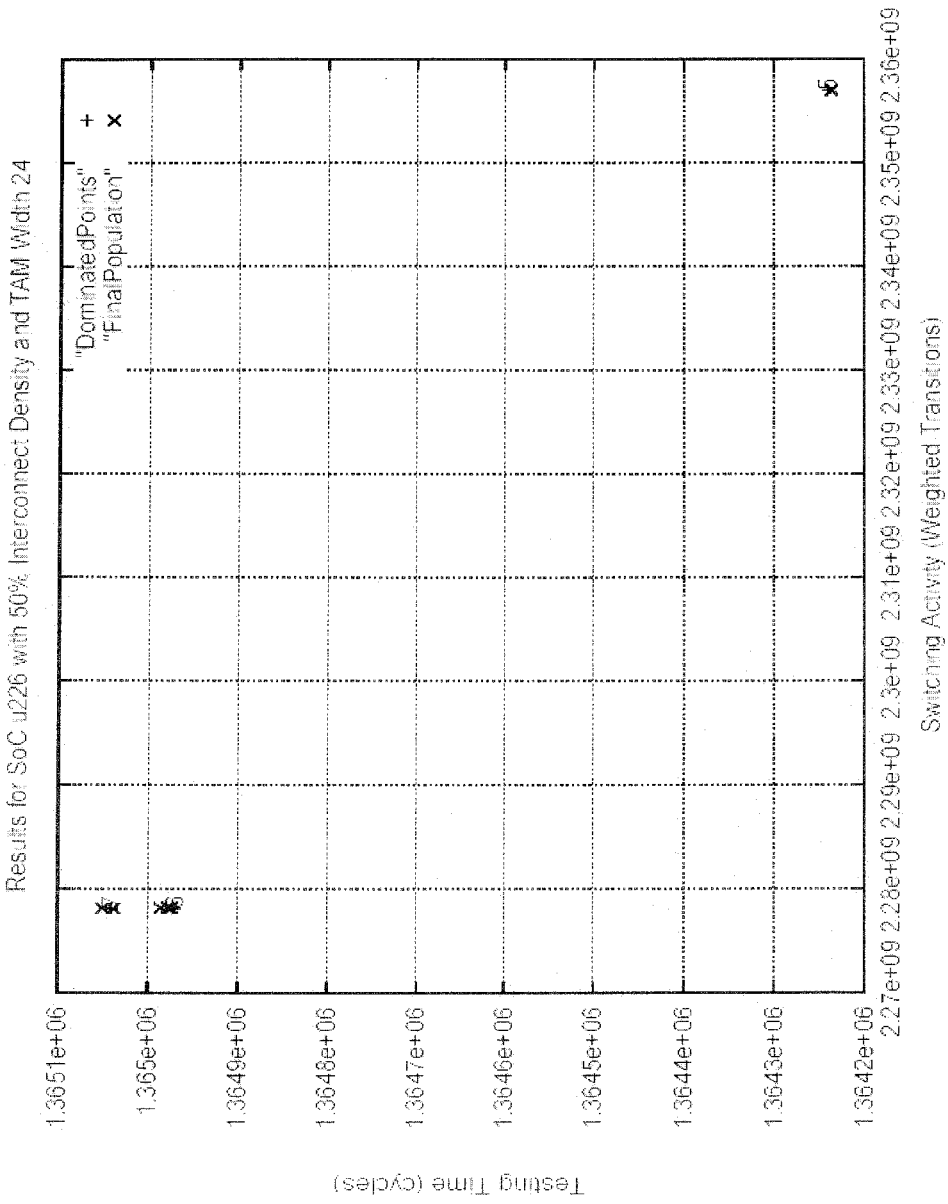


Fig.4.21: Results for SOC u226

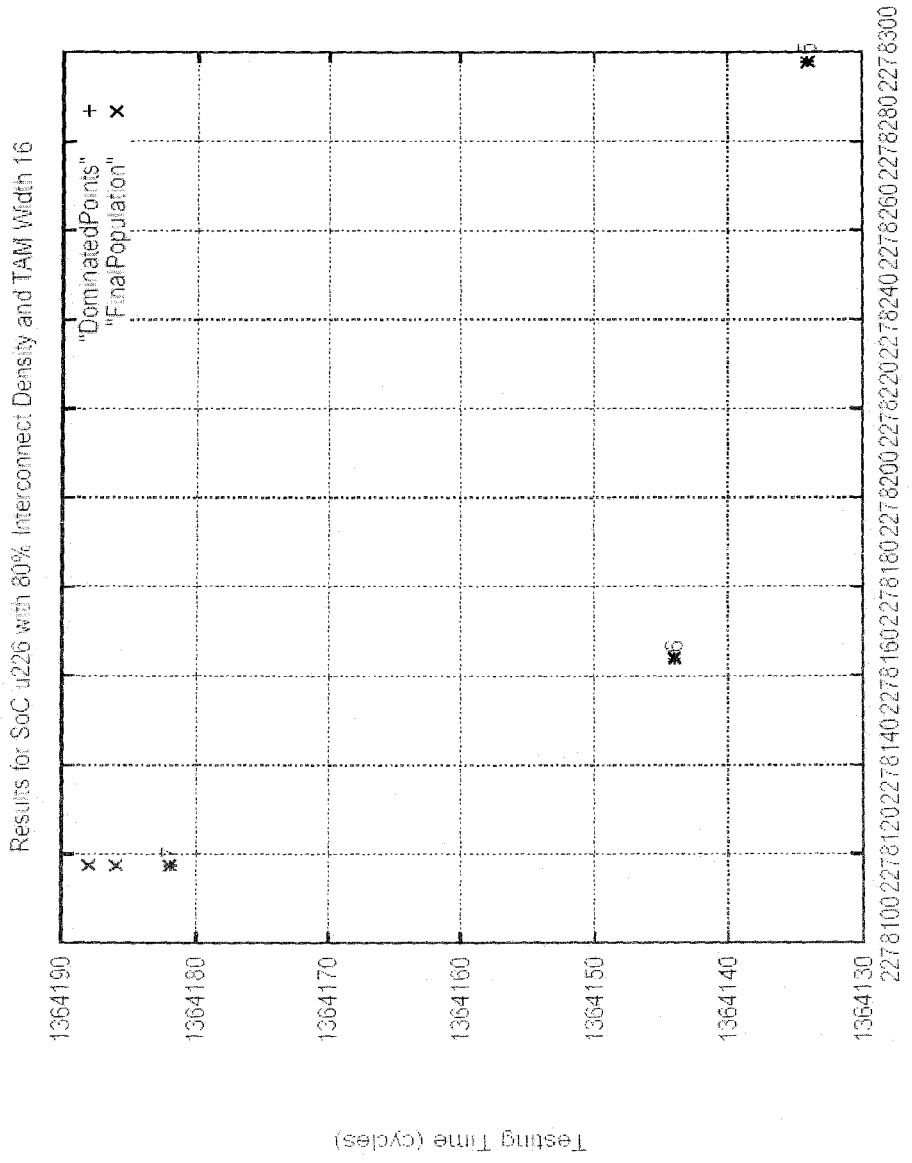


Fig.4.22: Results for SOC u226

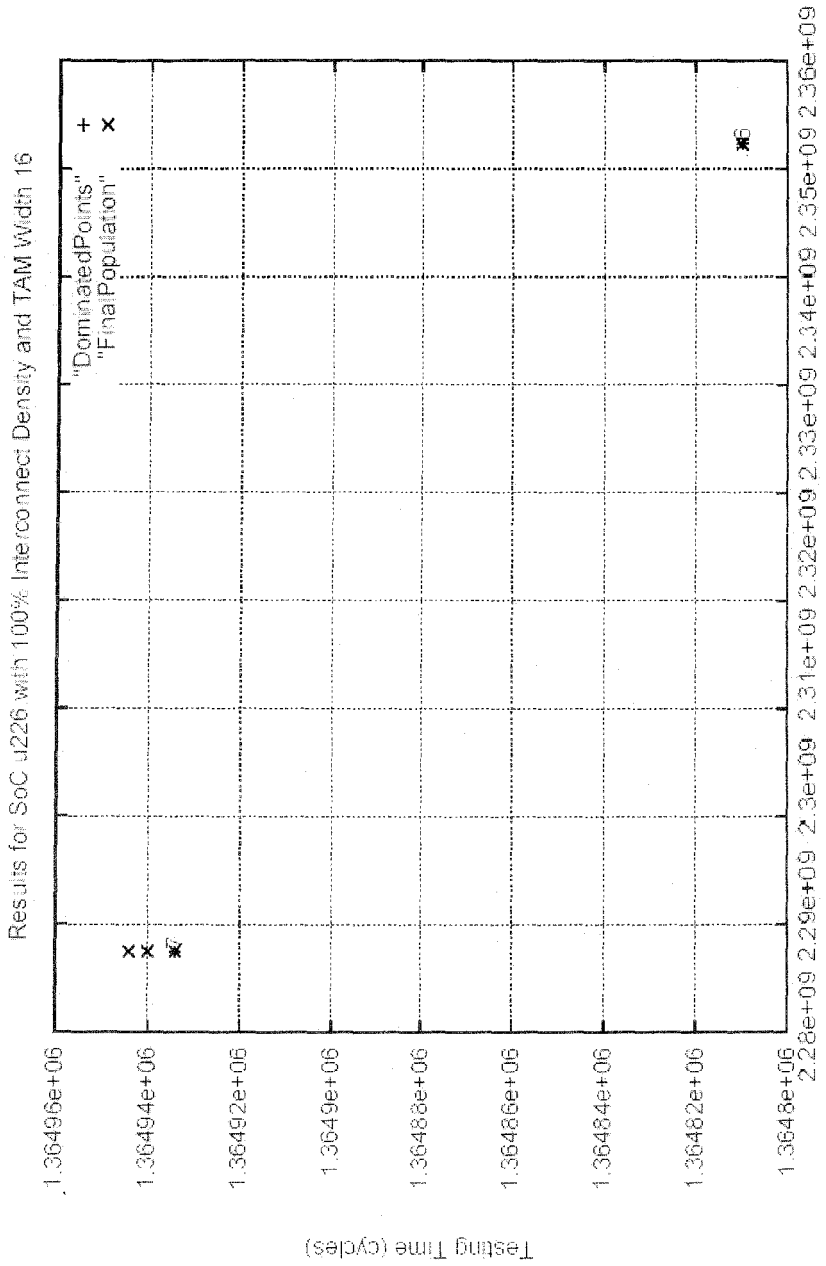


Fig.4.23: Results for SOC u226

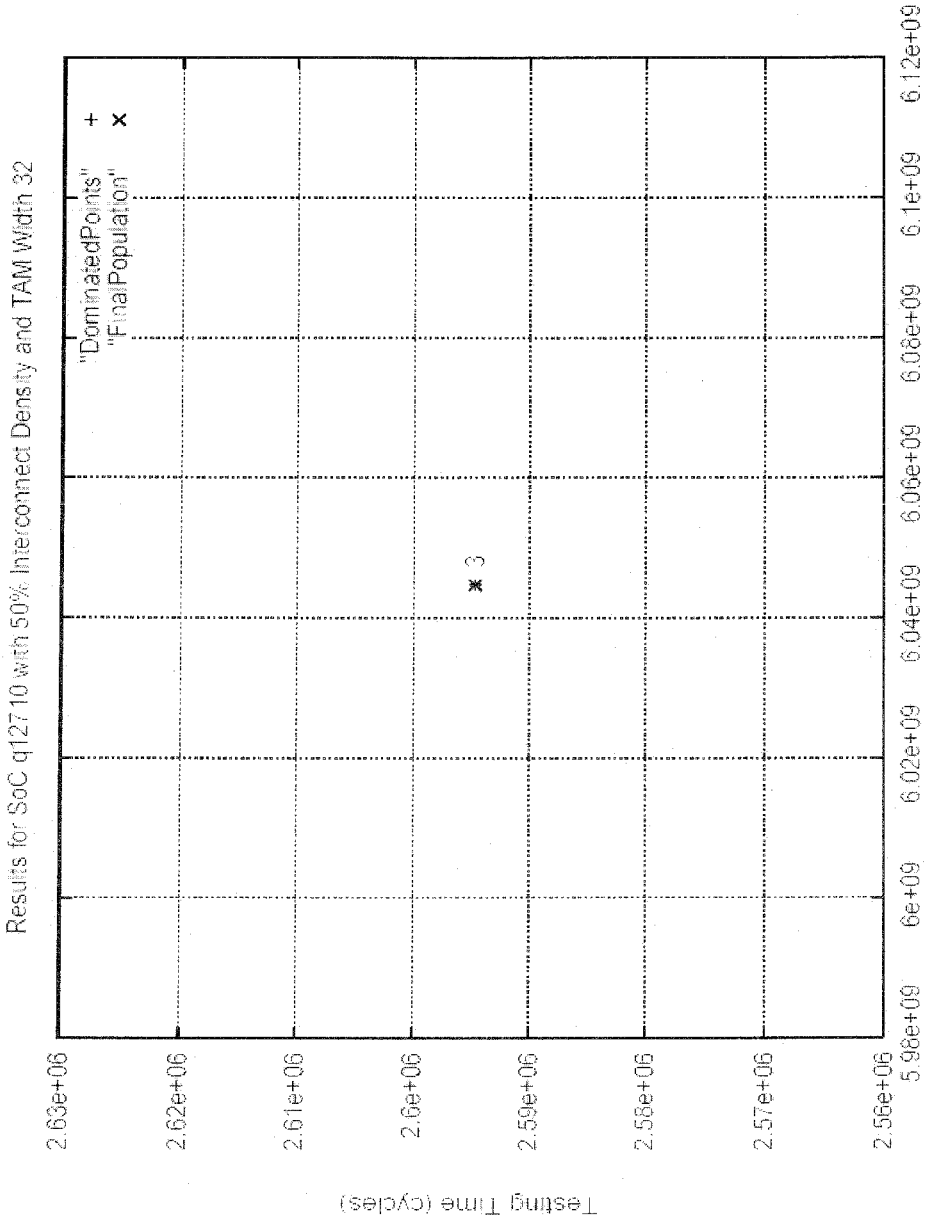


Fig.4.24: Results for SOC q12710

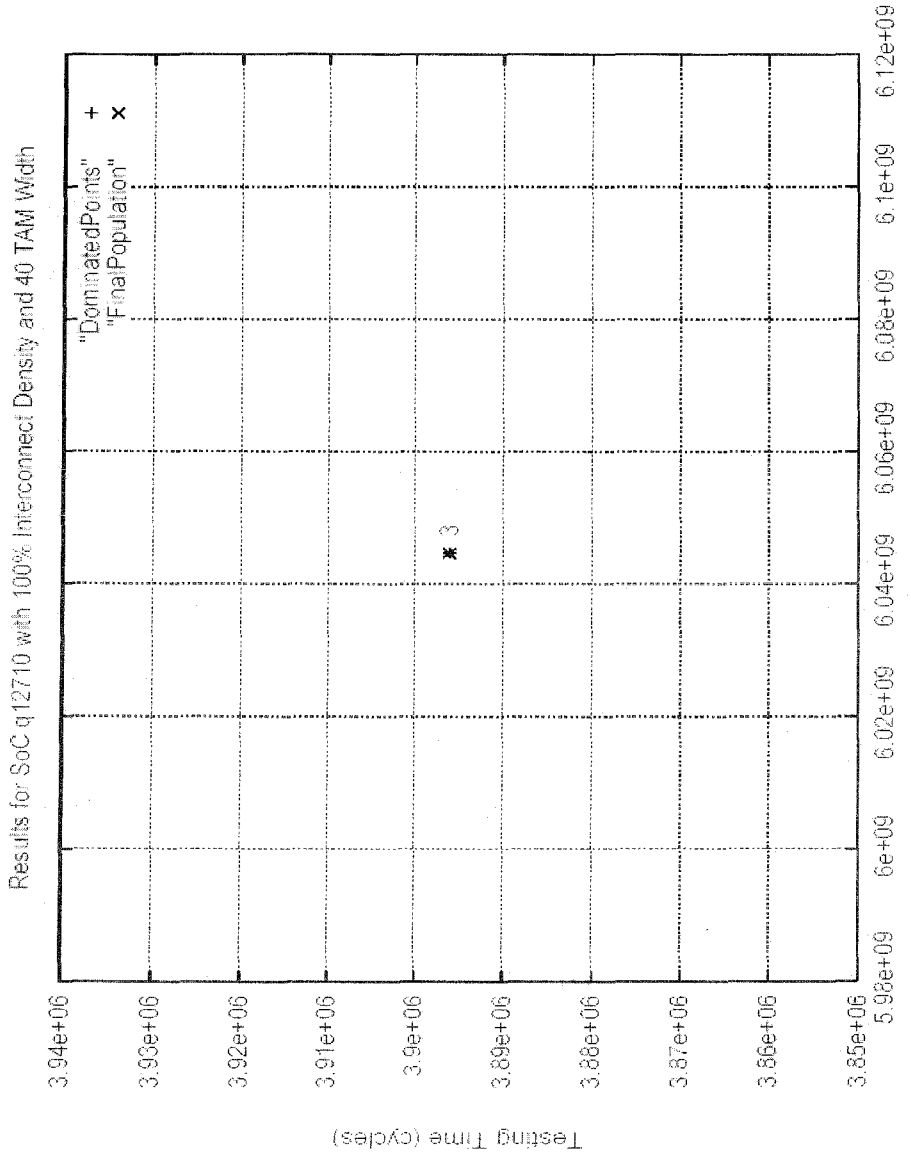


Fig.4.25: Results for SOC q12710

Chapter 5

Scheduling – A Rectangle Packing Approach

In the last two chapters, we have seen test scheduling approaches based on TAM partitioning. Huang et al [79] first suggested a different approach in which the TAM lines are not physically partitioned. A test wire used as a part of test lines delivering patterns to a core-under-test, can become associated with a different set of wires while testing another one. This formulation gives rise to a bin-packing approach for test scheduling. The situation has been shown in Fig 5.1, in which TAM wires are assigned to four test sets over time. For a certain period of time, each test is assigned to some TAM wires. The problem that we concentrate on is, how to assign a start time, an end time and the set of TAM wires for each test in such a way that total test time is minimized. Another important issue in testing that has come up recently is about test power minimization. This is required as most of the chips today come up with a power budget. Thus, excessive power dissipation during test and the associated heat generated may cause permanent damage to the chip. Various strategies have been proposed in literature to reduce the test power. Based on these observations and due to the NP-hard nature of the TAM design algorithm, we have used a genetic algorithm (GA) based approach to solve the SOC test scheduling problem.

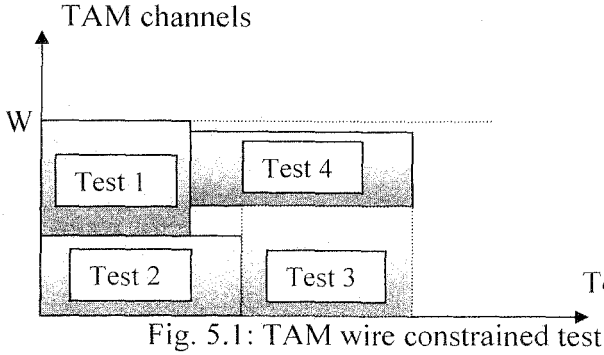


Fig. 5.1: TAM wire constrained test scheduling

The cores in an SOC are normally fitted with wrappers. The wrapper for a core facilitates the test application process. It can isolate the module from its surroundings and provide switching functionality between functional access to the modules and test access through the TAM. Though some cores may be equipped with wrappers, while others may not, in this work we consider only the wrapped cores. It is to be noted that the test time of a core depends on the length of the wrapper chains. The scan chains and the wrapper cells are configured to make a wrapper chain. An increasing number of wrapper chains reduce the test time due to the reduction in length of the wrapper chain. However, it takes more TAM wires and vice versa. We can formally describe the problem as follows.

Let the SOC design consist of N cores, and each core C_i $(1 \leq i \leq N)$ has

- n_i input terminals,
- m_i output terminals,
- bi-directional I/Os,
- S_i scan chains and
- for each scan chain k, the length of the scan chain (number of flip-flops) $l_{i,k}$.

Also assume that maximum peak power for each core during testing is given. Let, the total width of TAMs be W and each core must be tested with P_i patterns. So, the overall problem that we have to solve is as follows.

Given a set of N cores, their specific test parameters, the number of I/O pins for an SOC, maximum allowable peak power dissipation POW and peak power dissipation for each core, design the test schedule along with wrapper designs for all wrapper-based cores such that overall testing time is minimized and the peak power during testing never exceeds POW.

There are basically two steps in our approach to solve the problem. First, we generate possible optimized wrapper configurations for each core under specified TAM width. In the next step, we solve the test scheduling problem using the sets of optimized wrapper solutions under the maximum-allowable TAM width and power constraint. In our work, we consider the hard cores, for which, the number and length of the module-internal scan chains are fixed and cannot be changed any more while designing the SOC-level test architecture.

5.1 Core Wrapper Design

A test wrapper is the interface between the TAM and the core. Since larger cores typically have hundreds of terminals and the total number of TAM channels available depends on the limited number of SOC pins, wrappers facilitate test width adaptation when the TAM width is not equal to the number of core terminals. To design the wrapper for cores with internal scan chains, we have used the *Design_wrapper* algorithm proposed in [50]. To calculate test time T, for a wrapper we have used Eqn. 3.1.

Using the wrapper design method, for each core we can generate a set of wrapper configurations with the TAM wire usage of 1 to W, where W is the maximum number of TAM channels allocated to test the SOC. Hence, each wrapper configuration can be considered as a rectangle with width equal to the test time and height corresponding to the number of TAM wires allocated. So, each configuration is represented by a tuple $(w_{ij}, T(w_{ij}))$ - where, core i has been assigned a TAM width w_{ij} resulting in a test time $T(w_{ij})$ $(1 \leq j \leq W)$. From all the wrapper configurations for a core i, a smaller set of wrapper configurations need to be considered in the test scheduling. It is based on pareto-optimal design principle, where for a range of TAM widths, test time remains unchanged. Obviously, only pareto-optimal points are of interest since they make use of the lowest possible number of TAM channels to reach a certain test time. Table 5.1 shows the rectangles created for core 2 of SOC p93791.

TAM width	Testing time		TAM width	Testing time
1	7906		9	1155
2	4049		10-11	963
3	2891		12-13	962
4	2120		14-17	770
5	1734		18-19	769
6	1541		20-39	577
7	1348		40-64	385
8	1156			

Table 5.1: Rectangles created for core 2 of SOC p93791

5.2 Test Scheduling Problem

Suppose an SOC with N cores is to be tested using W TAM wires. Each core C_i $(1 \leq i \leq N)$ is represented by a set of wrapper configurations R_i. Each wrapper configuration is represented by a pair $(w_{ij}, T(w_{ij}))$, where w_{ij} stands for the width of the j-th wrapper configuration for core C_i and $T(w_{ij})$ is the test time of core C_i with wrapper width w_{ij}. Also, for each core, peak power during testing POW_i, is assumed to be available. So the objective is the assignment of core wrapper pins to the pins of SOC and for getting the test starting time and finishing time for each core such that overall test time is minimized satisfying power constraint.

This problem can be transformed into the well-known rectangle-packing problem, in which the SOC is represented by bin of width W and a set of R_i SOC wrappers for each core represented by a set of R_i rectangles with rectangle j having height w_{ij} and width $T(w_{ij})$. We want to choose one rectangle from each set of rectangles R_i and pack all the rectangles in the bin, so that width of the bin is minimized.

In this work, we treat this test scheduling problem as a collection of two different problems. First, we consider the test scheduling problem where no power constraint is considered. In the second one we impose the power constraint on test scheduling algorithm for the first problem and it is verified that for any time instant maximum allowable peak power POW (SOC power budget) will not be exceeded. To evaluate power requirement, we calculate the sum of the maximum peak power of all the cores under test at a given time instant. It is assumed that for entire test time of the core, the maximum peak power is same. So, it is needed to calculate only the sum of the peak powers when the testing of a core starts. Only one benchmark (h953) among the ITC'02 benchmark set has power dissipation numbers included. For other SOCs we have used the power consumption values for the tests in design p22810 and p93791 reported in Pouget et al [87] and given in Table 5.2.

5.3 Genetic Algorithm Based Approach

We now describe a genetic algorithm (GA) [88] based approach to solve the generalized version of rectangle packing that does not consider the power constraint of the SOC. The overview of the generalized rectangle problem using GA approach is as follows. Let us assume that W is the TAM width and N is the number of cores available. First of all, we calculate testing times for all cores for all possible widths, that is, *creating possible rectangles for all cores.* It is obvious that in the final solution, we have to select one rectangle form each set of rectangles available for all cores. The GA assigns precedence to cores to establish an order in which the cores are to be scheduled. We pack all the rectangles into the rectangular bin of height W using that order. A chromosome represents a solution identifying the rectangle to be used for a particular core and also the preference assigned to it.

Core	d695	p22810	p93791
1	660	173	7014
2	602	173	74
3	823	1238	69
4	275	80	225
5	690	64	248
6	354	112	6150
7	530	2489	41
8	753	144	41
9	641	148	77
10	1144	52	395
11	-	2505	862
12	-	289	4634
13	-	739	9741
14	-	848	9741
15	-	487	78
16	-	115	201
17	-	580	6674
18	-	237	113
19	-	442	5252
20	-	441	7670
21	-	167	113
22	-	318	76
23	-	1309	7844
24	-	260	21
25	-	363	45
26	-	311	76
27	-	2512	3135
28	-	2921	159
29	-	413	6756
30	-	598	77
31	-	-	218
32	-	-	396

Table 5.2: Power consumption values [87]

5.3.1 Solution representation

In the GA, we need to encode each solution as a chromosome. A chromosome in our approach consists of two parts namely *ordering part* and *rectangle-selection part.*

Ordering part of a chromosome gives the preference of the cores to be selected, *i.e.*, the order in which we select the cores one by one to pack into the rectangular bin. A core in the order will be packed into the rectangular bin if the available height is sufficient for the chosen core. Otherwise, the next core in that order will be chosen to be packed into the bin.

Rectangle-selection part of a chromosome tells us which rectangle to be selected from a set of rectangles for a core. This part randomly selects particular rectangles to be used for a solution for all cores. The number of rectangles available for a particular core is equal to the total TAM width. That is, if the TAM width or the rectangular bin height is equal to 32 then the number of rectangles available for a core is 32, each one corresponding to a varying width ranging from 1 to 32. It may be noted that we need not keep all these rectangles in the database – remembering only the pareto-optimal points suffices.

These two parts of the chromosome form a solution to the given problem.

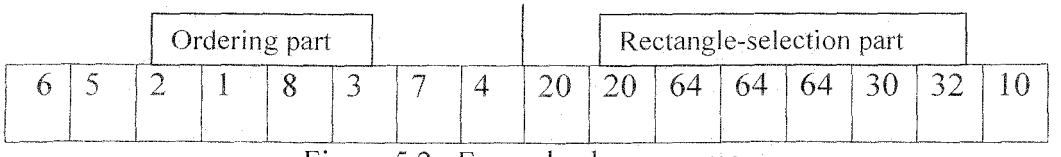


Figure 5.2: Example chromosome

Figure 5.2 shows an example chromosome. Here the rectangular bin width is 64 and the number of cores is 8. The ordering assigns sixth preference for core 1, fifth preference for core 2, second preference for core 3, first preference for core 4, and so on. The rectangle-selection part identifies the TAM widths for cores as follows: 20 for core1 and core2, 64 for cores 3, 4, and 5; 30 for

core 6, 32 for core 7, and 10 for core 8. This width assignment identifies the corresponding rectangle to be used for the core. The number of rectangles available for a core is assumed to be 64. Each rectangle gives different test application time depending on its height and test parameters. So, while packing the rectangles, we first consider core 4. Corresponding rectangle part gives the rectangle to be considered. In this case it is 64. Thus, the rectangle is selected and core 4 is scheduled from the beginning of test time for time units equal to the corresponding test time. At some point of time, let the bin be filled as shown in Fig 5.3. If the next core to be considered (depicted by chromosome) is core i, and the corresponding rectangle is r_i, we first check if r_i fits at start position t_1. If not, we try with the next core in the preference order. If for none of the cores the required rectangle fits starting from t_1, then the time t_1 to t_2 is left idle and we try to pack starting at t_2 again for core i. The same process is repeated.

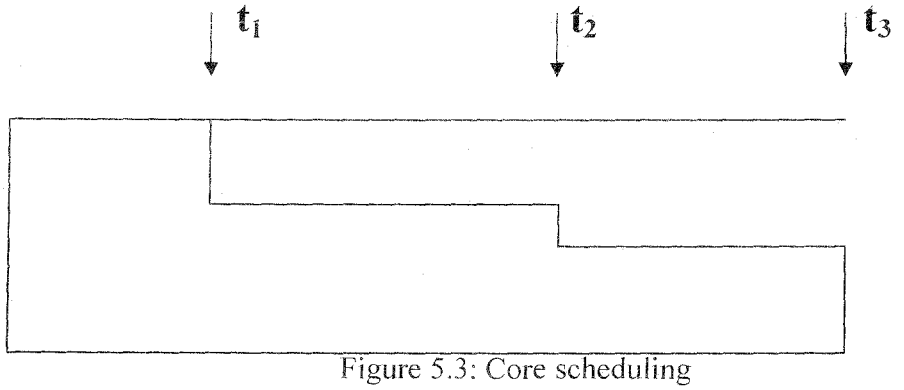


Figure 5.3: Core scheduling

5.3.2 Genetic operators

Two genetic operators crossover and mutation have been used to evolve new generations.

5.3.2.1 Crossover

Two chromosomes are selected randomly from the entire population. After selecting two chromosomes to participate in crossover, a single point

crossover is applied on the *ordering part* and *rectangle-selection part* of the chromosome. After generating new chromosomes, a check is made with the already generated chromosomes and duplicates are discarded.

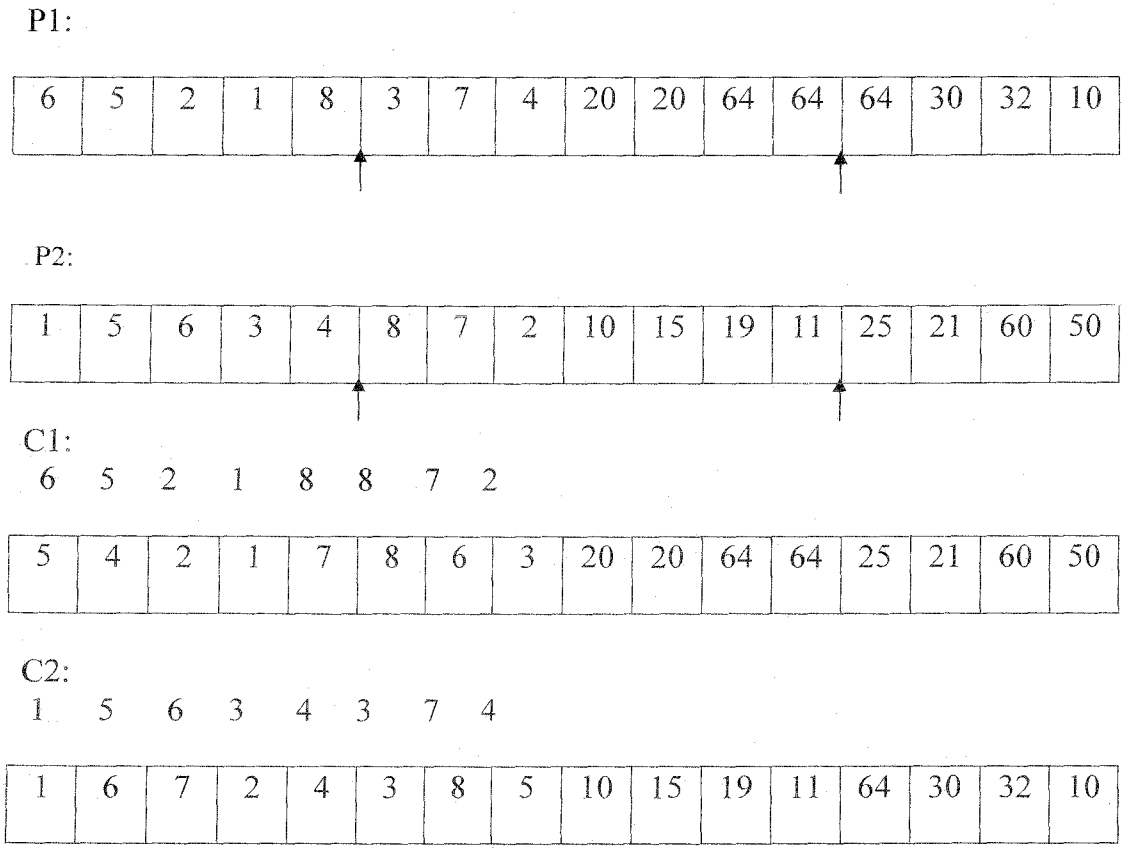


Figure 5.4: Example crossover operation

Figure 5.4 shows an example crossover operation. P1 and P2 are the parent chromosomes used to generate new children C1 and C2. C1 is generated by combining first part of P1 with second part of P2. Similarly, C2 is generated by joining the first part of P2 with the second part of P1.

While generating children in the above manner, we encounter the problem of two cores getting the same preference order. We resolve the problem in the following way. Let the order part of a chromosome with two cores having the same order be called *duplicate order part*. Our aim is to convert this *duplicate*

order part into *unique order part*. First of all, consider all cores with *preference* 1 from *duplicate order part*. The first such core is assigned *precedence* 1, while the second such core is allocated *preference* 2, and so on. Next, we consider all cores with *preference 2* from *duplicate order part* and assign next precedence to them. We continue this process until we get unique order part. For example, consider the *duplicate order part* of C2, mentioned in Figure 5.4. In the following, we show the steps to get the *unique order part* from this *duplicate order part*.

Duplicate order:	1	5	6	3	4	3	7	4
Step 1:	1	-	-	-	-	-	-	-
Step 2:	1	-	-	2	-	3	-	-
Step 3:	1	-	-	2	4	3	-	5
Step 4:	1	6	-	2	4	4	-	5
Step 5:	1	6	7	2	4	3	-	5
Step 6:	1	6	7	2	4	3	8	5

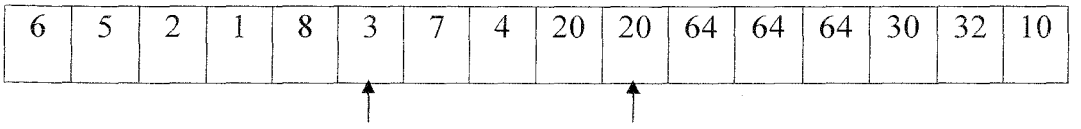
(Unique order part)

5.3.2.2 Mutation

Mutation is a very important operator as far as bringing variety into the population is concerned. As the population size is finite, the crossover operator alone cannot bring enough variation to the population. The mutation operator brings more effective variations into the chromosomes introducing newer search options. Figure 5.5 shows an example mutation operation. Here we take two separate mutation points for the ordering part and the rectangle-selection part. Based on the ordering mutation point, we change the preference of that particular core randomly. Similarly, based on the rectangle-selection mutation point, we choose a random width for the core selected. This gives another

solution for the problem. The duplicates in the ordering part are resolved similar to the crossover case.

P:



C:

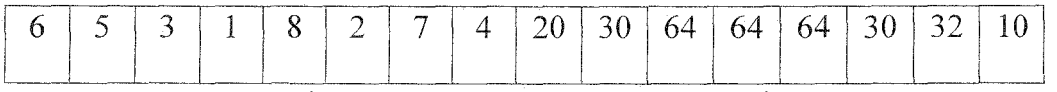


Figure 5.5: Example mutation operation

5.3.3 Fitness measure

Fitness of a chromosome is measured in terms of the cost of a solution, which is the total time required to test all cores in the system. The system testing time is the length of the rectangular bin after all cores are packed into that bin. The SOC test time is the fitness criteria in this method. The chromosome with minimum fitness value is the solution for the SOC.

5.3.4 Overall Genetic Algorithm

The evolution process is like this. We first generate the initial population chromosomes. The population size has been taken as 5000. Chromosomes are sorted in ascending order of their fitness values. First 20% chromosomes are copied directly to the next generation. We select 40% chromosomes by crossover and 40% chromosomes by mutation, as the population for the next generation. This process is repeated till no improvement in cost even after certain predefined number of generations (taken as 50 in our experimentation).

Algorithm GA

1. Create a random initial population of size POP_SIZE
2. For each chromosome *i* do

   2.1. Call *Place_rectangles(i)* to compute test time, T_i for chromosome *i*

   2.2. $Fitness_i = T_i$
3. Sort population
4. *No_of_gen_without_improv* = 0
5. While (*No_of_gen_without_improv* < 50) do

   5.1 Copy 20% best chromosomes to next generation

   5.2 Generate 30% new chromosomes via mutation

   5.3 Generate 50% new chromosomes via crossover

   5.4 Evaluate fitness of each chromosome using as in Step 2

   5.5 Sort population

   5.6 If (no improvement in fitness of best chromosome) then

   No_of_gen_without_improv ++

   Else

   No_of_gen_without_improv = 0

Procedure Place_rectangles(c)

/* **Possible_places:** an array of tuples <time_instant, wires_available> */

/* **No_places:** number of entries in array Possible_places */

1. No_places = 1
2. Possible_places[1] = <0, W> /* W is the total TAM wires available */
3. While all cores not placed do

3.1. $i = 1$

3.2. Let k be the next core chosen as per ordering part of c

3.3. Let the rectangle of k be R_k

3.4. Place R_k at time instant Possible_places[i].time_instant, if there is enough space to hold rectangle R_k

3.5. If R_k cannot be placed, choose a new core k from c; Goto Step 3.3

4. **Return** bin size as the total test time

End procedure

Algorithm 5.1: Overall Genetic Algorithm

5.3.5 Experimental results on unconstrained testing

In this section we present the experimental results. Table 5.3 presents the detailed comparison of our approach with other works reported in the literature. The approaches have been compared for 4 ITC'02 benchmark SOCs, namely d695 (10 cores), p93791 (32 cores), p34392 (21 cores) and p22810 (30 cores). For [86], we take the best of the two approaches reported there. To analyze the results, we have determined the best results in each case. The best results for each TAM width for a particular SOC has been shown in boldface in the table. Out of 28 such possible cases (7 for each SOC), [86] gives best results for 10 cases, [81] gives best results for 9 cases, our approach yields best results for 8 cases, whereas [87] gives the best result for a single case. Thus, our approach gives results comparable to the best ones known in the literature. It improves the basic scheme presented in [77] significantly. All the experiments have been carried out on a Pentium 4 machine with 1 MB main memory. The CPU time taken by the algorithm is a few seconds for the example SOCs. It may be noted that [181] presented another Genetic Algorithm based approach for SOC test scheduling. However, the results presented there are not on ITC'02

benchmarks. They have assumed some cores to have BIST within them. Thus, test times are not directly comparable with our work. Moreover, the total TAM width is also not mentioned for the examples that they have worked with.

5.4 Constraint driven scheduling

In this section, we extend the problem of integrated TAM scheduling to constraint driven test scheduling. The main constraints to be considered are

- Power constraints
- Precedence constraints

We solve the TAM scheduling problem such that the power and precedence constraints can be satisfied.

5.4.1 Power constraint

An efficient test schedule can reduce the testing time by allowing tests to be executed concurrently. However, executing tests concurrently increases the activity in the system, which leads to higher power consumption. It is important that the test power constraint must be considered carefully otherwise the system under test may be damaged due to overheating. SOCs in test mode can dissipate more power than in normal mode. This is because cores which do not normally operate in parallel may be tested concurrently to minimize testing time. Therefore, *power constrained test scheduling* is essential in order to limit the amount of concurrency during test application to ensure that the maximum power rating of the SOC is not exceeded. To take care of the power constraints, Step 3.4 of procedure **Place_rectangles** is modified to check the violation of power limit as well. The modified procedure looks as follows.

ITC'02 Benchmark	Strategy	Number of TAM wires						
		16	24	32	40	48	56	64
d965	Our	41891	28280	21307	17002	14420	12191	10744
	[86]	41553	**27982**	21014	**16908**	14236	**11988**	**10571**
	[87]	41847	29106	**20512**	18691	17257	-	13348
	[81]	**41604**	28064	21161	16993	**14183**	12085	10723
	[77]	43723	30317	23021	18459	15698	13415	11604
	[82]	41949	28327	21423	17210	16403	13023	12327
	[83]	44307	28576	21518	17617	14608	12462	11033
	[84]	46152	30777	22669	19551	16388	13877	11893
	[85]	42716	28639	21389	17366	15142	13208	11279
p93791	Our	**1743379**	1177300	886893	719191	**594168**	528255	447070
	[86]	1754980	1171190	88603	**706820**	600986	**501057**	**445748**
	[87]	1827819	1220469	945425	787588	639217	-	457862
	[81]	1757452	**1169945**	**878493**	718005	594575	509041	447971
	[77]	1851135	1248795	975016	794020	627934	568436	511286
	[82]	1775099	1192980	899807	705164	602613	521806	463707
	[83]	1791638	1185434	912233	718005	601450	528925	455738
	[84]	2404321	1598829	1179795	1060369	717602	625506	491496
	[85]	1791860	1200157	900798	719880	607955	521168	459233
p34392	Our	963000	657067	**544579**	**544579**	**544579**	**544579**	**544579**
	[86]	**939855**	637263	544579	544579	544579	544579	544579
	[87]	-	-	-	-	-	-	-
	[81]	944768	**628602**	544579	544579	544579	544579	544579
	[77]	1023820	759427	544579	544579	544579	544579	544579
	[82]	998733	720858	591027	544579	544579	544579	544579
	[83]	1010821	680411	551778	544579	544579	544579	544579
	[84]	1191829	828643	568133	544579	544579	544579	544579
	[85]	1016640	681745	544579	544579	544579	544579	544579
p22810	Our	**438306**	289780	226899	179897	151309	131180	117250
	[86]	438619	**289237**	226545	**167792**	153260	130949	116625
	[87]	473418	352834	236186	195733	159994	-	128332
	[81]	438619	289287	**218855**	175946	**147944**	**126947**	**109591**
	[77]	452639	307780	246150	197293	167256	145417	136941
	[82]	462240	361576	312662	278360	268474	266800	260639
	[83]	458068	299718	222471	190995	160221	145417	133405
	[84]	541402	356039	294788	220674	203257	175946	157527
	[85]	446684	300723	223462	184951	167858	145087	128512

Table 5.3: Test scheduling results.

Procedure Place_rectangles_power(c)

/* **Possible_places:** an array of tuples <time_instant, wires_available> */

/* **No_places:** number of entries in array Possible_places */

1. No_places = 1
   2. Possible_places[1] = <0, W> /* W is the total TAM wires available */
   3. While all cores not placed do
      3.1. $i = 1$
      3.2. Let k be the next core chosen as per ordering part of c
      3.3. Let the rectangle of k be R_k
      3.4. Place R_k at time instant Possible_places[i].time_instant, if there is enough space to hold rectangle R_k and power constraint is never violated throughout the schedule
      3.5. If R_k cannot be placed, choose a new core k from c; Goto Step 3.3
   4. **Return** bin size as the total test time

End procedure

5.4.1.1 Experimental results under power limitations

We applied our algorithm assuming different power constraint values. In the ITC'02 benchmark specification, no power data are given for benchmark files. We assumed the same power dissipation values for cores considered in the previous power constraint method [87]. Table 5.4 presents the results for SOC d695 for different power limit (P) values of 2500, 2000, 1800 and 1500. The total SOC testing time has been noted for different TAM widths. The results have been compared with those obtained in [87]. It can be seen that for SOC d695, the GA based approach needs, on an average, 9.34% lesser testing time than [87] under various power constraints. Table 5.5 presents the results for

SOC p22810 for different power limits and TAM widths. Here also the GA based approach needs 26.97% lesser testing time than [87]. Table 5.6 presents the results for SOC p93791 for various power limits and TAM widths. GA based approach needs 6.96% lesser testing time than [87] satisfying the power constraints.

Width	P=2500		P=2000		P=1800		P=1500	
	[87]	GA	[87]	GA	[87]	GA	[87]	GA
16	41847	42046	42450	41867	42450	42211	43541	42413
24	29106	28187	29106	28383	32054	28383	32663	29406
32	21931	21359	21942	21359	23864	21716	26973	22980
40	18691	16996	18691	17161	18774	17547	24369	18684
48	17257	14424	17467	15624	18774	16148	23425	17040
56	13963	12770	14563	12897	18774	13541	19402	15432
64	13394	12203	14469	12204	16804	12824	19402	15080

Table 5.4: Power constrained results for d695

Width	P=6000		P=5000		P=4000		P=3000	
	[87]	GA	[87]	GA	[87]	GA	[87]	GA
16	475961	441333	472026	438950	480223	441425	482963	443801
24	346461	292996	382507	293032	389243	296653	392525	303407
32	250487	226899	321930	229410	324478	226899	309255	226897
40	209559	189412	264038	189412	285307	189412	356215	192039
48	174928	156422	266166	156536	285814	156536	311632	155576
56	159686	145023	257600	135416	268272	145025	293528	147679
64	157568	124147	246110	116433	268856	121894	293012	121916

Table 5.5: Power constrained results for p22810

Width	P=2500		P=2000		P=1800		P=1500	
	[87]	GA	[87]	GA	[87]	GA	[87]	GA
16	1827819	1747112	1827819	1750012	1827819	1758396	1827819	1753643
24	1220469	1174193	1220469	1186000	1220469	11775316	1220469	1176422
32	965383	912246	957921	899338	1014616	904539	1117385	984578
40	821475	725531	821575	728513	848050	720951	1091210	870812
48	639217	603384	658132	606296	631214	599836	691866	602687
56	549669	532684	549669	525255	598487	528022	629051	571745
64	493599	445895	472653	458138	486469	469803	568734	508603

Table 5.6: Power constrained results for p93791

5.4.2 Precedence constraints

Precedence constraints impose a partial order among tests in a test suite. This can be motivated by several factors. For example, since BIST is likely to detect

more defects than an external test targeted only at random resistant faults, it may be desirable to apply BIST first to a core during manufacturing test. Similarly it may be desirable to test and diagnose memories earlier so that they can be used later for system test. Since larger cores are more likely to have defects due to their large silicon area, it may also be more desirable to test them first.

Let precedence constraints between cores be defined as, $i < j$ (test i must complete before test j is begun). We use the same problem formulation used in the generalized rectangle packing method but with the limitation that the precedence defined for that system must be satisfied. To take care of the precedence constraints, Steps 3.2 and 3.5 of procedure **Place_rectangles** are modified to check the violation of precedence. The procedure looks as follows.

Procedure Place_rectangles_precedence(c)

/* **Possible_places:** an array of tuples <time_instant, wires_available> */

/* **No_places:** number of entries in array Possible_places */

1. No_places = 1
2. Possible_places[1] = <0, W> /* W is the total TAM wires available */
3. While all cores not placed do
   - 3.1. $i = 1$
   - 3.2. Let k be the next core chosen as per ordering part of c satisfying precedence constrsints
   - 3.3. Let the rectangle of k be R_k
   - 3.4. Place R_k at time instant Possible_places[i].time_instant, if there is enough space to hold rectangle R_k
   - 3.5. If R_k cannot be placed, choose a new core k from c satisfying the precedence constraints; Goto Step 3.3

4. **Return** bin size as the total test time

End procedure

In the ITC'02 benchmark specification, no such precedence constraints are given for any system. We added some precedence constraints to SOC p22810 for simulation.

The added precedence constraints for SOC p22810 are:

- Core 12 can be tested after core 24
- Core 6 can be tested after core 12
- Cores 14 and 16 can be tested only after core 17
- Core 16 has to be tested before core 7
- Core 8 has to be tested after core 14
- Core 11 has to be tested after cores 7 and 8
- Cores 5 and 27 have to be tested after core 28
- Core 5 has to be tested before cores 1 and 2
- Core 27 has to be tested before cores 25 and 26
- Cores 1 and 25 have to be tested before core 20
- Cores 2 and 26 have to be tested before core 10
- Cores 10 and 20 have to be tested before core 15

Table 5.7 presents the testing time under the precedence constraints noted above for SOC p22810.

Width	Test Time
16	450562
24	304531
32	236521
40	193416
48	168569
56	161703
64	161588

Table 5.7: Results for SOC p22810 with given precedence constraints

5.5 Conclusion

In this chapter, we have presented a genetic algorithm based approach to solve the problem of System-on-Chip test scheduling using rectangular bin packing approach. For the unconstrained version of the problem, the scheme produces results comparable to the existing methods. The scheme has been extended to take care of the power constraints. The power constrained version produces results much improved than that reported in the literature. The basic scheme has also been modified for accommodating precedence constraints.

Chapter 6

Test Data Compression

System-on-Chip (SOC) integrated circuits composed of Intellectual Property (IP) cores in VLSI system design poses serious test challenges. The voluminous test data of these systems is one of the major concerns to the system tester. To cope with this, one can enhance the capability of an automatic test pattern generation (ATPG) tool to generate lesser number of test patterns. Another alternative is to use test data compression techniques in which the compressed patterns are stored in the memory of Automated Test Equipment (ATE) and are decoded by on-chip decoder to get back the original patterns for application to the core. Pre-computed test data set T_d provided by the core vendor is compressed into a much smaller test set T_e. But due to the limitation of test data bandwidth between the tester and the chip, testing cannot proceed faster than the amount of time required to transfer data. Hence *Test Time* $\geq$ *(amount of test data on the tester) / (number of tester channels)* $\times$ *(tester clock rate)*. Thus it is always a bottleneck.

Test data compression environment (TDCE) implies sending the compressed test data from the Automated Test Equipment (ATE) to on-chip decoder, decompressing the test data on chip and sending the decompressed test data to the circuit-under-test (CUT). There are two major components in TDCE: the **compression method**, used to compress the test set off-chip, and the associated **decompression method**, based on an on-chip decoder. On-chip decoder is used to restore the initial test set to be applied to the circuit.

This chapter proposes a scheme for test data compression based on Huffman coding. It is a modification of the Variable-length Input Huffman Coding (VIHC) technique proposed in [110]. It essentially splits the input file into two

data sets to achieve higher compression. Section 6.1 presents a brief overview of VIHC. Section 6.2 presents the motivation behind the suggested modification, while Section 6.3 enumerates the proposed scheme along with experimental results.

6.1 Brief overview of VIHC [110]

VIHC scheme uses variable length patterns instead of fixed length patterns, as input to the Huffman coding that exploits the test set features. The scheme tries to make long run of 0s, encoded by Huffman coding. Migration from fixed length patterns to variable length patterns does influence not only the compression ratio, but also provides an opportunity for parallel decoding of the encoded words, thereby achieving a reduction in test application time (TAT). Fig. 6.1 shows the example of VIHC scheme for a group size of 4 (m_h=4). Here the group size represents the maximum allowable runs of 0s (i.e. patterns 1, 01, 001, and 0001) that are allowed. Frequencies of these patterns are obtained by counting the occurrences of these patterns in the input test data. These patterns along with corresponding frequencies are given as input to the Huffman coding algorithm. Fig 6.1(a) shows the different patterns and their number of occurrences in the test vector t_{init}. Fig 6.1(c) shows Huffman tree corresponding to the dictionary of Fig 6.1(b).

Compression algorithm used in VIHC compression scheme can be divided into three phases. These are,

- **Preparation of initial test set:** As VIHC uses runs of 0s as input patterns, preprocessing of the test data set T_d, increases the number of such patterns. If the decoder is designed to have the capability of XORing the previous test pattern with the current pattern transmitted, the transmitter can send the XOR of two successive patterns. If these patterns are similar to each other, a large number of 0s will result. Such a difference pattern set is called T_{diff}. On the other hand, if such a feature

is not available, the original set T_d needs to be transmitted. The *don't care* bits need to be handled differently in the two cases. In the first case, for the compression of T_{diff}, the don't cares are mapped to the value of the previous vector on the same position; while for the compression of T_d, the don't cares are mapped to 0s. Next, the test set is reordered and denoted as *Reordered test set*. In the reordering process, starting with the test vector with minimum number of 1s, the next test vector t_{min} is determined, such that the following conditions are met. For T_{diff}, next test vector is selected such that the number of 1s in the difference between the test vector and the previous vector in t_{min} is the minimum. For T_d, the next test vector is selected such that the length of the minimum run of 0s is the maximum.

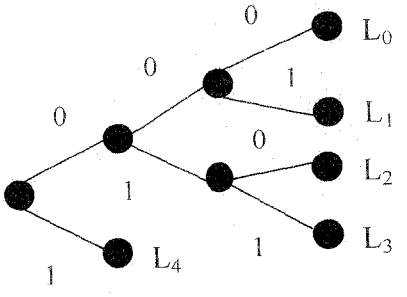
t_{init} : 1 01 0000 0000 0000 0001 0000 001

t_{comp} : 000 001 1 1 1 001 1 010

(a)

Pattern	Freq.	Code
$L_0 = 1$	1	000
$L_1 = 01$	1	001
$L_2 = 001$	1	010
$L_3 = 0001$	1	011
$L_4 = 0000$	4	1

(b)



(c)

Fig 6.1: VIHC example [110] (a) Test vectors (b) Dictionary (c) Huffman tree

- **Compute Huffman codes for the words:** Based on the selected group size m_h, the dictionary of variable-length patterns, and their frequencies are computed from reordered test set (T_d^R). Using these patterns, with corresponding frequencies, Huffman codes are computed.

- **Generate decoder information:** On-chip decoder for VIHC has to decode the Huffman codes, and it has to produce the original patterns (runs of 0s). This can efficiently be done by using a counter in the decoder circuit. Huffman decoder FSM in decoder unit generates the length of the pattern.

6.2 Motivation behind our work

Our work is based upon the observation that the frequency of a particular word may change over different regions of the test-data file. That is, while a word may be occurring very frequently in the first part of the file, may be very less frequent in the later part. Some other pattern may be more frequent in the second part. For example, Table 6.1 shows the distribution of different words for maximum run-length of 8 in the test data set for circuit s5378 over different regions of the file. The file has been divided into two partitions – first 50% and second 50%. In both the partitions, the pattern '1' has the highest number of occurrences. While, the occurrence of '01' has reduced from 492 in the first partition to 155 in the second partition, whereas, the occurrence of pattern '00000000' has gone up from 435 to 609. Similar changes in occurrence can be noted for the patterns '0000001' and '00000001'. Thus, coding a word based on its total number of occurrences in the file may lead to lesser compression than if the coding changes from first part of the file to the second part. The associated challenges are the following.

- Determining an optimal size for the partitions, that is whether the partitions be balanced etc.
- Reordering patterns within a partition.
- Designing the decoder for this modified VIHC encoding.
- Comparing the performance of this scheme with VIHC and other existing schemes in terms of compression ratio, test application time, decoder hardware overhead etc.

$m_h = 8$		
Patterns	Upto 50%	From 50-100%
1	4625	5777
01	492	155
001	287	79
0001	144	67
00001	79	32
000001	63	31
0000001	34	10
00000001	30	13
00000000	435	609

Table 6.1: Distribution of different patterns over different regions of the test file for circuit s5378

6.3 Cumulative VIHC (cVIHC) Scheme

Before going to the actual algorithm of cumulative VIHC let us take an example to illustrate the motivation. Consider a set of 4, 16-bit vectors t_1, t_2, t_3, t_4 as shown in Fig.6.2. If the group size m_h is taken as 4 bits, the test vectors can be considered to be consisting of the following patterns: 0, 01, 001, 0001 and 0000. Applying the VIHC coding scheme on this set, the compression length is obtained as 4×2+3×3+2×3+5×2+7×2=44 as shown in Fig. 6.3. Let us divide the test file into two halves (for the time being consider it to be divided into two equal parts). One half contains the vectors t_1, t_2 and the other half t_3 and t_4.

$\|t_i\|$ = 16 bits			m_h = 4			
t_1:	1	0000	1	001	001	0000
t_2:	01	0001	0000	0000	1	1
t_3:	01	0001	0001	01	0000	
t_4:	0001	0000	0001	0000		

Fig 6.2: VIHC coding scheme for four vectors [110]

Pattern	Freq.	Code
L_0=1	4	00
L_1=01	3	011
L_2=001	2	010
L_3=0001	5	10
L_4=0000	7	11

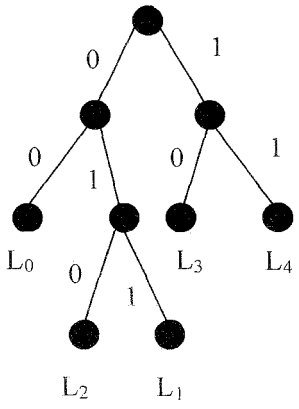
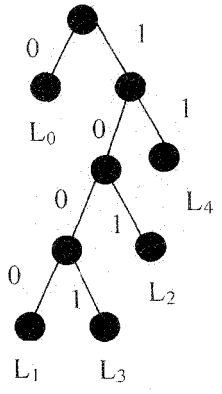


Fig 6.3: VIHC coding for file in Fig 6.2

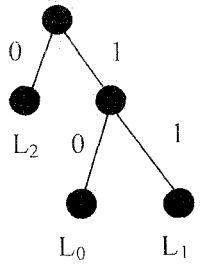
Now, if we apply the VIHC coding (for m_h = 4) separately on both of these files it is seen that total compression length becomes 26+14=40. This is shown in Fig. 6.4.

Pattern	Freq.	Code
L_0=1	4	0
L_1=01	1	1000
L_2=001	2	101
L_3=0001	1	1001
L_4=0000	4	11



(a) For file 1

Pattern	Freq.	Code
L_0=1	2	00
L_1=01	4	011
L_2=001	3	010



(b) For file 2

Fig 6.4: VIHC coding for two splitted files

Thus, dividing the original test file into two and reordering the vectors in a certain way, so that, we put similar vectors in one file, can really improve the compression ratio. Next, we need to address the issue related to file splitting – the size of individual partitions.

6.3.1 Compression Algorithm

This section presents the procedures used to implement the proposed method (cVIHC). The initial test set (T_D) is partially specified and the test vectors can be reordered. This is because ATPG (Automated Test Pattern Generator) tools generate a small number of care bits in every test vector [110]. This gives great flexibility to the mapping and reordering algorithm, which in a pre-processing step prepares the test set for compression by mapping the "don't cares" in the test set to '0' or '1' and reordering the test set such that cVIHC can increase its compression. Thus, the compression algorithm has four main procedures: (1) prepare initial test set, (2) calculate the break-point, (3) compute Huffman code and (4) generate decoder information.

(1) Prepare Initial Test Set: The procedure consists of two steps. In the first step, the 'don't cares' are mapped to the value of the previous vector on the same position. In the second step the test set is reordered such that the number of 1s in the difference between two test vectors is minimized and the length of the minimum run of 0s is maximized. This is exploited by the variable-length input patterns used for Huffman code computation.

(2) Calculate Break-point: Break-point calculation involves an exhaustive statistical analysis over the input test set. To find the break-point, we build dictionaries with first 1% of the input, 1-2% of the input ... 99-100% of the input. By doing statistical analysis over these dictionaries built, exact position of the break-point is calculated. The break-point depicts the probability of

maximum available runs of 0s. The probability of maximum available runs of 0s is quite high at one side of the break-point, and it is comparatively low at the other side. This allows the exploitation of the test set features for compression at its maximum.

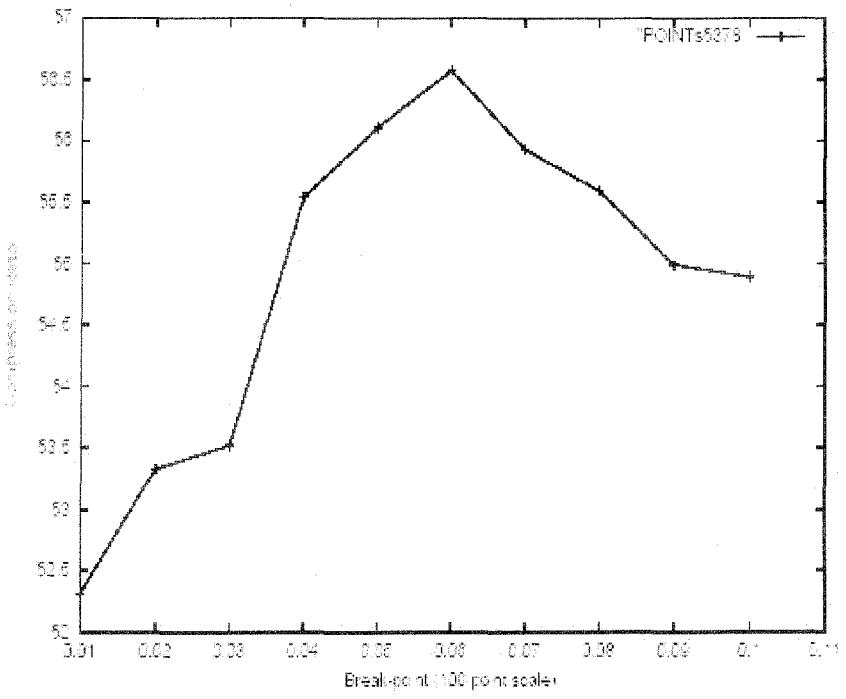


Fig. 6.5: Compression Ratio Vs Break-point

Acquainted with the distribution of the maximum available runs of 0s at each part of the test set, supplementary test patterns can be encoded efficiently, augmenting the compression ratio. Table 6.2 shows the break-points of various benchmark circuits, with 100 point scale. As shown in the Table 6.1, except for the circuit s35932, all circuits have their breakpoints at the early parts of the test set. Fig. 6.5 shows the variation of the compression ratio with respect to the break-point for the circuit s5378. As shown in the graph the compression ratio is quite high at certain value of the break-point, depicting its importance.

Circuit	Break-point
s5378	6
s9234	2
s13207	2
s15850	3
s35932	81
s38417	5
s38584	5

Table 6.2: Break-point of various benchmarks

(3) Compute Huffman Code: Based on the chosen group size (m_h) and the break-point, the dictionary of variable-length input patterns L and the number of occurrences P are determined from the reordered test set. Using L and P the code is computed by the Huffman encoding algorithm [169].

(4) Generate Decoder Information: This procedure computes the necessary information to build the decoder. Decoder information is generated for each part of the input. For each Huffman code c_i, a binary code b_i is assigned. The binary code is composed from the length of the initial pattern of $log_2 (m_h + 1)$ bits and a special bit which identifies the cases when the initial pattern is composed of runs of only 0s or runs of 0s ending in 1, thus $b_i = (L_i, 0)$ for $i < m_h$, and $b_{mh} = (L_{mh}, 1)$. The *Control and Generation Unit* uses the binary code to generate the pattern. The process is similar to the one noted in [110].

The complete cVIHC algorithm has been noted in Algorithm 6.1.

Algorithm cVIHC

Input: Test set T_d

Output: cVIHC decoder

1. Identify the break-point for T_d
2. Divide the set T_d into T_d^1 and T_d^2
3. SetDefaultValues(T_d^1); SetDefaultValues(T_d^2)
4. Perform reordering of T_d^1 and T_d^2
5. Generate Huffman code for T_d^1 and T_d^2
6. Design the controller for T_d^1 and T_d^2 as in [110]

Algorithm 6.1: cVIHC algorithm

6.3.2 Decompression Architecture

Fig. 6.6 shows the parallel decoder structure for cVIHC. It is similar to the VIHC decoder [110], in the sense that it consists of two parallel units – a finite state controller (FSM), called *code word detector* (CWD) and a code generation unit (CGU).

The CWD unit consists of two finite state machines (FSMs), one for each part of the compressed test set to detect Huffman code and output the corresponding binary code. A switcher controls the FSMs in CWD. Switcher, consisting of a counter, counts the number of clock cycles for which the FSM corresponding to first part of the test set is active. It should be noted that, during the time the FSM for first part of the test set is active, other FSM remains in its current state. The CGU is responsible for controlling the data transfer between the ATE and the CWD; generate the initial pattern and the scan clock for the circuit-under-test (CUT).

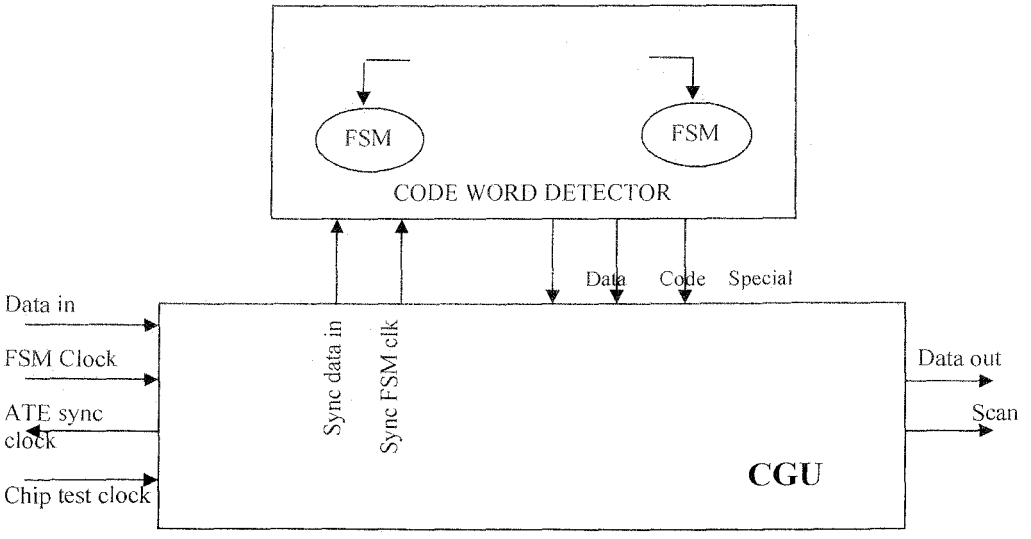


Figure: 6.6: cVIHC Decoder

The *data in* line is the input from the ATE synchronous with the external clock ATE clock. When the active FSM in CWD detects a codeword, the *code* line is high and the binary code is output on the *data* lines. The *special* input to the

CGU is used to differentiate between the two types of patterns, runs of 0s only and runs of 0s ending in 1. After loading the code, the CGU generates the pattern and the internal scan clock for the CUT. If the decoding unit generates a new code while the CGU is busy processing the current one, the *ATE sync* line is low notifying the ATE to stop sending data and the *Sync FSM clk* is disabled forcing the Huff-decoder to maintain its current state. Dividing the cVIHC decoder in CWD and CGU, allows the CWD to continue loading the next codeword while the CGU generates the current pattern. Thus, the CWD is interrupted only if necessary.

A detailed view of the *Control and Generator Unit* (CGU) is given in Fig. 6.7. It is similar to the CGU of VIHC scheme [110]. The CGU is formed from a $log_2 (m_h + 1)$-bit counter and additional logic. When the *code* line is high, the values at the *data* inputs are loaded into the counter. With the data loaded, the counter decrements to 1 setting the *data out* to 0. When the value of the counter is 1, *data out* is set to 0 if the *special* line is high, or to 1 if the *special* line is low.

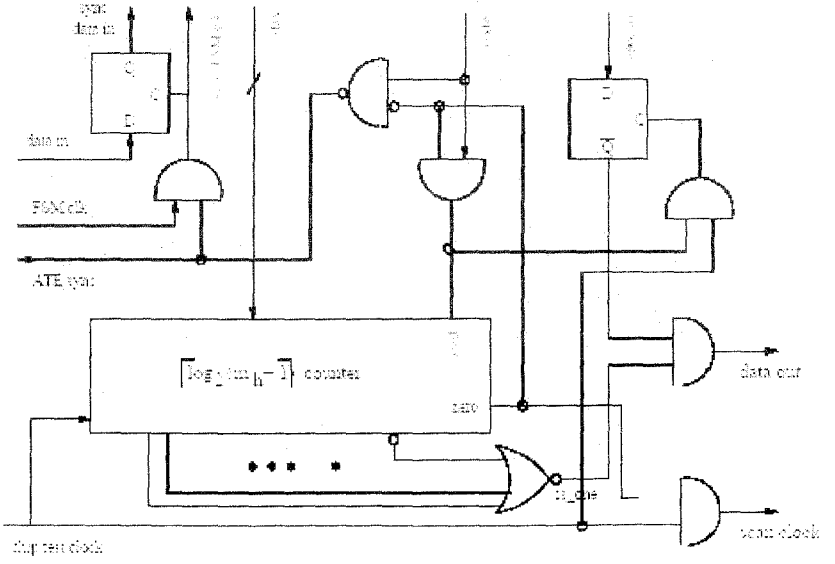


Fig. 6.7: Control and Generator Unit for cVIHC

6.3.3 Test Application Time (TAT) Analysis

TAT is the time taken to transfer and decode the compressed test data from ATE. Hence it depends on the compression ratio, type of the on-chip decoder and length of the pattern. A detailed discussion on calculating the amount of TAT needed has been presented in [110]. The function by which we can approximate the TAT is given by,

$$\tau(\alpha) \approx H(m_h) + \delta + n_0 \times (m_h - w_{min} \times \alpha) / \alpha$$

where,

- $\alpha = f_{chip} / f_{ate}$ is the ratio between the on-chip test frequency and the ATE operating frequency,
- n_0 is the number of patterns with length,
- δ is the number of extra ATE clocks needed to decode the last codeword,
- w_{min} is the minimum length of the coded word and
- $H(m_h)$ is the total compression length for group size m_h.

6.3.4 Experimental Results

To validate the efficiency of the proposed method, experiments were performed on the full-scan version of the largest ISCAS 89 benchmark circuits. Test sets used in this experiment were obtained using MinTest [171] dynamic compaction. We implemented the proposed method in C++ and the experiments were performed on a Pentium III 866MHz processor and 128 MB memory. To provide an uniform basis for the comparison between Golomb, SC, VIHC and cVIHC, as the four methods use a group size, we use the group size for which [106] reports the best results. Table 6.2 provides a compression comparison for the T_{diff} test sets. The compression percentage is computed as:
*Compression (%) =\{(Original size – Compressed size) / Original size\} * 100*

Circuit	GS	Break-Point	SC [103]	Golomb [106]	FDR [108]	VIHC [110]	cVIHC
s5378	4	6	52.33	40.70	48.19	51.52	**56.57**
s9234	4	2	52.63	43.34	44.88	54.84	**60.61**
s13207	16	2	77.73	74.78	78.67	83.21	**87.24**
s15850	4	3	63.49	47.11	52.87	60.68	**65.61**
s35932	16	81	65.72	N/A	10.19	66.47	**71.09**
s38417	4	5	67.26	44.12	54.53	54.51	**60.43**
s38584	4	5	58.74	47.71	52.85	56.97	**63.19**

Table 6.3: Compression comparisons for T_{diff}

Table 6.3 clearly demonstrates the superior compression performance of cVIHC. The cVIHC achieves the highest compression for all of the seven test cases we considered.

Table 6.4 illustrates the area overhead computed with the Synopsys Design Compiler for the four on-chip decoders. For all methods, the entire decoder including buffers, shift registers and counters was synthesized. Without loss of generality the decoders for s5378 were synthesized. As shown in Table 6.4, the area overhead for SC is almost double than the area overhead of cVIHC and area overhead of cVIHC is 1.5 times greater than the area overhead of VIHC.

Finally, Tables 6.5 and 6.6 provide a Test Application Time (TAT) comparison for the T_{diff} test sets. Tables clearly demonstrate the superior TAT performance of cVIHC. The cVIHC achieves the best TAT for all of the seven test cases we considered.

Compression Method	Area overhead in *tu**		
	Group Size		
	4	8	16
SC[103]	349	587	900
Golomb[106]	125	227	307
FDR[108]	320		
VIHC[167]	126	201	296
cVIHC	216	325	477

* technoloty units for the *lsi 10k* library (Synopsys Design Compiler)

Table 6.4: Area overhead compression

As shown in the Tables 6.5 and 6.6, when compared to VIHC, the proposed method obtains a maximum reduction of 13.42% (s9234), a minimum of 9.36% (s5378) and an average of 11.72% in Test Application Time.

6.4 Conclusion

In this chapter we have presented a scheme for varying the codes assigned to input patterns in different parts of the input stream resulting better compression and reduced test application time. The proposed compression algorithm is able to significantly increase the compression performance with minimal decoder circuit area overhead. Also, the proposed scheme achieves the lowest Test Application Time for all values of the ratio of on-chip frequency to the ATE frequency. The associated increase in area may be accommodated considering the shrinkage sizes of the VLSI circuitry with the advancement in technology. It may be noted that the proposed scheme has been augmented further in [182]. The modifications done are as follows.

1. A Genetic algorithm based formulation has been made to decide the partition to which a pattern should belong to maximize compression. This has enabled the authors to bring more related patterns into same

partition. This results in the improvement of average compression ratio from 66.40% (our work) to 71.75%. It also improves TAT.

2. The decoder design has been modified to have a single FSM for both the parts, with a special change-over pattern introduced at the end of first partition identifying the switch-over.

Circuit	Compression Method	TAT ATE Clock Cycles			
		$\alpha = 2$	$\alpha = 4$	$\alpha = 6$	$\alpha = 8$
s5378	SC[103]	15491	15491	15491	15491
	Golomb[106]	21432	16662	11892	11892
	FDR[108]	22263	14678	12968	12968
	VIHC[110]	15763	11516	11516	11516
	cVIHC	**14751**	**10324**	**10324**	**10324**
	% red. in cVIHC w.r.t VIHC	6.42	10.35	10.35	10.35
s9234	SC[103]	25324	25324	25324	25324
	Golomb[106]	33651	25693	17735	17735
	FDR[108]	36135	24128	21381	21381
	VIHC[110]	24743	17735	17735	17735
	cVIHC	**22561**	**15081**	**15081**	**15081**
	% red. in cVIHC w.r.t VIHC	8.81	14.96	14.96	14.96
s13207	SC[103]	89368	53490	45137	36784
	Golomb[106]	104440	66381	47783	47481
	FDR[108]	107059	63011	49858	41989
	VIHC[110]	89865	52769	44180	36065
	cVIHC	**85646**	**47365**	**38443**	**29915**
	% red. in cVIHC w.r.t VIHC	4.69	10.24	12.98	17.05
s15850	SC[103]	46067	46067	46067	46067
	Golomb[106]	63989	47164	30339	30339
	FDR[108]	62419	39628	33767	33767
	VIHC[110]	45765	30264	30264	30264
	cVIHC	**42596**	**26477**	**26477**	**26477**
	% red. in cVIHC w.r.t VIHC	6.92	12.51	12.51	12.51

Table 6.5 : TAT compressions for T_{diff} compressed test sets.

Circuit	Compression Method	TAT ATE Clock Cycles			
		$\alpha = 2$	$\alpha = 4$	$\alpha = 6$	$\alpha = 8$
s35932	SC[103]	16870	11749	10710	9671
	Golomb[106]	37869	32886	30509	30509
	FDR[108]	32509	20605	18438	17045
	VIHC[167]	17584	12076	10857	9645
	cVIHC	**15947**	**10661**	**9374**	**8295**
	% red. in cVIHC w.r.t VIHC	9.31	11.71	13.65	13.99
s38417	SC[103]	103604	103604	103604	103604
	Golomb[106]	143844	109801	75758	75758
	FDR[108]	144811	93450	81578	81578
	VIHC[167]	106411	74943	74943	74943
	cVIHC	**97262**	**65196**	**65196**	**65196**
	% red. in cVIHC w.r.t VIHC	8.59	13.05	13.05	13.05
s38584	SC[103]	124011	124011	124011	124011
	Golomb[106]	169356	127512	85668	85668
	FDR[108]	170143	110982	96677	96677
	VIHC[167]	123693	85668	85668	85668
	cVIHC	**113501**	**73301**	**73301**	**73301**
	% red. in cVIHC w.r.t VIHC	8.23	14.43	14.43	14.43

Table 6.6: TAT compressions for T_{diff} compressed test sets.

Chapter 7

Conclusion & Future Work

In this thesis, we have presented solutions to some of the very important issues in System-On-Chip testing. After performing a detailed survey of the works done in this field, we identified a few problems to work on.

The first problem tackled is that of test scheduling. Several approaches have been presented. We have reported a simulated annealing based technique that starting with a heuristic solution, improves upon it. The next chapter presented a test bus partitioning scheme. It also handles the issues of interconnect testing. Ordering of cores on test rails has been investigated to reduce the number of transitions in the scan chains. Next we have presented a more generalized scheme that performs core testing using a rectangle packing approach. Here also the peak power constraint of the SOC under test and the precedence constraints between the cores have been considered.

The other problem that we have addressed is of test data compression. We have modified an existing compression scheme based on Huffman coding to change the coding pattern in different regions of the input file, thus yielding better compression. It also reduces the test application time significantly.

However there are quite a few problems are coming up that needs to be handled to have complete solution to the problem of SOC testing. In the following we enumerate some of them.

(i) As the SOC is designed based on reuse philosophy, it is very much likely that a complete SOC designed today will be utilized as a core in a future design. Testing of such hierarchical core poses a difficult challenge as the test engineer will not have enough freedom to modify the embedded test structure.

(ii) The new designs are coming up with multiple frequency domains in which different regions of the same chip operates at different

frequencies. Testing an SOC in which individual cores are operating at their own frequencies in definitely a challenge posed to the test engineer to reduce test time.

(iii) New architectures for system chips are being proposed that will have an On-Chip network to be used for communication. Testing such chips using the underlying network to transport test patterns and responses needs to be addressed.

References

[1] International SEMATECH, "The international technology roadmap for semiconductors (ITRS)," 2001 Edition, http://public.itrs.net/Files/2001ITRS/Home.htm.

[2] Indradeep Ghosh, Sujit Dey, Niraj K. Jha "A Fast and Low Cost Testing Technique for Core-based System-on-Chip," Proceedings of the IEEE Design Automation Conference, pp. 542-547, June 1998.

[3] Zorian,Y., Marinissen, E.J., and Dey, S., "Testing embedded-core-based system chips," IEEE Computer, 1999,32, (6), pp. 52-60.

[4] Yervant Zorian. " System-Chip Test strategies," Proceedings of the IEEE Design Automation Conference, pp. 752-757, June 1998.

[5] "Virtual Socket Interface Architectural Document", VSI Alliance, Nov. 1996.

[6] Y. Zorian, "Test Requirements for Embedded Core-Based Systems and IEEE P1500," Proceedings of IEEE International Test Conference (ITC), pp. 191–199, IEEE Computer Society Press, Nov. 1997.

[7] Yervant Zorian. "A Distributed BIST Control Scheme for Complex VLSI Devices," Proceedings of IEEE VLSI Test Symposium, pages6–11, April 1993.

[8] Y. Zorian and E. J. Marinissen, "System Chip Test: How Will It Impact Your Design?," Proceedings of ACM/IEEE Design Automation Conference (DAC), pp. 136–142, Association for Computing Machinery, Inc., June 2000.

[9] M. Abramovici, M. A. Breuer, and A. D. Friedman, Digital Systems Testing and Testable Design. IEEE Press, 1990.

[10] M. L. Bushnell and V. D. Agrawal, Esentials of Electronic Testing for Digital,Memory and Mixed-Signal VLSI Circuits. Kluwer Academic Publishers, 2000.

[11] A. L. Crouch, Design-for-test for Digital IC's and Embedded Core Systems. Prentice-Hall, 1999.

[12] Iyenger, V., Chakrabarty, K., and Marinissen, E.J. : "Recent advances in test planning for modular testing of core-based SOCs," ATS, 2002, pp. 320-325

[13] V.D. Agrawal et al., "Built-In Self-Test for Digital Integrated Circuits," AT&T Technical J., Mar. 1994, p. 30.

[14] M. Sachdev, P. Janssen, and V. Zieren, "Defect Detection with Transient Current Testing and its Potential for Deep Sub-Micron CMOS ICs," Proc. IEEE Int'l Test Conf., IEEE CS Press, Los Alamitos, Calif., 1998, pp. 204-213.

[15] K. Wallquist, A. Righter, and C. Hawkins, "A General Purpose IDDQ Measurement Circuit," Proc. IEEE Int'l Test Conf., IEEE CS Press, Los Alamitos, Calif., 1993, pp. 642-651.

REFERENCES

[16] The 1997 National Technology Roadmap for Semiconductors, Semiconductor Industry Assoc., San Jose, Calif., 1997.

[17] P. Varma and S. Bhatia, "A Structured Test Re-Use Methodology for Core-Based System Chips," Proc. IEEE Int'l Test Conf., IEEE CS Press, Los Alamitos, Calif., 1998, pp. 294-302.

[18] E.J. Marinissen et al., "A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores," Proc. IEEE Int'l Test Conf., IEEE CS Press, Los Alamitos, Calif., 1998, pp. 284-293.

[19] Rajusman, R., "Testing a system-on-chip with embedded microprocessor," ITC 1999, pp. 499-508.

[20] Erika Cota , Y Luigi Carro , Alex Orailoglu , Marcelo Lubaszewski y "Test Planning and Design Space Exploration in a Core-based Environment," Proc. of the conference on Design, Automationand test in Europe, 2002 pp. 478.

[21] Venkata Immaneni and Srinivas Raman, "Direct Access Test Scheme - Design of Block and Core Cells for Embedded ASICs," Proceedings of IEEE International Test Conference, pages 488–492, September 1990.

[22] Prab Varma, "Evolving Strategies for Testing Systems on Silicon," Integrated System Design- Virtual Chip Design Supplement, September 1997.

[23] Prab Varma and Sandeep Bhatia, "A Structured Test Re-Use Methodology for Systems on Silicon," In Digest of Papers of IEEE International Workshop on Testing Embedded Core-Based Systems, pages 3.1–1–8, November 1997.

[24] Bruce Mathewson, "Core Provider's Test Experience," http://grouper.ieee.org/groups/1500/pastmeetings.html#dac98. Presentation at IEEE P1500 Working Group Meeting, Sunnyvale, CA, June 1998.

[25] IEEE Computer Society. IEEE Standard Test Access Port and Boundary-Scan Architecture - IEEE Std 1149.1-1990. IEEE, New York, 1990.

[26] Hansen, P., "Testing conventional logic and memory clusters using boundary scan device as virtual ATE channels," Proc. IEEE International Test Conference, pp. 166-173, 1989.

[27] Parker, K.P., "The impact of boundary scan on board test," IEEE Des. Test Comput, Vol.6, No.4, pp. 18-30, 1989.

[28] Lee Whetsel, "An IEEE 1149.1 Based Test Access Architecture for ICs with Embedded Cores," In Proceedings of IEEE International Test Conference, pages 69–78, November 1997.

[29] Debashis Bhattacharya., "Hierarchical Test Access Architecture for Embedded Cores in an Integrated Circuit," In Proceedings IEEE VLSI Test Symposium, pages 8–14, April 1998.

REFERENCES

[31] Bahram Pouya and Nur Touba, "Modifying User-Defined Logic for Test Access to Embedded Cores," In Proceedings IEEE International Test Conference, pages 60–68, November 1997.

[32] F. Beenker, B. Bennetts, L. Thijssen, "Testability Concepts for Digital ICs - Macro Test Approach" Volume 3 of Frontiers in Electronics Testing. Kluwer Academic Publishers, Boston, 1995.

[33] B.Bennetts, "A Design Strategy for System-On-Chip testing," Electronic Products, June 1997, pp. 57-59.

[34] Ghosh, I., Jha, N.K., and Dey, S., "Low overhead design for testability and test generation technique for core-based system-on-a-chip," IEEE Trans. Comput.-Aided Des., 1999, 18. (11), p. 1661.

[35] Freemen, S., "Test generation for data-path logic: the F-path method," IEEE J. Solid-state Circuits, 1988, 23 pp. 421-427.

[36] Ravi, S., Lakshminarayana, G., and Jha, N.K., "Testing of core-based system-on-a-chip," IEEE Trans. Comput.-Aided Des., 2001, 20, (3), pp. 426-439.

[37] Nourani, m., and Papachristou, C., "Structural fault testing of embedded cores using pipelining," J. Electron. Test., Theory Appl., 1999, 15 (1), p. 129.

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

[39] Semiconductor Industry Association. International Technology Roadmap for Semiconductors, http://public.itrs.net/files/1999 SIA Roadmap/Home.htm

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

[41] K. Chakrabarty, "Design of system-on-a-chip test access architectures using integer linear programming," Proc. VLSI Test Symposium, pp. 127-134, 2000.

[42] K. Chakrabarty, "Design of system-on-a-chip test access architectures under place-and-route and power constraints," Proc. Design Automation Conference, pp. 432-437, 2000.

[43] K. Chakrabarty, "Optimal test access architectures for system-on-a-chip," ACM Transactions on Design Automation of Electronic Systems, vol. 6, pp. 26–49, January 2001.

[44] T.J. Chakraborty, S. Bhawmik and C-.H. Chiang, "Test access methodology for system-on-chip testing," Digest of the International Workshop on Testing Embedded Core-Based System-Chips, pp. 1.1-1–1.1-7, 2000.

[45] E. Larsson and Z. Peng, "An integrated system-on-chip test framework," Proc. Design, Automation, and Test in Europe (DATE), pp. 138-144, 2001.

REFERENCES

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

[47] M. Nourani and C. Papachristou, "An ILP formulation to optimize test access mechanism in system-on-chip testing," IEEE International Test Conference, pp. 902–910, 2000.

[48] IEEE P1500 Standard for Embedded Core Test. http://grouper.ieee.org/groups/1500

[49] E.J. Marinissen, R. Kapur and Y. Zorian, "On using IEEE P1500 SECT for test plug-n-play," Proc. International Test Conference, pp. 770–777, 2000.

[50] V.Iyenger, K. Chakraborty, and E.J. Marinissen, "Test wrapper and test access mechanism co-optimization for system-on-chip," Journal of Electronics Testing: Theory and Applications, 18(2): 213-230, 2002.

[51] E.J. Marinissen, et al., "A structured and scalable mechanism for test access to embedded reusable cores," Proc. International Test Conference, pp. 284-293, 1998.

[52] M.R. Garey and D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Co., San Francisco, CA, 1979.

[53] V. Immaneni and S. Raman, "Direct access test scheme - Design of block and core cells for embedded ASICs," Proc. International Test Conference, pp. 488–492, 1990.

[54] I. Ghosh, S. Dey and N.K. Jha, "A fast and low cost testing technique for core-based system-on-chip," Proc. Design Automation Conference, pp. 542–547, 1998.

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

[56] P. Harrod, "Testing re-usable IP: A case study," Proc. International Test Conference, pp. 493–498, 1999.

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

[58] E. Larsson and Z. Peng, "An integrated system-on-chip test framework," Proc. Design, Automation, and Test in Europe (DATE), pp. 138-144, 2001.

[59] M. Nourani and C. Papachristou, "An ILP formulation to optimize test access mechanism in system-on-chip testing," IEEE International Test Conference, pp. 902–910, 2000.

[60] K. Chakrabarty, "Design of system-on-a-chip test access architectures using integer linear programming," Proc. VLSI Test Symposium, pp. 127-134, 2000.

REFERENCES

[61] K. Chakrabarty, "Optimal test access architectures for system-on-a-chip" ACM Transactions on Design Automation of Electronic Systems, vol. 6, pp. 26–49, January 2001.

[62] K. Chakrabarty, "Design of system-on-a-chip test access architectures under place-and-route and power constraints," Proc. Design Automation Conference, pp. 432-437, 2000.

[63] Marinissen, E.J., Kapur, R., Lousberg, M., Mclaurin, T., Ricchetti, M., and Zorian, Y., "On IEEE P1500's standard for embedded core test," J.Electron. Test., Theory Appl., 202, 18, (4/5), pp. 365-383

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

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

[66] Koranne, S, "A novel reconfigurable wrapper for testing of embedded core-based SOCs and its associated scheduling algorithm," J.Electron Test., Theory Appl., 2002, 18, (4/5), pp. 415-434.

[67] Koranne, S., "Design of reconfigurable access wrappers for embedded core-based SOC test," IEEE trans., VLSI SYST., 2003, 11, (5), PP. 955-960.

[68] Larsson, E., and Peng, Z., "A reconfigurable power-conscious core wrapper and its application to SOC test scheduling," ITC, 2003, pp. 1135-1144.

[69] Nicolici, N., and Al-Hashimi, B.M., "Multiple scan chains for power minimization during test application in sequential circuits," IEEE Trans., Comput, 2002, 51, (6), pp. 721-734.

[70] Vermaak, H., and Kerkhoff, H.G., "Enhanced P1500 compliant wrapper suitable for delay fault testing of embedded cores," ETW, 2003, pp. 257-262.

[71] Xu, Q., and Nicolici, N., "Wrapper design for testing IP cores with multiple clock domains," DATE, 2004, pp. 416-421.

[72] Hetherington ,G., Fryars, t., Tamarapalli, N., Kassab, M., Hassan, A., and Rajski, J., "Logic BIST for large industrial designs : real issues and case studies," ITC , 1999, pp. 358-367.

[73] Goel, S.k., "An improved wrapper architecture for paraller testing of hierarchical cores," ETS, 2004, pp. 147-152.

[74] Vikram Iyenger, K. Ckakrabarty, and E. J. Marinissen, "Test wrapper and test access mechanism cooptimization for system-on-chip," In proc. IEEE International Test Conference, Baltimore, MD, pages 1023–1032, october 2001.

[75] Vikram Iyenger, K. Ckakrabarty, and E. J. Marinissen "Efficient wrapper/tam co-optimization for large socs," Proc. Design Automation and Test in Europe (DATE) Conf., pages 491–498, 2002.

REFERENCES

[76] Edabi, Z.S., and Ivanov, "Design of an optimal test access architecture using a genetic algorithm," In Proceedings of IEEE Asian Test Syposium (ATS 01), pages 205–210, 2001.

[77] K. Chakrabarthy, V. Iyengar, and E J Marrinesen, "On using rectangle packing for soc wrapper/tam cooptimization," Proc. VLSI Test Symposium, pages 253– 258, 2002.

[78] R.M. Chou, K.K.Saluja, and V.D.Agarwal, "Scheduling tests for vlsi systems under power constraints," IEEE Trans. VLSI Systems, 5(2), 1997.

[79] Y.Huang et al., "Resource allocation and test scheduling for concurrent test of core-based soc design," Proc. Asian Test Symposium, pages 265–270, 2001.

[80] SoC Test Benchmarks ITC'02. http://extra.research.philips.com/itcsocbechm.

[81] W. Zou, S.R. Reddy, I. Pomeranz, and Y. Huang, "Soc test scheduling using simulated annealing," Proc. VLSI Test Symposium, 2003.

[82] V. Iyenger, K. Chakrabarty, and E.J. Marinissen, "Test wrapper and test access mechanism co-optimization for system on a chip design," Proc. International Test Conference, 2001.

[83] S.K. Goel and E.J. Marinissen, "Effective and efficient test architecture design for socs," Proc. International Test Conference, pages 529–538, 2002.

[84] S.K. Goel and E.J. Marinissen, "Effective and efficient test architecture design for socs," Proc. International Test Conference, pages 529–538, 2002.

[85] Y. Huang et al., "Optimal core wrapper width selection and soc test scheduling based on 3-d bin packing algorithm," Proc. International Test Conference, 2002.

[86] Y. Xia et al., "Using a distributed rectangle binpacking approach for core-based soc test scheduling with power constraints," Proc. ICCAD, pages 100–105, 2003.

[87] J.Pouget, E.Larsson, and Z.Peng, "Soc test time minimization under multiple constraints," In Proceedings of IEEE Asian Test Syposium(ATS 03), 8(2):312–317,November 2003.

[88] G.E. Goldberg, "Genetic algorithm in search, optimizations and machine learning," AddisonWesley, 1989.

[89] V. Iyengar, K. Chakraborty, and E.J. Marinissen, "Wrapper/TAM Co-optimization, Constraint –Driven Test Scheduling, and Tester Data Volume Reduction for SOCs." Proc. IEEE/ACM Design Automation Conference, pp. 685-690, 2000.

[90] A.Sehgal, V.Iyenger, M.Krasniewski and K.Chakraborty, "Test Cost Reduction for SOCs Using Virtual TAMs and Lagrange Multipliers," Proc. IEEE/ ACM Design Automation Conference, pp. 738-743, 2003.

REFERENCES

[91] Sandeep Korane, "On Test Scheduling for core based SOCs," In proceedings of IEEE Asia South Pacific Desing Conbference, Pages 505-510, Bangalore,India,January 2002.

[92] Erik Larsson, Zebo Peng and Gunnar Carlsson, "The Desing and Optimization of SOC Test Solutions," www.ida.liu.se/erlia/ ICCAD)!.pdf.

[93] J.H.Holland. Adaptation in Natural and Artificial Systems. Ann Arbor MI: University of Michigan press,1975.

[94] M. Sugihara, H. Date and H. yasuura, "A novel test methodology for core- based system LSIs and a testing time minimization problem," Proc. International Test Conference, pp. 465-472,1998.

[95] David E. Goldberg, John H. Holland, "Genetic Algorithms and Machine Learning," Vol. 3: pp. 95-99, 1998.

[96] Ebadi, Z. S. and Ivanov, A., "Design of an optimal test access architecture using a genetic algorithm," In Proceedings of IEEE Asian Test Symposium (ATS) (Kyoto, Japan, Nov.), 205-210, 2001.

[97] Erik Larsson, Zebo Peng and Gunnar Carlsson, "The Design and optimization of SOC Test Solutions," www.ida.liu.se/\~{}erila /ICCAD01.pdf.

[98] V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "Efficient wrapper / TAM co-optimization for large SOCs," Proc. DATE Conf., pp. 491-498, 2002.

[99] Park S., "A New Complete Diagnosis Patterns for Wiring Interconnects," Proc. DAC, pp. 203--208, 1996.

[100] Hashimoto, A. and Stevens, J., "Wire Routing by Optimizing Channel Assignment within Large Apertures," Proc. DAC, pp. 155-169, 1971.

[101] Oruganti, R., Sankaralingam, R. and Touba, N., "Static Compaction Techniques to Control Scan Vector Power Dissipation," Proc. VTS, pp. 35-42, 2000.

[102] Iyenger, V., Chakrabarty, K., and Murray B.T., "Built-in self testing of sequential circuits using precomputed test sets," Proc. IEEE VLSI Test Symposium, pages 418-423, 1998.

[103] Jas, A., Ghosh-Dastidar, J.,and Touba N.A., "Scan vector compression/decomposition using statistical coding," Proc. IEEE VLSI Test Symposium, pages 114-121, April 1999.

[104] Jas, A., and Touba N.A., "Test vector decomposition via cyclic scan chains and its application to testing core-based design," Proc. Intl. Test Conference, pages 458-464, 1998.

[105] Vranken, H., Hapke, H., Rogge, S., Chindamo, D., and Volkernik E., "ATPG padding and ATE vector repeat per port for reducing test data volume," Proc. Intl. Test Conference, pages 1069-1078, 2003.

[106] Chandra, A., and Chakrabarty K., "System-on-a-chip test data compression and decompression architectures based on Golomb codes," IEEE Trans. on CAD, 20:355-368, March 2001.

REFERENCES

[107] Golomb S.W., "Run-length encoding," IEEE Trans. Inf. Theory, IT-12:399-401, 1966.

[108] Chandra, A., and Chakrabarty K., "Frequency-directed run-length(FDR) codes with application to system-on-chip test data compression," Proc. IEEE VLSI Test Symposium, pages 114-121, April 2001.

[109] El-Maleh, A., and Al-Abaji R., "Extended frequency-directed run-length codes with improved application to system-on-a-chip test data compression," Proc.Intl. Conference on Elect. Ciec. and Systems, pages 449-452, 2002.

[110] Gonciari, P.T., Al-Hashimi, B.M., and Nicolici N., "Variable-length input Huffman coding for system-on-a-chip test," IEEE Trans. on CAD, 22(6) : 783

[111] Tehranipour, M.H., Nourani, M., and Chakrabarty K., "Nine-coded compression technique for testing embedded cores in SOCs," IEEE Trans. on VLSI Systems, 13(6):719-731, June-2005.

[112] Wolf, F.G., and Papachriston C., "Multiscan-based test compression and hardware decompression using LZ77," Proc. Intl. Test Conference, pages 331-339, 2002.

[113] Knieser, M.J., Wolf, F.G., Papachriston,C.A., Weyer, D.J., McIntyre, and D.R., "A technique for high ratio LZW compression," Design and Automation for Test in Europe, pages 116-121, 2003.

[114] Li, L., Chakrabarty, K., and Touba N.A., "Test data compression using dictionaries with selective entries and fixed-length indices," ACM Trans. on design automation of electronics systems, 8(4):470-490, October 2003.

[115] Iyenger, V., and Chakrabarty K., "System-On-Chip test scheduling with precedence relationship, preemption, and power constraints," IEEE Trans., on CAD of Integrated Circuits and Systems, (21):1088-1094, 2002.

[116] Chakrabarty K., "test scheduling for core based system using mixed integer linear programming," IEEE Trans. on CAD, pages 1163-1174, 2000.

[117] Jain, W., and Vinnakota B., "Defect-oriented test scheduling," Proc. VLSI Test Symposium, pages 433-438, 1999.

[118] Larsson, E., Pouget, J., and Peng Z., "Defect-aware SOC test scheduling," Proc. VLSI Test Symposium, pages 359-364, 2004.

[119] Marinissen, E. J., Kapur, R., Lousberg, M., Mclaurin, T., Ricchetti, M., and Zorian Y., "On IEEE p1500's standard for embedded core test," Journal of Electronic Testing: Theory and Application, 4/5(18) : 365-383, Aug. 2002

[120] Zorian Y., "A distributed BIST control scheme for complex VLSI devices," Proc. VLSI Test Symposium, pages 6-11, 1993.

REFERENCES

[121] Chou, R. M., Saluja, K.K., and Agrawal V.D., "Scheduling tests for VLSI systems under power constraint," IEEE Trans. on VLSI Systems, 2(5):175-184, 1997.

[122] Muresan, V., Wang, X., and Vladutiu M., "A comparison of classical scheduling approach in power constrained block-test scheduling," Proc. Intl. Test Conference, pages 882-891, 2000.

[123] Larsson, E., and Peng Z., "Test scheduling and scan-chain division under power constraint," Proc. Asian Test Symposium, pages 259-264, 2001.

[124] Iyenger, V., Chatrabarty, K., and Marinissen E. J., "Test access mechanism optimization, test scheduling and test data volume reduction for System-On-Chip," IEEE Trans. on Computers, (52):1619-1632, 2003.

[125] Iyenger, V., Goel, S.K., Marinissen, E.J., and Chakrabarty K., "Test resource optimization for multisite testing of SOCs under ATE memory depth constraints," Proc. Intl. Test Conference, pages 1159-1168, 2002.

[126] Huang Y. et al., "Resource allocation and test scheduling for concurrent test of core-based SOC design," Proc. Asian Test Symposium, pages 265-270, 2001.

[127] Huang, Y., Reddy, S.M., Cheng, W.T., Reuter, P., Mukherjee, N., Tsai, C.C. Samman, O., and Zaidan Y., "Optimal core wrapper width selection and SOC test scheduling based on 3-D bin-packing algorithm," Proc., Intl. Test Conference, pages 74-82, 2002.

[128] Jen Yi Wuu, Tung-chein Chen, and Yao-Wen Chang, "SOC test scheduling using the B* -tree based floor-planning technique," Asia Pacific Design automation Conference, pages 1188-1191, 2005.

[129] Sehgal, A., Iyenger, V., and Chakrabarty K., "SOC test planning using virtual test access architectures," IEE Trans. on VLSI Systems, 12, 2004.

[130] Sehgal, A., and Chakrabarty K., "Efficient modular testing of SOCs using dual-speed TAM architectures," Design and Automation for Test in Europe, pages 422-427, 2004.

[131] Li et al., "A hierarchical test methodology for System-On-Chip," IEEE Micro., 22(5):69-81,Sept./Oct. 2002.

[132] Sehgal, A., Goel, S.K., Marinissen, E.J., and Chakrabarty K., "p1500-complaint test wrapper design for hierarchical cores," Proc. Intl. Test Conference, 2004.

[133] Iyenger, V., Chakrabarty, K., Krasniewski, M.D., and Kumar G.N., "Design and optimization of multilevel TAM architectures for hierarchical SOCs," Proc. IEEE VLSI Test Symposium, pages 299-304, 2003.

[134] Chakrabarty, K., Iyenger, V., and Krasniewski M.D., "Test planning for modular testing of hierarchical SOCs," IEEE Trans. on CAD of Integrated Circuits and Systems, 2004.

REFERENCES

[135] Xu, Q., and Nicolici N., "Time/area trade-offs in testing hierarchical SOCs with hard megacores," Proc. Intl. Test Conference, pages 1196-1202, 2003.

[136] Wang, T.p., Tsai, C.y., Shich, M.D., and Lee K.J., "Efficient test scheduling for hierarchical core based design," Proc. Intl. Symp. On VLSI Design, Automation and Test, pages 200-203, April 2005.

[137] Chakrabarty, S., Monzel,J., Agrawal, V.D., Aitken, J., Barden,J., Figueras, j., Kumar, S., Wunderlich,H.J., and Zorian Y., "Power dissipation during testing: should we worry about it?," Proc. VLSI Test Symposium, page 456, 1997.

[138] Girard P., "Low power testing of VLSI circuits: problems and solutions," ISQED, pages 173-180, 2000.

[139] International SEMATECH. "The international technology roadmap for semiconductors(ITRS),"1999 edition, http://public/itrs/.net/1999siaroadmap/home.htm'.

[140] Needham W.M., "Nanometer technology challenges for test and test equipment," Computer, 11(32):52-57, Nov. 1999.

[141] Zorian, Y., Dey,S., and Rodgers M., "Test of future System-On-Chips," IEEE/ACM International Conference on CAD, pages 392-398, November 2000.

[142] Wang S., "Minimizing heat dissipation during test application," PhD thesis, University of Southern California, May 1998.

[143] Wang, S., and Gupta S.K., "ATPG for heat dissipation minimization during test application," IEEE Trans. on Computers, 2(47):256-262, February 1998.

[144] Wang, S., and Gupta S.K., "ATPG for heat dissipation minimization during test application," Proc. Intl. Test Conference, pages 250-258, 1994

[145] Kajihara, S., and Miyase K., "On identifying don't care inputs of test patterns for combinational circuits," IEEE/ACM International Conference on CAD, pages 364-369, 2001.

[146] Chakravarty, S., and Dabholkar V., "Two techniques for minimizing power dissipation in scan circuits during test application," Proc. Asian Test Symposium, pages 324-329, 1994.

[147] Dabholkar, V., Chakravarty, S., Pomeranz, I., and Reddy S.M., "Techniques for minimizing power dissipation in scan and combinational circuits during test application," IEEE Trans. on CAD of Integrated Circuits and Systems, 12(17):1325-1333, December 1998.

[148] Flores, P., Costa, J., Neto, H., Monteiro,J., and Marques-Silva J., "Assignment and reordering of incompletely specified pattern sequences targeting minimum power dissipation," Proc. Intl. Conf. VLSI design, pages 37-41, 1999.

REFERENCES

[149] Girard, P., Guiller, L., Landrault, C., and Pravossoudovitch S., "A test vector ordering technique for switching activity reduction during test operation," Proc. IEEE Great Lakes Symposium on VLSI, pages 24-27, 1999.

[150] Girard, P., Landrault, C., Pravossoudovitch, S., and Severac D., "Reduction of power consumption during test application by test vector ordering," IEEE Electronics Letters, 21(33):1752-1754, 1997.

[151] Girard, P., Landrault, C., Pravossoudovitch, S., and Severac D., "Reducing power consumption during test application by test vector ordering," Proc. Intl. Symp. on Circuits and Systems, pages 296-299, 1998.

[152] Vranken, H., Waayers, T., Fleury, H., and Lelouvier D., "Enhanced reduced-pin-count test for full-scan design," Proc. Intl. Test Conference, pages 738-747, October 2001.

[153] Gerstendofer, S., and Wunderlich H.J., "Minimized power consumption for scan-based BIST," Proc. Intl. Test Conference, pages 77-84, 1999.

[154] Hertwig, A., and Wunderlich H.J., "Low power serial built-in-self-test," Proc. IEEE European Test Workshop, pages 49-53, 1998.

[155] Saxena, J., Butler, K., and Whetsel L., "A scheme to reduce power consumption during scan testing," Proc. Intl. Test Conference, pages 670-677, October 2007.

[156] Whetsel L., "Adapting scan architectures for low power operation," Proc. Intl. Test Conference, pages 863-872, October 2000.

[157] Huang, T.C., and Lee K.J., "An input control technique for power reduction in scan circuits during test application," Proc. Asian Test Symposium, pages 315-320, 1999.

[158] Wang, S., and Gupta S.K., "ATPG for heat dissipation minimization during scan testing," Design Automation Conference, pages 614-619, 1997.

[159] Sinanoglu, O., and Oraiglu A., "Aggressive test power reduction through test stimuli transformation," Proc. Intl. Conf. on Computer Design, 2003.

[160] Sinanoglu, O., Bayraktaroglu, I., and Oraiglu A., "Scan power reduction through scan chain modification," Journal of Electronic Testing: Theory and Application, 2003.

[161] Sinanoglu, O., and Oraiglu A., "Scan power minimization through stimulus and response transformations," Design and Automation for Test in Europe, 2004.

[162] Gupta, S., Vaish, T., and Chattopadhyay S., "Flip-flop chaining architecture for power efficient scan during test application," Proc. Asian Test Symposium, 2005.

[163] Ghosh, D., Bhunia, S., and Roy K., "Multiple scan chain design technique for power reduction during test application time," Proc. IEEE Intl. Symp. on Defect and Fault Tolerance in VLSI systems, 2003.

REFERENCES

[164] Nicolici, N., and Al-Hashimi B., "Scan latch partitioning into multiple scan chains for power optimization in full scan sequential circuits," Design and Automation for Test in Europe, pages 715-722, 2000.

[165] Il-Soo Lee, Hur, Y. M., and Ambler T., "The efficient multiple scan chain architecture reducing power dissipation and test time," Proc. Asian Test Symposium, 2004.

[166] Vikram Iyengar, K. Chakrabarthy, and E. J. Marinissen., "Co-optimization of test wrapper and test access architecture for embedded cores," Journal of Electronic Testing, Theory and Applications, 8(2):213.230, April.

[167] P. T. Gonciari, B. M. Al-Hashimi, and N. Nicolici., "Improving compression ratio,area overhead, and test application time for system-on-chip test data compression/Decompression," Proceedings of Design, Automation and Test in Europe, pages 604.611, 2002.

[168] B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater., "A smartbist variat with guaranteed encoding," In Proceedings of the Asian Test Symposium, page 325330, Nov 2001.

[169] T. M. Cover and J. A. Thomas. Elements of information theory. 1991.

[170] D. Heidel, S. Dhong, P. Hofstee, M. Immediato, K. Nowka, J. Silberman, and K. Stawiasz., "High-speed serialiazing/deserializing design-for-test methods for evaluating a 1 ghz microprocessor," In Proceedings IEEE VLSI Test Symposium, pages 234.238, April 1998.

[171] Hamzaoglu and J. H. Patel., "Test set compaction algorithms for combinational circuits," In Proceedings of International Conference on Computer-Aided Design, pages 283.289, Nov 1998.

[172] Khoche, A., "Test resource partitioning for scan architectures using bandwidth matching," Digest of Int. Workshop Test Resource Partition, 2002, pp. 1.4.1-1.4.8.

[173] Krstic, A., and Cheng, K.-T., "Delay fault testing for VLSI circuits," Kluwer Academic Publishers, 1998.

[174] Xu, Q., and Nicolici, N., "Delay fault testing of core-based system-on-a-chip," DATE, 2003, pp. 744-749.

[175] S.Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi, "Optimization by simulated annealing," Science, pp. 671-680, Vol 220, No. 4598, 1983

[176] N. Metropolis, A. Rosenbluth, M. Rosenbluth. A. Teller. E. Teller, "Equation of state calculations by fast computing machines," J. Chem. Phys. 21. 1087 (1953).

[177] Harmanani H.M. and Farah R., "Integrating wrapper design, TAM assignment, and test scheduling for SOC test optimization," Joint 6th International IEEE Northeast Workshop on Circuits and Systems and TAISA Conference, 2008. NEWCAS-TAISA 2008.

[178] Goel S.K. and Marinissen E.J., "SOC test scheduling design for efficient utilization of bandwidth," TODAES, 8(4), pp. 399-429, 2003.

REFERENCES

[179] Su C. and Wu C., "A graph-based approach to power-constrained test scheduling," JETTA, 20, pp. 45-60, 2004.

[180] Pouget J., Larsson E. and Peng Z., "Multiple-constraint driven system-on-chip time oprimization," JETTA, pp. 599-611, 2005.

[181] Harmanani H.M. and Salamy H.A., "Power-constrained System-on-a-Chip Test Scheduling using Genetic Algorithm," Journal of Circuits, Systems and Computers, Vol 15, No. 3, 2006.

[182] Giri C., Mallikarjuna B., and Chattopadhyay S., "Split Variable-length Input Huffman code with application to test data compression for embedded cores in SOCs," *International Journal of Electronics*, 2009.