Test Infrastructure Design for Power Aware System-On-Chip Testing

Thesis submitted in partial fulfillment of the requirement for award of the degree of
DOCTOR OF PHILOSOPHY

by

Chandan Giri

Under the Guidance of
Dr. Santanu Chattopadhyay
Associate Professor

Department of Electronics & Electrical Communication Engineering
Indian Institute of Technology Kharagpur
Kharagpur-721302,West Bengal, INDIA
January, 2008

Dept. of Electronics & Electrical Communication Engg.
Indian Institute of Technology
Kharagpur-721 302.
India

CERTIFICATE

This is to certify that the thesis entitled **"Test Infrastructure Design for Power Aware System-On-Chip Testing"** submitted by **Chandan Giri**, a research scholar in the *Department of Electronics & Electrical Communication Engineering* at *Indian Institute of Technology, Kharagpur, India* for the award of degree of **Doctor of Philosophy** (Ph.D) is a record of bona fide research work carried out by him at the *Department of Electronics & Electrical Communication Engineering, Indian Institute of Technology, Kharagpur*, under my guidance and supervision.

In my opinion this work fulfills the requirements as per regulations of this Institute and has reached the standard needed for submission. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Dr. Santanu Chattopadhyay
Associate Professor
Department of Electronics and Electrical Communication Engineering,
Indian Institute of Technology, Kharagpur,
Ph: +91 322228 3564 (O)/3565 (R)
Fax: +91 322225 5303
e-mail: santanu@ece.iitkgp.ernet.in

Place: I.I.T, Kharagpur
Date:

To my Parents

Acknowledgements

The first person I would like to express my deepest appreciation is my advisor, **Dr. Santanu Chattopadhyay**, whose guidance, encouragements, enthusiasm to high-quality work and insightful advice helped me in all the time of research for and writing of this thesis. I will cherish the experience of learning and working with him forever.
In addition, I would like to thank **Dr. Indranil Sengupta, Dr. Anindya Sundar Dhar** and **Dr. Indrajit Chakraborty** for serving in my doctoral scrutiny committee. Many thanks also go to the faculty, administrative, technical and non-technical members in *Dept. of Electronics and Electrical Communication Engineering*, whose assistance was vital for this research. In particular, I would like to express my gratitude to **Dr. Swapna Banerjee**, who has given me permission to use the space and computing facilities of *CAD Lab* of this department. During the course of work, I had numerous pleasant moments and experiences with several friends and colleagues. Therefore, I would like to thank everybody who has supported me over the last two and half years. I would specially like to thank the following people:
My one time colleague **Prof. Narayan Chandra Maity**, for his support and encouragement throughout this research work. I would also like to thank all of my other colleagues at *College of Engineering & Management, Kolaghat, West Bengal*, specially, Suman, Jayanta and Kalyan for their warm friendship and encouragement. Ph.D students Saurbah Choudhury, Santanu Kundu, Tapas Maiti and Shambhu Pradhan for their constant support, encouragement and valuable discussions on some technical topics. M.Tech students Mallikarjuna, Naveen and Dilip and B.Tech students Pradeep, Nikhil and Soumojit for their fruitful discussions and contributory works related to my research topic. All the CAD Lab students and staffs, specially Shib Sundar Das (Das da), Apurba Adhikary (Apurba da) and Soma Sil (Soma di) for their friendliness, helps, encouragement and providing a home like environment to work in CAD lab.
Last but certainly not least, this work could have not been carried out without the unshakable love, encouragement and believing in me for higher studies of my parents, my brother, my sisters and their family and surely my wife Soma. Words cannot express my love and gratitude to my family.

Chandan Giri

Place: I.I.T, Kharagpur
Date:

Abstract

Rapid technological growth in the electronic design industry has made it possible to go for core-based System-on-Chip (SOC) design. Due to its complexity, testing SOCs provide a new dimension for the test engineers to exploit several strategies. Several standards (e.g., IEEE 1500 Std.) and state-of-the-art methodologies have been devised to get rid of the different testing challenges. However, several emerging problems need to be considered so that testing of SOC can be done in a cost-effective manner. For example, the existing SOCs can be used for the future generation SOCs. Hence, multi-level test infrastructure has to be designed in a cost effective way that also needs to be extended for multi-frequency domains.

This dissertation addresses three major issues from three different areas in the perspective of test resource optimization to reduce test cost. These correspond to test cost reduction via test time minimization, test data volume reduction and test power reduction. The solution shows how effectively test time can be reduced by test access architecture optimization and scheduling the cores taking power constraint into consideration. The work also addresses a dictionary and a non-dictionary based test data compression scheme. During compression, it also takes into account the reduction of scan power as well as the test bus power. A trade-off between compression and power has also been made that can be used as a tool by the test engineers to get how much compression is sufficient considering the power constraints and also the constraints from automated test equipment (ATE). Finally, a novel scan architecture has been introduced together with test vector reordering and proper filling of unspecified bits that reduces scan power significantly. Dissertation also shows how simultaneous reduction in power and delay (routing overhead) can be achieved for both single and multiple scan chains.

List of Symbols and Abbreviations

Symbols/Abbreviations	**Description**	**Definition**
SOC	System-On-Chip	page 1
ATE	Automated Test Equipment	page 6
UDL	User Defined Logic	page 2
RTL	Register Transfer Level	page 2
IP	Intellectual Property	page 3
PCB	Printed Circuit Board	page 3
DFT	Design For Testability	page 4
CTL	Core Test Language	page 5
STIL	Standard Test Interface Language	page 5
SI	Signal Integrity	page 6
TAM	Test Access Mechanism	page 7
CUT	Circuit Under Test	page 7
VTD	Volume of Test Data	page 8
VIHC	Variable Length Input Huffman Coding	page 10
GA	Genetic Algorithm	page 10
ILP	Integer Linear Programming	page 16
SC	Scan Chain	page 17
LPT	Large Processing Time	page 17
FFD	First Fit Decreasing	page 17
TDC	Test Data Compression	page 21
BIST	Built-In Self Test	page 21
ATPG	Automatic Test Pattern Generation	page 24
Δ	Small Change	page 35
LB	Lower Bound	page 49
PSF	Peak Switching Frequency	page 55
TAT	Test Application Time	page 112
CGU	Code Generation Unit	page 118
α	On Chip Test Frequency	page 123

Symbols/Abbreviations	**Description**	**Definition**
δ	Extra ATE Clock Cycles	page 123
$\ll$	Very Very Less	page 123

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Driven by the market, electronics industry continuously requires products with greater flexibility in terms of improved functionality, higher reliability, lower cost and shorter time-to-market. It is known that integrated circuit (IC) chips are considered as the foundation of the modern electronic products. Hence the need of faster and smaller products motivates the industry to develop complex chips. These chips can incorporate a wide range of complex functions that build an entire system, into a single die, called *System-On-Chip* (SOC).

One of the important issues in making SOC production practical and cost effective is the complexity of creating multi-billion transistors using conventional methods and design flow. To cope up with this problem, the design community has created a new chip manufacturing paradigm based on reuse philosophy [8], in which pre-designed and pre-verified building blocks with some extra logic are integrated to build the complex systems. These pre-designed and pre-verified building blocks are called *cores* and the new design paradigm that uses these cores is called core-based design paradigm. So, SOCs are composed of embedded cores that make it easier to import existing technologies and thereby shorten time-to-market through design reuse.

1.1 Core-Based Design Paradigm

The modern SOC often contains one or multiple programmable CPUs, DSP cores, application specific hardware blocks, embedded memories, network controllers and some peripherals [1]. Hence, the embedded cores cover a wide range of system functions with different design styles. One core may contain several other cores, which are called *hierarchical cores*. Hence the SOC may contain from sim-

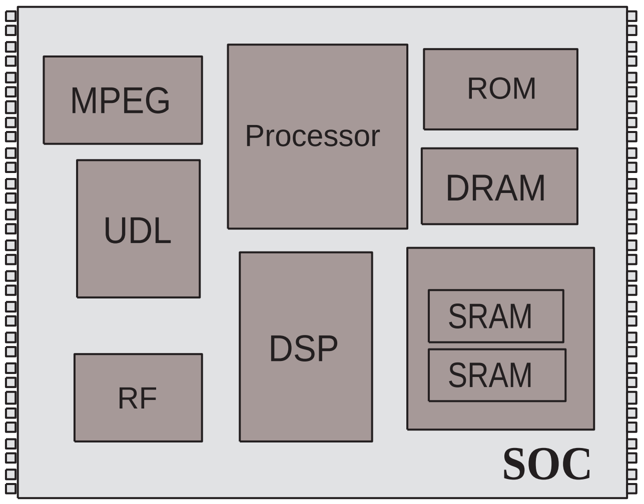


Figure 1.1: An example of core-based SOC[1]

ple cores to complex cores (hierarchical cores). One example of such is given in Fig. 1.1. Successful design of such a complex single-chip system requires expertise in diverse technology areas such as signal processing, encryption, and analog and RF designs, which are increasingly hard to find in a single design house. For this reason pre-designed and pre-verified blocks are being delivered from multiple sources. This causes various trade-offs and formats and hence enable the requirement of plug-and-play design methodology like IC design environment. This SOC design paradigm comprises of core vendors as providers and core users as customers. Customer here is a system integrator who combines all the available cores together with their custom user-defined logic (UDL) [1] to interface different cores such that the desired functionality is obtained. Hence core providers design and maintain a library of reusable cores, while a core user is responsible for design and manufacturing of system chips using various cores and user-defined logic modules.

Embedded cores provided by the core vendors are represented in hardware description levels. They spread from fully optimized layouts in GDSII format to widely flexible RTL (register transfer level) format. Depending on the hardware description levels, cores can be classified into the following three major categories [9]:

- *Soft cores* consist of synthesizable representation in the form of hardware description language (HDL) code. The advantage is that it can be re-targeted for different technologies and system constraints. As a HDL descriptions (behavioral or RTL), these cores leave much of the implementation to the designer. This flexibility also contributes to the unpredictability in area, performance and power requirement of the synthesized circuit.

- *Firm cores* usually consist of gate level net list representation ready for place and route. They give some flexibility to designer, are accompanied by simulation models and fit into standard HDL design methodologies. Hence, their size, aspect ratio and pin locations can be tuned according to the system integrator's needs.

- *Hard cores* on the other hand come with physical layout information and hence are technology-dependent. Despite the lack of flexibility, they are optimized for timing or power and have well known performance parameters.

Cores are often products of technology, software, and know how that are subject to patents and copyrights. Hence a core block represents intellectual property (IP) that the core provider licenses to the system integrator. Hence despite the core integration, verification and also manufacturing test of the entire SOC, system integrator is not always entitled to make any changes to the core and is forced to reuse it "as is", knowing only the cores' functionality without any implementation details.

Besides the immediate advantages of core based SOCs, the SOC paradigm brings forth new problems from the design and test perspective [1, 10, 2, 11], which require new test strategies for testing the core itself as well as the entire system [11, 12, 13, 14, 15, 16]. Hence, manufacturing test, a key step at the back end of the implementation flow of VLSI circuits, used to isolate good chips from the defective ones, has become an essential technology that enables the fabrication yields, certifies the product quality and influences the final cost of the devices. In an SOC design, ensuring an efficient manufacturing test methodology has become a challenge.

1.2 Why SOC testing is needed?

Although an SOC is made by simply "plug-and-play" of cores to make functions similar to the traditional Printed Circuit Board (PCB) design or System-On-Board (SOB) design, the SOC paradigm brings many challenges. As shown in Fig. 1.2(a) the IC providers perform chip design, manufacturing and test. The system integrator can assume the ICs to be fault free and is responsible for the design, assembly and test of the PCB. Hence, in this case, interconnection between the chips are to be tested. Whereas, in a core-based system-chip design, as shown in Fig. 1.2(b), the embedded cores are not yet manufactured when the system integrator puts them together; thus the core providers cannot test their products before the chip is actually fabricated. Therefore, SOC designer is responsible for manufacturing and testing of not only the interconnects between the cores, but also the cores themselves.

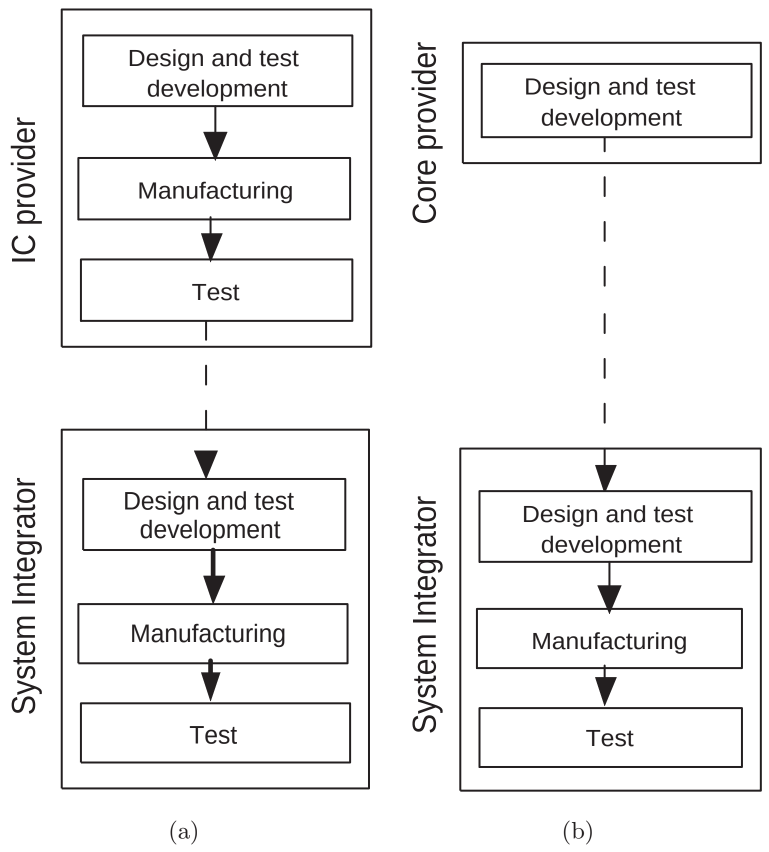


Figure 1.2: Design Development Strategies for PCB and SOC[2]

Another major difference between SOB and SOC is the accessibility of component pins. Every pin can be accessed directly and physically for probing necessary for manufacturing test of the SOB. But, in the case of an SOC, the cores are deeply embedded and hence no direct physical access to pins of the cores is possible by default. Since the system integrator does not have any knowledge about the design of the cores, the core provider has to provide the model, the design for testability (DFT) [17] structures and corresponding test vectors. The main difficulty for the system integrator is to provide a reliable and scalable test infrastructure for accessing the cores. This is required due to no direct access to the cores' terminals.

1.3 Core Based SOC Test

There exists several testing issues for the traditional deep-submicron chips, such as fault coverage, overall test cost and time-to-market. Apart from all these issues, testing SOC has three main challenges.

1. **Core Internal Test:**
   A core is generally provided by a core vendor in terms of hardware description. The internal test of a core is typically composed of required set of test patterns (both stimuli and responses) and internal DFT structures like scan architecture, test points or BIST if any. Hence for internal testing of the core, it requires in depth knowledge about the internals of the core. But due to the limited knowledge of the system integrator, it is very difficult to prepare a test for it, especially if a core is a hard one or is an encrypted intellectual property block. Hence, core providers should provide two things. Firstly, the parameters of the description of the core and secondly, the DFT structures and test patterns.

   IEEE Std. 1500 standardizes a test information transfer model, that is, IEEE 1450.6 Core Test Language (CTL) [18, 19] or Standard Test Interface language (STIL(IEEE 1450.0)) [20]. IEEE Stds. however, does not standardize the core's internal test methods or DFT structures.

2. **Core Test Access and Core Isolation:**
   In SOCs, cores are deeply embedded in the environment and usually their terminals are not directly accessible from the SOC pins. So it is needed to provide a test access path from the SOC primary pins to the embedded cores and vice-versa with sufficient bandwidth to fulfill the test requirements of the cores.

   Again, in order to apply the given set of tests to cores, modular testing has been advocated to simplify test access and test application [2]. To facilitate modular test, an embedded core must be isolated from surrounding logic and test access must be provided from the I/O pins of the SOC. Test wrappers are used to isolate the cores, while test access mechanism (TAM) transports test patterns and test responses between SOC pins and core I/Os [2].

   To enable the reuse of tests, the IEEE Std. 1500, that is, IEEE Standard for Embedded Core Test (SECT) [21, 22], has been developed by the IEEE working group [23]. A module-level test wrapper similar to IEEE 1149.1 boundary scan architecture [24], is suggested to surround each embedded core, that allows inter-core and intra-core tests to be carried out via TAMs. The test access to embedded cores and isolation of cores are the responsibilities of the SOC designer, that is, system integrator.

3. **Test Integration and Test Scheduling:**
   The test time for core-external interconnect shorts/opens is typically much less than that for core-internal logic. But as feature size is shrinking for newer process technologies, the test time for interconnect signal integrity (SI) cannot be neglected [25]. Hence, SOC designer needs to develop tests for the interconnect wiring and glue logic between the cores and the test facility has to be integrated under the same SOC-level test control mechanism that also controls the core-internal tests. The SOC-level test control mechanism is required to execute various tests and apply/capture the necessary test data. In addition to test integration, SOC test requires efficient test scheduling. The scheduling of core tests should be such that there are no test resource conflicts. During scheduling, it also has to be kept in mind that the chip-level constraints like test time and power dissipation during test are to be satisfied.

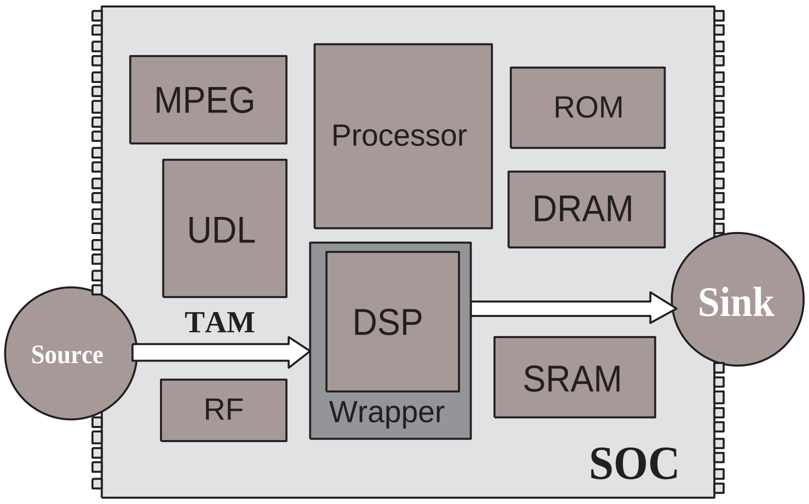


Figure 1.3: Conceptual SOC Test Architecture[2]

Driven by these testing challenges Zorian et al [2] presented a conceptual test architecture for testing embedded core-based SOCs that meets the requirements discussed above. Fig. 1.3 shows the conceptual test architecture for an example SOC. The architecture has the following three elements:

1. **Source and Sink:** Source and sink are used for providing the test stimuli and analyzing the test responses respectively. Source and sink can be implemented either off-chip or on-chip. The former (i.e. off-chip) is referred to as external deterministic test where the source and sink are represented by the automatic test equipment (ATE). For the latter (i.e. on-chip) the internal source and sink are represented by BIST structure. The source and sink can also be implemented by the combination of both.

2. **Test Access Mechanism (TAM):** Test access mechanism is responsible for transporting the test stimuli from the source to the circuit under test (CUT), and the responses from the CUT to the sink. Basically a TAM is a set of wires that makes physical communication between the sources, core and sink. In general, a wider TAM reduces the test application time, however, it will also increase the wiring overhead. Hence a trade-off exists between the test application time and the TAM bandwidth.

3. **Core Test Wrapper:** Core test wrapper provides an interface between the embedded core and its environment. It connects the terminals of the embedded core to the rest of the chip and the TAM. In case of mismatch between the number of core terminals and TAM width, the wrapper also provides width-adaptation by means of serial or parallel conversion. A core test wrapper must support the following mandatory modes [26, 23, 18]:

   - *Normal-operation:* when the core test wrapper is transparent and the core is connected to the chip environment.
   - *Core-internal test:* where the core test wrapper connects the TAM to the core such that the test patterns can be applied and the test responses observed.
   - *Core-external test:* when the testing of the interconnect logic and wiring is allowed by the core test wrapper.

All the above elements can be implemented in several ways each having their respective specific advantages and disadvantages with respect to the silicon area and test application time.

1.4 Cost of Test

Nag et al. [27] presented one test cost model that pointed out that the cost of test (C_{test}) consists of four main components:

$$C_{test} = C_{prep} + C_{exec} + C_{silicon} + C_{quality} \tag{1.1}$$

Where,

- C_{prep} captures the fixed costs of test generation, tester program creation, and any design effort for incorporating test-related features (all non-recurring costs, including software system).
- C_{exec} consists of costs of test-related hardware such as probe cards and cost incurred by tester use.
- $C_{silicon}$ is the cost required to incorporate DFT features.

- $C_{quality}$ is often ignored but is probably one of the most important parameters of the test cost. It is the cost of imperfect test quality. That is, the profit loss from performance degradation caused by the added DFT circuitry, the cost of test escape (the relation between fault coverage and fault occurrence – or simply yield), and cost of good dies being deemed faulty by imperfect test.

Eqn. (1.1) represents the cause-and-effect relationship between various test, manufacturing, and design attributes. Test engineers must make trade-offs among the above costs and apply a test strategy to decrease the overall cost. In order to reduce the test cost all the four parameters are sought small, C_{prep} is a fixed cost per design, $C_{quality}$ is directly dependent on the core DFT methodology and $C_{silicon}$ is a requirement for providing core level DFT. Hence, cost reduction in manufacturing test is possible if the C_{exec} is reduced. The cost of test execution can be approximated using the following equation:

$$C_{exec} \approx C_{hw} + T_{tester} \times C_{tester} \tag{1.2}$$

where, T_{tester} is the time spent on tester and has the two components $T_{inactive}$ and T_{active}. $T_{inactive}$ is mainly due to the inefficient tester usage. T_{active} on the other hand, is determined by the volume of test data (VTD), needed to test the CUT, and the ATE parameters, as illustrated below.

$$T_{active} = \frac{VTD}{n_{ate} \times f_{ate}} \tag{1.3}$$

where, n_{ate} represent the number of ATE channels used for the testing and f_{ate} represents the Automated Test Equipment (ATE) operating frequency. The product $n_{ate} \times f_{ate}$ is the bandwidth of the ATE. The test time increases due to the continuous increase in the VTD and the insufficient bandwidth when compared to the requirements.

1.5 Thesis Motivation

- **How to effectively and efficiently test SOCs with reduced test time and also with the cores with multiple levels of hierarchy?**
  As one of the objectives of SOC testing is to reduce the test application time, scheduling test solutions for embedded intellectual property (IP) cores can be used to achieve it. So to meet the shrinking product development schedule, test scheduling problem is one of the major issues in test integration for the SOCs. Again most of the prior works on this field assume that the SOC hierarchy is flattened during test. This assumption is unrealistic. As SOC design is based on reuse philosophy, the current generation SOCs can be embedded into the future generation SOC designs, thus leading to

a hierarchical SOC design paradigm. Therefore, novel test strategies for hierarchical SOCs need to be developed.

- **How to deal with increasing volume of test data?**
  As the number of cores within a chip for SOC design is increasing day by day, volume of test data (VTD) is also increasing. This also causes increase in test application time and ATE memory requirements. For the ATE to account for the continuous increase in VTD, upto 60% of the invested capital for ATE upgrade is needed to meet the new memory capacity requirement. Again due to the increase in VTD, power consumption during test is also increasing. Hence increasing VTD should be handled in power efficient manner so that test requirements can be met.

- **How to deal with the increased power consumption during testing?**
  As both the SOC designs and the deep submicron geometry become prevalent, larger designs, tighter timing constraints, higher operating frequencies and lower applied voltages all affect the power consumption in systems of silicon devices. More precisely, these factors affect energy, average power, instantaneous power and peak power. The industrial article [28] reports test power upto 30X higher compared to the normal operating mode. Hence test strategy to reduce power during testing is an open challenge.

1.6 Thesis Contributions and Organization

Providing a low-cost test solution for core based SOC does not only require the understanding of the factors that drives the cost, but it also imposes the need to comprehend the implications and limitations of previous approaches which address these factors. Chapter 2 addresses the various methods already presented in the literature regarding the SOC testing issues and challenges that have been considered in this research work. The dissertation gives emphasis mainly on three different test parameters, namely, test time, test data volume and test power. This includes the integrated approach of test architecture design and test scheduling that tries to reduce the overall test time, new test data compression schemes that try to reduce test data volume together with test application time and scan-based DFT architecture modification for reduction in power. Detailed literature survey in each of these areas have been introduced in Chapter 2. Based on the analysis and with the prime focus on reducing the cost of test, the next seven chapters detail the contributions of this dissertation.

Chapter 3 addresses the integrated solution for effective and efficient design of SOC test architectures with test scheduling. A best-fit heuristic based rectangle packing approach has been used for minimization of overall test time during test scheduling of the cores. For this purpose Genetic Algorithm (GA) has been considered to select better representative rectangle from each core present in the SOC. This work considers test scheduling for the flat cores only where all the cores are assumed to be at the same level of hierarchy. Power constrained test scheduling has also been addressed during minimization of overall test time.

Chapter 4 addresses the issue of test scheduling using TAM width partitioning. Here TAMs are considered to be operating in two different frequencies. Hence depending upon the scan frequency of the cores, TAMs are allocated. Effect of frequency on test power has also been considered during minimization of overall test time. GA has been used for optimal TAM width partitioning and assignment of cores so that overall test time is minimized.

In Chapter 5, the problem of test architecture design for SOCs with hierarchical cores is addressed. First a generic hierarchical core model is presented and next testing requirements for hierarchical cores are considered. Wrapper design aspects for hierarchical cores have been detailed. It has been followed by a GA based test scheduling approach using TAM width partitioning. The results are presented for three hierarchical cores present in ITC'02 benchmark circuits.

Chapter 6 presents a test data compression scheme based on dictionary that produces variable length indices using Huffman coding scheme. The proposed compression scheme leads to simultaneous reduction in the volume of test data and area overhead with respect to the previously reported dictionary based approach, that uses fixed length indices. A trade-off between compression and scan power is also presented based on filling of unspecified bits. This chapter also addresses the issue of test bus power reduction schemes by modifying the Huffman tree. The results are presented on ISCAS'89 benchmark circuits.

In Chapter 7 a compression scheme called *Split Variable Length Input Huffman Coding* (SVIHC) has been presented. Instead of using a single Huffman tree, as in VIHC scheme [7], proposed scheme considers two Huffman trees that improves the test application time and increases the reduction in volume of test data. However, the decoder clubs two separate Huffman trees into a single FSM.

Chapter 8 addresses the issues of reduction of both power and delay during scan based test. For this purpose, scan cell reordering, test vector reordering, scan architecture modification, test vector customization and proper filling of unspecified bits in the test set have been considered. The scheme has also been extended to multiple scan chains.

In Chapter 9, a new scan architecture has been proposed for power reduction. Instead of using only the D flip-flops, the proposed scan architecture utilizes a combination of both D and T flip-flops. Use of T flip-flops do not change the functionality of the original circuit. Genetic Algorithm has been used to obtain proper combination of D and T flip-flops to get maximum power reduction. A heuristic algorithm has also been presented for test vector reordering to get further reduction in power.

Finally, Chapter 10 summarizes the present work and enumerates the future scopes.

1.7 Conclusion

This thesis addresses several issues in SOC testing. Three main problems have been attempted – test scheduling, test data compression and test power reduction. To start with, in Chapter 2, a survey has been presented of the related works.

Chapter 2

Literature Survey

2.1 Introduction

Increasing complexity of core-based SOCs has forced test engineers to think about the modular test development approach. In this approach, test resources are partitioned and optimized to achieve effective and high quality testing at a reasonably low cost. There are mainly three types of SOC test resources. These are:

1. **Test Hardware:** Refers to special purpose hardware used for test generation and test application. This hardware can be off-chip or on-chip.

2. **Testing Time:** Refers to the time required for manufacturing test. This also includes the test application time for an SOC.

3. **Test Data:** Refers to the set of test patterns, test responses and control signals that are to be applied to the SOC for testing.

For each and every step of testing, it is needed to reduce the test cost by proper utilization of all the resources. Hence, a vast amount of research has endeavored to provide a better understanding of these areas. A large number of test strategies and algorithms have been developed to reduce the test cost. Test resource partitioning techniques can be broadly classified as shown in Fig. 2.1.

In this research work we consider one area from each of the test resource partitioning techniques. These are *test scheduling*, *test data compression*, and *scan chain organization*. Motivation behind choosing these areas has been described in Section 1.5.

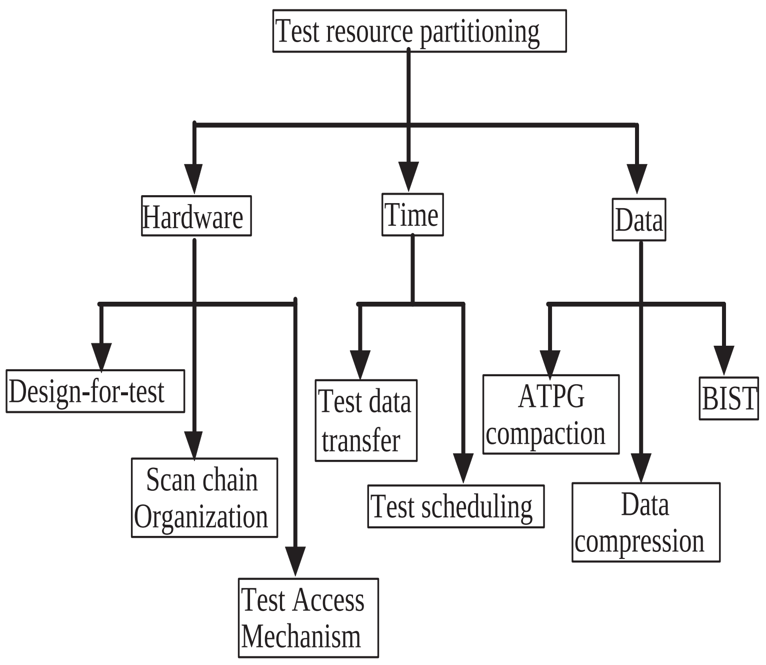


Figure 2.1: Test based on test resource partitioning[3]

In the following sections we will present a detailed survey of the existing techniques corresponding to each of the areas considered in this research work. Section 2.2 discusses the existing techniques on test scheduling. Works on test data compression have been reported in Section 2.3. Section 2.4 reviews the existing techniques on low power testing. Finally, Section 2.5 concludes the chapter.

2.2 Prior Works on Test Scheduling

Modular testing of embedded cores in an SOC is being increasingly advocated to simplify test access and test application[2]. To facilitate modular test, an embedded core must be isolated from surrounding logic, and test access must be provided from the I/O pins of the SOC. Test wrappers are used to isolate the core, while test access mechanism (TAM) transports test patterns and test responses between SOC pins and core I/Os. An effective modular test requires efficient management of the test resources. This involves the design of core test wrappers and TAMs, the assignment of test pattern bits to the ATE channels, the scheduling of core tests and assignment of ATE channels to the cores of SOC.

One of the important issues for core-based SOC testing is to design an effective and efficient TAM. A number of solutions exist for accessing the embedded cores from chip I/Os [2]. These are illustrated below.

1. **Direct parallel access via pin muxing:** This is the simplest approach. In this case, test patterns are applied directly and responses are observed through the chip level pins. Hence use of wrapper is unnecessary. But the main problem is that, as the number of cores are increasing, chip level pins are also increasing. Hence direct access requires large routing overhead and is not easily scalable.
   Bhatia et al. [29] proposed a grid-based *CoreTest* methodology that uses test-points like storage elements, embedded cells and observation points. Direct parallel access is made through the "soft" netlist to these test points via SOC I/O pins. In this method test logic and wiring can be shared, thus reducing the hardware overhead.

2. **Isolation ring access mechanism:** This is a serial access methodology with core isolation through a boundary scan-like structure. IEEE 1149.1 test architecture is a widely used DFT architecture. In this method a serial scan chain is built around every core that allows indirect yet full access to all the I/Os. This mechanism can be implemented either internally or externally. In [30], Whetsel presented a test access architecture that utilizes the 1149.1 test access port (TAP) and a novel TAM linking module for the overall SOC test control. This test access mechanism has low routing overhead and supports testing cores with more ports than chip pins. In [31], Touba et al. proposed a partial isolation ring based architecture. This avoids adding logic on critical paths. Later, same authors also proposed another scheme [32] that removes the need of isolation rings for certain cores.

3. **Functional access through functional buses:** Harrod [33] proposed a test strategy employed by ARM that enables test access through reusing 32-bit functional buses. As in this case functional buses are used, hardware overhead is very low. The proposed method is best suited for cores that are functionally tested and have a small number of non-bus I/Os. But this method allows testing only one core at a time, thus increasing the overall test time.
   Beenker et al [34] proposed *MacroTest* that uses functional paths for test data transfer to test core-based SOCs. Ghosh et al. [35, 36] proposed a test access architecture that requires all the I/Os of a core to be controllable/observable simultaneously, though indirectly. In [37] authors showed that complete controllability/observability may not be required and proposed to provide them on an "as needed basis". This method is more suitable for testing complex cores.
   To guarantee the consecutive propagation of arbitrary test stimuli/response sequences from the core's inputs to the core's outputs, Yoneda and Fu-

jiwara [38] introduced the concept of consecutive transparency of cores. Chakrabarty et al. [39] also had considered the same problem. But area overhead is smaller in case of [38] for making cores consecutive transparent with some latency, instead of being single-cycle transparent. Later, same authors in [40] used integer linear programming (ILP) to decrease hardware overhead further to make soft cores consecutively transparent with some bypass mechanisms using multiplexers. In [41] also, Nourani and Papachristou used a similar technique for core transparency. *Bypass* mode is considered using multiplexers and registers. In this technique a weighted graph based model has been used and accessibility of the core input and output is solved as a shortest path problem.

4. **Dedicated test buses:** In this approach, the core I/Os are accessed through a combination of core wrappers and dedicated test buses. Three basic types of test architectures were introduced by Aerts and Marinissen in [42]. These are Multiplexing, Daisy chain and Distribution architectures. Besides these test architectures, two other test architectures have been introduced in [43] and [44]. These are Test Bus [43] and TestRail [44] architectures. Test Bus architecture has used the concept of multiplexing and daisy chain architectures, whereas TestRail architecture can be seen as the hybrid of Daisy chain and Distribution architectures.

In the following section we will survey the proposed techniques for test scheduling and test architecture optimization.

2.2.1 Test Architecture Optimization and Test Scheduling

Optimization of modular test architectures and test scheduling are the important issues in testing of SOCs. An extensive research has been carried out in this area and still it is the subject of interest for the researchers. For an SOC with specified parameters of its cores, a test architecture and a test schedule are needed to be designed carefully so as to minimize the test cost like test application time and test area overhead.

Various constraints also need to be considered during test scheduling, for example, the *power constraint.* Parallel testing of cores, though reduces test time significantly, increases the test power. In addition, *concurrency constraints* may exist due to test resource sharing (like a wrapped core cannot be tested at the same time with its surrounding unwrapped User Defined Logic (UDL) because both of them use the wrapper boundary cells during test). Again, there may exist a user-defined partial ordering for the cores, the order in which the cores are to be tested. This is called *precedence constraint.*

Since wrapper design, TAM optimization and test scheduling are correlated to each other, review is started with the wrapper design optimization.

Wrapper Design and Optimization

Test wrapper design is important during system integration since it has the direct impact on test time. IEEE Std. 1500 standardizes the wrapper interfaces. Therefore, the internal structure of the wrapper can be adapted to the specific SOC test requirement.

Designing the wrapper mainly involves the construction of wrapper scan chains (SCs) that comprises of a number of wrapper boundary cells and/or core internal scan chains. This wrapper SCs determine the test time of the core and need to be interfaced with the TAM channels. Core test application time depends on the length of the wrapper scan chains and the wrapper chain of maximum length determines the core test application time. Hence, the main objective in wrapper optimization is to optimize the test time. Again two types of wrapper chains can be designed. One is *balanced wrapper* and the another one is *unbalanced.*

Marinissen et al. [45] first addressed the issue of designing the optimized wrapper consisting of balanced wrapper SCs. In [45] authors proposed two polynomial time heuristic approaches. The *LPT* (Large Processing Time) approach, which was originally used for Multi-Processing Scheduling problem, was adapted to solve the wrapper design problem in a very short computational time. The authors also presented another heuristic named *COMBINE* that obtained better results by using *LPT* as a starting solution, followed by a searching method (linear search) over the wrapper SC lengths with the First Fit Decreasing (FFD) heuristic. Iyenger et al. [46] proposed the *Design_wrapper* algorithm based on the Best Fit Decreasing heuristic, that tries to minimize the test application time and required TAM width at the same time. Authors also showed that test application time varies with the TAM width in a "staircase" nature. According to this nature only a few TAM widths between 1 and W (maximum available TAM width) are used when assigning TAM resources to the hard cores and these discrete widths are called *Pareto-optimal* points.

Reddy et al. [47] showed that for the cores having no internal scan chain, *Design_wrapper* algorithm does not always give lower test application time. In this case unbalanced wrapper design can be used to get a lower test time.

Koranne et al. [48, 49] proposed a reconfigurable wrapper that allows dynamic change in the TAM width while executing the core test. This is achieved by deploying extra hardware (multiplexers) at the input and output of each reconfigurable scan chain. In [50] authors proposed a power-conscious reconfigurable core wrapper design that connects the inputs of each scan chain to multiplexers, but not the outputs of the core. The reconfigurable core wrapper is useful for cores having multiple tests, where each of the tests has three different TAM width requirements. Xu et al. [51] proposed a wrapper design that supports multi-clock domains. It also supports multi-frequency at-speed testing. In this wrapper design approach they partitioned the wrapper elements into different virtual cores.

In [52], Goel proposed a wrapper architecture design for hierarchical cores that allows parallel testing of the parent and child cores, at the cost of an expensive wrapper cell design. In [5] Sehgal et al. also presented wrapper design for hierarchical cores which is compatible with the IEEE 1500 Std.

Test Scheduling

Test scheduling is the process in which different test resources are allocated to cores at different time instants targeting to reduce the overall test application time. Hence 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.
- power dissipation constraints and
- precedence constraints.

Test scheduling techniques [53, 54] can be either session-based, sessionless or preemptive. Generally, session based testing leads to long test application time because of the division of the schedule in test sessions. Hence most of the test scheduling schemes proposed are sessionless. Testing time can be decreased further through the selective use of test pre-emption[55]. It is mainly constraint driven test scheduling scheme. It is shown in [56] that test scheduling problem is *NP*-hard in nature.

As power consumption is one of the main design constraints for the circuits, power-constrained test scheduling is very much necessary to avoid the chip malfunction during test. Power-constrained test scheduling, therefore attempts to reduce the amount of concurrency during test application to ensure the maximum power budget of SOC. Huang et al [57] formulated the problem of SOC test scheduling using 3-D bin-packing approach considering power constraints. Iyengar et al. [58] presented a heuristic approach based on rectangle packing problem formulation by exploiting the Pareto-optimal TAM widths of the cores. They also extended their algorithm in [59] to incorporate precedence and power constraints by taking into consideration the preemption of tests. Several other works [60, 61, 62, 63, 64, 65, 55, 66] also consider SOC power dissipation constraints during scheduling. In [65] authors considered the test resources as queue and the core tests to be scheduled as the job entering corresponding queue. Power constraint was then imposed in the test scheduling problem as the single-pair shortest path problem by constructing graph. Chou et al. [66] proposed a method based on approximate vertex cover of resource-constrained test compatibility graph.

Since wrapper design, TAM optimization and test scheduling all have direct impact on the SOC test cost, all the three need to be considered together to achieve the optimized result. Next we will review the existing techniques for the integrated problem.

Integrated wrapper/TAM co-optimization and Test Scheduling

Modular test architecture can be classified into two different categories [58] depending on the TAM channel assignment strategy.

1. *Fixed-width test architecture:* In this case, total TAM width is partitioned among several test buses of fixed widths. Each core is then assigned to exactly one of the partitioned test buses.

2. *Flexible-width test architecture:* In this case, each core in the SOC can get any TAM width within the limit of the maximum SOC-level TAM width. Hence TAM lines are allowed to fork and merge, instead of partitioning into test buses.

In literature, various optimization algorithms for fixed-width test architecture have been proposed. Iyengar et al. [46] first proposed the integrated wrapper/TAM co-optimization problem using Integer Linear programming (ILP) model. Same authors also proposed a heuristic approach that reduces the computation time at the cost of increased test application time [67]. A graph based approach has been proposed by Koranne [68], where it considered the minimum average completion time criteria. Instead of the NP-nature of the problem, the graph-theoretic approach that considers the minimum weight perfect bipartite graph matching takes polynomial time. Koranne in [69] also formulated the test scheduling as a network transportation problem, where a 2-approximation algorithm using the result of the single source unsplittable flow problem has been proposed. All of these approaches consider the Test Bus architecture. Another efficient heuristic algorithm TR-Architect proposed by Goel and Marinissen [70] supports both Test Bus and Test Rail architectures. They also provide the lower bounds on SOC test application time. Same authors also extended the TR-Architect algorithm to minimize both test time and TAM wire length in [71].

Though the above mentioned approaches used the fixed-width test architecture, the main demerit of these is that it wastes part of the test resources due to the “staircase” [72] nature (for hard cores) of the test application time over TAM widths. For example, if a core is assigned to a TAM of width w and the same testing time can be achieved using only $w' < w$ wires, it leads to a waste of test hardware. This also leads to idle bits stored on the tester for extra $w - w'$ wires that increases tester data volume.

Due to this limitation, various approaches used the flexible-width test architecture. But it cannot always be concluded that flexible-width architecture is better than fixed-length architecture as discussed in [73]. Huang et al [74] formulated the problem of SOC pin allocation to cores and test scheduling using 2-D bin-packing or rectangle packing approach. A heuristic approach using the sequence pair representation for test scheduling was considered in [75]. Zou et al [47] proposed test scheduling algorithm based on simulated annealing (SA) using the sequence pair representation. A B*-tree based approach has been proposed in [76] to get the test schedule. Ant colony optimization (ACO) based approach [77] considers the rectangle packing for test scheduling solution. A two-stage genetic algorithm (GA) based technique was proposed in [78] where each solution is represented by a sequence pair.

It can be noted that all the above mentioned approaches assume static TAM assignments i.e TAM width is fixed during test. Koranne [48] first proposed a reconfigurable core wrapper approach that allows a dynamic change in the TAM width during the execution of core test, so that it may lead to more efficient test schedule. Recently in [79] SOC test scheduling with reconfigurable core wrappers has been used. Although reconfigurable wrappers lead to efficient test schedules, more gate and routing overheads are imposed. It also increases the control complexity of the wrapper.

Sehgal et al. [80] proposed a scheme that matches the ATE channels with higher data rate with core scan chain frequency using virtual TAMs considering the availability of dual-speed ATEs. But main drawback is the need for higher number of TAM wires on the SOC and the extra frequency division hardware for bandwidth matching. Same authors in [4] also proposed a TAM architecture optimization with test scheduling scheme based on rectangle packing approach considering two available data rates for the ATE channels. But it does not need any extra hardware. Authors in [81] proposed multi-frequency TAM architecture optimization and core scheduling using TAM width partitioning so that port-scalable testers can be used efficiently.

2.3 Prior Works on Test Data Compression

Due to the increase in number of cores in an SOC, enormous rise in the test data volume is necessary to achieve desired fault coverage. For example, the test data volume can be as high as several gigabits for an industrial ASIC [82]. As the cost of ATE is very high, most IC manufacturers do not replace ATEs with new SOC design. Due to the limitations of the older ATEs in terms of tester memory and bandwidth, test data compression (TDC) techniques are utilized to overcome the associated problems. Again, for full-scan embedded cores, the patterns generated by ATPG tools have a huge amount of unspecified bits ("don't care" or X) which can be assigned either to 0 or 1. By exploiting these don't cares test data volume can be reduced and simultaneously scan power can also be minimized. Following sections review the test data compression techniques and scan power reduction during compression.

2.3.1 Test data compression schemes

A number of compression techniques based on ATE-SOC interaction have been used for test data volume reduction. Though Built-In-Self-Test (BIST) is an alternative approach to reduce the need for expensive ATE, it does not give high fault coverage and for some of the tests it may lead to unacceptably long test time. For this reason deterministic test patterns are transferred from ATE to SOC through test access mechanism (TAM). Deterministic test patterns are available from individual core vendors.

All the test vector compression schemes fall broadly into three categories [83]. These are,

- Code-based techniques
- Linear-decompression based techniques
- Broadcast-scan-based techniques

Next we will review the reported techniques from each category.

Code-based techniques

In code-based techniques, test vectors are encoded. This is done by partitioning the test cubes into smaller symbols and then replacing each symbol with a code word to form the compressed test data. A decoder is then used to decode the

encoded patterns to get back the original symbols.
The code-based techniques can again be classified into different categories depending upon the encoding procedure. In the following, we introduce these categories one by one.

Statistical codes: In this encoding scheme, original data is partitioned into m-bit symbols and each symbol is assigned variable length code words based on their frequency of occurrences.

Statistical coding methods [84, 85] partition the original data into fixed bit-width symbols and assign the variable length code words to them using Huffman coding. Huffman code is an optimal statistical code, but the problem is that the decoder size grows exponentially with the number of symbols. Hence, selective Huffman coding [84] encodes only the most frequently occurring symbols, which is extended further in [86]. In [85] frequently occurring blocks are encoded using variable length indices.

Tehranipour et al. demonstrated a new compression scheme that coded exactly nine code words (9C) [87]. Kavousianos et al. proposed a multilevel Huffman coding based approach [88] where each Huffman codeword corresponds to three different kinds of information. All these methods are code based schemes that partitions the test cubes into different symbols and then encoded them to achieve compression.

Run-Length codes: This coding mechanism takes the opportunity of the occurrence of a particular bit (either 0 or 1) consecutively. Jas et al. [89] proposed a scheme based on runs of 0's using fixed length code words. To increase the runs of 0's, the scheme used a cyclical scan architecture to allow the application of difference vectors. Methods like Golomb coding [90], Frequency Directed Run length coding (FDR) [91], Extended FDR [92], mixed run length and Huffman coding [93, 94], mutation encoding [95], Alternate run length coding using FDR [96], MTC coding [97], Variable length input Huffman coding (VIHC) [7] used the presence of runs of repeated values (0's or 1's) and then encoded into either fixed length or variable length code words. In case of VIHC [7] scheme, symbols are formed by exploiting runs of 0's that produce variable length symbols and then the statistical information are passed to the Huffman algorithm. For these techniques, it requires a synchronization mechanism between tester and the on-chip decoder.

In [98] Ruan and Katti proposed a data-independent pattern run-length (PR) compression scheme for testing embedded cores in SOCs. The scheme applies the well known run-length coding to equal and complementary consecutive patterns of the pre-computed test data set. Ichihara et al. [99] proposed a compression scheme based on JPEG VLC algorithm targeting multimedia SOCs. Fang et al. [100] proposed a compression algorithm based on EFDR [92] called *RunBase-dReordering* (RBR). In this algorithm authors have applied scan chain reordering, scan polarity adjustment and test pattern reordering.

Dictionary codes: In this coding scheme, original data is partitioned into n bit symbols and a dictionary is used to store these symbols. The dictionary index can then be used as the code words for the partitioned symbols. The main problem in dictionary based coding is that dictionary size can become large. Hence area overhead becomes one of the major concerns. A dictionary with fixed length indices has been used in [6]. Dictionary size is constrained and the symbols which are not in the dictionary are transferred directly for decompression. So in this scheme, a partial dictionary is formed. Test data compression techniques based on LZ77 and LZW algorithm, which use dynamic dictionary, have been proposed in [101] and [102].

In [103] authors analyzed the concept of entropy and proposed arithmetic coding. Balakrishnan et al. [104] presented a novel matrix-based software data compression scheme. They also have provided an excellent discussion on the relationship between entropy and test data encoding scheme [105] that calculates maximum compression limit that can be achieved for a type of coding technique.

Linear-decompression based techniques

This technique is based on linear decompressor. Balakrishnan et al. proposed decompression for deterministic vector using linear operations [106]. In [107] authors proposed a compression method using statistical transformation for linear decompressor. The hybrid test approach [108] generates both random and deterministic patterns for the tested chip while it uses an on-chip linear feedback shift register (LFSR) to generate the random patterns. In [109] and [110] a combinational network has been used to compress test cubes. Other methods like reconfigurable interconnection network (RIN) [111], SOC-BIST [112] are also based on linear decompressor. LFSR reseeding [113, 114, 115, 116, 117] has been proposed to achieve test data reduction. An embedded deterministic test (EDT) [118] has been proposed for scan vector decompression based on linear sequential decompressor. Instead of using an LFSR, this work uses a ring generator.

In [119, 120] authors used a combinational decompressor in which a scan chain is driven by XORing some of the tester channels. Krishna et al. [121] proposed a method for improving the encoding efficiency of a combinational linear decompressor by dynamically adjusting the number of scan chains loaded in each clock cycle.

Broadcast-scan-based techniques

In this technique the same value is broadcasted to the multiple scan chains. Though the compression is higher in case of linear decompression based schemes due to the larger output space, faults detected in this technique is more than the linear decompression based method. Main advantage of this technique is that the automatic test pattern generation (ATPG) algorithm can be instructed to produce only encodable test cubes according to the decompresser.
Lee et al. [122] originally proposed broadcast scan scheme in the context of independent circuits i.e. same test set generated by ATPG can be used for different circuits. Again, the scan architecture that drives multiple scan chains from a single channel might result in reduced fault coverage. To overcome this problem authors in [123] proposed the Illinois Scan architecture that operates in two modes: broadcast and serial. The Illinois scan architecture provides a mechanism to reduce application time and data storage requirement [124]. But fault simulation and test generation are necessary as post processing to get high fault coverage in this architecture.

In [120, 125, 126] authors proposed efficient test compression schemes based on broadcast scan architecture that supports multiple tester channels. El-Maleh et al. [127] proposed a compression technique using broadcast scan with relaxation. Given a constraint on the number of tester channels, the technique classifies the test set into acceptable and bottleneck vectors.

All the above techniques can again be classified into two broad categories.

1. **ATPG-independent approach:** In this approach, in the traditional design flow, compression schemes are applied after test patterns have been generated. Mainly code based techniques like 9C [87], FDR [91], VIHC [7], MTC [97] methods belong to this category. A new block merging scheme[128] is also proposed which is ATPG-independent.

2. **ATPG-dependent approach:** In this case, the test compression procedure is incorporated during the stage of test generation. Techniques like linear decompression based methods and broadcast scan based methods fall into this category.

Lin et al. [129] has proposed a compression scheme called multi-layer data copy test data compression scheme. It utilizes a decoding buffer, which supports test loading using previously loaded data. This compression scheme achieves test data compression and test power reduction at the same time. This scheme can be applied to test in an ATPG-independent manner, or can be incorporated into an ATPG to generate highly compressible and power efficient test sets. The authors also proposed an ATPG-dependent tool customized for this approach. In [116], Wang et al. proposed a compression scheme based on LFSR reseeding. The proposed method does not require any special ATPG that is customized specifically for this scheme. Any commercial tool available can be used for that.

2.3.2 Scan power reduction during compression

Application of test vectors generated by an ATPG is very time consuming and thus compacting test vectors are unavoidable. Again excessive switching activity within circuit may cause larger peak and average power dissipation than those during normal operation. Hence power should be taken into consideration at the time of test. The compression/decompression technique may intensively increase the scan-in and scan-out power dissipation of scan chain elements.

To reduce power consumption, several works have been done. Test vector ordering [130, 131, 132, 133] scan-cell ordering [134], gated-clock scheme [135], scan latch partitioning [136] are among the proposed methods.

2.4 Prior works on Low power Testing

Various techniques that are used for test power reduction in a scan based environment have been proposed. They can be categorized in the following way:

Low power ATPG: In this category test vectors are generated in such a way that it reduces the transitions in the circuit under test (CUT). [137] proposes a path oriented, decision making algorithm where don't care bits are intelligently specified so that it reduces the transitions in the CUT between two consecutive test vectors. Wang et al. [138] adapt their approach to the full-scan sequential circuit and the proposed ATPG exploits all possible don't cares that occur during scan shifting, test application and response capture to minimize switching activity in the CUT. In [139, 140] authors also proposed ATPG algorithms that take care of low switching activity during application.

Wu et al. [141] proposed a low power test pattern generation technique which minimizes the peak power consumption associated with the scan and capture operations. The proposed algorithm is based on PODEM [137].

Clocking Scheme: Sankaralingam et al. [142] proposed a technique that uses full-scan circuits with multiple scan chain. This technique depends on generating and ordering of the test set so that some scan chains can have their clocks disabled for portions of the test set. Hence, it reduces the transitions in the combinational part as well as the clock tree. A gated clock scheme is used by Bonhomme et al. [135] that reduces the clock rate on the scan cells during shifting operations without affecting test application time. Thus, it reduces the transition density in the CUT, scan path, and as well as clock tree, feeding the scan path.

Use of the externally controlled gates [143, 144] have shown the reduction of power by sacrificing additional gate delay. It also needs the additional global signal to enable the test points. Hertwig et al. [145] proposed a strategy that modifies the scan cells in such a way that scan chain transitions are completely isolated from the combinational part of the circuit. Here logic is added to hold the output of the scan cells at a constant value during scan shifting, thereby reducing power dissipation. In [146, 147] authors proposed an efficient technique to reduce both dynamic and static power dissipation in scan architectures. In this approach, the scan cell outputs which are not on the critical path(s) are multiplexed to fixed values during scan mode. These constant values and primary inputs are selected such that transitions occurring on non-multiplexed scan cells are suppressed and the leakage current during scan mode is decreased.

Input Control: Assignments of proper primary inputs [148, 149] help to reduce transition propagation from scan chain to the circuit under test. This mainly reduces the transition count in the combinational part of the full-scan circuit during test application.

Test vector ordering and scan cell ordering: Test vector ordering [130, 131, 150] and scan-cell ordering [151, 152] are the alternative techniques for reducing scan power dissipation. In [150] authors presented efficient heuristics for test vector ordering and scan-latch ordering techniques to minimize power dissipation in full-integrated scan circuits by using a graphical method.
Xiang et al. [153] presented a new scan architecture where the scan cells are ordered according to their functional interaction to reduce power. Further, the technique has been enhanced in [154] for routability and fault aliasing. Though the technique produces promising results, its complexity prohibits industrial designs.

Scan chain modification and test vector customization: Scan chain modifications and test stimuli transformation for scan power reduction is presented in [155, 156, 157]. In [155] authors used inverters between scan cells with don't care filling and test vector reordering. In [158], an adaptive scan chain architecture is presented. Here single scan chains are segmented to minimize switching activity during scan shifting. The modified scan chain by inserting XOR gates [156, 157] though reduces scan transitions, area overheads are high as the schemes require non-local connection patterns among the scan cells. In these schemes, number of inputs to the XOR gates is higher as observed in the high area overheads noted in those papers. This coupled with long interconnections to feed the XOR gates can effectively increase the circuit area and delay overheads significantly. This is particularly true for sub-micron design where interconnects act almost as a device on the silicon floor. Sinanoglu et al. [157] proposed a scan architecture modification strategy that transforms the stimuli through logic gate insertion between scan cells, reducing scan transitions. A novel scheme has been presented in [159] by the same authors to reduce test power consumption by freezing scan segments that do not have any care bits in the next test stimuli. By only loading the segments that have care bits, data volume, application time and test power consumption are all reduced at a single go. Only one segment is controlled and observed at a time.

Li et. al [160] proposed a scan power reduction technique through scan chain adjustment to eliminate unnecessary transitions in scan chain. They have proposed an extended weighted transition metric (ETWM) to estimate dynamic power dissipation in CUT caused by the transitions in test stimuli and responses. Kim et al. [161] proposed the total scan power reduction architecture(TOSCA) that results in better fault coverage, as well as reduced switching activity using the transition monitoring window (TMW) [162]. In [163, 164, 165, 166] authors addressed the issue of scan chain design taking into consideration optimization of scan power and routing overhead using scan cell reordering aided by physical layout information.

Al-Yamini et al. [167] proposed a segmented addressable scan test architecture that addresses reduction in test data volume, test application time, test power consumption and tester channel requirements using a hardware overhead of few gates per scan chain. A two-stage scan architecture has also been proposed by Xiang et al. [168] to constrain transition propagation within a small part of scan flip-flops. The first stage includes multiple scan chains, where each scan chain is driven by a primary input. In the second stage, each flip-flop of the multiple scan chain drives a group of flip-flops. There exists no transition at the scan flip-flops in the second stage when a test vector is applied to multiple scan chains. Chen et al. [169] presented a novel DFT technique, *response inverse scan cells* (RISC), to reduce the peak capture power. The outputs of RISC remain unchanged during specified system clocks, so that their power is unchanged.

In [136] authors proposed a multiple scan chain design approach based on the classification of the scan latches as *compatible, incompatible and independent.*

They have also proposed a new test application strategy that applies an extra test vector to primary input during shifting out test responses for each scan chain. This minimizes power dissipation by eliminating the spurious transitions. In [170] test-set independent multiple scan chain design technique based on graph partitioning has been proposed to reduce power. In [171] efficient use of unspecified bit in input test cube and the corresponding response test cube reduces power and test time in a multiple scan chain architecture.

2.5 Conclusion

This chapter has presented a detailed survey of various state-of-the-art testing strategies for reducing test cost. Though several techniques have been proposed to resolve the testing challenges, with the increasing complexity, new testing strategies are needed to be devised to tackle the emerging problems. From next chapter onwards different test resource optimization strategies developed in this research work have been presented. Chapter 3 starts with the test scheduling strategy for the core based SOCs targeting test time optimization.

Chapter 3

Non-hierarchical SOC Test Scheduling

3.1 Introduction

Integrating reusable cores into an SOC involves complicated design and test issues. One of the basic objectives of SOC testing is to reduce the test application time. Scheduling test solutions for embedded intellectual property (IP) cores can be utilized for effective minimization of test time. Hence, test scheduling problem is one of the major issues in test integration. Though reuse philosophy of IP cores reduces the design cycle for SOCs, rapid increase in transistor and pin count ratio creates a major bottleneck for the manufacturing test of SOCs [2]. Hence, modular test based approach has come up where test re-usability has been utilized. An embedded core needs to be isolated during test using a core wrapper. A test access mechanism (TAM) [2] is used to transport test data from/to test source/sink. A general problem for SOC test integration consists of the design of test access mechanism (TAM) architecture that transports test data between SOC pin and core wrapper. Wrapper provides an interface between TAM and the core and can be operated in several modes [45]. Chakrabarty [56] showed that the TAM design problem is NP-hard. Many TAM design algorithms have been proposed for optimizing the test application time. Test scheduling is a process that determines the start and finish time for testing each core in the SOC such that the overall test application time is minimized for a given TAM architecture. 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.

The chapter is organized as follows. Section 3.2 discusses the contribution we made in this work on test scheduling along with the corresponding test architecture. Section 3.3 describes the overall SOC test scheduling problem. Section 3.4 discusses the wrapper design optimization method used. Section 3.5 discusses the test scheduling algorithm with the proposed bin-packing approach. The GA formulation for selection of one rectangle from the set of rectangles of each core is presented in Section 3.6. We present our experimental results based on the ITC'02 benchmark [172] SOCs in Section 3.7. Section 3.8 concludes the chapter.

3.2 Summary of Contribution

The primary objective of this work is to achieve minimal test time while satisfying two constraints: 1) given number of SOC pins and 2) allowable SOC peak power consumption. We represent wrapper design as a rectangle with width being the number of TAM channels required and height being the test time. To get the complete test scheduling we have used the two-dimensional bin-packing approach based on a best-fit heuristic. It uses the sequence of rectangles obtained from GA and places one rectangle after another, each time optimizing the placement based on the local scenario reached by the placement of all the previous rectangles. For a particular rectangle, the placement algorithm evaluates the possible placement time instants on the basis of two cost parameters, TAM width utilization (U) and increase in total test time (T). Experimental results show that the proposed method obtains better test time results for the SOCs with larger number of cores than the recently proposed works.

3.3 Problem Formulation

A core-based system is said to be testable if all the cores present in the SOC is equipped with a test method and corresponding test sets. A testable unit can be a core, user defined logic (UDL) or interconnections. In our work, we consider only the core testing. We assume that a test set for a testable unit is stored in an Automated Test Equipment (ATE). Hence TAM wires are required for the transportation of test data from ATE to the testable unit. Again, at any point of time, only one testable unit can use a TAM wire, as illustrated in Fig. 3.1. The figure shows assignment of TAM wires to four test sets over time. For a certain period of time each test is assigned to some TAM wires.

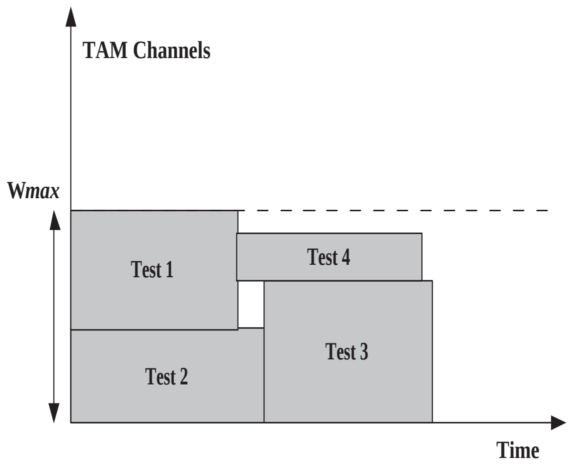


Figure 3.1: TAM wire constrained test scheduling

Hence, the problem that we mainly 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. But the assignment should consider the constraints as shown in Fig. 3.1, where at any point of time, a particular TAM wire can be used by only a single testable unit.

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 our 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.

3.4 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. There are two kinds of wrapper designs, balanced and unbalanced [45].

- For cores having no internal scan chains (that is, containing input and output pins only), unbalanced wrapper design is preferred since it can obtain a lower test time than the balanced one [47].
- To design the wrapper for cores with internal scan chains, we have used the *Design_wrapper* algorithm proposed in [72]. It produces balanced wrappers.

To calculate test time T, for a wrapper we have used the well-known formula [45] given below.

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

Where P is the number of test patterns and $S_i(S_o)$ denotes the length of longest wrapper scan chain used during scan-in(out) for a core. 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 two 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 can be considered in the test scheduling. It is based on pareto-optimal design [72] 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.

3.5 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 R_i wrapper configurations. 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 obtain 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 width w_{ij} and height $T(w_{ij})$. We want to choose one rectangle from each set of rectangles R_i and pack all the rectangles in the bin, so that height of the bin is minimized.

In this work, we treat this test 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. It may be noted that a cycle-accurate power estimation model has been proposed in [173]. For the sake of simplicity, we continue with the peak-power model as proposed in [66], though the scheme can always be extended to incorporate the new power model proposed. 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 [64] and given in Table 3.1.

Test	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	46
27		2512	3135
28		2921	159
29			6756
30			77
31			218
32			396

Table 3.1: Power consumption values for the tests in design d695, p22810 and p93791

The test schedule of an SOC is feasible if no two cores are assigned to the same SOC pin at the same time instant and for each core all its wrapper pins are assigned to SOC pins for the entire time needed to test that core. Therefore, in ?

the process of bin-packing, no rectangle should be overlapped. It should be noted that the total number of SOC pins should be sufficient to feed test data to all the cores being tested at any point of time. Since, at a particular time instant, the number of test lines needed is equal to the sum of widths of the rectangles, we have the following constraint.
Constraint: At any instant within the entire test time of the SOC, the sum of the widths of all the rectangles that are scheduled at that point should be equal to or smaller than the total TAM width provided to test the given SOC.
To satisfy this constraint over the entire test time, the constraint has to be satisfied at the test start time t_{si} of each core i. If this is fulfilled at every core test start time t_{si}, it will also hold for all the time instances from 0 to T_{end}, where T_{end} is the time needed to complete testing of the given SOC. The following section discusses the proposed rectangle packing algorithm.

3.5.1 Placement Algorithm

As mentioned earlier, we have used a best-fit heuristic based placement or packing algorithm satisfying the above mentioned constraints (with or without power constraint) for placing the rectangles in a bin of width W (maximum allowable TAM width) whose height is infinite (∞). The algorithm tries to place one rectangle after another each time optimizing the placement based on a cost function. The cost is rectangle specific and is calculated considering the existing profile of the already placed rectangles. The profile provides the information regarding TAM channels used in various time instants. Hence, cost function depends on two parameters, TAM width utilization (U) and increase in test time (T). Utilization at a particular instant is the number of TAM wires that are being used at that instant. We use weighted sum of these two parameters to determine the final cost of a particular placement. This is

$$Cost = wt \times u(U) + (1 - wt) \times t(T), (0 \leq wt \leq 1) \tag{3.2}$$

Where wt is the weight in favor of the utilization parameter and $u(U) = U/U_{max}$ and $t(T) = T/T_{max}$. The utilization parameter is a measure of utilization of TAM lines during the entire height of the rectangle being placed. It is maximized by minimizing the number of unused TAM lines. The variable U represents the sum of unutilized TAM lines at each time instant during the testing time of the core, that is, the height of the rectangle. Suppose that the rectangle is placed between the time instants t_1 and $t_1 + h$. In this interval, there will be subintervals during which the number of unutilized lines does not change. The changes occur only at the boundaries of subintervals. For instance, may be the TAM lines used in the interval between t_1 and $t_1 + \triangle$ be w_1. If W be the total TAM width used, the total number of unutilized TAM lines for this interval is $(W - w_1) \times \triangle$. For each such interval, the number of unutilized TAM lines is calculated and summed up to get the total unutilized TAM lines U. Once all such U values have been calculated, for different possible placements, U_{max} is taken to be the maximum of U values among all these placements. Thus, $u(U) = U/U_{max}$ gives a normalized value of unutilized TAM lines. The second parameter, increase in test time (T) is an obvious attempt to attach some penalty with those possible placements that result in putting the upper edge of the rectangle in consideration, beyond the test time required by all the rectangles placed before it. Inclusion of such a parameter can be justified on the basis of the simple fact that it is the testing time, which we are trying to minimize. The value of $wt = 0.8$ has been found to work well with the ITC'02 benchmark [172] SOCs.

For a particular rectangle, the algorithm evaluates the possible placement points (time instants) on the basis of cost parameters and the point with lowest cost value is selected for placing the rectangle. Now, the important question that we face is that of determining which of the innumerably abundant possible placement points to consider in order to evaluate their relative goodness. This aspect is addressed in Theorem 3.1. The theorem considers the occupancy function $W(t)$ which can be defined as the number of TAM lines used by the already placed cores at time instant t. While implementing the placement algorithm, we successively update the value of the occupancy function.

Theorem 3.1. *For any given configuration of placed rectangles, there will always be an optimum placement (with respect to the cost function mentioned above) which has at least one of the horizontal edges of the rectangle collinear with the points where the occupancy function changes value.*

Proof. Let us assume to the contrary that, there is no such optimal placement. Then, we consider one of the already existing optimum points. As far as the utilization parameter is concerned, it is clear that if we move the rectangles upwards, we are surely going to reduce the area formed by the free TAM wires. This will continue till we reach an upward edge of the occupancy function. Another such possible optimal point - that has a gradient pulling the rectangle down of its present placement. These cases are shown in Fig. 3.2.
We can do these upward and downward movements as long as we do not cross the testing time bound, set by the already placed rectangles. When this appears, we are already at a point where the occupancy function changes value, namely the last time instant at which $W(t)$ is non-zero. Beyond which, present $W(t)$ is always zero, as all the already placed cores have finished their testing by that time.
Hence it implies that there can be no optimal placement, except at the points which have at least one of the horizontal edges of the rectangle collinear with the points where the occupancy function changes value. □

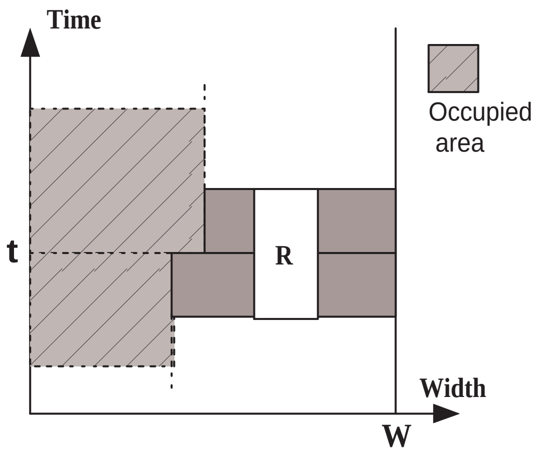


Figure 3.2: An optimum placement of the incoming rectangle

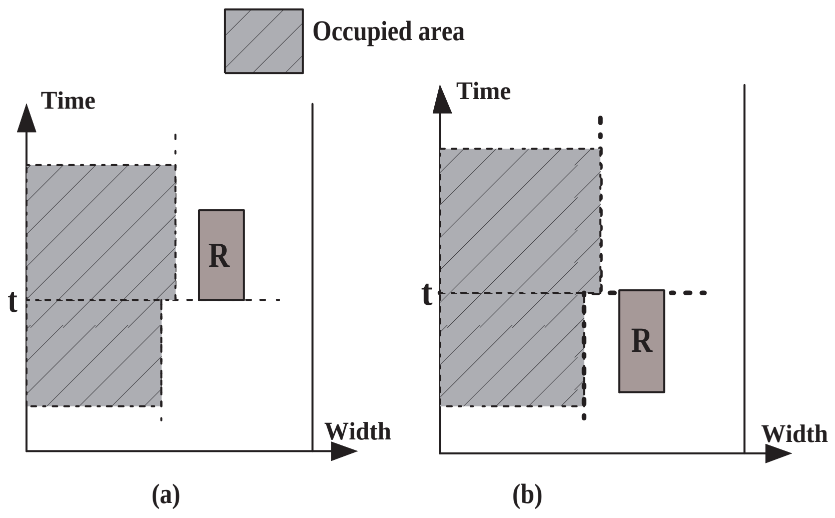


Figure 3.3: (a) Upward and (b) downward oriented placements

While implementing the algorithm, we only store the values of the occupancy function at the points at which it changes value. In other words, we only store the tuples $< t, W(t) >$. When a rectangle is to be placed, we consider the placements aligning either of the horizontal edges of the rectangle with these stored values of t. For each possible placement point (time instant) we consider two types of orientations - up and down, which denotes whether the rectangle is placed upwards (or downwards). These two possible placements are shown in Fig. 3.3.

The above theorem (Theorem 3.1) will now help us in selecting the placement points to evaluate before finding the optimum one. For each of the possible

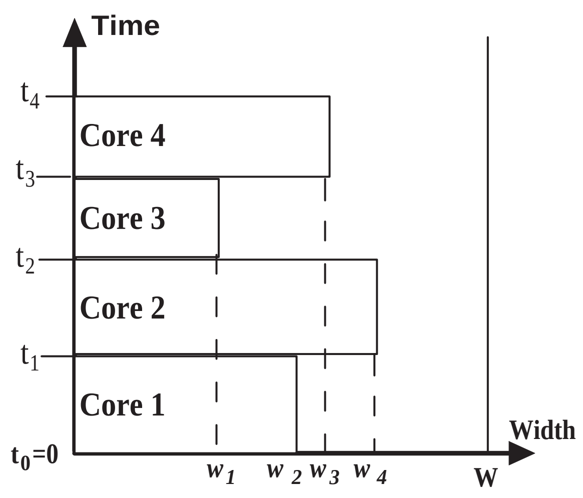


Figure 3.4: A possible profile of the time-width plot

placement points we check whether the constraint (with or without power constraint) is satisfied or not and the cost is evaluated. The point with lowest cost is selected for placement of the rectangle. This process is continued for all the cores and the last time instant provides the final test time for the selected sequence of rectangles obtained from GA.

Data Structure used:

- The sequence of representative rectangles for each core generated by GA is in the form of a queue L_{CORE}.
- The current profile of the time-width plot (for previously placed rectangles) is maintained as a doubly-linked list, *time_instants*, which stores the value of the width at the particular time instant where it changes. Each node in this list is a tuple like $< time\_instant(t), width(w) >$ and nodes are sorted by increasing time_instant values. Two consecutive nodes denote an interval between which the width is fixed by the w-value stored in the node that is ahead in the list. That is, if the i^{th} and $(i+1)^{th}$ tuples are $< t_i, w_i >$ and $< t_{i+1}, w_{i+1} >$, the width in the interval $[t_i, t_{i+1}]$ is w_i and beyond t_{i+1}, width is w_{i+1}. Corresponding to the example shown in Fig 3.3, the list would be $< 0, w_2 >, < t_1, w_4 >, < t_2, w_1 >, < t_3, w_3 >, < t_4, 0 >$. Note that $[t_i, t_{i+1})$ is a closed interval at left side.
- We also use a list to store the locations and orientation of the placed rectangles.

The placement procedure without power constraint is represented in following algorithm.

Algorithm 3.1: Rectangle Placement Algorithm

```
Input : $L_{CORE}$: a list of rectangles with one rectangle corresponding to
        each core.
Output: Final schedule after placing all the cores in the list $L_{CORE}$
        satisfying constraints.
1  begin
2      Pick first rectangle $C_0$ from $L_{CORE}$ and place at 0-th time instant with
       'up' orientation;
3      $Time\_instants = < 0, C_{0w} >, < C_{0h}, 0 >;$
       /* $C_{0w}$ and $C_{0h}$ are the width and height of the rectangle
          $C_0$ respectively                                              */
4      while ($L_{CORE}$ is NOT empty) do
5          a)Pick rectangle $C_i$ from $L_{CORE}$ in the order of the list;
6          b)Find possible time instants $t_k$'s satisfying constraints for both
           the orientations and calculate cost for each of these possible time
           instants according to Eqn. (3.2);
7          c)Place the rectangle $C_i$ at time instant $t_k$ giving minimum cost (in
           case of ties, the placement with the lowest $t_k$ value and at such
           level a 'down' orientation is preferred);
8          d)Update time_instants list;
9      end
10 end
```

To impose power constraint simple modifications in the data structure and Algorithm 3.1 are sufficient. In this case each node in the *time_instants* list is a triplet like $< time\_instant(t), width(w), power\_value(POW_t) >$ and nodes are sorted by increasing *time_instant* values. Two consecutive nodes denote an interval between which the width and the total peak power consumption by the cores is fixed. During calculation of possible time instants, in this case, not only TAM width constraint is to be met, but also the maximum allowable power budget POW has to be satisfied.

3.6 GA Formulation

In this section, Genetic Algorithm (GA) has been used to get the sequence of rectangles to be placed. The placement is performed using Algorithm 3.1. For each core in the SOC we have to select one rectangle from the set of rectangles (wrapper configurations) generated for that core so that total testing time will be less. The order in which we select the rectangle for each core depends on the average area of the rectangles obtained from wrapper design algorithm used. For this purpose, cores are sorted in terms of decreasing average rectangle area. Throughout the execution of the algorithm, this order is maintained. Next we define the individual representation of chromosome, mutation and crossover operators, fitness function and the selection procedure.

3.6.1 Chromosome structure and fitness

A chromosome has been taken to be an array of N floating point values, N being the number of cores present in an SOC. Floating point values f_i are taken within a range of 0 to 1. Each f_i is used to select one rectangle from the set of rectangles obtained for the i^{th} core using wrapper design process. The value f_i is multiplied by the number of rectangles for core i to identify the rectangle selected for it. For each chromosome, fitness value is calculated using the best-fit heuristic based placement algorithm discussed in Section 3.5. This fitness value is calculated after getting sequence of representative rectangles from each core.
Example:
Consider an SOC A with 10 cores. The order in which the cores are considered for testing is the decreasing average rectangle area obtained from wrapper design. Let us assume that for maximum available TAM width of 16, number of wrapper configurations for each core is obtained as $4:14, 2:12, 10:8, ..., 3:15$. Here $4:14$ represents that for core number 4, total number of wrapper configurations or rectangles obtained is 14. The chromosome structure for this SOC A is shown in Fig. 3.5. The chromosome contains 10 floating-point values generated randomly. Each floating-point value is used to select the representative rectangle for a core. For selection of the representative rectangle of a core, all the rectangles obtained for that core from wrapper design are sorted in terms of decreasing average area. Now, consider the core number 2 which has 12 rectangles obtained from wrapper design process. Hence the representative rectangle which will be used for scheduling is $0.521 \times 12 = 6^{th}$. Similarly for other cores also the representative rectangles are obtained.

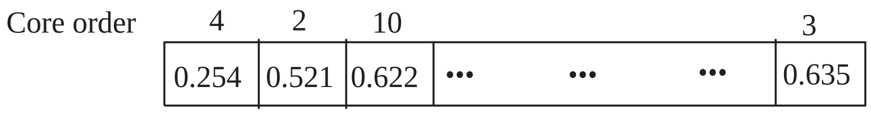


Figure 3.5: Chromosome Structure

3.6.2 Genetic Operators

To move towards the promising region of search space, different evolutionary operators are used. To create population for the new generation, 20% best fit chromosomes are directly copied and remaining 80% are created using crossover and mutation operators. In our GA approach we have used parameterized mutation and crossover operation [174].

Crossover operation: The crossover operation that we have considered is multi-point crossover operation between two chromosomes. For each of the index of the chromosomes a random number between 0 and 1 has been generated. If randomly generated number is ≥ 0.7 then the values of the corresponding indices are interchanged.

Mutation operation: For mutation operation also, multi-point mutation has been considered. For each index of the chromosome a random value between 0 and 1 has been generated. If the randomly generated number is > 0.7, the value f_i at that point is changed to $(1 - f_i)$.

The algorithm is run for 150-200 generations with $4000 - 5000$ population sizes depending on the number of cores present in the SOC.

3.7 Experimental Results

In this section we present our results of experimentation with the ITC'02 SOC benchmarks [172].

Table 3.2 shows the comparison of our test time results with recently proposed methods. Numbers in bold represent the best result for each case. Row "Our" indicates our method. The results reported for the proposed method are the best among five runs of the procedure. For SOCs with larger number of cores our proposed method provides better results for most of the cases with different TAM widths. The maximum CPU time required to run the algorithm for SOC p93791 containing 32 cores with 3000 population size is approximately 1812.25 sec for $W = 64$ in a 2.66GHz Pentium IV machine.

Table 3.3 summarizes the comparison of our results with the recently proposed test scheduling approaches according to Table 3.2. It is showing for how many cases (taking each TAM width value as one case for each core) our results are better, equal or not better than the recently proposed schemes. The comparison has also been depicted in Fig. 3.6. It is seen from Fig. 3.6 (and Table 3.3) that our proposed approach is better than ACO[77] for 60% of the cases (19 out of 28), more than 50% of the cases compared to 2-Stage GA[78], more than 20% of the cases compared to B*-SA[76], more than 40% of the cases for EA(c)[62], more than 60% of the cases for EA(nC)[62] and almost 50% of the cases compared to

ITC'02 Benchmark Circuits	Solution Methods	Number of TAMwires						
		16	24	32	40	48	56	64
		Test Time (Clk Cyc.)	Test Time (Clk Cyc.)	Test Time (Clk Cyc.)	Test Time (Clk Cyc.)	Test Time (Clk Cyc.)	Test Time (Clk Cyc.)	Test Time (Clk Cyc.)
d695 (10 cores)	LB_T **[62]**	40951	27305	20482	16388	13659	11709	10247
	Our	41604	27767	20957	16913	14273	12084	10723
		(-1.59)	(-1.69)	(-2.31)	(-3.20)	(-4.49)	(-3.20)	(-4.64)
	ACO[77]	41737	28080	21098	17075	14310	12110	10783
	2stageGA[78]	40691	28060	20977	16894	14129	11453	10573
	B*-SA[76]	**39489**	**26203**	**19773**	**16149**	**13649**	**11285**	**9885**
	EA(C)[62]	41553	27982	21014	16908	14240	11988	10571
	EA(nC)[62]	41809	27989	21142	17015	14236	12134	10788
	SA[47]	41604	28064	21161	16993	14183	12085	
p22810 (28 cores)	LB_T **[62]**	419466	279644	209734	167787	139823	119848	104868
	Our	428852	**286352**	**216570**	175946	**147898**	127071	112498
		(-2.24)	(-2.39)	(-3.25)	(-4.86)	(-5.77)	(-6.02)	(-7.27)
	ACO[77]	**424889**	289190	218035	177214	**147898**	130479	115791
	2stageGA[78]	438619	288565	216747	177633	148832	**123857**	**103321**
	B*-SA[76]	438619	287999	216747	178223	149592	129624	115406
	EA(C)[62]	438783	292824	226545	167792	153260	133094	117638
	EA(nC)[62]	438619	289237	228732	183433	153525	130949	116625
	SA[47]	438619	289287	218855	175946	147944	126947	109591
p93791 (32 cores)	LB_T **[62]**	1746657	1164442	873334	698670	582227	499053	436673
	Our	1750830	1170620	**877073**	704272	**587117**	505586	**442455**
		(-0.24)	(-0.53)	(-0.43)	(-0.80)	(-0.84)	(-1.30)	(-1.32)
	Reconfig[79]	1752336	1174252	877977	**703219**	592214	511925	442478
	ACO[77]	**1747504**	1175988	891103	716112	598286	517692	452951
	B*-SA[76]	1782067	1190565	890092	707664	609580	517017	452245
	EA(C) [62]	1754980	1171190	886038	706820	600986	**501057**	445748
	EA(nC)[62]	1754980	1184630	900388	724758	611029	520868	458389
	SA[47]	1757452	**1169945**	878493	718005	594575	509041	447974
p34392 (19 cores)	LB_T **[62]**	932790	621903	**544579**	**544579**	**544579**	**544579**	**544579**
	Our	939855	**626122**	**544579**	**544579**	**544579**	**544579**	**544579**
		(-0.76)	(-0.68)	(-0.0)	(-0.0)	(-0.0)	(-0.0)	(-0.0)
	ACO[77]	**931588**	631035	**544579**	**544579**	**544579**	**544579**	**544579**
	B*-SA[76]	935649	635237	**544579**	**544579**	**544579**	**544579**	**544579**
	EA(C) [62]	939855	641514	**544579**	**544579**	**544579**	**544579**	**544579**
	EA(nC)[62]	939855	637263	**544579**	**544579**	**544579**	**544579**	**544579**
	SA[47]	944768	628602	**544579**	**544579**	**544579**	**544579**	**544579**
g1023 (14 Cores)	**Our**	30755	20498	15843	**14794**	**14794**	**14794**	**14794**
	B*-SA[76]	**29765**	**20032**	**14913**	**14794**	**14794**	**14794**	**14794**
f2126 (14 cores)	**Our**	357088	**335334**	**335334**	**335334**	**335334**	**335334**	**335334**
	B*-SA[76]	**350030**	**335334**	**335334**	**335334**	**335334**	**335334**	**335334**
t512505 (31 Cores)	**Our**	10530995	**10453470**	5268868	**5228420**	**5228420**	**5228420**	**5228420**
	B*-SA[76]	**10504020**	**10453470**	**5228420**	**5228420**	**5228420**	**5228420**	**5228420**

Table 3.2: Comparison of test scheduling times for different ITC'02 SOC benchmarks

SA[47].

Test time results obtained in our method are also compared with the theoretical lower bound test time results (LB_T) as reported in [70]. It has been shown in the rows marked LB_T in Table 3.2. Deviation from the lower bound test times are shown just below the test time results obtained in our method. It is seen that for most of the cases, our test time results are within 3% of the lower bound results reported in [70]. Only for a few cases (p22810 with higher TAM width configurations) test time results are about 7% higher than lower bound results. On an average, our results are within 2.14% of the lower bound results.

To get power constrained test scheduling results using ITC'02 benchmark

Better than	Our	Reconfig	ACO	2-Stage GA	B*-SA	EA (C)	EA (nC)	SA
Our	–	6	19	8	15	15	21	16
Reconfig	1	–	6	NA	7	5	7	5
ACO	3	1	–	3	8	11	20	9
2-Stage GA	6	NA	11	–	4	9	12	10
B*-SA	14	0	15	8	–	15	20	11
EA(C)	7	2	13	3	8	–	15	12
EA(nC)	1	0	3	1	3	6	–	5
SA	4	2	14	3	10	11	17	–

Table 3.3: Comparison of number of cases with different test scheduling schemes

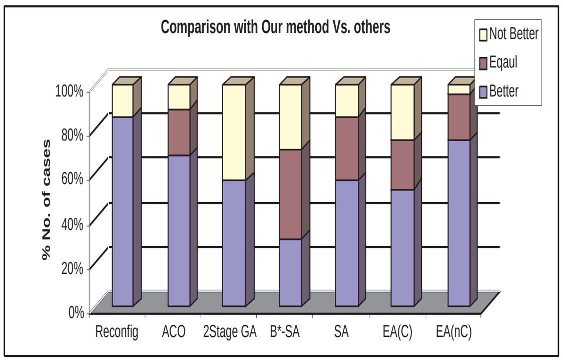


Figure 3.6: Comparison of number of cases with our techniques Vs. recently proposed techniques

circuits, we have made the assumption that the core power values given in the benchmarks are the peak power values during test. Only one benchmark (h953) among the ITC'02 benchmark set has power dissipation numbers included. For other three SOCs d695, p22810 and p93791, we have used the power consumption values reported in Pouget et al [64] and reproduced in Table 3.1. Table 3.4 represents the results for power constrained test scheduling times for SOC h953. To compare with [62] we have used the same 3 power limits for the SOC power budget, 6×10^9,7×10^9 and 8×10^9, chosen above the maximum power consumed by one of the core 5.75×10^9. It is seen that our proposed method provides better results than [62].

Tables 3.5,3.6 and 3.7 represent the test time results for SOCs with higher number of cores. SOCs d695, p22810 and p93791 are containing 10, 28 and 32 cores respectively. The power limitations for d695 are in the range from 1500 to 2500, while for p22810, power limits are in the range of 3000 to 6000 and for p93791 the range is from 10000 to 35000. It is seen from the table that as the

Power Limit	Number of TAM Wires	
(POW)	[47]	Proposed Method
5753800192	122636	119357
6×10^9	122636	119357
7×10^9	119357	119357
8×10^9	119357	119357

Table 3.4: Test time results under power constraints for h953

Power Limit	Number of TAM Wires						
(POW)	16	24	32	40	48	56	64
1500	42189	30054	23297	18883	17083	15670	17695
1800	41925	28491	21724	17174	16320	13692	14674
2000	41804	27999	21592	17067	15652	12987	14434
2500	41718	27999	21236	16965	14434	12862	11646

Table 3.5: Test time results under power constraints for d695

power limit is increasing, the test times are meeting with the unconstrained test time results shown in Table 3.2.

Power Limit	Number of TAM Wires						
(POW)	16	24	32	40	48	56	64
3000	431475	290113	222095	181489	156041	138346	120414
4000	430869	287136	217980	175946	148690	127382	116625
4500	429792	286545	217771	175946	147898	127382	116625
5000	428852	286352	217083	175946	147898	127382	116625
6000	428852	286352	216185	175946	147898	127382	113400

Table 3.6: Test time results under power constraints for p22810

Power Limit	Number of TAM Wires						
(POW)	16	24	32	40	48	56	64
10000	1770954	1192015	1033988	841594	609751	573720	552410
20000	1759656	1174517	886869	712053	600632	508947	450977
25000	1756326	1173939	883885	708150	599339	508437	449657
30000	1752935	1171446	883641	706741	589603	506884	445501
35000	1750830	1171446	878583	705941	587103	505586	443362

Table 3.7: Test time results under power constraints for p93791

3.8 Conclusion

In this chapter, we have proposed SOC test scheduling algorithm with power constraints based on 2-dimensional rectangle bin packing approach considering best-fit heuristic. A genetic algorithm based scheme has been utilized for selection of representative test rectangle for each core present in the SOC. The test scheduling results are obtained for ITC'02 benchmark circuits and it is seen that our proposed method provides better results than the recently proposed techniques. In the next chapter, we present a scheduling technique for dual-speed TAMs.

Chapter 4

Test Scheduling using Dual Speed TAM with Power Constraints

4.1 Introduction

In the last chapter we have seen a power-constrained test scheduling algorithm that assumes all cores to be operating at same frequency. However, in a more realistic situation, different cores of an SOC are expected to operate at different scan clock frequencies. Hence to test the cores with higher scan frequency, test data need to be transported at higher data rate along some selected tester channels of the ATE. To provide such flexibility, ATE vendors manufactured new class of tester devices that can drive simultaneously different tester channels at different data rates [175]. Example of such ATE devices are like Agilent 93000 series tester based on port scalability and test-processor-per-pin architecture [176], and the Tiger system from Teradyne [177]. In case of Teradyne technology based testers, test data rate can be increased for selected pin groups to match the SOC requirement. But, due to different constraints like ATE resource limitations, SOC power ratings, limitations of scan clock frequencies etc., it is needed to efficiently use the ATE channels at higher data rates such that test time can be reduced, which in turn reduces the test cost.

The test methodology follows modular design process where embedded cores are isolated from the surrounding logic using test wrapper and test access mechanism (TAM). This modular testing also supports the use of port scalable testers for dual-speed testing of SOCs. For port scalable testers, the group of TAM wires connected to a particular port can be configured to operate at the same scan data rate. Hence, depending on the availability of high-speed and low-speed TAM wires, ports can be configured accordingly to operate at on-demand frequency.

In this work we concentrate on the problem of designing an optimized dual-speed TAM architecture using the port scalability feature of the ATE. The main contributions of this work are the following:

1. We describe a Genetic Algorithm (GA) based heuristic approach for TAM width partitioning, that provides test solution better than rectangle packing approach as presented in [4]. With a partitioned approach, the test controller also becomes simpler.
2. Lower bound in test time calculation has been done.
3. Studied the effect of power constraint on the overall test time minimization.

The rest of the chapter is organized as follows. Section 4.2 defines the test schedule optimization problem for ATEs with dual-speed channels. Section 4.3 determines the lower bound of the proposed solution. Section 4.4 describes the GA method used for solving the concerned problem and the impact of dual-speed TAM on test power is discussed in Section 4.5. Section 4.6 describes the experimental results obtained and Section 4.7 draws the conclusion.

4.2 Optimization with Dual-Speed ATE channels

This section describes the main techniques employed in our TAM optimization algorithm. The general integrated wrapper/TAM co-optimization and test scheduling problem that we address is as in [4].
P_{dual_speed}: *Given the test data parameters for the embedded cores, total SOC-level TAM width W, a total of V available high-speed ATE channels ($V < W$), and the ratio f of the high-speed data transfer rate to the low-speed data transfer rate, determine the wrapper design, TAM width, and test data rate for each core, and the SOC test schedule such that a) the total number of TAM wires utilized at any moment does not exceed W, b) the number of TAM wires driven at the high data rate does not exceed V, and c) the SOC testing time is minimized.*

The test set parameters for each core include the number of primary inputs, primary outputs, bidirectional I/Os, test patterns, scan chains, and scan chain lengths. In this work, we assume the cores to be hard cores. For a given TAM width and wrapper design for a core, it is assumed that its testing time at the high data rate is f times lesser than its testing time at the low data rate. In other words, if it takes $T_{ih}(w_i)$ seconds to test core i at the high data rate with a TAM width w_i, the time taken to test it at the low data rate is $T_{il}(w_i)$ seconds, where $T_{il}(w_i) = f \times T_{ih}(w_i)$. It is also assumed that a core cannot be connected to both high and low data rate lines at a time. The optimization procedure described here is needed to determine an efficient TAM architecture for given values of f and V.

The *Design_wrapper* algorithm from [72] has been used to design a wrapper and determine the testing time for a core for several possible TAM widths. These testing times include the cases of both the high data rate and low data rate channels. These pre-calculated testing times are subsequently used in the TAM optimization procedure. We formulate the dual-speed TAM optimization problem as follows:
Given two sets of TAM widths (with total TAM width W), one representing the high data rate TAM lines (V) and the remaining representing the low data rate lines ($W - V$), partition both the high data rate and low data rate TAMs and assign each core to the appropriate partition such that,

1. the maximum TAM width is not exceeded at any time.
2. each core is assigned to either a high speed or a low speed TAM partition
3. total test item is minimized and
4. power constraint is honored.

It is to be noted that a core vendor can specify an upper limit on the scan test frequency for a core. Hence, if this upper limit is lower than the higher data rate of ATE channel then the core can be tested only at lower data rate. Otherwise it can be tested at higher data rate also.

4.3 Lower bound on test time

To derive the lower bound (LB) on the overall system test time, we have borrowed the concept from rectangle packing approach. We assume that there can be an overlap of the scan-out operation for the last test pattern of a core with the scan-in operation for the first pattern of the next core on the same TAM wire. This is possible since the TAM wires directly feed the wrapper scan chains and get output from the scan chain only. Thus, connecting to a bidirectional ATE channel is possible.

From wrapper design algorithm, for a range of TAM widths the testing times of a core can be obtained as a set of rectangles. Let for each core i $(1 \leq i \leq N)$ of an SOC, a set R_i of rectangles be obtained. Now according to the optimization

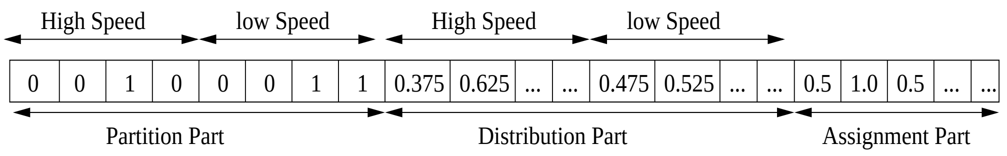


Figure 4.1: Chromosome structure

algorithm one rectangle from each R_i $(1 \leq i \leq N)$ has to be selected and packed into a bin of fixed height W (maximum available TAM width) such that overall testing time can be minimized and no two rectangles overlap.

If overall testing time is T, then the area of the bin occupied by the rectangles is $T \times W$.

Now, if a core i is being tested at TAM width w, then the area of the rectangle is given by $R_i(w) = T_i(w) \times w$, where $T_i(w)$ is the testing time of the core i. Let $R_i^{min} \epsilon R_i$ be the minimum area rectangle for a core i, that is, $R_i^{min} = min_i R_i(w)$, $1 \leq w \leq W$. Now if this core is tested with the frequency f then the test time of the core will be $T_i(w, f) = T_i(w)/f$. Hence area occupied by this core is given by $R_i(w, f) = T_i(w, f) \times w$, where $0 \leq f \leq f_i^{max}$. f_i^{max} is the maximum frequency of core i.

It was shown in [81] that $R_i^{min} = R_i(1, f_i^{max})$. Hence, lower bound $LB_1 = \sum_{i=1}^{N} R_i(1, f_i^{max})/W$. This lower bound is more accurate for smaller W values. For higher W values we obtain another lower bound from [178]. This is obtained as $LB_2 = max_j\{min_i T_i(w, f_i^{max})\}$, $1 \leq i \leq N$ and $1 \leq j \leq N_B$, where N_B is the number of TAM partitions. Hence overall lower bound $LB_T = max\{LB_1, LB_2\}$.

4.4 Test Scheduling using Genetic Algorithm

In this section, we enumerate the Genetic Algorithm (GA) based formulation for SOC test scheduling on dual-speed TAMs.

4.4.1 Solution Representation

A chromosome for the given problem conceptually consists of three parts. The first two parts basically consist of again two parts each with same kind of information with different sizes. Hence total number of parts in the chromosome is five as shown in Fig. 4.1.

First part, named as *partition part*, is the number of TAM partitions. The sub-part 'High Speed ' of it is the number of partitions for the high-speed TAM of width V. Since the total TAM width V for high-speed can be encoded with a binary string of $(log_2 V+1)$ bits, this partition part is a bit-array of size $(log_2 V+1)$. Second sub-part is the number of TAM partitions for the low-speed TAM channels of width $(W-V)$, where W is the maximum allowable TAM width (including high-speed and low-speed) for an SOC. Since the total low-speed TAM width $(W-V)$ can be encoded with a binary string of $(log_2(W-V)+1)$ bits, this part is an array of $(log_2(W-V)+1)$ bits.

The *distribution part* gives the widths of each of the partitions of the high-speed and low-speed TAMs. It consists of two components - 'High Speed' and 'Low Speed'. The 'High Speed' part is an array of real numbers between 0 and 1, where the j^{th} entry of this array multiplied by the high-speed TAM width (V) represents the width of the bus j. The array size is equal to the decimal value of the first part. However, to keep the chromosome length fixed, we have used W entries here, only the first few are actually used (depending upon the value in partition part). Similarly, the fourth part (also called the distribution part for low-speed TAM), gives the partitions of the low-speed TAM of width $(W-V)$. This is also an array of real numbers between 0 and 1, where the j^{th} entry of this array multiplied by the TAM width $(W-V)$ represents the width of low speed bus j. The array size is equal to the decimal value of the second part.

Fifth part is the *assignment part*. This is also an array of real numbers between 0 and 1, where the j^{th} entry of the array multiplied by the decimal value of the first part represents the bus assigned to the j^{th} core. The array size is equal to the total number of cores in the SOC. These five parts of the chromosome form a solution to the given problem.

Example:

For the example in Fig. 4.1, number of bits in partition part of high speed TAM is 4 and the number of partitions is 2 (0010). So, in the distribution part of high speed TAM, first two entries will be considered and the respective TAM widths are $(0.375 \times V)$ and $(0.625 \times V)$. Similarly, number of bits in the partition part for low speed TAM is 4 and the number of partitions is 3 (0011). So, in the distribution part of low speed TAM, first three entries will be considered. Fifth part is assignment part and size of this part will be equal to the number of cores in the SOC. As the total number of partitions, that is, summation of partition part of high speed and low speed TAM is 5 and first value in the assignment part is 0.5, hence TAM number 1 $(0.5 \times 2 = 1)$ of width $0.375 \times V$ is assigned to core 1 and so on.

4.4.2 Genetic Operators

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

Crossover

Our GA formulation has been biased towards selecting the chromosomes with better fitness to participate in crossover. For this purpose, the whole population is sorted according to their fitness values. A certain percentage of population with better fitness value is defined to be the "Best Class". To select a chromosome participating in crossover, first a uniform random number between 0 and 1 is generated. If the number is greater than 0.5, a chromosome from the "Best Class" is selected randomly. Otherwise, a chromosome from the entire population is selected. After selecting two chromosomes to participate in crossover, a single point crossover is applied on each of the *partition part*, *distribution part* and the *assignment part* of the chromosome.

Mutation

To select a chromosome participating in mutation, a uniform random number between 0 and 1 is generated. If the number is greater than 0.5, a chromosome from the "Best Class" is selected randomly. Otherwise, a chromosome from the entire population is selected. Then we select a random point in each of the five fields of the chromosome and change its value. For the first field, we complement a randomly selected bit among the least significant $\lceil logW \rceil$ bits, W being the total TAM width. For the second and third fields we replace with randomly generated values in the range 0 to 1. For the distribution part, normalization is carried out to ensure the unity sum requirement.

4.4.3 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 and is explained next. Consider an SOC consisting of N cores with B_h and B_l being the number of partitions for high-speed and low-speed TAMs respectively. Hence B $(= B_h + B_l)$ is the total number of TAM partitions.

Different TAMs can be used simultaneously for delivering test data to the cores while the cores assigned to the same TAM are to be tested sequentially. It is to be noted that a core cannot be assigned to both high-speed and low-speed TAMs. So, total testing time for a TAM is the sum of the testing times for all the cores assigned to it. The total time to test all the cores in the SOC is the maximum of the times taken among the TAMs. The testing time $T_{ih}(w_j)$ for a core i assigned to a high-speed TAM of width w_j is calculated as

$$T_{ih}(w_j) = (1 + max\{S_i, S_o\}) \times P_i + min\{S_i, S_o\} \tag{4.1}$$

Where, P_i is the number of test patterns for Core i and $S_i(S_o)$ is the length of the longest wrapper scan-in(scan-out) chain obtained from the wrapper design algorithm presented in [72]. To get the test time needed to test all the cores on TAM j we use binary variables x_{ij} (where $1 \leq i \leq N$ and $1 \leq j \leq B_h$) to determine the assignment of cores to high-speed TAMs in the SOC. Let variable x_{ij} be defined as follows:

$$x_{ij} = \begin{cases} 1 & \text{Core } i \text{ is assigned to high-speed TAM } j \\ 0 & \text{otherwise} \end{cases}$$

Hence, the test time required to test all cores on TAM j is given by $\sum_{i=1}^{N} T_{ih}(w_j) \times x_{ij}$.
Similarly for low-speed TAMs the time required to test all cores on TAM k is given by $\sum_{i=1}^{N} T_{il}(w_k) \times y_{ik}$, where $y_{ik}(1 \leq i \leq N$ and $1 \leq k \leq B_l)$ is the binary variable that determines the assignment of cores to low-speed TAMs in the SOC and be defined as

$$y_{ik} = \begin{cases} 1 & \text{Core } i \text{ is assigned to low-speed TAM } k \\ 0 & \text{otherwise} \end{cases}$$

Since all the TAMs can be used simultaneously for testing, the system testing time equals

$$T = max_{\{1 \leq j \leq B_h \text{and} 1 \leq k \leq B_l\}}\{\sum_{i=1}^{N} T_{ih}(w_j) \times x_{ij}, \sum_{i=1}^{N} T_{il}(w_k) \times y_{ik}\} \tag{4.2}$$

4.4.4 Evolution Process

It has been found that a population of size 3000 produces good results for the benchmark circuits. First 20% of the chromosomes with lesser test time are taken as the "best class chromosomes". We copy the best class chromosomes directly and select 80% chromosomes by crossover. This new population is sorted according to their fitness values. Next, we copy 80% directly to the next generation and select 20% via mutation as the population for the next generation, keeping the population size fixed. The resulting population is sorted according to their fitness values. The population thus obtained by operating both the genetic operators is termed as new generation. This process is repeated up to certain predefined number of generations. For our experimentation we took 500 generations to obtain the results.

Algorithm 4.1: Power aware test scheduling algorithm

Input :

1. A chromosome identifying TAM assignments to cores.
2. Power values of each core.
3. Maximum power budget (P_{max})

Output: Final test time and complete schedule after placing all cores in the list satisfying power constraint

```
1  Data Structure:
2  time_instants
   /* Global list of tuples like < time_instant, power_value >. The
      entry time_instant represents a particular instant of time
      and power_value represents the power consumed by the cores
      scheduled for testing at that particular instant          */
3  begin
4      time_instants=NULL;
5      Schedule first core as specified by the chromosome;
6      $time\_instants = time\_instants \bigcup \{< 0, core\_test\_power >\} \bigcup \{< core\_test\_time, 0 >\}$;
7      while (list of cores not null) do
8          $R$=Take next core from the list;
9          $TAM_i$= TAM assigned to R (depicted by the chromosome);
10         foreach (free time interval $T_{gap}$ on $TAM_i$) do
11             if ($T_{gap} \geq$ Test time of R) then
12                 Check for power constraints at each entry in time_instants
                   list in the range of $T_{gap}$;
13                 if (power constraint met) then
14                     Update time_instants list;
15                     Schedule the core $R$ at $T_{gap}$;
16                     Update the list of cores scheduled on $TAM_i$;
17                     Break from loop;
18                 end
19             end
20             if (no such time gap exists) then
21                 Determine the next available time instant after the last core
                   scheduled so far on $TAM_i$, such that power constraint is
                   met;
22                 Update time_instants list;
23                 Schedule the core $R$ at $T_{gap}$;
24                 Update the list of cores scheduled on $TAM_i$;
25             end
26         end
27     end
28     Return the maximum time amongst all TAMs;
29 end
```

Core	Peak power(PSF)
c432	660
c499	602
c880	823
c1355	275
c1908	690
c2670	354
c3540	530
c5315	753
c6288	641
c7552	1144

Table 4.1: Peak Power Data for ISCAS-85 cores in d695

4.5 Impact of Dual-Speed TAMs on Test Power

Excessive power consumption in an SOC during scan testing can cause overheating, which may lead to damage of the chip [179]. Hence, it is important to find out the effect of the TAM design on power consumption during test and also at the same time it is required to reduce the testing time. In this section we have studied the impact of dual-speed TAM architecture on overall SOC test power. As power consumption is directly proportional to the operating frequency of the circuit [180], the use of high-speed TAM, though reduce the test time significantly can cause an increase in power consumption during scan testing. We have studied the impact of using high-frequency TAM lines on test power for one of the ITC '02 benchmark circuits. Since power estimation models for the ISCAS benchmark circuits have been published in the literature, we utilize one of the SOC benchmarks that consist of ISCAS circuits.

Though during test, power consumption varies over time [64], to simplify, we assume peak power value attached to each test. Although these power values are for functional patterns, we use these values for scan test power due to the lack of any additional power information for these circuits. The power of a core scheduled on the high-frequency TAM is calculated as the frequency factor f times its power for the low-frequency TAM. The peak power is measured as peak switching frequency (PSF) per node [181, 57] and the average power is measured in mW [182]. Table 4.1 (reproduced from Table 3.1) shows the peak power data that is considered for each core in d695. The power constrained test scheduling algorithm (Algorithm 4.1) is used to estimate the test time of the SOC under power constraints.

(a)

p22810										
T_n	n=100%		n=75%		n=50%		n=25%		n=0%	
	Method [4]	Ours	Method [4]	Ours	Method [4]	Ours	Method [4]	Ours	Method [4]	Ours
W=16	11315.97	10960.92	14652.25	12755.62	17236.85	14404.20	25329.25	17617.50	22631.95	21870.00
LB	10313.47		11835.15		13696.55		16339.55		20626.95	
W=24	7694.45	7559.62	9987.25	8685.00	10838.75	10290.60	14807.95	12316.60	15389.00	15258.90
LB	6966.75		7864.74		9391.17		10984.85		13932.05	
W=32	6153.75	5844.92	6966.00	6789.60	8641.20	7666.10	10437.95	9781.30	12307.50	11582.25
LB	5156.74		5775.39		7270.85		8179.46		10313.47	
W=40	4932.37	4865.65	5890.00	5444.32	6966.00	6714.60	7914.70	7776.10	9864.65	9617.30
LB	4125.39		5150.57		5805.78		7270.43		8250.78	
W=48	4181.35	3976.70	5159.30	4833.425	6069.95	5626.90	6691.10	6722.15	8362.80	8553.20
LB	3635.40		3962.49		5155.12		5399.65		7270.85	
W=56	3635.40	3635.40	4393.70	4219.20	5150.55	5148.25	6424.55	5944.35	7270.85	7270.85
LB	3635.40		3635.40		5148.25		4617.65		7270.85	
W=64	3423.45	3415.30	3985.75	3864.37	4915.35	4175.55	5531.70	5257.90	6847.05	6833.20
LB	3339.67		3635.40		3649.05		4085.38		6679.35	

(b)

p34392										
T_n	n=100%		n=75%		n=50%		n=25%		n=0%	
	Method [4]	Ours	Method [4]	Ours	Method [4]	Ours	Method [4]	Ours	Method [4]	Ours
W=16	25592.45	24969.62	33798.55	28117.95	42318.60	32301.77	51660.45	39973.95	51191.00	49939.25
LB	23422.03		28117.95		32301.78		39152.02		46844.06	
W=24	18021.35	16911.95	22469.20	18962.40	25830.20	23444.55	31902.15	28646.70	37971.35	33826.35
LB	16579.82		18745.90		22118.87		25587.78		31229.37	
W=32	13614.45	13673.45	18985.65	14931.72	18404.10	17655.75	21813.80	20775.15	27228.95	27346.90
LB	13614.45		14690.22		16579.82		18620.49		27228.95	
W=40	13614.45	13614.45	13614.45	13614.45	14703.20	14703.20	17008.75	16689.30	27228.95	27228.95
LB	13614.45		13614.45		14703.20		15879.45		27228.95	
W=48	13614.45	13614.45	13614.45	13614.45	14703.20	14703.20	14889.85	14703.20	27228.95	27228.95
LB	13614.45		13614.45		14703.20		14703.20		27228.95	
W=56	13614.45	13614.45	13614.45	13614.45	13614.45	13614.45	14703.20	14703.20	27228.95	27228.95
LB	13614.45		13614.45		13614.45		14703.20		27228.95	
W=64	13614.45	13614.45	13614.45	13614.45	13614.45	13614.45	14703.20	14703.20	27228.95	27228.95
LB	13614.45		13614.45		13614.45		14703.20		27228.95	

(c)

p93791										
T_n	n=100%		n=75%		n=50%		n=25%		n=0%	
	Method [4]	Ours	Method [4]	Ours	Method [4]	Ours	Method [4]	Ours	Method [4]	Ours
W=16	46278.15	43747.72	60845.35	49966.07	85952.05	58301.27	102033.40	69828.85	92556.75	87189.20
LB	42677.38		48838.55		57130.59		67939.70		85354.75	
W=24	31135.45	29681.70	43795.65	338806.80	60795.50	39269.97	76184.00	47122.50	62439.75	59471.50
LB	28451.58		32585.88		37792.67		45486.94		56903.17	
W=32	24375.40	21972.72	31135.45	25221.82	37761.05	29723.97	59911.05	35351.45	48750.80	43947.20
LB	21338.69		24189.09		28411.75		34319.71		42677.38	
W=40	19850.50	17927.80	23184.00	20253.55	28588.90	24647.08	33035.45	28541.80	39701.00	36225.30
LB	17070.95		19378.60		22550.58		27424.18		34141.90	
W=48	15698.10	15007.10	20691.35	17185.15	23665.90	20079.57	28436.15	24078.70	31396.70	29833.15
LB	14225.79		16201.07		18984.05		22799.43		28451.58	
W=56	14210.90	12965.05	17915.55	15041.12	20383.35	17710.07	24223.55	21710.50	28421.80	25895.95
LB	12193.54		13704.15		15766.08		19770.77		24387.07	
W=64	12782.15	11483.87	15727.15	12979.50	16806.55	15488.62	22794.75	18061.30	25564.30	22968.05
LB	10669.34		12120.14		14066.12		17240.74		21338.69	

Table 4.2: Testing Time Results for (a) p22810 ($f = 2$) (b) p34392 ($f = 2$) (c) p93791 ($f = 2$)

4.6 Experimental results

In this section, we present experimental results on test scheduling for the three largest SOCs (in terms of the number of cores) from the ITC'02 SOC Test Benchmark suite [172]. All the experiments have been conducted on a 2.66 GHz Pentium IV machine with 512 MB memory. The proposed algorithm has been implemented in C. The testing time is calculated in μs, the low-speed TAM lines are assumed to be driven at 20 MHz, and the high-speed TAM lines are assumed to be driven at $20f$ MHz for different values of f. The following sections show the test time results with and without power constraints.

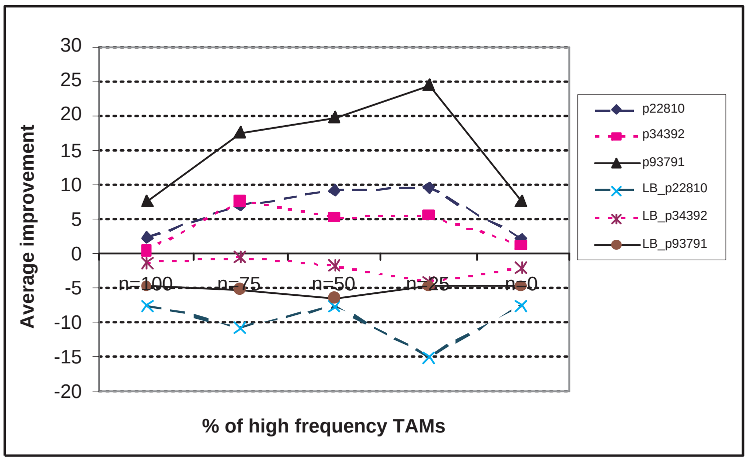


Figure 4.2: Comparison of average % improvement over [4] and lower bound (LB) results

4.6.1 Test scheduling without power consideration

In Table 4.2, we present the test time results for three benchmark circuits: p22810, p34392 and p93791. Testing times are presented for various values of TAM widths W and a range of values for V. Number of TAM wires driven by the high-speed ATE channels is represented in terms of n, where n is the % of total TAM lines considered to be at high speed. We also assume in this set of experiments that the high-speed data rate is twice that of the low-speed data rate, i.e., $f = 2$. Table 4.2 also gives a comparison of the testing time with [4]. The values corresponding to 'LB' are the lower-bound values as described in Section 4.3. The maximum improvement in test application time has been found to be 40.99% for p93791 with a TAM width of $W = 32$ among which 25% of the TAM lines are high speed. In some cases it can be observed from Table 4.2 that the result produced by our method is a bit higher as compared to the one in [4]. For example, for core p22810 with $W = 48$, $n = 25\%$, test time of our approach is 0.46% higher than [4]. This happens due to the less number of TAM partitions for low speed TAMs. For this case, out of 36 (for $n = 25\%$) low speed TAM lines only 2 partitions have been made. These are of widths 5 and 31. Similarly, for core p34392 with $W = 32$, $n = 100\%$ and $n = 0\%$ the results are 0.43% poorer, for both cases.

(a) p22810

W	Comparison with	n=100%	n=75%	n=50%	n=25%	n=0%
16	[4]	3.13	12.94	16.43	30.45	3.37
	LB	-5.90	-7.21	-4.91	-7.25	-5.68
24	[4]	1.75	13.04	5.05	16.82	0.84
	LB	-7.84	-9.45	-8.74	-10.81	-8.69
32	[4]	5.02	2.53	11.28	6.29	5.89
	LB	-11.77	-14.94	-5.16	-16.37	-10.95
40	[4]	1.35	7.57	3.60	1.75	2.50
	LB	-15.21	-5.39	-13.53	-6.50	-14.21
48	[4]	4.89	6.32	13.29	-0.46	2.89
	LB	-8.58	-18.01	-8.38	-19.67	-10.47
56	[4]	0	3.97	0.04	7.47	0
	LB	0	-13.84	0	-22.32	0
64	[4]	0.23	3.04	15.05	4.94	0.20
	LB	-2.21	-5.93	-12.61	-22.3	-2.25
Avg. % impr. over	[4]	2.34	7.06	9.25	9.64	2.24
	LB	-7.36	-10.68	-7.62	-15.03	-7.46

(b) p34392

W	Comparison with	n=100%	n=75%	n=50%	n=25%	n=0%
16	[4]	2.43	16.8	23.67	22.62	2.44
	LB	-6.19	0	0	-2.06	-6.19
24	[4]	2.02	15.60	9.23	9.24	7.26
	LB	-1.96	-1.94	-5.65	-10.68	-7.68
32	[4]	-0.43	21.35	4.07	4.85	-0.43
	LB	-0.43	-1.62	-6.09	-10.37	-.43
40	[4]	0	0	0	1.87	0
	LB	0	0	0	-4.85	0
48	[4]	0	0	0	1.25	0
	LB	0	0	0	0	0
56	[4]	0	0	0	0	0
	LB	0	0	0	0	0
64	[4]	0	0	0	0	0
	LB	0	0	0	0	0
Avg. % impr. over	[4]	0.57	7.68	5.28	5.69	1.32
	LB	-1.23	-0.51	-1.68	-3.99	-2.04

(c) p93791

W	Comparison with	n=100%	n=75%	n=50%	n=25%	n=0%
16	[4]	5.46	17.88	32.17	31.56	5.80
	LB	-2.45	-2.26	-2.01	-2.70	-2.10
24	[4]	4.67	22.81	35.41	38.15	4.75
	LB	-4.14	-3.61	-3.76	-3.47	-4.32
32	[4]	9.85	18.99	21.28	40.99	9.85
	LB	-2.89	-3.72	-4.41	-2.92	-2.89
40	[4]	9.68	12.63	13.79	13.60	8.75
	LB	-4.78	-4.32	-8.50	-3.92	-5.75
48	[4]	4.40	16.94	15.15	15.32	4.82
	LB	-5.21	-5.73	-5.46	-5.31	-4.63
56	[4]	8.77	16.04	13.11	10.37	8.89
	LB	-5.95	-8.89	-10.97	-9.26	-5.83
64	[4]	10.16	17.47	7.84	20.76	10.15
	LB	-7.09	-6.62	-9.18	-4.54	-7.09
Avg. % impr. over	[4]	7.57	17.53	19.82	24.39	7.57
	LB	-4.64	-5.02	-6.33	-4.59	-4.66

Table 4.3: Average % improvement in test time results over [4] and lower bound values

However, on an average, the partitioning approach based on GA performs better than the bin-packing approach of [4]. This is depicted in Table 4.3. The table presents the % improvement in test time over [4] (rows marked with [4]) and also over the lower bound results (rows marked with 'LB'). From Table 4.3(a) it is noted that maximum average improvement of 9.64% in test time is obtained when $n = 25\%$. But in this case also test time results are on an average 15% higher than the lower bound results. So, better heuristic method can be applied to minimize the deviation from the lower-bound. From Table 4.3(b) it is noted that SOC test time results are close to the lower-bound values. From $W = 40$ onwards for almost all of the cases test time results and lower-bound values are same. Again from Table 4.3(c) it is noted that maximum average improvement 24.39% is achieved for $n = 25\%$. For this SOC (p93791) also test time results are closer to the lower-bound values (within the limit of 6%). Results of Table 4.3 is summarized in Figure 4.2. It represents the average % improvement in test time over [4] for the SOCs and also the average percentage by which the results are worse than the corresponding lower bound values.

4.6.2 Test scheduling with power consideration

In this section, we present results for power-constrained test scheduling for d695 in Table 4.4 considering power values given in Table 4.1. The results are presented for power limits of $2,500$ PSF and $3,000$ PSF for $n = 25\%$, 50% and 0%. We compare the test time (TP_1 and TP_2 for power limits of $2,500$ PSF and $3,000$ PSF respectively) of the power-constrained test schedule for different values of W to that of unconstrained test schedule TP_0. It is important to note that, in case of power-constrained test scheduling, we do not attempt to optimize the test schedule for test power. The goal here is minimize the test time under power limits. It is seen from the table that for most of the TAM widths our method provides better results than the heuristic approach in [4]. It is also to be noted from the Table 4.4 that for $n = 25\%$ and $n = 50\%$ improvement in test time is more than that for the case of $n = 0\%$. Maximum average improvement of 40.05% in test time is obtained for $n = 50\%$ with power limit of 2500 PSF.

For both unconstrained and constrained testing our approach produces better result than [4]. The reasons that can be attributed are as follows.

1. While [4] uses a bin packing approach, we have used a TAM partitioning technique. The search space for our optimization problem is simpler, since the test rectangles are now being packed in a proper fashion only.

2. Genetic Algorithm explores the search space better than the heuristic in [4].

(a) n=0%

W	TP_0		TP_1		TP_2	
	Method [4]	Ours	Method [4]	Ours	Method [4]	Ours
16	2146.95	2113.40	2146.95	2121.40	2146.95	2113.40
24	1463.55	1414.60	1463.55	1419.45	1463.55	1414.60
40	884.50	883.85	898.75	892.80	884.50	883.85
48	743.65	733.30	789.35	848.80	762.50	848.75
56	621.05	644.20	668.80	647.05	647.05	646.75
64	580.20	521.70	624.80	548.75	618.80	521.70
Avg. % impr.	**2.14**		**2.21**		**1.76**	

(b) n=25%

W	TP_0		TP_1		TP_2	
	Method [4]	Ours	Method [4]	Ours	Method [4]	Ours
16	2786.10	1690.75	3016.35	1690.75	2786.10	1690.75
24	1738.30	1145.20	1738.25	1145.20	1738.30	1145.20
40	881.20	682.60	1070.60	693.40	917.90	682.60
48	623.50	582.25	815.45	588.65	876.60	582.25
56	551.65	503.45	704.95	517.35	625.60	503.45
64	531.25	449.5	704.95	470.40	587.20	449.50
Avg. % impr.	**19.69**		**32.60**		**31.02**	

(c) n=50%

W	TP_0		TP_1		TP_2	
	Method [4]	Ours	Method [4]	Ours	Method [4]	Ours
16	1724.50	1399.80	2117.65	1405.00	1928.30	1399.80
24	1131.25	958.25	1466.30	964.70	1202.75	958.25
40	672.55	572.55	1031.55	587.55	788.95	572.55
48	551.65	493.45	927.80	501.35	682.30	493.45
56	521.70	475.85	848.75	493.45	639.10	493.45
64	464.10	374.20	767.90	405.00	602.35	374.20
Avg. % impr.	**13.73**		**40.05**		**25.93**	

Table 4.4: Testing Time in μs, with no power limit (column TP_0), and power limits of 2,500 PSF (column TP_1) and 3,000 PSF (column TP_2) for (a) n=0% (b) n = 25% (c) n = 50%

4.7 Conclusion

In this chapter we have presented a Genetic Algorithm based test scheduling approach for minimizing testing time considering two different data rates for ATE channels. We use the SOC-level TAM architecture optimization by partitioning TAMs and then assigning cores to the TAMs. As far as core testing configurations are concerned, for different number of TAM wires allocated in wrapper design, the test time will vary. In our formulation, we have considered only some of the configurations which are needed for various partitions depicted by the chromosome structure. The bin packing approach [4] attempts to explore all possibilities. However, the search space becomes too large. As it is shown in the experimental results, our approach though does not explore the entire search space, produces better results. Of course, a better search technique with bin packing may yield even better results. Experimental results have been presented for ITC'02 benchmark circuits and compared with [4]. We have also considered the impact of dual-speed TAM on power consumption and its effect on overall SOC test time.

Chapter 5

Test Infrastructure Design for Hierarchical SOCs

5.1 Introduction

Most of the works on test scheduling (including our works reported in last two chapters) consider the SOC design hierarchy as flattened for the purpose of simplification. But this assumption is not always realistic in practice, especially when for design-reuse, older-generation SOCs are used as hard cores in new SOC designs [183]. A 'parent' core may thus contain several 'child' cores, which in turn may contain their own child cores. The core vendor may have already designed an optimized TAM within the hierarchical core that is provided to the system integrator. Hence assigning extra TAM wires to sub-cores is not feasible. A hierarchical core itself may be an SOC; hence it has its own constraints. A complete scheduling should consider the constraints of both the top-level SOC and hierarchical embedded cores at the same time.

In this work, test infrastructure design approach for hierarchical cores and chip level test scheduling have been proposed. The approach is based on hierarchy aware wrapper design for parent core, TAM optimization for the SOC and parent cores and chip level test scheduling. In [183, 184, 51] scheduling works have been done in reducing test time for SOCs with hierarchical cores. But the wrapper design for the parent cores of an hierarchical SOC in these works does not consider the child core scan-lengths in $INTEST_P$ mode (that is, while testing parent cores), which is proposed in [5]. Only the work in [185] considered this architecture. In this work, we address the test scheduling problem for hierarchical cores on both non-interactive and interactive design transfer model [183] (detailed in Section 5.2) considering the wrapper architecture presented in [5]. A new TAM design space exploration engine using genetic algorithm (GA) is proposed. For non-interactive case of TAM assignment for a hierarchical SOC, it has been seen that with the proposed method upto 31% improvement in test time can be achieved over contemporary approaches. For the interactive case upto 34% improvement has been achieved.

5.2 Hierarchical Embedded Core Design Scenarios

A hierarchical SOC is designed by integrating heterogeneous technology cores at various levels of hierarchy. The ability to reuse the embedded cores in a hierarchical manner helps the electronic industry to reduce the turn around time of integrated circuit chips. This in turn shows that today's SOC is tomorrow's embedded core. Two broad design transfer models are emerging in hierarchical SOC design flows [183]. One is non-interactive deign transfer model and the other is interactive design transfer model.

5.2.1 Non-interactive

In this design transfer model there is limited communication between the core vendor and the core user. The hard cores are taken off-the-shelf and integrated into designs as optimized layouts. There is no scope for the core user or system integrator to modify these hard cores and should use them as it is. In this design transfer model, the parent core provider implements wrappers and TAM architecture for the child cores of a hierarchical core before delivery to the system integrator. The system integrator only designs the overall wrapper/TAM architecture for the hierarchical SOC. The parameters supplied for the hierarchical core includes the information about the test infrastructure of the child cores. System integrator uses this information to design the wrapper for hierarchical cores in $INTEST_P$ and $INTEST_C$ mode as discussed in the next section and then schedule the cores.

Hence test schedule optimization problem can be defined as:
$P_{TP\_non-int}$:*Given 1) the test set parameters for the top-level cores among which one or more top level core may be a parent core with embedded child cores, 2) the total SOC level TAM width W for the SOC, 3) the wrapper and TAM architecture for embedded child cores, determine*

a) a wrapper design for each top-level core (including hierarchical core),

b) an optimal partition of W among the cores in the test schedule *such that the SOC testing time is minimized under the constraints that total width W is not exceeded at any time.*

5.2.2 Interactive

In this design transfer model, there exists certain amount of communication between the core vendor and the core user during system integration. The communication of the core user's requirements to the core vendor can help in determining the core specification to a limited extent.
In this design transfer model, the system integrator can influence the choice of TAM width supplied to the child cores by the core vendors based on system-level TAM width requirements of the cores. Hence, multi-level TAM optimization problem in this model can be described as:

P_{TP_int}:*Given 1) the test set parameters for the top-level cores among which one or more top level core may be a parent core with embedded child cores, 2) the total SOC level TAM width W for the SOC, determine*

a) a wrapper design for each top level core including child cores (in case of hierarchical cores).

b) an optimal partition of W among the cores in the test schedule,

such that the SOC testing time is minimized under the constraints that total width W is not exceeded at any time.
For wrapper design of the hierarchical core we consider the approach as discussed in next section.

5.3 Wrapper design for hierarchical core

For modular testing of the core based SOCs, wrapper provides the interface between the test access mechanism (TAM) and I/Os of the cores. But testing hierarchical cores is different from testing non-hierarchical cores. Main difference occurs due to the functionality of the wrapper cells that buffer the functional inputs and outputs of the core. The input wrapper cell shifts and applies test stimuli in the INTEST mode, and captures and shift test responses in the EXTEST mode. The operation of the output wrapper cells is the reverse of the input wrapper cells in two modes. In this work we have used the configuration of the wrapper cell as shown in Fig. 5.1. In this configuration when the parent core is in INTEST mode, the embedded child cores have to be in EXTEST mode. Hence, simultaneous testing of parent and child cores cannot be done in INTEST mode. Due to this constraint, wrapper architecture and wrapper/TAM optimization techniques used for the non-hierarchical cores cannot be applied directly to hierarchical cores.

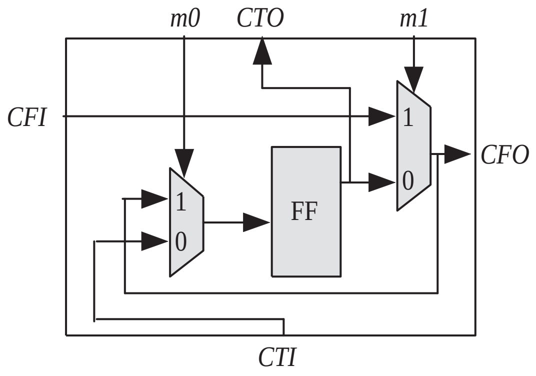


Figure 5.1: Wrapper cell configuration[5]

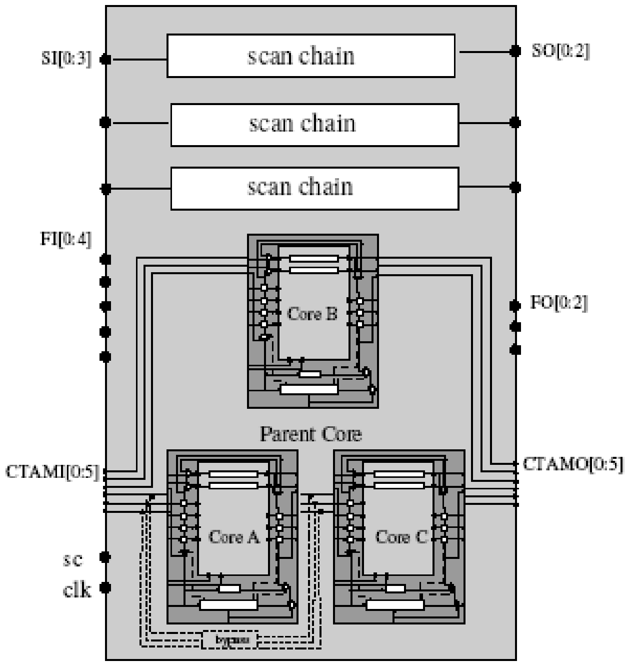


Figure 5.2: Example of wrapper architecture for hierarchical cores with three child cores[5]

This work has considered the wrapper architecture for the hierarchical cores as discussed in [5]. Here in addition to the terminals like functional only, test data and control terminals, a parent core has terminals that provide test access to the wrapped child cores called CTAM terminals as shown in Fig. 5.2. These are inputs and outputs at the parent core level that can be connected directly to the TAM wires of the child core. Taking the architecture into consideration, parent core can have two INTEST modes and one EXTEST mode. For the parent INTEST mode ($INTEST_P$), the test data is scanned through the parent core scan chains, the parent core wrapper cells and the child core wrapper cells. In this case child cores are in EXTEST mode to ensure complete internal testing of the parent core. Similarly, child INTEST mode ($INTEST_C$) ensures about the internal testing of the child cores. In $INTEST_P$ mode the available TAM wires have to be distributed between the parent core scan chains, wrapper cells as well as child core TAM architecture. But in $INTEST_C$ mode all the TAM wires can be utilized by the child cores for their INTEST testing. Both $INTEST_P$ and $INTEST_C$ have to be time multiplexed.

As wrapper architecture mostly influences the test time, wrapper/TAM optimization is one of the crucial problems for efficient testing of the cores. In this work, the wrapper and TAM optimization problem is solved both for the non-interactive and interactive design scenarios. The following section discusses about how wrapper/TAM can be optimized efficiently.

5.3.1 Child INTEST mode ($INTEST_C$) optimization

In this mode, all the available TAM wires can be utilized for child cores. For hard implementation (non-interactive design scenario) of the child cores, a fixed number of TAM wires are available and in that case number of CTAM chains is limited by the specified TAM width assigned to the child cores. But for the soft implementation (interactive design scenario), child core architecture can be designed for any number of CTAM terminals.

So formally, the wrapper design problem in this mode can be defined as:
$P_{wrap\_INTEST_c}$:*For a given set of CTAM chains M, set of child cores N and for each child core $c \in N$, the number of test patterns p_c, total scan-lengths, scan-in and scan-out parameters $sl_{c,k}$, $si_{c,k}$ and $so_{c,k}$ respectively on k chains ($k \in M$) and a set of parent core level TAM wires w ($1 \leq w \leq W$) available for testing, determine the wrapper design for the parent core, such that overall test time to test all the child cores can be minimized and w is not exceeded any time. Scan-lengths are the sum of all scan elements (wrapper I/O cells and the scan chains) in a particular chain.*

In this optimization approach two cases may arise. These are the following.
Case I :($w \geq |M|$)
In this case all the available TAM wires can be connected directly one by one to test the child cores. Hence, total test time taken to test all the child cores N on

w TAM wires is the maximum of the testing time on any of the CTAM chains and this can be represented as

$$T(w) = max_j \sum_i T_i \times x_{ij}, 1 \leq i \leq |N|, \text{ and } 1 \leq j \leq |M| \tag{5.1}$$

Where

$$T_i = \{1 + max(si_i, so_i)\} \times p_i + min(si_i, so_i) \tag{5.2}$$

is the testing time of the core i and si_i and so_i are the maximum scan-in and scan-out lengths of core i respectively among $k(1 \leq i \leq |M|)$ TAM chains.
In Eqn. (5.1), x_{ij} is the binary variable that can be defined as

$$x_{ij} = \begin{cases} 1 & \text{Core } i \text{ is assigned to high-speed TAM } j \\ 0 & \text{otherwise} \end{cases}$$

Case II :($w < |M|$)
In this case available TAM wires are less than the number of CTAM chains $|M|$. So, all the available TAM wires have to be distributed over the CTAM chains, such that, overall test time can be minimized. To do this, two or more CTAM chains are to be daisy-chained to form a TAM chain that shares the same TAM wires. But obviously the scan-lengths of the cores will change accordingly after daisy chaining.

Let us consider two CTAM chains ct_1 and ct_2 and let the set of cores on these CTAMs be P_1 and P_2 respectively. If ct_1 precedes ct_2, then the scan-in and scan-out times of core i on ct_2 can be defined for new CTAM chain ct as

$$\begin{aligned} si_{i,ct} &= \sum_{j \in P_1} sl_{j,ct_1} + si_{i,ct_2} \\ so_{i,ct} &= so_{i,ct_2} \end{aligned} \tag{5.3}$$

If ct_2 precedes ct_1 then

$$\begin{aligned} si_{i,ct} &= si_{i,ct_2} \\ so_{i,ct} &= \sum_{j \in P_2} sl_{j,ct_1} + so_{i,ct_2} \end{aligned} \tag{5.4}$$

If same cores are connected to both the ct_1 and ct_2 then the maximum of si and so from Eqns. (5.3) and (5.4) is chosen as the scan-in and scan-out times of the core on it.

After forming the daisy chain, number of CTAM chains will be equal to the number of available TAM wires and the testing time of the cores are calculated using Eqn. (5.2) where,

$$si_i = max_j(si_{i,j}), 1 \leq i \leq w \text{ and}$$
$$so_i = max_j(so_{i,j}), 1 \leq j \leq w$$

Overall testing time can then be calculated using Eqn. (5.1).

To get the distribution of the TAM wires on the CTAM chains and formation of daisy chain, we have proposed a Genetic Algorithm (GA) based formulation. Due to daisy chaining of CTAMs, test lengths are likely to be increasing. Hence, to reduce the test length we consider the core bypass feature that allows bypassing an entire core in one clock cycle by using a bypass register.

For interactive design scenario only Case I has to be considered. As in this case the system integrator has the choice of selecting the optimized TAM width and all the TAMs can be connected directly to the CTAMs. So number of TAMs and CTAMs are matched in this case. In Section 5.4 we will describe the GA formulation used for daisy chaining.

5.3.2 Parent INTEST mode ($INTEST_P$) optimization

In order to minimize the test time of a core, scan-in and scan-out lengths are to be minimized. In this mode the TAM items are parent core scan chains, parent core wrapper input and output cells and child core wrapper cells.

Before designing the wrapper for parent core, all the information about the number of wrapper I/O cells, number of scan chains and their lengths available at the parent core level are provided by the core vendor. Also the child core scan-lengths, scan-in and scan-out information can be obtained from the $INTEST_C$ wrapper design method. To make the calculation simple during wrapper design, we have used only the scan-length values, not the scan-in and scan-out values. This will of course increase the overall test application time of the core, which is negligible. Hence the problem of wrapper design can now be stated as follows:
$P_{wrap\_INTEST_p}$:*Given a set of wrapper input cells n_i , wrapper output cells m_i, a set of bi-directional inputs b_i, a set of parent core internal scan chains S_i and their corresponding length $l_{i,k}$, a set of CTAM scan-lengths S_c with their corresponding length $l_{c,v}$ and a set of w TAM wires, determine a TAM partition having w subsets such that overall test length of the core is minimized for all partitions.*

The problem is same as the wrapper design problem for non-hierarchical cores and it has been shown that it is an NP-hard problem [56]. To solve the problem we have used the algorithm same as *Design_wrapper* proposed in [72]. The steps are as follows:

1. Partition the parent core internal scan chains into w subsets corresponding to w TAM chains such that the maximum sum of scan-lengths assigned to a TAM chain is minimized.
2. Assign the child core scan-lengths to the w subsets such that sum of the scan-lengths to a TAM chain is minimized.
3. Assign the wrapper input cells to the subsets obtained from Step 1 and 2 such that maximum scan-in lengths of all the TAM chains are minimized.
4. Assign the wrapper output cells to the subsets from Step 1 and 2 such that maximum scan-out lengths of all the TAM chains are minimized.

5.4 Genetic Algorithm Formulation

In this section we discuss the genetic algorithm (GA) formulation that are employed to solve the daisy chain construction problem in $INTEST_C$ mode and the overall test scheduling problem. First we will start with the formulation to construct daisy chains when number of TAM wires are less than the number of CTAMs allotted for child cores in non-interactive design transfer model.

5.4.1 GA formulation for daisy chaining

A chromosome represents the solution of the problem that determines which CTAMs are to be daisy-chained depending on the number of available TAM wires w. In this problem the chromosome structure is an array of integers $< r_1, r_2, ..., r_w >$. Size of the chromosome depends on the number of CTAM chains available for the child cores. For initial population of the GA all the r_i values are generated randomly between 1 and w. For example, consider that the number of CTAM chains available for the child cores be 5 and the available TAM width is only 3. So 2 CTAM chains are to be merged to get 3 CTAM chains to match with the available TAM wires. So, if the chromosome is $< 1, 3, 2, 1, 3 >$, then for this case 3 CTAM chains after daisy chaining will be $(1, 4)$, $(2, 5)$ and 3. Here CTAM chain 1 and 4 are to be daisy chained and similarly 2 and 5. CTAM chain 3 will remain as it is.

For the evolution of generations, two operators – crossover and mutation are carried out. For the crossover operation, two randomly selected chromosomes are chosen and single point (randomly selected) crossover operation is performed. For mutation operation, the chromosome is selected randomly and single point (randomly selected) mutation operation is carried out. Mutation is performed by generating random value between 1 and w and use it to replace the value at the mutation point.

The cost of the chromosome gives the testing time of the child cores in $INTEST_C$ mode when available TAM width is smaller than $|M|$. This is calculated using the Eqn. (5.2) as discussed earlier.

As a small number of CTAMs are allotted to the child cores of a hierarchical core, we have taken the population size to be 100. Among them 20 best chromosomes are copied directly to the next generation and remaining 80 are created using crossover and mutation operations. For every generation, the population is kept sorted according to their cost values. The GA is run for 50 generations and the best chromosome is considered as the solution for this problem.

5.4.2 GA formulation for the overall scheduling

This section describes the main technique employed in our hierarchical test scheduling algorithm. In essence, the test scheduling algorithm is developed based on GA.

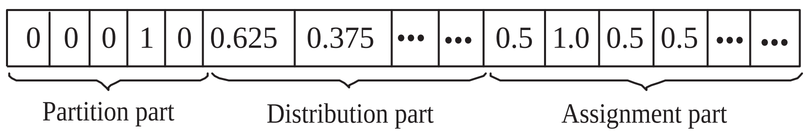


Figure 5.3: Chromosome structure for determining TAM partitions and Scheduling

Solution Representation

A chromosome in our approach consists of three parts as shown in Fig. 5.3. First part, named as partition part, is the number of TAMs for the SOC. Since the total TAM width W can be encoded with a binary string of $(log_2 W + 1)$ bits, the partition part is an array of $(log_2 W + 1)$ bits. Second part termed as distribution part gives the partition of the total width among the TAMs. This is an array of real numbers between 0 and 1, where the j^{th} entry of this array multiplied by the total TAM width represents the width of bus j. The array size is equal to the decimal value of the first part. However to keep the chromosome length fixed, we have used W entries here, only the first few are actually used (depending upon the value in partition part). Third part is the assignment part and is also an array of real numbers between 0 and 1, where the j^{th} entry of the array multiplied by the decimal value of the first part represents the bus assigned to the j^{th} core. The array size is equal to the total number of cores in the SOC. These three parts of the chromosome form a solution to the given problem. For the example in Fig. 5.3, number of bits in partition part is 5 and number of partitions is 2 (00010). So, in distribution part first two entries will be considered and the respective TAM widths are $(0.625 \times W)$ and $(0.375 \times W)$. Third part is the assignment part. This part will be of size equal to the number of cores in SOC. As in this example, number of partitions is 2 and first value in the assignment part is 0.5, TAM number 1 $(0.5 \times 2 = 1)$ of width $0.625 \times W$ is assigned to core 1 and so on. It may be noted that if the value of partition part is k, the first k values of the distribution part are normalized, such that their sum is unity. This is required to ensure that the total TAM width utilized is equal to W.

Genetic operators

Crossover: The whole population is sorted according to their fitness values. To select a chromosome participating in cross-over, first a uniform random number between 0 and 1 is generated. If the number is greater than 0.5, a chromosome from the "Best Class" is selected randomly. Otherwise, a chromosome from the entire population is selected. After selecting two chromosomes to participate in crossover, a single point crossover is applied on the each of the parts.
Mutation: To select a chromosome participating in mutation, first a uniform random number between 0 and 1 is generated. If the number is greater than 0.5, a chromosome from the "Best Class" is selected randomly. Otherwise, a chromosome from the entire population is selected. Then we select a random point in each of the three fields of the chromosome and change its value. For the first field, we complement a randomly selected bit among the least significant $\lceil logW \rceil$ bits, W being the total TAM width. For the second and third fields we replace with randomly generated values in the range of 0 to 1. For the distribution part, normalization is carried out to ensure the unity sum requirement.

Fitness measure

Overall test time required to schedule all the cores in SOC represents the cost of the solution. Hence, fitness of the chromosome can be represented by the overall test cost and can be obtained as follows:

For the hierarchical cores, test times are set to the values obtained after designing the wrapper for the parent core in $INTEST_P$ and $INTEST_C$ mode. For non-interactive case, the CTAM lines allotted to a hierarchical core is fixed. During TAM partitioning, if the number of allotted TAM lines is less than the CTAM lines then daisy chains are to be formed in $INTEST_C$ mode otherwise all the TAM lines can be directly connected. For interactive design scenario number of CTAM lines is matched with the parent core level TAM wires. So, all the TAM lines can be connected directly to the CTAM lines. If the TAM width allotted is w_j and corresponding test time for the hierarchical core is $T_i(w_j)$, then

$$T_i(w_j) = T(w_j)_{INTEST_P} + T(w_j)_{INTEST_C} \tag{5.5}$$

$T(w_j)_{INTEST_C}$ and $T(w_j)_{INTEST_C}$ values are obtained from wrapper design in $INTEST_P$ and $INTEST_C$ mode.

The time required to test all cores on TAM j is given by $\sum_{i=1}^{N} T_i(w_j) \times x_{ij}$. Binary variable x_{ij} is used (where $1 \leq i \leq N$ and $1 \leq j \leq B$) to determine the assignment of cores to TAMs in the SOC.

Since all the TAMs can be used simultaneously for testing, the system testing time equals $max_j\{\sum_{i=1}^{N} T_i(w_j) \times x_{ij}\}$.

Evolution process

Initial population of size 3000 is taken. First 20% of the chromosomes with less test time are taken as the “best class chromosomes”. We copy the best class chromosomes directly and select 80% chromosomes by crossover. This new population is sorted according to their fitness values. Now we copied 80% directly to the next generation and select 20% via mutation as the population for the next generation, keeping the population size fixed. The resulting population is sorted according to their fitness values. The population thus obtained by operating both the genetic operators is termed as new generation. This process is repeated upto certain predefined number of generations. For our experimentation we took 100 generations to obtain the results.

p22810		p34392		p93791	
Module	#CTAM	Module	#CTAM	Module	#CTAM
Module 1	15	Module 2	20	Module 1	16
Module 5	16	Module 10	20	Module 6	16
		Module 18	16	Module 14	16
				Module 17	16
				Module 20	16
				Module 23	16
				Module 27	16
				Module 29	16

Table 5.1: Number of CTAMs allotted for the child cores of the hierarchical cores in non-interactive design scenario

Hierarchical core 1			Hierarchical core 5		
W	$INTEST_P$	$INTEST_C$	W	$INTEST_P$	$INTEST_C$
2	508527	602632	2	255565	292789
3	347411	369090	3	170511	212485
4	260951	282816	4	127884	139129
5	208283	270490	5	102307	132437
6	175276	196541	6	85458	95631
7	173705	196541	7	73279	95631
8	173705	196541	8	64144	95631
9	173705	196541	9	57243	92998
10	117113	147898	10	52371	92285
11	102965	147897	11	47702	70260
12	102965	110266	12	43644	66201
13	102965	110266	13	43644	66201
14	102965	110266	14	43644	52848
15	102965	61623	15	43644	52848
			16	43644	26764

Table 5.2: Test times of hierarchical cores in $INTEST_P$ and $INTEST_C$ mode of p22810

5.5 Experimental Results

In this section, we present experimental results for both the non-interactive and interactive design transfer models. The experimental results are presented for three SOCs: p22810, p34392, and p93791 from the ITC'02 SOC test benchmarks [172]. These SOCs are appropriate for the experiments because they are hierarchical, containing multiple levels of embedded cores. Algorithms are implemented in C language and executed with a 2.6 GHz Pentium IV machine with 512 MB memory. First we will start with the test time results obtained in non-interactive design scenario.

5.5.1 Non-interactive Design Transfer Model

To obtain the hard implementation of the child core architecture we have used the GA as discussed in Section 5.4. The child core architectures consider the CTAM lines as in [5]. The CTAM widths used for the hierarchical cores of the SOCs p22810, p34392 and p93791 are shown in Table 5.1. Table 5.2 shows the results obtained after designing the wrapper for the parent core in $INTEST_P$ and $INTEST_C$ mode for two hierarchical cores of the SOC p22810.

Table 5.3 shows the overall testing time results for the SOCs with various TAM widths. Number of TAM partitions and their corresponding widths for which these test time results are obtained have also been shown. We compare the testing times (in clock cycles) of the proposed hierarchical TAM optimization

p22810				
	Proposed Hierarchical Method			
W	B	TAM Partitions	Test Time (in clock cycles)	CPU Time (in sec.)
16	3	5,5,6	486805	90.45
24	3	6,10,8	350416	111.20
32	3	6,15,11	266495	115.45
40	3	9,15,16	224038	119.73
48	4	11,16,15,6	179912	120.24
56	4	16,17,15,8	164588	119.33
64	4	13,19,17,15	164588	119.55
p34392				
	Proposed Hierarchical Method			
W	B	TAM Partitions	Test Time (in clock cycles)	CPU Time (in sec.)
16	2	8,8	1199751	41.43
24	4	9,10,1,4	780315	41.57
32	3	15,5,12	673914	37.47
40	4	15,6,18,1	606261	35.33
48	3	11,21,16	606261	40.29
56	4	20,12,5,19	606261	45.09
64	4	1,12,26,25	606261	47.14
p93791				
	Proposed Hierarchical Method			
W	B	TAM Partitions	Test Time (in clock cycles)	CPU Time (in sec.)
16	3	7,3,6	1832531	108.45
24	2	8,16	1214462	100.19
32	2	23,9	920788	100.29
40	2	16,24	746167	97.04
48	2	24,24	643404	93.54
56	3	24,16,16	534247	78.17
64	3	23,24,17	471392	65.85

Table 5.3: Overall test time results for the non-interactive scenario

method with those of the corresponding method in [185]. Column 'B' of Table 5.1 denotes the number of SOC-level TAM partitions and W denotes total bus width available. In Table 5.3, we present the results for p22810, p34392 and p93791. For SOC p22810 test times are not decreasing further after $W > 48$. This is mainly due to the hierarchical core 1. Whereas, testing times for the SOC p34392 reaches a minimum value of 606261 clock cycles at $W = 40$. This lower bound is the minimum testing time for core 18 and becomes the bottleneck core. But for SOC p93791, there is no such bottleneck core and hence test times are

p22810			
W	Our	[185]	% improvement
16	486805	544763	10.64
24	350416	386464	9.33
32	266495	291539	8.59
40	224038	239205	6.34
48	179912	239205	24.79
56	164588	239205	31.19
64	164588	239205	31.19

p34392			
W	Our	[185]	% improvement
16	1199751	1484442	19.18
24	780315	976347	20.07
32	673914	792544	14.97
40	606261	606261	0
48	606261	606261	0
56	606261	606261	0
64	606261	606261	0

Table 5.4: Comparison of Test time results with previously proposed method (non-interactive design scenario)

decreasing with the increase in TAM widths. Again it is to be noted that test time of any hierarchical core cannot decrease below the test time of its child core architecture when CTAM width matches with the parent core level TAM width.

We compare our results with the similar approaches. It is seen from the Table 5.4 that our proposed method provides better test time results than the previously proposed approach [185]. For p22810, maximum improvement has been obtained and it is noted that upto 31.19% (with width $W > 48$) improvement in test time can be achieved with the proposed method. As p34392 contains a bottleneck core (core 18), no improvements in test time occurs for $W > 32$. But for $W \leq 32$ improvements in test times are significant. Results of p93791 are not available in [185]. Thus, no further comparison could be made. We have of course provided our results in Table 5.3.

5.5.2 Interactive design transfer model

We present the test time results for interactive design scenario in Table 5.5. Number of SOC level TAM partitions and the corresponding TAM widths are also shown in Table 5.5. It shows the test time results for p22810, p34392 and for p93791. It is noted from the Table 5.5 that for p22810 test times are not decreasing as much with the increase in SOC level TAM width. This is mainly

p22810				
	Proposed Hierarchical Method			
W	B	TAM Partitions	Test Time (in clock cycles)	CPU Time (in sec.)
16	3	5,5,6	458255	120.09
24	3	6,7,11	315003	112.03
32	4	11,10,7,4	251084	109.73
40	4	5,11,11,13	201759	115.67
48	4	6,14,11,17	179488	118.54
56	4	22,11,15,8	164588	113.21
64	4	22,16,6,20	157151	107.90
p34392				
	Proposed Hierarchical Method			
W	B	TAM Partitions	Test Time (in clock cycles)	CPU Time (in sec.)
16	3	9,6,1	1129320	33.29
24	3	10,19,4	770310	33.17
32	4	15,5,11,1	630934	34.26
40	4	6,2,16,16	606261	32.82
48	3	6,25,17	593924	39.35
56	3	17,31,8	581588	42.07
64	3	20,10,34	581588	38.74
p93791				
	Proposed Hierarchical Method			
W	B	TAM Partitions	Test Time (in clock cycles)	CPU Time (in sec.)
16	2	9,7	1788940	111.55
24	2	16,8	1205333	102.12
32	2	23,9	897921	100.58
40	2	16,24	743066	99.57
48	3	13,12,23	622363	95.11
56	2	47,9	532028	85.62
64	2	48,16	470386	96.07

Table 5.5: Overall test time results for the interactive design scenario

due to the hierarchical core 1. For p34392 lower bound on test time is reached after $W > 48$. This is again mainly due to the bottleneck core 18. For p93791 test times are decreasing significantly with the increase in TAM widths.

In this case also, we compare our test time results with the similar other approach [185]. It is seen from Table 5.6 that maximum improvements are obtained for the SOC p22810. For this maximum improvement 34.23% is obtained. For p34392 our test time results are larger than the test times reported in [185] for

p22810			
W	Our	[185]	% improvement
16	458255	530788	13.66
24	315003	343942	8.41
32	251084	288273	12.90
40	201759	263624	23.47
48	179488	251299	28.58
56	164588	238974	31.13
64	157151	238974	34.23
p34392			
W	Our	[185]	% improvement
16	1129320	1154719	2.19
24	770310	774221	0.50
32	630934	606261	-0.04
40	606261	593924	-2.03
48	593924	581588	-2.07
56	581588	581588	0
64	581588	581588	0
p93791			
W	Our	[185]	% improvement
16	1788940	1854566	3.54
24	1205333	1272220	5.26
32	897921	940318	4.51
40	743066	765715	2.96
48	622363	640488	2.83
56	532028	551849	3.59
64	470386	473726	0.70

Table 5.6: Comparison of test time results with previously proposed method (interactive design scenario)

few of the TAM widths. But for all other cases including p93791 our proposed method provides better results than [185].

5.6 Conclusion

This work addressed the issues of wrapper design for the hierarchical cores.The wrapper architecture is IEEE 1500 compatible. Besides the wrapper design the work also addressed the TAM optimization and test scheduling for hierarchical SOCs in two design scenarios. The use of GA for daisy chaining of CTAMs results in better grouping of them. The GA used for overall test scheduling also explores the search space well. The combined effect of these two produces better results than the previously reported works in this domain.

Chapter 6

Test Data Compression with Compression and Scan-Power Trade-off using Huffman Coding

6.1 Introduction

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)* [83]. Thus it is always a bottleneck.

Again power dissipation is an important factor in today's chip design. Power dissipation in a CMOS circuit is directly proportional to the switching activity in the circuit [149]. As huge number of don't cares are present in the test pattern set, these provide an opportunity for both increased compression and power reduction. As we shall see in the later part of this chapter, these two goals are contradictory to each other, and a suitable trade-off is necessary to achieve a balance between the two.

6.1.1 Contribution and Chapter Organization

In this chapter, we present a compression/decompression technique to reduce test data volume and test application time. Our work improves upon the scheme proposed in [6] to utilize a Huffman coded dictionary. Our proposed technique is advantageous for the following reasons:

a) It has a better compression ratio compared to [6] and other recently proposed techniques, for most of the benchmarks. Hence memory requirement is significantly reduced.

b) The number of tester channels between the ATE and the SOC is one. There is no need for additional synchronization and handshaking signal between the SOC and ATE.

c) Decompression can be flexibly conducted by the SOC processor or on-chip circuits. Area overhead is small compared to [6].

Our approach, on an average results in upto 17% improvement in the compression over [6], 8% more than VIHC [7] coding, 25% more than Golomb [90] coding, 13% more than RL-Huffman [98] coding, 14% more than the Selective Huffman coding (SHC) [84], 4% more than V9C coding [87], 9.5% more than MDC [129] coding, 13.8% more than arithmetic coding [103], and 6% more than data independent pattern run-length (PR) [98] coding. With increasing dictionary size, the scheme can result in further improvement in compression ratio without sacrificing of the advantages of dictionary based approaches like reduced pin count test and narrow interface between the system and environment [13]. The associated decoding hardware has also been presented along with area overhead. We have also performed a trade-off between power and compression ratio by reverting back some of the code words to their original form (due to the presence of unspecified bits in the test cubes). This trade-off can provide test engineers a tool to take a decision on how much compression can be used for a circuit with limited power budget. A test bus power reduction scheme has also been presented during transmission of coded words from ATE to the on-chip decoder.

The rest of the chapter is organized as follows. Section 6.2 discusses the proposed compression technique. Decompression architecture for Huffman coded words is discussed in Section 6.3. Section 6.4 provides the tradeoff between compression and scan power. Test bus power reduction techniques are discussed in Section 6.5. Finally, Section 6.6 draws the conclusion.

6.2 Compression Scheme

In this section we present the compression scheme proposed in this work. For the sake of discussion, we assume that the original test set is T_d, and the number of scan-chains to which the patterns are fed is m. The whole process of compression can now be divided into two phases. In Phase-I, we identify the set of m-bit words from the test set T_d. Once the words have been determined, the next task is to select the set of words to construct the dictionary. For this purpose, first, the set of compatible words are determined. Two words are said to be compatible if they differ only in don't care positions. That is, for each position in the range 1 to m either, $u_i = v_i$ or $u_i = X$ or $v_i = X$.

To identify all such sets of compatible words, we form a compatibility graph G, in which each node corresponds to a unique word. Two nodes are connected via an undirected edge if and only if the patterns corresponding to the nodes are compatible to each other. Next, a clique-partitioning algorithm similar to that in [6] is used to identify the sets of compatible words. One word from each clique represents all patterns belonging to it. Total number of occurrences of all the words in a clique gives the frequency of the word representing the clique. These information are passed on to Phase-II, which generates the Huffman codes. The idea behind identifying such compatible words is that they can be represented by a single dictionary entry. If the number of dictionary entries be $|D|$, the larger sized $|D|$ cliques are selected to be represented in the dictionary.

6.2.1 Phase-I

Let the pre-computed test data set T_d be consisting of n test patterns $p_1, p_2, , p_n$ and the number of scan chains be m. The scan elements of the circuit are assumed to have been divided into m-scan chains in as balanced a manner as possible. Thus each pattern consists of m-sub-vectors. If some of the vectors are short, we pad extra don't care bits at the end to make it balanced so that all the sub-vectors have same length, denoted by l. We construct m-bit words by taking each bit from the same bit position of m-sub-vectors. Hence a total of nl number of m-bit words are formed. Fig. 6.1 illustrates the construction of nl number of words for multiple scan chains. The following Table 6.1(a) is obtained from Figure 6.1. As this table contains small number of words and the length of the scan chain (or word) is also small, we have taken another bigger example of a case in which a larger amount of test data has been divided into multiple scan chains, as shown in Table 6.1(b).

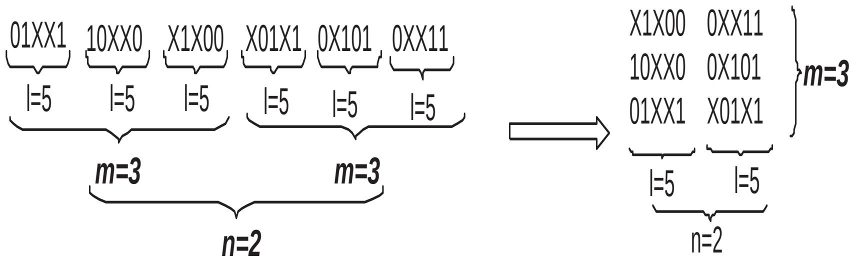


Figure 6.1: Construction of multiple scan chain

(a) Test data of Fig. 6.1 divided into multiple scan chains

Scan Chain Index	Word index									
	10	9	8	7	6	5	4	3	2	1
1	X	1	X	0	0	0	X	X	1	1
2	1	0	X	X	0	0	X	1	0	1
3	0	1	X	X	1	X	0	X	X	1

(b) A larger set of test data divided into multiple scan chains[6]

Scan Chain Index	Word index															
	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	1	1	0	1	1	1	1	0	0	0	X	X	0	X	0	1
2	X	0	1	1	0	0	1	X	X	X	1	X	X	0	1	0
3	X	X	X	X	0	X	0	1	0	0	1	1	0	X	X	X
4	X	0	X	0	X	X	0	X	0	0	0	0	0	X	0	1
5	0	0	0	0	X	0	X	0	X	X	X	0	X	1	0	X
6	0	X	1	0	1	0	X	X	1	X	0	0	X	0	X	X
7	1	0	1	X	X	X	X	1	1	0	X	1	0	0	1	0
8	1	X	0	X	0	1	X	1	0	X	X	X	X	X	X	1

Table 6.1: A set of test data after division of multiple scan chains

Figure 6.2 indicates the graph constructed after mapping the words present in the table 6.1(b) according to their compatibility. If a dictionary of size $|D| = 4$ is to be formed, according to the clique partitioning algorithm, we obtain 4 cliques: $\{5, 6, 13, 16\}, \{2, 8, 14\}, \{3, 4, 7\}, \text{and}\{1, 11\}$. Hence, the entries in the dictionary are $\{11100011, 01000110, 0000100X, 10X10001\}$. The frequency of each entry will be the number of words covered by each clique. The words in the dictionary entries are basically the mapped words obtained from Table 6.1. The information about the dictionary entries and their corresponding frequency are passed on to Phase-II for generating Huffman codes. It may be noted that the words that could not be accommodated in the dictionary are shifted directly to the scan chains as explained in Section 6.3.

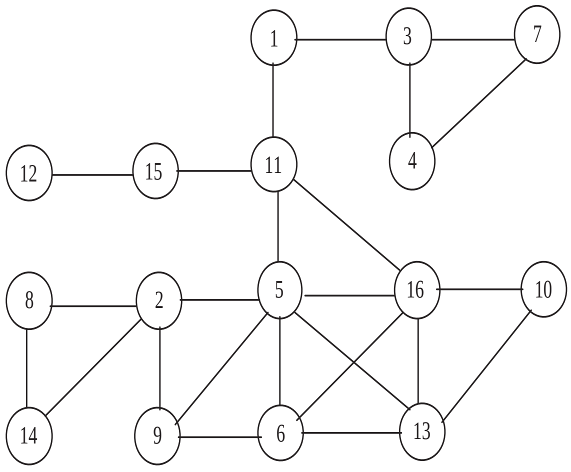


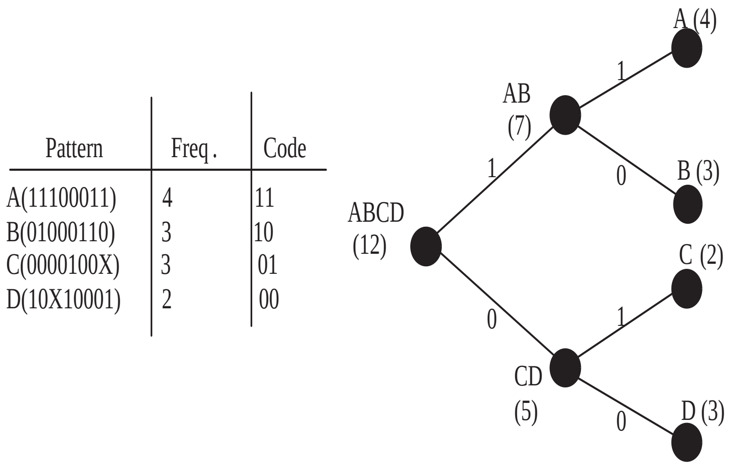
Figure 6.2: Graph constructed for Table 6.1(b) as given in [6]

6.2.2 Phase-II

The set of dictionary words and their frequency are now utilized to generate Huffman code for the dictionary words. It may be noted that Huffman coding is a statistical data coding method that assigns lesser length code to the words with higher frequency of occurrences, thus resulting in good compression of the input data set. Figure 6.3 illustrates the Huffman coding of the words in our example according to the frequencies obtained from Phase-I. First of all, the words are to be ordered according to their increasing frequency. Then, two words with the lowest number of occurrences are combined to form a new word with the frequency of occurrence given by the sum of the occurrences of these two words. This step is repeated until all the nodes are merged into a single node. Hence this will make a Huffman tree. The Huffman codes are obtained by assigning binary '0's and '1's to each segment starting from the root of the tree. Using the Huffman codes we can compress the initial test vector set. For the given example, size of the compressed data is $4 \times 3 + 3 \times 3 + 3 \times 3 + 2 \times 3 + 4 \times 9 = 72$ bits, which corresponds to a compression of 43.75%. Here, an additional bit is needed as prefix to represent whether the word is a dictionary entry or not.

6.2.3 Experimental Results

In this section, we present our compression results and compare with other similar works reported in the literature. The proposed compression scheme has been implemented in C language on a Pentium IV, 2.80 GHz machine with 512 MB main



Pattern	Freq.	Code
A(11100011)	4	11
B(01000110)	3	10
C(0000100X)	3	01
D(10X10001)	2	00

Figure 6.3: Huffman tree and corresponding code

memory. The experimentation has been carried out with the largest ISCAS89 full-scan benchmark circuits [186]. The test patterns have been generated using Mintest ATPG program [187].

Circuits	Compression ratio for different m values											
	m=16		m=32		m=48		m=64		m=128		m=200	
	Fix.	HCD	Fix.	HCD	Fix.	HCD	Fix.	HCD	Fix.	HCD	Fix.	HCD
s5378	45.27	61.45	67.20	71.15	72.33	73.43	73.28	73.30	62.97	61.48	45.39	42.99
s9234	46.04	61.38	65.49	68.89	65.22	66.66	67.45	68.44	70.72	70.39	57.15	56.46
s35932	38.97	46.30	68.73	71.54	58.94	60.63	73.75	74.89	88.93	89.33	95.03	96.28
s38417	43.96	53.30	49.00	51.14	61.79	64.53	42.73	43.18	36.75	36.82	30.65	30.10
s38584	45.78	66.07	64.26	69.17	67.98	71.78	70.86	72.90	70.77	71.43	73.23	70.06
s15850	47.40	69.35	67.57	76.33	74.40	78.82	75.87	78.91	81.98	82.87	80.72	81.05
s13207	49.22	80.13	73.40	86.83	81.09	89.84	85.42	91.26	91.53	94.16	94.84	96.20

Table 6.2: Compression ratio for the proposed scheme with varying scan chains for $|D| = 128$

Table 6.2 presents the results for various number of scan chains(m) using a fixed dictionary size of 128 words. The column marked "Fix." refers to the work [6], in which the dictionary words were coded using fixed length indices. The column marked 'HCD' represents the proposed work. It can be observed that for lower values of m (e.g. $m = 16$), $15 - 20\%$ more compression have been achieved in our scheme as compared to [6]. This is because of the fact that as m increases, the number of possible words also increases. Thus keeping the dictionary size fixed at 128, we have to lose many sets of words that have been found in Phase-I of the algorithm which could be merged. However, a small dictionary size has prohibited us from accommodating them in the dictionary in Phase-II

Table 6.3 shows the case for $m = 48$ with varying dictionary sizes. It shows that as the number of dictionary entries increases, the compression ratio also

Circuits	Compression ratios for $m = 48$		
	$\|D\| = 128$	$\|D\| = 256$	$\|D\| = 512$
s5378	73.43	82.27	82.27
s9234	66.66	78.69	81.31
s35932	60.63	77.74	82.87
s38417	64.53	67.25	72.69
s38584	71.78	74.08	78.55
s15850	78.82	84.82	86.55
s13207	89.84	91.42	91.42

Table 6.3: Compression ratios with varying dictionary entries

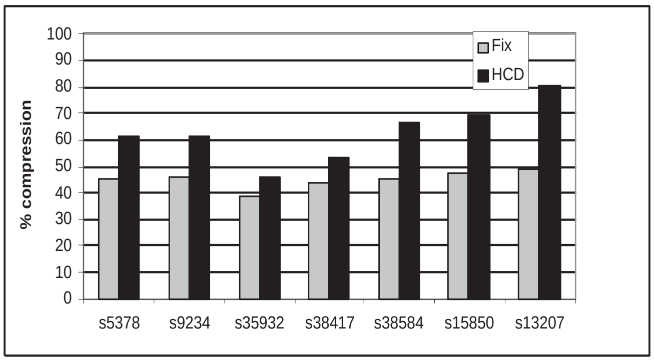
increases due to the inclusion of more and more words in the dictionary. Table 6.4 presents a comparison of the work with other similar strategies proposed in the literature. Results presented in column "HCD" of Table 6.4 gives the maximum % compression obtained for any m value as given in Table 6.2. In column "Fix[91]" results are presented for best compression % obtained for number of scan chains m for fixed length dictionary based approach. Column "VIHC[7]" shows the results of VIHC scheme for maximum compression % using different values for the parameter m_h (the maximum allowable runs of 0's allowed in any block). The best compression obtained using Golomb coding [90] has been included in the table for comparison. Column "RL-Huff[93]" shows the results of maximum compression % reported for parameter K (maximum run length block size) using RL-Huffman coding scheme. Column "SHC[84]" is for Selective Huffman Coding. For this also the best compression results have been compared. Column "V9C[87]" represents the maximum compression % for dictionary based V9C coding scheme for different L bit vectors. The best compression for MDC [129] is presented for different buffer organization. Column "Arith[103]" provides the maximum compression % using 4 bit arithmetic coding and best compression results are provided for pattern run-length (PR-1) [98] scheme using $t = 5$ bit binary expansion with $k(\leq 2^t - 1)$ bit pattern run-length. Average improvements are calculated by excluding the results of s35932 due to the non-availability of the compression results for most of the published schemes. On an average, our scheme results in compression 0.78% more than [6], 7.8% more than VIHC [7] coding, 24.97% more than Golomb [90] coding, 13.76% more than RL-Huffman [93] coding, 14.16% more than the selective Huffman coding (SHC) [84], 4.29% more than V9C coding [87], 9.5% more than MDC [129] coding, 13.8% more than arithmetic coding [103], and 6% more than data independent pattern run-length (PR) [98] coding.

Fig. 6.4(a) and 6.4(b) show the compression ratios for different ISCAS89 benchmark circuits for number of scan chains $m = 16$ and $m = 64$ respectively

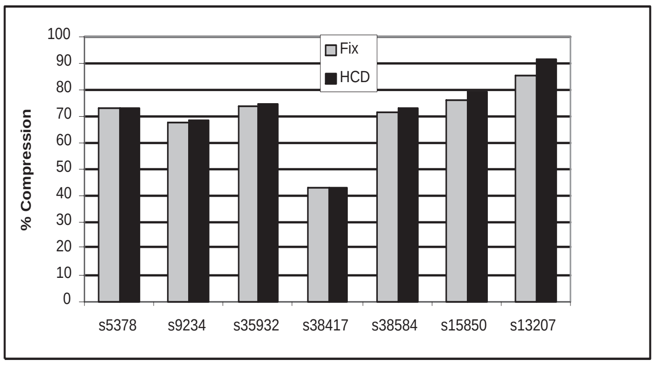
[*]s35932 is not included during average improvement calculation

Circuits	Compression ratio for different m values									
	HCD	Fix. [91]	VIHC [7]	GOLOMB [90]	RL-Huff [93]	SHC [84]	V9C [87]	MDC [129]	Arith [103]	PR-1 [98]
s5378	73.43	73.29	60.73	37.11	53.75	55.10	63.24	56.20	54.00	60.53
s9234	70.40	70.72	60.96	45.25	47.59	54.20	61.02	54.70	58.00	61.46
s35932	96.28	95.03	71.91	N/A	89.26	N/A	N/A	76.10	N/A	N/A
s38417	64.53	61.79	66.38	28.38	64.17	59.00	71.39	61.80	56.00	58.72
s38584	72.90	73.23	66.29	57.17	62.40	64.10	73.77	71.20	57.00	75.45
s15850	82.87	81.97	72.34	62.83	67.34	66.00	75.94	72.80	68.00	75.53
s13207	96.20	94.84	86.83	79.75	82.51	77.00	89.22	86.50	84.00	88.66
Avg.(%)*	**76.72**	**75.94**	**68.92**	**51.75**	**62.96**	**62.56**	**72.43**	**67.2**	**62.83**	**70.06**

Table 6.4: Comparison of compression values with other schemes



(a) **For** $m = 16$



(b) **For** $m = 64$

Figure 6.4: (a)Comparison of compression ratios for various circuits for $m = 16$ (b) Comparison of compression ratios for various circuits for $m = 64$

for both of the methods Fixed length dictionaries [6] and proposed HCD. From these figures it is clear that, HCD provides good compression for lower number of scan chains, this is because as the number of scan chains increases, word size increases, as a result of this, number of words covered by cliques decreases (that is, frequency of the pattern decreases). This has been shown in Table 6.5 for the s5378 circuit. Consequently the Huffman code representing the patterns need more bits, thereby reducing the compression for larger number of scan chains.

m	**Top ten clique sizes**
16	335,305,113,103,96,45,33,27,23,21
32	69,62,45,43,33,22,20,20,19,17
48	26,24,23,22,17,17,17,13,12,11
64	15,15,12,11,11,10,10,10,9,9
128	4,4,3,3,3,2,2,2,2,2
200	4,4,3,3,3,2,2,2,2,2

Table 6.5: Variation of clique sizes with the number of scan chains for s5378

6.3 Decoder design

Figure 6.5 illustrates the decompression architecture. It consists of Huffman decoder (FSM), an m-bit shifter that takes data from $Data_{in}$ when the signal *shift* equals 1 and a $log_2 m$-bit counter used to indicate whether the shifting of a codeword has finished. It operates as follows.

a) The signal *reset* resets the counter to 0.

b) If *inc=1*, the counter is incremented.

c) When the value of the counter reaches m, the output m_{flag} equals 1.

Huffman decoder, which is basically an FSM, is used to decode the dictionary entries. Number of states in the FSM depends on the number of entries (i.e., number of different patterns in the Huffman coded dictionary). The FSM starts by checking the first bit of the data shifted from $Data_{in}$. If this bit is 0, implying that, the m-bits directly constitute a word. The FSM shifts this word into m-bit shifter by enabling the *shift* for m-clock cycles, which then feeds this word into the scan chains during the clock cycle in which the decoder checks the first bit of the next codeword. On the other hand, if the first bit is 1, the next few bits constitute the Huffman code corresponding to an entry of the Huffman coded dictionary. The FSM starts from the initial state and continues the execution until once again it reaches the initial state. After reaching the initial state the *Select* line will be enabled (i.e., decompression for that code is now complete). During these transitions, the signal *shift* will be disabled. The output of the FSM consists of m-bit word, which will be applied to the scan chains.

The state transition diagram in Fig. 6.6 shows the FSM for our example taken in Fig. 6.3. As the number of entries into the dictionary is $|D| = 4$, total number of states will be three. Each time a valid Huffman code is detected, the state will be S0 giving the actual test pattern for which the Huffman coded word has been decoded, together with whether it is *valid* or not. If it is valid then the valid signal is 1, else it is 0.

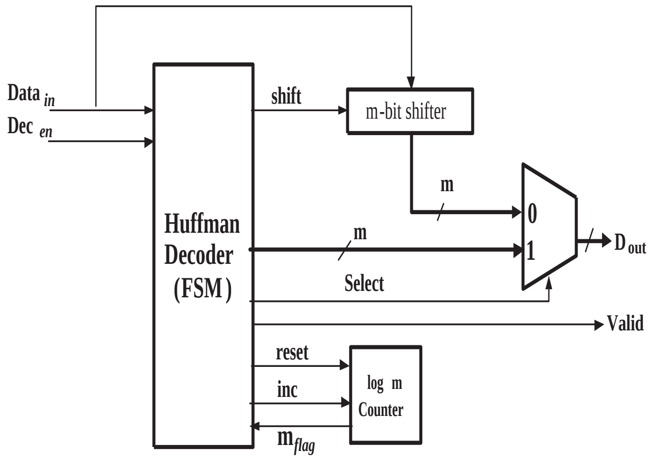


Figure 6.5: Decompression Architecture

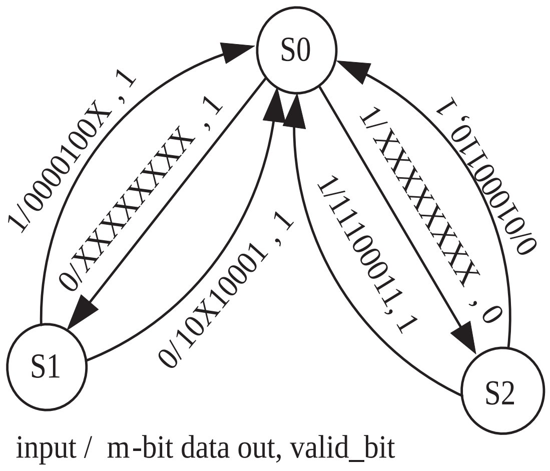


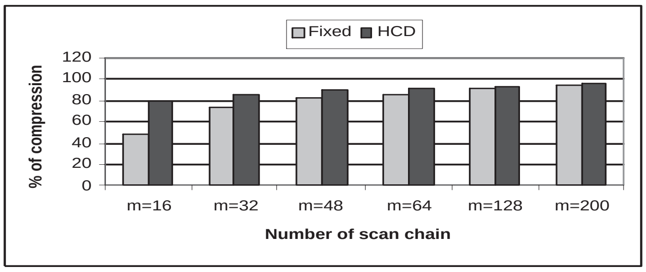
Figure 6.6: State transition diagram of Huffman decoder for $|D| = 4$

Circuits	% of Compression and No. of cells for the HCD FSM for different m values											
	m=16		m=32		m=48		m=64		m=128		m=200	
	Comp	**Gates**	**Comp**	**Gates**	**Comp**	**Gates**	**Comp**	**Gates**	**Comp**	**Gates**	**Comp**	**Gates**
s5378	61.45	555	71.15	740	73.43	787	73.30	874	61.48	928	42.99	928
s9234	61.38	579	68.89	766	66.66	884	68.44	951	70.39	1117	56.46	1117
s35932	46.30	527	71.54	627	60.63	638	74.89	689	89.33	743	96.28	590
s38417	53.30	646	51.14	885	64.53	834	43.18	994	36.28	1175	30.10	1391
s38584	66.07	582	69.17	851	71.78	1051	72.90	1182	71.43	1455	70.06	1572
s15850	69.35	539	76.33	708	78.82	855	78.91	969	82.87	1251	81.05	1274
s13207	80.13	554	86.83	768	89.84	929	91.26	1066	94.16	1206	96.20	1206

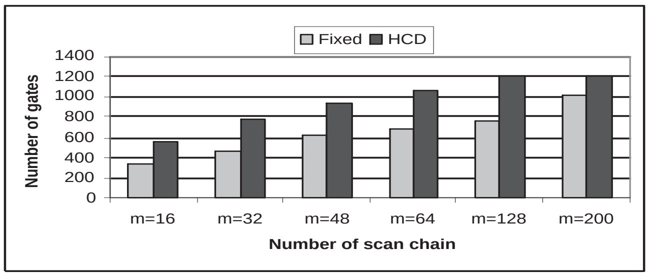
Table 6.6: Compression and area overhead for ISCAS89 benchmark circuits for various numbers of scan chains

6.3.1 Experimental Results

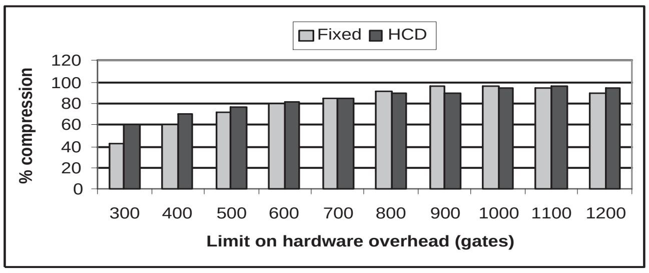
Table 6.6 gives the area overhead and % compression for ISCAS 89 benchmark circuits. Decoder architecture has been synthesized in *Synopsys Design Com-*



(a)



(b)



(c)

Figure 6.7: (a) Compression ratio for different number of scan chains(m) for s13207, (b) Area over head for the corresponding compression ratios For $|D| = 128$ with different scan chain for s13207. (c) Compression versus limit on the hardware overhead for circuit s13207

piler with *Xlite* core library and columns "Gates" represent the number of gates required to implement the FSM of the decoder. Fig. 6.7(a) shows the percentage compression versus the number of scan chains for s13207 for dictionary size $|D| = 128$. Fig. 6.7(b) shows the hardware overhead to implement the corresponding dictionaries. The proposed method for Huffman coding provides higher compression than the fixed length dictionaries proposed in [6]. However, the difference in the compression for the two methods becomes almost indistinguishable for large values of m. The finite state machine decoder for HCD requires higher area overhead compared to fixed length dictionary [6], but HCD provides a high compression at lower number of scan chains with a little bit higher area overhead compared to fixed length dictionary [6]. It is clear from Fig. 6.7(b), that for $m = 16$, HCD produces a compression ratio of 31% more than Fixed length dictionary [6], with an extra area over head of approximately 200 cells. In order to compare the methods accurately, we have taken the compression ratios and corresponding area overheads for different values of m and have drawn a graph between area overhead (number of gates) and compression as shown in Fig. 6.7(c). It is clear from this figure that for lower limit values of area overheads, HCD produces better compression than fixed length dictionaries of [6], and at larger area overhead limits, both the methods produce almost same compression. As area overhead is proportional to the number scan chains (from Fig. 6.7(b)), for lower number of scan chains HCD produces a good compression.

The technique that we have discussed so far provides a Huffman coded dictionary for test data compression. In the following section we will discuss on strategies to reduce the power consumption by reducing the number of transitions occurring while shifting in the coded words.

6.4 Compression Power Trade-off

In test data set, a large number of don't care bits exist. As noted in Section 6.2, test data compression is mainly achieved by filling the unspecified bits in such a manner that the number of similar words in the test set increases. The following discussion will show that this filling may not be advisable from power reduction view point and needs to be revisited. The approach suggested here is applicable to any dictionary based compression technique.

The coding scheme used for the compression specifies don't cares in an efficient manner to get high compression ratio. In other words, the assignment of don't cares should skew the occurrence frequencies of the coded words. Hence affect the compression ratio. But it is seen that coding scheme used in compression techniques may eventually increase the test power. It might not always be an advantage to use a powerful compression scheme if it increases test power significantly. It is necessary to look at the two aspects simultaneously and have some strategy, which would provide a balance between the test data compression and test power. In this work, we have demonstrated one such idea. When these compression techniques are used for a test vector set, some of the don't cares in the test vectors are set to '0' or '1' during the encoding process. If no compression technique is used, these don't cares could have been put to '0' or '1' so as to give minimum number of transitions in the test vector, ensuring minimum possible test power. But when a compression technique is used, this kind of reduction in flip count is not possible, as don't cares will already be set by the compression algorithm. Consider that some don't cares in a test vector are set to '0' or '1' by a dictionary based coding scheme. If we put back some of the don't cares in this test data and replace them with '0' or '1' so as to give least possible flip count, it would have two effects. Firstly, the compression ratio would be reduced as some of the dictionary words will be reduced in frequency. Secondly, it will reduce the flip count of the test vectors, thereby reducing test power. If this replacement of don't cares (and placement of '0' or '1' to give minimum transition count) is done for one vector at a time, we have a reduction in test power but also a reduction in compression at each stage. Thus, we can achieve a trade-off between test power and test data compression by doing this, for a given compression technique. Each point in this trade-off has a particular value of compression ratio and test power. By utilizing this trade-off analysis, one can arrive at a compression scheme that utilizes the original compression technique to such an extent that gives desired test performance in terms of test power and test application time, given the test equipment and power constraints of the System-on-Chip.

6.4.1 Compression effect on test power

In this section we will show how the coding scheme will affect the test power. Then we will show how don't cares can be specified to reduce test power. So a trade-off between compression and test power can be obtained to provide test engineers a tool to take a decision on how much compression can be used for a circuit with limited power budget.

Example:

Consider the dictionary based compression technique applied on a test vector set T. As an example, consider the test vectors given in the Section 6.2 in Fig. 6.1. $T = 01XX110XX0X1X00, X01X10X1010XX11$. Suppose don't cares in T are each set to '0' or '1' so as to give minimum transition count. If don't cares are specified in such a way that it ensures minimum number of transitions, $T = 011111000011000, 001110110101111$. Now, T will have a weighted transition count [149] of $4 + 11 = 15$. Filling of don't cares and transition count is obtained after dividing the test vector for $m = 3$ scan chains. If the encoding scheme for $m = 3$ scan chain (as explained in Section 6.2 using Fig. 6.1) is applied here, T will give rise to 10, $m = 3$ bit words. The 10 words are $X10, 101, XXX, 0XX, 001, 00X, XX0, X11, 10X, 111$ as shown in Fig. 6.1. After applying the clique partition algorithm on this set the cliques formed are as given in Table 6.7.

Suppose that all of these cliques are present in the dictionary. So, any of the 10 words, on decompression, will be decoded as its representative dictionary pattern. For example, $0XX$ will be decoded as 010. The modified test vectors are, $T_r = 010011011001000, 101110110100111$. T_r has a weighted transition count of $17 + 13 = 30$. Thus, due to the usage of the compression technique, while com-

Pattern	**Frequency**
010	4
101	2
001	2
111	2

Table 6.7: Pattern and their corresponding frequency after clique partition

Pattern	**Frequency**
010	3
101	2
111	2
001	1
110	1
000	1

Table 6.8: Pattern and their corresponding frequency after reverting back to original pattern

pression is achieved, the test vector flip count is increased from 15 to 30. Now, if we put back some of the don't cares in T_r, the flip count will come down, but the compression achieved will also be reduced. For example, from T_r, if we put back all the don't cares which were originally present in the second vector and set them to '0' or '1' so that the flip count is at the minimum possible (without changing the first vector) value, we get $TT = 010011011001000, 001110110101111$. TT has weighted transition count of $17 + 11 = 28$. Now, the 10 words are $010, 101, 010, 010, 001, 000, 110, 111, 101, 111$. New cliques formed are given in Table 6.8. It can be seen that TT will have a lower compression compared to T_r. If we put back the don't cares in the first vector also, we get back to T. T will have lower compression than TT. Thus, by successively putting back the don't cares and placing '0' or '1' that would give least flip count, the compression ratio is also gradually reduced, while the test power is gradually reduced.

6.4.2 Test Data Compression and Test Power - A Trade-off

Consider a test vector set T which has been transformed to set T_r due to application of the dictionary based compression scheme for test data compression. T_r has higher test power than T. Each test vector t in T is transformed to a vector t_r in T_r, which has a (possibly) higher transition count and lower number of don't cares. Now, replace t_r with t in the new set T_r, that is, revert back the changed don't cares in t and then replace them with '0' or '1' to give least transition count for t. This will give rise to a new set of vectors TT. TT will have lower transition count than T_r. But it will possibly have a higher test time than T_r as the words of t_r, which are in the dictionary could be dropped out of their compatible partition due to reverting back the don't cares and placing '0' or '1' to give minimum transition count. Hence we have the following relations:

(a) Test power (T) $\leq$ Test power (TT) $\leq$ Test power (T_r)

(b) Test time (T) $\geq$ Test time (TT) $\geq$ Test time (T_r)

It can be seen that by successively reverting back the don't cares of each vector, we get a trade off between test power and compression, starting at a point of lower compression length (that is, higher compression) but high power to a point of low power but higher compression length (that is, lower compression). At each step, test power would reduce but the compression length would increase. Based on the various requirements and constraints, this trade-off can be used to find the best test vector set (a particular point in the trade-off). Typical criteria could be:

1. The ATE has a maximum bit rate. In this case it is necessary to find the vector set that gives the least power, while giving necessary compression.

2. The SOC could have a maximum power rating that should not be violated while testing. Then it is required to get the best compression vector set that complies to the power constraint.

3. A cost function could be defined as a function of test power and test data compression. It is required to find a vector set that gives the least cost function.

Algorithm 6.1 performs such a trade-off for a given SOC test vector set. The procedure takes as input the original test vector set and the set of words (computed as given in Section 6.2), the modified test vector set and the modified set of words after a dictionary based compression technique is applied, and also the set of words along with their frequency from clique partition algorithm. The procedure would give as output a plot of percentage compression of the test vector data vs. weighted transition count of the vector set (which represents the test power). At each step, the don't cares in one vector are placed back and then replaced with '0' or '1' so as to give minimum number of transitions. At each step, the following are to be computed:

(a) Total compression of the current test vector set, on application of the dictionary based compression.

(b) Total bit transitions during scan-in of the current test vector set. Power consumption is estimated by the weighted transition count metric [149], which is strongly correlated to the switching activity during scan-in operation of the test input patterns. The number of transitions caused by a test vector for a scan-in (ignoring the original contents of scan chain) is given by equation 6.1. $Size\_of\_chain$ is the number of flip-flops in the scan chain. $Position\_of\_Transition$ is indexed from the LSB for the input vectors and for output response vectors it is indexed from MSB.

$$Transitions = \sum(Size\_of\_Chain - Position\_of\_Transition) \qquad (6.1)$$

To compute the compression, set of cliques at each step is required to be computed. As clique partitioning is computationally complex, the cliques at each step are found by updating the set of cliques obtained from previous set. Thus, this procedure does not take long execution time.

The nature of the trade-off plot would depend on the order in which the vectors are selected for reverting the don't cares. In our procedure, the vectors are initially ranked in the decreasing order of the reduction of flip count on reverting the don't cares and then the vectors are replaced in that order. Firstly, such an order for reverting the don't cares would give a good trade-off performance as vectors that would give higher reduction in flip count on reverting the don't cares are selected first. Secondly, arriving at such an order is very simple, and so not much computational overhead is added in the process of ranking the vectors.

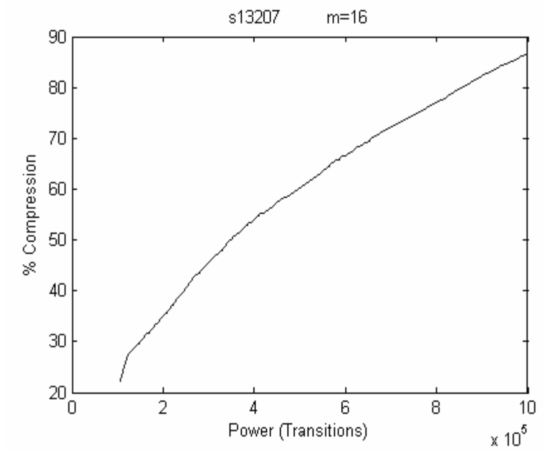


Figure 6.8: Typical trade-off plot between compression and test power

The proposed test power vs compression trade-off has been performed on full scan version of the ISCAS89 benchmark circuits. A typical trade-off for circuit s13207 (for $m = 16$) is shown in Fig. 6.8. X-axis represents the transition counts during scan-in of the test data. Y-axis is representing the % compression (compression ratio) of the test data. Now consider the following cases:

Case 1: Suppose the available test equipment and test application time put a lower limit of 70% on the compression. In the absence of this trade-off, one would

Algorithm 6.1: Compression_power_trade-off

```
Input :
    • Original pattern set $T$ with don't cares.
    • Number of scan chains $m$.
    • Modified pattern set $T_m$ used by compression procedure.
Output: Compression ratio Vs. scan transition plot.
1  begin
2      Let $T_r = T$ with don't cares filled to minimize transitions;
3      Let $C_m$ and $trans_m$ be the compression ratio and transitions for $T_m$
       respectively;
4      Plot $(C_m, trans_m)$;
5      Let $C_r$ and $trans_r$ be the compression ratio and transitions for $T_r$
       respectively;
6      Plot $(C_r, trans_r)$;
7      for $i$= 1 to no_of_vectors do
8          Let $diff[i]$=Transition for $T_m[i]$ -Transitions for $T_r[i]$;
9          Sort $T_m$ into $T_m^s$ on the decreasing order of $diff$ values;
10     end
11     for $i$=1 to no_of_vectors do
12         Replace $T_m^s[i]$with its corresponding vector in $T_r$;
13         $trans_m = trans_m - diff$ corresponding to $T_m^s[i]$;
14         Compute new compression ratio $C_m$;
15         Plot $(C_m, trans_m)$;
16     end
17 end
```

have to use this compression technique to the full extent, which would mean a test power proportional to transitions 10×10^5 in the vectors.

By using this trade-off, however, one can arrive at a conclusion to use the compression scheme to the extent given by the point $(6.6 \times 10^5, 70\%)$, thereby reducing the flip count to 6.6×10^5, while satisfying the compression constraint.

Case 2: Suppose the SOC has the maximum power rating proportional to a transition count of 4×10^5, which should not be violated while testing. In the absence of this trade-off, one would have to choose between using the compression technique to the full extent or not using it at all. In this case, due to the power restrictions, the compression cannot be used, thereby putting the compression at 22.2%. By using this trade-off, however, one can arrive at a conclusion to use the compression scheme to the extent given by the point $(4 \times 10^5, 54\%)$ thereby increasing the compression upto 54%, while satisfying the power constraint.

Case 3: Suppose the SOC designer has defined a cost function in terms of test power and test application time. By using this trade-off, one can locate a point on the plot that would give the transition count and compression length that would result in minimum cost function. Hence the proposed trade-off provides flexibility to choose compression ratio within power budget.

Next, we present the results of compression vs. scan transition trade-off.

6.4.3 Experimental Results

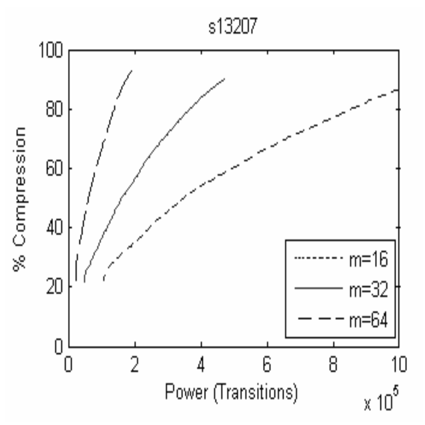
Achieved trade-off plots for the benchmark circuits are shown in Fig. 6.9. It can be observed from the graphs that more trade-off exists for lower number of scan chains. This is because with increasing number of scan chains, the length of individual chains decreases significantly. So number of transitions contributed by these smaller sized scan chains is less. A transition traverses a shorter distance through scan flip-flops since the length of the chain is small. Thus, for all the circuits, good trade-off can be observed for $m = 16$ scan chains. The trade-off reduces as the number of scan chains is increased to 32 or 64.

To make the analysis complete, we have also included the transitions while scanning out the test responses. The output of the circuit for a particular input test vector is obtained using the FSIM simulator [188]. The test data set at each point in the trade-off plot is known. So using FSIM simulator, output response for each of the vectors can easily be obtained. Then the new trade-off plots are made by plotting total transition count (scan-in transitions + scan-out transitions) using Eqn. (6.1). To get scan-out transition count, position of transition is started from MSB of the output responses. The trade-off plots for the circuits s5378 and s13207 are shown in Fig. 6.10(a) and 6.10(b) respectively.

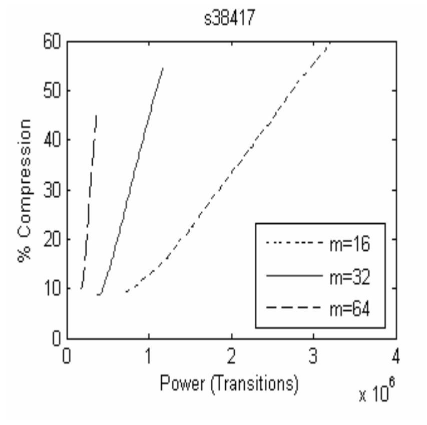
After addressing the issue of scan power reduction, in the next section we consider the power reduction during transmission of test patterns through the ATE channel and TAM lines. We have proposed a modification in the Huffman tree construction procedure to realize power saving.

6.5 Test Bus Power Reduction in Compression

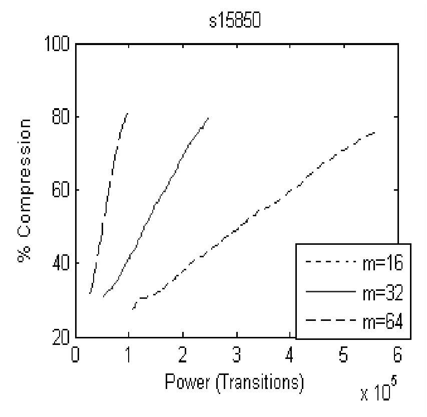
Most of the compression techniques reported in the literature use various strategies to reduce test power [94, 189, 190]. These strategies mostly focus on the reduction of transitions in the scan chains. None of the techniques pay attention to the power consumption in the test bus that carries the test data to the on-chip



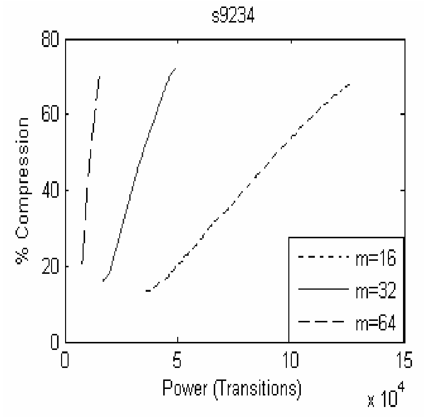
(a)



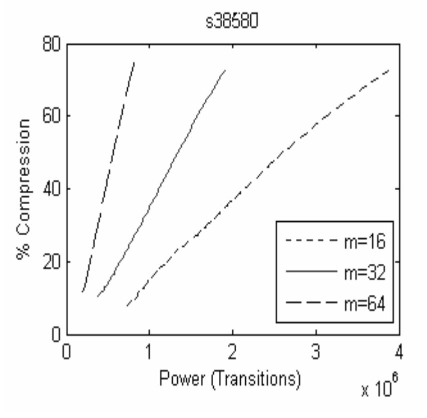
(b)



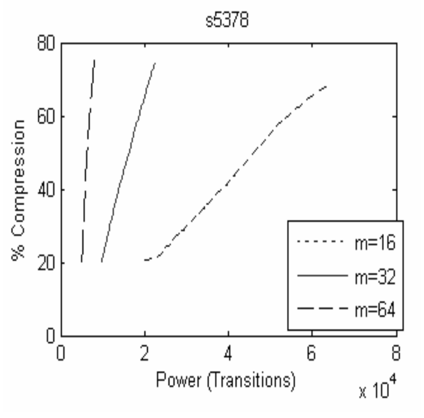
(c)



(d)

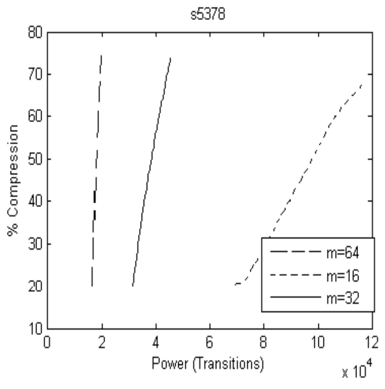


(e)

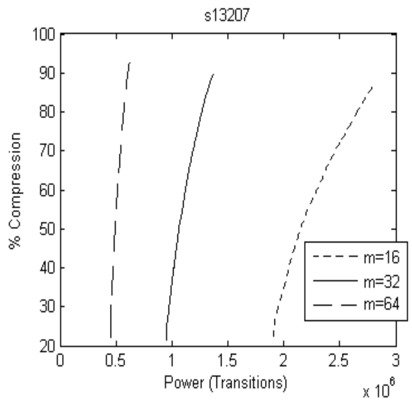


(f)

Figure 6.9: Trade-off results for various ISCAS89 benchmark circuits. (a) for s13207, (b) for s38417, (c) for s15850, (d) for s9234 (e) for s38580 and (f) for s5378



(a)



(b)

Figure 6.10: Trade-off results for (a) s5378 and (b) s13207 including scan-out transitions

decoder for the core and in the decoder itself. However, we feel that there exists quite a few reasons to try out test-bus power reduction as follows:

1. The trend for digital circuits exhibits shrinking of geometries, scaling of supply voltages, increased interconnect density, faster clock rates and higher integration levels. Under this type of situation, wires have been shown to account for a remarkable percentage of the total on-chip power dissipation (upto 40-50%)[191, 192].

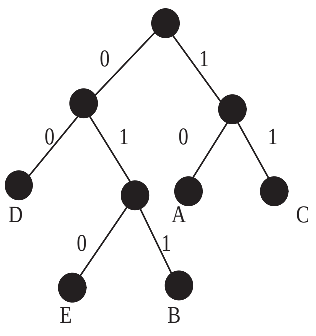
2. To avoid unnecessary power dissipation in the combinational block during the scan cycle, blocking circuitry is often used for gating the scan cell output [135, 193, 194]. However, though in the sub-micron technology interconnects act more as a device on the silicon floor, no such blocking is possible for the test bus.

3. On the other hand, low power bus encoding strategies [195, 196, 197] cannot be used, as it will add to the on-chip decoder complexity.

In the light of the above discussion, in this section, we have presented schemes to reduce the transitions and thus power consumption on the test bus while shifting-in the compressed test patterns. Most importantly, the schemes do not interfere with any power-saving measures taken in the test patterns to reduce scan transitions. It supplements the test power reduction by reducing transitions in the test-bus and decoder input without any extra test-area or test time overhead. Moreover, it also gives an avenue to reduce test-bus power further with a reduced test compression (increased test time) which may be tried out as a trade-off between compression and test bus power. First we modify Huffman coding style to reduce flips in the codes produced. The procedure does not sacrifice the compression ratio, the test application time and area overhead. The scheme has next been extended to perform a trade-off between compression ratio and number of flips in the codes produced. Experimental result shows that without sacrificing compression, test bus transition can be reduced upto 87% for circuit s35932 and on average upto 30% for group size 4. Also a good amount of trade-off can be achieved between the compression ratio and number of bus transitions.

6.5.1 Power Saving Scheme

In this section, we present schemes to modify the Huffman coding style to reduce the switching while transmitting the coded words on the test bus. First, we present an algorithm to label the edges of the tree in an optimal fashion. Next we present a trade-off between the compression and switching activity.



Word	Freq.	Code
A	3	10
B	2	011
C	3	11
D	2	00
E	1	010

Figure 6.11: A Huffman tree

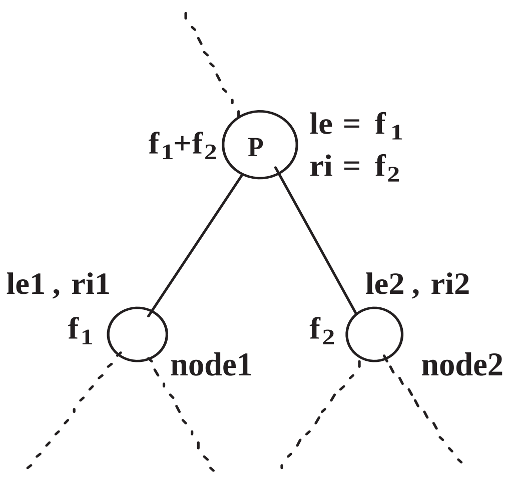


Figure 6.12: Part of the Huffman Tree

Reducing switching activity by 0 and 1 assignment to the branches of the Huffman Tree

Consider an example of Huffman coding as shown in Fig. 6.11. Here A, B, C, D and E are the coded patterns, each of variable length. Let the frequency of each of the coded pattern be as shown in the figure. Using Huffman coding algorithm we obtain the corresponding Huffman tree. Here we trivially assign 0 to the left-branches of the tree and 1 to the right branches of the tree. From the Huffman coded words it is seen that total number of transitions is $3 \times 1 + 2 \times 1 + 1 \times 2 = 7$. As the frequency of A is 3 and only one $1 - 0$ pair exists in the code for A, the total number of transitions for this word A will be $3 \times 1 = 3$. Similarly, for B it will be $2 \times 1 = 2$. For C and D, coded words do not have any $0 - 1$ or $1 - 0$ pair existing in their codes, so these will not contribute to the transition count. For E, frequency is 1 and $0 - 1$ and $1 - 0$ pairs are 2. So E will contribute $1 \times 2 = 2$ transitions. Hence total number of transitions is 7. But we can reduce the number of transitions as follows:

Consider a part of the Huffman tree as shown in Fig.6.12. At the time of constructing the tree, two lowest frequency nodes, say $node1$ and $node2$, have been selected and say the frequencies of these nodes are f_1 and f_2 respectively. The parent of these two nodes is P and its frequency should be $f_1 + f_2$. Now at the time of coding, we have to assign 0 and 1 to the corresponding branches of the tree. The conventional method is to assign 0 to the left branch and 1 to the right branch. So the sequence of 0s and 1s from the root to a leaf node represents the Huffman code of that node.

As the power consumption depends on the number of transitions [149], our intention here is to assign 0 and 1 to the branches of the tree such that transitions are reduced. The assignment of 0 and 1 that we are following here is based on two counts (w_1 and w_2 as used in algorithm 6.2) of nodes. Let,
$le1$= Total frequency of words appearing in the branch labeled 0 from $node1$.
$ri1$=Total frequency of words appearing in the branch labeled 1 from $node1$.

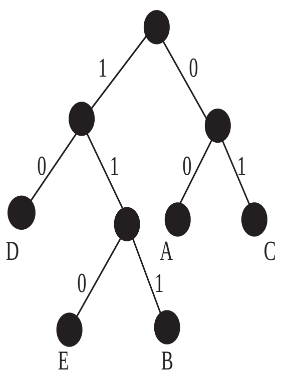
Similarly, $le2$ and $ri2$ are defined for $node2$. Now, if branch $(P, node1)$ is labeled 0 (and thus branch $(P, node2)$ labeled 1), number of 0 to 1 transitions will be $w_1 = ri1 + le2$. This is because for $ri1$ words that are passing through branch $(P, node1)$ there is a transition as their codes will have "01" subsequence for this part of the tree. Similarly, $le2$ words will have "10" subsequence. Again, if branch $(P, node1)$ is labeled 1 (and thus branch $(P, node2)$ labeled 0) number of 1 to 0 transitions will be $w_2 = le1 + ri2$. Hence, to optimize number of transitions, depending on the values of w_1 and w_2, we can label the corresponding branch either as 0 or 1. If $w_1 \leq w_2$, assign the label 0 to the branch $(P, node1)$

else assign 0 to the branch $(P, node2)$. The other branches are assigned 1. This process will be continued until we reach the root of the tree.

Algorithm 6.2: Power_optimized_Huffman_Tree

```
   Input  : A list $W$ of $n$ nodes each has its own frequency.
   Output: A Huffman tree T with assignment of 0 or 1 to each branch.
1  begin
2      Step 1: Create list F from W with increasing order of frequencies;
3      Step 2: Set parameter $le = 0$ and $ri = 0$ for each node in list $F$;
4      Step 3: while (F has more than one element) do
               a. Find two nodes $T1$ and $T2$ in $F$ that has minimum values of frequencies
               b. Remove $T1$ and $T2$ from $F$
               c. Construct a binary tree rooted at $T$ with children $T1$ and $T2$
               d. The frequency of the newly created node $T4$ is sum of the frequency of the
                  nodes $T1$ and $T2$
               e. Let $w_1 = ri(T1) + le(T2)$ and $w_2 = le(T1) + ri(T2)$
               f. if $(w_1 \leq w_2)$ then
                       f.i   Assign 0 to the branch $(T, T1)$ and 1 to $(T, T2)$
                       f.ii  Set $le(T)$ =frequency of $T1$
                       f.iii Set $ri(T)$ =frequency of $T2$
                  end
                  else
                       f.i   Assign 0 to the branch $(T, T2)$ and 1 to $(T, T1)$
                       f.ii  Set $le(T)$ = frequency of $T2$
                       f.iii Set $ri(T)$ = frequency of $T1$
                  end
               g. Insert $T$ to the list $F$ such that the list will be again in increasing order of
                  frequency
5      end
6  end
```

In the given Algorithm of 6.2, $le(T)$ and $ri(T)$ determine the frequencies of the nodes present at the left and right subtrees respectively, considering T as root. For the given example of Fig. 6.11 using the proposed method we compute



Word	Freq .	Code
A	3	00
B	2	111
C	3	01
D	2	10
E	1	110

Figure 6.13: Assignment of 0 and 1 according to algorithm 6.2

the total number flips as $3 \times 1 + 2 \times 1 + 1 \times 1 = 6$. Assignment of 0 and 1 is shown in Fig. 6.13. Lengths of coded words are same as before. Hence compression ratio is not affected.

Theorem 6.1. *Algorithm Power_optimized_Huffman_tree produces a tree resulting in minimum number of transitions for transmitting code words.*

Proof. The proof follows from the fact that at any point of combining two subtrees rooted at two nodes, we are assigning 0/1 to the branches that minimize the transitions for all the patterns passing through the node. Hence by induction on the levels of the tree, we can state that the tree produced is optimum. □

Trade-off between transition count and compression ratio using subtree reordering

Here we are trying to find a trade-off between compression ratio and the transition count by swapping the leaf nodes. Main goal is to further reduce the switching activity, sacrificing the compression ratio to some extent. We consider the swapping of two nodes only when the switching activity is reduced. Suppose, in original Huffman tree, node i has the number of occurrences O_i and node j has the number of occurrences O_j and we are trying to swap these two nodes. Also, let before swapping the length of the code word for i be L_i^1 and for j be L_j^1. Similarly consider the length of the code words for i and j to be L_i^2 and L_j^2 respectively after swapping the nodes i and j. Hence, change in compression length,
$\triangle c = (L_i^2 \times O_i + L_j^2 \times O_j) - (L_i^1 \times O_i + L_j^1 \times O_j)$
$= (L_i^2 - L_i^1) \times O_i + (L_j^2 - L_j^1) \times O_j$.
So, if initial compression length is C, the current compression length will be $C' = (C + \triangle c)$. Similarly for the transition count we can get $F' = (F + \triangle f)$, where $\triangle f = (f_i^2 - f_i^1) \times O_i + (f_j^2 - f_j^1) \times O_j$, f_i^1 and f_i^2 are the transition counts

before and after swapping for node i and f_j^1 and f_j^2 are for node j. Now if we take the weighted sum of both the transition count and compression ratio then we have the relation

$$M = w \times C + (1 - w) \times F \text{ before swapping the nodes } i \text{ and } j \tag{6.2}$$

and

$$M' = w \times C' + (1 - w) \times F' \text{ after swapping the nodes } i \text{ and } j \tag{6.3}$$

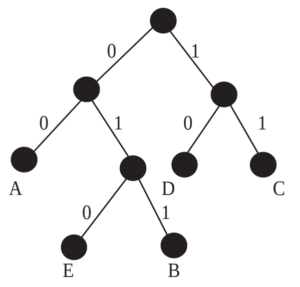
We will accept this swapping for the pair of nodes only when $M' < M$ for a particular value of w, where w varies between 0 and 1. The value of w is influenced by the emphasis on compression and power. For more emphasis on compression, a value close to 1 be selected for w, while to reduce number of transitions in test bus, a low value of w has to be set.

This process will be carried out for every pair of nodes. If the tree contains n number of leaf nodes then for ${}^nC_2$ pair of nodes i and j we check whether the swapping is possible or not. Between all possible swappings, the one giving maximum improvement is taken and the process is repeated till there is no further improvement.

For example, consider Fig. 6.11. Suppose nodes A and D are selected for swapping. The code word length for node A is $L_A^1 = 2$ and for node D is $L_D^1 = 2$. The number of flips present in the code word of A is $f_A^1 = 1$ and for node D is $f_D^1 = 0$. The frequency of these two nodes are 3 and 2 respectively, as shown in Fig. 6.14, which represents the Huffman tree after swapping of the node A with D. The code word lengths for both the nodes A and D are same. Hence $L_A^2 = L_D^2 = 2$. Thus, change in compression length $\triangle c = (2 - 2) \times 3 + (2 - 2) \times 2 = 0$. After swapping, number of flip count in the code word of A is $f_A^2 = 0$ and for node D it is $f_D^2 = 1$. Hence change in flip count $\triangle f = (0 - 1) \times 3 + (1 - 0) \times 2 = -1$. So after swapping of the nodes A and D, overall flip count is reduced by one. For any weight w (excepting $w = 1$), it is obvious that $M' < M$ according to the equations (6.2) and (6.3). So this swapping is accepted. Similarly, for other pair of nodes also this checking has to be done for possibilities of further swapping, depending on the condition discussed above.

Assignment of 0 and 1 to differentiate dictionary and non-dictionary entries

Another avenue of reducing flip-count is the judicial selection of the prefix bit to distinguish between the dictionary and non-dictionary words. According to our compression algorithm as discussed in Section 6.2 some entries are in dictionary, which are the Huffman coded words and some entries are not in the dictionary obtained from clique partitioning algorithm. At the time of testing, the dictionary



Word	Freq .	Code
A	3	00
B	2	011
C	3	11
D	2	10
E	1	010

Figure 6.14: After swapping of node A and node D

```
Algorithm 6.3: Procedure for reordering nodes
   Input  : A Huffman tree $T$
   Output: Tree with reordering of the nodes of Huffman tree $T$
1  begin
2      for (every pair of node from 1 to $|D|$) do
           /* D being the set of nodes in the tree                */
           1. If any node say A or B in the pair (A, B) of nodes is already selected
              previously for reordering then choose another pair of nodes including
              other node for reordering and go to Step 3.
           2. If node A and B in the pair (A, B) both selected previously for reordering
              then choose another pair of nodes for reordering.
           3. Compute $M$ and $M'$;
              if ($M' \geq M$ (according to equation 6.2 and 6.3)) then
                  Reordering not possible;
              end
              else
                  Reorder the node pair and update the compression length C
                  and flip count F;
              end
3      end
4  end
```

entries are decoded, whereas, non-dictionary entries are sent directly. So, decoder should differentiate between the dictionary and non-dictionary entries. For this, prefix bit 0 or 1 is attached before each and every entry. The choice of assigning 0 and 1 can be done depending upon the resulting number of flips in the overall test data. In the experimentation, we have seen that about 95% of reduction in flip-count can be obtained by suitable prefix bit selection.

6.5.2 Experimental results

Change in Internal flip count using Power_Optimized_Huffman_Tree

Table 6.9 shows the number of internal flips for different numbers of scan chains (m). It may be noted that the test vectors are decomposed into m-bit words. For each m-value, the column "Before" notes the number of transitions in the Huffman coded words in which the left branches of the tree are always labeled 0 and the right branches 1. The column "After" notes the number of flips for Huffman tree constructed using *Power_optimized_Huffman_Tree* algorithm proposed in this work. It shows that the number of flips are reduced mostly for $m = 16$ for every circuit. The row marked "Avg. imp." shows the average improvement in power consumption. The highest improvement of 3.74% occurs for circuit s35932 with $m = 48$ scan chains. Though the average improvement figures are small, considering that these are optimal values and that to get overall power, the number of flips is going to be multiplied by the square of supply voltage, load capacitance and frequency, it will result in good amount of power saving.

Circuit	**m=16**		**m=32**		**m=48**		**m=64**		**m=128**		**m=200**	
	Before	After	Before	After	Before	After	Before	After	Before	After	Before	After
s5378	2432	2376	1621	1603	1264	1256	1060	1032	466	458	466	458
s9234	4340	4192	2556	2506	1935	1863	1263	1249	668	655	668	655
s35932	3249	3185	2094	2056	1043	1004	976	970	590	569	370	364
s38417	19511	19272	8518	8328	5495	5362	3444	3416	1450	1424	953	936
s38584	13359	13123	8849	8524	6811	6545	5554	5389	3120	3073	2139	2086
s15850	5948	5370	3483	3439	2571	2540	2010	1935	1165	1157	1023	992
s13207	5244	4940	3895	3481	3186	2801	2798	2557	1872	1850	1603	1568
Avg. Impr.(in %)	3.74		3.23		2.40		2.93		1.67		2.11	

Table 6.9: Change in internal flip count using proper assignment of 0 and 1 to Huffman tree

Trade off between compression length and flip count by subtree reordering

Table 6.10 shows the result of trade-off between the flip count and the overall compression ratio. As discussed in Section 6.5, it is possible to reduce the switching activity by swapping the leaf nodes. But in this case compression length may increase. It is noted from Table 6.11 that, on an average $10 - 25\%$ flip count can be reduced by sacrificing on an average 1% compression ratio. Maximum reduction of the flip count can be obtained as 26.38% by sacrificing only 1.91% compression ratio for the circuit s38417 with $m = 48$. Graphs presented in Fig. 6.15 holds the same idea as in Table 6.11. To get an idea about what can be a good choice for w, for each circuit and each m-value, we have normalized the compression length and flip count values by dividing those by the corresponding maximum in that group. For example, for s5378 and $m = 16$, the maximum compression length is 10418 and maximum flip count is 2551. All entries in that group corresponding to different w-values are divided by these numbers. These

Circuit	Wt.	**Trade off between compression length and flip count by subtree reordering**											
		m=16		**m=32**		**m=48**		**m=64**		**m=128**		**m=200**	
		Comp	Flip	Comp	Flip	Comp	Flip	Comp	Flip	Comp	Flip	Comp	Flip
s5378	0.0	10418	2136	7572	1512	6602	1201	6560	981	9151	457	13543	457
	0.2	9755	2041	7192	1419	6463	1127	6531	921	9151	456	13543	456
	0.4	9755	2041	7192	1419	6463	1127	6531	921	9151	456	13543	456
	0.6	9649	2035	7065	1414	6463	1127	6531	922	9151	456	13543	456
	0.8	9191	2272	6934	1488	6353	1179	6407	941	9151	456	13543	456
	1.0	9157	2551	6853	1608	6312	1269	6348	1045	9149	460	13541	460
s9234	0.0	17803	3863	13308	2348	14506	1771	12703	1158	11806	628	17278	628
	0.2	15862	3788	12638	2196	13644	1520	12527	1139	11764	597	17236	597
	0.4	15862	3788	12638	2196	13644	1520	12527	1139	11764	597	17236	597
	0.6	15164	3827	12638	2196	13644	1520	12527	1139	11764	597	17236	597
	0.8	15253	3979	12330	2347	13243	1675	12527	1139	11764	597	17236	597
	1.0	15167	4408	12216	2631	13092	1889	12395	1279	11629	674	17107	668
s35932	0.0	18754	3074	8738	1993	11379	942	7482	944	3177	558	1141	351
	0.2	16153	2936	8626	1791	11284	908	7395	852	3082	524	1083	333
	0.4	15558	2960	8626	1791	11284	908	7395	852	3082	524	1083	333
	0.6	15193	3096	8492	1742	11284	908	7395	852	3082	524	1063	342
	0.8	15199	3061	8055	1982	11141	945	7196	846	3020	554	1053	354
	1.0	15147	3281	8028	2099	11105	1028	7084	974	3009	565	1048	375
s38417	0.0	82596	16872	81094	7915	60538	4737	94062	3156	104296	1355	115213	891
	0.2	87896	15519	81905	7604	61572	4045	94716	3061	104266	1223	115204	902
	0.4	87896	15519	81905	7604	61572	4045	94716	3061	104266	1223	115204	902
	0.6	82582	15336	81905	7604	61572	4045	94716	3061	104266	1223	115204	902
	0.8	77284	18162	80811	7920	59061	4699	94716	3061	104266	1223	115204	902
	1.0	76938	19489	80489	8335	58436	5374	93596	3436	104076	1424	115158	939
s38584	0.0	98657	13013	82746	8085	57034	5764	54963	5177	58942	2870	59759	2000
	0.2	73984	10821	62565	7351	57436	5295	55124	4849	57472	2729	60143	1801
	0.4	73984	10821	62565	7351	57436	5295	55124	4849	57472	2729	60143	1801
	0.6	71065	11215	62565	7351	57436	5295	55124	4849	57472	2729	60143	1801
	0.8	67695	12899	62009	7742	57436	5295	54726	4973	57472	2729	60143	1801
	1.0	67557	13301	61387	8496	56179	6624	53967	5541	56883	3076	59608	2091
s15850	0.0	33767	5283	29344	3379	17438	2114	16801	1834	13401	1087	14988	964
	0.2	26034	4578	18793	3045	16731	2068	16503	1739	13316	1050	14659	915
	0.4	26034	4578	18793	3045	16731	2068	16503	1739	13316	1050	14659	915
	0.6	24751	4751	18793	3045	16731	2068	16503	1739	13316	1050	14659	915
	0.8	23644	5278	18290	3292	16378	2423	16423	1764	13314	1038	14659	915
	1.0	23597	5493	18226	3434	16303	2545	16233	1949	13185	1194	14592	1015
s13207	0.0	72879	4671	43467	3304	29388	2614	15593	2251	10907	1697	10363	1517
	0.2	34776	4231	22215	3175	17022	2584	14678	2307	10383	1581	6672	1362
	0.4	34776	4231	22215	3175	17022	2584	14678	2307	10383	1581	6672	1362
	0.6	34627	4220	22207	3173	17022	2584	14678	2307	10199	1507	6459	1385
	0.8	33014	4690	21996	3214	16858	2680	14605	2320	9722	1754	6307	1510
	1.0	32827	5105	21760	3735	16785	2845	14432	2739	9650	1890	6284	1606

Table 6.10: Trade off between compression ratio and flip counts

Circuit	Maximum reduced flip count Vs. Change in comressio ratio											
	m=16		m=32		m=48		m=64		m=128		m=200	
	Comp.	Flips	Comp.	Flips	Comp.	Flips	Comp.	Flips	Comp.	Flips	Comp.	Flips
s5378	-2.07	16.32	-0.90	12.77	-0.64	10.84	-.77	13.11	-0.01	2.14	-0.01	2.14
s9234	-1.77	12.70	-1.07	14.08	-1.41	21.44	-0.34	9.82	-0.19	15.11	-0.33	15.11
s35932	-3.57	9.63	-1.65	16.81	-0.64	12.94	-0.40	13.32	-0.26	11.19	-0.12	10.00
s38417	-0.39	6.91	-0.86	10.73	-1.91	26.38	-0.68	11.12	-0.11	15.65	-0.03	5.35
s38584	-3.22	18.90	-0.58	16.93	-0.63	22.25	-0.58	12.69	-0.30	12.53	-0.27	15.80
s15850	-3.14	23.03	-0.74	12.57	-0.56	19.56	-0.35	13.48	-0.17	10.90	-0.08	10.55
s13207	-1.1	19.52	-0.27	7.70	-0.15	18.89	-0.15	17.55	-0.33	19.50	-0.23	15.03

Table 6.11: Maximum reduced flip count Vs. change in compression length (%)

normalized values are then averaged for each w-value. The results are shown in Table 6.12, as well as in Fig 6.16. As expected, for $w = 1.0$, compression length is the minimum. However, interestingly, reduced flip counts are obtained for $w = 0.2$, $w = 0.4$ and $w = 0.6$ – not for $w = 0.0$.

Weights (w)	Average normalized Compression length	Average normalized Flip counts
0.0	0.997	0.926
0.2	0.909	0.868
0.4	0.910	0.868
0.6	0.901	0.869
0.8	0.887	0.912
1.0	0.880	1.0

Table 6.12: Average normalized compression length and flip count for different weights

Assignment of 0 and 1 to differentiate dictionary and non-dictionary entries

Table 6.13 shows the result for the total number of flip count after assignment of 0 and 1 to differentiate dictionary and non-dictionary entries. Column $N_d = 0$ represents the case in which non-dictionary entries are identified with prefix 0 and dictionary entries are identified by prefix 1. Table 6.14 shows the relative change in flip count in % if we assign $N_d = 0$ instead of $N_d = 1$. Negative values here indicates the increase in flip count for $N_d = 1$. The same results are also represented in Fig. 6.17. Table 6.14 and Fig. 6.17 show that for most of the cases for $N_d = 0$ a significant amount of switching activity can be reduced. It is noted that flip count can be reduced upto 95% if $N_d = 0$ instead of $N_d = 1$ for the circuit s13207 with $m = 16$. For few of the cases, like circuit s38584 with $m = 16$, gives better results when $N_d = 1$. For the circuit s13207 with $m = 128$ if we take $N_d = 1$, it is giving the better result instead of $N_d = 0$. It shows that the choice of prefix bit should also be done judiciously to reduce switching.

Circuit	Total number flip count for different scan chains											
	m=16		m=32		m=48		m=64		m=128		m=200	
	$N_d = 0$	$N_d = 1$	$N_d = 0$	$N_d = 1$	$N_d = 0$	$N_d = 1$	$N_d = 0$	$N_d = 1$	$N_d = 0$	$N_d = 1$	$N_d = 0$	$N_d = 1$
s5378	4088	4595	2849	2890	2208	2275	1851	1904	1152	1179	1152	1179
s9234	7053	7776	4788	5357	4404	4497	3454	3591	2045	2116	2045	2116
s35932	7155	7550	3955	3936	5196	5267	2881	2884	1117	1214	494	533
s38417	38576	36975	26087	28008	19887	19969	23926	24425	20372	20699	16823	17046
s38584	32991	31132	25585	27594	18513	21092	17267	18228	9460	9723	9076	9165
s15850	10501	12800	7704	7535	5708	6491	5098	5575	3127	3192	2430	2609
s13207	10931	21406	8118	12269	6682	8333	5305	6984	3967	3190	2528	2683

Table 6.13: Total number of flip counts after assigning 0 and 1 to differentiate dictionary and non-dictionary entries

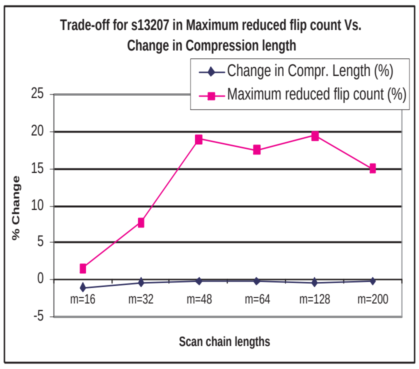
6.6 Conclusion

In this work we have presented a scheme for test data compression using Huffman coded dictionary. It significantly increases the compression ratio producing superior results compared to those reported in the literature. We also tried out

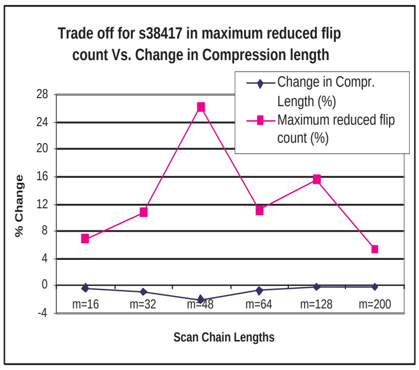
Circuit	Change in flip count for $N_d = 0$ Vs. $Nd = 1$					
	m=16	m=32	m=48	m=64	m=128	m=200
s5378	-12.40	-1.43	-3.03	-2.86	-2.34	-2.34
s9234	-10.25	-11.88	-2.11	-3.97	-3.47	-3.47
s35932	-5.52	0.48	-1.37	-0.10	-8.68	-7.89
s38417	4.32	-7.36	-0.41	-2.08	-1.60	-1.33
s38584	5.97	-7.85	-13.93	-5.58	-2.78	-0.98
s15850	-21.98	2.24	-13.71	-9.36	-2.08	-7.37
s13207	-95.83	-51.13	-24.70	-31.65	24.35	-6.13

Table 6.14: Relative change in flip count for $N_d = 0$ and $N_d = 1$ (all in %)

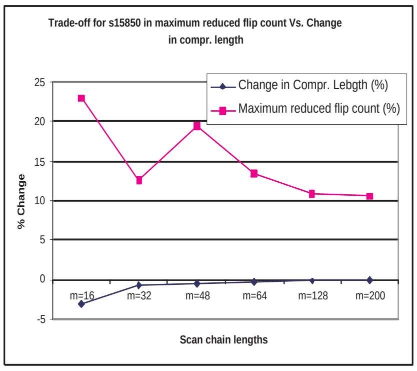
with the proposed scheme to give a trade-off analysis for the given compression technique. The scheme can be utilized to arrive at a compression scheme that would give optimal / desirable performance in terms of test application time and test power. The trade-off scheme also extended further to consider the scan-out power. The bus power reduction issues have also been addressed.



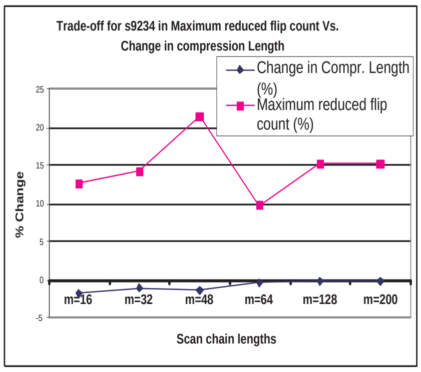
(a)



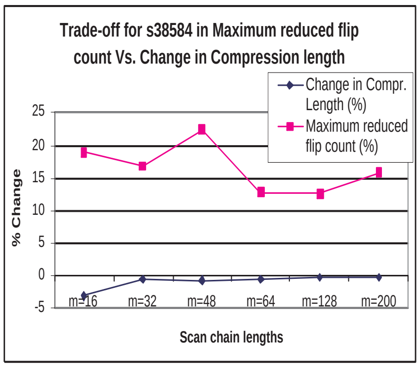
(b)



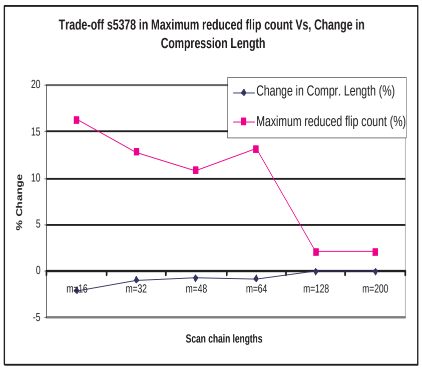
(c)



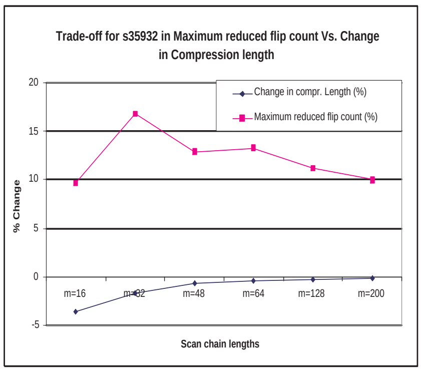
(d)



(e)



(f)



(g)

Figure 6.15: Trade-off results for maximum reduced flip count Vs. change in compr. length. (a) for s13207, (b) for s38417, (c) for s15850, (d) for s9234 (e) for s38580 and (f) for s5378 (g) for s35932

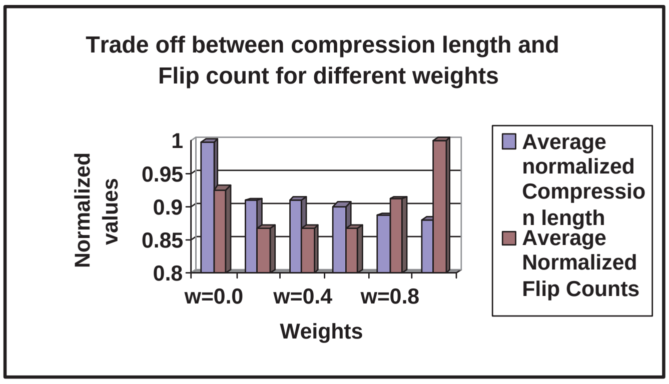


Figure 6.16: Trade off between compression length and Flip count for different weights

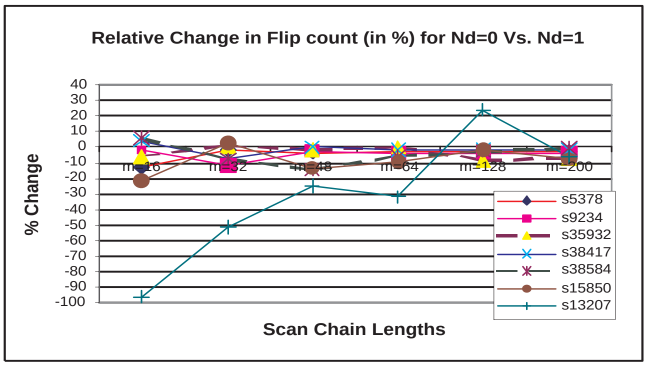


Figure 6.17: Relative change in flip count for $N_d = 0$ Vs. $N_d = 1$

Chapter 7

Test Data Compression by Split–VIHC(SVIHC)

7.1 Introduction

In the last chapter, we have seen a test data compression scheme that uses a dictionary to store the frequently occurring words in the test input vectors. This chapter enumerates another compression scheme, again based on Huffman coding, but does not use an explicit dictionary. The proposed scheme is a modification of Variable Length Input Huffman Coding (VIHC) technique [7]. It essentially splits the input file into two data sets to achieve higher compression.

The chapter is organized as follows. Section 7.2 gives a brief overview of the VIHC coding technique. Section 7.3 highlights the main motivation behind this work. Section 7.4 discusses about the proposed compression scheme and Section 7.5 deals with the decoder for decompressing the compressed test data. Detail discussion on experimental results are given in Section 7.6 and Section 7.7 draws the conclusion.

7.2 Brief overview of Variable Length Input Huffman Coding (VIHC) [7]

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 0's, 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 achieves a reduction in test application time (TAT). Fig. 7.1 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 0's (i.e. patterns of 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. Figure 7.1(a) shows the different patterns and their number of occurrences in the test vector t_{init} . Figure 7.1(c) shows the Huffman tree corresponding to the dictionary of Figure 7.1(b). Compression algorithm used in VIHC compression

$m_h = 4$ $|t_{init}| = 26$ bits $|t_{comp}| = 16$ bits

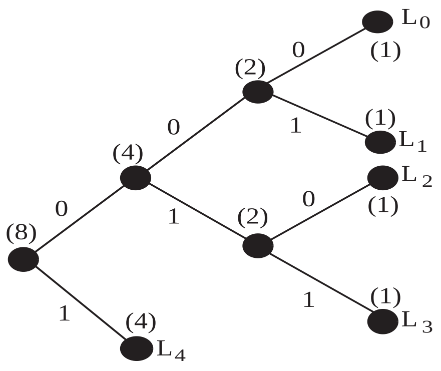
t_{init} 1 01 0000 0000 0000 0001 0000 001 ◀- - - - - Initial test vector

t_{comp} 000 001 1 1 1 001 1 010 ◀- - - - - Compressed test vector

(a)

Pattern	Frequency	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)

Figure 7.1: VIHC example [7]. (a) Test vectors (b) Dictionary (c) Huffman tree

scheme can be divided into three phases. These are,

1. **Preparation of initial test set:** As VIHC uses runs of 0's 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 0's will result. Such a difference pattern set will be called T_{diff}. On the otherhand, 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 0's. Next, the test set is reordered and will be denoted as *Reordered test set* in the algorithm. In the reordering process, starting with the test vector with minimum number of 1's, 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 1's in the difference between the test vector and the previous vector in t_{min} is minimum. For T_d, the next test vector is selected such that the length of the minimum run of 0's is maximum.

2. **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.

3. **Generate decoder information:** On-chip decoder for VIHC has to decode the Huffman codes, and it has to produce the original patterns (run's of 0's). 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.

7.3 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 7.1 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 change in occurrence can be noted for the patterns "0000001" and "00000001". Exchange of patterns between the partitions is expected to produce more such inversions. 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 scheme is thus similar to dynamic Huffman coding[198] to some extent, however due to the increased complexity of the associated decoder, we consider only two partitions. An example presented in Section 7.4 clarifies the idea further. Experimental results presented in Section 7.6 also support the idea where the change in coding strategy results in on an average

2-6% (for different run length of 0's) more compression than the original VIHC algorithm[7]. The test application time also reduces upto 29%. The scheme has been aptly named as Split-VIHC(or SVIHC for short). The associated challenges are the following.

1. Determining an optimal size for the partitions, that is whether the partitions be balanced etc.
2. Determining the patterns to belong to each of the partitions. Obviously, similar patterns being kept in a partition will increase the compression ratio.
3. Reordering patterns within a partition.
4. Designing the decoder for this Split-VIHC encoding.
5. 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 7.1: Distribution of different patterns over different regions of the test file for circuit s5378

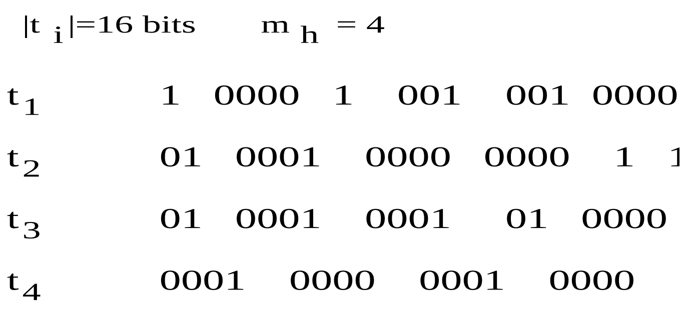


Figure 7.2: VIHC coding scheme for 4(four) different vectors[7]

7.4 Split VIHC(SVIHC) Scheme

Before going to the actual algorithm of split 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. 7.2. 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 $4 \times 2 + 3 \times 3 + 2 \times 3 + 5 \times 2 + 7 \times 2 = 44$ as shown in Fig. 7.3. Now, 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.

Pattern	Frequency	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

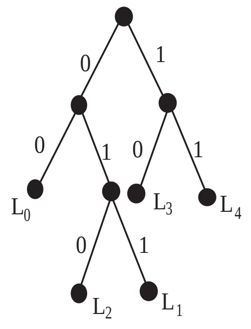


Figure 7.3: VIHC coding of the file in Figure 1

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. 7.4. 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 two important issues related to file splitting – the size of individual partitions, and the distribution of patterns to the subfiles.

For the first issue, splitting of the file has been done using Genetic Algorithm (GA)based approach. For the second one, it may be noted that for full scan circuits, we can freely change the order of the patterns. This has been exploited in the following subsection to develop a partitioning strategy for the file.

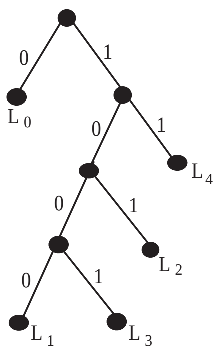
7.4.1 Partitioning strategy

It consists of the following steps:

Step 1: Divide the file into two parts using GA and for each vector determine the partition to which it should belong.

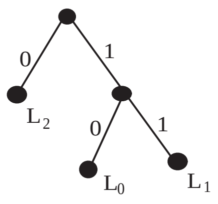
Step 2: Compute compression length using VIHC [7] scheme for each of the

Pattern	Frequency	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	Frequency	Code
L_0 =1	2	10
L_1 =0001	4	0
L_2 =0000	3	11



(b) For File 2

Figure 7.4: VIHC coding for two splitted file

partitions.

In Step 1 we divide the complete test data file into two subfiles. So, if a file contains N number of vectors and after splitting one file contains P number of vectors, the second file contains remaining $(N - P)$ number of vectors.

In this work GA formulation is used to partition the vector file into two and then for each vector determine to which partition it should belong. In order to solve the concerned problem we have to define individual representation of chromosome, mutation and crossover operators, fitness function and selection procedure. In the following we describe the GA formulation.

GA formulation for partitioning the vector set and allocation of vectors to partitions

For this problem chromosome structure is encoded with binary bit string $< p_1, p_2, ..., p_N >$ of length N depending on the number of test vectors present in the test data set, where p_i's are either 0 or 1. Each p_i value indicates the partition to which a pattern i belongs. All patterns with p_i value 0 are put into one partition, while others with $p_i = 1$ are put into the other partition. For example, if test vector set is of length 4, the chromosome $< 0110 >$ represents that the 1st and 4th vectors are kept in one partition, while the 2nd and 3rd are put into another. For each chromosome, fitness value is calculated. Here fitness value is the compression length obtained by applying VIHC [7] coding scheme on both the partitions.

To move towards the promising region of search space, different evolutionary operators are used. To create populations for new generation, 20% best-fit chromosomes are directly copied and remaining 80% chromosomes are created using crossover and mutation. From the entire population we select two chromosomes randomly for participating in crossover operation and two new chromosomes are produced after crossover operation for randomly selected crossover point of the chromosome. Mutation operation randomly selects a chromosome from the population and modifies the value within this for a randomly selected index. In this case the value at the selected index is to be inverted (i.e., 0 to 1 or 1 to 0).

The algorithm is run for 100 generations with $500 - 1000$ population sizes depending on the number of test vectors present in test vector set of the circuit. Chromosome with lowest fitness value provides the optimal solution for our problem.

Algorithm 7.1: Complete SVIHC Algorithm

```
1  begin
2      for k=1 to Number_of_generation do
3          foreach $chromosome_i$ in the population do
4              Divide the TestDataSet $T_d$ into two TestDataSet $T_d^1$ and $T_d^2$;
5              SetDefaultValues($T_d^1$, compression_type);
6              SetDefaultValues($T_d^2$, compression_type);
7              $T_d^{R1}$ = GetReorderTable($T_d^1$, compression_type);
8              $T_d^{R2}$ = GetReorderTable($T_d^2$, compression_type);
9              Generate Huffman Code for both $T_d^R$ (i.e. $T_d^1$ and $T_d^2$);
10             $Complength_i$=Complength($T_d^{R1}$)+Complength($T_d^{R2}$);
11         end
12         Sort $Complength_i$ in decreasing order;
13         Apply Genetic operators to get new population for the next
           generation;
14     end
15 end
```

Algorithm 7.1 gives the complete procedure proposed in this work. Input to the algorithm is the original test set T_d and compression type. Like other compression schemes, we consider two different types of compression. One in which the successive test vectors are XORed (compression type=*diff*) and the other in which the patterns are used directly (compression type =*ord*).

Algorithm 7.2: GetReorderTable(T$_d$,compression_type)

```
1  begin
2      $T_d^R$={$t_{min}$:the first vector in $T_d$};
3      Remove($t_{min}$,$T_d$),previous_vector=$t_{min}$;
4      while ( $T_d$ not empty) do
5          $t_{min}$=DetermineNextVector($T_d$,compression_type);
6          if compression_type $\neq$ diff then $T_d^R = T_d^R \bigcup t_{min}$;
7          else $T_d^R = T_d^R \bigcup$ XOR($t_{min}$,previous_vector);
8          Remove($t_{min}$,$T_d$), previous_vector=$t_{min}$;
9      end
10 end
```

Function *SetDefaultValues(T$_d$,compression_type)* maps the don't cares. For compression_type *diff*, the don't cares are mapped to the value of the previous vector on the same position, whereas for *ord*, don't cares are mapped to '0's to get long run of '0's. This is because, in '*diff*' type, the transmitted pattern is XORed with the previous test vector to get the new vector. Thus keeping the bits same in successive patterns will create long runs of 0s. On the otherhand, for '*ord*' type no XORing is done.

Function *GetReorderTable(T$_d$,compression_type)* performs reordering of the vectors within individual files according to the Algorithm 7.2. Depending on the type of the compression, function *DetermineNextVector(T$_d$,compression_type)* selects the vector such that number of 1's in the difference between the test vector and the previously selected vector is minimum and the length of the minimum run of 0's in the difference is maximum for *diff* compression. For *ord* it will be the vector which has maximum number of minimum runs of '0's (i.e 0's of length m_h).

7.5 Decoder design

Fig. 7.5 shows the parallel decoder structure for SVIHC. It is similar to the VIHC decoder[7] in the sense that it consists of two parallel units – a finite state controller(FSM) and a code generation unit(CGU). The interface between the two units is also similar to the VIHC decoder. The FSM provides *data, code* and *special_bit*. While *data* is the binary index of the word to be output by the CGU, *code* identifies the time when data is valid. *Special_bit* distinguishes an all zero block from the block having only the last bit as 1. We will first have a brief look on CGU design as it is exactly same as in [7].

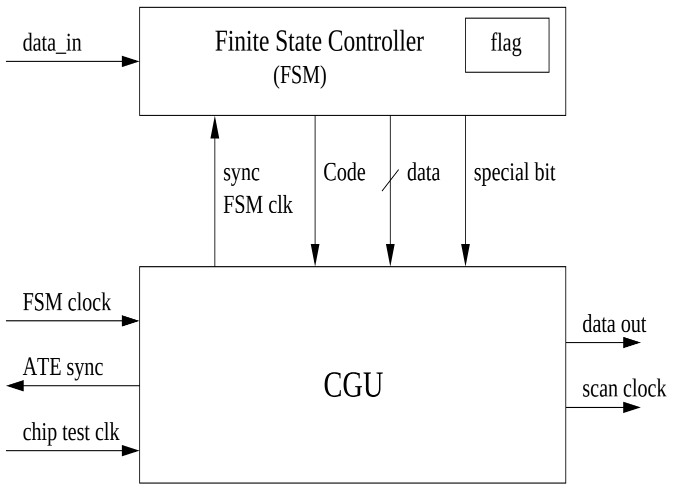
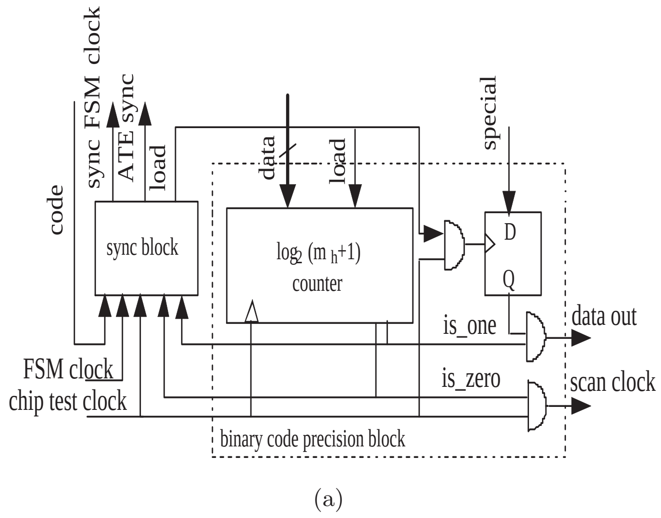


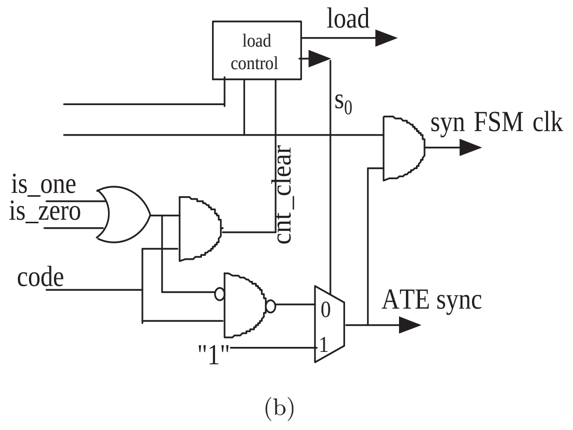
Figure 7.5: VIHC decoder

7.5.1 Code Generation Unit (CGU)

A high-level view of the CGU is shown in Fig. 7.6(a). There are two main components of the CGU: the binary code processing block and the synchronization block. The binary code processing block comprises of a counter that holds the length of the initial pattern, and a flip flop which holds the special bit . Being a parallel decoder, there are two issues to be considered. Firstly, the binary code processing block should be loaded only when a new code is identified; and secondly, the load of new data is done only when the pattern corresponding to the already loaded binary code was fully processed. These tasks are controlled by the synchronization block (*sync*) illustrated in Fig. 7.6(b). As noted earlier, when the Huff-decoder identifies a code, it will output the associated binary code and set the *code* line high. When the *code* line is high, the *sync* block will determine the values of *sync FSM clk*, *load*, and *ATE sync*.

The key element in the synchronization block is the load control FSM, whose state diagram is detailed in Fig. 7.6(c). The inputs into the load control FSM are the *cnt clear* and the FSM clock. The *cnt clear* will notify the FSM when the current code has been processed (that is, the counter value is either zero or one) and there is a new code available (that is, the code line is high). The FSM clock is used to synchronize the load control FSM with the Huff-decoder FSM. When *cnt clear* is 1, the FSM will change to S_1 and set the load to 1. After one clock cycle, the FSM will set the load to 0. If FSM clock is 1, the FSM will return to state S_0; otherwise, it will remain in state S_1. This last condition is needed since a new code can only occur after the FSM clock was 1; this is how the Huff-decoder is controlled. When the FSM is in state S_0, the S_0 line in





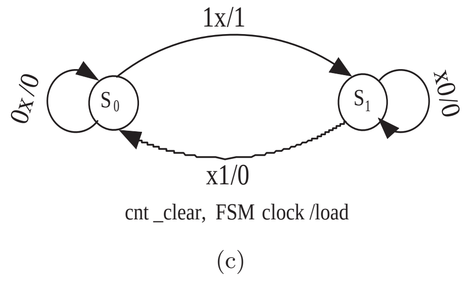


Figure 7.6: CGU for SVIHC decoder (a) CGU (b) Synchronization block (c) load control FSM [7]

Fig. 7.6(b) together with the logic behind the multiplexer will ensure that the Huff-decoder continues the load of data from the ATE once the data has been loaded into the binary code processing block, thus stopping the stream of data from the ATE to the Huff-decoder only when necessary.

Pattern	Frequency	Code
$L_0 = 1$	4	11
$L_1 = 01$	1	0111
$L_2 = 001$	2	00
$L_3 = 0001$	1	010
$L_4 = 0000$	4	10
$L_C = --$	1	0110

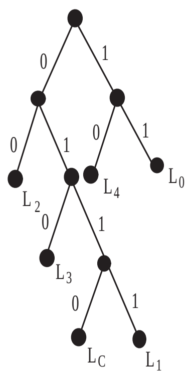


Figure 7.7: VIHC coding the file1 after inserting change over pattern

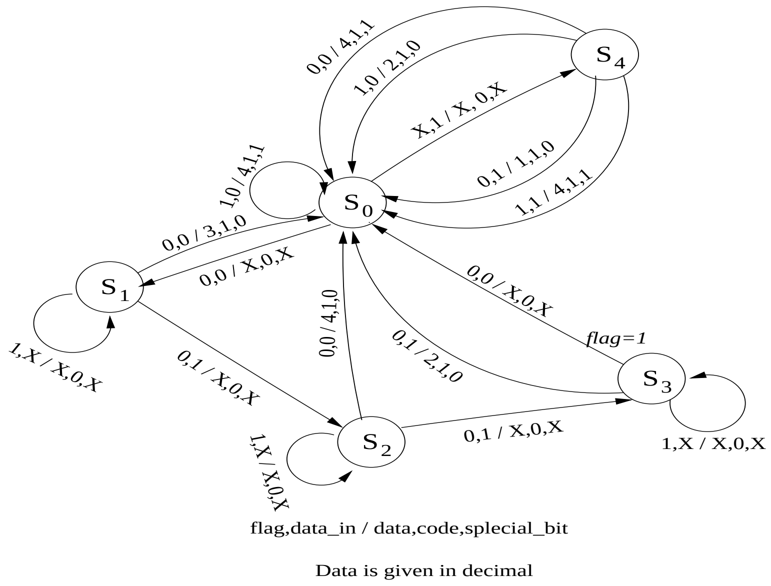


Figure 7.8: Final FSM for Huffman Decoder

7.5.2 Finite State Controller(FSM)

FSM detects the Huffman code and generates the corresponding binary code as proposed in the algorithm. According to SVIHC technique, test data is divided into two files and encoded separately by using two different Huffman trees. So, a straight forward approach will need two separate FSMs to detect code words from two files. But in SVIHC controller, only a single FSM is used to detect code words from two files. Decoding two files with single FSM is possible by introducing a *file change over pattern* at the end of first file along with an extra internal control bit *flag* in the decoder FSM(as shown in Fig. 7.5). The *change* *over pattern* helps to determine the switching point for decompressing the second file. Let us take an example to discuss the scheme we consider for the FSM.

Example: Consider that after splitting of the original test data file of Fig. 7.2 and application of the proposed algorithm in SVIHC, we have two dictionaries corresponding to the two splitted files, as shown in Fig. 7.4. As the Huffman decoder is a single FSM, the FSMs for *file1* and *file2* are to be merged by taking a change over pattern in *file1*. The dictionary along with change over pattern and the Huffman tree for *file1* is shown in Fig. 7.7. Change over pattern is indicated here as L_C which is of frequency 1. Corresponding final FSM for this example is shown in Fig. 7.8. FSM of the decoder takes two inputs, one is *data_in* and the other is *flag*, which is a control signal internal to the decoder. Starting from S_0, depending on the values of *data_in*, the Huffman decoder changes its state. It is important to note the following:

- Initially internal control signal *flag* is set to 0.
- After the controller detects a codeword, it gets back to state S_0. For example, if the data stream is 010(first bit first) of first file T_d^1, then FSM first changes its state from S_0 to S_1, then S_1 to S_2, after which back to S_0, and sets *data* and *special_bit* to the corresponding binary code ((100,0) in this case) and *code* line is set high.
- When the data in stream is 0110 (it is Huffman code of *file change over pattern* in the case of Fig. 7.7), FSM starts from S_0 and goes through states S_1,S_2,S_3 and back to S_0. During the transition from S_3 to S_0, FSM outputs *data* and *special_bit* as invalid and *code* line is 0, but internal control signal *flag* changes to 1. Thereby, the decoder starts identifying the patterns of second file T_d^2.

In this way, by using a *file change over pattern* and a *flag* bit, a single FSM is used to detect two files; thereby area overhead of the on-chip decoder is reduced.

7.5.3 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 [7]. The function by which we can approximate the TAT is given by

$$\tau(\alpha) \approx H(m_h) + \delta + n_0 \times \left\lceil \frac{m_h - w_{min} \times \alpha}{\alpha} \right\rceil \tag{7.1}$$

where,

- $\alpha = \frac{f_{chip}}{f_{ate}}$ is the ratio between the on-chip test frequency and the ATE operating frequency,

- n_0 is the number of patterns with length m_h(the patterns L_{m_h-1} and L_{m_h} are found from the pattern set $L = L_0, L_1, ..., L_{m_h}$ after applying VIHC[7] coding scheme for the group size m_h),

- 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.

As $\delta \ll H(m_h)$, so δ can be ignored. In this case the CGU should be able to generate the pattern L_i without stopping the Huffman decoder in the number of internal clock cycles needed by the Huffman decoder to identify the next codeword. Hence if the next codeword length be w_{i+1}, we should have $L_i < \alpha \times w_{i+1}$. Otherwise, the number of wait states will be $L_i - \alpha \times w_{i+1}$.

But if we put a buffer between the Huffman decoder and the CGU, the decoder gets an extra delay of length w_{i+2} while the CGU generates the pattern for L_i. Hence, in order for the Huffman decoder to run without being stopped by the CGU, the CGU should be able to generate pattern L_i in the number of internal clock cycles needed by the Huffman decoder to identify the codewords of length w_{i+1} and w_{i+2}, that is, $L_i < \alpha \times (w_{i+1} + w_{i+2})$. Otherwise, the number of wait states will be $L_i - \alpha \times (w_{i+1} + w_{i+2})$. The advantage of putting an extra buffer between Huffman decoder and CGU is that the total number of wait cycles suffered by the Huffman decoder is reduced, which in turn reduces TAT. Now, for the group size m_h, if total number of wait cycles suffered by the Huffman decoder is T_wait then the total TAT will be $H(m_h) + T\_wait$.

Circuit	**Compression Ratio with different group size(m_h)for T$_{diff}$**											
	4		**6**		**8**		**12**		**14**		**16**	
	V	SV	V	SV	V	SV	V	SV	V	SV	V	SV
s5378	51.52	53.18	54.46	57.00	55.23	58.25	55.85	60.38	56.66	59.35	57.33	59.86
s9234	54.84	56.61	58.12	60.46	58.75	61.81	58.45	63.14	58.76	63.22	59.02	63.01
s13207	69.02	69.88	75.09	77.07	79.07	80.35	82.03	83.59	82.69	84.33	83.21	84.94
s15850	60.69	62.62	65.67	67.96	67.48	70.18	68.65	71.74	68.86	72.21	68.99	72.32
s35932	40.35	42.94	49.92	50.32	56.97	57.10	61.08	61.84	62.54	62.62	66.47	66.73
s38417	54.51	56.09	58.75	60.80	59.96	62.55	60.86	65.49	61.22	65.24	61.98	65.49
s38584	56.97	58.13	61.21	62.31	62.50	65.21	63.01	67.53	63.00	67.76	62.97	68.07
Avg. Impr. (in %)	1.65		1.84		2.21		3.39		3.0		2.92	

Table 7.2: Compression ratio with different group size for T$_{diff}$

Circuit	**Compression Ratio with different group size(m_h)for T$_d$**											
	4		**6**		**8**		**12**		**14**		**16**	
	V	SV	V	SV	V	SV	V	SV	V	SV	V	SV
s5378	41.13	41.18	42.15	46.05	42.85	47.50	44.85	47.76	45.73	48.57	46.94	49.78
s9234	45.27	45.97	46.14	46.16	45.27	48.30	46.14	48.07	46.02	48.31	46.14	48.18
s13207	67.60	67.85	74.12	74.26	76.92	77.26	79.49	79.85	80.03	80.90	80.36	81.30
s15850	58.01	58.70	62.43	63.18	63.73	65.43	64.42	67.63	64.28	67.97	64.06	68.16
s35932	29.76	35.38	38.73	41.91	44.24	46.44	47.07	48.47	47.89	49.08	51.84	52.85
s38417	37.19	37.53	40.45	44.18	43.78	45.43	46.97	48.34	47.34	49.12	47.79	49.55
s38584	54.43	56.47	57.89	60.82	58.81	62.68	58.90	63.79	59.17	64.04	59.62	64.21
Avg. Impr.(%)	1.38		2.09		2.49		2.30		2.50		2.51	

Table 7.3: Compression ratio with different group size for T$_d$

7.6 Experimental results

In this work, we have experimented on the full-scan ISCAS89 benchmark circuits[186]. The experiments have been performed on a Pentium IV 2.6 GHz machine with 512MB RAM.

Table 7.2 and 7.3 present the results of the compression ratio with different group size for compression type T_{diff} and T_d respectively. It can be noted that for both the cases, compression ratio is increased by 2-6% from the original VIHC compression[7]. For the compression type T_{diff} the maximum increase in compression ratio is 5.1% for the circuit s38584 of group size $m_h = 16$. Maximum average improvement obtained is 3.39% for $m_h = 12$. It is to be noted from Table 7.2 that for s35932 the results are similar to the VIHC scheme. This is due to the small number of vectors (only 16) presents in the test data set of s35932. After dividing the file into two, the number of vectors present in each partition is small. Hence improvements are not good as compared to other circuits.

Similarly, for compression type T_d, the maximum increase in compression ratio is 5.62% obtained for the circuit s35932 of group size $m_h = 4$. In the case of T_d, as the distribution of 0's is mainly dependent on the mapping rather than the reordering technique, the increase in compression ratio is on an average smaller than the T_{diff} case.

Table 7.4 shows the comparison of the compression ratios obtained for different compression techniques (Table 7.4(a) is for compression type T_{diff} and Table 7.4(b) is for compression type T_d) proposed in the literature. It is seen that our proposed scheme gives more compression than the methods with their maximum compression ratio. For example for the compression type T_{diff}, we have compression ratio 10.62% more than Statistical coding(SC)[85], 22.12% more than Golomb coding[90], 22.87% more than FDR[91], 2.4% more than VIHC[7] coding scheme with their corresponding maximum compression ratio. Similarly for the

(a)

Circuit	T_{diff}				
	SC [85]	Golomb [90]	FDR [91]	VIHC [7]	SVIHC
s5378	52.33	40.70	48.19	60.73	64.51
s9234	52.63	43.34	44.88	60.96	64.42
s13207	77.73	74.78	78.67	86.83	88.27
s15850	63.49	47.11	52.87	72.34	75.63
s35932	65.72	N/A	10.19	71.91	71.94
s38417	57.26	44.12	54.53	66.38	68.72
s38584	58.74	47.71	52.85	66.29	68.79
Avg. % Compression	61.13	49.63	48.88	69.35	71.75

(b)

Circuit	T_d						
	SC [85]	Golomb [90]	FDR [91]	VIHC [7]	SVIHC	RL_Huff [94]	9C [87]
s5378	43.64	37.11	48.02	51.78	53.32	53.75	51.64(-)
s9234	40.04	45.25	43.59	47.25	48.88	47.59(-)	50.91
s13207	74.43	79.74	81.30	83.51	84.51	82.51(-)	82.31(-)
s15850	58.84	62.82	66.22	67.94	69.46	67.34(-)	66.38(-)
s35932	64.64	N/A	19.37	56.08	56.65	89.26	–
s38417	45.15	28.37	43.26	53.36	54.34	64.17	60.63
s38584	55.24	57.17	60.91	62.28	64.63	62.40(-)	65.53
Avg. % Compression	54.57	51.74	51.81	60.31	61.68	66.71	62.90

Table 7.4: Comparison with compression ratio for different techniques.(a) for compression type T_{diff}. (b)for compression type T_d

compression type T_d also we got 7.11% more compression than SC [85] , 9.94% more than Golomb [90], 9.87% more than FDR [91], 1.37% more than VIHC [7] with their corresponding maximum compression ratio. Our scheme performs at per with RL-Huffman[94] and 9C[87]. It produces better results than RL-Huffman in 4 out of 7 cases. Compared to 9C, it performs better in 3 out of 6 cases. However, as will be shown in Table 7.7, both RL-Huffman and 9C schemes require higher area compared to our approach.

Table 7.5 shows the comparison of Test Application Time(TAT) of our scheme for various α–values(α being the ratio between the on-chip test frequency and the ATE operating frequency($\frac{f_{chip}}{f_{ate}}$)). The calculation is done similar to the VIHC scheme[7] as discussed in Section 7.5. It is noted from Table 7.5 that on an average 21.29% reduction in TAT has been obtained for smaller value of α,($\alpha = 2$). Whereas, for $\alpha = 8$ upto 14.44% TAT reduction has been obtained for the compression type T_{diff}. Similarly for the compression type T_d, reduction of 29.83% has been achieved for $\alpha = 2$. For $\alpha = 8$, reduction is 19.75%.

The on-chip decoder for both VIHC[7] and Split-VIHC have been synthesized using the Synopsis Design Compiler[199]. The results are shown in Table 7.6 for group size m_h=4 for the circuit s5378. It is seen that total area overhead has increased by only 6.11% from the VIHC scheme. Table 7.7 shows a comparison of decoder area overheads of various compression schemes taking VIHC decoder area as unity. The total decoder area for RL-Huffman is not mentioned in[94]. Only the equivalent gate count for the controller has been mentioned. Going by this measure, experiments performed show that the FSM of our decoder requires 88% less gate count as compared to the RL-Huffman controller.

7.7 Conclusion

In this chapter, a new compression technique using Huffman coding on different sections of test data file separately has been proposed. Implementation of Split–VIHC technique on ISCAS89 benchmark circuits has shown up to 89% compression.

(a)

Circuit	Comp Method	TAT(ATE clock cycles) for T_{diff}			
		2	4	6	8
s5378	SVIHC	11851	9461	9461	9461
	VIHC	15868	11569	10777	10137
s9234	SVIHC	18920	15411	14525	14525
	VIHC	24895	17994	16905	16905
s35932	SVIHC	14967	10302	9945	9945
	VIHC	17584	12076	10857	9645
s38417	SVIHC	82697	63376	56839	56839
	VIHC	104642	74293	67940	62625
s15850	SVIHC	37386	25952	21306	21306
	VIHC	47366	32513	29437	26692
s38584	SVIHC	95682	71705	63563	63563
	VIHC	126969	92632	83130	82582
s13207	SVIHC	72920	39432	24877	24877
	VIHC	89865	52769	44180	36065
Avg. % reduction in SVIHC		21.29	18.57	20.83	14.44

(b)

Circuit	Comp Method	TAT(ATE clock cycles) for T_d			
		2	4	6	8
s5378	SVIHC	14066	11958	11933	11933
	VIHC	17668	13740	12914	12782
s9234	SVIHC	21486	20240	20240	20240
	VIHC	27235	23860	21154	21154
s35932	SVIHC	17581	13753	13373	13373
	VIHC	90920	55229	47319	40048
s38417	SVIHC	96133	83926	83113	83113
	VIHC	118989	92777	87914	87002
s15850	SVIHC	37318	28033	24517	24517
	VIHC	48169	34735	31055	30871
s38584	SVIHC	98655	77703	71253	71253
	VIHC	127849	92377	85783	80392
s13207	SVIHC	70827	41610	30320	30320
	VIHC	90920	55229	47319	40048
Avg. % reduction in SVIHC		29.83	24.65	23.29	19.75

Table 7.5: Test Application Time(TAT) comparison for $m_h = 16$. (a) for compression type T_{diff}. (b)for compression type T_d

Compression Scheme	Area Overhead for group size $m_h = 4$
	Total Decoder Area
VIHC	19822
SVIHC	21113

Table 7.6: Area overhead for group size $m_h = 4$(Synopsis Design Compiler)

Compression Scheme	Overhead
VIHC	1.0
SVIHC	1.065
SC	2.57
Golomb	0.92
FDR	2.35
9C	1.14

Table 7.7: Area overhead for different compression scheme taking VIHC decoder as unity

Chapter 8

Scan Chain Design for Delay and Power Minimization During Testing

8.1 Introduction

To alleviate the problem of sequential circuit testing, researchers have proposed scan-based architectures to connect the flip-flops present in the circuit forming one/many shift registers in test mode. This makes possible to consider the circuit as a combinational one for testing purpose. The test patterns can now be applied serially to the shift register input and the circuit outputs can similarly be shifted out. Such a chain of flip-flops is traditionally called a scan chain. The process, though significantly reduces the number of test pins, it also increases the test application time. To make the matter worse, the interconnects do not scale down in the same way as the devices on the silicon floor. Thus, with the usage of more circuitry to form a complete SOC, interconnect delay has become a major bottleneck.

Again power consumption during testing has emerged as a major challenge to both design and test engineers due to rapid changes in SOC design technology. Parallel testing of SOC cores though reduces testing time, might result in excessive energy and power dissipation. This problem is more acute in a scan-based environment as shifting of the input test stimuli and the test response create toggles in scan cells feeding inputs to the combinational circuit. Hence switching activity within the internal logic of Circuit Under Test (CUT) increases the power dissipation. Article [28] has shown that scan testing for some designs may consume peak power almost 30X over its normal operation. Hence it is important for the designers to ensure reduction in power dissipation in test mode also. This extra power consumption can give rise to severe hazards in circuit reliability or, in some cases, can provoke instant circuit damage. Moreover, it can create problems such as increased product cost, difficulty in performance verification, reduced autonomy of portable systems, and decrease of overall yield.

Works reported in the literature, though address those issues (interconnect delay and power consumption), few works have been done that considers both power and interconnect delay minimization simultaneously. In this chapter we have proposed a Genetic Algorithm (GA) based approach for this dual minimization problem. We consider our proposed method both for single and multiple scan chains.

The rest of the chapter is organized as follows. Section 8.2 discusses the estimation of power consumption and interconnect delay metric considered in this work. Section 8.3 addresses the mechanisms for reduction of power and delay. Section 8.4 provides GA formulation for optimization of both power and interconnection delay. Section 8.5 discusses the results obtained for both single and multiple scan chain designs using the power and delay minimization schemes. Finally Section 8.6 concludes the chapter.

8.2 Estimation of Power and Delay consumption

In this section we detail the power and delay estimation techniques used in the present work.

8.2.1 Estimating power consumption

In a scan environment, test patterns are applied by shifting in the test vectors and shifting out the output responses either by *test-per-scan* or *test-per-clock* method. To estimate the scan-in power dissipated by a test vector (or scan-out power dissipated by the corresponding output response), a *weighted transition metric* has been introduced in [149]. This weighted transition metric models the fact that the scan-in power (or the scan-out power for the corresponding output response) depends not only on the number of transitions in it, but also on their relative positions. The weighted transition metric has been shown to be strongly correlated to the switching activity at the internal nodes of the circuit during the scan-in and scan-out operations. It was shown experimentally that scan vectors (test vectors or output responses) with higher weighted transitions dissipate more power. This metric is therefore a good way to accurately estimate the power consumed during scan testing and hence avoid time consuming and

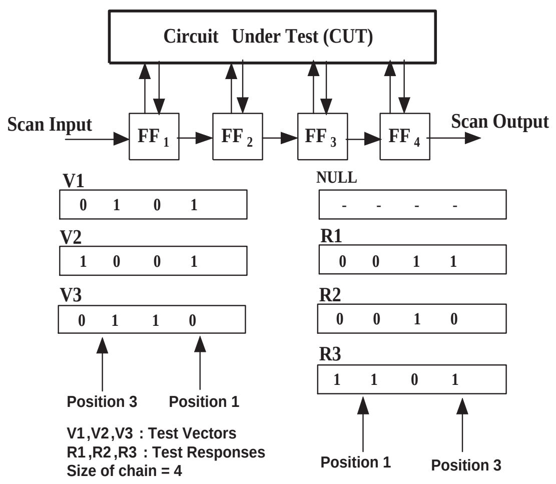


Figure 8.1: Circuit under test and input vector and output responses with their position of transitions

size limited simulations. More formally, the number of weighted transitions in a test vector or an output response is given by

$$Transitions = \sum(Size\_of\_chain - Position\_of\_Transitions) \tag{8.1}$$

Where, the first term, $Size\_of\_Chain$ is the number of flip-flops in the scan chain and the second term, $Position\_of\_Transitions$ is indexed from the least significant bit(LSB) for the input vectors, and from the most significant bit(MSB) for the responses.

Example:

Consider the scan chain shown in Fig. 8.1. Total number of test patterns is three (vector-response pairs). Initial content of four scan cells is ignored, which is indicated by NULL. Using Eqn. (8.1), total number of transitions can be calculated for scanning in V1, V2 and V3 (test vectors), and scanning out R1, R2 and R3(output responses). Cost of scanning in $V1 = (4-3)+(4-2)+(4-1) = 6$ and the cost of scanning out $R1 = (4-2) = 2$. Similarly, for other vectors, the transitions can be calculated. Total number of transitions using Eqn. (8.1) is $6+4+4+2+3+3 = 22$.

Apart from the scan-in 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. Thus, the total power consumed during scan testing is comprised of scan-in power, scan-out power and power consumed due to the above mentioned transitions. The event leading to this extra transition is called a *clash*. Total transition count is then given by Eqn. (8.2).

$$Total\_Transitions = \sum(Size\_of\_Chain - Position\_of\_Transition) + Size\_ofChain \times Clashes \quad (8.2)$$

For the given example in Fig. 8.1, number of clashes is 1 (for V2-R1 pair) as shown in Fig. 8.2. Hence including clash, total number of transitions calculated using Eqn. (8.2) will be $22 + 1 \times 4 = 26$. Thus, for any given set of test patterns we can compute the total number of transitions for traditional scan chain architecture.

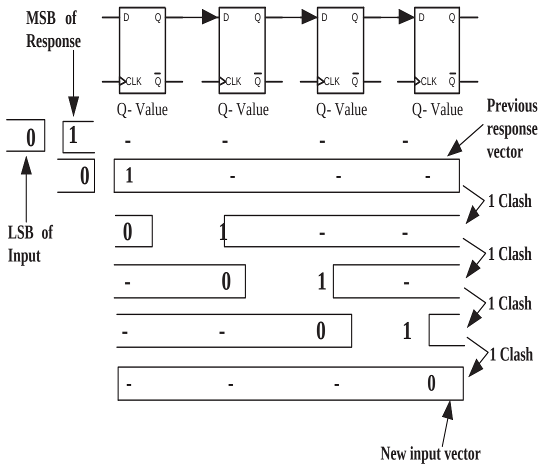


Figure 8.2: Example of Clash

8.2.2 Estimation of Delay

Delay can be calculated from the length of the scan chain. So if length of the scan chain is l, the delay can be obtained as [200] :

$$Delay \propto l^2 \quad (8.3)$$

8.3 Power Reduction Mechanisms

8.3.1 Scan Cell Ordering

With the normal connection of the flip-flops, without any ordering of scan-cells to form the scan chain, it results in particular number of transitions and a particular amount of delay. But, if the scan-cell ordering is employed then we can optimize the switching activity and interconnect delay between the flip-flops. For example, in Fig. 8.3, we assume four flip-flops with their positional coordinates (normalized between 0 and 1) and the test vectors as specified.

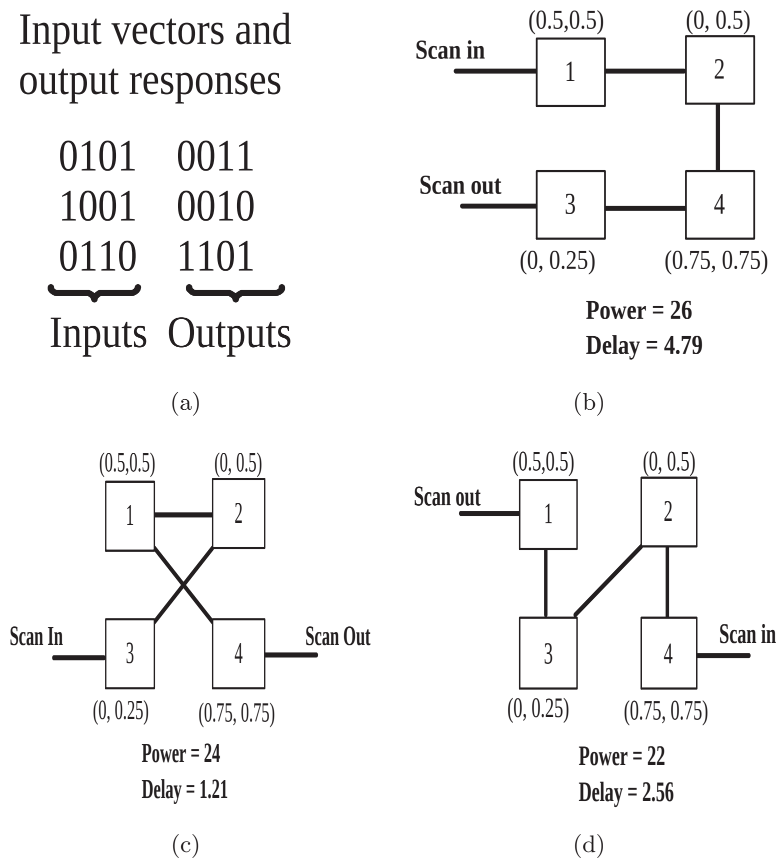


Figure 8.3: Example of scan cell ordering with delay and power calculation

We can see from the Fig. 8.3(b) that if we go for a normal connection of flip-flops without any ordering of scan-cells, the number of transitions is 26 and the delay (calculated as sum of the Cartesian distances between the cells) is 4.79. Now if we go for scan-cell ordering, we can minimize either the switching activity or the delay. From Fig. 8.3(c), we can observe the maximum optimization of delay with respect to the normal connection. Fig. 8.3(d) shows the maximum optimization of switching activity resulting in minimum number of transitions with respect to the normal connection.

8.3.2 Test Vector Ordering

In this section we try to reduce the switching activity by reducing the clashes, which occur when the most significant bit (MSB) of the previous output being scanned-out is different from the least significant bit (LSB) of the present input being scanned-in. This occurrence of a clash causes a transition count equal to the length of the scan chain (i.e. equal to the number of flip-flops in the scan chain).

To reduce the switching activity, that is, transition count further, we optimally reorder the test patterns. This essentially removes the clashes. Following steps summarize the algorithm for test vector reordering:

1. Attach a *tag* XY to every test pattern; where X is LSB of the test vector, and Y is MSB of the output response.
2. Four types of *tags* are possible: 00, 01, 10 and 11. Divide the complete set of test patterns into a maximum of 4 groups on the basis of these tags.
3. List all the test patterns in the group 00.
4. Pick test patterns from group 01 and 10 alternately. If any one of these two groups exhausts, list all the remaining patterns from the other group ignoring the alternate picking policy.
5. List all the test patterns in the group 11.

The algorithm shown above is based on a simple observation. Consider two test patterns A and B with tags $X_A Y_A$ and $X_B Y_B$ respectively. A followed by B would cause a *clash* if and only if $Y_A \neq X_B$. This follows directly from the definition of a clash: Y_A is the MSB of the predecessor's response and X_B is LSB of successor's vector. In the algorithm we minimize the number of such occurrences. Thus upon implementing the presented algorithm along with the scan cell ordering we can get more improvement in the switching activity when compared to the results with only scan-cell ordering.

8.3.3 Scan Architecture Modification

Conventional way of connecting two consecutive flip-flops in a scan chain involves connecting the Q (output of predecessor) to D (input of successor). In [155], a simple modification to this approach has been suggested to allow $\overline{Q}$ (negated output of predecessor) connection to D (input of successor). This latter type of connection is selectively done at various positions within the scan chain. During the scan-in/scan-out of vectors/responses, any two differing consecutive bits within the vector/response cause flip-flop transition at every clock tick. These transitions can be reduced in number if we set up the flip-flop interconnections optimally. Any modification of the scan cell architecture will necessitate adaptation of test vectors such that after scan-in they take the original form.

Our objective is to make a scan architecture for effective handling of all the given test patterns, that is, we must consider all test vector-response pairs together and decide connections (at each flip-flop junction) that lead to the best possible overall reduction in the number of transitions. The test patterns are pre-determined using a suitable ATPG (ATALANTA [139] in our case).

We can compute the cost at every index i of the scan chain while having 1) $Q-D$ connection and 2) $\overline{Q}-D$ connection. The calculation given by Eqn. (8.4) decides an optimal connection type for every scan chain index.

$$Cost^{i}_{10,01} = VBitTotal^{i}_{10,01} \times i + RBitTotal^{i}_{10,01} \times (SizeofChain - i) \quad (8.4a)$$

$$Cost^{i}_{11,00} = VBitTotal^{i}_{11,00} \times i + RBitTotal^{i}_{11,00} \times (SizeofChain - i) \quad (8.4b)$$

$$FF^{i}_{connection} = Cost^{i}_{11,00} \geq Cost^{i}_{10,01}, \forall i \epsilon \{1, SizeofChain - 1\} \quad (8.4c)$$

In Eqn. (8.4a), i is the index of the flip-flop junction in the scan chain, that is, the junction between i^{th} and $(i+1)^{th}$ flip-flop. $VBitTotal^{i}_{10,01}$ is the number of times the consecutive bits differ at position i when considering all the test vectors. $RBitTotal^{i}_{10,01}$ is the number of times the consecutive bits differ at position i when considering all the test responses. $VBitTotal^{i}_{11,00}$ and $RBitTotal^{i}_{11,00}$ are similarly defined when consecutive bits are same. $Cost^{i}_{10,01}$ stands for the number of transitions that will take place due to consecutive bits at indices i and $i+1$ while keeping flip-flop connection as $Q-D$. Similarly, $Cost^{i}_{11,00}$ is defined while flip-flop connection is kept as $\overline{Q}-D$. $FF^{i}_{connection}$ is assigned a boolean value *true* or *false* whichever favors a lower value for transition cost. *True* implies a $Q-D$ connection, whereas a *false* implies a $\overline{Q}-D$ connection.

Test vector customization

Test vector customization is responsible for the following task:

- It takes the test vectors and adapts them to the optimized scan cell architecture such that after scanning in they assume the desired original form. The rule to be followed for this customization is as shown below.

While scanning in the input vector, flip the value of the bits which face odd number of $\overline{Q}-D$ connections else sustain the bit value same as in the original test vector.

The above step ensures that the test vector after getting scanned-in, assumes the desired original value.

8.3.4 Filling unspecified bits for test vectors

Filling the unspecified bits carefully can reduce the transitions significantly. For example, consider an original input test vector as $10XX$ without scan-cell ordering and after the reordering of the flip-flops the test vector to be applied to CUT is transformed into $1XX0$. As the LSB of the input vector is scanned-in first, the don't cares are specified to assume the same bit value on its LSB side. In this example, the don't care values assume the bit value on its LSB side, that is, logic 0 and hence the test vector becomes specified as 1000.

However, if a test pattern contains X's, the output cannot be specified completely. Thus, actual number of transitions cannot be computed. For this, once all the X's of a pattern have been specified, we need to simulate the circuit to get the response.

8.4 GA Formulation for Scan Cell Ordering

In this section, we present a Genetic Algorithm based formulation to perform scan cell ordering. As noted earlier, it is assumed that the scan cell positions are known. They are to be chained in such a way that the overall scan shifting time and number of scan transitions are reduced. Further, as we have seen in the last section, there can be a number of ways in which the scan transitions can be reduced further. The techniques are test vector ordering, scan architecture modification by using both true and complemented outputs of the flip-flops, don't care filling etc. These techniques have been integrated with GA so that the fitness measure of a chromosome takes care of these added features over and above the basic scan cell ordering.

8.4.1 Chromosome representation

For the current problem, for a scan chain with N flip-flops, the chromosome structure consists of N distinct integers $p_1, p_2, p_3, ..., p_N$ where p_i represents the i^{th} flip-flop in the scan chain. Each p_i can assume a distinct random value between 1 and N. For example, if there are 5 flip-flops to be chained, a chromosome $< 1, 5, 4, 2, 3 >$ indicates that the flip-flops are to be connected in this order.

8.4.2 Genetic operators

Two genetic operators, namely, crossover and mutation have been used to evolve the population over generations.

Crossover operation: Crossover operation is performed between two randomly selected chromosomes. To evolve better offspring, the selection of chromosomes has been biased towards better fit ones. For this purpose, entire population is sorted on fitness values. Top 20% of the population is marked as best class. To select a chromosome, first a random number between 0 and 1 is generated. If the number is greater than 0.5, a chromosome from the best class is selected, otherwise a chromosome is selected from the entire population. After selecting two chromosomes to participate in crossover, a random number between 1 and N is generated. Off-springs are created by exchanging the portions of chromosomes around this point. It may be noted that in the process, some of the numbers may be repeated within a chromosome. To resolve this, these values are replaced by the values not occurring in the chromosome. Thus, a single point crossover is used to create new offspring.

Mutation operation: Mutation is carried out by first selecting a chromosome as in the case of crossover. Then, two random positions on the chromosome are swapped to introduce a mutation.

To evolve a new generation from the current generation, the best class chromosomes are copied directly. 30% of the chromosomes of the new generation are generated via mutation. The remaining 50% of the population is created via crossover. Population size has been varied between 200 and 1000 based on the number of scan flip-flops. The GA is terminated when there is no improvement in the best fitness value over the last 50 generations.

8.4.3 Fitness Measure

In this problem, we take the fitness function as the weighted sum of both normalized value of delay and total number of transitions. This can be represented as

$$Fitness = w \times Normalized\_total\_transitions + (1 - w) \times Normalized\_delay \tag{8.5}$$

where

$$Normalized\_total\_transitions = Total\_transitions / Maximum\_total\_transitions \tag{8.6}$$

and

$$Normalized\_delay = delay / Maximum\_delay \tag{8.7}$$

$Maximum\_total\_transitions$ and $Maximum\_delay$ are obtained from the initial population. These are the maximum values amongst all chromosomes in the initial population. Values for $Total\_transitions$ and $delay$ are obtained from Equations (8.2) and (8.3) respectively. The weight w is varied between 0 and 1.

8.5 Experimental Results

In this section, the experimental results are presented for the ISCAS'89 sequential benchmark circuits. Test patterns have been generated from the full-scan versions of the benchmarks using the combinational test pattern generator ATALANTA[139]. Since no placement data is available for the benchmarks, the position of scan flip-flops have been generated randomly on a two dimensional plane with co-ordinates varying in the range of 0 to 1. A sample of the co-ordinates generated for the circuit s510 has been shown in Table 8.1. The power and delay reduction schemes proposed in Section 8.3 are applied both to the single scan chain as well as the multiple scan chain designs for different w values determining the weightages on power and delay. The following sections discuss the results obtained for both.

Flip-flop Number	Co-ordinates
1	(0.500,0.333)
2	(0.333,0.500)
3	(0.833,0.833)
4	(0.000,0.167)
5	(0.500,0.667)
6	(0.167,0.000)

Table 8.1: Co-ordinate values for the circuit s510

The results have been presented in the following sequence.

1. First, only scan cell reordering is done for single scan chain.

2. Next, test vector reordering augments the scan cell reordering.

3. The scan architecture is modified along with scan cell and test vector reordering incorporated into it.

4. Proper filling up of the don't care bits is merged with the schemes considered upto Step 3.

In each step power consumption and delay have been estimated for different weights (w) varying between 0 and 1. From this estimation, power-delay product (PDP) is determined, that reflects the amount of energy required to perform an operation. Hence, lower the PDP, better is the circuit performance.

8.5.1 Scan cell reordering

In Table 8.2, the column "Original" notes the transition count (Pow) and delay (Del) for the initial flip-flop ordering as input to the system without applying any power consumption reduction strategy.

Circuits	No.of FFs	**Original**	
		Pow	Del
s298	14	2804	32.6
s344	15	2524	39
s349	15	2775	68.2
s382	21	7045	83.4
s400	21	6815	83.4
s420.1	16	7214	57.6
s444	21	6175	83.4
s526	21	12730	75.4
s526n	21	13751	75.4
s713	19	8121	114
s838.1	32	51782	300
s953	29	28588	288
s1196	18	23154	85
s1238	18	23650	85
s1423	74	172329	1228.7

Table 8.2: Results for single scan chain without any of the proposed power optimization schemes

Circuits	**w=0.0**		**w=0.2**		**w=0.5**		**w=0.8**		**w=1.0**	
	Pow	Del	Pow	Del	Pow	Del	Pow	Del	Pow	Del
s298	3118	6.1	3000	6.5	2754	8.5	2648	14.6	2521	35.2
s344	2815	12.4	2700	12.2	2435	15.6	2315	19.9	2198	54.2
s349	2980	30.5	2592	31.2	2365	37.2	2227	48.9	2240	66.1
s382	7228	31	7540	30.8	6987	34.1	6283	44.1	5974	105.4
s400	7554	31.1	7396	31.2	6198	40	6053	44.7	5747	83.4
s420.1	7127	19.7	6007	21.6	5913	21.9	5649	26.8	5471	55.1
s444	6720	31.1	6065	32.7	5703	34	5514	44	5327	83.6
s526	13509	28	12470	27.2	11725	32.3	11028	47.6	10703	86.4
s526n	14443	28	13284	28.4	12236	33.6	11131	54.3	10808	109.5
s713	9667	8.9	8359	11.6	8004	15.9	7358	33.5	7027	74.8
s838.1	52980	81.1	46866	83.1	44380	85.6	40373	114.1	38640	288.3
s953	29931	75.8	27439	78.9	24789	85.1	23138	119.4	22480	211.5
s1196	23644	12	22690	14.3	20308	17.3	19475	22.5	18535	104.9
s1238	24201	12	21847	15.1	20618	15.9	19753	19	19029	65.5
s1423	182213	541.2	165274	551	157721	584	14537	749.3	143998	1541.3
Avg. % impr.	**-7.21**	**70.1**	**1.7**	**68.9**	**9.3**	**64.4**	**14.6**	**51.5**	**17.6**	**-5.9**

Table 8.3: Results with scan cell ordering for single scan chain

Table 8.3 presents the results obtained after scan cell ordering. It may be noted that $w = 0.0$ optimizes only the delay, taking no concern for power. Similarly, $w = 1.0$ optimizes only the power without any concern for delay. Rest of the w values each optimizes a weighted sum of delay and power. The results have been summarized in the row marked "Avg. % Impr.". It may be noted that for $w = 0.0$, power has effectively increased, whereas the minimization in delay is the maximum among all w-values. The other extreme has occurred for $w = 1.0$, where the improvements in delay is the minimum, but for this, power improvement is the maximum. Intermediary values show similar trends. Lower

w values have produced solutions optimizing delay more than power, whereas higher w-values have done the reverse. Thus the GA formulation has been able to provide the test designer an effective trade-off in test time and scan power minimization using scan flip-flop reordering.

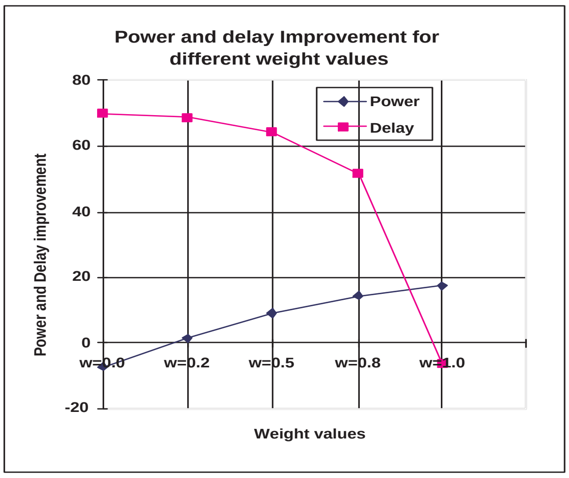


Figure 8.4: Power and delay improvement with scan cell ordering

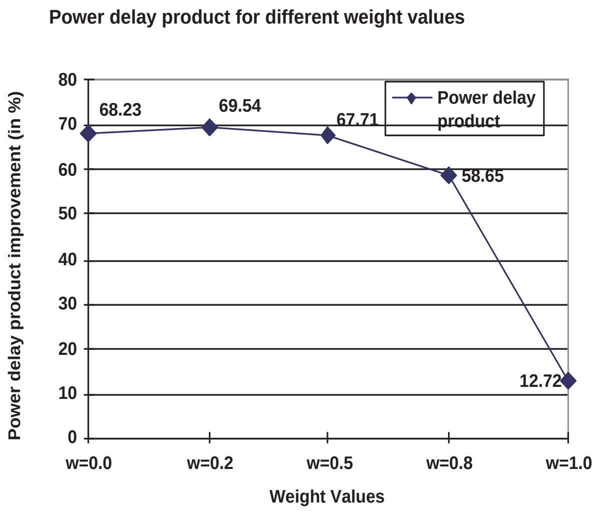


Figure 8.5: Power delay product with scan cell ordering

The graph in Fig. 8.4 shows a drastic fall in delay improvement after $w = 0.8$. Maximum average power improvement 17.6% is achieved through scan cell reordering. It suggests other avenues like pattern reordering, don't care filling etc. be tried in conjunction with scan cell ordering. Fig. 8.5 shows the graph for average % improvement in PDP. It is seen that after weight $w = 0.8$, PDP is falling drastically due to the drastic change in delay, that has increased in this case from the delay value for weight $w = 0.8$. So, it is better to select the weight $w = 0.8$ to optimize both the power and delay, as in this case average improvement in PDP is 58.65%.

Circuit	w=0.0		w=0.2		w=0.5		w=0.8		w=1.0	
	Pow	Del	Pow	Del	Pow	Del	Pow	Del	Pow	Del
s298	2821	6.1	2454	7.7	2383	9.4	2250	11.9	2158	36.5
s344	2720	12.4	2386	13.4	2184	16.4	2005	21.6	1893	48.5
s349	2585	30.5	2311	33.8	2164	34.6	2078	42.9	2019	70.9
s382	7745	31	6243	32.6	5946	36.5	5537	41.5	5230	99.6
s400	7233	31.1	6237	31.2	5608	36.4	5194	47.2	5051	100
s420.1	6595	19.7	5445	20.4	5065	23.3	4800	29.4	4695	43.5
s444	6521	31.1	5518	31.7	4962	35.5	4783	44	4421	84.4
s526	12821	28	11602	29.8	10641	35	10274	44.3	9882	82.5
s526n	13646	28	12701	28.7	11170	32.8	10092	52.6	9908	136
s713	9124	8.9	8477	9.6	7496	14.1	6611	21.6	6221	65.3
s838.1	50289	81.1	44010	81.5	40459	95.8	36832	132.4	35813	266.3
s953	28297	75.8	25246	79	22255	86.5	20890	113.8	20362	290.4
s1196	23228	12	20697	14.5	19357	19.1	18164	21.6	17071	100.1
s1238	23052	12	21273	13.4	19402	18.1	18536	26.6	17795	67.6
s1423	179607	541.2	161379	546.3	151776	606.4	146166	721.1	141076	1844.8
Avg. % impr.	**-1.66**	**69.7**	**10.36**	**68.3**	**17.76**	**63.4**	**23**	**52.6**	**26.2**	**-9.4**

Table 8.4: Results with scan cell ordering and test vector reordering for single scan chain

8.5.2 Test Vector Reordering

In Table 8.4, result has been shown corresponding to the power and delay improvements, where test vector ordering has been used to augment the scan cell reordering. It can be noted that the maximum average power consumption improvement is 26.2%. Though in this case delay has increased upto 9.6%. Fig. 8.6 represents the average improvements in delay and power for different weights. Similar to the only scan cell ordering scheme, it is better to select the weight $w = 0.8$ to optimize both power and delay. In this case average improvement in PDP has improved to 63.6% and this is depicted in Fig. 8.7.

8.5.3 Polarity and test vector customization

In this step the results are obtained after modifying the scan architecture by proper selection of connection Q or $\overline{Q}$ of one flip-flop to the D input of the next flip-flop.

From Table 8.5 it can be seen that the maximum average power consumption has improved by 25.9% for weight $w = 1.0$. But again in this case delay has increased by 5.7%. Table 8.5 is summarized in Fig. 8.8. Fig. 8.9 depicts the average improvement in PDP. It also shows that $w = 0.8$ is a better choice to

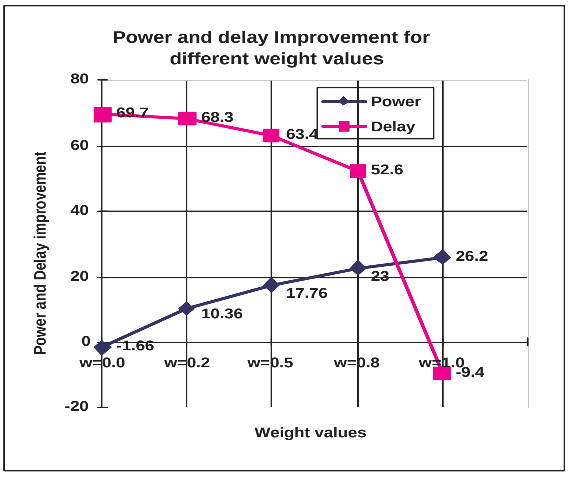


Figure 8.6: Power and delay improvement with scan cell ordering and test vector reordering

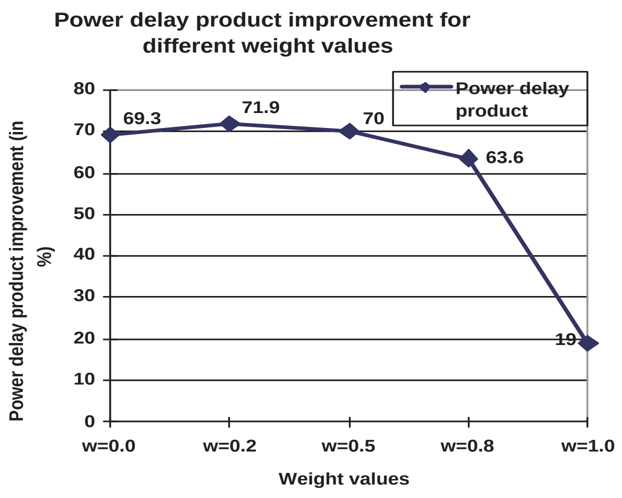


Figure 8.7: Power delay product with scan cell ordering and test vector reordering

get the optimized power consumption and delay. In this case average PDP has improved by almost 64.7%.

8.5.4 Filling unspecified bits

Test vectors are generated using ATPG tools (like ATALANTA, in our case) for ISCAS'89 benchmark circuits contain a large number of don't cares. If don't cares are present in the input vector, there are chances that there will be don't

Circuit	w=0.0		w=0.2		w=0.5		w=0.8		w=1.0	
	Pow	Del	Pow	Del	Pow	Del	Pow	Del	Pow	Del
s298	2657	6.1	2447	7.1	2317	8.3	2157	13.3	2157	13.3
s344	2196	12.4	2188	12.2	1940	15.8	1857	21.4	1857	21.4
s349	2215	30.5	2050	31.2	1949	36.2	1757	48.4	1757	48.4
s382	6794	31	5676	34.5	5640	37.6	5229	41.7	5229	41.7
s400	6452	31.1	5549	33.8	4993	36.7	4967	39.3	4967	39.3
s420.1	6594	19.7	5407	21.7	5064	23.3	4719	32.2	4719	32.2
s444	5776	31.1	5484	33.3	4700	38.5	4449	47.1	4449	47.1
s526	12320	28	11380	29.6	10550	33	9737	53.8	9737	53.8
s526n	12479	28	11664	28	10905	31.9	9964	48.2	9964	48.2
s713	8602	8.9	7046	10.8	6469	14.5	6163	20.5	6163	20.5
s838.1	50205	81.1	44270	81.7	39785	90.5	37830	118.6	37830	118.6
s953	26956	75.8	22939	78.2	21959	83.3	20439	106	20439	106
s1196	20424	12	18857	15.3	18590	16.3	16825	28	16825	28
s1238	19661	12	18330	14.8	17744	15.8	16710	23.4	16710	23.4
s1423	171143	541.2	154724	560.4	145280	611.9	142286	760	142286	760
Avg. % impr.	**7.06**	**69.7**	**16.7**	**68**	**22.4**	**64**	**27.1**	**51.4**	**29.9**	**-5.7**

Table 8.5: Results with polarity and test vector customization for single scan chain

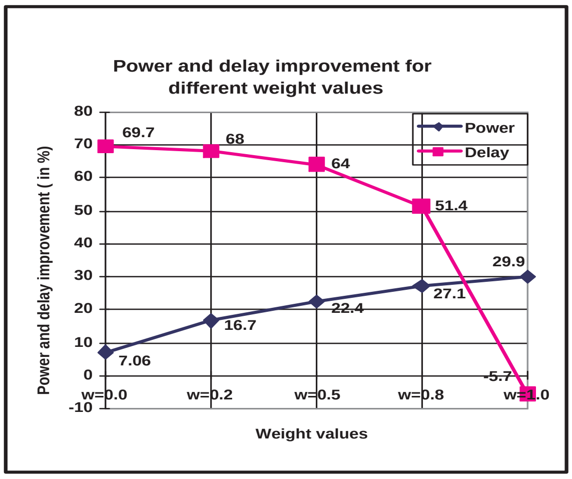


Figure 8.8: Power and delay improvement with scan-cell ordering and test vector reordering and polarity with test vector customization

Circuits	w=0.0		w=0.2		w=0.5		w=0.8		w=1.0	
	Pow	Del	Pow	Del	Pow	Del	Pow	Del	Pow	Del
s420.1	6926	19.7	5396	22.4	5343	22.5	5150	23.6	5043	63.2
s526	13048	28	11884	30.4	11726	30.7	11061	38.8	10788	118.7
s526n	12471	28	11325	30.3	10992	38	10512	44	9751	109.7
s713	9054	8.9	7443	10.22	6861	13.5	6326	19.5	6057	51.4
s838.1	41284	81.1	34734	82.3	31318	87.8	30979	99.1	28190	293.9
s953	26526	75.8	23010	75.9	21446	84.7	20069	105.5	19366	215.7
s1196	19007	12	17635	14.2	17434	15.5	15786	26.3	14921	102.5
s1238	19657	12	16768	15.9	16676	16.1	15739	24.4	15297	76.9
s1423	174321	541.2	152503	560	142931	619	135010	786.5	131922	1457
Avg. % impr.	**13.1**	**72.8**	**24.6**	**70.8**	**27.8**	**68**	**31.8**	**60**	**34.8**	**-6.7**

Table 8.6: Results with don't cares properly filled for single scan chain

cares even in the corresponding output response, but filling those output response bits is not possible as they depend upon the input vector being applied to the

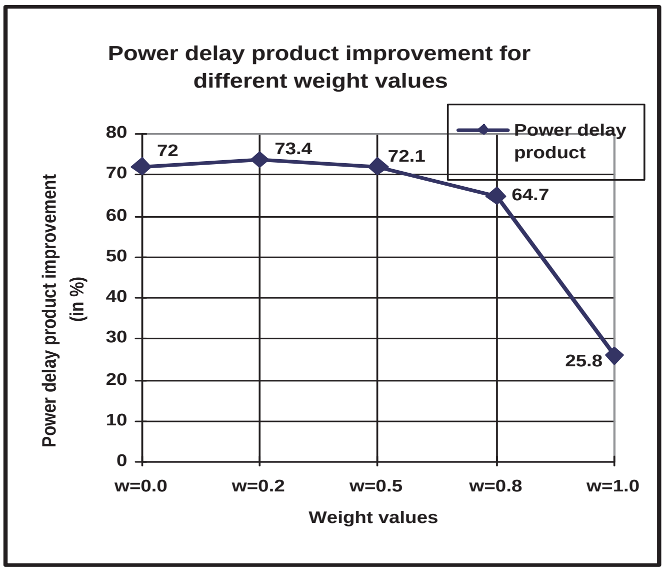


Figure 8.9: Power delay product with scan-cell ordering, test vector reordering and polarity with test vector customization

CUT. Hence, don't cares in the input vectors are properly filled first and then FSIM fault simulator is used to get the output responses. The required vectors are then extracted, that are to be applied to the flip-flops of the CUT.

In this section, results have been reported after implementing the scan-cell ordering, test vector reordering, and polarity selection with test vector customization, and proper don't care filling.

Table 8.6 presents results when unspecified bits are filled properly. It can be noted from the table that maximum average improvement in power is 34.8%, whereas the delay has increased in this case upto 6.7%. The graphs represented in Fig. 8.10 show the average improvement in power and delay. It is seen from the figure that though the power has increased for different weights, after weight of $w = 0.8$ delay has increased. In Fig. 8.11, improvement in power-delay product is shown, that shows the same trend as before. So selection point for optimizing power and delay as weight $w = 0.8$ is the better choice. In this case average improvement in PDP is 71.9%, whereas, at $w = 1.0$ it reduces to 28.7%.

Fig. 8.12 summarizes the comparison of power consumption results obtained at different steps of the flow of the scheme. It is seen that the average power consumption results are improved in each step as the different strategies are introduced one by one in steps.

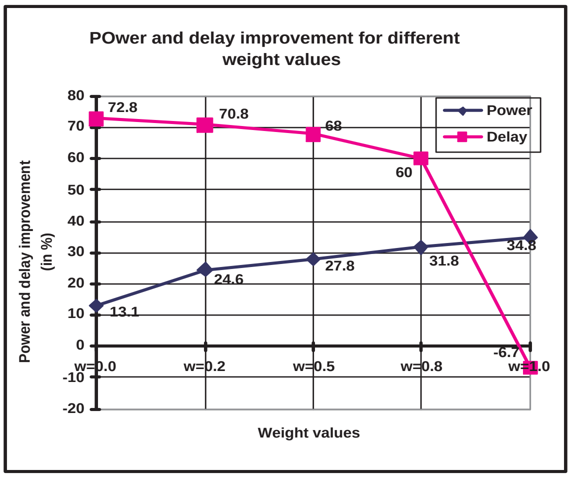


Figure 8.10: Power and delay improvement with scan-cell ordering, test vector reordering and polarity with test vector customization with don't cares properly filled.

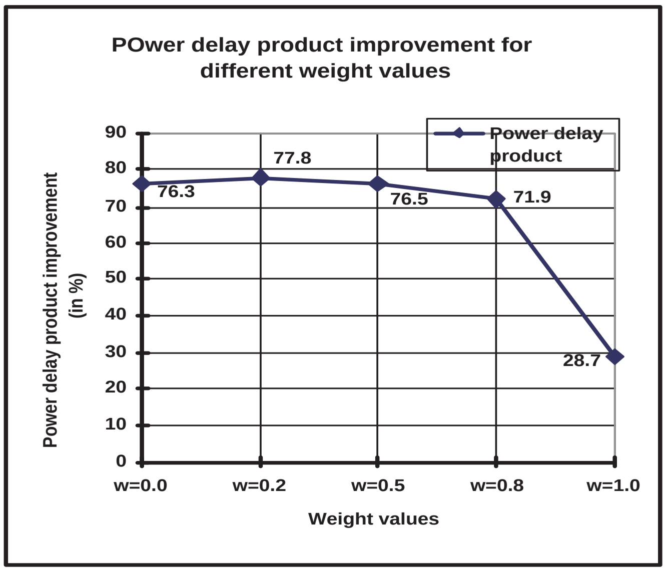


Figure 8.11: Power delay product with scan-cell ordering, test vector reordering and polarity with test vector customization with don't cares properly filled.

8.5.5 Power and Delay Minimization for Multiple Scan Chain

We have seen different power optimization strategies like scan-cell ordering, test vector reordering and scan architecture modification employed in a single scan chain. In this section, we form a multiple scan chain by dividing the single scan chain into specified number of smaller scan chains. Upon division of a single scan chain into multiple scan chains, reduction in delay and the switching activity to

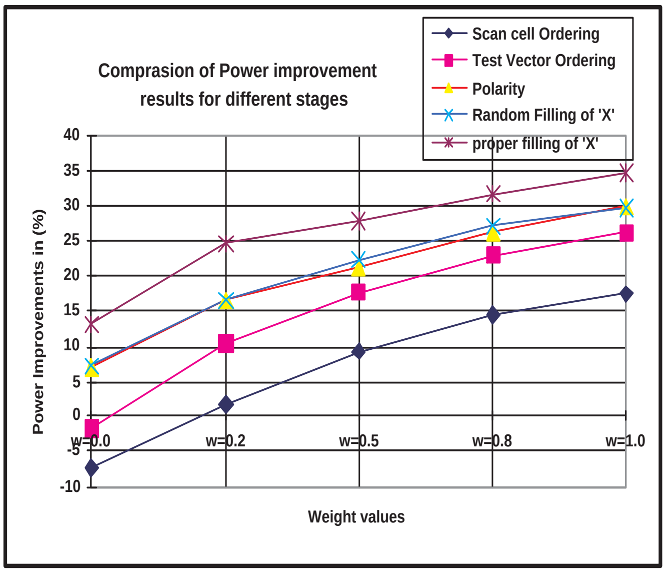


Figure 8.12: Comparison of all the results for Single Scan Chain

maximum extent can be achieved. The only technique that cannot be employed in the multiple scan chain is the test vector reordering which was feasible in single scan chain to remove the clashes that results in transitions equal to the length of scan chain (equal to the number of flip-flops in the chain). Due to the presence of many smaller scan chains, it is difficult to justify that the test vector reordering can result in minimizing the clashes in all the smaller chains.

Thus in this section, a single scan chain is broken up into specified number of smaller scan chains which results in further improvement of delay and switching activity. The results are obtained after implementing the scan-cell ordering and scan architecture modification with test vector customization.

Vectors generated by ATALANTA with random filling of don't care bits

Table 8.7 shows the results obtained for multiple scan chain for deterministic test pattern set. The average improvement in delay and power for different weights are shown with respect to the single scan chain result marked "Original" in Table 8.1. Column "Scan Chains" denotes the number of scan chains formed. It is seen from Table 8.7 that the maximum improvement in power consumption is 76.3% and the corresponding delay improvement is 88.1%. The graphs in Fig. 8.13 and Fig. 8.14 represent the results for average power and delay improvement and power-delay product respectively according to Table 8.7. It is to be noted that

Circuits		w=0.0		w=0.2		w=0.5		w=0.8		w=1.0	
	SCs	Pow	Del	Pow	Del	Pow	Del	Pow	Del	Pow	Del
s298	2	866	0.7	776	0.8	726	1.1	686	1.4	667	3.3
s344	3	793	0.9	739	0.93	690	1.23	648	2.1	620	4.4
s349	3	789	2.1	761	2.17	664	2.77	634	3.7	618	8.23
s382	3	2293	2.5	2177	2.52	1947	3.37	1786	5.65	1739	8.92
s400	3	2161	2.5	2028	2.52	1914	2.73	1756	5.63	1695	1.83
s420.1	3	2479	1.48	2178	1.48	1908	2.2	1892	2.93	1790	7.55
s444	3	2011	2.5	1850	2.56	1638	4.05	1537	7	1525	9.75
s526	3	4243	2.44	3895	2.48	3722	2.72	3506	4	3385	18
s526n	3	4431	2.44	3913	2.7	3664	3.4	3532	6.71	3408	12.5
s713	3	3089	1	2596	1.12	2439	1.31	2265	2.33	2137	7.6
s838.1	4	13800	3.76	11701	3.89	10830	5.1	9915	8.1	9644	17.13
s953	4	7399	3.5	6223	3.6	5892	4.4	5456	7.6	5275	15
s1196	3	7157	1.04	6540	1.45	6400	1.61	6107	3.54	5905	10.3
s1238	3	6944	1.04	6508	1.1	6225	2	6157	2.3	5968	10.7
s1423	4	44625	28.82	39396	30.4	37176	39.2	36275	47.3	35114	123
Avg. % Impr.		**69.1**	**97.8**	**72.1**	**97.6**	**74.1**	**96.9**	**75.5**	**95**	**76.3**	**88.1**

Table 8.7: Results for multiple scan chain for test vector generated with random filling of don't cares

though maximum improvement in power consumption is achieved for $w = 1.0$, but average PDP has decreased from 98.7% (for $w = 0.8$) to 97.1% as depicted in Fig. 8.14. Hence, $w = 0.8$ is the better choice for optimizing delay and power consumption. Though in this case, reduction in average power consumption is a bit smaller (only 0.8%), the delay improvement is much higher (almost 7%).

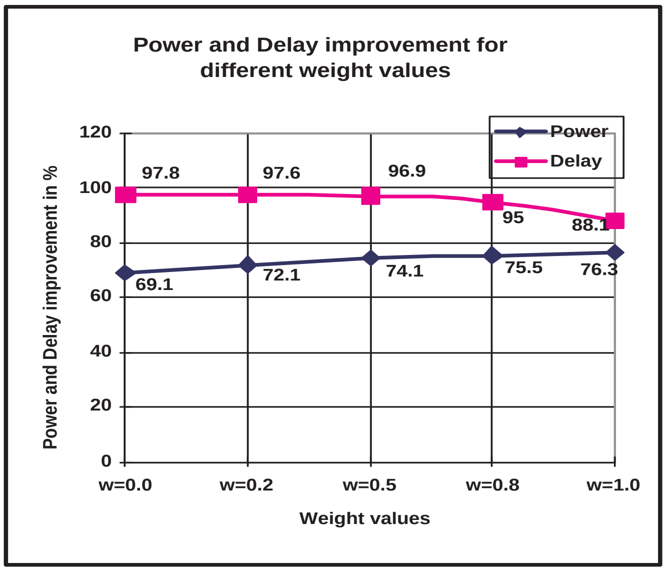


Figure 8.13: Power and delay improvement with test vector generated by ATALANTA with random filling of don't cares in multiple scan chain

Vectors generated with sustained don't cares and with proper filling

In this section the work is continued from the Section 8.3.4 for single scan chain to divide the single scan chain into multiple scan chains. Here, single scan chain

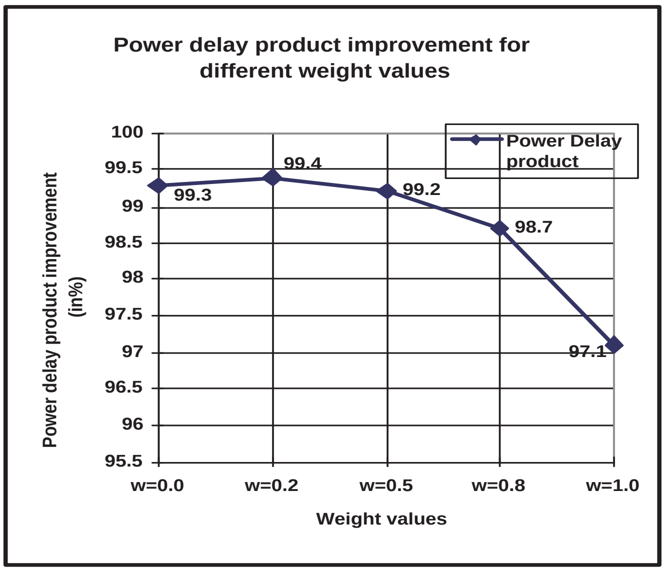


Figure 8.14: Power delay product with test vector generated by ATALANTA with random filling of don't cares in multiple scan chain

Circuits		w=0.0		w=0.2		w=0.5		w=0.8		w=1.0	
	SCs	Pow	Del	Pow	Del	Pow	Del	Pow	Del	Pow	Del
s420.1	3	2496	1.5	2132	1.7	1982	2.3	1927	2.9	1836	7.8
s526	3	4491	2.4	4231	2.5	3973	3.9	3943	4.1	3775	16
s526n	3	4454	2.4	4005	2.5	3798	3.1	3617	5.1	3420	14
s713	3	3263	1	2676	1.2	2444	1.8	2274	3.4	2293	6.2
s838.1	4	12032	3.8	9453	4.1	9007	4.2	8620	8.2	8140	17.2
s953	4	7751	3.5	6279	3.54	5632	4.22	5359	5.9	4961	14.1
s1196	3	6540	1.04	6407	1.05	6019	1.4	5861	2.4	5605	10.9
s1238	3	6411	1.04	6072	1.2	5787	1.5	5630	2.3	5386	6.5
s1423	4	43887	28.81	37626	30.5	36867	35.1	33434	52.4	32018	137.1
Avg. % Impr.		**71.6**	**98.1**	**75**	**97.9**	**76.5**	**97.3**	**77.5**	**96.1**	**78.4**	**88.8**

Table 8.8: Results for multiple scan chain for test vector generated with don't care sustained and properly filled

is divided into multiple scan chains and then the power reduction strategies are incorporated. Don't cares are filled properly for each smaller scan chain and the scan architecture modification is carried out for each scan chain. Here, every small scan chain is considered to be like a single scan chain and the number of transitions and delay are calculated for each. Thus, the overall number of transitions in a multiple scan chain is the sum of the transitions in each smaller scan chains and the test time of multiple scan chain is the test time of the longest scan chain amongst the multiple scan chains. Proper filling of don't cares result in further reduction in switching activity.

Table 8.8 shows the results obtained for power dissipation for different weights ranging from 0 to 1 together with scan cell reordering and scan architecture modification. Average improvements are obtained with respect to the results noted

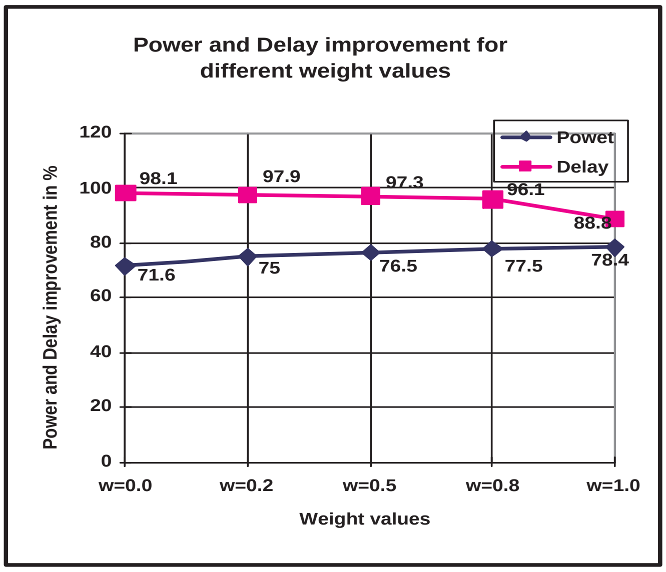


Figure 8.15: Power and delay improvement with test vector generated with don't cares sustained and properly filled in multiple scan chain

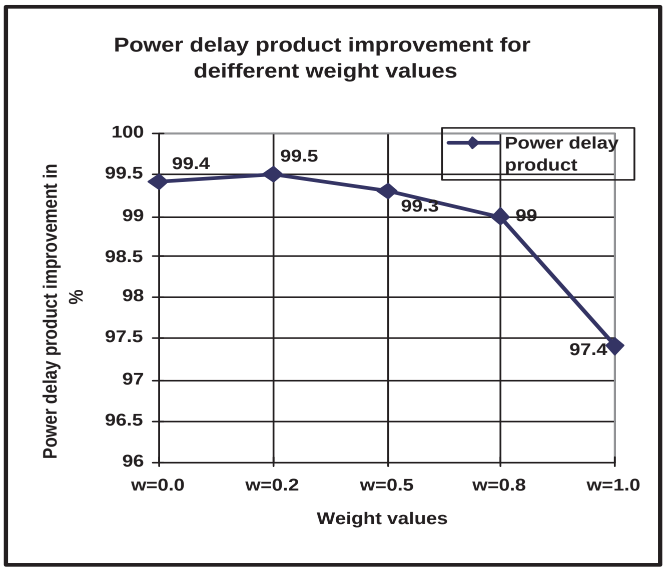


Figure 8.16: Power Delay Product with test vector generated with don't cares sustained and properly filled in multiple scan chain

in Table 8.1. Table 8.8 shows that for weight $w = 1.0$, on an average upto 78.4% power consumption can be reduced with delay upto 88.8%, whereas 98.1% delay can be improved for $w = 0.0$. Fig. 8.15 summarizes Table 8.8 and shows power consumptions and delay improvements for different weights. It is also to be noted from Fig. 8.16 that average improvement in PDP has been obtained upto 99%

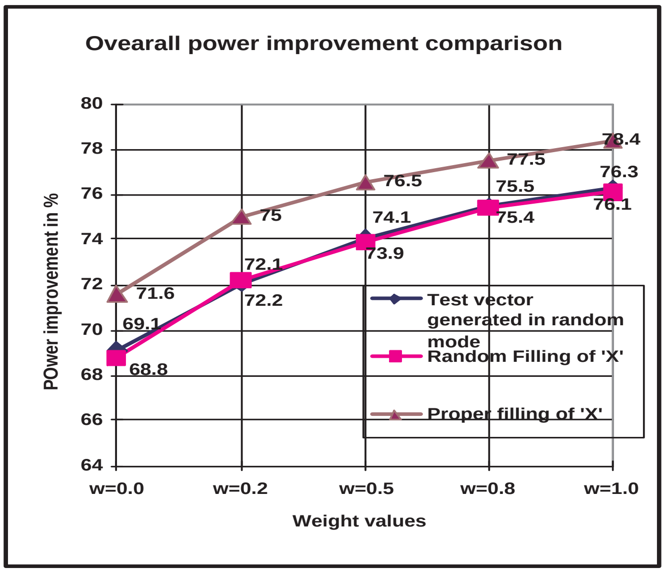


Figure 8.17: Overall Comparison of power improvement results for Multiple Scan Chain

for $w = 0.8$.

The graph shown in Fig. 8.17, gives a summary of all the results corresponding to the multiple scan chain, which include scan-cell ordering, test vector reordering, scan architecture modification, and proper filling of don't cares respectively.

8.6 Conclusion

In this work, we have presented a power minimization method using Genetic Algorithm which considers weighted sum of both the delay and transitions occurring at the time of scanning in/out of the test vector/responses. The previous works considered either delay or transition minimization only. It provides an effective trade-off to the test engineer.
In next chapter, a novel test architecture modification scheme has been proposed using D and T flip-flop combinations. The main target in this scheme is to reduce the scan power.

Chapter 9

Scan Architecture Modification for Power Reduction

9.1 Introduction

In Chapter 8 we have seen different schemes for reduction of power during shift-in of the input next test stimuli and shift-out of the test response. These operations create many toggles in scan cells, which in turn affect the circuit power. Decreasing the number of transitions during shifting of the input vectors and output responses that occurs inside the scan cells can reduce test power dissipation. Shifting in/out the complementary values in consecutive clock pulse results in toggling within each scan cell through which complementary bits are to be passed. Hence high correlation exists between the consecutive stimuli bits with the scan chain transitions.

In traditional scan-based test, scan chains are designed by D flip-flips with additional hardware to provide normal and test mode operation by multiplexer (MUX). In the proposed methodology, we modify the scan chain so that during the scan operation, some of the flip-flops get converted into T-flip-flop by insertion of XOR gate. The logic gates inserted impact only the scan path, keeping intact the design functionality and the critical paths. This also transforms the stimulus to be inserted to the scan flip-flops over numerous shift cycles. The proposed scan chain modification and the consequent test vector transformation together with test vector reordering can thus be utilized to reduce the number of scan chain transitions in the stimuli to be inserted, reducing the test power dissipation by significant amount.

The chapter is organized as follows. Section 9.2 describes the proposed scan architecture, test stimuli transformation and test vector reordering techniques. Section 9.3 gives the Genetic algorithm formulation for D-T flip-flop selection for scan architecture design. Formulation also considers test vector reordering together with D-T flip-flop selection. Experimental results are presented in Section 9.4 and 9.5 draws the conclusion.

9.2 Optimized Scan Architecture, Test Stimuli Transformation and Test Vector Reordering

In this section we describe our proposed method of modifying the scan chain so that total number of transitions is reduced during scan in/out. Our approach fundamentally consists of three steps, but can be subdivided into four steps as shown in Fig. 9.1. At the end, we obtain scan cell architecture and a completely specified test vector sequence which results in significant reduction in power, without changing fault coverage, however at the cost of a small area overhead. The control flow of algorithm can be gathered from Fig. 9.1.

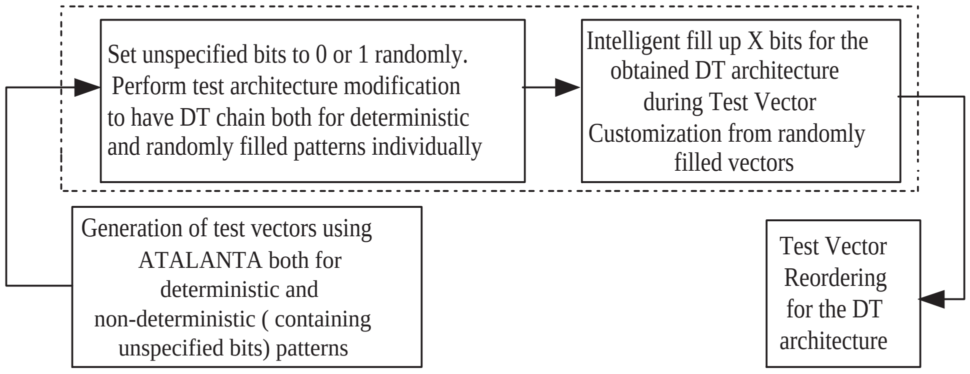

Set unspecified bits to 0 or 1 randomly. Perform test architecture modification to have DT chain both for deterministic and randomly filled patterns individually
Intelligent fill up X bits for the obtained DT architecture during Test Vector Customization from randomly filled vectors
Generation of test vectors using ATALANTA both for deterministic and non-deterministic (containing unspecified bits) patterns
Test Vector Reordering for the DT architecture

Figure 9.1: Control Flow of the work

In traditional scan chain architecture D flip-flops are used. We made a simple modification in it by choosing a combination of D and T flip-flops in the scan chain so that total number of transitions for scanning in/out of the test stimuli is less. However, since modifying flip-flop type may affect circuit's normal functionality, we have added one extra XOR-gate to convert the D-flip-flop to T-flip-flop during scan as shown in Fig. 9.2(c). Next step deals with customization of the test vectors for fully and partially specified test vectors for the modified scan architecture. In final step we do a test vector re-ordering that does the job of reducing total number of clashes and consequently the overall number of transitions.

9.2.1 Test Architecture Modification

Consider an example as shown in Fig. 9.2. Scan chain length is 4 and we have to scan in the pattern 1101. Initial content of the flip-flops is considered to be 0000. In traditional scan chain, as shown in Fig. 9.2(a), the total number of transitions occurring is 9. But if we use a judicious combination of D and T flip-flops (like Fig. 9.2(b)), the total number of transitions will be 5. Here it can be noted that original test vector is modified such that after scan-in, it takes the original form. For the given example, 1111 after being scanned into the new scan cell architecture becomes the originally desired vector 1101. Such an adaptation is necessary because of the modified scan cell architecture.

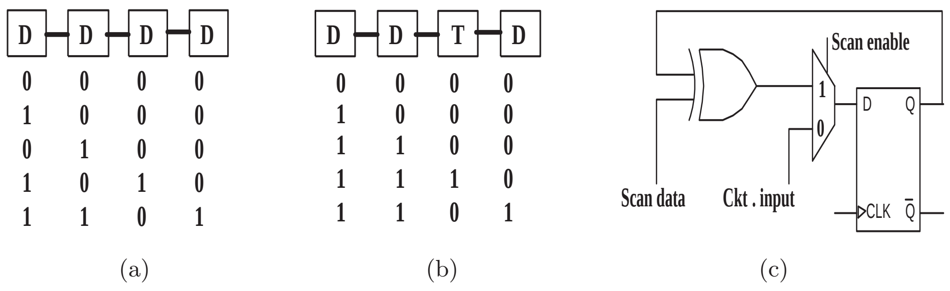


Figure 9.2: (a)Traditional Scan Architecture (b) Modified Scan Architecture (c) Converting D-flip-flop into T-flip-flop by adding one XOR gate for scan operation

So, our aim is to get a judicious combination of D and T flip-flops that yields minimum number of transitions for scan-in of test vectors and to scan-out their responses. Hence for a given circuit we have to consider all the test stimuli and their responses together and obtain optimized scan cell architecture. If the scan chain length is N, it is possible to make 2^N combinations of D and T flip-flops. For smaller values of N, it is possible to find total number of transitions for every combination of D and T flip-flops and select the combination resulting in the minimum number of transitions. But for large values of N, it is difficult. So, for getting power optimized scan cell architecture, we used a genetic algorithm based formulation as discussed in Section 9.3.

9.2.2 Test Stimuli Transformation

This step is responsible to compute the modified set of test vectors based on D-T flip-flop selection. This step is important since we want the test vectors to be in their original form after complete scan-in. In this section, we propose algorithms to do the transformation for the fully specified and incompletely specified pattern sets. First we propose the procedure to be followed for the fully specified set, which will next be extended for patterns with don't cares.

For Deterministic Pattern Set

Algorithm 9.1 shows how the modified test stimuli are to be obtained to get back the original test vector after scanning in the modified test stimuli. Here we assume that the vectors are completely specified, they do not posses any unspecified don't care bits.

In this algorithm (Algorithm 9.1) we consider the number of flip-flops in the scan chain to be N. We consider the index of the flip-flop starting with 0 to $(N-1)$ from left to right. To scan-in an input vector of length N bit (equal to scan chain length), N number of clock pulses are needed. FF is a two dimensional array, where FF_{ij} contains the intermediate values for the j^{th} flip-flop at i^{th} clock tick. FF_0 holds the previous response vector. It can be noted that complexity of the algorithm is $O(N^2)$. For the given example in Fig. 9.2(b), scan chain length is 4 and it contains both the D and T flip-flops in scan mode. Four clock pulses are needed to scan-in the vector 1101. Using *Step 1* we can find out the intermediate values of the flip-flops in each clock cycle. According to *Step 2*, we get the modified test vector to be shifted in to the first flip-flop (type D) of the scan chain is 1111. *Step 3* determines the total number of transitions occurring to scan in the modified vector 1111 through the scan chain. This step of the algorithm counts the number of 0-to-1 or 1-to-0 pairs in every column after scanning-in of the modified vector 1111, which in this case turns out to be 5.

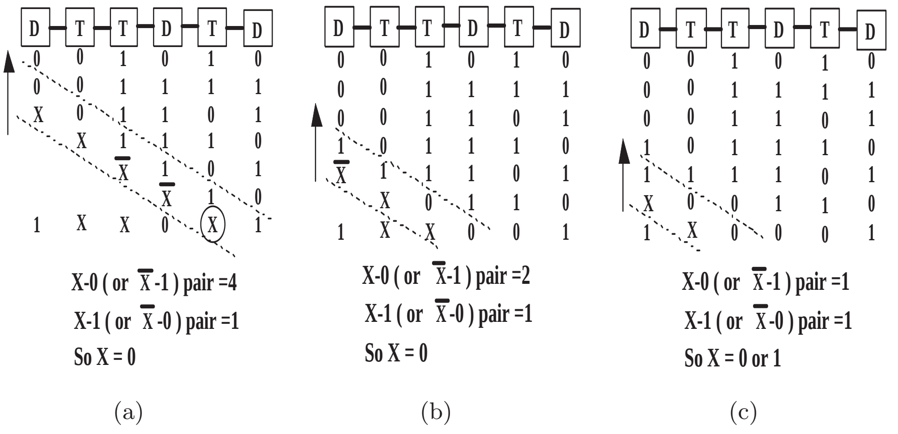

D T T D T D
X-0 (or $\overline{X}$-1) pair =4
X-1 (or $\overline{X}$-0) pair =1
So X = 0
X-0 (or $\overline{X}$-1) pair =2
X-1 (or $\overline{X}$-0) pair =1
So X = 0
X-0 (or $\overline{X}$-1) pair =1
X-1 (or $\overline{X}$-0) pair =1
So X = 0 or 1
(a)
(b)
(c)

Figure 9.3: Don't care filling with test vector customization

For Non-deterministic Pattern set

A major feature of the Automated Test Pattern Generators (ATPGs) is that they leave a large number of bits unspecified in the pattern set generated. A major source of dynamic power saving can be achieved by judiciously filling up the unspecified bits.

Algorithm 9.1: Test Vector Customization

```
Input :
    a) Modified scan chain architecture i.e. combination of D-T flip-flops.
    b) Original test vector.
Output: Modified scan-in vector
/* $FF[N+1, N]$ is a 2-dimensional array, $FF[0]$ holds the response
   of previous vector                                                  */
begin
    Step 1:
    /* Determine intermediate values or states of the flip-flops       */
    for ($i \leftarrow 1$ to $N-1$) do
        for ($j \leftarrow i$ to $N-1$) do
            if (Flip-flop$_j$ is D ) then $FF_{ij} = FF_{(i-1)(j-1)}$;
            else $FF_{ij} = FF_{(i-1)(j-1)}$ XOR $FF_{(i-1)j}$;
        end
    end
    for ($i \leftarrow N-1$ downto $0$) do
        for ($j \leftarrow N-1$ downto $1$ ) do
            if (Flip-flop$_j$ is D ) then $FF_{i(j-1)} = FF_{(i+1)j}$;
            else $FF_{i(j-1)} = FF_{(i+1)j}$ XOR $FF_{ij}$;
        end
    end
    Step 2:
    /* Generate modified Input vector                                  */
    if (First flip-flop in the chain is a D flip-flop) then
        for ($j \leftarrow N$ downto $1$ ) do
            Print values of $FF_{j0}$;
        end
    end
    else
        for ($j \leftarrow N$ downto $1$ ) do
            Print values of $FF_{j0}$ XOR $FF_{(j-1)0}$;
        end
    end
    Step 3:
    /* Determine total number of transitions for scanning in of the
       input vector                                                    */
    $TRANSITIONS = 0$;
    for ($j \leftarrow 0$ to $(N-1)$) do
        for ($i \leftarrow 0$ to $(N-1)$) do
            if ($FF_{ij} \neq FF_{(i+1)j}$) then
            $TRANSITIONS = TRANSITIONS + 1$;
        end
    end
end
```

Algorithm used for customization of the vectors with don't care sustained is same as the Algorithm 9.1 discussed in the previous section for test vector customization of deterministic test patterns, excepting *Step 1.* The proposed algorithm handles the response of previous vector and next input vector at a time. It is assumed that response of the previous vector is fully specified. The example in Fig. 9.3 illustrates the proposed scheme and later we will provide the scheme in the form of an algorithm for modifying *Step 1.* Don't cares in the input vector is denoted as X. $\overline{X}$ denotes the complement of X. We handle one don't care (X) bit at a time in the algorithm to specify it either as 1 or as 0. The proposed algorithm is based on the count of $'X$-to-$1'$ pair (hence $\overline{X}$-to-0 pair) and $'X$-to-$0'$ pair (hence $\overline{X}$-to-1 pair) for a particular X. Next, we look into an example illustrating the don't care fill-up approach.
Example: Consider the scan chain (D-T-T-D-T-D) containing 6 flip-flops obtained after modification of scan chain. Assume that the response to the previous test vector is 001010. Thus, the scan elements are currently holding these values. Let the next input vector to be customized be $1XX0X1$. After each iteration of algorithm, the intermediate values of the flip-flops are shown in the Fig. 9.3. Let us consider the filling of first X circled in Fig. 9.3(a). In each iteration, we have to count the number of X-to-0 (or $\overline{X}$-to-1) pair and the number of X-to-1 (or $\overline{X}$-to-0) pair within the diagonally marked zone. From Fig. 9.3(a), (b) and (c) it is seen that the number of X-to-0 (plus $\overline{X}$-to-1) pair is 4 whereas X-to-1 (plus $\overline{X}$-to-0) pair 1. Hence if we consider X as 1 it will inject 4 transitions. Setting $X = 0$ we can reduce these 4 transitions. So whichever pair will be greater, we have to take decision in its favor. Fig. 9.3(b) and 9.3(c) shows don't care filling for other positions.

So, here we can see that after don't care filling test vector $1XX0X1$ becomes 100001. Also, for the given example, 101100 after being scanned into the scan cell architecture becomes original test vector 100001. Algorithm 9.2 shows the modified *Step*1 to fill-up don't cares in a partially specified test set.

9.2.3 Test Vector Reordering

A different test vector ordering may result in different power consumption during testing. In our proposed method, test vector ordering is based on the number of internal transitions. So the problem to be addressed here is *"for a set of input vectors $T = t_1, t_2, t_3, ..., t_n$ find an optimal vector ordering $< s_1, s_2, ..., s_n >$ of T such that total number of transitions are minimized"*. The proposed test pattern reordering algorithm is shown in Algorithm 9.3.

Algorithm 9.2: Don't care filling

```
Input :
    a) Modified scan chain architecture i.e. combination of D-T flip-flops.
    b) Original test vector with don't cares.
Output: Completely specified test vector.
1  begin
2      Modified Step 1:
       /* Determine intermediate values or states of the flip-flops */
3      for ($i \leftarrow 1$ to $N-1$) do
4          for ($j \leftarrow i$ to $N-1$) do
5              if (Flip-flop$_j$ is D ) then $FF_{ij} = FF_{(i-1)(j-1)}$;
6              else $FF_{ij} = FF_{(i-1)(j-1)}$ XOR $FF_{(i-1)j}$ using rules $X$ XOR $0 \rightarrow X$,
               $X$ XOR $1 \rightarrow \overline{X}$ and $X$ XOR $X \rightarrow 0$;
7          end
8      end
9      for ($k \leftarrow 2$ to $N$) do
10         $X - to - 1\_pair \leftarrow 0$;
11         $X - to - 0\_pair \leftarrow 0$;
12         $i \leftarrow N-1$;
13         $j \leftarrow N-k$;
14         while ($i > k-2$ AND $j \geq 0$) do
15             if (Flip-flop$_{j+1}$ is D) then $FF_{ij} \leftarrow FF_{(i+1)(j+1)}$;
16             else $FF_{ij} \leftarrow FF_{(i+1)(j+1)}$ XOR $FF_{i(j+1)}$;
               /* Considering two adjacent lower sub-diagonals of FF
                  matrix (intermediate state matrix) */
17             if ((($FF_{ij} = X$) AND ($FF_{(i-1)j} = 1$)) OR (($FF_{ij} = \overline{X}$) AND
               ($FF_{(i-1)j} = 0$))) then $X - to - 1\_pair + +$;
18             if ((($FF_{ij} = X$) AND ($FF_{(i-1)j} = 0$)) OR (($FF_{ij} = \overline{X}$) AND
               ($FF_{(i-1)j} = 1$))) then $X - to - 0\_pair + +$;
19             Decrement $i$ and $j$;
20         end
           /* Assignment of X as 1 or 0 depending upon count X-1 and
              count X-0 pair */
21         if ($X$-to-1_pair $\geq$ $X$-to-0_pair) then $FF_{N(N-k+1)} \leftarrow 1$;
22         else $FFN_{(N-k+1)} \leftarrow 0$;
23     end
24 end
```

Algorithm 9.3: Test Vector Reordering

```
Input  : T= A set of test stimuli with their response.
         /* Assume initially scan elements are set to clear.      */
Output: Ordered set of test stimuli with their response.
1 begin
2     Step 1:Initialize $S = \Phi$;
3     Step 2: Find test stimuli $t_x \epsilon T$ to get the optimal internal scan transitions
      using test vector customization algorithm;
4     Step 3: Set $S = S \cup \{t_x\}$ and $T = T - \{t_x\}$;
5     Step 4:Set initial content of the scan elements to response of the vector $t_x$;
6     Step 5:Repeat step $2 - 4$ until $T = \Phi$;
7 end
```

9.3 Genetic Algorithm Formulation

In this work GA formulation is used to get the modified scan architecture using D and T flip-flop for reducing transitions. In order to solve the concerned problem we have to define individual representation of chromosomes, mutation and crossover operators, fitness function and selection procedure. Following sections describe the GA formulation.

It may be noted that ideally the steps of flip-flop selection, don't care fill-up and test vector reordering be integrated together. However, to reduce the complexity, we have solved the following two versions of the problem.

- With D-T selection and test vector ordering for deterministic test set.
- With D-T selection for test set with don't cares. This is augmented by don't care filling and test vector reordering separately.

9.3.1 GA formulation for D-T flip-flop selection with test vector reordering

For current problem, the chromosome structure has two parts. First part is a set of integer values ranging from 1 to P, where P is the number of input test vectors to be applied to the scan chain. The sequence of numbers represents the ordering of the input vectors. Second part also consists of N number of integer values $p_1, p_2, p_3, ..., p_n$, where p_i's are either 0 or 1. In our problem 0 denotes a D flip-flop and 1 a T flip-flop. For example, if the chromosome is $<< 1, 2, 5, 6, 3, 7, 4, 8 >< 010 >>$, the scan chain contains three flip-flops of type D, T and D respectively. The input vectors are to be applied in the order of $1, 2, 5, 6, 3, 7, 4$ and 8.

The fitness function is the total number of transitions for that particular chromosome. The total number of transitions have been calculated using the Algorithm 9.1 discussed in Section 9.2.

We select 20% of the population as the best class chromosomes after sorting them based on fitness. From the entire population we select two chromosomes randomly for participating in crossover operation. Two new chromosomes are produced after performing crossover operation around two randomly selected crossover points, one each on individual parts of the chromosome.

Mutation operation randomly selects a chromosome from the population and modifies the value at some positions of the chromosome. For example, for first part of the chromosome, two random index points are selected and then the values of these two selected points are swapped. For second part of the chromosome, randomly any index point is selected and its value is inverted (i.e., 0 to 1 or 1 to 0).

9.3.2 GA formulation for D-T flip-flop selection without test vector ordering

For this problem chromosome structure is encoded with binary bit strings $< p_1, p_2, ..., p_N >$ of length N depending on the number of flip-flops present in the scan chain, where p_i's are either 0 or 1.
Here fitness value is the number of transitions incurred for that particular scan chain and test data set for the corresponding circuit. This fitness value is calculated using the test vector customization algorithm for pattern set with don't care sustained as proposed in Section 9.2.

To create the population for new generation, 20% best-fit chromosomes are directly copied and remaining 80% chromosomes are created using crossover and mutation. Crossover and mutation operations are done as discussed in the previous formulation. The algorithm is run for 100 generations with population size of 100 to 500, depending on the number of flip-flops in the circuit. Chromosome with lowest number of transitions provides the optimal solution for our problem.

9.4 Experimental Results

The complete algorithm has been implemented in C and executed in a 2.66GHz P-IV machine with 512 MB memory and verified on ISCAS89 benchmark circuits [186]. The test patterns are generated using ATALANTA [139].

Circuit	**#FF**	**#PAT**	**TA**	**MA+TVO**	**No. of T FFs**	**% Impr.**	**Area Overhead in %**
s510	6	122	1030	781	3	24.17	4.16
s1488	6	250	1972	1529	2	22.46	2.78
s349	15	48	2744	2150	7	21.65	3.89
s344	15	48	2448	2122	6	13.32	3.33
s298	14	69	2758	2375	4	13.89	2.38
s444	21	64	6138	5270	4	14.14	1.58
s400	21	68	6728	5790	5	13.94	1.98
s382	21	72	6991	5729	4	18.05	1.58
s713	19	114	8075	6716	4	16.83	1.75
s420	16	143	7044	6067	2	13.87	1.04
s526	21	128	12667	11170	6	11.81	2.38
s526n	21	132	13635	11543	6	15.34	2.38
s953	29	180	28541	25359	1	11.06	0.29
s1196	18	294	23085	20694	4	10.35	1.85
s1238	18	296	23577	20684	3	12.27	1.39

Table 9.1: Power consumption results for ISCAS89 circuits

9.4.1 Results for deterministic pattern set

The results obtained for our algorithm is based on the random filling of the don't care bits for vector-response pairs of the circuit. Results are shown in Table 9.1 for traditional scan architecture and the modified architecture.

Column **TA** lists the total number of transitions for traditional architecture. On the other hand, the column marked **MA+TVO** lists the total number of transitions for modified architecture with test vector reordering. The immediate next column shows the % improvement obtained in the case of GA. Maximum improvement 24.17% is obtained for the circuit s510. On an average 16% reduction in switching activity is obtained considering all the circuits. The area overhead is shown in the column for insertion of extra gates to make T flip-flops. For area overhead estimation, we have computed the 2-input gate equivalents for the scan chains. For a D-flip-flop, the gate count has been taken as 12 [201, 202]. As it is seen from the table, area overhead is very less.

9.4.2 Results for pattern set with don't cares

Table 9.2 shows the result for proposed scan architecture. Test vectors with sustained don't cares are used. First, don't cares are filled randomly and this pattern set is used to determine the scan architecture consisting of D and T flip-flops using GA. Based on this scan architecture we determine the optimal don't care assignment for the test pattern set using algorithm proposed in Section

9.2. Column **TA** shows the number of transitions for the traditional scan chain architecture where don't care bits are filled randomly. Column **DT** represents the number of transitions obtained by modifying the scan architecture using D and T flip-flops. This is obtained using GA and in this case also test vectors used are same as the traditional scan architecture. But if don't care bits are filled with the proposed algorithm, the number of transitions will be less as shown in column **DT+ X's Filling**. Again, if reordering of the test vectors is performed on the resulting test sets further reduction of transitions is possible as shown in column **DT+X's Filling+ Reorder**. It can be noted that for the proposed scan architecture together with proper filling of don't care bits in test vectors and their reordering, 34% improvement in transitions can be achieved for circuit s15850 which has 534 flip-flops. For larger circuits, saving in number of transitions is also significant. As it is seen from the table, area overhead is quite small. For the sake of comparison, we have noted results of [155]. It may be noted that [155] uses inverters between scan cells with don't care filling and test vector reordering. Test patterns used in our tool and [155] have been generated using similar procedure (using ATALANTA - X option). As noted in the table, for most of the circuits there are significant reductions in transition counts.

Circuit	**#PAT**	**TA**	**DT**	**DT+X's Filling**	**DT+ X's Filling+ Reorder**	**#T FFs/ #D FFs**	**% impr.**	**% impr.** [155]	**Area over-head in %**
s420	100	3432	3352	2972	2515	3/16	26.17	N/A	1.562
s526	128	12425	11943	11308	10708	4/21	13.81	14.06	1.587
s713	110	7968	7430	6497	6323	1/19	20.64	13.05	0.443
s953	180	28052	25938	19883	19323	4/29	31.11	13.65	1.149
s1238	300	23418	22066	18714	17884	1/18	23.63	15.17	0.462
s1423	130	156237	152267	140110	137513	5/74	11.98	N/A	0.563
s5378	500	3549420	3446298	2969128	2956048	73/179	16.71	N/A	3.398
s9234.1	730	8094195	7997039	6012308	5945861	27/211	26.54	N/A	1.106
s15850	800	56453668	54934994	37057074	36989582	27/597	34.47	21.59	0.376
s13207	900	91687648	90799928	69327703	69091600	80/669	24.64	N/A	0.996
s38584	1300	658302233	651433007	529927872	519914756	405/1452	21.02	N/A	2.324
s38417	1800	1160101317	1136899290	858474974	852674466	409/1636	26.50	N/A	2.083

Table 9.2: Power consumption results for ISCAS89 circuits with proper filling of don't cares and test vector reordering

9.5 Conclusion

Power efficiency of devices is a must in the current trend of sub-micron technology. During testing formation of scan chain by interconnecting flip-flops plays a major role in the power (dynamic) consumption. We have proposed a mechanism for forming the scan chains by introducing extra XOR gate at selected places, converting D flip-flop to T flip-flop. This architecture shows upto 34% reduction in switching activity within circuit. The approach does not involve, at any point, reordering of the of scan cells, which might be used for future work. This enhancement coupled with flip-flop selection is expected to produce much better saving in test power.

Chapter 10

Conclusion and Scope of Future Works

This dissertation tackles different testing challenges and in every case tries to optimize the test cost in terms of time, area and power. The following are the main contributions of the works carried out in the thesis.

In Chapter 3, an automated tool has been developed that can assist the test engineers to select the cost-effective test architecture and schedule for the embedded cores in the SOC. To get the cost-effective test architecture, both balanced and unbalanced wrapper scan chain design has been considered. An efficient best-fit heuristic based rectangle packing approach is considered to get the scheduling of the cores to optimize the test application time. Our proposed approach shows better results over the contemporary approaches. Test time results are within 3% of the lower bounds reported in [70]. The proposed approach is better than ACO[77] for 60% of the cases (19 out of 28), more than 50% of the cases compared to 2-Stage GA[78], more than 20% of the cases compared to B*-SA[76], more than 40% of the cases for EA(c)[62], more than 60% of the cases for EA(nC)[62] and almost 50% of the cases compared to SA[47]. The packing algorithm can also take care of the power constraints during scheduling.

Motivated by the fact that the ATE machines consist of port-scalability features, in Chapter 4, we have addressed the problem of test scheduling with dual speed TAM architecture. TAM width partitioning scheme has been developed and dual speed TAM architecture optimization has been accommodated with that. The maximum improvement in test application time has been found to be 40.99% for p93791 with a TAM width of $W = 32$ among which 25% of the TAM lines are high speed. Effect on test power has also been studied resulting from the use of two different speeds for different channels and results show maximum average improvement of 40.05% in test time.

In Chapter 5, test scheduling algorithm for hierarchical cores has been developed using TAM width partitioning scheme. The wrapper has been designed for child cores and the parent core of a hierarchical core efficiently in a cost effective way. For non-interactive design scenario it is noted that upto 31.19% improvement in test time can be achieved with the proposed method. Whereas, for interactive scenario, the maximum improvement of 34.23% has been obtained.

To handle large volume of test data, a dictionary based compression scheme using Huffman coding has been proposed in Chapter 6. On an average, our scheme results in compression 0.78% more than [6], 7.8% more than VIHC [7] coding, 24.97% more than Golomb [90] coding, 13.76% more than RL-Huffman [93] coding, 14.16% more than the selective Huffman coding (SHC) [84], 4.29% more than V9C coding [87], 9.5% more than MDC [129] coding, 13.8% more than arithmetic coding [103], and 6% more than data independent pattern run-length (PR) [98] coding. A scheme has also been proposed for the test engineers to select the amount of compression, keeping in mind the constraints like power and ATE memory depth. A test bus power reduction strategy has been proposed by proper assignment of 0 and 1 to the branches of the Huffman tree as well as by modifying the Huffman tree. The highest improvement of 3.74% in internal flip count by proper assignment of 0 and 1 to the branches of Huffman tree is obtained. Whereas, by modifying the Huffman tree maximum reduction in flip count obtained is 26.38% by sacrificing only 1.91% compression ratio.

In Chapter 7, a non-dictionary based coding scheme called Split-VIHC (SVIHC) has been developed that improves upon the well known VIHC scheme. The proposed scheme improves upon the compression ratio as well as test application time by modifying the decoder architecture. It can be noted that compression ratio is increased by 2-6% and almost 30% improvement in test application time from the original VIHC compression[7]. Though we have not studied out the compression and scan power trade-off, but it can be considered as a future work for the same.

Another test challenge, that is always a major concern, is test power consumption during testing. In Chapter 8, simultaneous reduction in power and delay has been considered by introducing different strategies like scan cell reordering, test vector reordering, scan architecture modification and proper filling of unspecified bits. This reduction is considered both for single and multiple scan chains.

For single scan chain, maximum average power improvement 34.8% is obtained, whereas the delay has increased for this case upto 6.7%. On the other hand for multiple scan chains, on an average 78.4% power consumption can be improved with delay upto 88.8%.

Finally, a novel scan architecture using D and T flip-flops has been proposed and discussed in Chapter 9. Two heuristic algorithms are proposed for proper filling of the unspecified bits and test vector reordering. The filling of unspecified bits is done during scan-vector customization for modified scan architecture. For the proposed scan architecture together with proper filling of don't care bits in test vectors and their ordering, 34% improvement in transitions can be achieved.

Due to the reduction in feature size, the necessity to identify new faults, application of test vectors at high speed and also for increased clock domain, testing SOCs become critical. The methodologies proposed in this dissertation may provide a possible solution that accounts the above requirements. But still, several emerging testing challenges exist for future work. The following may be some such directions.

1. For reducing the cost of SOC testing, both test architecture optimization and test vector compression can be integrated together in a power optimized manner.

2. How to effectively test high speed interconnect logic for signal integrity with the existing DFT logic.

3. Due to increased density, thermal aware design and test are assuming importance significantly. Thus, to make the test schedule temperature aware is a major concern.

4. In higher degree of technology scaling, leakage power is contributing more to the total system power consumption. Thus, all power estimation and optimization strategies need to be revisited to incorporate the effect of leakage as well.

5. On the architecture side, new on-chip network based communication is going to replace the bus-based system-on-chip design. Designing efficient power-aware techniques for testing such designs exploiting the available on-chip communication back-bone is a real challenge. Though some works have already started in this line, there are lot of problems still to be explored.

References

[1] R. K. Gupta and Y. Zorian. Introducing Core-Based System Design. *IEEE Design and Test of Computers*, 14(4):15–25, December 1997. [cited at p. xi, 1, 2, 3]

[2] Y. Zorian, E. J. Marinissen, and S. Dey. Testing Embedded Core-Based System Chips. *IEEE Computer*, 32:52–60, June 1999. [cited at p. xi, 3, 4, 5, 6, 14, 29]

[3] K. Chakrabarty. Low-Cost Modular Testing and Test Resource Partitioning for SOCs. *IEE Proc. Digiatl and Computer Techniques*, 152(3):427–441, May 2005. [cited at p. xi, 14]

[4] A. Sehgal and K. Chakrabarty. Optimization of Dual-Speed TAM Architectures for Efficient Modular Testing of SOCs. *IEEE Trans. on Computers*, 56(1):120–133, January 2007. [cited at p. xi, xiv, 20, 48, 56, 57, 58, 59, 60, 61]

[5] A. Sehgal, S. K. Goel, E. J. Marinissen, and K. Chakrabarty. IEEE P1500-Complaint Test Wrapper Design for Hierarchical Cores. In *Proc. International Test Conference*, pages 1203–1212, 2004. [cited at p. xi, 18, 63, 64, 66, 74]

[6] L. Li, K. Chakrabarty, and N. A. Touba. Test Data Compression Using Dictionaries with Selective Entries and Fixed-Length Indices. *ACM Trans. on Design Automation of Electronic Systems*, 8(4):470–490, October 2003. [cited at p. xi, 23, 80, 81, 82, 83, 84, 85, 86, 89, 90, 164]

[7] P. T. Gonciari, B. Al-Hashimi, and N. Nicolici. Variable-Length Input Huffman Coding for System-on-a-Chip Test. *IEEE Trans. on CAD*, 22(6):783–796, June 2003. [cited at p. xii, 10, 22, 24, 80, 85, 86, 111, 112, 114, 115, 117, 118, 120, 122, 123, 124, 125, 126, 164]

[8] M. Kneating and P. Bricaud. *Resuse Methodology Manual for a System-on-a-chip Design*. Kluwer Academic Publishers, Norwell, Massachusettes, 1999. [cited at p. 1]

[9] VSIA Alliance, http://www.vsi.org. *Virtual Socket Interface Architectural Document*, November 1996. [cited at p. 2]

[10] B. T. Murray and J. P. Hayes. Testing ICs: Getting to The Core of The Problem. *IEEE Computer*, 29(11):32–38, November 1996. [cited at p. 3]

[11] L. Whetsel. Core Test Connectivity, Communication and Control. In *Proc. International Test Conference*, pages 303–312, October 1999. [cited at p. 3]

[12] L. Whetsel. Optaddressable Test Ports :An Approach to Testing Embedded Cores. In *Proc. International Test Conference*, pages 1055–1064. IEEE Computer Society Press, September 1999. [cited at p. 3]

[13] H. Vranken, T. Waayers, H. Fleury, and D. Lelouvier. Enhanced Reduced Pin-count Test for Full-scan Designs. *Proc. International Test Conference*, pages 738–747, 2001. [cited at p. 3, 80]

[14] R. Kapur. Strategies for Low Cost Test. *IEEE Design and Test of Computers*, 18(6):47–54, December 2001. [cited at p. 3]

[15] J. Rajaski. DFT for High-Quality Low Cost Manufacturing Test. In *Proc. Asian Test Symposium*, pages 3–8. IEEE Computer Society Press, November 2001. [cited at p. 3]

[16] J. Bedsole, R. Raina, A. Crouch, and M. S. Abadir. Very Low Cost Testers : Opportunities and Challenges. *IEEE Design and Test of Computers*, 18(5):60–69, September 2001. [cited at p. 3]

[17] A. L. Crouch. *Design-For-Test for Digital ICs and Embedded Core Systems.* Prentice Hall, 1999. [cited at p. 4]

[18] R. Kapur, B. Keller, B. Koenemann, M. Lousberg, P. Reuter, T. Taylor, and P. Varma. P1500-CTL: Towards a Standard Core Test Language. In *Proc. VLSI Test Symposium*, pages 489–490, Dana Point, CA, April 1999. [cited at p. 5, 7]

[19] R. Kapur, M. Lousberg, T. Taylor, B. Keller, P. Reuter, and D. Kay. CTL-The Language for Describing Core-Based Test. In *Proc. International Test Conference*, pages 131–139, Baltimore, MD, October 2001. [cited at p. 5]

[20] IEEE Computer Society. *IEEE Standard Interface Language (STIL) for Digital Test Vector Data - Language Manual– IEEE Std.1450.0.* IEEE, New York, September 1999. [cited at p. 5]

[21] A. Hale and E. J. Marinissen. IEEE 1500 Web Site. http://grouper.ieee.org/groups/1500/. [cited at p. 5]

[22] IEEE Std. 1500. IEEE Standard for Embedded Core Test- IEEE Std. 1500-2004. Technical report, IEEE, New York, 2004. [cited at p. 5]

[23] E. J. Marinissen, R. Kapur, M. Lousberg, T. Mclaurin, M. Ricceheti, and Y. Zorian. On IEEE P1500's Standard for Embedded Core Test. *Journal of Electronic Testing: Theory and Applications(JETTA)*, 18(4/5):365–383, August 2002. [cited at p. 5, 7]

[24] K. P. Parker. *The Boundary Scan Handbook.* Springer Verlag, 3rd edition, 2003. [cited at p. 5]

[25] Q. Xu, Y. Zhang, and K. Chakrabarty. SOC Test Architecture Optimization for Signal Integrity Faults on Core-External Interconnects. In *Proc. IEEE Design Automation Conference*, pages 676–681, San Diego, California, USA, June 2007. [cited at p. 6]

[26] Y. Zorian. Test Requirements for Embedded Core-Based Systems and IEEE 1500. In *Proc. International Test Conference*, pages 191–199. IEEE Computer Society Press, November 1997. [cited at p. 7]

[27] P. Nag, A. Gattiker, S. Wei, R. Blanton, and W. Maly. Modeling of Economics of Testing: A DFT Perspective. *IEEE Design and Test of Computers*, 19(1):29–41, January 2002. [cited at p. 7]

[28] C. Shi and R. Kapur. How Power Aware Test Improves Reliability and Yield. Technical report, EEDesign.com, September 15 2004. [cited at p. 9, 129]

[29] S. Bhatia, T. Gheewala, and P. Varma. A Unifying Methodology of Intellectual Property and Custom Logic Testing. In *Proc. International Test Conference*, pages 639–648, October 1996. [cited at p. 15]

[30] L. Whetsel. An IEEE 1149.1 Based Test Access Architecture for ICs with Embedded Cores. In *Proc. International Test Conference*, pages 69–78, November, 1997. [cited at p. 15]

[31] N. Touba and E. McClauskey. Altering a Pseudoradom Bit Sequence for Scan-Based BIST. In *Proc. International Test Conference*, pages 167–175, October 1996. [cited at p. 15]

[32] B. Pouya and N. Touba. Modifying User-Defined Logic for Test Access to Embedded Cores. In *Proc. International Test Conference*, pages 60–68, November 1997. [cited at p. 15]

[33] P. Harrod. Testing Reusable IP - A Case Study. In *Proc. International Test Conference*, pages 493–498, 1999. [cited at p. 15]

[34] F.P.M Beenker, K.J.E van Eerdewijk, R.B.W Gerritsen, F.N Peacock, and M. van der Star. Macro Testing: Unifying IC and Board Test. *IEEE Design and Test of Computers*, 3(6):26–32, December 1986. [cited at p. 15]

[35] I. Ghosh, S. Dey, and N. K. Jha. A Fast and Low-Cost Testing Technique for Core Based System-Chips. *IEEE Trans. on CAD*, 19(8), August 2000. [cited at p. 15]

[36] I. Ghosh, S. Dey, and N. K. Jha. Low Overhead Design for Testability and Test Genration Technique for Core-Based System-on-a-Chip. *IEEE Trans. on CAD*, 18(11), November 1999. [cited at p. 15]

[37] S. Ravi, G. Lakshminarayana, and N. K. Jha. Testing of Core-Based Systems-on-a-Chip. *IEEE Trans. on CAD*, 20(3):426–439, March 2001. [cited at p. 15]

[38] T. Yoneda and H. Fujiwara. Design for Consecutive Testability of System-on-a-Chip with Built-In Self Testable Cores. *Journal of Electronic Testing: Theory and Applications(JETTA)*, 18(4/5):487–501, August 2002. [cited at p. 16]

[39] K. Chakrabarty, R. Mukherjee, and A. S. Exnicious. Synthesis of Transparent Circuits for Hierarchical and System-on-Chip Test. In *Proc. IEEE Intl. Conf. on VLSI Design*, pages 431–436, Bangalore, India, January 2001. [cited at p. 16]

[40] T. Yoneda and H. Fujiwara. Design for Consecutive Transparency of Cores in System-on-a-Chip. In *Proc. VLSI Test Symposium*, pages 287–292, Napa, CA, April 2003. [cited at p. 16]

[41] M. Nourani and C. Papachristou. Structural Fault Testing of Embedded Cores using Pipelining. *Journal of Electronic Testing: Theory and Applications(JETTA)*, 15(1-2):129–144, August-October 1999. [cited at p. 16]

[42] J. Aerts J. and E. J. Marinissen. Scan Chain Dsign for Test Time Reduction in Core-based ICs. *Proc. International Test Conference*, pages 448–457, October 1998. [cited at p. 16]

[43] P. Varma and S. Bhatia. A Structured Test Re-use Methodology for Core Based System Chips. *Proc. International Test Conference*, pages 294–302, October 1998. [cited at p. 16]

[44] E. J. Marinissen, R. Arendsen, G. Bos, M. Lousberg Dingemanse, and C. Wouters. A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores. *Proc. International Test Conference*, pages 284–293, October 1998. [cited at p. 16]

[45] E. J.Marinissen, S. K. Goel, and M. Lousberg. Wrapper Design for Embedded Core Test. *Proc. International Test Conference*, pages 911–920, 2000. [cited at p. 17, 29, 32]

[46] V. Iyengar, K. Chakraborty, and E. J. Marinissen. Co-optimization of Test Wrapper and Test Access Architecture for Embedded Cores. *Journal of Electronic Testing: Theory and Applications(JETTA)*, 18(2):213–230, 2002. [cited at p. 17, 19]

[47] D. Zou, S. M. Reddy, I. Pomeranz, and Y. Huang. SOC Test Scheduling Using Simulated Annealing. *Proc. VLSI Test Symposium*, pages 325–330, 2003. [cited at p. 17, 20, 32, 42, 44, 163]

[48] S. Koranne. A Novel Reconfigurable Wrapper for Testing of Embedded Core-based SOCs and its Associated Scheduling Algorithm. *Journal of Electronic Testing: Theory and Applications(JETTA)*, 18(4/5):415–434, August 2002. [cited at p. 17, 20]

[49] S. Koranne. Design of Reconfigurable Access Wrappers for Embedded Core Based SOC Test. *IEEE Trans. on Very large Scale Integration(VLSI) Systems*, 11(5):955–960, August 2003. [cited at p. 17]

[50] E. Larsson and Z. Peng. A Reconfigurable Power-Conscious Core Wrapper and its Application to SOC Test Scheduling. In *Proc. International Test Conference*, pages 1135–1144, Charlotte, September 2003. [cited at p. 18]

[51] Q. Xu and N. Nicolici. Time/Area Trade-off in Testing Hierarchical SOCs with Hard Mega-Cores. In *Proc. International Test Conference*, pages 1196–1202, 2004. [cited at p. 18, 63]

[52] S. K. Goel. An Improved Wrapper Architecture for Parallel Testing of Hierarchical Cores. In *Proc. IEEE Intl. Conf. on European Test Symposium*, pages 147–152, Corsica, France, May 2004. [cited at p. 18]

[53] M.-L. Flottes, J. Pouget, and B. Rouzeyre. Sessionless Test Scheme: Power-Constrained Test Scheduling for System-on-a-Chip. In *Proc. IEEE Intl. Conf. on IFIP VLSI-SOC*, pages 105–110, Montpellier, France, December 2001. [cited at p. 18]

[54] E. Larsson and H. Fujiwara. Power Constrained Premptive TAM Scheduling. In *Proc. IEEE Intl. Conf. on European Test Workshop*, pages 119–126, Corfu, Greece, May 2002. [cited at p. 18]

[55] V. Iyengar and K. Chakrabarty. Precedence-Based, Preemptive, and Power Constrained Test Scheduling for System-On-Chip. *Proc. VLSI Test Symposium*, pages 368–374, 2001. [cited at p. 18, 19]

[56] K. Chakrabarty. Test Scheduling for Core-based Systems using Mixed-Integer Linear Programming. *IEEE Trans. on CAD*, 19(10):1163–1174, 2000. [cited at p. 18, 29, 69]

[57] Y. Huang, S. M. Reddy, W. T. Cheng, and P. Reuter. Optimal Core Wrapper Width Selection and SOC Test Scheduling Based on 3D-Bin Packing Algorithm. *Proc. International Test Conference*, pages 74–82, 2002. [cited at p. 18, 55]

[58] V. Iyengar, K. Chakraborty, and E. J. Marinissen. On Using Rectangle Packing for SOC Wrapper/TAM Co-optimization. *Proc. VLSI Test Symposium*, pages 253–258, 2002. [cited at p. 18, 19]

[59] V. Iyengar, K. Chakrabarty, and E. J. Marinissen. Integrated Wrapper/TAM Co-optimization, Constant-driven Test Scheduling, and Tester Data Volume Reduction for SOCs. *Proc. IEEE Design Automation Conference*, pages 253–258, June 2002. [cited at p. 19]

[60] C. P. Ravikumar, A. Verma, and G. Chandra. A Polynomial Time Algorithm for Power Constrained Testing of Core-based Systems. *Proc. Asian Test Symposium*, pages 107–112, 1999. [cited at p. 19]

[61] C. P. Ravikumar, G. Chandra, and A. Verma. Simultaneous Module Selection and Scheduling for Power Constrained Testing of Core Based System. *Proc. IEEE Intl. Conf. on VLSI*, pages 462–467, 2000. [cited at p. 19]

[62] Y. Xia, M. Chrzanowska-Jeske, B. Wang, and M. Jeske. Using a Distributed Bin-Packing Approach for Core-based SOC Test Scheduling with Power Constraints. *IEEE/ACM International Conference on CAD*, pages 100–105, 2003. [cited at p. 19, 41, 42, 43, 163]

[63] C.-P. Su and C.-W Wu. A Graph-Based Approach to Power-Constrained SOC Test Scheduling. *Journal of Electronic Testing: Theory and Applications(JETTA)*, 20(1):45–60, February 2004. [cited at p. 19]

[64] J. Pouget, E. Larsson, and Z. Peng. Multiple-Constraint Driven System-On-Chip Test Time Optimization. *Journal of Electronic Testing: Theory and Applications(JETTA)*, 21(6):599–611, December 2005. [cited at p. 19, 34, 43, 55]

[65] D. Zhao and S. Upadhyay. A Generic Resource Distribution and Test Scheduling Scheme for Embedded Core-Based SOCs. *IEEE Trans. on Instrumentation and Measurement*, 53(2):318–329, 2004. [cited at p. 19]

[66] R. M. Chou, K. K. Saluja, and V. D. Agrawal. Scheduling Test for VLSI Systems under Power Constraints. *IEEE Trans. on Very large Scale Integration(VLSI) Systems*, 5(2):175–185, 1997. [cited at p. 19, 33]

[67] V. Iyengar, K. Chakraborty, and E. J. Marinissen. Efficient Wrapper/TAM Co-optimization for Large SOCs. *Proc. Design, Automation and Test in Europe*, pages 491–498, March 2002. [cited at p. 19]

[68] S. Koranne. On Test Scheduling for Core-based SOCs. *Proc. IEEE Intl. Conf. on VLSI*, pages 505–510, January 2002. [cited at p. 19]

[69] S. Koranne. Formulating SOC Test Scheduling As a Network Transportation Problem. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 21(12):1517–1525, December 2002. [cited at p. 19]

[70] S. K. Goel and E. J. Marinissen. Effective and Efficient Test Architecture Design for SOCs. *Proc. International Test Conference*, pages 529–538, 2002. [cited at p. 19, 42, 163]

[71] S. K. Goel and E. J. Marinissen. Layout Driven SOC Test Architecture Design for Test Time and Wire Length Minimization. *Proc. Design, Automation and Test in Europe*, pages 738–743, March 2003. [cited at p. 20]

[72] V. Iyengar, K. Chakrabarty, and E. J. Marinissen. Test Access Mechanism Optimization, Test Scheduling, and Tester Data Volume Reduction for System-on-Chip. *IEEE Trans. on Computers*, 52(12):1619–1632, December 2003. [cited at p. 20, 32, 33, 49, 53, 70]

[73] Q. Xu and N. Nicolici. Multi-Frequency TAM Design for Hierarchical SOCs. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 25(1):181–196, December 2006. [cited at p. 20]

[74] Y. Huang. Resource Allocation and Test Scheduling for Concurrent Test of Core-based SOC Design. *Proc. Asian Test Symposium*, pages 265–270, 2001. [cited at p. 20]

[75] S. Koranne and V. Iyengar. On the Use of k-Tuples for SOC Test Schedule Representation. *Proc. International Test Conference*, pages 539–548, 2002. [cited at p. 20]

[76] J.-Yi Wuu, T.-C. Chen, and Y.-W. Chang. SOC Test Scheduling Using B*-Tree Based Floor Planning Technique. *Proc. IEEE Asia South Pacific Design Automation Conference*, pages 1188–1191, 2005. [cited at p. 20, 41, 42, 163]

[77] J.-Ho Ahn and S. Kang. SoC Test Scheduling Algorithm Using ACO-Based Rectangle Packing. *Proc. Intl. Symp. on Intlligent Computing*, pages 655–660, August 2006. [cited at p. 20, 41, 42, 163]

[78] Y. Yu, X. Y. Peng, and Y. Peng. A Test Scheduling Algorithm Based on Two-Stage GA. *Proc. International Symposium on Instrumentation Science and Technology*, pages 658–662, 2006. [cited at p. 20, 41, 42, 163]

[79] E. Larsson and H. Fujiwara. System-On-Chip Test Scheduling with Reconfigurable Core Wrappers. *IEEE Trans. on Very large Scale Integration(VLSI) Systems*, 14(3):305–309, March 2006. [cited at p. 20, 42]

[80] A. Sehgal, V. Iyengar, M. D. Krasniewski, and K. Chakrabarty. Test Cost Reduction for SOCs using Virtual TAMs and Langrange Multiplier. In *Proc. IEEE Design Automation Conference*, pages 738–743, 2003. [cited at p. 20]

[81] A. Sehgal and K. Chakrabarty. Test Planning for the Effective Utilization of Port-Sclable Testers for Heterogeneous Core-Based SOCs. In *IEEE/ACM International Conference on CAD*, pages 88–93, 2005. [cited at p. 20, 50]

[82] G. Hetheringten, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, and J. Rajski. Logic BIST for Large Industrial Designs. In *Proc. International Test Conference*, pages 358–367, 1999. [cited at p. 21]

[83] N. A. Touba. Survey of Test Vector Compression Techniques. *IEEE Design and Test of Computers*, pages 294–303, July-August 2006. [cited at p. 21, 79]

[84] A. Jas, J. Ghosh-Dastidar, N. Mom-Eng, and N. A. Touba. An Efficient Test Vector Compression Scheme Using Selective Huffman Coding. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 22(6):797–806, June 2003. [cited at p. 22, 80, 85, 86, 164]

[85] A. Jas, J. Ghosh-Dastidar, and N. A. Touba. Scan Vector Compression/Decompression Using Statistical Coding. *Proc. IEEE VLSI Test Symposium*, pages 114–121, April 1999. [cited at p. 22, 124, 125]

[86] E. Kalligaros and D. Nikolos. Optimal Selective Huffman Coding for Test Data Compression. *IEEE Trans. on Computers*, 56(8):1146–1152, August 2007. [cited at p. 22]

[87] M. H. Tehranipour, M. Nourani, and K. Chakrabarty. Nine-Coded compression Technique for Testing Embedded Cores in SOCs. *IEEE Trans. on Very large Scale Integration(VLSI) Systems*, 13(6):719–731, June 2005. [cited at p. 22, 24, 80, 85, 86, 125, 164]

[88] X. Kavousianis, E. Kalligeros, and D. Nikolos. Multilevel Huffman Coding: An Efficient Test Data Compression Method for IP Cores. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 26(6):1070–1083, June 2007. [cited at p. 22]

[89] A. Jas and N. A. Touba. Test Vector Decompression via Cyclic Scan Chains and its Aplication to Testing Core-based Design. *Proc. International Test Conference*, pages 458–464, 1998. [cited at p. 22]

[90] A. Chandra and K. Chakrabarty. System-on-a-chip Test Data Compression and Decompression Architectures based on Golomb Codes. *IEEE Trans. on CAD*, 20(6):355–368, March 2001. [cited at p. 22, 80, 85, 86, 124, 125, 164]

[91] A. Chandra and K. Chakrabarty. 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. [cited at p. 22, 24, 85, 86, 124, 125]

[92] A. El-Maleh and R. Al-Abaji. Extended Frequency-directed Run-length Codes with Improved Application to System-on-a-chip Test Data Compression. *Proc. Intl. Conference on Elect. Cirs. and Systems*, pages 449–452, 2002. [cited at p. 22, 23]

[93] M. H. Tehranipoor, M. Nourani, K. Arabi, and A. Afzali-Kusha. Mixed RL-Huffman Encoding for Power Reduction and Data Compression in Scan Test. *Proc. IEEE Intl. Symposyam on Circuits and Systems*, 2:II–681–II–684, 2004. [cited at p. 22, 85, 86, 164]

[94] M. Nourani and M. H. Tehranipour. RL-Huffman Encoding for Test Compression and Power Reduction in Scan Applications. *ACM Trans. on Design Automation of Electronic Systems*, 10(1):91–115, January 2005. [cited at p. 22, 96, 125, 126]

[95] S. Reda and A. Oriaglu. Reducing Test Application Time Through Test Data Mutation Encoding. *Proc. Design, Automation and Test in Europe*, pages 387–393, March 2002. [cited at p. 22]

[96] A. Chandra and K. Chakrabarty. A Unified Approach to Reduce SOC Test Data Volume, Scan Power and Testing Time. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 22(3):352–363, March 2003. [cited at p. 22]

[97] P. Rosinger, P. Gonciari, B. Al-Hashimi, and N. Nicolici. Simoultaneous Reduction in Volume of Test Data and Power Dissipation for Systmm-on-a-chip. *Electronic Letter*, 37(24):1434–1436, 2005. [cited at p. 22, 24]

[98] X. Ruan and R. S. Katti. Data-Independent Pattern Run-Length Compression for Testing Embedded Cores in SOCs. *IEEE Trans. on Computers*, 56(4):545–556, April 2007. [cited at p. 22, 80, 85, 86, 164]

[99] H. Ichihara, Y. Setohara, Y. Nakashima, and T. Inoue. Test Compression/Decompression Based on JPEG VLC Algorithm. In *Proc. Asian Test Symposium*, pages 87–90, Beijing, October 2007. IEEE CS Press. [cited at p. 23]

[100] H. Fang, C. Tong, and X. Cheng. RunBasedReordering: A Novel Approach for Test Data Compression and Scan Power. In *Proc. IEEE Asia South Pacific Design Automation Conference*, pages 732–737, 2007. [cited at p. 23]

[101] F. G. Wolf and C. Papachristou. Multiscan-based Test Compression and Hardware Decompression using LZ77. *Proc. International Test Conference*, pages 331–339, 2002. [cited at p. 23]

[102] M. J. Knieser, F. G. Wolf, C. A. Papachriston, D. J. Weyer, and D. R. McIntyre. A Technique for High Ratio LZW Compression. *Proc. Design, Automation and Test in Europe*, pages 116–121, 2003. [cited at p. 23]

[103] H. Hashempur and P. Lombardi. Application of Arithmatic Coding for Compression of VLSI Test Data. *IEEE Trans. on Computers*, 54(9):1166–1177, September 2005. [cited at p. 23, 80, 85, 86, 164]

[104] K. J. Balakrishnan and N. A. Touba. Matrix based Software Test Data Compression for System-on-a-chip. *Journal of System Architecture*, 50:247–256, 2004. [cited at p. 23]

[105] K. J. Balakrishnan and N. A. Touba. Relation between Entropy and Test Data Compression. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 26(2):386–395, February 2007. [cited at p. 23]

[106] K. J. Balakrishnan and N. A. Touba. Deterministic Test Vector Decompression in Software using Linear Operations. *Proc. IEEE VLSI Test Symposium*, pages 719–731, 2005. [cited at p. 23]

[107] S. I. Ward, C. Schattauer, and N. A. Touba. Using Statistical Transformations to Improve Compression for Linear Decompressors. In *Proc. IEEE Intl. Symp. on Defect and Fault-Tolerance in VLSI Systems*, pages 42–50, October 2005. [cited at p. 23]

[108] A. Jas, B. Pouya, and N. A. Touba. Test Data Compression Technique for Embedded Cores Using Virtual Scan Chains. *IEEE Trans. on Very large Scale Integration(VLSI) Systems*, 12(7):775–780, July 2004. [cited at p. 23]

[109] I. Bayraktaroglu and A. Orailoglu. Test Volume and Application Time Reduction through Scan Chain Concealment. *Proc. IEEE Design Automation Conference*, pages 151–155, 2001. [cited at p. 23]

[110] I. Bayraktaroglu and A. Orailoglu. Decompression Hardware Determination for Test Data Volume and Time Reduction through Unified Test Pattern Compaction and Compression. *Proc. IEEE VLSI Test Symposium*, pages 113–120, 2003. [cited at p. 23]

[111] L. Li and K. Chakrabarty. Deterministic BIST based on a Reconfigurable Interconnection Network. *Proc. International Test Conference*, pages 460–469, 2003. [cited at p. 23]

[112] CA Synopsys Inc. Mountain View. SOC-BIST Deterministic Logic BIST user Guide, 2004. [cited at p. 23]

[113] C. Krishna, A. Jas, and N. A. Touba. Test Vector Encoding Using parital LFSR Reseeding. *Proc. International Test Conference*, pages 885–893, 2001. [cited at p. 23]

[114] E. Volkernik and S. Mitra. Efficient Seed Utilization for Reseeding based Compression. *Proc. IEEE VLSI Test Symposium*, pages 232–237, 2003. [cited at p. 23]

[115] B. Koenmann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater. A SmartBIST Variant with Guaranteed Encoding. In *Proc. Asian Test Symposium*, pages 325–330. IEEE CS Press, 2001. [cited at p. 23]

[116] S. Wang, W. Wei, and S. T. Chakradhar. A High Compression and Short Test Sequence Test Compression Technique to Enhance Compression of LFSR Reseeding. In *Proc. Asian Test Symposium*, pages 79–86, Beijing, October 2007. IEEE CS Press. [cited at p. 23, 25]

[117] J. Lee and N. A. Touba. LFSR-Reseeding Scheme Achieving Low-Power Dissipation During Test. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 26(2):396–401, February 2007. [cited at p. 23]

[118] J. Rajaski. Embedded Deterministic Test. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 23(5):776–792, May 2004. [cited at p. 23]

[119] K. S. Kim. XMAX:A Practical and Efficient Compression Architecture. In *Proc. International Test Conference*, pages 1–2, 2005. [cited at p. 23]

[120] S. Mitra and K. S. Kim. XPAND: An Efficient Test Stimulus Compression Technique. *IEEE Trans. on Computers*, 55(2):163–173, February 2006. [cited at p. 23, 24]

[121] C. V. Krishna and N. A. Touba. Adjustable Width Linear Combinational Scan Vector Decompression. In *IEEE/ACM International Conference on CAD*, pages 863–866. IEEE CS Press, 2003. [cited at p. 23]

[122] J. Lee, J. J. Chen, and C. H. Huang. Using a Single Input to Support Multiple Scan Chains. In *IEEE/ACM International Conference on CAD*, pages 74–78. IEEE CS Press, 1998. [cited at p. 24]

[123] I. Hamzaoglu and J. H. Patel. Reducing Test Application Time for Full Scan Embedded Cores. In *Intl. Symp. on Fault Tolerant Computing*, pages 260–267, 1999. [cited at p. 24]

[124] F. Hsu, K. Butler, and J. Patel. A Case Study on the Implementation of the Illinois Scan Architecture. *Proc. International Test Conference*, pages 538–547, 2001. [cited at p. 24]

[125] L.-T Wang, K. S. Abdel-Hafez, S. Wu, X. Wen, H. Furukawa, F.-S. Hsu, S.-H. Lin, and S.-W. Tsai. Virtualscan : A New Compressed Scan Technology for Test Cost Reduction. In *Proc. International Test Conference*, pages 916–925, 2004. [cited at p. 24]

[126] H. Tang, S. M. Reddy, and I Pomeranz. On Reducing Test Data Volume and Test Application Time for Multiple Scan Designs. In *Proc. International Test Conference*, pages 1079–1088, 2003. [cited at p. 24]

[127] A. El-Maleh, M. I. Ali, and A. A. Al-Yamani. A Reconfigurable Broadcast Scan Compression Scheme Using Relaxation Based Test Vector Decompression. In *Proc. Asian Test Symposium*, pages 91–94, Beijing, October 2007. IEEE CS Press. [cited at p. 24]

[128] A. El-Maleh. An Efficient Test Vector Compression Technique based on Block Merging. In *Proc. IEEE Intl. Symposyam on Circuits and Systems*, pages 1447–1450, 2006. [cited at p. 24]

[129] Lin S.Ping, Lee C.Len, Chen J.E, Chen J.Jan, Luo K.Lun, and Wu W.Ching. A multilayer Data Copy Test Data Compression Scheme for Reducing Shifting-in Power for Multiple Scan Design. *IEEE Trans. on Very large Scale Integration(VLSI) Systems*, 15(7):767–776, July 2007. [cited at p. 24, 80, 85, 86, 164]

[130] P. Flores, J. Costa, J. Neto, J. Monteiro, and J. Silva. Assignment and Reordering of Incompletely Specified Pattern Sequences Targeting Minimum Power Dissipation. *Proc. IEEE Intl. Conf. on VLSI*, pages 37–41, 1999. [cited at p. 25, 26]

[131] P. Girard, C. Laundrault, S. Provossoudovitch, and D. Severac. Reducing Power Consumption during Test Application by Test Vector Ordering. *Proc. IEEE Intl. Symposyam on Circuits and Systems*, pages 296–299, 1998. [cited at p. 25, 26]

[132] V. Dabholkar, S. Chakrabarty, I. Pomeranz, and S. M. Reddy. Techniques for Minimizing Power Dissipation in Scan and Combinational Circuits During Test Application. *IEEE Trans. on CAD*, 17(12):1325–1333, December 1998. [cited at p. 25]

[133] S. Kajihara, K. Ishida, and K. Miyasi K. Test Power Reduction for Full Scan Sequential Circuits by Test Vector Modification. *Second Workshop on RTL ATPG and DFT*, pages 140–145, November 2001. [cited at p. 25]

[134] S. Ghosh, S. Basu, and N. A. Touba. Joint Minimization of Power and Area in Scan Testing by Scan Cell Reordering. *Proc. IEEE Computer Society Annual Symposium on VLSI*, pages 246–249, 2003. [cited at p. 25]

[135] Y. Bonhomme, P. Girard, L. Guiller, C. Landrault, and S. Pravossoudovtch. A Gated Clock Scheme for Low Power Scan Testing of Logic ICs or Embedded Cores. *Proc. IEEE VLSI Test Symposium*, pages 253–258, 2001. [cited at p. 25, 26, 98]

[136] N. Nicolici and B. M. Al-Hashimi. Scan Latch Partitioning into Multiple Scan Chains for Power Minimization in Full Scan Sequential Circuits. In *Proc. Design, Automation and Test in Europe*, pages 715–722, Paris, France, 2000. [cited at p. 25, 27]

[137] S. Wang and S. K. Gupta. ATPG for Heat Dissipation Minimization During Test Application. *IEEE Trans. on Computers*, 47(2):256–262, February 1998. [cited at p. 25]

[138] S. Wang and S. K. Gupta. An Automatic Test Pattern Generator for Minimizing Switching Activity During Scan Testing Activity. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 21(8):954–968, August 2002. [cited at p. 25]

[139] H. K. Lee and D. S Ha. On the Generation of Test Patterns for Combinational Circuits. Technical Report 12-93, Dept. of Elec. Eng. Virginia Polytechnic Institute and State University., 1993. [cited at p. 25, 135, 138, 159]

[140] F. Corno et al. A Test Pattern Generation Methodology for Low Power Consumption. *Proc. VLSI Test Symposium*, pages 453–459, 1998. [cited at p. 25]

[141] M.-F Wu, K.-S Hu, and J.-L Huang. An Efficient Peak Power Reduction Technique for Scan Testing. In *Proc. Asian Test Symposium*, pages 111–114, Beijing, October 2007. [cited at p. 25]

[142] R. Sankaralingam, B. Pouya, and N. A. Touba. Reducing Power Dissipation During Testing Using Scan Chain Disable. In *Proc. VLSI Test Symposium*, pages 319–324, Los Alamitos, 2001. [cited at p. 25]

[143] H. J. Wunderlich and S. Gerstendorfer. Minimized Power Consumption for Scan based BIST. *Proc. International Test Conference*, pages 85–94, 1999. [cited at p. 26]

[144] R. Sankaralingam and N. A. Touba. Inserting Test Points to Control Peak Power During Scan Testing. *Proc. IEEE Intl. Symp. on Defect and Fault-Tolerance in VLSI Systems*, pages 138–146, 2002. [cited at p. 26]

[145] A. Hertwig and H. J. Wunderlich. Low Power Serial Built-In Self-Test. In *Proc. IEEE Intl. Conf. on European Test Workshop*, pages 49–53, 1998. [cited at p. 26]

[146] S. Sharifi, J. Jaffari, M. Hosseinabady, A. Afzali-Kusha, and Z. Navabi. Simultaneous Reduction of Dynamic and Static Power in Scan Structures. In *Proc. Design, Automation and Test in Europe*, pages 846–851, 2005. [cited at p. 26]

[147] S. Sharifi, J. Jaffari, M. Hosseinabady, A. Afzali-Kusha, and Z. Navabi. Scan Based Structure with Reduced Static and Dynamic Power Consumption. *Journal of Low Power Electronics*, 2(3):477–487, 2006. [cited at p. 26]

[148] T. Huang and K. Lee. An Input Control Technique for Power Reduction in Scan Circuits During Test Application. *Proc. Asian Test Symposium*, pages 315–320, 1999. [cited at p. 26]

[149] M. A. Cirit. Estimating Dynamic Power Consumption of CMOS Circuits. *IEEE/ACM International Conference on CAD*, pages 534–537, 1987. [cited at p. 26, 79, 91, 93, 100, 130]

[150] V. Dabholkar V., S. Chakravarty, I. Pomeranz, and S. M. Reddy. Techniques for Minimizing Power Dissipation in Scan and Combinational Circuits during Test Application. *IEEE Trans. on CAD*, 17(12):1325–1333, December 1998. [cited at p. 26]

[151] R. Gupta and M. Breuer. Ordering Storage Elements in Single Chain. *IEEE/ACM International Conference on CAD*, pages 408–411, 1991. [cited at p. 26]

[152] S. Chakravarty S. and V. Dhabolkar. Two Techniques for Minimizing Power Dissipation in Scan Circuits during Test Application. *Proc. Asian Test Symposium*, pages 324–329, 1994. [cited at p. 26]

[153] D. Xiang, J. Sun, M. Chen, and S. Gu. Cost-effective Scan Architecture and a Test Application Scheme for Scan Testing with Non-scan Test Power and Test Aplication Cost. U. S. Patent 6959426B2, October 25 2005. [cited at p. 26]

[154] D. Xiang, K. Li, and H. Fujiwara. Design for Cost-effective Scan Testing by Reconstructing Scan Flip-flops. In *Proc. Asian Test Symposium*, pages 318–321, 2005. [cited at p. 26]

[155] S. Gupta, V. Tarang, and S. Chattopadhyay. Flip-flop Chaining Architecture for Power Efficient Scan during Test Application. In *Proc. Asian Test Symposium*, pages 410–413, December 2005. [cited at p. 26, 134, 161]

[156] O. Sinanoglu, I. Bayraktaroglu, and A. Orailoglu. Test Power Reduction Through Minimization of Scan Chain Transitions. *Proc. VLSI Test Symposium*, pages 166–171, 2002. [cited at p. 26]

[157] O. Sinanoglu, I. Bayraktaroglu, and A. Orailoglu. Reducing Average and Peak Test Power Through Scan Chain Modification. *Journal of Electronic Testing: Theory and Applications(JETTA)*, 2003. [cited at p. 26, 27]

[158] L. Whetsel. Adapting Scan Architectures for Low Power Operation. In *Proc. International Test Conference*, pages 863–872, 2000. [cited at p. 26]

[159] O. Sinanoglu and A. Orailoglu. A Novel Scan Architecture for Power Efficient, Rapid Test. In *IEEE/ACM International Conference on CAD*, pages 299–303, 2002. [cited at p. 27]

[160] J. Li, Y. Hu, and X. Li. A Scan Chain Adjustment Technology for Test Power Reduction. In *Proc. Asian Test Symposium*, pages 11–16, 2006. [cited at p. 27]

[161] Y. Kim, D. Song, K. Kim, I. Kim, and S. Kang. TOSCA: Total Scan Power Reduction Architecture based on Pseudo-Random Built-in Self Test Structure. In *Proc. Asian Test Symposium*, pages 17–24, 2006. [cited at p. 27]

[162] Y. Kim, M.-H. Y, Y. Lee, and S. Kang. A New Low Power Test Pattern Generator using a Transition Monitoring Window. In *Proc. Asian Test Symposium*, pages 230–235, 2005. [cited at p. 27]

[163] M. Hirech, J. Beausang, and X. Gu. A New Approach to Scan Chain Reordering using Physical Design Information. In *Proc. International Test Conference*, pages 348–355, 1998. [cited at p. 27]

[164] L.-C Hsu and H.-M Chen. On Optimizing Scan Testing Power and Routing Cost in Scan Chain Design. In *Proc. Intl. Symp. on quality Electronic Design*, pages 451–456, 2006. [cited at p. 27]

[165] B. B. Paul, R. Mukhopadhyay, and I. S. Gupta. Genetic Algorithm based Scan Chain Optimization and Test Power Reduction using Physical Information. In *TENCON*, pages 1–4, 2006. [cited at p. 27]

[166] G. Seok, I.-S. Lee, T. Ambler, and B. F. Womack. An Efficient Scan Chain Partitioning Scheme with Reduction of Test Data under Routing Constraint. In *Proc. IEEE Intl. Symp. on Defect and Fault-Tolerance in VLSI Systems*, pages 145–156, 2006. [cited at p. 27]

[167] A. Al-Yamani, N. Devta-Prasanna, E. Chemlar, M. Grinchuk, and A. Gupta. Scan Test Cost and Power Reduction Through Systematic Scan Configuration. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 26(5):907–918, May 2007. [cited at p. 27]

[168] D. Xiang, K. Li, H. Fujiwara, K. Thulasiraman, and J. Sun. Constraining Transition Propagation for Low-Power Scan Testing Using a Two-Stage Scan Architecture. *IEEE Trans. on Circuits and Systems-II:Express briefs*, 54(5):450–454, May 2007. [cited at p. 27]

[169] B.-H Chen, W.-C Kao, B.-C Bai, S.-T Shen, and J. C.-Li. Response Inversion Scan Cell (risc) : A Peak Capture Power Reduction Technique. In *Proc. Asian Test Symposium*, pages 425–430, October 2007. [cited at p. 27]

[170] D. Ghosh D., S. Bhunia, and K. Roy. Multiple Scan Chain Design Technique for Power Reduction during Test Application in BIST. In *Proc IEEE Intl. Symp. on Defect and Fault Tolerance., VLSI Systems*, pages 191–198, 2003. [cited at p. 28]

[171] I.-S Lee, Y. M. Hur, and T. Ambler. The Efficient Multiple Scan Chain Architecture Reducing Power Dissipation and Test Time. In *Proc. Asian Test Symposium*, pages 94–97, November 2004. [cited at p. 28]

[172] E. J. Marinissen, V. Iyengar, and K. Chakrabarty. A Set of Benchmarks for Modular Testing of SOCs. *Proc. International Test Conference*, pages 519–528, 2002. [cited at p. 30, 36, 41, 57, 74]

[173] S. Samii, K. Chakrabarty E. Larsson, and Z. Peng Z. Cycle-Accurate Test Power Modelling and 0its Application to SOC Test Scheduling. *Proc. International Test Conference*, October 2006. [cited at p. 33]

[174] Melanie Mitchell. *An Introduction to Genetic Algorithm.* MIT Press, 1998. [cited at p. 41]

[175] R. Dorsch, R. H. Rivera, H-J Wunderlich, and M. Fischer. Adapting an SoC to ATE Concurrent Test Capabilities. In *Proc. International Test Conference*, pages 1169–1175, 2002. [cited at p. 47]

[176] Agilent Technologies, Available online 16243240485664 at:http://cp.literature.agilent.com/litweb/pdf/5988-7344EN.pdf. *Winning in the SOC Market.* [cited at p. 47]

[177] Teradyne Technologies, http://www.teradyne.com/tiger/digital.html. *Tiger: Advanced Digital with Silicon Germanium Technology.* [cited at p. 47]

[178] K. Chakrabarty. Optimal Test Access Architecture for System-on-a-Chip. *ACM Trans. Design Automation of Electronic Systems*, 6(1):26–49, January 2001. [cited at p. 50]

[179] Y. Bonhomme, P. Girard, C. Landrault, and S. Pravossouvitch. Test Power: a Big Issue in Large SOC Designs. In *IEEE Intl. Workshop on Electronic Design, Test and Applications*, pages 447–449, 2002. [cited at p. 55]

[180] N. Nicolici and B. M. Al-Hashimi. *Power-Constrained Testing of VLSI Circuits.* Kluwer Academic, 2003. [cited at p. 55]

[181] M. S. Hsiao, E. M. Rudnick, and J. H. Patel. Effects of Delay Models on Peak Power Estimation of VLSI Sequential Circuits. In *IEEE/ACM International Conference on CAD*, pages 45–51, 1997. [cited at p. 55]

[182] R. Burch, F. Najm, P. Yang, and T. Trick. A Monte Carlo Approach for Power Estimation. *IEEE Trans. on Very large Scale Integration(VLSI) Systems*, 1(1):63–71, March 1993. [cited at p. 55]

[183] K. Chakrabarty, V. Iyengar, and D. M. Krasniewski. Test Planning for Modular Testing of Hierarchical SOCs. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 24(3):435–448, March 2005. [cited at p. 63, 64]

[184] T.-P. Wang, C.-Yu Tsai, M.-Der Shieh, and K.-Jone Lee. Efficient Test Scheduling for Hierarchical Core Based Design. In *Proc. VLSI-TSA International Symposium on VLSI Design, Automation and Test*, pages 200–203, April 2005. [cited at p. 63]

[185] A. Sehgal, S. K. Goel, E. J. Marinissen, and K. Chakrabarty. Hierarchy-Aware and Area-Efficient Test Infrastructure Design for Core-Based System Chips. In *Proc. Design, Automation and Test in Europe*, pages 285–290, 2006. [cited at p. 63, 75, 76, 77, 78]

[186] F. Brglez, D. Bryan, and K. Kozminski. Combinational Profiles of Sequential Benchmark Circuits. *Proc. Intl. Symp. on Circuits and Systems*, pages 1929–1934, May 1989. [cited at p. 84, 124, 159]

[187] I. Hamzaouglu and J. H. Patel. Test Set Compaction Algorithms for Combinational Circuits. In *IEEE/ACM International Conference on CAD*, pages 283–289, 1998. [cited at p. 84]

[188] H.K. Lee and D.S. Ha. An Efficient Forward Fault Simulation Algorithm based On the Parallel Pattern Single Fault Propagation. In *Proc. International Test Conference*, pages 946–955, October 1991. [cited at p. 96]

[189] A. Chandra and K. Chakrabarty. Low Power Scan Testing and Test Data Compression for System-on-a-Chip. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 21(5):597–604, May 2002. [cited at p. 96]

[190] P. M. Rosinger, P. T. Gonciari, B. M. Al-Hashimi, and N. Nicolici. Analysis Trade-offs in Scan Power and Test Data Compression for System-on-a-chip. *IEE Computers and Digital techniques*, 149(4):188–196, July 2002. [cited at p. 96]

[191] D. Bertozzi, L. Benini, and G. D. Micheli. Low Power Error Resilient Encoding for On-chip Data Buses. *Proc. Design, Automation and Test in Europe*, 2002. [cited at p. 98]

[192] D. Liu. Power Consumption Estimation in CMOS VLSI Chips. *IEEE Journal of Solid State Circuits*, 29:663–670, June 1994. [cited at p. 98]

[193] S. Gerstendorfer and H. J. Wunderlich. Minimized Power Consumption for Scan-based BIST. *Journal of Electronic Testing: Theory and Applications(JETTA)*, 16(3):203–212, June 2000. [cited at p. 98]

[194] M. Zellerohr, A. Hertwig, and H. J. Wunderlich. Pattern Selection for Low-power Serial Built-In Self-Test. *Proc. IEEE International Test Synthesis Workshop*, 1998. [cited at p. 98]

[195] D. C. Suresh, B. Agrawal B., J. Yang, W. Najjar, and L. Bhuyan. Power Efficient Encoding Techniques for Off-chip Data Buses. *In Proc. CASES*, pages 267–275, 2003. [cited at p. 98]

[196] Z. Khan, T. Arslan, and A. T. Erdogan. Low Power System on Chip Bus Encoding Scheme with Crosstalk Noise Reduction Capability. *IEE Computers and Digital techniques*, 153(2):101–108, March 2006. [cited at p. 98]

[197] C.-G. Lyuh and T. Kim. Low-power Bus Encoding with Crosstalk Delay Estimation. *IEE Computers and Digital techniques*, 153(2):193–100, March 2006. [cited at p. 98]

[198] Donald E. Knuth. Dynamic Huffman Coding. *Journal of Alogorithms*, 6(2):163–180, June 1985. [cited at p. 113]

[199] Synopsys Inc. Design Compiler Reference Manual. [cited at p. 126]

[200] D. A. Pucknel and K. Eshraghian. *Basic VLSI Design.* Prentice Hall, 1994. [cited at p. 132]

[201] S. Chattopadhyay, D. R. Chowdhury, S. Bhattacharjee, and P. P. Chaudhuri. Cellular-Automata-Array-Based Diagnosis of Board Level Faults. *IEEE Trans. on Computers*, 47(8):817–828, August 1998. [cited at p. 160]

[202] M. G. Karpovsky and S. M. Chaudhry. Design of Self-Diagonistic Boards by Multiple Signature Analysis. *IEEE Trans. on Computers*, 42(9):1035–1044, September 1993. [cited at p. 160]

Publications

1. Chandan Giri, B. Naveen Kumar and Santanu Chattopadhyay, "*Scan Flip-Flop Ordering with Delay and Power Minimization During Testing*", Proc. Intl. IEEE Annual India Conference (INDICON), Chennai, pages:467-471, December, 2005.

2. Chandan Giri, B. Mallikarjuna Rao and Santanu Chattopadhyay, "*Test Data Compression using Huffman Coded Dictionary for System-On-Chip Testing*", Proc. Intl. Conference Electronic and Photonic Material, Devices and Systems (EPMDS), Kolkata, pages:I8-I10, January, 2006.

3. Chandan Giri, B. Mallikarjuna Rao and Santanu Chattopadhyay, "*Reducing Power for System-On-Chip Testing*", Proc. Intl. Conference on Emerging Applications of IT (EAIT),Computer Society of India (CSI), Kolkata, February, 2006.

4. Chandan Giri, Nikhil Reddy Cheruku and Santanu Chattopadhyay, "*Test Vector Ordering for Power reduction During Transmission of compressed Test Patterns to Embedded System-On-Chip*", Proc. IEEE Annual India Conference (INDICON), New Delhi, pages:1-5, September, 2006.

5. Chandan Giri and Santanu Chattopadhyay, "*Power Optimized Dictionary Coding for Test Data Compression*", Proc. IEEE Intl. Conference on Industrial Technology (ICIT), Bombay, pages:2541-2545, December,2006.

6. Chandan Giri, B. Naveen Kumar and Santanu Chattopadhyay, "*Multiple Scan Chain Design for Power and Delay Minimization During Test*", Proc. Intl. Conference on Computers and Devices for Communication (CODEC), Kolkata, December, 2006.

7. Chandan Giri, B. Mallikarjuna Rao and Santanu Chattopadhyay, "*Test Data Compression by Split-VIHC (SVIHC)*", Proc. Intl Conference on Computing: Theory and Applications (ICCTA), Kolkata, pp:146-150, March, 2007.

8. Chandan Giri, Dilip Kumar Reddy Tipparthi and Santanu Chattopadhyay, "*Genetic Algorithm Based Approach for Hierarchical SOC Test Scheduling*", Proc. Intl. Conference on Computing: Theory and Applications (ICCTA), Kolkata, pp:141-145, March,2007.

9. Chandan Giri and Santanu Chattopadhyay, "*Reducing Test-bus Power Consumption in Huffman Coding Based Test Data Compression for SOCs*", Proc. IEEE International Symposium on Circuits and Systems (ISCAS), New Orleans,USA, pages:3679-3682, May, 2007.

10. Chandan Giri, Soumojit Sarkar and Santanu Chattopadhyay, "*Test Scheduling for Core-based SOCs Using Genetic Algorithm based Heuristic Approach*", Proc. Intl. Conference on Intelligent Computing (ICIC), Qingdao, China, pages:1032-1041, August, 2007. Also as a book chapter in Lecture Notes in Computer science (LNCS), Springer, ISSN:0302-9743 (Print) 1611-3349 (Online),Vol. 4682/2007,ISBN:978-3-540-74201-2.

11. Chandan Giri, Nikhil Reddy Cheruku and Santanu Chattopadhyay, "*Compression-Power Trade-off in Dictionary based Test Data Compression*", Proc. IEEE VLSI Design and Test (VDAT) Symposium, Kolkata, August, 2007.

12. Chandan Giri, Pradeep Kumar Choudhary and Santanu Chattopadhyay, "*Scan Architecture Modification with Test Vector Re-ordering for Test Power Reduction*", Proc. IEEE International Symposium on Integrated Circuits (ISIC), Singapore, September, 2007.

13. Chandan Giri, Pradeep Kumar Choudhary and Santanu Chattopadhyay, "*Scan Power Reduction through Scan Architecture Modification and Test Vector Reordering*", Proc. Intl. IEEE Asian Test Symposium (ATS),Beijing, China, pages:419-424, October, 2007.

14. Chandan Giri, Soumojit Sarkar and Santanu Chattopadhyay, "*A Genetic Algorithm Based Heuristic Technique for Power Constrained Test Scheduling in Core-based SOCs*", Proc. IFIP Intl. Conference on Very Large Scale Integration (VLSI) SOC,Atalanta,USA,October,2007.

Communicated to Journals

1. Chandan Giri, B. Mallikarjuna Rao and Santanu Chattopadhyay, "*Split Variable-Length Input Huffman Code (SVIHC) With Improved Application to Test Data Compression for Embedded Cores in SOCs*", Submitted to **Journal of Electronics**.

2. Chandan Giri, B. Mallikarjuna Rao, Nikhil Reddy Cheruku and Santanu Chattopadhyay, "*System-on-Chip Test Data Compression using Huffman*

Coding with Compression ratio and Scan-Power Trade-off", Submitted to **The Computer Journal**.

3. Chandan Giri, Soumojit Sarkar and Santanu Chattopadhyay, "*A Genetic Algorithm Based Heuristic Approach for SOC Test Scheduling with Power Constraint*", Submitted to **The CSI Journal on Computer Science and Engineering (JCSE)**.

Author's Biography

Chandan Giri was born in West Midnapur district of West Bengal on 22nd December, 1976. After finishing his schooling in 1994, he obtained B.Sc(H) degree in Physics from Calcutta University in 1997. Then he got his Bachelor of Technology(B.Tech) in Computer Science & Engineering from Calcutta University, India in 2000 and subsequently Masters of Engineering(M.E) in Computer Science & Engineering with specialization in Image processing from Jadavpur University, Kolkata, India in 2002. He served as a lecturer in an Engineering college for a duration of three years just after his masters degree. Since July, 2005 till date, he is a research scholar in the *Department of Electronics & Electrical Communication Engineering*[*], *Indian Institute of Technology, Kharagpur*[†], India, a premier institute for engineering and technology. His current area of research includes low power testing of digital circuit and system-on-chip (SOC) testing issues. He is associated with live project funded by esteemed Government agency Department of Science and Technology (DST), India. During his research, he presented papers in many International Conferences. He can be contacted at: `chandan@ece.iitkgp.ernet.in` & `chandangiri@gmail.com`.

[*] http://www.iitkgp.ac.in/departments/home.php?deptcode=EC
[†] http://www.iitkgp.ac.in/