EFFICIENT TEST SOLUTIONS FOR SYSTEM ON CHIP

A THESIS

SUBMITTED IN FULFILLMENT OF THE REQUIREMENT

FOR THE AWARD OF DEGREE OF

DOCTOR OF PHILOSOPHY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

BY

HARPREET VOHRA

Reg. No. 950906046

SUPERVISORS

Dr. INDRANIL SENGUPTA
PROFESSOR,
DEPT OF COMPUTER SCIENCE AND ENGG.
IIT,KHARAGPUR,
WEST BENGAL

Dr. AMARDEEP SINGH
PROFESSOR & HEAD,
DEPT. OF COMPUTER ENGG.
PUNJABI UNIVERSITY,
PATIALA, PUNJAB

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR UNIVERSITY, PATIALA – 147004 (INDIA)

2017

DECLARATION

I, Harpreet Vohra hereby declare that the work which is being presented in this thesis entitled "*Efficient test solutions for System on Chip*" infulfillment of requirement for the award of the Degree of Doctor of Philosophy submitted at Electronics and Communication Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Prof. (Dr.) IndranilSengupta, Department of Computer Science and Engineering, IIT, Kharagpur and Prof. (Dr.) Amardeep Singh, Head, Department of Computer Engineering, Punjabi University, Patiala. The matter presented in this thesis has not been submitted either in part or full to any other university or institute for the award of any other degree.

Date: 2nd November'17

(Harpreet Vohra)
Reg. No. 950906046

CERTIFICATE

It is certified that the work content in the thesis titled "*Efficient test solutions for System on Chip*" by Ms. Harpreet Vohra has been carried out under our supervision and that this work has been submitted elsewhere for any other degree.

(Dr. Indranil Sengupta)
Professor,
Department of Computer science and Engineering
Indian Institute of technology,
Kharagpur, West Bengal

(Dr. Amardeep Singh)
Professor & Head,
Department of Computer Engineering
Punjabi University,
Patiala, Punjab

ACKNOWLEDGEMENT

A Ph. D thesis submission is one of the most fulfilling experiences in the career of any researcher as this is not only the most challenging task but also the most accomplishing moment. This thesis is the epitome of the hard work, commitment and dedication that I have put in my research, which has enlightened me to discover new dimensions of knowledge.

Now that the moment of completing a chapter in my life has arrived, I would like to extend my heartfelt gratitude and appreciation to the people who have helped me bring this study to reality. I would like to extend my profound gratitude to my supervisors Dr. Indranil Sen Gupta & Dr. Amardeep Singh. Both of them have helped me with their valuable advice and guidance & encouraged me consistently to deliver to the best of my ability.

I would also like to thank honorable director Dr. Prakash Gopalan, Thapar University for providing me a chance to enroll and complete the study.

I pay my gratitude to honorable Dean, Dr. O. P. Pandey of Thapar University for guiding me about the system & guidelines of the university.

Special thanks to honorable Head of Electronics and Communication Engineering department Dr. Alpana Agarwal of Thapar University for her guidance and support.

Sincere thanks to honorable Ex. Head & Prof. Dr. Sanjay Sharma of Thapar University for his consistent encouragement and motivation to complete my research.

I am thankful to all the members of my doctoral committee. I extend my special thanks to Dr. Seema Bawa, professor, Computer Science and Engineering department, Thapar University for her valuable suggestions and cooperation during the course of Ph.D work.

Pursuing Ph. D was both pleasant and painful experience for me. It was pleasant as it gave me immense learning experience, but this journey was painful as I lost my mother when I was in the completion stage of my research work. It was the most difficult time of my life as she was the only person who believed in me even at the time when every other person had lost their trust in me. I owe my study to my father Er. Sat Pal Vohra and my mother Late Mrs. Usha Vohra. I would like to dedicate my Ph. D to my mother who always blesses me from heaven.

I am greatly indebted to all my family members for their encouragement and support.

Words are not enough to express my heartfelt gratitude towards my husband Mr. Sunil Bhageria and my little sons(Akshit and Kuber) who remained the constant source of love, support and inspiration.

I owe my special thanks to mother in law Mrs. PromilaBhageria for being a consistent support during the entire journey. I would like to thank my teachers, friends, colleagues, students and all my well-wishers who have helped me in one way or other during the journey of this research work.

Last but not the least, I thank almighty to enable me reach this far. Had he not helped me, I would not have been able to achieve my mother’s dream.

Harpreet Vohra

Patiala

ABSTRACT

To cope up with the ever growing demands of the market, more and more number of components are being integrated on a single chip. Realization of complete system comprising of heterogeneous mix of digital and analog logic blocks on a common platform has become feasible owing to the advancements in the design and fabrication techniques. Embedded core based integrated design style has proven to be an effective solution for curtailing the production time and needed design effort for creation of such systems. It involves integration of pre- designed and pre- verified IP cores on a single silicon platform to constitute a system on chip or the so called SoC. Such IP cores can be provided either by the same or some third party design houses in hard, soft or firm information formats.

Advancements in the semiconductor manufacturing technology have led to the development of 3D structures wherein multiple dies are stacked together in vertical dimension. Such structures offer advantages of much higher complexity, lower footprint, lower average power and increased performance. The interconnections between the various dies are provided using low capacity, high density conducting nails called through silicon vias (TSV). Based on the circuit partitioning, 3D SoCs can be categorized as fine grain partitioned SoCs or coarse grain partitioned SoCs. In fine grain partitioned SoC design style, the various core elements may be distributed over multiple dies while in coarse grain, the cores are still 2D but the entire system can be spread over different dies. The necessary interconnections for functional access, power distribution etc. is made possible by using the through silicon vias. Another technological advancement that has taken place in semiconductor industry is the introduction of on chip network as an alternative for bus based interconnects. Being free from different parasitic effects, it offers the advantages of high performance, efficient utilization of bandwidth, low latency, increased throughput, low power dissipation etc. However, such high end design and manufacturing technologies have brought in new design and test challenges.

Manufacturing of billions of components at deep sub-micron level is prone to numerous defects associated with imperfect resolution, misalignment, occurrence of shorts, bridging etc. Such defects can lead to faults which may further degrade the functionality and performance of the systems. To test all the components, they need to be accessed using various input pins and fed with appropriate test patterns. The quality of the systems can be decided by comparing the pre-saved desired outputs with the response of the system to the test patterns which are specifically designed to test the various faults. Increasing level of

integration has led to multifold increase in the amount of possible defects which further increases the test data volume. Generation or storage of such a huge test data increases the test cost. Meanwhile, it also increases the time needed to transport the test data between its source and the SoC periphery. Increase in the difference between the on chip components and SoC pins has led to the controllability and observability issues for individual test points.Likewise, other challenges faced by test engineers include: increasing test power, limited test bandwidth, heterogeneous mix of logic styles, increasing on-chip frequency etc. which are making the entire test process a tedious task. To efficiently address the problem of testability of such systems, modular test approach has proven to be a promising solution. It comprises of test infrastructure consisting of test wrappers and test access mechanism which help in providing the necessary isolation and application of test data to the circuit under test. The test solution so developed needs to be optimized to save the test cost. Test time being an integral component of the overall test cost can be reduced by concurrently testing multiple cores. Meanwhile, it is important that the test schedule should not violate the various constraints imposed by test power, test bandwidth, precedence, hierarchical status of the cores etc. The problem of test architecture development for 2D SoCs need to be adapted for 3D SoCs to address the constraint set by maximum number of TSVs available for test purpose. The main objective of the proposed research work has been to develop an efficient test solution for the system on chip. The key contributions include the development of new test data volume minimization techniques and efficient test architecture for 3D SoCs.

This thesis describes five test data compression scheme namely, 10 Coded run length based encoding, Selective Count Compatible Run length encoding, Hierarchical Block Merging based Run length encoding, Optimal Selective Count Compatible Run Length Encoding and Adaptive Block Merging Based Run Length Encoding. All the techniques attempt to improvise over the previously proposed test data compression techniques so as to increase the achievable compression efficiency and reduce the test application time. The minimization of test data is done by employing encoding schemes that merges the test data at block and intra block levels. Existence of compatibility and other properties among the test data blocks are utilized to develop the suitable codewords which can replace the long sequence of test bits. Such schemes reduce the needed ATE memory and test application time.

A test wrapper optimization technique for fine grain partitioned 3DSoC is presented that reduces the test time by appropriately balancing the wrapper chains. Insertion of the various functional elements is done such that the cumulative TSV requirement of all the wrapper

chains does not exceed the maximum TSV limit. An efficient test solution for coarse grain partitioned 3D SoC is also presented that reduces the test cost by successively optimizing the test architecture at die levels.

To show the effectiveness, the proposed techniques have been applied to different ISCAS'89 and ITC'02 benchmark circuits. The results so obtained are compared with other previously proposed techniques. Various metrics used to gauge the competence of the test compression techniques include the test compression efficiency, decoder area, test application time and test power. Similarly, the test lengths of the complete SoCs are used to show the effectiveness of the test architectures.

TABLE OF CONTENTS

Declaration		
Certificate		
Acknowledgement		*i*
Abstract		*iii*
CHAPTER – 1	**INTRODUCTION**	**1-24**
1.1	Introduction	1
1.2	Modular System on Chip Test Model	5
	1.2.1 Test Data Volume and Its Minimization	8
	1.2.2 Core Test Wrapper	9
	1.2.3 Test Access Mechanism	12
1.3	Thesis Organization	22
CHAPTER – 2	**LITERATURE REVIEW**	**25-51**
2.1	Introduction	25
2.2	Code Based test data compression techniques	25
	2.2.1 Don't care filling techniques	26
	2.2.2 Statistical encoding techniques	29
	2.2.3 Run Length coding techniques	31
2.3	Test architecture development	34
	2.3.1 Test Wrapper Design and optimization	35
	2.3.2 Test Access Architecture Optimization	38
2.4	Investigation of Network on Chip based system on chip testing	45
2.5	Gaps in the Existing work	47
2.6	Objectives of proposed work	
2.7	Summary	51
CHAPTER – 3	**METHODOLOGY OF EFFICENT TEST DEVELOPMENT FOR SoC**	**52-99**
3.1	Introduction	52
3.2	Proposed Test Data Volume Minimization Techniques	53
	3.2.1 10 Coded Run length encoding technique	54
	3.2.2 Selective Count Compatible Run length encodingtechnique	56
	3.2.3 Hierarchical Block Merging based encoding technique	57
	3.2.4 Optimal selective Count Compatible Run length Encoding technique	62
	3.2.5 Decompressor architectures	63
	3.2.6 Adaptive Block Merging test data compression	69

3.3 Development of Wrapper Design for Fine Grained Partitioned 3D System on Chip 73
3.3.1 Problem formulation of fine grain partitioned test wrapper design and its proposed solution 74
3.4 Test architecture development for Coarse grain partitioned 3D SoC 80
3.4.1 Problem formulation of 3D SoC testing 82
3.4.2 Test Solution for 3D SoCs with fixed but soft inter and flexible intra TAM architectures 87
3.4.3 Test solution for 3D SoCs withflexible inter and flexible intra TAM architecture 96
3.5 Summary 98
CHAPTER – 4 RESULTS OF TEST DATA VOLUME COMPRESSION TECHNIQUES 99-110
4.1 Introduction 99
4.2 Results and observations 99
4.2.1 Test data compression efficiency 99
4.2.2 Decoder area overhead 105
4.2.3 Test application time 106
4.2.4 Scan-in Test Power 108
4.3 Summary 109
CHAPTER – 5 RESULTS OF INTEGRATED TEST ARCHITECTURE FOR 3D SoC 111-123
5.1 Introduction 111
5.2 Test wrapper design for fine grain partitioned 3D SoC 111
5.2.1 Experimental Set up 111
5.2.2 Results and Observations 112
5.3 Test architecture for Coarse grain partitioned 3D SoC 114
5.3.1 Experimental Set up 114
5.3.2 Results and observations 123
5.4 Summary 123
CHAPTER – 6 CONCLUSIONS AND FUTURE SCOPE 124-126
6.1 Conclusions 124
6.2 Future Scope 125
References 127-139
List of Publications 140

LIST OF FIGURES

Sr. No.	Figure Details	Page No.
Figure 1.1	Conceptual view of 3D structure with multiple towers	2
Figure 1.2	SoC Test Model	7
Figure 1.3	Test data compression- decompression model	9
Figure 1.4	Conceptual view of the test architecture of SoC	10
Figure 1.5	Conceptual view of: a) Test shell b) P1500 Test wrappers	11
Figure 1.6	Conceptual view of the test wrapper cell for hierarchical cores.	12
Figure 1.7	Test Architecture Designs styles :(a) Multiplexing Architecture (b) Daisy chain architecture and (c) Distributed architecture	13
Figure 1.8	TAM types: (a) Test bus (b) flexible TAM with fork and merge (c) Test rail	14
Figure 1.9	Test scheduling techniques a) session based (b) session less with run to completion and (c) preemptive	15
Figure 1.10	Conceptual view of TAM design for 3D SoC with flexible inter and intra TAM	16
Figure 1.11	A test plan for the 3D SoC capable of executing the testing at the pre bond, post bond and post bond test	16
Figure 1.12	DFT test wrapper for Coarse grain partitioned 3D SoC	17
Figure 1.13	Conceptual view of SoC having an on chip network	19
Figure 1.14	Regular and irregular network topologies a) Mesh b) Torus c) Binary Tree d) Octagon	19
Figure 1.15	NoC phit size variance according to the variation in the frequencies	20
Figure 1.16	Conceptual view of the switch design for NoC	21
Figure 3.1	The codeword generation using SCCPRL	57
Figure 3.2	Pseudo code of HBMTC	61
Figure 3.3	Decompressor Architecture of OSCCPRL	65
Figure 3.4	FSM of Decompressor Architecture of OSCCPRL	66
Figure 3.5	Decompressor architecture of HBMTC	68
Figure 3.6	FSM of Decompressor architecture of HBMTC	68
Figure 3.7	Flow Chart of ABMTC	70
Figure 3.8	(a) 3D model of a circuit partitioned IP core b) Possible formation of the wrapper chain	77
Figure 3.9	Pseudo code for scan chain insertion in wrapper chains	78
Figure 3.10	Di-graph for various scan chains	79
Figure 3.11	Di-graph to represent the scan chain and layer connectivity	79
Figure 3.12	Possible stitching of the scan chains	79

Figure 3.13	Pseudo code for a) input scan cell insertion and b) output scan cell insertion	80
Figure 3.14	Test architecture of 3 D SoC	81
Figure 3.15	Conceptual view of the test architecture problem of 3D SoC	83
Figure 3.16	Examples of 3D SoCs with hard inter and flexible Intra die TAM test architecture with 2D dies supporting (a) Test Rail and b) Test Bus TAM Architectures	87
Figure 3.17	Flowchart for performing TAM architecture initialization	89
Figure 3.18	Flowchart for TAM optimization stage 1	90
Figure 3.19	Flowchart for TAM optimization stage 2	91
Figure 3.20	Flowchart for TAM optimization stage 3	92
Figure 3.21	Flowchart for 3D TAM optimization	93
Figure 3.22	Flowchart for 3D SoCs with flexible inter and intra TAM architectures	97
Figure 4.1	Frequency of occurrence of unique and compatible blocks for block size =10	100
Figure 4.2	Frequency of occurrence of the special 8 cases among the unique cases for various benchmark circuit with block size =10 bits.	100
Figure 4.3	Frequency of occurrence of normalized values of the various special cases among the inter group one, two and three.	101
Figure 4.4	Frequency of occurrence of inter block merging and unique case for different benchmark circuits with fixed block size =8 bits	103
Figure 4.5	Frequency of occurrence of special eight cases for different benchmark circuits with block size = 8 bits	103
Figure 5.1	3D SoC formed by stacking multiple dies having independent SoCs	121
Figure 5.2	Graphical comparison of TSV usage corresponding to different placement of cores for a) d695 and b) p22810 and c) p93791	122

LIST OF TABLES

Table No.	Table Description	Page No.
3.1	Encoding scheme of 10 coded compression technique	55
3.2	Example to demonstrate 10C encoding compression efficiency over 9C	55
3.3	Encoding scheme of test data compression using Hierarchical block merging Technique	59
3.4	Encoding scheme of Optimal Selective Count Compatible Run Length encoding technique	63
3.5	Inter/ Intra block merging at 32 bits level	71
3.6	Unique-Intra block merging at 16 bits level	71
3.7	inter/ intra block merging at 8 bits level	72
3.8	Inter/ intra block merging at 4 bits level	73
3.9	Encoding of 32 bits unique block	73
4.1	Compression efficiencies (in percentage) achieved by application of HBMTC for different block sizes.	101
4.2	Comparison of achievable compression efficiencies (in percentage) between HBMTC and various other techniques	102
4.3	Comparison between OSCCPRL and other test-independent compression techniques in terms of compression efficiency (in percentage).	104
4.4	Comparison of decompression area overhead achieved between proposed compression schemes and the various other techniques	106
4.5	Comparison ofTest application time between proposed compression schemes and the various other techniques	107
4.6	Comparison ofScan-in peak power transitions between ABMTC scheme and various other techniques	108
4.7	Comparison ofScan-in average power transitions between ABMTC scheme and various other techniques	109
5.1	Test Wrapper results for core 7 of SOC d281 in comparison to [94]	112
5.2	Test Wrapper results for core 7 of SOC d281 in comparison to [96]	112
5.3	Test Wrapper results for core 4 of SOC p93791 in comparison to [94]	113
5.4	Test Wrapper results for core 4 of SOC p93791 in comparison to [96].	113
5.5	Test time calculation for benchmark d695 having two dies and two partitions for thermally / power unaware cases.	115
5.6	Test time calculation for benchmark d695 having two dies and two partitions for thermally / power aware cases.	115

5.7 Test time calculation for benchmark p22810 having three dies and three partitions for thermally / power unaware cases. 116

5.8 Test time calculation for benchmark p22810 having two dies and two partitions for thermally / power aware cases. 116

5.9 Test time calculation for benchmark p93791 having four dies and four partitions for thermally / power unaware cases 117

5.10 Test time calculation for benchmark p93791 having four dies and four partitions for thermally / power aware cases. 117

5.11 Results of d695 for case P_{SS} for cores spread over 3 dies 118

5.12 Results of p22810 for case P_{SS} for cores spread over 3 dies 118

5.13 Results of p93971 for case P_{SS} for cores spread over 3 dies 118

5.14 Test time calculation for 3D SoC with multiple test schedules in comparison to results of [45] 120

LIST OF ABBRIVIATIONS

SoC	System on chip
IP	Intellectual Property
HDL	hardware descriptive language
IC	Integrated Circuits
TSV	Through silicon vias
NoC	Network on chip
2D	Two dimensional
3D	Three dimensional
ITRS	International technology roadmap of semi- conductors
BIST	Built in self-test mechanism
ATPG	Automatic test pattern test pattern generator
ATE	Automatic Test Equipment
CUT	Circuit under test
I/O	Input / Output
DFT	Design for testability
EDA	Electronic Design Automation
TAM	Test Access Mechanism
CTDC	Code based test data compression
GALS	globally asynchronous and locally synchronous
NI	network Interface
10C	10 Coded compression technique
SCCPRL	selective count compatible run length encoding
HBMTC	Hierarchical block merging based test data compression technique
OSCCPRL	Optimal selective count compatible run length encoding
ABMTC	Adaptive block merging based test data compression technique
R-Fill	Random Fill
MT Fill	Minimum Transition Fill
CBS	Column-wise bit stuffing
FDR	frequency directed run length
EFDR	Extended Frequency directed run length encoding technique
9C	Nine Coded
PRL	Fixed Length Pattern Run Length
CCPRL	Count Compatible Pattern Run Length
BM	Block Merging Technique
BM-8C	Block Merging with Eight Coding technique

LPSC	low power selective pattern compression
UDL	User Defined Logic
WC	wrapper chains

1. INTRODUCTION

1.1 Introduction

Ever growing advancements in the field of the semiconductor industry, design techniques and CAD tools have led to an increase in the number of on-chip components. Owing to such advancements, the semiconductor industry has been able to sustain Moore's law. Systems which were spread over different PCBs are now being commonly found on a single chip or what we call as System on chip (SoC) [1-2]. To cater to the escalating needs of the consumers and serve the shorter time to market, formation of the VLSI design from scratch is no more a practical solution. In order to design complex system chips in a timely manner and to alleviate the need of external design expertise, the reusable core based embedded solutions have emerged as the most promising and feasible ones [3]. In embedded core-based design approach the predesigned and pre-verified Intellectual Property (IP) cores which are either designed in house or bought from the third party are integrated to realize a complete system. The information of the IP cores can be provided to the system integrator in one of three formats, which include: technology specific layout description called hard format,technology independent hardware descriptive language (HDL) called soft format or in between the two called firm format [4].The system integrator needs to integrate various IP cores along with needed interconnects and user defined blocks to realize a complete system. Example of different varieties of cores includes digital signal processors, media processors, mixed signal modules, and memories etc. SoCs provide the benefits ofmore functionality, higher performance, minimum average power, minimum average interconnects length, smaller footprint and heterogeneous mix of analog and digital circuitry on the same chip.

Increase in the on-chip circuit complexity and clock frequency has led to the reduction of the mean free path of current carriers in conductors. As a result, the on-chip interconnects have started introducing various parasitic effects making them behave as a transmission line instead of pure conductors [5]. At deep micron level, the conductors lead to issues like performance degradation and increase in system's power dissipation.

The problem of long interconnects have been solved to some extent by the advancing the Integrated Circuits (IC) manufacturing in the third (vertical) dimension. Multiple two dimensional (2D) dies are stacked together to realize more complex three dimensional (3D) System on chips [6-7]. Stacking of dies is either done in face to face or face to back order where vertical interconnects are supported by wire bonds or Through silicon vias (TSV). TSVs are high density conducting nails that spread out of the back side of one die to allow

vertical interconnections with other dies. They consume less power, offer low capacitance and thus are capable of catering to high frequency applications. The dies can be mounted either on top of each other to form a single tower or side by side like parallel towers ona common substrate. An example to show the conceptual view of a formation of 3D structures is given in figure 1.1. The introduction of the 3D SoCs has provided many benefits that include smaller footprint, improved performance, lower power etc. [7-8]. Based on the circuit partitioning, 3D SoCs can be categorized as fine grain partitioned SoCs or coarse grain partitioned SoCs [9]. In fine grain partitioned SoC design style, the various core elements may be distributed over multiple dies while in coarse grain, the cores are still 2D but the entire system can be spread over different dies.

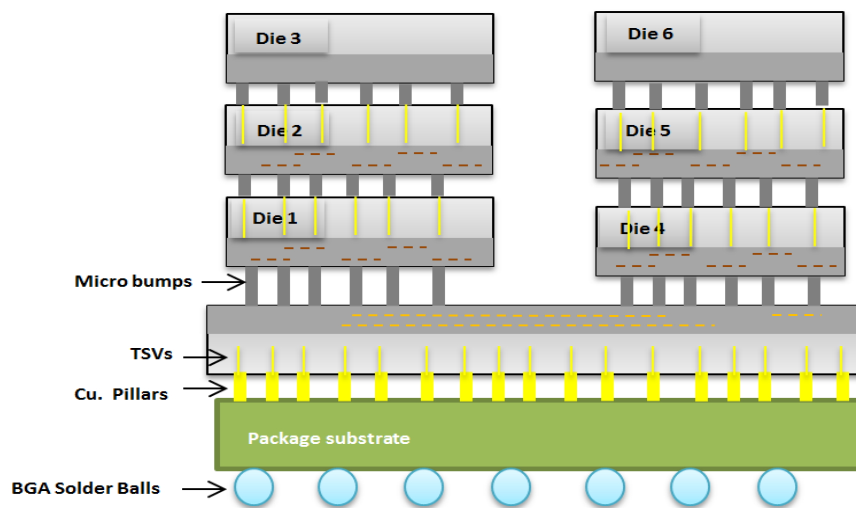


Figure 1.1 Conceptual view of 3D structure with multiple towers

Another technological advancement that has taken place to cope up with the issue of wiring crisis is the use of on-chip network [10-11]. Network on chip or the so called NoChas emerged as a new interconnect paradigm [12]. It is capable of transporting (transmitting and receiving) the data between the different on-chip blocks in the form of packets [13]. On chip IP cores attached to a network can communicate using network interfaces, routers and their associated links. Recent advancements have also led to the introduction of 3D NoC by combining the merits of 3D manufacturing technology and network on chip [14].

Manufacturing of exceedingly complex systems at deep submicron level with high resolution is normally prone to numerous possibilities of defects like shorts between the wires, imperfect doping, bridging etc. [15-16]. In addition to these, the 3D fabrication steps like thinning, alignment, bonding etc. are also susceptible to the imperfections [17-18]. Such defects lead to faults which ultimately can lead to performance or functional failure of the

devices. For the quality assurance of the developed systems, it becomes necessary to do the testing prior to its delivery to the market. Test vector (stimuli) generation, its application to all the on-chip test points and the corresponding analysis of their response need to be done to determine whether the device is good or bad.

With manufacturing at deep submicron level for 2D or 3D SoC designs, the possibility of occurrence of defects have increased by many folds which ultimately lead to increase in the test data volume. As per the ITRS roadmap, the amount of test data is likely to increase by 25 times in coming years [19] which become a bottleneck of the test application time reduction. Meanwhile, the generation of this much amount of test data with sufficient fault coverage needs a very efficient test pattern generation mechanism. The possible source of test vectors can either be an on-chip Built in self-test mechanism (BIST) or some external software based Automatic test pattern test pattern generator (ATPG) [4]. BIST approach uses on-chip linear feedback shift register based hardware to generate the pseudo random test patterns [15]. Multiple input signature analyzers can be used for test response compaction and analysis. For the generation of test data specific to the circuit under test, the BIST design has to be customized. IP cores like memory can be efficiently tested by using a BIST. However, incorporation of an on-chip BIST may not be very efficient and feasible all the time. The IP cores being bought from third party vendors are normally imported in hard or firm form. Such descriptions don't reveal much about the structural details of the core which may make the installation of the BIST to be infeasible. Alternative to the BIST approach is to either use software based automatic test pattern generator (ATPG) or the pre-generated test sets provided by the core vendor. Such ready to use test data can be saved externally in the memory of Automatic Test Equipment (ATE)and can be further supplied to the Circuit under test (CUT)at the time of testing [4, 15]. Being immune to underlying structural details, ATPG approach has emerged as the conventional test pattern generation technique.

Ironically, the designing of the Automatic test equipment is not being advanced at the same pace as that of the CUT. For example: the frequency and memory capacity of an ATE are not usually increased at the same rate as that of the system’s clock and on chip complexity. The huge amount of test data necessitates theATE memory to be increased which makes its installation to be quite expensive. At the same time, application of large test data volume further increases the test time of the circuit under test. The increase in on-chip complexity of the designs leads to issues in accessingvarious test points. The limited number ofavailable pins for test purpose necessitates the application of the test data to be done serially which

further increases the test transportation time. Limited Input /Output (I/O) channels (dedicated for test purpose), huge test data volume and low ATE clock frequency aggravates the problem of rising test cost [16]. Similarly, in the case of the 3D SoCs, the limited number of TSVs for test access work as a constraint for its test plan development [17].

In order to reduce the test time, more number of cores should be tested concurrently. However, activation of a large number of on-chip blocks simultaneously for test purpose increases the test power which, generally doesn't happen during the normal functional mode of the system. If the peak or average test power exceeds the maximum power dissipation limits then the device performance and functionality can be affected. Under worst circumstances, it can lead to the thermal runway as well. Apart from the above mentioned issues, there are various other constraints (posed by the resource conflicts, precedence, and hierarchy, multiple test sets associated with the cores etc.) that need to be addressed while developing an efficient test solution for the system on chip. A summary of the various test challenges posed by the increase in level of integration are as follows:

- The heterogeneous mixof technologies: A SoC may contain cores with logic, processors, memory and analog design technologies. All these cores must be tested after the SoC is manufactured. It is almost impossible for a system integrator to develop all the tests alone. Hence, assistance from the core providers should be available and a standard way of communication between the SoC integrator and core provider should be established.
- Deeply embedded cores: A core may be deeply embedded in a chip which makes its terminals to be difficult to control and observe. Hence, some kind of mechanism is required using which the various test points can be efficiently accessed.
- A mix of flat and hierarchical cores:As per the structural details of a core, it can be categorized as either a hierarchical core or flat core depending on whether it contains other cores embedded within it or not.Hence, the various test wires allotted to the core need to be optimally distributed among the various core componentsbased ontheir hierarchical status.
- Different core providers: The core information provided by the different vendors can be in hard, soft or firm format.Therefore, some kind of design and test standard is essential for the integration purpose.

- IP Protection/ test reuse: Detailed internal structure of the core is usually unavailable to the SoC integrator due to the IP protection consideration. It is thus desirable to be able to reuse the test provided by the core developer with very limited or no modifications at all. A standard core test interface and protocol are also essential to address this problem.
- Higher performance core Input/ outputs than SoC pins: The clock rate inside a core can be significantly higher than the usual test clocks provided by the external testers. At the same time, the system clock can be slower than the clock of the core. Such a scenario usually cannot support at speed testing and hence it necessitates the use of normal functional units to create the required at speed test environment.
- Expensive and inefficient external ATE: The installation and maintenance of a high-end ATE are quite expensive. Also, the gigantic increase in the test data volume necessitates the ATE memory to be increased. Such a case enhances the cost of the ATE even more.
- Long test application times: Sequential testing of the cores can lead to long test application times which increases the test execution cost. It delays the time of the system delivery to the market as well.To alleviate such problems, it becomes essential to test as many cores as possible in parallel.
- Large test power: Parallel testing of the cores has an adverse effect on the system test power. Lot many circuit components need to be activated at the same time which leads to a dramatic increase in the resultant switching activity per unit area. Such effects increase the test power dissipation which can degrade the performance and functionality of the system. In a worst scenario, if the instantaneous test power exceeds beyond the peak power which can be handled safely by the packaging and cooling mechanism, then the thermal runway can also occur.
- Testable design automation: Design for testability (DFT) insertion that solves the testing problems is desirable that includes ATPG, standard test circuitry, test architecture and test plan or schedule. A strong support with Electronic Design Automation (EDA) is also needed to lessen the issues faced by the test engineer.

1.2 Modular System on Chip Test Model

To perform the testing of the various cores embedded deep inside the SoC, it is important to reach the site of the fault, apply the test data and extract the test responses which may be further compared against the expected true responses. As the complexity of ICs is increasing,

it is becoming more and more tedious to access the individual points so a careful and efficient test planning needs to be done to ensure the quality of such devices. The basic approaches can be to employ the test point insertion techniques or some ad-hoc based design for testability approaches such as scan based test or built-in self-test [14-15].

Zorian proposed a modular test approach in [20]as shown in figure 1.2. It consists of three main components that help in the application of the test data to the circuit under test, namely: Test data source and sink, Core test wrappers and Test access mechanism. The source and sink of the test data can either be an on-chip BIST [34] or off chip ATE. Both work as test stimuli generator and the response analyzer. The choice between the two approaches can be made based on the flexibility in making the design changes and availability of various resources. The test data can be transported between the ATE and SoC periphery using input/ output channels dedicated for test purpose. Once the data reaches the SoC, it is applied to the various IP cores using on-chip test wires called Test Access Mechanism (TAM). In order to facilitate the application of test vectors carried by TAM to the core under test, an embedded core must be isolated from its surrounding logic for test purpose. Test wrappers form an interface between TAM and core.It works on deserialization of the test data (as received from the TAM) and its application to the respective core terminals (functional input/ output and internal scan chains). Test wrappers design algorithms which attempt to optimize the test time of the core by insertion of its elements in fixed number of wrapper chains has been proven to be NP-hard problem [21].

SoC test model presented in fig. 1.2 consists of different modules namely: PC1, PC2, PC3, and RAM. The test data originating from a test source (ATE etc.) is applied to the circuit under test (CUT) by the on-chip wiresthat serve as test access mechanism. The test response data generated by the circuit under test istransported to the test sink where a comparison between the test responses produced by the CUT and the pre-saved golden responses (desired responses to applied test vectors) is done to decide on the quality of the design. Out of the cores shown in the figure 1.2, PC1 and PC3 are surrounded by test wrappers. Out of the two cores (PC1 and PC3), PC 1 is flat core (with no child cores) while PC3 is hierarchical core (with two child cores: CC1 and CC2). The IP core denoted as RAM is shown to have its own BIST hardware responsible for test vector (specific for testing of RAM) generation and analysis of the test responses. Meanwhile, the core PC2 is shown to be unwrapped.

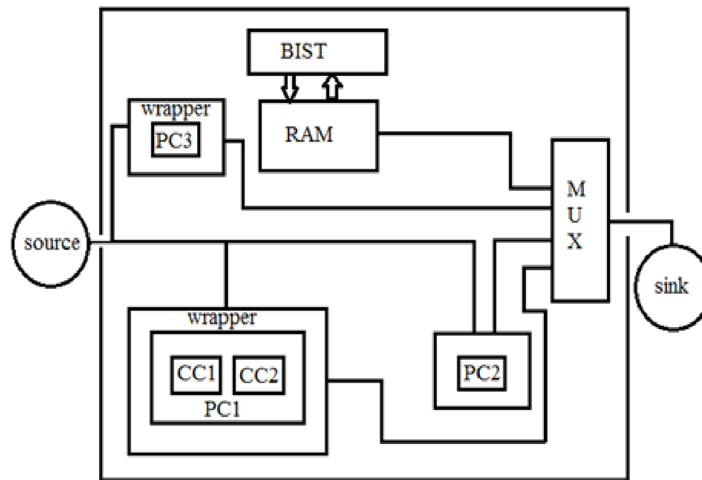


Figure1.2 SoC Test Model

As per the ITRS roadmap, the test cost per transistor is likely to become an integral component of the overall cost of the system. The cost of testing a transistor may even become comparable to the cost of manufacturing it. A generic cost model for testing a system on chip has been described in [22-23]which takes into account the cost involved in preparation of test vectors, installation of ATE, on-chip silicon overhead and execution of test mode. Out of these, the test generation and ATE cost which are expected to be a non-recurring component[24] are largely dependent on the circuit complexity.More advanced and complex the circuit becomes, more will be amount of test data. The silicon overhead is largely dependent on cost involved in the implementation of the Design for testability (DFT) infrastructure on the IC periphery. The execution cost majorly depends on the performance degradation that occurs during operation of the circuit in test mode.

As discussed in section 1.1, increase in the level of integration has led to a multifold increase in the amount of test data. To reduce the amount of ATE memory and test time which largely impact the test cost, test data should be reduced as much as possible [25-26]. The TAM bandwidth distribution, assignment of cores to them and scheduling of various core tests using them put a huge impact on the overall test cost of the SoC. Therefore, an efficient test access architecture design constitutes an integral part of the test plan development of SoC [27].

With introduction of the 3D SoCs, the modular test model applicable for the 2D SoC needs to be adapted to cater to the requirements specific to 3D structure [28]. The interconnect paradigm shift towards the use of on-chip network affects the test model too since the network components like routers etc. need to be tested in addition to various IP blocks [29].

The use of the network on chip as a test access mechanism is being explored in the recent years [30].

Hence, an effective test solution development becomes an integral component in the minimization of the overall test cost of a system on chip. Test time being the major bottleneck of the overall cost should be reduced by implementing different optimization techniques such that various constraints like power, TAM bandwidthetc. are not violated. Its solution can be achieved by sequentially addressing the problems of: minimization of test time needed to transport the test data between ATE, SoC periphery and the individual cores, appropriate distribution of the TAM wires and their assignment to the on-chip IP cores and bringing parallelization while scheduling the test of the cores as much as possible. A lot of research has been done on the development of test cost optimization techniques. A brief overview of various approaches is as discussed below:

1.2.1 Test Data Volume and Its Minimization

Use of ATE has proven to be more promising as compared to BIST due to the reasons discussed above. The test data generated by the software based Automatic test pattern generation is saved on the ATE memory from where it is further routed to the SoC using input/ output channels. With increase in the volume of the test data, more and more ATE memory is required to store it. If the available ATE memory is limited (which holds true since the ATE equipment is not improvised very frequently) [24], then application of the test data demands multiple ATE reloads which aggravates the test time. Meanwhile, the limited number of test pins and slow ATE clock frequency (relative to the frequency of the SoC) increases the time associated with shifting the test data serially.

To address the issues of ATE memory and transportation time it becomes essential to minimize the test data as much as possible. Code based test data compression (CTDC) techniques which encode test data using lesser number of bits have proven to be a promising solution to cope up with the issues of escalating amount of test data [26]. Compressed test data so obtained after application of CTDC can hence be stored in ATE requiring less amount of memory. Encoding data is further applied to DUT using available I/O channels dedicated for test purpose. On-chip decompressors (as shown in figure1.3) are used to retrieve the original test stream. The decoded test data is delivered to various SoC components using TAM.

For a specific fault diagnosis, complete path carrying the fault site needs to be sensitized using appropriate test patterns.

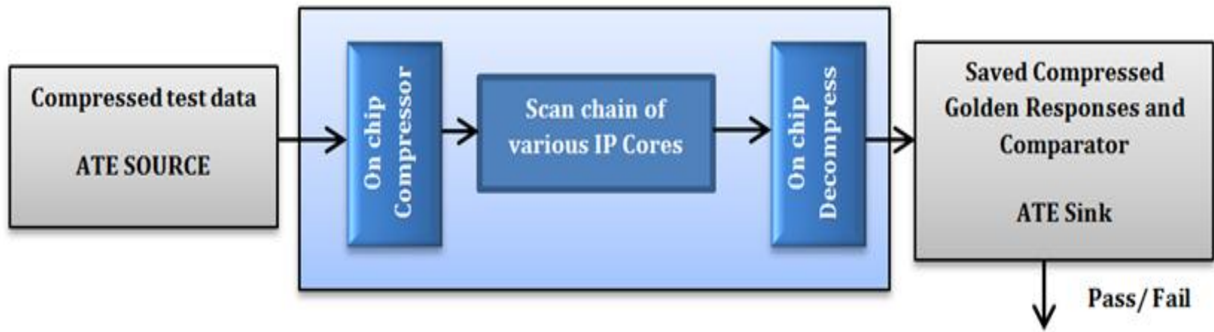


Figure1.3Test data compression- decompression model

For particular fault activation and its path sensitization to ensure the observability of its effect demands only some of the primary inputs to be asserted to specific logic value. Meanwhile, other primary inputs which, if helpful in forwarding the test response to an observable point can be set to known static value else left in don't care state [15-16]. Here, don't care simply means that its status doesn't really impact the fault diagnosis process and can be set to any logic value. As a result, the three possible values that the various circuit pins can attain are: low/zero (*0*)/ high (*1*) or don't care (*X*). Test data consisting of test vectors (specific to the various faults) may have many bit arrangements like long runs of *ones*/*zeroes* or some combinations of *0's*, *1's* and don't care etc.

Various code based techniques have been developed so far to compress the count of test bits. Such techniques utilize the schemes like encoding long runs of fixed bit value, appropriate filling of don't care values etc. for reduction of test bits. For instance, filling the don't care value with a fixed logic value can increase the run length of that type of bit value etc. Section 2.3 reviews the various test data compression techniques. It can be found that code based techniques can also be used to reduce the switching activity likely to occur on scan cells associated with the circuit under test.

1.2.2. Core Test wrapper

A test wrapper facilitates modular testing by isolating core under test from the external environment [31]. It aims at the application of test bits to the appropriate components of the CUT using the serial or parallel interface provided by the TAM wires allotted to it.A conceptual view of a typical test environment for facilitating the testability of various cores lying on a chip is shown in figure1.4. It consists of core/ IP modules, test wrapper and the host. Out of these, the core or IP module is the circuit under test and is expected to include

some DFT circuitry. The host provides the environment and is equipped with a test controller meant for directing the test data and test instructions to the appropriate cores while the Test wrapper connects the core to the host environment. The test wrapper design style shown in the figure1.4 is chosen to be P1500.

The design of the test wrapper revolves around appropriate distribution of various core components(like functional input/ outputs and internal scan chains) to form the wrapper chains (WCs) [31]. It helps in ensuring the testability of the core at the expense of little area and performance overhead. The test time of an individual core directly depends on the time needed to apply the test stimuli and capture its response. It is directly proportional to the number of the test vectors that need to be exercised along with the lengths of the wrapper's input and output scan chains.

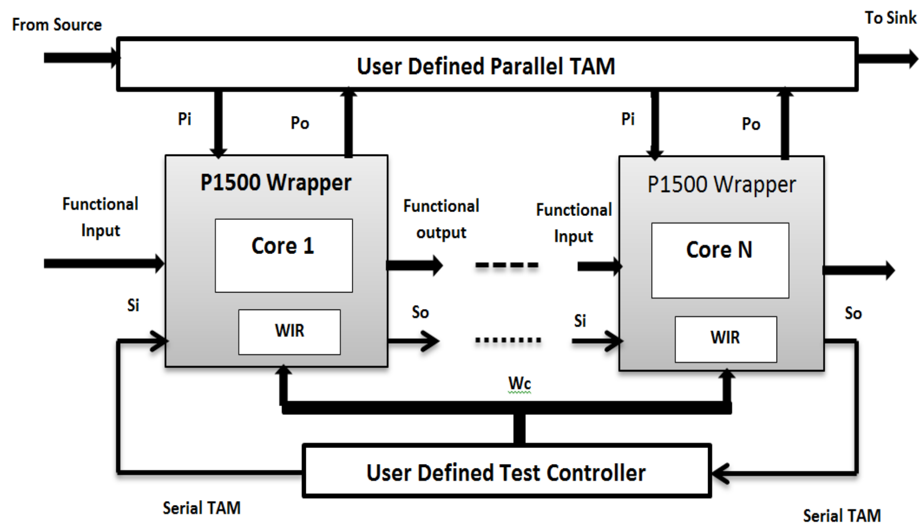


Figure 1.4Conceptual view of the test architecture of SoC

Three scalable and structured test wrappers design styles included Test shell [32] and Test Collar [33]. Establishment of IEEE embedded core test standard introduced P1500[31]style of test wrapper for testing of SoC cores.Figures 1.5 (a) and (b) present the test shell and P1500 styles of test wrapper design.

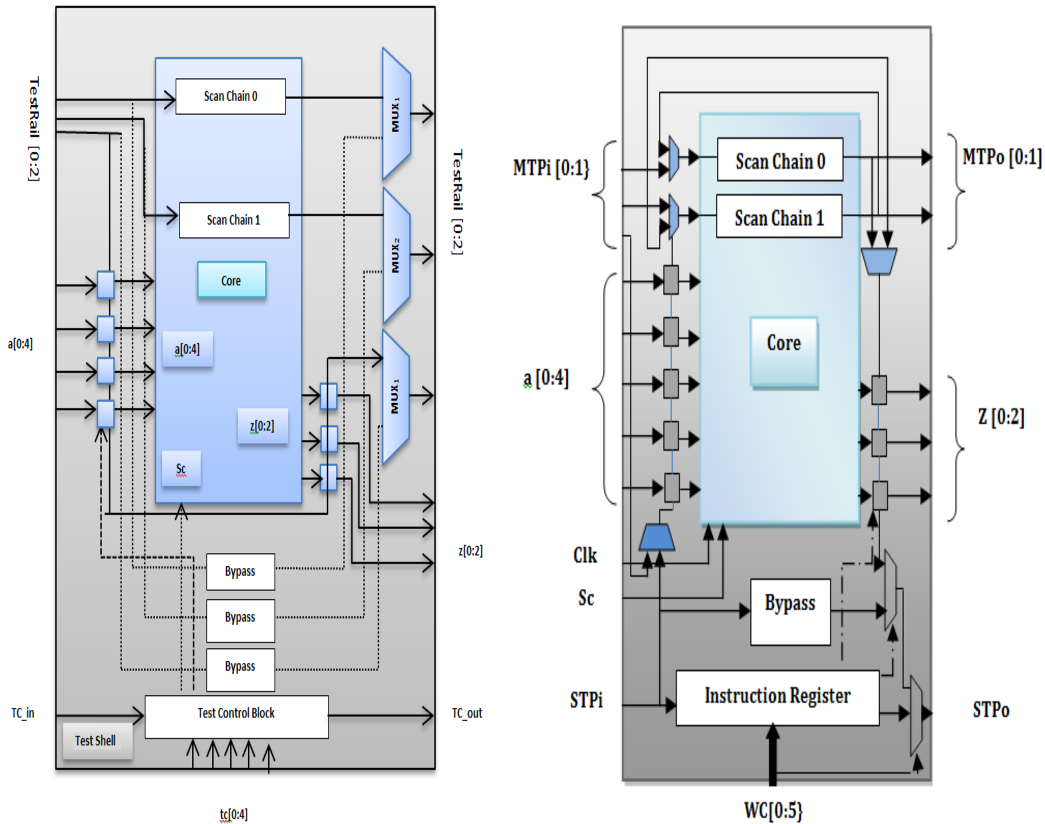


Figure1.5 Conceptual view of: a) Test shell b) P1500 Test wrappers

The test shell (as shown in figure 1.5(a)), is equipped with three terminals namely: functional input/outputs (having a direct correspondence to the normal input/ outputs of the core); test rail input/ outputs which are the test access mechanism for the test shell with variable width, an optional bypass and direct test I/O that are used for signals which cannot be provided through the test rail due to their non-synchronous or digital nature. The bypass feature plays an important role in avoiding the unnecessary switching activity at the various terminals of the core when it is not being tested. Test Collar, approach is similar to the Test Shell. The only difference is the absence of the bypass feature which reduces the flexibility as it restricts multiple cores to be tested simultaneously.

As 1149.1 JTAG standard deals with testing of chips on the board, IEEE P1500 standard deals with test knowledge transfer and the test access to the IP cores of SoC [35-36]. The P1500 wrapper (as shown in figure 1.5 (b)) supports a serial interface for test data and control instruction transfer along with multi-bit parallel ports (which may be optional). The wrapper also supports a bypass register which helps in avoiding the unnecessary switching across the boundary scan cells present at the core's terminals which are not to be tested at that very moment.The additional wrapper instruction register and serial control present in the wrapper design facilitates the wrapper control mechanism. While such styles decide the interface of

the wrapper design, the designing of the internal wrapper chains needs to be optimized to reduce the test time of individual core.

Later, it was observed that the test needs of the hierarchical cores are different from flat cores. Hence, it becomes mandatory to modify the test wrapper design accordingly. Test wrapper design algorithmfor hierarchical cores has been proposed in [37]. It performs the wrapper design based on the needs of the child and parent cores. The TAM bandwidth allotted to the core is redistributed to balance the internal wrapper chains of both parent and child cores. Parallel testing of a parent and its child core(s)at the expense of little expensive wrapper design style is done to optimize the test time of entire core. The wrapper cell design as applicable to carry both the parent and child core's test [38] is as shown in figure 1.6.

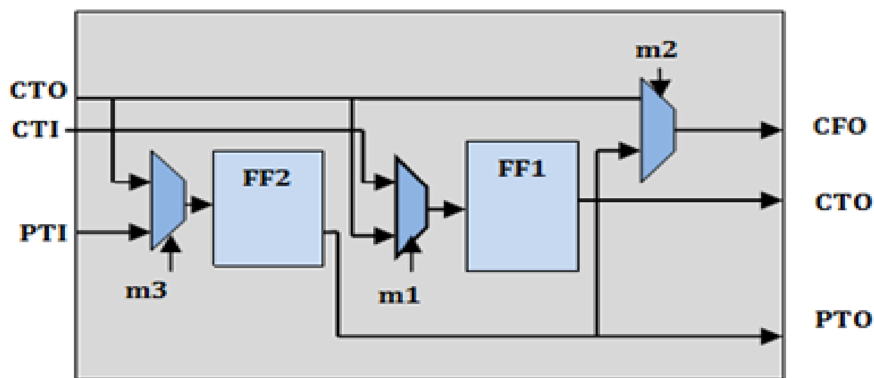


Figure 1.6Conceptual view of the test wrapper cell for hierarchical cores.

The wrapper design problem has been proven to be NP-hard and many [21] optimization techniques have been developed `so far for efficiently optimizing the test time of the individual cores.

1.2.3 Test Access Mechanism

The test architecture design plays an integral role in optimization of testing time of a SoC [21]. It dictates how the test data after being decompressed has to be routed to the different IP blocks. The designing and distribution of the available TAM width are done in a way to ensure the testability of the cores. Various algorithms have been proposed in the past two decades which work on distribution of the TAM wires among the various cores.

Aerts and Marinissen [39]proposed three different buses (shown in figure1.7) based TAM architecture that employs multiplexing,daisy-chaining and distribution of the available TAM width. Multiplexed TAM architecture gives each core access to the full TAM width on time

multiplexed bases (shown in figure1.7 (a)). Such an approach has the benefit of providing complete test bandwidth to each core but, the need of sequential test scheduling (due to time multiplexing) of all the SoC cores increases the test time vividly. The second architecture proposed in [39] is called Daisy chain as shown in figure1.7(b).It connects all cores using one long TAM. Like Multiplexed approach, it assigns the whole test data bandwidth to each core reducing the cores testing time to be as much as possible. Each core has a bypass cell connected in parallel to it which helps in isolation of the cores for the test purpose. Once a specific core has been tested then it can be bypassed during the test of the successive cores so as to reduce the test time. The third approach is the distribution approach as shown in figure 1.7(c). Using this technique the complete TAM is divided into partitions of varying TAM widths. The cores assigned to different TAM partitions can be tested concurrently irrespective of previous two TAM schemes wherein the testing of the cores need to be done sequentially. The major drawback of the distribution TAM architecture over other techniques is that it does not allow the interconnection testing that lies between the different cores.

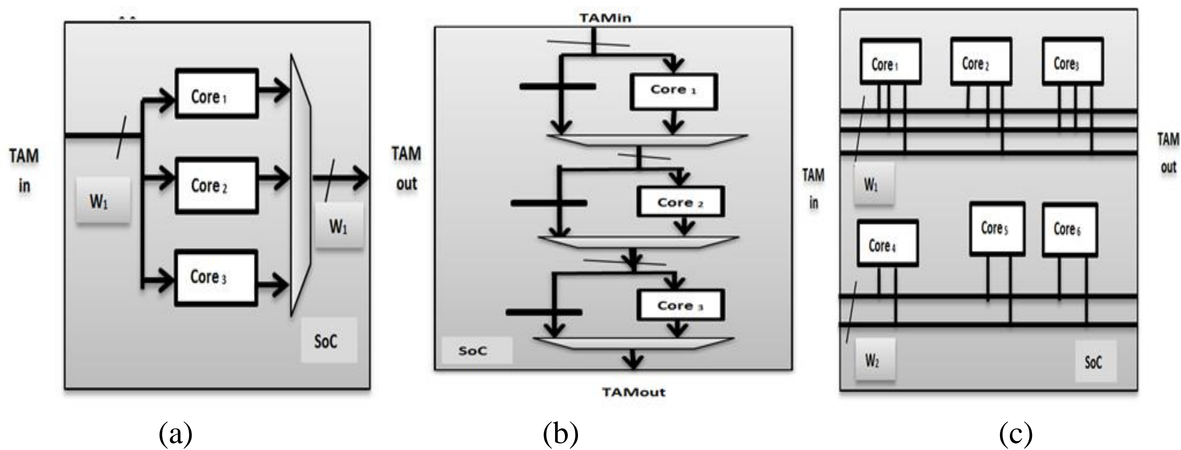


Figure 1.7 Test Architecture Designs styles :(a) Multiplexing Architecture (b) Daisy chain architecture and (c) Distributed architecture

A combination of the multiplexed and distributed TAM approaches is the test bus approach [33] as shown in figure 1.8 (a). As pointed by the figure, the whole TAM bandwidth is partitioned to form different buses. This type of approach offers both the sequential and concurrent testing of multiple cores. Cores allotted to the same TAM have to be tested sequentially while the cores assigned to different TAM partitions can be tested concurrently.However, it has a limitation of inflexibility in test width adaption and core's external testing.

Two variants of the test bus approach [40] are the fixed width test bus (figure 1.8(a)) and flexible width test bus (figure 1.8(b)). In case of the flexible width test bus more flexibility isgiven to the SoC test integrator by allowing a dynamic variation in allotment of test bandwidth to each core. Such an approach can reduce the test time of individual cores which further minimizes the test time of the complete SoC. An amalgamation of the daisy chain and the distribution architectures called test rail architecture is proposed by E.J Marinissen in [32]. Figure1.8(c)presents the test rail type of TAM architecture design. A single test rail works similar to the daisy chain architecture. It allows all the cores to be allotted the maximum available TAM width as per the assigned partition. The bypass path in the individual test buses is same as that of the daisy chain architecture. Cores lying on different test rails can be tested in parallel. The advantage of test rail over test bus approach is that it allows multiple core wrappers to become activated simultaneously and hence allow the external/ interconnect testing. Test Shell style of test wrapper can be used with the Test Rail type of the test access mechanism

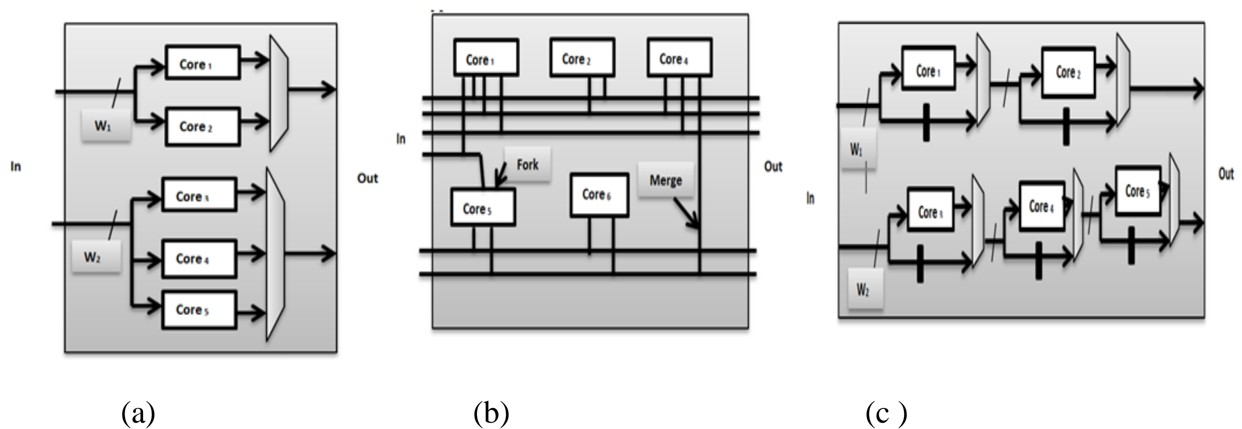


Figure 1.8 TAM types: (a) Test bus (b) flexible TAM with fork and merge (c) Test rail.

The width of a test rail is referred to as the test data width since it governs the overall system testing time. Cores connected to a TAM of type test rail can be tested in parallel or in serial order. A simultaneous test of two cores can be done by concatenating the scan chains and wrappers of the cores.

Once the TAM architecture is designed, next task is to decide the test schedule which determines the start/ stop time for testing of individual cores and the finish time of the overall SoC. The schedule of the test of various cores also helps to determine the sharing of all the test resource among them. Broadly test schedules can be divided into three categories (as shown in figure 1.9), namely: session based testing [41], sessionless testing with a run to

completion [42] and the preemptive testing [43-44]. In figure 1.9, *x* axis presents the test time while the *y* axis represents the TAM allotment. In session based testing (as shown in figure1.9(a), the compatible (in terms of various conflicts like power, resource etc.) cores are grouped together and scheduled in the same session.No new cores can be assigned to a TAM partition until all the cores in the same session finish their testing. Hence, such an approach leads to large idle times of the TAMs which further increases the overall test application time of the complete SoC. Most of the practically used scheduling techniques fall under the sessionless scheduling schemes (shown in figure 1.9(b) and (c)). The test scheduling technique based on run to completion approach is as shown in figure 1.9(b). It assigns new cores to the TAM partitions as soon as they become free. The concurrent testing of various cores assigned to different TAM partitions is done depending upon their compatibility. It reduces the idle time issue faced in session based testing.

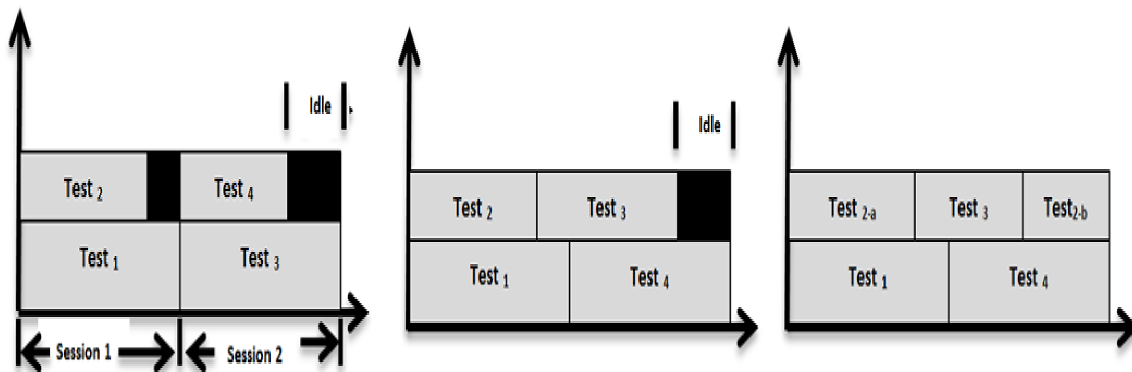


Figure 1.9 Test scheduling techniques a) session based (b) sessionless with run to completion and (c) preemptive.

The pre-emptive test scheduling approach as shown in figure 1.9 (c) is another type of sessionless approach. It allows the intermediate interruption and resumption of various tests of the cores. Such a technique is quite useful in a constraint-driven testscheduleswherein the occurrence of idle blocks (which are incapable of accommodating entire test of a core) is quite frequent.Such a technique can reduce the test application time. However, not all the cores tests are pre-emptive. For instance, the test vectors dedicated fora sequential circuit like memory etc. are dependent on each other and need to be applied in a predefined order. Disruption of such tests can jeopardize the whole process. Another issue related to it is the logic/hardware overhead required for performing the pre-emption which can increase the test cost. Based on the limitations and demerits, pre-emptive scheduling technique is seldom used. Many researchers have proposed solutions to the problem of scheduling the tests of

System on Chip concurrently. The most recent advancements in the 3D technology necessitated the consideration of the number of through silicon vias as another constraint for TAM design. A conceptual view of the 3D TAM design is as shown in figure 1.10.The cores are assumed to be 2D with their placement on different dies.As per the figure 1.10, the number of the dies is shown to be three with their individual TAM partitions specified by individual N_{TAM}. Each TAM partitions are allowed to traverse between the various dies for the necessary test data delivery.

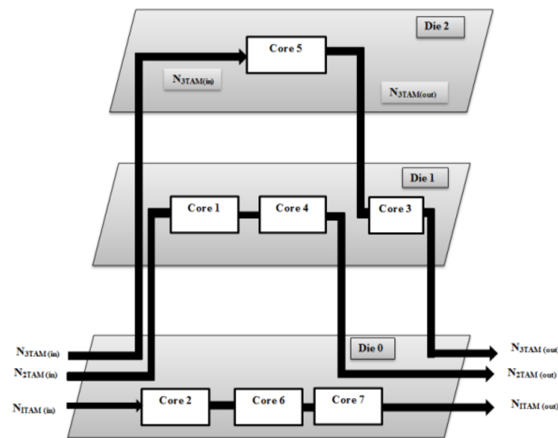


Figure 1.10Conceptual view of TAM design for 3D SoC with flexible inter and intra TAM

A test plan for 3D SoCs as described in[45-46] is shown in the figure 1.11.The authors suggested the need of multiple test insertion of various dies so as to facilitate the testing of complete 3D SoC. To ensure an early detection of faulty components in the manufacturing process, the dies are tested at pre, partial and post bond levels. This way the test cost can be reduced.

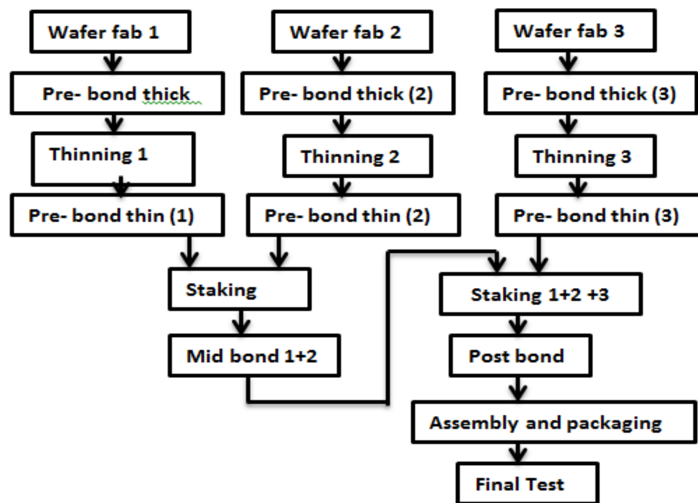


Figure 1.11 A test plan for the 3D SoC capable of executing the testing at the pre bond, post bond and post bond test

A standardized 3D DFT architecture is presented in [47] that provisions modular test approach for the 3D SoC. It allows the testing of various cores and their associated interconnects to accommodate the test plan mentioned given in [45]. The bottom dies being in direct contact with the SoC pins can be tested directly while test elevators are required to carry the test data to the higher dies in the stack. As shown in the figure 1.12, the architecture is P1500 compliant. The test model considers each die as the component around which, a die test wrapper is designed which provides the necessary serial and parallel interface for test purpose. In other words, the various die wrappers are considered as children for the parent 3D DFT architecture. Test data is supplied using I/O channel, from where it is distributed further to the various dies. The wrapper on the die acts as the parent to the test wrappers of the various cores present on the individual dies.

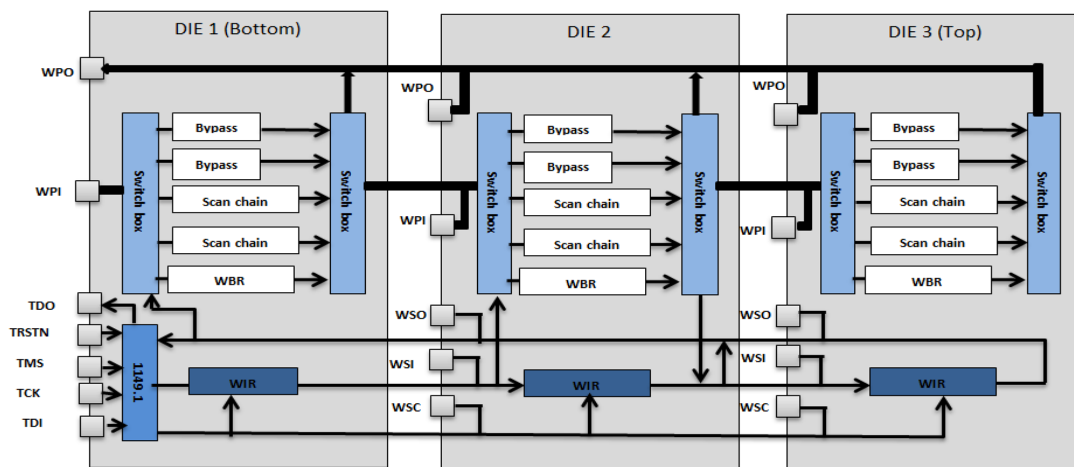


Figure 1.12 DFT test wrapper for coarse grain partitioned 3D SoC

Increasing numbers of on-chip components in a core based SoC have led to a stringent demand on a foolproof interconnect system which should not only deliver the on-chip data accurately and swiftly (in range of Gigahertz frequency) but, also be free from latency and power dissipation. Contrary to a bus based communication between the different IP cores, on chip network has emerged as interconnect paradigm [11]. The network on chip or the so-called NoC is capable of transporting (transmitting and receiving) the data between the different on chip blocks in the form of packets. Various on-chip IP cores attached to a network can communicate using the various network interfaces, channels connected through routers and links. An example of a SoC comprising of network on chip is shown in figure 1.13.

NoC offers an increased bandwidth and speedwhich is always a concern in a bus based system. Where a data movement through a bus by an IP core blocks its usage by other cores, switch based NoC offers a non-blocked data transaction which further increases the concurrency in communication. The introduction of pipelined links in NoC based design has led to an increase in its throughput and performance. Packet based data transactions allow sharing of the link resourced in a multiplexed way which enhances the resource utilization [48]. The quality of service in case of NoC based SoC design is guaranteed by implementing error detection at the packet and link level in contrast to the end to end error detection used for long interconnects in case of the bus based Interconnects. It is noteworthy that where long interconnects have their own penaltiesowing to introduction of delays, power consumption and possibility of introduction of errors; shorter switch to switch links in case of NoC are more error resistant.

Rerouting in case of fault occurrence on the data paths is possible in case of NoC based design while it leads to system failure in case of a bus-based design which makes it be a more reliable approach. The arbitration is done using a single /shared arbiter in case of a bus based design which turns to be slow and massive, leading to degradation of the speed. On the other hand, the arbitration in a NoC based design is done by using distributed arbiters which focus on local traffic information making them a faster approach. A global traffic condition needs to be checked to reduce the cost of whole system. NoC based design is modular and less complex as a switch /link design can be re-instantiated (being non-customized) as and when required thus lessening the design effort. The NoC based design is capable of more scalability

as the network bandwidth can be changed based on the network size whereas its counterpart (bus-based approach) becomes slow as the network size is increased.Usage of the globally asynchronous and locally synchronous (GALS) clocking strategy used in NoC [49-50] solves the synchronization problem by avoiding the need of using long clock interconnects spreading over whole chip which is prone to clock jitter. However, the overheads associated with additional switches/routers, packetizing, synchronizing etc. need to be taken care of while designing and using such structures.Driven by the benefits of improved performance, better quality of service, huge parallelism along with much reduced power dissipation on-chip network design has been a choice of research in the recent years.

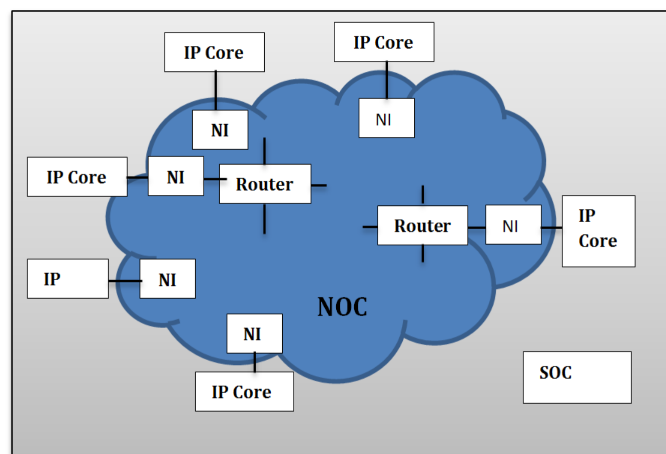


Figure 1.13: Conceptual view of SoC having an on chip network

As shown in the figure 1.13, the routers are responsible for appropriately routing the data packets between the sources and destination. Interfacing between the routers and associated cores is done through network Interface (NI) [51]. The data packets travel between the various network components like routers and NIs through the channels called links. Similar to the wireless networks, NoC can have different topologies in which the various routers can be connected. Based on the performance and other parameters one of the topologies like mesh, star, irregular application specific etc. can be chosen by the SoC developer. One of various network topologies can be opted based on the parameters like latency, throughput, and energy dissipation [52]. Figure1.14 shows the exemplary network topologies.

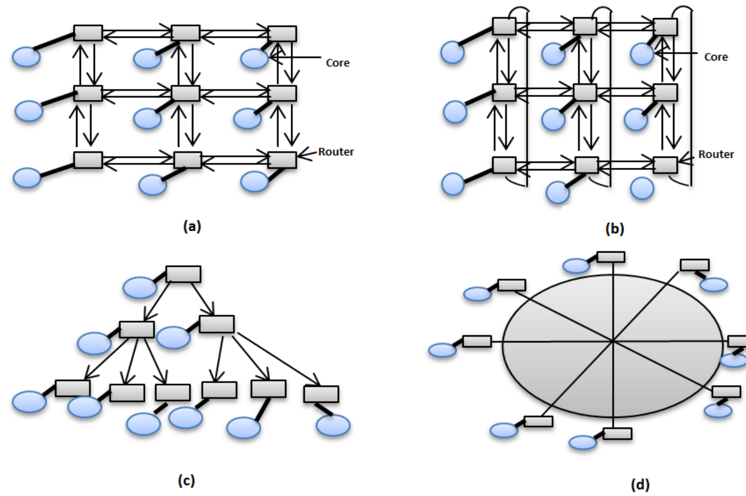


Figure 1.14 Regular and irregular network topologies a) Mesh b) Torus c) Binary Tree d) Octagon

The NoC approach also helps in reduction of the problems associated with clock skew and the power dissipation associated with the otherwise long interconnects running across the periphery of the system. SoC consists of a heterogeneous mixture of blocks like analog/ digital and memory blocks which operate at their different frequencies. To synchronize such blocks using the same clock can lead to performance issues caused by the problems like clock skew etc. Also, the power dissipation and the propagation delay associated with the wires lead to concerns of device malfunctioning. To solve such problems NOC offers an advantage of locally synchronous and globally asynchronous approach [49].

The message/information which is to be routed by the network is composed of three parts namely: header, payload, and trailer.Out of these, the header helps in establishing the path for the transmission. The messages are further segmented into data packets having the same format as that of the message. The data packets are further composed of subunits called flits. Flits are the basic storage and bandwidth allocation units that do not contain any routing information and just follow the route of the whole packet. Flits are further composed of Phits that are the actual physical data units transferred every clock cycle through the core network [51], [53]. The size of phit is determined in accordance with the bit width of a link. It is important to select an appropriate phit size as it affects the switch area which in turn determines the cost and performance of the network. A smaller phit size can reduce the number of switch fabric size and interconnect wires which in turn can reduce the area, power consumption and coupling capacitance associated with the wires. This is done by making the phit size smaller than the packet length and hence a serialization of the data needs to be done.

Figure 1.15 shows the NoC phit size variation that needs to be done in accordance with the variation in the frequencies. However, to maintain the same performance the operational frequency of the network needs to be increased by the scaling factor (SERR) between the packet size and the phit size.

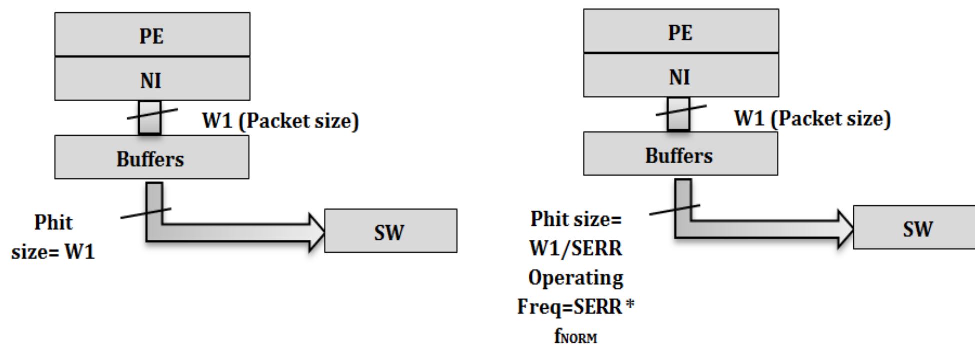


Figure 1.15 NoC phit size variance according to the variation in the frequencies

The routers/switches responsible for the data packet switching consist of input and output ports responsible for transmitting data between the other routers of the network and the network interface of the connected cores. The routers consist of routing logic blocks and ports for providing the input and output. Figure1.16 shows the conceptual view of the switch design consisting of input/output ports with their buffers.

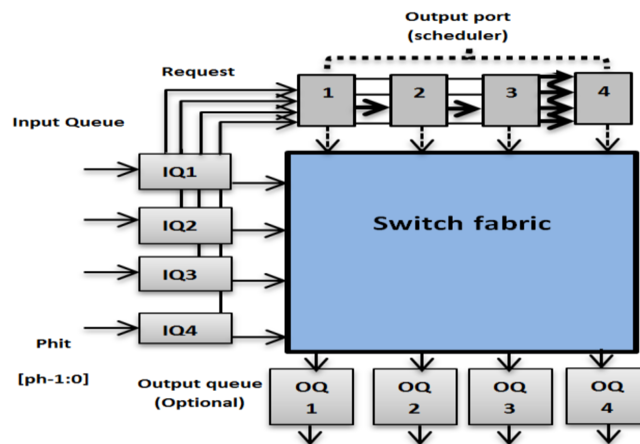


Figure 1.16 Conceptual view of the switch design for NoC

The scheduling of the packets through the output ports can be done using various scheduling algorithms like round robin, propagation basis and maximum weight matching etc.[54]. The switch fabrics can be designed using the cross bar or multiplexer based switches. Routing of the data can either be done using circuit switching or packet switching which decides which network switches need to set for data delivery. Out of the two types of switching: circuit

switching needs to reserve the entire physical path between the transmitter and receiver which though offers the benefit of one to one buffer less connectivity. However, such an approach suffers due to excessive latency induced by blockage which further affects the communication bandwidth.

Packet switching makes use of partitioning the message in fixed size packets which are transmitted without reserving the entire path. It offers the advantage of better bandwidth utilization but the arrival timings of the various packets are the function of the availability of various links and buffers being used for the communication. The contention problem associated with the various links can be overcome by negotiation between the eligible data packets. Buffers are further used for keeping a log of the packets which are left pending while solving the contention. Such buffers can either be placed at the input ports or output ports.

Packet switching can be further classified as wormhole switching, store and forward switching and virtual cut switching. The wormhole switching involves forwarding the packets consecutively ones their destination is identified based on the header information contained in the first arriving packet. Wormhole type packet switching doesn’t require buffering and hence offers minimal latency as only first packet needs to wait while rest are forwarded without any interruption and wait. This type of switching reserves the whole path. However, once the header packet is blocked, it would lead to blockage of entire data routed behind it leading to congestion and fall in performance of the network. Store and forward technique stores all the packets coming from the incoming link using large buffers which are further forwarded to the destination side. In Virtual Flow through technique, the path determination is done similar to wormhole. In case the next hop is preoccupied, the tail information contained in the packet is stored in a local buffer waiting for clearance of the path. Its buffer size can be smaller than the store and forward technique but, if the packet size is large then, many local buffers would be occupied leading to fall in the throughput.The advantages and disadvantages of each type are discussed in [52,55] and one of the above discussed techniques should be judiciously chosen.

The flow control mechanism of the network helps in the determination of the network resources like buffer capacity, channel bandwidth, and control used for the data packet [51]. Such a control can be buffered or bufferless. Out of these the bufferless flow control is mainly used for circuit switching and has issues like low latency and throughput. The buffered flow control, on the other hand, is preferred for the packet switching technique and

can further be classified as credit based, handshake based, ACK/ NASK, STALL/GO T-error flow control mechanisms.

Like any integrated system, NoC based SoCs are also prone to different types of manufacturing defects, therefore they also need to be tested. While many researchers have proventhe NoC as a reliable alternative for the bus based interconnects, it is being advocated as a promising solution for carrying the test data as well.

1.4 Thesis Organization

The organization of thesis is as follows:

Chapter 1: This chapter presents an overview of the System on chip design style and its advancements. It highlights the various test challenges and describes the modular test approach for SoC. It discusses the techniques that are being used for test cost reduction. It presents the motivation for the research undertaken along with an outline of the thesis organization.

Chapter 2: This chapter provides a review of the various test data compression techniques that may be used to encode the test data using lesser number of bits. Such techniques help in reducing the test application time along with a reduction in the amount of ATE memory and its reloads for storing the test data. Discussion of test data compression techniques is further followed by a review of the various test wrappers and test access mechanism (TAM) designs styles. The assignment of the various cores to the TAMs, their test scheduling and the adaptation of the 2D SoC modular test approach to fulfilling the needs of 3D SoC are also discussed. The discussion on the test infrastructure is followed up by a review on NoC testing and its use as a carrier of test data. A discussion of the various gaps that highlights the loopholes of the previous techniques is presented that led to conceiving of the idea of the proposed work. The chapter concludes with the description of the proposed objectives of the thesis work and a summary.

Chapter 3:presents the research methodology for the development of the efficient test solution for System on chip. It begins with a description of the proposed test data compression techniques. Five different test data compression techniques are presented namely: 10 Coded compression technique (10 C), selective count compatible run length encoding (SCCPRL), Hierarchical block merging based test data compression technique (HBMTC), Optimal selective count compatible run length encoding (OSCCPRL) and Adaptive block merging based test data compression technique (ABMTC). These techniques work on the reduction of

test data which in turn reduces the test application time and ATE memory. A description of the proposed integrated test solution for the 3D SoC is presented. A TSV conscious test wrapper design for fine grain partitioned 3D SoC is presented that optimizes the wrapper chains so as to reduce the test time of the cores. It is followed by the description of proposed test architecture for circuit partitioned 3D SoCs having 2D cores which may support pre, partial and post bond testing. The proposed work explores and solves the practically viable test cases which may occur based on the flexibility available to the test engineer. The 2D architectures are optimized to reduce the test time of the individual dies followed by an optimization of the 3D architecture. Being an integrated solution, it gives a variety of options to the test engineer to design the test infrastructure. The chapter is concluded by a brief summary of the proposed work.

Chapter 4: presents the simulation results for various test data compression techniques. The test data of different sequential ISCAS'89 benchmark circuits is used to compare the performances of the proposed algorithms with respect to the previously proposed test data compression techniques developed by other researchers. The parameters used to show the effectiveness of the proposed techniques include the compression efficiency, test application time, decoder area and test power. The chapter is concluded with a summary of the work done and its achievements.

Chapter 5: presents the simulation results obtained by using the proposed 3D test wrapper design algorithm and test architecture for 3D SoCs. Various ITC'02 benchmark circuits are utilized to reflect the usefulness of the proposed work. A brief summary concludes the chapter.

Chapter 6: Summarizes the contributions of this research work. It also throws light on some of the future works that can be undertaken to extend the work.

2. LITERATURE REVIEW

2.1 Introduction

Advancements in the semiconductor industry have led to impressive improvements in the level of achievable on chip density. In order to keep pace with the level of integration and fast delivery of products to the market, the design engineers have been embedding heterogeneous mix of IP cores on the same Silicon base [1]. The fabrication of such systems can be realized using two dimensional or three dimensional IC manufacturing techniques or may even incorporate an on-chip network as an alternative for on-chip interconnects.

The system integrator assembles the predesigned cores provided by the various core vendors along with other user-defined proprietary blocks to form a complete system [56]. Once such complex systems are designed, they are manufactured using the various semiconductor technologies. As the level of integration increases so do the complexities associated with their manufacturing at deep submicron level. Various fabrication processes are prone to an enormous amount of defects which can hamper system's performance [57]. Imperfections in the manufacturing processes like doping, alignment, thinning and stacking of different dies (in the case of 3D fabrication) at deep sub-micron level increases the likelihood of the various faults.

A SoC developer (or integrator) has to consider how to develop a complete test for cores provided by different vendors. Core information delivered by the various core vendors can vary in characteristics like information formats (e.g. soft or hard cores), implementation technologies, operating speeds etc. Since the core provider normally doesn't have the information about the system level integration so he only provides the core's design description along with the set of test patterns (with high fault coverage). The developer must also consider test application issues such as accessible test resources, allowable test time, tolerable test power and available automation tools. Therefore, it becomes the responsibility of the system integrator to decide and design the test architecture for testing the on-chip logic and wires between them. Ideally, test development for IP blocks should be carried out with reuse and system-level integration in mind.

To ensure the quality of the designs, the test data volume has increased by many folds which further increase the various recurring and non-recurring cost factors [24]. With an increase in complexity, controlling and observing independent on-chip components for test purpose using external pins has become very difficult. Consequently, it has become necessary to employ various Design for testability (DFT) approaches.

The overall aggregated test cost has three main components namely: cost related to the setting up of the Automatic test equipment (that includes test vector generation, storage and response analysis); cost associated with the on-chip test hardware (that may include the components like BIST which may employ LFSR [92] or an embedded processor [109], test data decompressors, test architecture etc.) and cost associated with the test application (in terms of time, power etc.).

The overall test cost can be reduced by the minimization of the test application time and power which majorly depend upon the amount of the test data [26], [58], concurrency of the execution of the tests of various cores and the on-chip test hardware [43], [59]. The section 2.2 reviews the various test data compression techniques which can reduce the needed amount of ATE memory and its reloads. The section also highlights how the appropriate test data encoding techniques can be used for reduction of the switching power dissipation at the boundary scan cells (to some extent). It is further followed by Section 2.3which describes the various test solutions approaches for both 2D and 3D SoC. Network on chip based SoC design and it's usage as a TAM mechanismare described in section 2.4. Section 2.5 presents the gaps in the work done by the various researchers which motivate for further research avenues in the area of SoC test optimization. Section 2.6 describes the objectives of the proposed work. Section 2.7 presents the summary.

2.2 Code based test data compression techniques

Various software based compression techniques have been developed in last three decades to minimize the amount of test data. Such approaches being easy to implement can be used to reduce the needed ATE memory and data transportation time. Various code based encoding techniques can be broadly categorized as Statistical codes and Run length codes. Following discussion reviews, some of such techniques starting with an overview of don't care filling techniques which can be used to reduce the test data and test power.

2.2.1 Don’t care filling techniques

Based on the test data analysis, it can be found that an average of 80-90 % of the completeset of test patterns associated with any circuit consists of don’t care (*X*)values. Such values (don’t care)can be suitably filled to compress the test data. Zero Filling technique [60] involves replacing all *don’t care* values with zeroes (*0*’s). The technique leads to an increase in the resultant run lengths of *0*’s. For example: Considering a random test data sequence ‘*00X0XXX0X0*’,which on being applied with zero filling, results in a run length equal to

'*0000000000*'. The resultant sequence can be translated as a pattern with a continuous run of ten zeroes. By increasing the run length of zeroes (same bit type), the subsequent switching activity likely to happen at the consecutively occurring boundary scan cells can be reduced to a large extent. Lesser the number of transitions, lesser will be the dynamic power dissipation. Apart from the reduction in the number of transitions, long run length of zeroes can be compressed using various code based compression techniques. However, the extent of increase in compression efficiency and reduction in test power is subject to the test data itself. For instance, if zero filling is done to fill don't care (*X*'s) bits sandwiched between two *1*'s then, itwouldrather lead to an increase in the resultant switching activity instead of reducing it. Also, more number of test bits will be used for encoding the subsequent pattern.
Random Fill(R-fill)[61] is another scheme used for appropriate filling of *X*'s bits in the test data. This scheme randomly replaces don't care bits with *1*'s or *0*'s. It may sometimes provide a benefit of detecting new (additional) faults as the updated test vector developed (after application of R-fill) can help in detection of new faults. Multiple faults detection using the same vector can lead to shrinkage of the number of faults that need to be tested (termed as fault dropping). Hence, it reduces the test data. However, R-fill may have an opposing effect on the test data and test power as the switching activity can increase due to non-consideration of correlation between the consecutive bits. An option for R-Fill is normally available in all the existing ATPG tools.
Authors in [62] have proposed a Minimum Transition Fill (MT Fill) approach that does *X*'s filling with a motive to reduce switching power. It fills don't care bits with the same status as that of the adjacent bit values. Such filling, when done for the *X's* sandwiched between two bits of same value, can lead to an increase in the run length of that bit value type. For example when don't care values at bit locations: second, third, seventh and eight positions are filled with '*1*', '*1*', '*0*' and '*0*' respectively in the test sequence '*1XX110XX0*' changes it to test sequence: '*111110000*'. As evident such a filling will reduce the amount of resultant switching activity at scan cells more as compared to above discussed Zero Fill and Random Fill techniques. For the special cases when the don't care bits are sandwiched between opposite bit values (i.e.*1* and *0* bits), then don't care bits can be replaced with either a *0* or *1*. For instance:'*1X0*' can be filled to get either *110* or *100*.
Column-wise bit stuffing (CBS) presented in[63] improves the compression by employing the benefits of both the *X*'s filling techniques and the difference vector calculation. The bit stuffing process starts by filling *X*'s in the first test vector sequence with zeroes. After preparing the initial vector, the *X*'s filling in successive vectors is done in correspondence

with the bit value at the same bit positions in the preceding vector. This is done to ensure that the difference between two successive vectors can be reduced by increasing the number of zeroes. Hence, by employing such a technique, the compression can be improved more as compared to the previously discussed don't care filling techniques. An ILP based don't care filling technique used to reduce the capture power is proposed in [70].

Block compatibility technique described in[64] is another technique for the compression of the test data. Entire test data is segmented in blocks of the same length (*l*). The patterns in consecutive blocks are investigated for the existence of compatibility among them. Two bits are said to be compatible if the bit values lying at the same positions (in both of the blocks) either carry same value or one of them is don't care. On the other hand, if the bits have opposite value then they are termed to be inverse of each other. Meanwhile, two blocks are termed to have compatibility/ inverse-compatibility among each other if the sum of existing compatible/ inverse-compatible and don't care cases in a block is equal to block length. If the compatibility/inverse compatibility exists between two or more blocks then such blocks can be merged to form a representative pattern block [64].

Block information can be preserved by an encoding scheme which retains the representative block followed bythe information aboutthe count of blocks being merged and the type of compatibility relation among them. For better understanding, consider the following examples: two blocks:*'10XX0111XX0'* and *'XXXX0111X00'* are compatible and can be merged to form a pattern *'10XX0111X00'*. For developing the compatibility, the initial two *X'*s of second pattern blocks are replaced with *'1'* & '*0*' respectively. Similarly, pattern blocks: *'1010101010'* and *'010101010X'* are inverse-compatible and can be merged to form a pattern *'1010101010'*. Sometimes the bit patterns in the various test data blocks have a unique combination which cannot be merged with other blocks. Such blocks are termed to be unique blocks or '*U'* cases.

To improve the compression efficiency even further, the retained/unique pattern can be further exercised for compression at the half block length level. For instance, the two half-length block (HLB) patterns can be compatible with respect to each other as described in [74]. For example: Filling *X*'s with *0*'s and *1*'s appropriately in a random block*'00X0011XXX*' results in a pattern*'0000011111'* which can be termed as a *'01*'case (an HLB consisting of *all zeroes* followed by another HLB having *all ones*). Similarly, the block: *'111X1000XX'* can be encoded as '*1111100000'*or *'10*'case (HLB with *all ones*followed by another HLB having *all zeroes*). As per the various permutations, the combinations possible at half block length level are: Unique HLB followed by another HLB consisting of *all*

zeroes(*U0*) case, Unique HLB followed by another HLB with *all ones* (*U1*), *all zeroes* followed by unique HLB (*0U)* or *all ones* followed by a unique HLB (*1U)*. For example, the pattern block: *'1011X00X00'* when fed with *0*'s in place of don't care bits can be quoted as a *U0* case.

2.2.2. Statistical encoding Techniques

The statistical codes [65] work on the principle of encoding the test data based on the frequency of occurrence of same test patterns. The encoding scheme begins with segmentation of the test data in equal length blocks. In case the complete length of the test data is not an exact multiple of the block length then extra/redundant *X'*s bits can be appended in its start so as to make it so. Application of such extra bits (retrieved after off-chip decompression) onto the scan cells doesn't hamper the test process as long as the actual test vector content is kept intact. If the block size is '*l'* bits then all the exhaustive combination (2^l) of such length are taken into consideration and analyzed for knowing their frequency of occurrence.Thereafter,the blocks are encoded with different codewords based on their frequency of occurrence.

The principle is to encode the most frequently occurring blocks with less number of bits as compared to least frequently occurring ones. This results in a fixed block size to variable length codewords based encoding technique. Correspondingly, the fixed dictionary of codes is generated which may help to encode the original test stream with less number of bits. The efficiency of a statistical code in compressing the amount of test data depends upon the extent of variation in the frequency of occurrence of same pattern blocks. More the number of times the various combinations occur more will be their chances to get replaced with a shorter length codeword. A statistical code based scheme named Huffman code presented in[65] compresses the test data using Huffman tree based uniquely definable code words. Corresponding to each distinct pattern block, a codeword is generated by following the Huffman tree's branches. Bit values associated with the root down to the leaves for various branches designed as per the different weights of occurrence of each distinct block are used to extract the code words. The efficiency of the Huffman encoding is increased further by making use of various don't care filling approaches so as to increase the probability of occurrence of some of the blocks. For a test data of length $T_D$when segmented into blocks of length l bits each, it results in constitution of n blocks (where $n= T_D/ l$). For the length l, 2^ldifferent combinations of distant blocks are formed. Considering a random sequence of 120 bits when segmented at a length of 4 bits, results in blocks (as shown to be separated by an

extra space) as follows: '*1010 1010 1010 1110 1010 001X XX00 00XX X111 1XX0 101X X010 1010 XX00 1010 1101 0001 0001 0100 0100 1010 1010 0001 1100 0111 0001 1000 1110 1010 1010*'. As evident the frequency of occurrence of various distinct blocks is follows: block *1010*occurred ten times (that includes two blocks in which *X*'s bits are appropriately filled to match *1010*); *1110* occurring thrice (including *1XX0*wherein *X*'s are filled with '*11'* to make it match *1110*) ; *0011* occurring twice(wherein *X*'s in *001X* and *00XX*are filled with '*1*' to match *0011)*; *1100* occurring thrice as in *XX00*, *XX00*,*1100*;*0100* occurring twice; *0111* occurring once; *0001* occurring thrice; 1000 occurring once and so on. The advantage of the Huffman code being free from prefix simplifies its decoding process as it nullifies the need of any look ahead. The compression efficiency of Huffman code relies on the correlation among the test data blocks. More, the correlation more would be the chances of having a higher frequency of occurrence of some blocks in comparison to others. Whereas, if all the blocks have nearly same occurrence frequency then the extent of compression can suffer as not much of variability can be maintained among the length of distinct code words. The disadvantage of the Huffman code lies in the overhead and complexity associated with its decoder design.

The problem of the large area overhead associated with the decoder design is resolved by the selective Huffman coding proposed in [66]. This technique works on the idea of shortening up the size of the tree by encoding only the most frequently occurring codewords while leaving the rest intact and hence unencoded. The discrimination between the unencoded and encoded blocks is done by using a single bit which precedes the codeword. The status of the prefix bit is set to '*0*' if the distinct block is left unencoded and preserved as such whereas, it is set to '*1*' if some codeword is used to replace the block. The decoder design is quite simplified in the case of selective Huffman technique. However, it suffers in terms of achievable average compression.

The overhead of extra bit that precedes both coded and unencoded blocks can be quite taxing in terms of codewords lengths of most frequently occurring blocks. This loophole is resolved by using Optimal Selective Huffman Encoding technique[67] which works on the addition of prefix bits only to unencoded blocks leaving shorter code word lengths for most frequently occurring blocks. The overhead is of a single bit which just needs to be added to the unencoded blocks. Such a change leads to huge improvement in compressing most frequently occurring blocks which also compensate for the expense of extra bits in the case of unencoded blocks (which are normally less in number). It leads to the number of resultant codewords to be one more than the number of frequently occurring blocks i.e. one (for all

less/unique frequently occurring blocks) and *n* (for all frequently occurring blocks). Associated decoder area is also much reduced in comparison to Huffman and selective Huffman coding techniques.

Mixed R L- Huffman[68-69] coding technique amalgamates the benefits of the run length based encoding technique and the statistical Huffman encoding technique. The encoding is done by following a number of steps: Initially, run lengths of *zeroes* and *ones* are increased by using don't care filling techniques. Such a step also minimizes the amount of switching activity likely to occur at the scan cells of the circuits/cores to be tested. The second step involves encoding the resultant test data by using the Huffman encoding technique. The principle of encoding the most frequently occurring block values can be efficiently utilized by increasing the occurrence of specific block combinations by using run-length encoding techniques. Later, the encoding is done using the Huffman tree as explained earlier.

2.2.3 Run length encoding techniques

Run length based techniques utilize the various test data properties (as described in section 2.1) to compress the test data. As discussed above, don't care filling techniques help in increasing the run lengths of various bits patterns and increasing the compatibility among the various data blocks. Run length techniques further encode the test data using lesser number of bits. Such approaches find wide applicability due to their use of a simple algorithm and less complex FSM based decompressor design. A brief description of various run length encoding techniques is as follows:

Golomb Code based technique described in [25] works on encoding the runs of zeroes with different codewords such that the longer runs of zeroes can be encoded using lesser number of bits. This is achieved by executing a two-step process. The first step involves the calculation of difference vectors such that the run lengths of *zeroes* can be increased while the second step works on the selection of appropriate Golomb code parameter, referred to as the group size which is further used for subsequent code word formation. The occurrence of runs of *zeroes* in the test data is dependent upon the design of the circuit under test. Therefore, the selection of the group size is done experimentally.Once the group size is selected, the runs of *zeroes* in calculated difference vector are mapped to various sets of different group sizes to extract the resultant codeword. For the sake of the clarity, it may be recalled that the difference vector is computed by doing the EX-ORing of the consecutive vectors which increase the runs of *zeroes*. The amount of the longest runs of *zeroes* in the difference vector is used to determine the count of the groups needed to do the encoding.

Therefore, encoding of a test sequence is done according to the lengths of *zeroes*. For instance: the random test sequence '*00011001*' is viewed as having run of 3 zeroes (*0001*:3 zeroes ending with a *one*) followed by a *1*, *0* runs of zeroes followed by a one (as there lies a consecutive run of two ones without any zero in between) which are further followed by a run of two *zeroes* (*001*). The codewords corresponding to different run lengths are used to encode such a test pattern.

Authors in [71] proposed a frequency directed run length (FDR) encoding technique which categorizes group number (used to define the prefix of the codeword) based on the frequency of occurrence of the run lengths of *zeroes*. To improve the runs of *0*'s, the difference vector calculation is done similar to the Golomb coding (described above). FDR code utilizes the statistics of the frequency of occurrence of runs of *zeroes* to improve the compression efficiency even further. As per the experimental results (reported in the paper), the occurrence frequency of successive runs of zeroes is more for the lengths ranging up to 20 beyond which it degrades gradually. Different lengths of the prefixes are used to develop the code word which makes the scheme to be a variable to variable length encoding technique.

Another technique called Extended Frequency directed run length encoding technique (EFDR) proposed in [72] increases the compression efficiency beyond what can be achieved using FDR code. It extends the frequency led encoding technique of FDR to encode the lengths of *one*s too. Herein, the lengths of *zeroes* followed by a *1* and lengths of *ones* followed by a *0* are calculated and thereafter encoded. The encoding code-words so developed are exactly similar to that of FDR with an only difference of one extra bit which helps in distinguishing between the encoding being done for the runs of *zeroes* or *ones*. This extra bit is used by the on-chip decoder logic to retrieve the original stream of data.

The alternative run length encoding technique [73] gives a new way of looking at the run lengths of *0*'s or *1*'s. Instead of terming the whole concept as a finite number of *zeroes or ones* followed by their opposite bit values, it encodes the data on the basis of their alternative nature of existence. For example a random test sequence '*11000110001101*' is encoded as having the following run lengths in chronological order: 2 *ones*_3 *zeroes* _ 2 *ones* _ 3 *zeroes* _2 *ones*_1 *zeroes* _*1 ones*. AR coding removes the overhead of extra bit (used to distinguish between the run length of *0* or *1*) by considering the alternative occurrence of *0*'s and *1*'s as the thumb rule for encoding.

Nine Coded (9C) compression technique[74] added a new dimension to the test data compression techniques by encoding the fixed length test data blocks. Whole test data is segmented in blocks of the same size. Nine codewords are used to do the encoding based on

the existence of the relation between the HLBs formed by partitioning the *l* bit blocks into two halves of length *l*/2 each. On analyzing the HLBs it may be found that they may have one of following nine combinations: both HLB having *all zeroes* (case *00)* or *all ones* (case *11*), First HLB having *all zeroes* followed by an *all ones* HLB (case *01*), an *all ones* HLB followed by *all zeroes* HLB (case *10*), a *unique* HLB followed by an *all zeroes* HLB (case *U0*) , an *unique* HLB followed by an *all ones* HLB (case *U1*), *all zeroes* HLB followed by an *unique* HLB (case *0U*) , an *all ones* HLB followed by *a unique* HLB (case *1U*) or an *unique* HLB followed by another *unique* HLB (case *UV*).

Corresponding to each of the 9 combinations (stated above), nine different codewords are used for encoding. The compression gain was further shown to be extended in the same paper by using a variable length 9C code [74]. The technique dynamically chooses the blocks lengths so as to increase the test data compression. The difference between the various run length based encoding techniques and 9C is the use of fixed block length and utilization of 9 special cases.

Fixed Length Pattern Run Length (PRL) Coding[75] employs the block merging technique to compress the test data. The entire test data is divided into equal length blocks and investigated for the occurrence of compatibility/ inverse compatibility. The blocks are merged together to retain a representative pattern block. For each block being merged, a '*00*' or '*01*' is added after the retained pattern to represent the existence of compatibility/ inverse compatibility with respect to the retained pattern. The entire codeword is further followed by a '*0*' to represent the end of compatibility match. The search of compatibility check and its corresponding encoding using appropriate codeword is done for the entire test data.

Count Compatible Pattern Run Length (CCPRL) Encoding Technique proposed in [76] works on improvement of the fixed pattern run length encoding by replacing the repetitive inclusion of 2-bit compatibility codes ('*00*' and '*01*') by count code and compatibility code. The codeword so formed consists of *block code* (binary conversion of the length of the pattern block that needs to be sent only once during the start of the encoding), *pattern code* (merged/retained block sequence), *count code* (a binary count to represent number of times matches are found between the retained pattern and its adjacent blocks), *compatible code* (consisting of relation bits that signify the compatible/inverse compatible relation between the retained and successive blocks). It uses only one bit to represent the compatibility relation instead of two (as used for PRL). Also, the last zero (*0*) bit used to represent the end of a particular count is not required since the count code itself takes care of it.

Block Merging Technique (BM) for Test Data Compression[64]also exploits the correlation among the test data patterns to allow the block level merging. This type of encoding technique consists of a merge block along with a count to indicate the number of the blocks that can be merged. It dominantly categorizes the encoding of the merged blocks on the basis of *all zeroes, all ones* or *unique* combinations. The merged blocks are grouped into different categories based on the count of the blocks being merged. Based on the frequency the number of blocks being merged, variable length code words are generated. Another block merging technique is proposed in [77].

Another variant of the block merging is described in Block Merging with Eight Coding technique (BM-8C) [78]. In this technique, inter-block merging is done on the basis of the existence of compatibility to retain a representative block. The retained pattern block is further encoded on the basis of eight different subcases to enhance the compression efficiency. Eight different cases are used for intra block level compression: HLB level compatibility, inverse compatibility, U0, U1, 0U, 1U and all unique combinations. BM-8C technique decreases the test data volume which in turn reduces the test application time of the circuit under test.

A low power selective pattern compression (LPSC) technique described in[79] attempts to resolve the issue of the test cost reduction by minimizing both the test time and test power. To achieve it, this technique partitions the entire test data into two groups namely test data compression (TDC) and Average Power Reduction (APR) on the basis of the occurrence of don't care values. The TDC group carries the test vectors with a higher number of don't care values. Such test data is compressed by employing a statistical code based technique. Don't care filling techniques are used in the APR group to reduce the switching power at the input /output scan cells. A scheme to reduce the capture power (defined as the difference between the test stimuli and its responses of the same scan cells) is also proposed in the same paper. LPSC scheme is reported to achieve test compression efficiency comparable to many other reported techniques along with a much-reduced average and peak power dissipation.

2.3. Test architecture development

Once the test data gets decompressed by using the on-chip decompressors, it needs to be delivered to their associated test sites via on-chip test infrastructure. Distribution of all wires dedicated for carrying the test data, their allotment to various SoC blocks and scheduling of the various cores tests form the backbone of the efficient test solution development [80-82].

Testing of the whole system is done by employing the "divide and rule" technique wherein various on-chip components are isolated from the surrounding environment and fed with the data bits for test purpose. This isolation is achieved by wrapping up the cores and User Defined Logic (UDL) in test wrappers. Test wrappers help in de-serialization and application of the test bits supplied by the TAM wires at the appropriate terminals of the core under test. The successful delivery of the test vectors (specific to the core under test) is controlled and scheduled by the on-chip test controller using the appropriate design and distribution of test access wires.

The objective of an effective test solution development for a system on chip is to minimize the test cost by reduction of the test application time such that various constraints like power, test bandwidth etc, are not violated. Such an optimization problem can be addressed by sequentially addressing the problems of minimization of test time of individual cores, appropriate distribution of the TAM wires to the various cores and bringing parallelization in scheduling the test of the core as much as possible.

The reduction of the test time of individual cores can be achieved by optimization of the test wrapper design of the cores. The time related to the test vector delivery to the various cores can be reduced by optimally designing the test architecture. Test architecture depends on the appropriate assignment of the cores to the individual test access terminals and scheduling the various cores' tests. A discussion on the various trade-offs that effect the decision of the overall test optimization has been presented in chapter1. Following discussion reviews the various approaches that have been developed so far for the achievement of efficient test solution for SoC.

2.3.1 Test Wrapper Design and optimization

A test wrapper facilitates modular testing by isolating core under test from the external environment. Test wrapper aims at the test vector application to the various cores components (functional input/ outputs and internal scan chains) using the serial or parallel interface provided by the TAM wires allotted to it. Its design revolves around forming the wrapper chains (WCs) by stitching the various core components so as to match the available TAM [31]. It helps in ensuring the testability of the core at the expense of more area and performance overhead.

Structured tests re-use methodology and infrastructure for core based system chips is presented in [32]. It makes use of 4-5 test wires to supply the test clock, test data and test instructions. The need of a standardization to test the cores of an IC as felt by both academia and industry led to the introduction of the IEEE P1500 standard [36], [83]. An approach combining P1500 and the Test Shell approaches has been proposed by authors in [31]. A major advantage is a flexibility introduced by the bypass technique. Two types of bypass styles are defined: wrapper bypass and scan-chain bypass. The wrapper bypass is similar to the one proposed in Test Shell approach while the scan-chain bypass is a flexible structure which can be inserted at any place between a terminal input and a terminal output. It allows a non-clocked bypass structure which can be used to bypass the complete core.

To facilitate the testing of the cores internal and external components, a test wrapper can support four modes of operationsnamely: INTEST mode (cores internal test mode), EXTEST mode (cores external/interconnect test mode), BYPASS mode (which bypasses the core not be tested at that moment) and NORMAL mode (which means non-test or normal functional mode of the core) [84-85].

As previously mentioned, the various wrapper design styles only specify the structural design meant for interfacing the core to the surrounding test infrastructure [31]. The wrapper's internal design majorly revolves around the construction and balancing of the wrapper chains such that the testing time of the individual core can be minimized. Many wrapper design algorithms have been proposed in previous two decades such that the lengths of the scan-in and scan-out paths of the chains can be reduced. The wrapper design algorithm performs two functions: construction of the WC equal in number as that of the available TAM wires and balancing the lengths of the WCs. If the difference between the maximum values of the scan-in and scan-out lengths is minimized so would be the testing time of the core. Therefore, the wrapper design algorithm plays a major role in affecting the testing time of an individual core.

Authors in [31] proposed heuristic based algorithms to address the wrapper optimization problem. Out of the proposed techniques: first technique termed as Largest Processing Time (LPT) heuristic, adapts the Multi-Processor Scheduling problem to balance the wrapper chains for cores test time optimization. For cores having a small number of components, a solution is reported to be obtained in a very short computational time. The paper proposes one more heuristic based solution named Combine which integrates the Largest processing

time algorithm with another linear search based first fit decreasing algorithm to obtain much better results.

A design wrapper algorithm based on the Best Fit Decreasing heuristic for Bin Packing problem is proposed in [86]. The algorithm proposes a heuristic to reduce the number of the wrapper chains along with their optimization such that testing time of the core can be reduced. The authors use the experimental results to suggest that the testing time of the cores varies as a "staircase" function with respect to the number of TAM wires allocated to them. In other words, the test application time reduces only in steps when the number of TAM wires is increased. Such discrete steps sizes termed as Pareto optimal points/widths help in determining an important conclusion that even if the number of the TAM wires are increased, no reduction in the test application time can be achieved till their number doesn't become equal to the next Pareto optimal point. And hence, such an approach can have a huge impact on the distribution of the available TAM width among the various cores of the system on chip.

Authors in [87] proposed a core wrapper that solves the testability problems raised by embedded cores having different clock domains. The paper proposes a heuristic based test wrapper optimization algorithm that considers the conflicts set by wrapper resources like channels associated with the tester, area, power and test time. A reconfigurable core wrapper design to extend the flexibility of TAM width allotment to the core by dynamically changing it using some control blocks is proposed in [88-89]. The author proposed that varying the TAM width in accordance with the amount of the test data being fed to the core can help in reduction of the test application time. Consequently, reconfigurable wrapper chains can be formed by the use of some extra multiplexers. An automatic procedure based on the graph representation of the core wrappers is also proposed to arrive at the decision of the needed DFT hardware. A reconfigurable power-conscious core wrapper design that works by multiplexing the inputs of the scan chain is proposed by Larsson and Peng in [90]. The proposed technique amalgamates the reconfigurable wrapper and scan chain clock gating concepts [91]. Reconfigurable core wrapper design proves to be more beneficial for cores having specific requirements related to the generation of the test data. For example: if a core needs to be fed with multiple test sets generated by both BIST and ATPG then, such an approach offers better optimization of test time by varying the TAM assignment dynamically.

All the wrapper design algorithms discussed above considered the cores to be flat with all core components lying at the same level of design. However, in practical cases, the cores embedded in the designs can be hierarchical and hence have other smaller cores as the part of

them. In such a case the mega core itself becomes the parent core with its underlying cores behaving as its child cores [37]. For the effective optimization of testing time of such mega cores, the wrapper design needs to be done in a hierarchy aware way. The testing of the parent cores need the wrapper chains to include the child's input/output wrapper cells. Similarly, the TAM design inside the mega cores needs to cater to the child cores as well. Authors in[37-38] and [93-94] have proposed wrapper design algorithms for hierarchical SoCs. These perform the wrapper cells design by accepting the available TAM count and optimization of the internal wrapper chains for both parent and child. They identified and worked on the INTEST and EXTEST operational modes of both the parent and child cores. The proposed solution allowstesting of the parent core and its child core(s) to be done in parallel at the expense of little expensive wrapper design style.

With the expansion in the vertical dimension, 3D SoCs [6-8] are gaining considerable attention by the designers. Such a kind of design structures introduced new design and test challenges [17-18]. To ensure the testing, modular test approach as applicable for the 2D SoCs need to be modified to cover the various constraints specific to 3D structures. The test planning approach so formed should ensure to take care of the constraints set by the number of available Through Silicon Vias (TSV) for test purpose, allowable test power dissipation, core placement etc. Based on the fabrication style of the 3D structures, they can be categorized as fine grain partitioned SoCs and Coarse grain partitioned SoCs [9]. The fine-grained SoCs refer to the structures wherein the cores are 3D and spread over multiple layers. Such kind of structures utilizes multiple TSVs for supporting interconnects specific to the Cores themselves. Therefore, as the number of such interconnects increases in correspondence with the cores on the SoC so would be the number of needed TSVs. Corresponding to the different core formation, the wrapper designs have to be modified in fine grain partitioned SoC. 3D Core test wrapper designs for circuit/ fine grain partitioned SoCs have been proposed in [95-97]. The algorithms optimize the testing time of the core by balancing the wrapper chains while addressing the constraints set by available TSV and TAM bandwidth. Once the wrappers have been properly designed, any 2D test assignment and schedule can be followed to reduce the test time. One thing that needs to be taken care of is that: all the core terminals start and end at ground die which means all the test wires that go up have to come down which affect the number of the TSVs used.

2.3.2 Test Access Architecture Optimization

Increase in the order of integration has led to many challenges for the test engineers. Some of the difficulties include accessibility of the individual cores which are embedded deep inside the SOC; huge difference between the number of functional input/output terminals associated with the various cores and the number of pins available at the SOC periphery. Limitations of available interconnect bandwidth for carrying the test data worsen the problem of the transportation and application of test data to the various cores. To ensure the test data delivery and accessibility of all test sites, various DFT mechanisms have been explored so far. The simplest approach can be to have as many IC terminals as there are core terminals. Such an approach can make each core to be directly accessible [98]; however, the area overhead associated with the routing of so many wires makes it infeasible. Another approach that can be used for the test access is the use of isolation rings as per the IEEE 1149.1 DFT technique [99]. It is widely used approach for board and system level testing. It employs a serial scan chain that spans around the boundary of every core. Multiplexers are utilized at every core terminal to allow an access using the boundary scan chain. Whetsel [100]utilized the 1149.1 test access ports for core level test access architecture formation. Such an approach offers advantages like low wiring and area overhead, accessibility to various cores terminals using lesser number of input/output pins of the chip, isolation of the core under test from its surrounding logic, testability of the various interconnects etc. However, use of such a technique is demotivated by the long test application time associated with the serial shifting of the test data through serial scan chain and the delays introduced by the multiplexers at each core terminal.

Another approach based on the partial isolation rings which avoids the additional logics (like multiplexers) on the critical path without affecting the fault coverage was presented by Touba and Pouya[101]. The technique was further modified by the same authors wherein they described the use of UDL to supply the test vectors and hence eliminated the need for the isolation rings. However, the advantage of reduction of the extra hardware and delay posed by the multiplexers led to the increase in the computational complexity. Another approach based on reusing the existing functional paths for test purpose is proposed in [102].The paper describes the test strategy used by ARM that employs the 32-bit AMBA functional bus to carry the test vectors. A macro test based ad-hoc approaches for ensuring testability is described in [103-104]. A technique based on core transparency that employs finding and using the functional paths between the input/output terminals for propagating the test data/

control information is proposed in [105-106]. Such a technique can reduce the area and delay overhead but, the need of knowledge of the functional description of the core limits its use since it is normally not available. Another limitation of such an approach is the difficulty in its adaptability to support other DFT schemes such as scan or BIST. An approach based on making the cores single cycle transparent is proposed in [107].An ILP based TAM optimization technique proposed in [108] make the soft cores transparent on the consecutive basis by employing minimal hardware. Aerts and Marinissen [39] proposed three different bus based TAM architecture that employs multiplexing, daisy-chaining and distribution of the available TAM width. A combination of the multiplexed and distributed TAM approaches is the test bus approach [33]. Two variants of the test bus approach[40] are the fixed width test bus and flexible width test bus. A test rail architecture that combines the test bus and daisy chain architectures is presented in [32].

Once the TAM architecture and assignment of the cores have been done, next step is to decide the test schedules of the various cores. In fact, the test scheduling decides the start and end of the tests of the various cores. It has been observed that the TAM design, assignment of cores to TAM and their test schedules are very interdependent problems. Many heuristic based optimized test scheduling techniques have been developed in past two to three decades based on re-usage of optimization algorithms applicable in other fields of engineering. The test schedules can be divided into three categories namely: session based testing, sessionless testing with a run to completion and the preemptive testing[41-42], [44].

K. Chakrabarty formulated the SoC test problem as an m-processor open shop scheduling problem in[42]. The approach assumed the test architecture to be of test bus type and solved the scheduling problem using mixed integer linear programming technique. Authors in [110] formulated the SoC test schedule as a combinatorial problem and proposed a solution wherein a selection and schedule of the test set for every core (out of the multiple test sets provided by the core vendor) is done by utilizing a heuristic approach. Various fixed and flexible width test bus architectures based test scheduling techniques have been proposed in the literature [80]. A brief description highlighting their merits and demerits are as given below.

Authors in [111] propose a test solution that optimizes the test wrapper and TAM design problems together. It works in an incremental way by optimizing the test wrapper to reduce the test time, partitioning of the available TAM bandwidth to obtain test buses, assignment of

cores to the so formed buses and then finally scheduling the various tests. The only drawback of such a solution is the long computation time which increases extensively with an increase in the number of cores or the TAM partitions. Same authorstried to reduce the test time by combining the basic ILP solution with other efficient algorithms in [70]. It included the precedence, pre-emption and power constraints. The paper also investigated the relation between test data volume and TAM.

The concept of flexible width TAM architecture design has been supported by authors in [40] and it claimed to offer the allotment of TAM wires to the various cores as per their Pareto optimal width. The scheduling of the cores assigned to the different TAM wires can be decided in the initial phase of test architecture development. However, on the flip side, such an approach majorly depends on the effectiveness of the optimization algorithm being used for the TAM design. For instance, if the number of the cores is more, then the search to find the best possible TAM partition can be quite a complex problem (being an NP-hard problem). Also, the flexible design necessitates a major number of test wires to be used as control pins which leave fewer wires for the test data transport.

Koranne proposed a polynomial time solution that optimizes the average completion time of all the TAM partitions [112-113]. The problem of SoC test planning is reduced to the minimum weight perfect bipartite graph matching problem. A TPLAN test planning tool so developed models the problem in the form of a graph with test source/ sink as the source/sink in the network, cores as nodes and the TAMs as the channels (with their capacity being equivalent to their width). It aims at reducing the average test completion time of all the TAMs, however, no conflicts like precedence, power etc. are considered.

Goel and Marininssenproposed a test plan based on the use of Test Rail architecture with fixed width test access mechanism in [114]. In the solution, the authors mandated the total Test bandwidth to be more than the number of cores present in the SoC. This limitation was removed by the same authors in their next work[82] and [115]wherein they used a heuristic based TR Architect solution that provided a general solution for the Test architecture design. Being versatile, it caters to all cores having fixed or flexible scan chains and supported both test bus and test Rail TAM architecture. In the same paper, authors have presented a lower bound of the SoC test application time based on the number of test patterns, various functional input/ outputs cells and scan chains/ flip flops in the SOC. This lower limit has worked as a benchmark for the later research as all the TAM optimization techniques try to bring the test time as close as possible to it. Authors described three types of idle bits which

lead to increase in the test application time. These are categorized as extra bits which get added due to: imbalanced TAM completion times; non-optimal width assignment to the various cores and imbalanced scan chains inside the cores. In order to avoid such bits and reach an efficient test schedule, the TR Architect solution works in four steps. An initial solution is created that assigns one TAM wire to each core. Minimization of the total test application time is done by performing redistribution of the test wires. It involves merging noncritical TAMs to free some TAM wires which can be added to the critical TAMs to reduce its total test time. Merging of critical TAM with other TAMs is also done such that the cumulative test time is reduced. Finally, a reshuffling of the cores is done to reduce the test time even more. As a result, some of the cores from the critical TAMs can be reassigned to other non-critical cores. Same authors extended the TR Architect algorithm to minimize the wire length which reduces the wiring area overhead in combination with a reduction of test application time.

Various other researchers adapted the well-known optimization problems like genetic algorithm [116], simulated annealing [117], Ant Colony Optimization [118] etc. to solve the test scheduling problem assuming fixed test bus architecture. Authors in [117] modified the sequence pair based solution for rectangle placement problem of place and route to arrive at optimized test schedule. Authors of [117] propose a simulated annealing optimization technique which alters the initial sequence pair and rectangle transformation for scheduling the tests of various cores. Though the approach provides a reduction of test time but it lacks to address other constraints like power etc. In [119], authors havemapped the SoC test scheduling problem as a well-known 2D bin packing problem to test the cores concurrently. The solution models each core as a rectangle with its width and height representing the TAM bandwidth allotted to the core and its test application time respectively. The best-fit algorithm is used to pack the rectangles into a bin of fixed width (equal to the available TAM) such that the height (corresponding to the overall total test application time of the SOC) can be minimized. Meanwhile, the solution takes the test power as a constraint which should not be violated throughout the test schedule. Another approach by the same authors incorporating the multiple test sets is presented in [120].

In [121], the authors extend the concept of Pareto optimal TAM width selection for test time reduction to allow precedence and power constraint. Tests were allowed to be preempted to best optimize the test time. The tester data volume and its relationship with the TAM width are chosen as the figure of merit to decide the TAM architecture plan. In [121], the same

authors further proposed to reduce the test application time by reducing the ATE memory reloads and supporting multisite SoC testing. Rectangle bin packing algorithm was further used to push the idle times to the end of test time of each TAM partition so that unnecessary (idle bits) need not be stored on the ATE. Such an approach reduces the overall testing time of a single chip. A 3D cube based formulation of the SoC test problem is proposed in [122]. The cube has peak test power, total test bandwidth and test time as the three dimensions. A heuristic-based solution is proposed that addresses the integrated test problem comprising of test wrapper design, TAM architecture and test schedule optimization under the constraint of test power and TAM width. Koranne[113] proposed a test schedule solution by combining the network flow and malleable job scheduling algorithms. The approach is reported to provide the solution in polynomial time. The proposed algorithm employs the reconfigurable test wrappers to minimize the testing time of the individual cores.

An integrated framework for constraint (test resources and test power) driven test schedule and TAM design is proposed in [27]. A simulated annealing based algorithm was proposed[124] which reduced the CPU run time for SoCs having many cores. Another test scheduling approach to reduce the test power in a system on chip is proposed in [123]. Authors in [125] proposed a graph-theoretic formulation based solution for the SoC test schedule. This technique considers the power constraint in the initial solution formulation and prepares sets of tests that are compatible in terms of power, test time and resources. Dynamic TAM partitioning and assignment is done to reduce the testing time as much as possible without introducing idle times. Another graph-based test solution based on greedy TAM assignment approach is proposed in [126]. It initially prepares a Test Compatibility graph (TCG) for the entire SoC, then it finds a subset of the graph that satisfies the various constraints and reduces the overall test application time. A Tabu search based approach is used to reduce the computation time.

Area and performance overheads introduced by the wrapper boundary cells are reduced in the solution proposed by Xu ad Nicolici in [127]. The authors present a heuristic-based solution to the rectangle packing based SoC test scheduling problem. Though the technique reduces the overall DFT area but the sharing of the infrastructure introduces more resource conflicts which leads to an increase of the testing time.

SoCs comprising of hierarchical cores have different test needs for both parent and child cores which necessitate the Test architecture to be modified accordingly. In [128] authors have presented a multilevel TAM optimization technique for SoCs. It incorporates TAM

width adaptation using a combination of integer linear programming (ILP) and enumeration for both the interactive and non-interactive types of cores provided by different vendors with a perspective to reduce test time. In [38], authors have proposed a test schedule and TAM architecture for the hierarchical SoC which allows the testing of both parent and child to be done concurrently.

The most recent advancements in the 3D technology necessitated the consideration of a number of through silicon vias as another constraint for TAM design. The cores are assumed to be 2D with their placement on different dies. TAM design problem of 3D SoCs has been addressed using ILP formulation in [28]. The solution considers the test time and test power as the optimization objectives with a number of available TSVs as a constraint. The solution attempts to minimize the total test time of complete 3D stack, however; no considerations are given to the pre-bond and mid bond level die testing. A simulated annealing based TAM optimization approach is proposed in [129]. Based on a user-defined cost factor it reduces the test time along with the TAM wire length. The paper assumes the TAM wires to start and end at any die which makes it practically unrealistic. Another paper [130] tries to partly solve the above problem by allowing pre-bond die testing through dedicated probe pads (which are considered to be limited in number). A heuristic based technique is developed to provide a pre and post bond 3D SoC test such that the test time and test wire length are minimized. An approach proposed in [131] allows the pre-bond and post-bond testing in addition to TSV based interconnect testing. The technique provides a modular test for 3D SoC structures by reusing the preexisting test wrappers of the various cores. It reduces the area overhead owing to reuse of test infrastructure of individual dies. The paper, however, ignores the circuitry in between the wrapped cores which make it be practically unrealistic.

Authors have emphasized on the need and approach to test the 3D SoC at pre, partial and post bond level in [132]. The approach proposed in [47] utilizes the pre-existing DFT infrastructure available at the core, die, and product level. It proposed the concept of die level wrapper based on either P1500 or IEEE 1149.1 standard. In [45], the authors have suggested that the pre-bond test access can be done by direct probing using fine pitch micro-bumps or optional extra pads at the die under test. In order to ensure a fault-free delivery of the 3D SoC structures, the test developer needs to do multiple test insertions of various dies so as to facilitate the pre-bond, partial bond, and full stacked (post bond) testing. Herein, the testing which is performed on the wafers before thinning is termed as the pre-bond test. On stacking the dies, to ensure the quality of the intermediate interconnects and the partially stacked dies,

the pre-bond tested dies are retested together with the connected dies at the partial stacking level. Finally, before the packaging, the completely stacked SOC is tested to ensure the quality of the whole system. For the test data delivery, the TSVs are used as test elevators. In [46], authors have presented a modular test approach based DFT architecture for 3D SoCs that supports the pre-bond and post-bond die level and stack testing. A die level wrapper based on P1500 is proposed which supports probe based die level test, test elevators for carrying the test data and control information and a hierarchical instruction register.

In [106], authors have presented a cost model for calculating the D2D (die to die) and D2W (die to the wafer) stacking considering the manufacturing test, packaging cost and yield as the affecting parameters. A formulation for the test scheduling of the 3D SoC is presented in [133-134] that considers many constraints set by power and hardware (e.g number of test pins and use of dynamic voltage and frequency scaling in hardware). The solution works on a reduction of test cost optimization and also supports multiple test insertions.

A standardized 3D DFT architecture is presented in [47]that provisions modular test approach for the 3D SoC, in which the various cores along with associated interconnects can be tested separately. The architecture assumes a hierarchical test approach that supports P1500 complaint die level and IEEE 1149.1 compliant stack level test infrastructure. The die maker provides the P1500 complaint test wrapper for ensuring the test of SoCs lying on different dies. Such a wrapper design perform the function of providing serial and parallel test interface with respect to the other dies attached to it. It also supports the testability of die's internal IP cores and their associated interconnects.

2.4 Investigation of Network on Chip based system on chip testing

For ensuring the quality of a NoC based system, it is important to initially assure that the NoC infrastructure (comprising of links, NIs and routers) is fault free so as to reuse it for the carrying the test data and hence work as TAM [142-144]. Later, the whole system comprising of the IP cores and interconnects need to be tested. Such an approach provides a benefit of cost effectiveness as the network is a part of the SoC itself and hence no additional test infrastructure overhead would be present. It is found that if this resource is used during test then it can lead to reduced test times, while other cost factors such as pin count and area overhead are strongly minimized. The main advantage of the on-chip network reuse is the availability of several accesses to each core, depending on the number of system functional inputs and outputs reused during test.

Authors in [10-11] have proposed the use of integrated switching networks as an alternative approach to interconnect cores in SoCs. Such networks, called networks-on-chip (NoCs), meet the two key requirements of future systems: reusability and scalable bandwidth. A study presented in [13],[48] shows that NoCs have better communication performance than buses for a number as low as eight cores, if intensive communication, e.g. each core exchanging messages with another one, is required. For lighter workloads (fewer messages with reduced size), the performance of a central bus will be better than the NoC in systems with up to 16 cores. Therefore, it is clear that NoCs can potentially become the preferred SoC interconnection approach in the near future. Authors in [53] presented how the network technology which originated in parallel computing could be used for on chip communication. It compared the packet based and bus based communication models and their ability to address the issues raised by a heterogeneous mix of on chip embedded processors manufactured at deep submicron level. It also discussed the various services offered by the network to support the flawless packet delivery.

Authors in [30] advocated the use of the NoC as TAM based on the comparison done between NoC based TAM and bus based TAM mechanism. Analysis is done on the basis of the testing time, routing cost, DFT area, test control complexity and test reliability [48]. The authors have also proposed the lower bound of the testing time that can be achieved using NOC similar to one proposed by authors in [82]. It considers the resource conflicts along with the underutilization of the external network bandwidth in comparison to the core’s internal bandwidth.

The NoC based approach provides a parallel access to multiple cores for the test purpose which lead to a concurrent testing of the cores[30], [145]. Test parallelization on one hand reduces the test time while on other it increases the test power. A power aware test scheduling technique for the NoC based SoC technique is proposed in [146]. The total test power which is taken as the optimization criteria comprises of the power associated with the packet transmission and the cores. Out of these the packet power is calculated on the basis of the routers and active channels being used to establish the path for the packet transmission. Hence, the power consumed by the router and the channel are added to calculate the packet power. The algorithm proposed schedules the cores in a way so as to reduce the total consumption power during the test. New DFT modules for using the NoC as TAM have been presented in [139], [144]. DFT circuitry like test wrapper and test pin insertion are proposed to aid the testability. The paper presents how the test data transportation can be done

irrespective of the underlying network and its topology knowledge. An IEEE 1500 compliant test wrapper for the network routers has been presented in [147]. Proposed designs help in achieving the testability of the routers without incurring much from area and test time penalties. A DFT technique for QDI asynchronous circuits is presented in [139] that supportsglobally asynchronous and locally synchronous scenario of NoC network. Being scalable and modular, it is reported to be suitable for networks of any size. A test pattern generation scheme for network routers is proposed in [139] that automatically generate the patterns based on the network topology and size. The algorithm is reported to provide a fault coverage as high as 99.86% (for single stuck at fault models).

Like the SoC test methodology, it is important to decide the schedule of the testing of various IP cores in a network on chip based environment too. Many researchers have proposed pre-emptive and non-preemptive styles of test schedules based on various optimization algorithms. A non-preemption based scheduling technique for system on chips that uses the NoC as test access mechanism presented in [148]. The approach utilizes the genetic algorithm for achieving the test time optimization. A rectangle based test scheduling technique for the NoC based testing approach is proposed in [149]. Test schedules for NoC based SoC testing with precedence, power and thermal constraints have been presented in [137-138],[150-151]. A particle swarm optimization based preemptive test schedule for NoC based SoC testing is presented in [152].

2.5 Gaps in the existing work

Based on the comprehensive literature survey, it can be observed that the development of efficient test solution plays an integral role in the reduction of the test cost of the overall system. One of the major components in determining the test cost is the testing time which can be categorized as the time needed to a) transport the test vectors between the ATE and the SoC periphery using the I/O pins and b) deliver and apply the test vectors from the SoC pins to the appropriate test points. Out of these, part (a) can be reduced by using compression schemes while part (b) can be reduced by efficiently designing the TAM architecture. Out of the solutions discussed above, following gaps can be extracted out as follows:

- Scope for improvement in test data compression schemes: Codes like Golomb [25],FDR [71], EFDR [72]codes are efficient in encoding the consecutive runs of zeroes and ones in the test data. However, the compression efficiency still suffers as the existence of compatibility among successive pattern blocks is not used.Utilization of special 9 cases and

dynamic variation of the test block lengths are utilized to boost the compression efficiency in fixed and variable 9C code techniques proposed in [74]. However, this increases the decoder complexity. The careful analysis of the 9C scheme demonstrates it to be still deficient in the use of the existence of compatibility within the unique blocks. Block merging [64], 2^n PRL [77]techniques utilize the inter block merging technique to improve the compression efficiency. Dynamic block size selection is utilized in CCPRL to improve the compression. The technique however has a limitation of inefficient coding of the unique blocks. The BM-8C technique utilizes encoding scheme to compress the test data at block and subblock levels. BM-8C attempts to utilize the benefits of 9C and block merging to increase the compression efficiency. However, it suffers from some drawbacks. Also, BM-8C handles the longer run lengths of same bit types inefficiently. Analyses of the various test data compression techniques reveal that not much consideration is given on the test power optimization which is yet another major component in determining the test cost. Hamming distance based test vector reordering technique has reported to address the power issue. However, the test compression needs the improvement.

Hence, a lot can be done to improve the compression efficiency of the test data by utilizing the various test data properties. Based on the frequency of occurrence of various compatible and unique cases, inter and intra block merging should be efficiently utilized to do the encoding with lesser bits. Test power has become a very critical issue as the transistor density is increasing which needs to be addressed while developing the test volume reduction codes.

- Need to optimize the TAM architecture :A lot of research has been done to optimize the time and power related to test data delivery and its application. The approaches proposed have been developed for efficient test wrapper and TAM architecture designs. Optimized test schedules can address the various constraints set by the test resources, power, TAM bandwidth etc. Many papers have addressed the test wrapper design and TAM architecture development optimization problemsindependently. Others have proposed the test schedules based on various optimization algorithms like ILP, MILP, Genetic etc. Hence, an integrated solution that co-optimizes either two or all the three problems of test wrapper design, TAM architecture development, and test scheduling need to be developed. However, such approaches still lagged the consideration for the hierarchical structure of the cores. In [38]and [93],authors haveworked on the development of efficient test wrapper and TAM architecture for SoCs with hierarchical cores and allowed the parallel testing of parent and child cores. However, not much research has been done on the further optimization of the test schedules relevant to such cases. For example various algorithms like genetic, ant

colony optimization etc. are still some platforms that may be explored to find better solutions.

With the emergence of 3D SoCs, the modular test approach of 2D SoCs need to be optimized in accordance with the 3D structures.Authors in [95] and [97]have worked on the optimization of the test wrapper for fine grain partitioned 3D cores such that the test time can be reduced while keeping the TSV under the maximum limits. In [28], authors have worked on the development of the test solution for coarse grain partitioned SoCs. However, the proposed approach has drawbacks of not supporting multiple test insertion and tests of the TSVs. No information is given about the test of the interconnections as well. The solution tries to optimize the TAM architecture while addressing the maximum TSV limit but, it doesn't consider the TSV limit for individual dies. In [130], authors have assumed the test wires to start and end at any die in the stack which is contrary to the actual scenario which mandates the I/O pins to lie on the bottom die only. In [47], authors have proposed a test wrapper for 3D SoCs that can support pre, partial and post bond tests but no information about the test schedule and test architecture at 2D level is given. To efficiently support the 3D test, reuse of the existing TAM architecture on 2D dies is appreciated in [45]. In [134], authors have worked on the test cost development and test scheduling but the optimization of the DFT architecture is missing.

With the emergence of the NoC as an alternative for bus base interconnects, its use as TAM is being advocated in [30]. Authors in [137-138]have proposed the test schedules techniques for NoC based SoC testing. The test wrapper design to support the test of NoC has been described in [139]. Few papers have described the testing of the IP cores but do not consider the test of the various network components. DFT mechanism to test the routershas been described in [139] but, no insight of the IP test mechanism is given. Not much work is done on the integrated DFT solution for network Modules and IP cores. Similarly, efficient network topologies and IP mapping techniques can be clubbed such that the test cost can be reduced. The effect of various NoC topologies with time and power constraint also needs to be worked upon.

2.6 Objectives of proposed work

The investigation will concentrate on testing cost reduction of SoCs with time and power constraints with the following main objectives:

i. To compare the existing algorithms and to modify and develop new low-cost scheduling algorithm for testing

ii. To design integrated test solution with both testing time, power, data volume constraints for flat SoCs.
iii. To design integrated test solution with both testing time, power, data volume constraints for hierarchical SoCs.
iv. Study and analysis of usage of NoC as a test access mechanism and to develop time and power optimized solution for SoC.

2.7 Summary

This chapter briefed about the works done in the domain of the efficient test development for SoC. The test time being a major component of the overall test cost is reduced by reducing the amount of test data and optimizing the on-chip test infrastructure. The work done for testing the 3D SoCs and use of NoC to carry the test data are also discussed.The following chapter describes the proposed research methodology to minimize the test cost. Various gaps described in this chapter have been worked upon to compress the test data and develop an efficient test solution for 3D SoC. Five different test data optimization techniques are described that attempts to compress the test data by utilizing the existence of compatibility at the block and sub-block levels. 3D test wrapper design for fine grain partitioned SoCs and test architecture development for coarse grain partitioned 3D SoCs are also described.

3. METHODOLOGY OF EFFICIENT TEST DEVELOPMENT FOR SoC

3.1 Introduction

ITRS Roadmap[19] predicted that test cost will be the integral component of the overall cost of the system design as the advancement continues in the level of integration. Reduction of the test cost necessitates the minimization of various implicit and explicit cost factors which are involved in the preparation and execution of the tests. One of the non-recurring factors is the cost involved ininstallation and upgradation of the Automatic test equipment. To transport the test data between ATE and SoC periphery in minimum time and reducing the amount of the ATE memory requirement, it becomes important to compress the data as much as possible. Section 3.2 presents the different test vector reduction techniques that have been developed as part of the proposed work.

Embedded core based SoC design has made the inclusion of a heterogeneous mix of IP cores to be integrated on a small silicon base. Information of such cores can be provided in different formats and may consist of different logics.The IP cores can have flat or hierarchical design. For instance, if an IP core is made up of few other cores, then it becomes the parent core with other internal cores as its child cores. Reduced device feature sizes, increasing density of components per unit silicon area has led to the controllability and observability issues related to access of deeply embedded chip components. Delivering the test vectors efficiently and appropriately to the various components under test requires a foolproof test access infrastructure. Also, it is important to have an optimized test plan for the complete SoC such that the test cost can be reduced. Modular test methods for 2D SoC have by far been well developed and emerged as a promising solution.

To perform the testing of the 3D SoCs, the conventional problem of modular core based testing needs to be modified to determine a test plan comprising of TAM architecture and core test wrapper design such that the test cost is minimized. The various test parameters that need to be taken care of while developing the test solution include: hierarchy conscious TAM and test wrapper design (specific to the stacked SOC design style), multiple test insertion of the various dies, testing of cross core and cross die interconnections, sharing of the TAM wires while assignment of the cores, power dissipation during test application, total available test bandwidth and TSVs. All above stated parameters have to be kept below or at most equal

to the global limits set by the SoC integrator. Section 3.3 proposes an efficient test wrapper design for circuit partitioned 3D core. Section 3.4 describes the proposed heuristic based test architecture which provides an integrated test solution for 3D SoC. The solution initializes by the development of an optimized test plan for 2D dies which progresses towards the development of a composite test solution for the stacked 3D SoC. Various practical scenarios have been worked upon which can be opted by the SoC test engineer based on the available flexibility. Finally the chapter is concluded with a summary.

3.2 Proposed Test Data Volume Minimization Techniques

The objective of an efficient test data volume compression technique is to reduce the amount of test bits as much as possible such that both the ATE memory and its reloads can be reduced. Meanwhile, such code based techniques should also try to reduce the switching activity associated with the shifting of the test bits through the various scan cells. The techniques that have been developed include: a) 10 coded (10C) run length based compression technique, b) Selective count compatible run length encoding (SCCPRL) c) Hierarchical block merging based (HBMTC) test data compression, d) Optimal Selective count compatible run length encoding (OSCCPRL) and e) Adaptive block merging based test data compression (ABMTC) technique. ABMTC technique works on the reduction of test power in combination to the test data. Out of these five: 10C, SCCPRL, OSCCPRL and HBMTC focuses on the compression of the test data while ensuring that the decoder area overhead to be near minimal. ABMTC not only tries to reduce the data but also the test power by minimizing the switching activity that may happen at the consecutive scan cells of the core's test wrappers.

The techniques utilize the test compression at inter and intra block level to improve the compression efficiency. The schemes iteratively improve upon the short comings of some of the recently proposed works in the literature. For instance: 10C and SCCPRL attempts to incorporate improvements over the 9C and CCPRL schemes while OSCCPRL further mixes the two to obtain even much higher compression. HBMTC works on the improvement of the BM-8C technique by incorporating few changes inspired by the analysis of the experimental results. Being simple code based approaches; the complexity of the on chip decompressors is kept minimal. The ABMTC technique uses the test vector reordering and blocks merging to reduce both the test power and test data. Hence, one of the proposed approaches can be chosen by the test engineers to save the test time, ATE memory, decoder area and test power. The description of the various techniques is as follows.

3.2.1 10 Coded Run length encoding technique

Proposed 10 coded compression scheme works on the optimization of 9C by compressing the unique cases on the basis of the compatibility/ inverse compatibility at the half block length level. It categorizes the unique combination as: *UU* (consisting unique but compatible half-length blocks (HLB)), *UU'* (unique but inverse compatible HLBs) and *UV* wherein the two HLB's have no compatibility.

The coding scheme consists of three parts namely: *prefix* bit, *sub-prefix* bit and *tail* bits. Out of these, the *prefix* bit helps in discriminating betweenblocks with occurrence of special case (*00/01/10/11/0U/1U, U0* or U1 combinations) and unique blocks. *Prefix bit* is set to '*1*' to represent special case and '*0*' to signify unique case. As stated earlier, unique cases are further categorized as compatible (either *UU* or *UU'*) or *UV* which are represented by setting the *sub-prefix* as '*1*' or '*0*' respectively. Among the compatible blocks the categorization between *UU* and *UU'* cases is done by setting the *tail* bit as '*1*' or '*0*' respectively. Similarly, the *tail* bits are appropriately set to encode for the occurrence of one of special eight cases. The whole coding scheme is as shown in table 3.1. The unique block or HLB pattern are added to the code word in cases of occurrence of *UV* and one out of: *0U,1U,U0,U1, UU* and *UU'*to avoid any loss of information and accurate retrieval of the test data.

Like 9coded technique, 10C also starts by segmenting the test data into blocks of equal sizes. Thereafter, each block is further partitioned at its half block length level and investigated for the presence of special cases as discussed above. Based on the occurrence the various cases, code words are appropriately developed.

On comparison with the 9C coding technique, the proposed 10C offers an advantage of 1 bit for the following sub cases : *10,01,U1,U0,0U* and *1U*, *1*+(length of HLB) bits for *UU* and *UU'* sub-cases and 2 bits for*UV* case. However, it has a penalty of 3 bits for *all zeroes* all 2 bits for *all ones* sub-cases.

Table 3.1 Encoding scheme of 10 coded compression technique

Input Block	Sym-bol	Pre-fix	Sub-Prefix	Tail Code word	Decoder Input for 10c	No of bits (10C)	Decoder Input for 9C	No of bits (9C)
00000_00000	0	0	NA	000	0000	4	0	1
11111_11111	1	0	NA	111	0111	4	10	2
00000_11111	01	0	NA	001	0001	4	11000	5
11111_00000	10	0	NA	010	0010	4	11001	5
00000_UUUUU	0U	0	NA	011	0011_UUUUU	9	11010_UUUUU	10
11111_UUUUU	1U	0	NA	100	0100_UUUUU	9	11011 _UUUU	10
UUUUU_00000	U0	0	NA	101	0101_UUUUU	9	11100_UUUUU	10
UUUUU_11111	U1	0	NA	110	0110_UUUUU	9	11101_UUUUU	10
UUUUU_UUUUU	UU	1	1	0	110_UUUUU	8	1111_ UUUUU_UUUUU	14
UUUUU_U'U'U'U'U'	UU'	1	1	1	111_UUUUU	8	1111_ UUUU_U'U'U'U'	14
UUUUU_VVVVV	UV	1	0	NA	10_UUUUU_VVVVV	12	1111_ UUUUU_VVVVV	14

For example, a random sequence:'*0000XX0XX0111X110101X00X0111 X1X00X000XX001XX1010110101110100*' when segmented in blocks of size 10 bits result in the patterns given in column no 2 of table 3.2. On encoding the blocks information using the 9C and 10C codes, the resultant encoded data length becomes 45 and 37 respectively. As evident from the example, use of 10C leads to reduction of 8 more bits which can enhance the achievable compression.

Table 3.2 Example to demonstrate 10C encoding compression efficiency over 9C

Input Block	Combinations with block length K = 10	Encoding combinations as per 9C with			Encoding combinations as per 10C		
		Cases	Codeword	No. of bits	Cases	Codeword	No. of bits
1	0000X_X0XX0	00	0	1	00	0000	4
2	111X1_10101	1U	11100_10101	10	1U	0100_10101	9
3	X00X0_111X1	01	11000	5	01	0001	4
4	X00X0_00XX0	00	0	1	00	0000	4
5	01XX1_01011	UU	1111_ 01XX1_01011	14	UU	110_01011	8
6	01011_10100	UU'	1111_ 01011_10100	14	UU'	111_01011	8
Total no of bits				45			37

The extent of the test data compression achievable by the adoption of 10C over fixed 9 C is solely dependent on the test data. Lesser the occurrence of *all zeroes* and *all ones* sub cases in comparison to other sub cases, more will be the increase in the test data compression.

3.2.2 Selective Count Compatible Run length encoding technique

The proposed selective CCPRL technique, aims at optimization of CCPRL by adding a detect bit which is used to categorize the blocks as compatible or unique. The differentiation between the unique and compatible blocks is done with the help of *detect bit* which is added as a prefix to code word. The status of the *detect bit* is set to *'1'* if the merging is possible else it is set to *'0'*.

The corresponding coding scheme so formed consists of encoding prefixes (*block code*which indicates the binary conversion of pattern length and needs to be added in the beginning, *detect bit* used to represent if merging is possible or not) and tails (*retained pattern code* which presents the retained pattern sequence with length equal to the decimal value corresponding to block code, *count code* to represent the count of compatible blocks with retained pattern and*compatibility code* which signifies compatibility relation of each block with retained pattern block and is set to *'0'* if blocks are compatible else set to*'1'* if they are inverse compatible). For *Unique cases*, retained pattern needs to be just preceded by the *detect bit* (equal to '*0'*). This nullifies the need to send redundant count code (with all zeroes to represent 0 block matching). The encoding of the test data in case of inter block merging is done in a way very similar to CCPRL.

An example to show the effectiveness of the SCCPRL over CCPRL for the sequence '*010101010101101010101010110110111110'* is provided in figure 3.1. The pattern length *k* is taken to be 6 (fixed) and the lengths of block and count code is taken to be 3 bits each.The example shown in the figure 3.1 employs fixed block size due to which the block count (*110*) needs to be sent just once in the beginning of the encoded output. It may be noted that if the same encoding had to be done without using detect bit then additional count code (*000*) had to follow the retained patterns to signify unique cases.

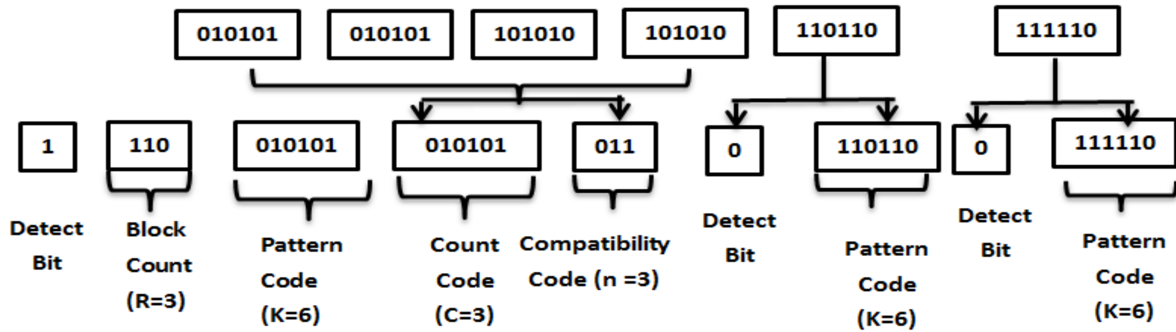


Figure 3.1 The codeword generation using SCCPRL

For increasing the compression even further, variable block sizes can be used. This is done as follows: for a block count of R bits, all combinations of block sizes ranging from 3 to 2R-1 are tried to find the optimum compression. For example, for a block count of 4 bits, iterations are run for block sizes equal to 3, 4,5,6….15 and the corresponding compression efficiencies are calculated and compared.

The best block size value is chosen and accordingly the code word is generated. The minimum value block size equal to '3' is chosen since below 3 bits the number of the test data bits will be expanded instead of getting compressed [76]. In each iteration, search is done till no more compatibility is found and a code word is generated accordingly. Left over bit sequences are put for next round of investigation. For keeping the decoder design simple the number of bits in the block count and count code are chosen to be same.

3.2.3 Hierarchical Block Merging based encoding technique

Though the use of 10 C can increase the test data compression beyond what was achievable using 9C but, the restriction of comparison of only two HLBs (at maximum) still limits its benefits. Based on the experimental data analysis, it was found that the compression can be increased by merging various compatible/inverse compatible blocks. Such an idea worked as the motivation behind the proposal of Hierarchical Block Merging based test data compression technique (HBMTC).

In HBMTC, the test data is segmented in fixed size blocks which are further compressed using inter and intra block level merging techniques. Initially, the blocks are categorization as compatible and unique blocks. To enhance the compression, unique blocks are investigated for occurrence of special eight cases (*00, 01, 10, 11, 0U, 1U, U0, U1, UU (compatibility)/ UU' (inverse compatibility)* at HLB level. If any of the cases exists, then a suitable code word is developed to encode the pattern. The compatible blocks on other hand, are merged to form

a retained pattern block along with a block count and compatibility code to represent the relation between the retained block and the blocks being merged. To improve the compression even more, the retained block is further investigated for existence of compatibility at the HLB level. Such an approach is termed as inter-intra merging since merging is initially done between the blocks and thereafter between the HLBs formed out of the retained pattern.The HBMTC technique consists of six types of encoding bits namely: *detect*, *inter block expander prefix (IBEP)*,*count code*, *prefix*, *tail and compatibility bits* as shown in the Table 3.3. The significance of various bits are as follows:

The status of the *detect bit* is set to '*0'* and *'1'* to signify unique and compatible blocks. The block count is split into two parts: *IBEP* and *count_code*. Once the block count is calculated, the first task is to identify the inter/group case which carries the information about grouping of the number of blocks being merged in multiple of eight. This categorization is done to incorporate the frequency directed encoding of the block count. As per the experimental results, the occurrence of blocks merging is more upto a block count of 20 beyond which it gradually falls. Therefore, the grouping of the block counts in multiples of eight is more economical in terms of size of the codewords used for encoding. The information about theintercase number (factor of 8 used to express the block_count) can be extracted by number of *1'*s kept in it. The *IBEP* is ended by a 0 for appropriate identification. The remaining value of block count (in the range of 0- 7) is held in the *count_code*. Use of such type of scheme to encode the variable block_count is done to represent its value in a more uniform way. And hence the complete block_count value can be represented as per eq. 3.1.

Block_count = $8 * i + (decimal\ convesion\ of\ Count\ code)$ (3.1)

where variable *i* represents IBEP group index

IBEP and *count_code* are further followed by *prefix bit* which specifies the existence of compatibility between two HLBs. As done in the case of 10C, the *prefix bit* is set to '0' and '*1*' respectively to signify if the two HLBs are compatible/inverse compatible or non-compatible. The existence of special cases or type of compatibility between the HLBs is represented by the additional *tail bits.* The status of *tail bits* can be asserted according to the column 4 of table 3.3 to represent the existing sub case. In case of various inter-cases (except for unique cases), *compatibility bits* follows the whole code word so developed to specify the relation between the representative pattern block and successive merged block.

Table 3.3 Encoding scheme of test data compression using Hierarchical block merging Technique

Detect bit	Inter block expander (IBEP)	Count code	Prefix	Tail	Sub case	Codeword
0 (unique case)	Not required	Not required	0	000	00	00_000
			0	001	01	00_001
			0	010	10	00_010
			0	011	11	00_011
			0	100	0U	00_100_b/2
			0	101	1U	00_101_b/2
			0	110	U0	00_110_b/2
			0	111	U1	00_111_b/2
			1	NA	UU	01_UU
1 (inter case-1)	0	3 bits (1-7)	0	0	half_comp	10_3bits_00_b/2_ compatibility bits
			0	1	half_inv_comp	10_3 bits_01_b/2_compatibility bits
			1	NA	Unique	10_3bits_1_b (block_size)
1 (inter case-2)	10 – 8 to 15 no of blocks matched (8*1+(0-7))	3 bits (0-7) 111	0	0	half_comp	110_3 bits_00_b/2_compatibility bits
			0	1	half_inv_comp	110_3 bits_01_b/2_compatibility bits
			1	NA	Unique	110_3bits_1_b (block_size)

The pseudo code for the HBMTC is as shown in the figure 3.2. The algorithm has 4 main functions: *store_pattern_code*, *compare_block, Intra10c_check and do_encode_intra.* Out of these functions, *store_pattern_code* function (as specified in step 1 of the pseudo algorithm), helps initialize the *pattern_code* (representative block) of the length equal to *block_size*. The *pattern_code* starts from the *start_idx* (used as a pointer for start of the block under consideration) to that of *start_idx* plus *block_size*. The *next_block_idx* used to denote the start of the next block is initialized with a size equal to *start_idx* plus *block_size.*

The step 2 of the algorithm performs the investigation of compatibility among successive blocks and determination of the resultant *block_count* value. The *compare_block* function (as

specified in step 2) examines the contents of *pattern_code*with its consecutive data block starting from *next_block_idx* to *next_block_idx* plusthe *block_size*. It sets the *merge_possible* variable to *'1'* if the blocks are compatible else it is set to *'0'*. Depending upon the status of the *merge possible*, the counter *Block_count* is incremented by one. The compatibility *block_relation* array is updated with the necessary compatibility bit each time the *Block_count* is incremented to show the relation between the *pattern_*code and the successive merged block. This process is repeated until the *merge possible* signal is found to be *'0'*. The *merge possible* variable is also used to appropriately set the *detect bit as per the* table 3.3.

Another task performed in step 2 of the pseudo code is to appropriately set the values of *IBEP* and *count_code*based on the value of *Block_count.* A comparison is made between the *Block_count* and *MAX_COUNT_CODE*(whose value is set to be eight). *If Block_count* value is found to be larger, then *'IBEP* variable is incremented and the *block_count* is updated by subtracting eight from it. Herein, the increment operation leads to addition of *'1'* infront ofa *0* to signify the multiples of eight which are further used to categorize various inter cases.This whole process is repeated until its value becomes smaller. The leftover *Block_count* value is converted to binary format to form the *count_code*.

The *pattern_code* (representative block) is used to initialize two HLBs of length equal to half the *block_size* termed as *patt_Subblock1* and *patt_Subblock2* (as specified in step 3). These blocks are further investigated for the existence of compatibility at the sub block level using the *Intra10c_check* function. This function finds out the occurrence of one of the special cases (*00, 11, 01, 10, 0U, 1U, U0, U1, UU, UU'* and *UV*). The function appropriately sets the value of *encoded_case* output based on the value of *Block_count* (to include the *prefix, sub-prefix* and *tail* bits). Another variable *encoded_case_idx* keeps the count of the length of *encoded_case*. Based on the outcome of *Intra10c_check* and the status of *detect_bit*, the final *encoded_array* is updated using either of the *Do_encode_inter* or *Do_encode_intra* functions.

```
Algorithm1: Pseudo Code for the Test Data compression using Hierarchical Block Merging Technique

While (start_idx<final_idx) {

Step 1: store_pattern_code (pattern_code, input_array, start_idx, block_size);

merge_possible=1;    // setting merge_possible=1 so as to check for merging each time

next_block_idx= start_idx+block_size; //initialize the next_block idx, start_idx

Step2: while (merge_possible && (next_block_idx<final_idx)) {

        merge_possible=compare_block (pattern_code, final_idx, input_array, start_idx, next_block_idx, block_size, equal);

        if (Block_count>=MAX_COUNT_CODE){

                Increment IBEP;

                Block_count=Block_count- (MAX_COUNT_CODE +1);}

                Decimal_to_binary ( count_code, Block_count);

        if (merge_possible) {

                Set detect_bit;

                compatibility_bits [block_count] =equal;

                Block_count++;

                next_block_idx+=block_size; }}

Step3: for (i=0; i < (block_size/2); i++) {

                patt_Subblock1[i] =*(pattern_code+ i);

                patt_Subblock2 [i] =*(pattern_code + (block_size/2) +i);}

Step4: Intra10c_check (patt_Subblock1, patt_Subblock2 (block_size/2), block_count, encoded_case_idx, encoded_case);

Step 5: if (Block_count==0 && detect= '0') // Unique cases

        Do_encode_intra (encoded_case_idx, encoded_case, encoded_array_idx, encoded_array);

      else // inter Block case

        Do_encode_inter (count_code, block_bin, IBEP, Block_count, pattern_code, block_size, compatibility_bits,
        encoded_array_idx, encoded_array);

        start_idx=next_block_idx;

         Reset detect_bit; }

Block_count=0;}

}
```

Figure3.2Pseudo code of HBMTC

The value of the *encoded_case*is concatenated to the *encoded_array* used to store the compressed stream of test data. Finally, the updating of the *encoded_array_idx* is done based on occurrence of inter/intra block merging. The inter-intra block level encoding is done using *Do_encode_intra* function. It does the encoding based on the existence of compatible cases (*00, 11* and *UU*)/ inverse compatible cases (*01*, *10* and *UU'*) or unique case (*UV*) at the sub block or block level of the retained pattern. The non-compatible patterns (signified by *detect_bit* equal to '*0'*) are encoded as per the existence of one out of 10 cases (*00/01/10/11/0U/1U/U0* and *U1*). Such a intra-intra level block encoding is done using the

Do_encode_intra function. The patterns which cannot be merged at block or sub block level are encoded to reserve the whole pattern information to avoid loss of information.

3.2.4 Optimal selective Count Compatible Run length Encoding technique

OSCCPRL technique is a consolidation of achievable benefits of 10 C and SCCPRL techniques. Initially categorization of the blocks is done on the basis of uniqueness and compatibility of the blocks using the *detect bit.* Like 10C, if the pattern is found to be unique, it is further investigated and encoding is done as per the presence of the special eight cases. Meantime, if the blocks are found to be compatible/ inverse compatible, SCCPRL encoding format is used to develop the code word. Thus, two extra bits namely, *detect bit* and *prefix bit* as described in the table 3.4, are used to develop the codeword. It may be observed that the status of the *prefix* bit helps in identification of the occurrence of one of *special eight cases* and incompatible blocks.

OSCCPRL techniques works on the principle of achieving maximum compression by opting either inter or intra block merging on the basis of the compression achieved by both. This technique works as follows: based on the value *R* (number of bits to represent block code), inter block level investigations are done using different block sizes. Encoding is done similar to SSCPRL using *detect bit*, *block_size*, *pattern_code*, *count code* and *compatibility code*. Based on the analysis of the compression efficiency achieved in each case, selection of the optimum block size and its associated compression efficiency is done. Similarly, the pattern block of fixed size is selected and considered for intra block checks (to explore the possibility of occurrence of one of special eight cases as mentioned under sub cases 1-8 of table 3.4). The encoded code word so formed includes detect bit followed by prefix and tail code word.

Table 3.4 Encoding scheme of Optimal Selective Count Compatible Run Length encoding technique

Cases	Sub Case (i)	Input Block (K=12)	Case	Detect Bit	Prefix	Tail Code word	Decoder Input
A (Intra Block)	1	000000_000000	00	0	0	000	00_000
	2	111111_111111	11	0	0	111	00_111
	3	000000_111111	01	0	0	001	00_001
	4	111111_000000	10	0	0	010	00_010
	5	000000_UUUUUU	0U	0	0	011	00_011_UUUU
	6	111111_UUUUUU	1U	0	0	100	00_100_UUUU
	7	UUUUUU_000000	U0	0	0	101	00_101_UUUU
	8	UUUUUU_111111	U1	0	0	110	00_110_UUUU
	9	UUUUUU_VVVVVV	UV	0	1	UUUUUU_ VVVVVV	01_UUUUUU_VVVVVV
B (Inter Block)	10	(Bi/B'(i+1) Or	Comp/ Inv_comp	1	NA	Selective CCPRL	1 _Pattern Code_ Count code _compatible Code

Using the codeword length, Intra block level compression efficiency is also calculated. Based on the comparison of the compression efficiencies obtained in each case, either of the two is selected. For rare cases when both inter block and intra block level merging fail, then the unique pattern is sent as such along with *detect* and *prefix* bits being set to *'0'* and *'1'* respectively. As is evident from the table 3.4, the number of the redundant bits that need to be sent in case of unique pattern is always two irrespective of the size of the pattern code. It may be noted that the compatibility check at the sub block level as mentioned in 10C technique has been intentionally avoided since its inclusion would require an extra sub prefix bit which nullifies the advantage offered by SCCPRL

3.2.5 Decompressor architectures

The retrieval of the original test data requires inclusion of an on chip decompressor. The following discussion presents the decompressor designs for proposed OSCCPRL and HBMTC techniques.

3.2.5.aDecoder design for OSCCPRL

The decompression architecture for the optimal selective count compatible run length encoding is shown in figure3.3. From the top level the decompressor has four incoming signals: *ATEclk, CUTclk, data_in* and *Ack*and one out going signal: scan_out. The encoded data stream is received from ATE through *data_in* at ATE clock frequency *(ATEclk),* decoded and fed to the circuit under test (CUT) at the circuit frequency *CUT_clk* through *scan _out* line. The major blocks of decompression architecture are FSM, module R/ module *C* counter, *K* bit counter, *K* bit shift register and buffer.

The FSM is used to generate appropriate control signals based on the various bits received from the *data_in*. The counter1 :module *R*/*n* are used to count till the block count and count code while the *K* bit counter is used for receiving the pattern code. The *K* bit multiplexer is used to fill in the appropriate bits to deliver the test data based on the select lines received from the FSM. The *scan_en* signals finally controls when to enable and let go the test data onto the circuit under test. The description of the various blocks of the decoder is as follows:

*Counters:*The decompression circuit consists of two counters: counter1: *module R/module C* counter and counter2: *K/n* bit counter. The functions of the first counter is to count and hold the value of the *retained pattern length* (module *R*) in the start of the decoding process and *count code* (module *C*) during the later part when merging is being done at the block level. As described in section 3.2.2, *R* and *C* values helps in defining *retained pattern length* and number of such compatible pattern blocks. The counter 2 performs three functions: reception of the pattern block of *K*bit length/ sub block of *K /2* bit length or compatibility bits equal in number as defined by the *C* value. The working of the counters can be explained in reference to selective CCPRL case and eight special cases as follows:

Inter block merging (Case1): On the initialization of reception of encoded data from ATE as signified by ACK signal, the initial R bits are loaded into the counter 1 using *inc*and *flag1* signals. For holding the *block code*, the *flag1* is set to *'1'*. On completion of the *R* bits, the counter asserts a *'1'* at the *done1* line. On receiving *done1*=1, the FSM sets flag2 *[1:0] = '00'* to make counter2 work as *K* bit counter. *Shift, dec2* signals help shift the data to the *K* bit scratch register through *pattern_in* line. On completion of the counting/ resetting of counter the *done2* signal is set to '*1'*. Next, FSM generates the control signal *flag1 ='0'* to make the counter1 work as module *C* counter. Once the counter starts its task it maintains *done1= '0'*. On completion of module *C*, the counter asserts *done1= '1'* again. Occurrence of high on *done1* leads the counter2 as *n* bit counter to receive the compatibility bits. This is done by

setting the *flag2[1:0]* lines equal to *'01'*. Corresponding to the compatibility bits the FSM generates the appropriate select signals for MUX which either feeds the *retained pattern* or inverse of it to the buffer. The data on the *K* bit buffer is then shifted out to the circuit under test using the *scan_en* signal generated by the FSM. From here the data would be sent to the circuit under test at *CUTclk*.

Intra block merging: special eight cases (Case2): During the occurrence of one of the *'0U,1U,U0,U1'* combinations as decided by the FSM, the counter 2 and scratch register are made to work till *K/2*. This is done by setting the status of the *flag2[1:0]* lines = '*10*'. The rest of the data packet formation and transfer to CUT is done in more or less the same way as explained earlier relative to select line *sel[1:0]* and *scan_en*. The occurrence of one out of rest four cases *(00, 01, 10, 11),* deactivates the counters. FSM sets the appropriate *sel[1:0]* lines for appropriate data packing.

Unique case (Case 3): On reception of *detect='1'* the FSM sets *Flag2 [1:0]* is *'11'* and hence the counter2 and scratch registers are made to work for *K* bits. However here the select lines allow the pattern received to be directly fed to the CUT through the *K* bit buffer.

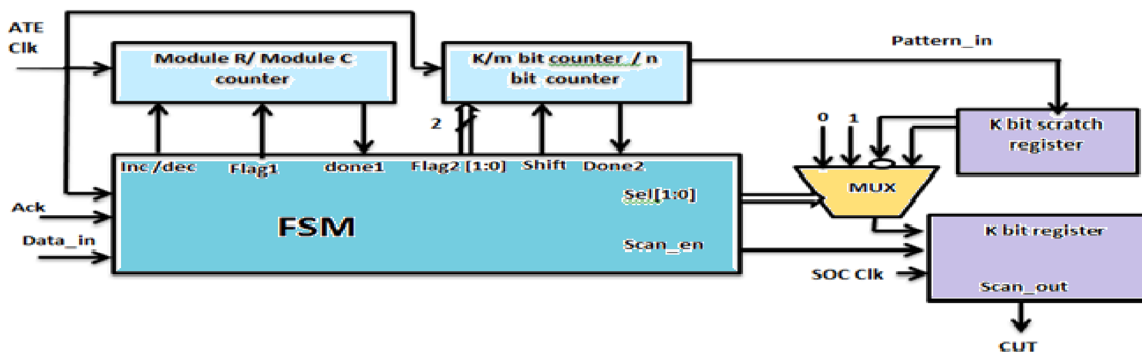


Figure 3.3 Decompressor Architecture of OSCCPRL

FSM : The FSM of the decoder is shown in figure3.4. It initiates in state *S0* wherein the initial fixed block count value is uploaded in counter 1. On receiving the complete count value, counter generates the done1 signal. From here if the *data_in* value is equal to '*1*', then the FSM generates control signal for SCCPRL at block level. On the other hand, receiving a '*0*' on *data_in* signal, makes the FSM generate an output analogous to occurrence of one of *special eight cases* (described earlier). Depending upon the various states, the FSM generates the select lines for the output buffer and various counters. The leaf nodes: *C0-C9* represents the different encoding cases.

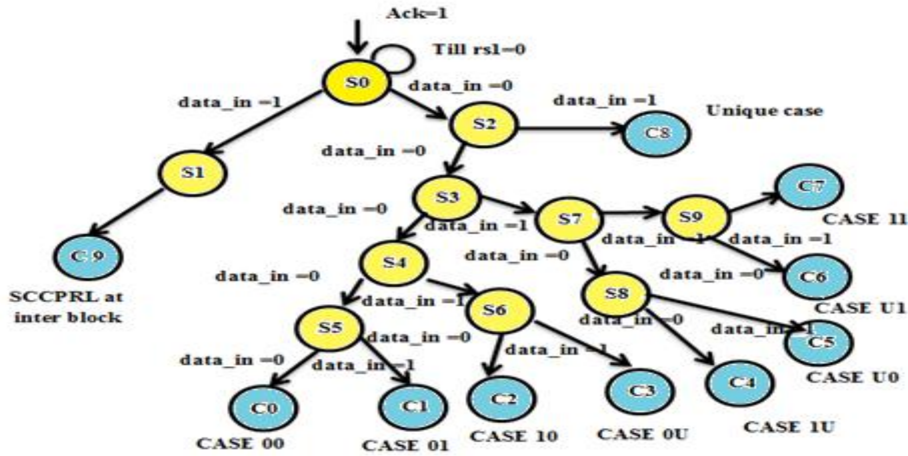


Figure 3.4FSM of Decompressor Architecture of OSCCPRL

3.3.5.bDecoder design for HBMTC

The decompression architecture for the proposed Hierarchical block merging technique is shown in Figure 3.5. From the top level the decompressor has four incoming signals: *ATEclk, CUTclk, data_in* and *Ack*and one out going signal: *scan_out*. The encoded data stream is received from ATE through *data_in* at ATE clock frequency *(ATEclk),* decoded and fed to the circuit under test (*CUT*) at the circuit frequency *CUT_clk* through *scan _out* line. The major blocks of decompression architecture are FSM, counter1 comprising of n bit register and subtractor, Counter2, *K* bit MUX and buffer.

The FSM is used to generate appropriate control signals based on the various bits received from the *data_in*. The counter1 consists of an *n* bit register and *n* bit subtractor which are used to hold the *IBEP* Code. Counter 2 performs the following function: work as module seven counter to receive the count code; *K* (block size) or *K*/2 (block_size/2) bit counter to aid the FSM in the extraction of the retained pattern; and counter to receive the compatibility bits so as to feed the appropriate bit pattern to the CUT. A *K* bit MUX is used to allow the appropriate bits out of the 4 inputs (0/1/D_i or D_i') to reach the buffer. The complete *K* bit block pattern so received by the buffer is further supplied to the CUT through *scan_out* line at *SOCclk* frequency based on the status of *scan_en*. The description of the various blocks of the decompressor and counters in reference to Inter and Intra level block merging led encoding is as follows:

Inter block merging: Initialization of reception of encoded data from ATE is signified by *ACK* signal. The initial *K* bits are received by the FSM to represent the block size. Block Size is fixed for the complete data stream. FSM initiates the register to receive/count and hold the *IBEP* value which specifies the IBEP Code by setting *flag1* equal to HIGH. Here IBEP = $\log_2$

(max(*IBEP_code*)). The register keeps receiving and shifting the IBEP value till zero is not encountered. At the end it pushes *done 1* equal to HIGH. Hence a corresponding n value is H in the register. Meanwhile, the IBEP bit subtractor is kept inactive using the *sub_en*signal.

When a HIGH status is received by the FSM on the *done1* signal, it asserts the *flag[1:0]* equal to *'00'* to enable the counter2 to count the value of the Count code. When the complete Count code value is received by the counter 2, it asserts the done2 signal to HIGH. A high on *done2* pushes the count value onto the 3 bit shift register. When register 2 is loaded, the FSM asserts a LOW on *done2* and clears the counter2 using *Load* signal. Thereafter, the FSM enables the counter2 to count till K/2 or *K* based on its various states as shown in figure 3.4.

Depending on the existence of compatibility or uniqueness a at sub block level, the FSM either asserts *flag[1:0]* = '*01*' (to enable the counter2 to count till k/2) or *flag[1:0]* = '*10*' (to enable the counter to till k bits). Meanwhile, appropriate retained pattern length is received by the decompressor. On completion of counting, counter2 asserts a HIGH on *done2*. Appropriate status is set on *Sel[1:0]* by FSM and appropriate bits are fed in the scratch register through the MUX. The FSM then asserts *flag[1:0]* = '*11*' to make the counter2 to get reloaded with the count code that was saved in the register 2 . With this the *inc /dec'* is asserted low and counter2 starts decrementing till it reaches zero. This makes *done2* equal to HIGH again. On this FSM sets the *Sub_en* equal to HIGH,00 and makes *n* bit subtractor to get decremented by one. FSM clears the counter2 and asserts *done2* equal to LOW. *Inc/Dec'2* signal is set to HIGH by FSM and hence counter2 starts counting again and hits *done2* equal to HIGH on completion. This decrements the n bit subtractor again. The counter2 is reloaded again as done earlier. On completion, the n bit subtractor again decrements. This process of reloading and decrementing is done till the n bit subtractor reaches zero. This is done to retrieve the complete compatibility bits.

Corresponding to the compatibility bits the FSM generates the appropriate select signals for MUX which either feeds the retained pattern or inverse of it to the buffer. The data on the k bit buffer is then shifted out to the circuit under test using the *scan_en* signal generated by the FSM. From here the data is sent to the circuit under test at *CUTclk*.

During the occurrence of one of the *'0U, 1U, U0, U1'* (*Intra block merging cases)* combinations as decided by the FSM, the counter 2 and scratch register are made to work till K/2. This is done by setting the status of the *flag2[1:0]* lines = '01'. The rest of the data packet formation and transfer to CUT is done in more or less the same way as explained

earlier. The occurrence of one out of rest four cases *(00, 01, 10, 11),* deactivates the counters. FSM sets the appropriate *sel[1:0]* lines for appropriate data packing.

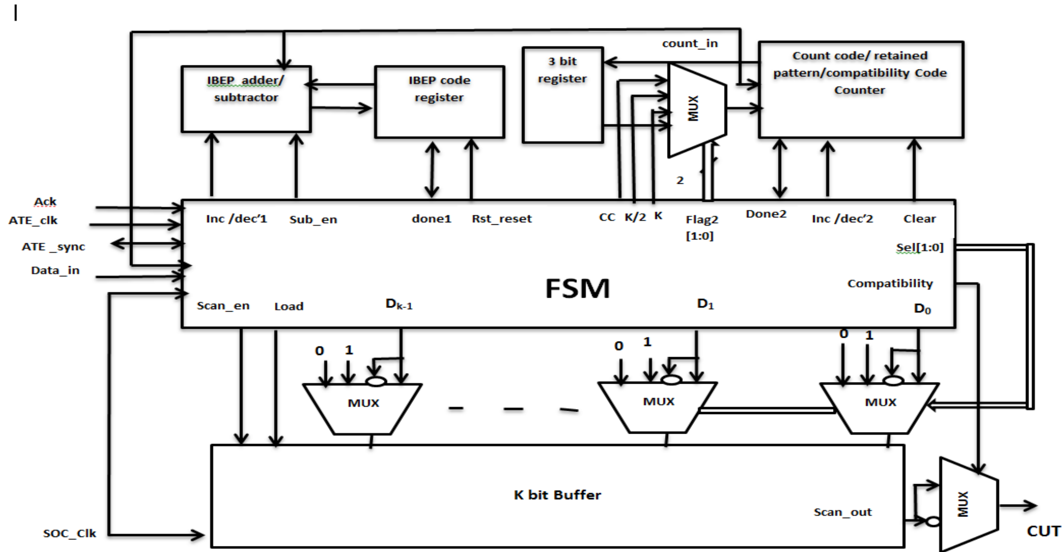


Figure 3.5Decompressor architecture of HBMTC

FSM:The FSM of the decoder is shown in figure 3.6. It initiates in state *S0* where in the initial block size value is received. On receiving the *done2* equal to HIGH and depending on the status of signal *data_in*(*detect_bit*) the FSM either enters state S1(inter block case)or S2(Unique case). The next values of the *data_in*signal helps the FSM to generate the necessary control signal for various combinations at block and sub block levels. The leaf nodes *C0-C12* represent the different encoding cases.

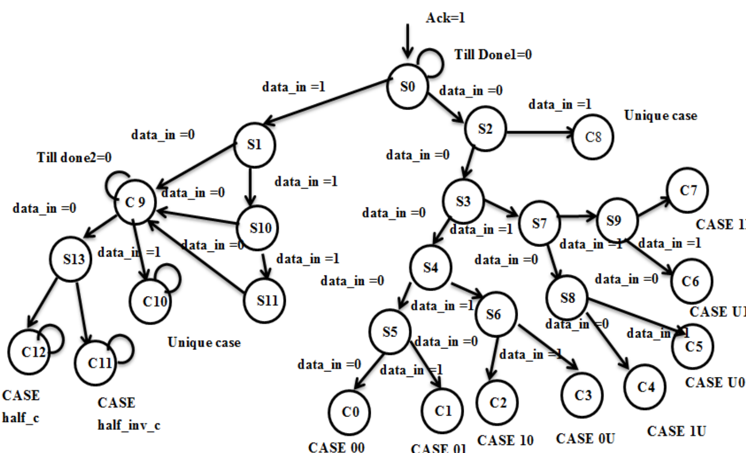


Figure 3.6FSM of Decompressor architecture of HBMTC

3.2.6 Adaptive block merging test data compression

Adaptive block merging based test data compression tries to reduce the switching activity along with test data by employing the hamming distance based reordering and block level merging. It initially reorders the test data on the basis of the hamming distance such that the consecutive test vectors may have minimum switching activity. Thereafter, the test data is divided in blocks of equal lengths which are further investigated for reducing the number of test bits. Like the various techniques discussed above, the proposed technique initially tries to merge the blocks on the basis of inter block compatibility to reduce the number of test bits, if it fails then individual block is iteratively segmented at half block length level and investigated for intra level block merging.

Similar to HBMTC and OSCCPRL, a *pattern_code* of block_size equal (which is chosen to be 32) is compared with its successive block for existence of compatibility or inverse compatibility. If two or more blocks are found to be compatible then a suitable encoding is done that consists of *retained pattern (pattern_code)*, *count_code* to represent the count of the number of blocks being merged and the *relation code (compatibility code)*. The *relation code* consists of an array of length equal to the decimal count held in *count_code* and its values being defined by the type of compatibility between the *retained pattern*and its successive blocks. The relation bit is chosen to '*1*' or '*0*' to show the existence of compatibility or inverse compatibility between the *retained pattern*and the block being merged. The block size of 32 bits has been chosen since it can be segmented iteratively to perform the best possible compression. The encoding scheme for ABMTC is as presented in figure3.7.

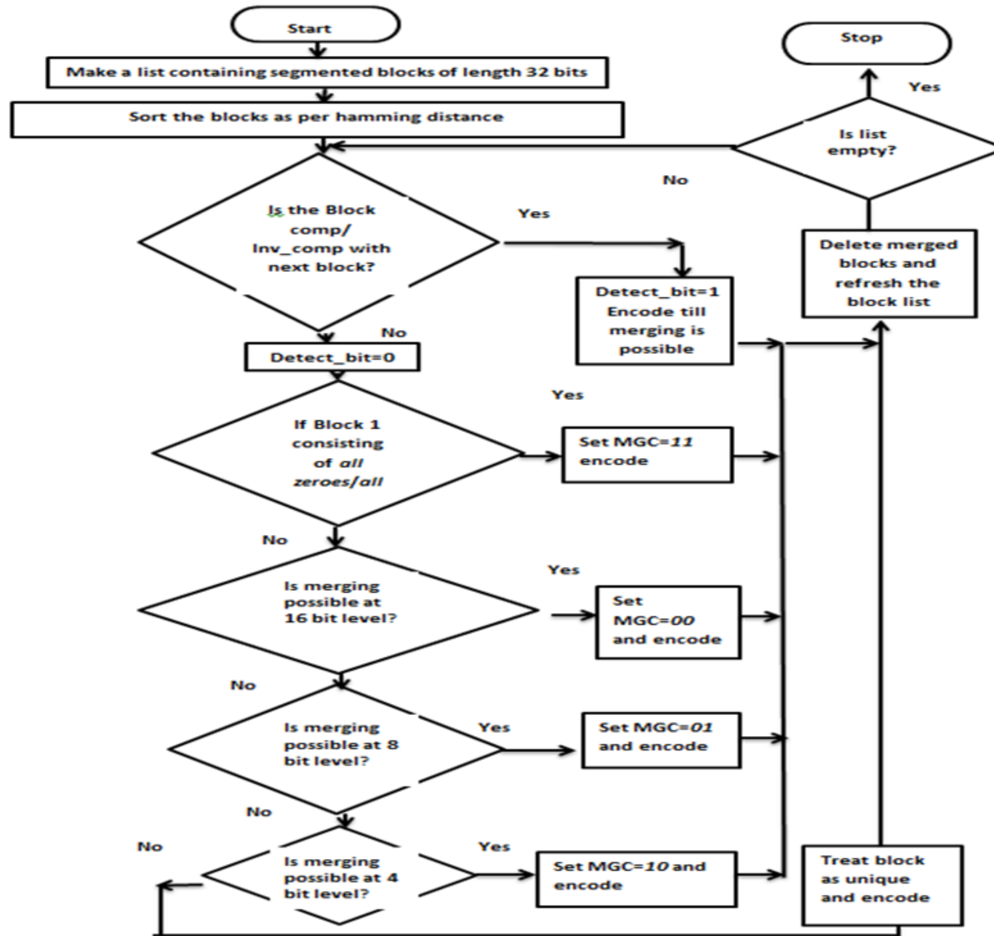


Figure 3.7 Flow Chart of ABMTC

The encoding scheme consists of *inter/intra bit, group code, unique_intra_merge_prefix bit (UIMP)* and *tail* bits. The *inter/intra bit,* is set to '*1*' if the 32 bit inter block merging is possible else it is set to '*0*' to represent a unique case. The *group code* represents the block length at which the intra level encoding is being done. Its value is set to '*00*', '*11','10'* and *'01'* to represent merging being done at block sizes of 32, 16, 8 or 4 bits level. The *unique_intra_merge_prefix bit (UIMP)* is set to '*1*' if compatibility is found at intra block level else, it is set to '*0*' to carry the unique pattern intact. The *UIMP* is further followed by *tail bits which* are used to represent the sub cases.The pattern block also called the parent block of block_size equal to 32 is investigated for the occurrence of block merging.The parent block on being found unique is further checked for conditions of *all zeroes* or *all ones* at the block size (32 bits) level. The code words for *all zeroes* or *all ones* cases can be generated as per the second case of the encoding scheme shown in table 3.5. For example: if the *pattern_code* of 32 bits length is found to be consisting of only *X*'s and *0*'s then, the complete pattern is filled with *0*'s to make it an *all zeroes* block. The corresponding code

becomes: *0* (*inter/intra*)+ *00* (*group code* for 32 bits block) + *1*(*UIMP*) +*0*(*all zeroes* case). Similarly, encoding can be done for *all ones* case at 32 bits level.However, if both the above conditions fail then the *pattern_code*of 32 bitslength is segmented further in two halves of length 16 bits each. Two HLBs so formed are investigated further for existence of one of eight special cases *U0, U1, 0U, 1U,01, 10, comp* and *inv_comp)* among themselves.

Table 3.5 Inter/ Intra block merging at 32 bits level

Case	Inter/ intra	Group_code		UIMP	Tail bits		Code word
		Code	Inference		Code	Inference	
1	1	Not reqd.	32 bit inter block merging	Not required.	Inter block Count code + inter block relation	Block merging at inter block level	1_retained pattern block (32 bits) _Inter block Count code _Relation Bits
2	0	11	32 bit merging	0	10	*all zeroes* in 32 bits	0_11_0_10
					11	*all ones* in 32 bits	0_11_0_11

In case, if one of the following relations: *U0, U1, 0U* and *1U* exists between the two HLBs then the *UIMP bit* is set to '*0*' and the corresponding encoding is done as per table 3.6. Similarly, if the cases of compatibility like: *01, 10*, *comp*., *inv_ comp***.**is found then the *UIMP* is set to '*1*' followed by the appropriate code-words that consists of the *group code* being set to '*11*'. The code word formation for the 16 bits sub block merging is as shown in table 3.6.

Table 3.6 Unique Intra block merging at 16 bits level

Case	Inter/ intra	Group_code	UIMP	Tail bits		Code word
		16 bit merging		Code	Inference	
1	0	00	0	00	Unique sub block followed by *all zeroes (U0)*	0_00_0_00_unique sub block (16 bit)
2				01	Unique sub block followed by *all ones(U1)*	0_00_0_01_unique sub block (16 bit)
3				10	*all zeroes* followed by Unique sub block *(0U)*	0_00_0_10_ unique sub block (16 bit)
4				11	*all ones* followed by Unique sub block *(1U)*	0_00_0_11_ unique sub block (16 bit)
5	0	00	1	00	*all ones* followed by *all zeroes (10)*	0_00_1_00
6				01	*all zeroes* followed by *all ones* (01)	0_00_1_01
7				10	Compatible at sub block(16 bit level) UU	0_00_1_10_ unique sub block (16 bit)
8				11	Inverse Compatible at sub block(16 bit level) UU'	0_00_1_11_ unique sub block (16 bit)

However, if none of the above stated cases exists then, each 16 bitsHLB is further segmented into two sub blocks of length 8 bits each. Here,the 16 bits sub block becomes the parent block and examination for existence of compatibility between its children of 8 bits sub blocks is done. If any of the cases exist then encoding is done as per the table 3.6 corresponding to each pair of sub groups. The *group_code*is set to *'01'*to represent the merging being done at the sub block of lengths 8 bits each. In order to increase the compression further, the inter_intra block merging is also tried since the number of groups has been increased to 4. The existence of such case is represented by the setting *inter_intra_prefix bit* equal to '*1*'. If the sub blocks are found to be non-compatibility (unique) then they are retained as such in the encoding stream with an*inter_intra_prefix bit* set to '*0'*. For example: if three (consecutively occurring) sub blocks are found to be compatible then encoding is done by preceding the codeword by a '*1*' (*inter_intra_prefix bit)* while the leftover unique sub block is preceded by *inter_intra_prefix bit* set to '*0'* (as shown in table 3.7).

Table 3.7 inter/ intra block merging at 8 bits level

Case	Inter/ intra	Group_code	UIMP	Tail bits		Code word
		8 bit merging		Code	Inference	
1	0	01	0	00	Unique sub block followed by *all zeroes (U0)*	0_00_0_00_unique (sub block)
2				01	Unique sub block followed by *all ones(U1)*	0_00_0_01_unique(sub block)
3				10	*all zeroes* followed by Unique sub block *(0U)*	0_00_0_10_unique(sub block)
4				11	*all ones* followed by Unique sub block *(1U)*	0_00_0_11_unique(sub block)
5	0	01	1	00	*all ones* followed by *all zeroes*	0_00_1_001
6				01	*all zeroes* followed by *all ones*	0_00_1_010
7				10	Compatible at sub block(8 bit level) UU	0_00_1_10_unique (sub block)

In case, none of the above cases turn to be true then, the 8 bits sub blocks are further divided to get the 4 bits sub blocks. In this case each 8 bit block becomes the parent for its underlying 4 bits sub blocks. The four bit blocks so formed are further investigated for existence of various compatibility cases as described for 8 bits sub blocks. If any of the cases is found to exist then the encoding is done as per what is shown in the table 3.8, with *group_code*being set to *'10'*.To avoid the chances of expansion in place of compression; two restrictions are imposed while selecting the unique_intra block merging which are: a) the number of compatible cases is even in number and b) not more than 2 unique pairs are left.

Table3.8 Inter/ intra block merging at 4 bits level

Case	Inter/ intra	Group_code	UIMP	Tail bits		Code-word
		4 bit groups		Code	Inference	
1	0	10	0	00	Unique sub block followed by *all zeroes (U0)*	0_11_0_00_ unique (sub block)
				01	Unique sub block followed by *all ones(U1)*	0_11_0_01_ unique (sub block)
				10	*all zeroes* followed by Unique sub block *(0U)*	0_11_0_10_ unique (sub block)
				11	*all ones* followed by Unique sub block *(1U)*	0_11_0_11_ unique (sub block)
2	0	10	1	00	*all ones* followed by *all zeroes (10)*	0_11_1_001
				01	*all zeroes* followed by *all ones (01)*	0_11_1_010
				10	Compatible at sub block (4 bit level UU)	0_11_1_10_ unique (sub block)
				11	Inverse Compatible at sub block (4 bit level UU')	0_11_1_11_ unique (sub block)

In case, all the above discussed cases fail then the unique 32 bits long *pattern_code* is placed in the encoded data stream alongwith the necessary four bits prefix and tail bits (as shown in Table 3.9.)

Table 3.9 Encoding of 32 bits unique block

Case	Inter/ intra	Group_code		UIMP	Tail bits		Code word
		Code	Inference		Code	Inference	
1	0	11	32 bit merging	1	Not required.	Unique	0111 +32 bit unique pattern code

3.3 Development of Wrapper Design for Fine Grained Partitioned 3D System on Chip

The test wrapper design works as an interface between the core under test and the Test access mechanism attached to it. The objective of a wrapper design algorithm for a 2D IP core is to design all its wrapper chains such that they are balanced in terms of length and resulting test time of the core can be minimized. Balancing of the wrapper chains is necessary to reduce the difference between the maximum and minimum scan lengths.

For a 2D IP core, the problem can be stated as: Given a core having its functional detail as follows: number of inputs I_{core}, number of outputs O_{core}, number of bidirectional pins

B_{core},number of scan chains SC_{core} with their respective lengths $L.SC_{core}(i)$ where $1 \le i \le SC_{core}$, test vectors TV_{core} and TAM bus width TAM_{core}, determine the optimal placement of core elements into wrapper chains such that the length of the longest wrapper chain is minimized. The testing time of an individual core *j* for available TAM width can be calculated as per the eqns. 3.1-3.3.

Test time: $T_{core}(j) = (1 + \max(SI_{core}(j), SO_{core}(j)))\, TV_{core}(j) + Min(SI_{core}(j), SO_{core}(j))$ (3.1)

Where $SI_{core}(j)$ and $SO_{core}(j)$)represents the scan-in and scan-out lengths of the longest wrapper*k* chains of core *j* Their values can be extracted as follows:

$$SI_{core}(j) = I_{core}(j,k) + B_{core}(j,k) + \sum_{i=1}^{sc(j,k)} L.SC_{core}(j,k,i) \quad (3.2)$$

$$SO_{core}(j) = O_{core}(j,k) + B_{core}(j,k) + \sum_{i=1}^{sc(j,k)} L.SC_{core}(j,k,i) \quad (3.3)$$

Since, the test wrapper is being designed for an individual core so, for here on $j=1$ and is neglected from core parameters. Similar problem when adapted to the 3D partitioned cores with all its functional elements spread over multiple dies/layers need to minimize the number of TSV used per wrapper chain in addition to the constraints set by length. For instance, the available input/ output cells (each consisting of 1 flip flop length) and the start and end of the various internal scan chains can lie on any die on the 3D stack. The optimization problem for a circuit partitioned 3D SoC can be described as follows:

3.3.1 Problem formulation of fine grain partitioned test wrapper design and its proposed solution

The objective of an optimized test wrapper design for 3D core can be described as follows: Given a 3D core with its functional details (available TAM width C_{tam}, number of dies N_{dies} and maximum allowable TSV$C.TSV_{max}$), determine its test wrapper design such that the test time of the core can be optimized while keeping the total $C.TSV_{used}$ for all the Wrapper chains to be bounded by the TSV limit. As per the wrapper design [34], the insertion of the functional elements in each wrapper chains needs to be done in following order: input cells, internal scan chains and then output cells.

The core having number of functional input *n*, functional output *m* and internal scan chains *SC*. Placement of Individual components on various dies ranging between 1 and N_{dies} can be described as: $I.Core_i$ signifies the die number of functional input cell *i* where $1 \le i \le n$; . $O.Core_i$signifies the die number of functional output cell *I* where $1 \le i \le m$; $I_SC.Core_i$ signifies the die number of input cells of the scan chain *i*, $O_SC.Core_i$signifies the die number of output cells of the scan chain *i*. Lengths of various functional elements can be described as: $Len_I.Core_i$and $Len_O.Core_i$represents the lengths of the functional input and output

cellsrespectively which are set to one since, both of them are composed of only one flip flop. The length *Len_SC.Core$_i$* of the scan chains i can have any value based on the core's details. The proposed algorithm performs a wrapper chains design of the 3D core such that:

- The length of the longest wrapper chain WC is minimized. This is done to minimize the test time of the core (as calculated by eq.3.1).
- The number of internal wrapper chains WC is equal to the external TAM i.e :
  $\sum_{j=1}^{WC_{total}} WC(j) = C_{tam}$ (3.4)
- The total TSV_{used} should be less than or at max equal to the *C.TSVmax*. i.e.:
  $\sum_{j=1}^{WC_{total}} TSV_{WC}(j) \leq \quad C.TSVmax$ (3.5)

Total number of TSVs used by any wrapper chain j (as given in eq.3.5) can be calculated by adding the TSVs used in connecting the functional elements inserted in it i.e

$TSVwc(j) = S.TSV(j) + I.TSV(j) + O.TSV(j)$ (3.6)

where, $S.TSV(j)$, $I.TSV(j)$and$O.TSV(j)$ represents the *TSVs* used in insertion of the internal scan chains, core input and output elements respectively. The calculations of each of these can be done using the eqns. 3.7-3.9

Theorem 3.1 The Minimum number of TSVs, $S.TSV(j)$ required to implement a wrapper scan chain *j* after insertion of the scan chains is given by

$$S.TSV(j) = \sum_{\mathrm{s}=1}^{\mathrm{sc(j)}} |(I.SC \quad (j, s+1) \quad - O.SC(\mathrm{j,s}) \quad)| \qquad (3.7)$$

where$SC(j)$denotes the number of scan chains inserted in test wrapper chain *j*.

Proof: Case 1: When consecutive scan chains of same wrapper chain *j* reside on same die then no TSV would be required to insert them and hence $S.TSV(j)$=0.

Case 2: If the scan chains lie on different dies, or there is a difference between the output and input scan cell of the scan chains being connected then, TSVs number will increased. Herein, $S.TSV(j)$ will be equal to the difference between the die numbers of the output and input cells of the consecutively scan chains. It may be noted that while doing the calculation of $S.TSV(j)$, the TSVs used as a part of the scan chain formation are not considered.

Theorem 3.2: The value of the minimum number of TSVs $I.TSV(j)$ needed to insert the functional inputs *n(j)* in the wrapper chain *j* is given by :

$$I.TSV(j) = |I(j,1) - (N_{die=0})| + \sum_{i=1}^{n-1(j)} |I(j,i+1) - I(j,i)| \quad) + |I.SC \quad (j,1) - I(j,1)| \qquad (3.8)$$

Proof: Case 1: If all the functional input cells and the input scan cell of the first internal scan chain inserted in the wrapper chain *j* all lie on the same die; then, the only $I.TSV(j)$ needed

would be the ones that connects the first input cell to the bottom layer. This is reflected by taking N_{die}=0 in the first part of the eq. 3.8.

Case 2: If there is a difference between the die numbers on which the consecutively occurring input cells of wrapper chain $j$liethen, the difference between their die numbers contribute to the overall $I.TSV(j)$. Same is reflected by the second part of eq. 3.8.

Case 3: Finally, the difference between the last functional input cells and the input scan cell of the first internal scan chain inserted in the wrapper chain j need to be added (as shown in part three of eq. 3.8) to calculate the $I.TSV(j)$.

Theorem 3.3: The Minimum number of TSVs, $O.TSV(j)$. required to insert the functional outputs m in the wrapper chain i is given by:

$$O.TSV(j) = |O(j,m) - (N_{die} = 0)| + \sum_{i=1}^{m-1(j)} |O(j,i+1) - O(j,i)| \;)\; |O.SC \;\; (j,s) - O(j,1)| \qquad (3.9)$$

Proof:*Case 1:* If all the functional output cells $O(j,m)$ and the output scan cell of the last internal scan chain s inserted in the wrapper chain j all lie on the same die; then, the only $O.TSV(j)$ needed would be the ones that connects the last output scan cell $O(j,m)$ to the bottom layer. This is reflected by taking N_{die}=0 in the first part of the eq. 3.9.

Case 2: If there is a difference between the placement of consecutive output cells inserted in wrapper chain j then, the difference between their die number contribute to the overall $O.TSV(j)$. Same is described by the second part of eq. 3.9.

Case 3: The difference between the last functional output cells and the output scan cell of the last internal scan chain inserted in the wrapper chain j need to be added (as shown in part three of eq. 3.9) to calculate the $O.TSV(j)$.

The calculation of the $TSVwc(j)$ can become more by considering an illustrative example shown in figure 3.8. A hypotheticalmodel of the 3D core consisting of 7 functional inputs, 6 functional outputs, 6 scan chains (with length equal to 8,6,5,5,5 and 4 scan cells respectively) is given in figure 3.8 a). Out of the various internal scan chains: 3 scan chains lie on same die while rest 3 span different dies (i.e. die number of their input and output cells are different).

figure 3.8 b) represents a possible style of insertion of these elements in the wrapper chains. In this example, the number of wrapper chains or TAM wires is taken to be three. As can be seen in this case the lengths of different wrapper chains have been balanced out to minimize difference between the minimum and maximum scan lengths (as per eq. 3.1). Meanwhile, the TSV utilization is minimized by inserting the components lying on one layer to same wrapper chain. The $TSVwc$as calculated using eq.3.5 is 8 for all the wrapper chains. It may be noted

that TSVs that cater to the vertical interconnect between the scan chains which span on different layers have not been included in the total TSV utilization since they are a part of the core's functional design itself.As can be observed from the example discussed, the scan chains are the integral component in determining the length of the wrapper chains. Therefore, it becomes necessary to insert them first in the wrapper chains. To balance the lengths of the wrapper chains, appropriate scan chains are selected such the lengths and $TSVwc$ are not exceeded.

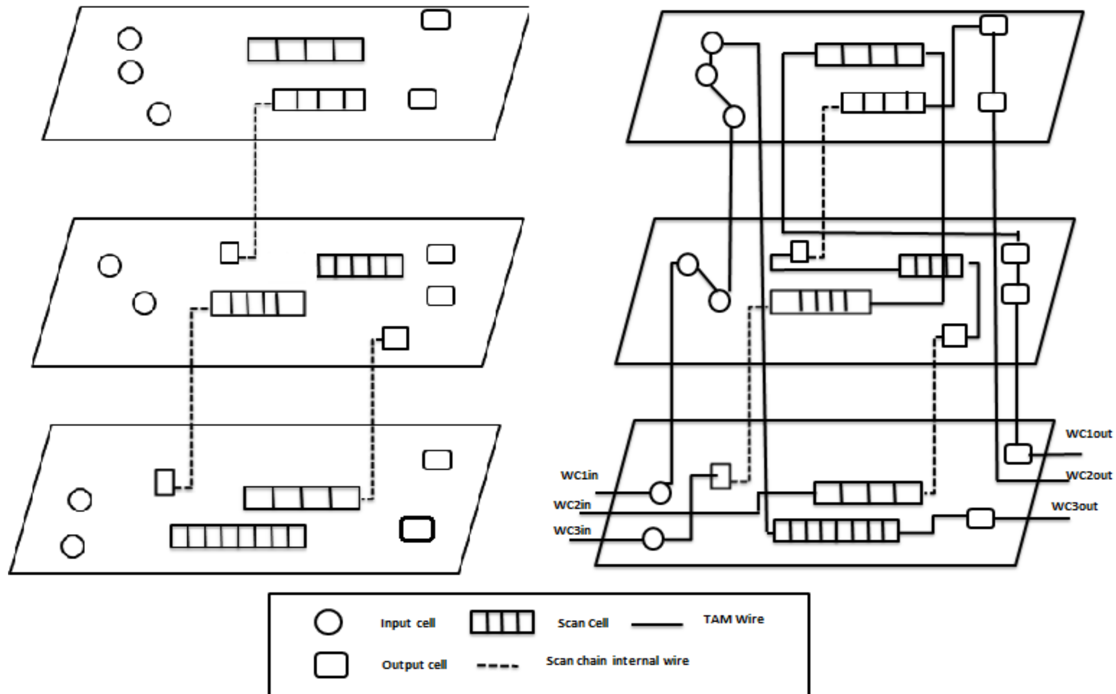


Figure 3.8(a) 3D model of a circuit partitioned IP core b) possible formation of the wrapper chain

The complete task of insertion of the scan chains and functional input/output elements can be described as follows:

3.3.1 aScan Chain insertion and reordering

This step is very important as it helps in defining the approximate length of the various wrapper chains. Each scan chain *SC(In, Type, Len, Flag, Out)* is associated with four parameters: *In (*Input die), *Type* (for scan chains its value is equal to 3), *len (*to signify thelength of scan chain), *flag* (set equal to *1* if both input and output dies are same else *0),Out (*output die number). The pseudo code for the scan chain insertion is as shown in the figure3.9. It initializes by arranging the internal scan chains in descending order of their lengths (shown in step 1). All the wrapper chains parameters like TSV_{usage}, scan chains in all the wrapper chains (as shown in step 2), die number of the input and output cells are initialized to be zero. The algorithm starts with an initial solution (as shown in step 3) by

inserting the scan chains in the wrapper chains. The list of the scan chains is updated by deleting the scan chains which have been inserted in the wrapper chains. While scan chains are still left in the list *SC*, further insertion in wrapper chains is done such that the length is optimized along with usage of minimum number of TSV usage.

Pseudo code for scan chain insertion	**Pseudo code for selecting the scan chain for insertion**
Given Total scan chains *SC* and TAM wires *Ctam* **Step 1: /*initialization of scan chain list** **For (*i* =0; *i*<*SC*, *i* ++){** • **Sort all the Scan chains in decreasing order of their length** • **Initialize the Scan chain with their appropriate flag value;** **}** **Step2: /*Initialize different wrapper chains** **For (i=1 to *Ctam*) {** ***WC.length (i)= WC.TSVused(i)= WC.start(i)= WC.end(i)=0;*** **}** **Step3 /* Insert scan chain** **For (i=0; i<*Ctam*; i++){** **Insert *SC* (0) to *WC* (i)** **Update *WC.length (i)*, *WC.start (i)*, *WC.end (i)*;** **Delete *SC* (0) from *SC* list;** **}** **Step 4: /*while (*SC*! =0) {** **Find the *WC* with *Smax*** **Find *WC* with *Smin*** ***ch_SC* =Search (*SC*, *WC* [*Ctam*], *Smax*, *Smin*)** **Insert *ch_SC* to *WC* and update all parameters of *WC*;** **Delete *ch_SC* from *SC*** **}** **Step 4: /* reordering of the scan chains in wrapper chains** **For (i=1 to Wmax){** **Reorder all the *Scan* chains so as to minimize *WC.TSVused*;** **Update the *WC* parameters;** **}**	**//Find a *Scan* chain such that its insertion doesn't not increase the length of the wrapper chain and TSV used by large extent** **Function Search (*SC*, *WC*[*Ctam*], *Smax*, *Smin*){** **If (length (*Smax*- (len.*SC*+*Smin*) !>*Smax*)){** **If (flag==1){** **If((O.*WC*min=I.*SC*|| I.*WC*min==0.*SC*)** **Return *SC*;** **Else if ((*O.WC*min! =I.*SC* && I. *WC*min==0.*SC*){** **Find *SC* with min ((*O.WC*min-I.S), (I. *WC*min-0.*SC*))** **Return *SC*;** **}** **Else {** ***SC* with minimum length** **Return *SC*** **}** **}**

Figure 3.9 Pseudo code for scan chain insertion

3.3.1.bFormation of Di Graphs and scan chain insertions

For the objective of the TSV minimization scan chains are placed in different wrapper chains such that the lengths of the wrapper chains are more or less balanced. This makes a heterogeneous mix of scan chains with same or different start and end layers. This is done by insertion of all the scan chains with flag bit equal to 1 in various wrapper chains followed by others as per lengths. The TSV calculation is done as per given in the theorems 3.1 to 3.3. Once the scan chains are insertion is decided, in between the wrapper chains reordering is done. For this the Di graphs are prepared which keep track of the TSV used in traversing between different scan chains as shown in figure 3.10. The scan chains are shown as nodes

with different parameters and the edges showing the possible connectivity between them. The numbers written adjacent to each edge reflects the number of TSVs used if one traverses from starting node of the edge to ending node.

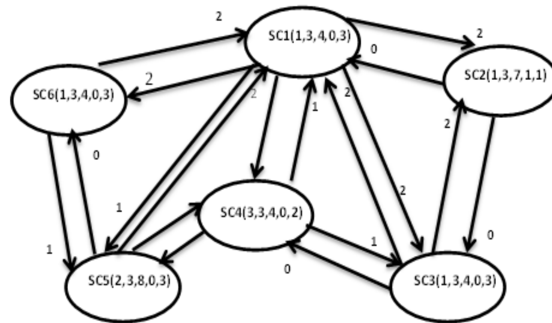


Figure 3.10 Di-graph for various scan chains

With 3 different layers, each element can have 1 out of 2^3 different possible combinations corresponding to their starting and ending layer. Initially, all the elements are sorted, counted for flag=*1* and then distributed between the available number of wrapper chains. Thereafter similarly the scan chains with flag=*0* are sorted and counted as per their *I_SC.Core* and *O_SC.Core*. Distribution into different bins is done so as to obtain a heterogeneous mix of the scan chains in the different wrapper chains.

For Individual wrapper chains construct the diagraphs as shown in figure 3.11 with layers as the nodes and scan chains as the edges. As shown there are edges between 1 to 3 for scan chains 1,3,6, and between 3 to 1 for Scan chain 8 etc. The connectivity of the scan chains should be done such that if a scan chain takes the wrapper chain from Node 1 to 2nd then the next scan chain should take it either back from layer 2 to 1 or from 2 to 3.

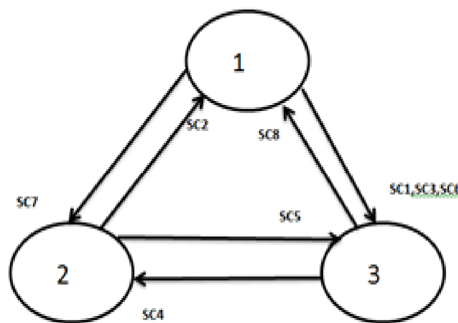


Figure 3.11 Diagraph to represent the scan chain and layer connectivity

Thereafter the insertion is done in the successive wrapper chains on the basis of the route with minimum TSV utilization in an order as shown in figure 3.12

Figure 3.12Possible stitching of the scan chains

Insertion of the functional input and output cells is done as described with the help of the pseudo algorithm shown in figure 3.13(a) and (b).

Pseudo code for input cell insertion	**Pseudo code for output cell insertion**
// for adding functional input cells to the wrapper chains created in Part I **For all input cells {** **For (*i*=0; *i*< *Ctam*; *i*++) {** • **For each input cell, calculate** $O.TSV(j)$ **using equation 3.8** • **Insert input cells to *WC (i)* such that** $O.TSV(j)$ **is minimum and *Smax* is not exceeded.** • **Delete the used Output cells** **}** ***WC.TSVused*** **+ =** $I.TSV(j) + \|I(j,1) - number\ of\ the\ bottom\ die\ \|$ **Update length of each wrapper chain.** **}**	**// for adding functional outputs to the wrapper chains created in Part I** **For all output cells {** **For (*i*=0; *i*< *Ctam*; *i*++) {** • **For output cells, calculate** $O.TSV(j)$ **using equation 3.9** • **Insert output cells to *WC (i)* such that** $O.TSV(j)$ **is minimum and *Smax* is not exceeded.** • **Delete the used Output cells** **}** ***WC.TSVused*** **(i) + =** $O.TSV(j) + \|O(j,1) - number\ of\ the\ bottom\ die\ \|$ **Update length of each wrapper chain.** **}**

Figure 3.13 Pseudo code for a) input scan cell insertion and b) output scan cell insertion

Initially only those input and output elements are inserted that lie at the same layer as the starting layer of first scan chain and the output layer of the last scan chain of the individual wrapper chains respectively. Remaining I/O cells are tried to be added to the wrapper chains such that their lengths are balanced and TSV_{usage} doesn't exceed the maximum allowable limit. The benefit of starting the wrapper chains with different layers provide the advantage of adding as many input cells as possible that lie on the same layer.

3.4 Test architecture development for Coarse grain partitioned 3D SoC

Stacking of multiple 2D dies with necessary logicis done to realize 3D SoCs. To ensure the quality of the integrated circuits, the 3D SoCs have to be tested hierarchically at various stages like pre bond (after wafer thinning operation), partial bond (after intermediate stacking level) post bond (after full stacking) and before assembly and packaging. The conceptual view of the 3D test which allows testing at various design levels is shown in figure3.14.

While pre bond testing of the individual 2D dies can be done by reusing its DFT infrastructure with the help of the direct probe pads; test elevators are needed to transport the test stimuli and test response between the ATE and design under test. DFT infrastructure present on the 2D die can support the test of both the cores and their associated interconnects. Tested and fault free individual dies when bonded together using various micro bumps etc. should be tested at partial level before formation of complete stack to assure early detection of faults and hence an enhanced yield. For performing the post bond (partial and global

bonding) tests, the test data vectors and the test instructions need to be carried from the bottom die to core under test and back by using the TSVs that act as test elevators. 3D test controller feeds the wrapper around the dies which in turn feeds the wrappers around the individual IP cores lying on the various dies[47]. The same is mentioned in figure 1.12

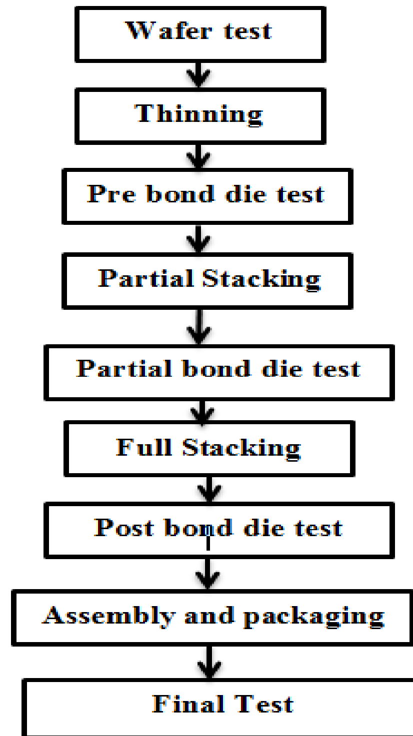


Figure 3.14 Test architecture of 3 D SoC

The test solution developed should ensure the optimization of test cost while addressing all the constraints as described in chapter 1. The key contributions of the proposed technique are as follows:

- Optimization of the test architecture of pre bond 2D dies such that for a given design details and their associated constraints, its test time can be minimized.
- Optimization of the 3D test architecture such that test time of the entire 3D SoC can be minimized while keeping the TSV usage below the maximum global limit.
- Investigation of the multiple test insertion to perform the partial and full stack level testing.

3.4.1 Problem formulation of 3D SoC testing

The objective of an effective test solution development for a 3D system on chip is to minimize the test cost by reduction of the test application time such that various constraints like test bandwidth, number of stacked dies, maximum TSV limit, maximum test power limit

and various precedence constraints are not violated. Such an optimization goal can be achieved by sequentially optimizing the test architectures of: individual dies followed by the complete 3D SoC.

To perform the testing of the 3D SoC, the DFT infrastructure available on 2D dies can be reused to transport and apply the test data. Therefore, for optimization of the 3D SoC test, it becomes essential to have an optimized DFT architecture for ensuring the testability of 2D dies. As discussed in chapter 1, the DFT architecture for 2D dies includes: designing of test wrappers andTAM architecture, assignment of the cores to the TAM and determination of the test schedules of each core.Based on the flexibility available to the test engineers for test infrastructure design; the test architectures of individual dies can be optimized to reduce test cost. Once the die level test architecture is finalized for any given TAM width, the 3D TAM designing can be done such that TSV usage and global testing time can be reduced. The problem of the 3D test can be categorized on the basis of the flexibility provided to the test engineer at the inter die and intra die level as shown in the figure 3.15. If test architecture between the dies and within the dies i.e. at inter and intra die level is fixed (as shown in the left half of the figure 3.15), then the test engineer has to simply form the 3D test architecture involving them. In such a case, the 3D SoC test integrator is relieved of the task of the architecture design at 2D SoC level. However, such a case limits the flexibility of adapting the architecture for test cost optimization.

On the other side, if the inter and intra TAM architectures are flexible then DFT infrastructure can be much better optimized. Based on the structural composition details of the various cores, test wrappers of individual cores need to be optimized followed by a 2D TAM designing. The same is shown on the right half of the fig 3.15. Once 2D TAMs are designed,3D architecture and its wrapper design which assumes the individual dies as the child components (as described in section 1.3) can be done to ensure the test of the complete stack.

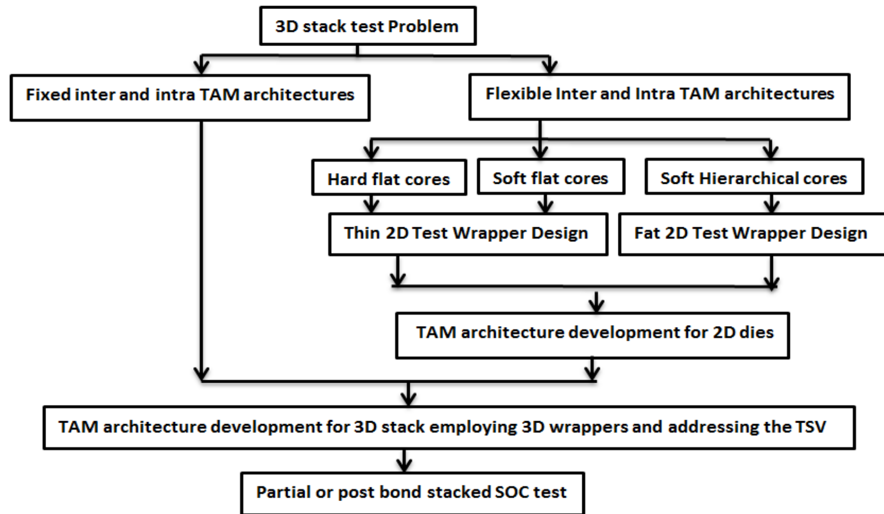


Figure 3.15Conceptual view of the test architecture problem of 3D SoC

Hence, the global problem of testing of 3D SoC can be solved by sequentially solving the following problems:

Test architecture design of 2D SoC P_{2TAW}: Given the details of the cores lying on individual dies and the maximum TAM width, determine its test architecture and wrapper designs such that its test time can be optimized. The problem P_{2TAW} can be further classified on the basis of the flexibility in the designing of TAM architecture and cores test wrappersbased on the flat vs. hierarchical structure of the core. Also, in some cases the internal scan chain of the cores can either be predesigned or need to be designed so as to optimize the test wrapper. Hence, 2D test solution development revolves around the optimization of test cost.

Test architecture design of 3D SoCP_{3TAW}: Given the number of stacked dies, structural details of each 2D die, maximum available TAM bandwidth and TSV limit, determine its test architecture such that the test cost of the 3D SoC can be reduced.The 3D TAM architecture can be categorized as fixed or flexible. If it is fixed, the TAM wires allotted to one die cannot cater to the test needs of cores on other dies whereas flexible TAM architecture allows TAM to traverse different dies. Flexible TAM can transport the test data to appropriate cores irrespective of their association with different dies.However, such an approach makes the test engineer responsible for the stacking the dies to do all the ground work related to test architecture development of all the cores lying on all the dies.

Mathematically, these can be described as follows: Given 3D stack with following details: The total TAM bandwidth N_{TAM}, number of stacked dies N_{dies}andmaximum TSV limit TSV_{max}; Similarly, the details of each die i comprises of maximum TAM bandwidth $N_{TAM}(i)$ andits total number of cores be $N_{core}(i)$ where $1 \leq i < N_{dies}$. For each individual *jth* core of *ith*die $N_{core}(i,j)$ given the details like: no of inputs $I_{core}(i,j)$, no of outputs $O_{core}(i,j)$, no of

bidirectional elements $B_{core}(i,j)$, no of scan chains $SC_{core}(i,j)$ with their lengths $L.SC_{core}(i,j,s)$, level of core (to represent the structural details like flat or hierarchical core) *l.core(i,j)* and number of test vectors $TV(i,j)$where $1 \leq j \leq N_{core}(i)$, $1 \leq i \leq N_{die}$ and $1 \leq s \leq SC_{core}$.

The first objective is to design a TAM architecture of a 2D die (as applicable for the problem P_{2TAW})such that its test time $T.Time_{die}$for a given test bandwidthN_{TAM}can be minimized. The value of the $T.Time_{die}$can be calculatedby adding the test time $T.N_{core}(j)$of each core *j*for a given a TAM width (N_{width})allotted to it. Eqns. 3.10 and 3.11 can be used to compute the test time $T.Time_{die}$of different dies for all the cores lying on them with test bus architecture to be of type test bus[86] or test rail[32]respectively. Meanwhile, the test time of individual cores can be calculated by either using the BFD algorithm (for flat cores) [21] or FAT wrapper algorithm [38] (for hierarchical cores). It is assumed that the total TAM width of 2D die *i* i.e. $N_{TAM}(i)$ be distributed among different partitions N_{bus}. Hence, the sum of the TAM widths of each partition is equal to the maximum available TAM bandwidth of that die as described by eq.3.12.

$$\text{Minimize } \max_{1 \leq k \leq Nbus(i)} \sum_{j=1}^{Ncores(i)} T.N_{core}\left(j, (N_{width}(k)).(x(i,j,k))\right) \quad (3.10)$$

$$\text{And Minimize } \max_{1 \leq k \leq Nbus(i)} \left(\sum_{k=1}^{N_{bus}(i)} \sum_{j=1}^{N_{core}(i)} \left(TV_{core}(i,j) - TV_{core}(i,j-1)\right) (i-1) \sum_{j=i}^{Ncore(i)} \max(SI(j), SO(j)) \right) \quad (3.11)$$

Where:

- $N_{width}(k)$ is the number of TAM wires allotted to test partition *k*. and
- $x(i,j,k) = x_{ij} * x_{ik} = \begin{cases} 1 & If\ the\ core\ j\ lying\ on\ die\ i\ is\ assigned\ to\ TAM\ partition\ k \\ 0 & otherwise \end{cases}$
- $\sum_{k=1}^{N_{bus}(i)} T_{width}(k) = N_{tam}(i)$ Where $1 \leq i \leq N_{die}$ (3.12)

Similarly, the multi objective test cost optimization problem for 3D SoC can be expressed as a problem of minimizing the total test time ($T.Time_{stack}$) and TSV utilization (TSV_{stack}) for test access mechanism. Both can be expressed as shown in eqns. 3.13 and 3.14.

$$\text{Minimize } \left(max\left(\sum_{i=1}^{N_{die}} T.Time_{die}(i)\right) \right) \quad (3.13)$$

$$\text{Minimize } \sum_{i=1}^{N_{die}} TSV_{used}(i) \quad (3.14)$$

At the same time, the solution should also address the constraints set by: a) Multiple test sets $TS_{core}(j,t)$ of a core *j* (if present) which should be exercised during same time slot *m* during which the test of that core is scheduled (as represented in eq. 3.15), b)the tests of various

cores lying on all dies $T.Time_{stack}$should be less than the maximum timeT_{max} which is taken to be maximum test time if all the cores on various dies are to be tested sequentially. Same is represented in eq. 3.16, c)Each core be scheduled only once and the end of its test time should lie within the total time as given in eq. 3.18, d) precedence among the tests of different cores which mandates the test of core *p* to be scheduled before the test of core *q*. It is expressed in eq. 3.19, and f)to avoid thermal runway, the power dissipated during the entire test should be less than the maximum limit as set by the SoC integrator. Same can be expressed as given in eq. 3.20.

- $\sum_{t=1}^{TS_{core}(j)} T.Time_{core}(j,t) \leq m$ (3.15)
- $T.Time_{stack} \leq T_max$ for all *i,j and k* (3.16)

  where

  $T.Time_{stack} = \sum_{i=1}^{N_{die}} \sum_{k=1}^{N_{bus}(i)} \sum_{j=1}^{N_{core}(i)} \sum_{t=1}^{TS_{core}(j)} T.Time_{core}(i,j,k,t), x(i,j,k)$

  (3.17)
- $\sum_{m=1}^{T_{total}} T.Time_{core}(j,m) = 1$ (3.18)
- $e_{core}(j=p) < b_{core}(j=q)$ where $cores\ p\ and\ q \in N_{core\ (i)}$(3.19)
- $\sum_{i=1}^{N_{die}} \sum_{i=1}^{N_{tam}(i)} \sum_{k=1}^{N_{bus}(i)} \sum_{j=1}^{N_{core}(i)} \sum_{t=1}^{TS_{core}(j)} Power_{core}(i,j,k,t) < P_{stack}$ (3.20)

Two practical scenarios of the 3D SoC test architecture design those have been addressed as a part of this research work are as follows:

Test solution development for 3D SoCs with flexible inter and intra TAM architectureP_{SS}: Given 3D stack with following details: The total TAM bandwidth N_{TAM}, number of stacked dies N_{dies}andmaximum TSV limit TSV_{max}; details of total number of cores $N_{core}(i)$ lying on each die *I* where $1 \leq i \leq N_{dies}$. For each individual *jth* core of *ith* die $N_{core}(i,j)$ given the details like: no of inputs $I_{core}(i,j)$, no of outputs $O_{core}(i,j)$, no of bidirectional elements $B_{core}(i,j)$, no of scan chains $SC_{core}(i,j)$ with their lengths $L.SC_{core}(i,j,s)$, level of core (to represent the structural details like flat or hierarchical core) *l.core(i,j)* and number of test vectors *TV(i,j)* and test power $P_{core}(i,j)$where $1 \leq j \leq N_{core}(i)$, $1 \leq i \leq N_{die}$ and $1 \leq s \leq SC_{core}$.. Determine: a) A test solution for 3D SoC such its testing time is reduced while constraints like TAM bandwidthN_{TAM}, maximum TSV limit $TSV_{max}$and test power P_{stack} are not violated. In this case, inter and intra die TAM is taken to be soft, therefore, no constraint regardingpermissible TSV limit between two successive dies is taken into consideration.

Test solution development for 3D SoCs with fixed inter and flexible intra TAM architectureP_{HS}: Given 3D stack with following details: The total TAM bandwidth N_{TAM},

number of stacked dies N_{dies}andmaximum TSV limit TSV_{max}; the details of each die i comprises of maximum TAM bandwidth $N_{TAM}(i)$ andits total number of cores be $N_{core}(i)$ where $1 \leq i \leq N_{dies}$; For each individual *jth* core of *ith* die $N_{core}(i,j)$ given the details like: no of inputs $I_{core}(i,j)$, no of outputs $O_{core}(i,j)$, no of bidirectional elements $B_{core}(i,j)$, no of scan chains $SC_{core}(i,j)$ with their lengths $L.SC_{core}(i,j,s)$, level of core (to represent the structural details like flat or hierarchical core) $l.core(i,j)$ and number of test vectors $TV(i,j)$where $1 \leq j \leq N_{core}(i)$, $1 \leq i \leq N_{die}$ and $1 \leq s \leq SC_{core}$.It may be noted that except the topmost die, each die needs TSV for providing the interconnectivity with the die on its top. Determine a) 2D TAM architecture such that its test time is minimized. b) A test solution for 3D SoC such that the test time and TSV usage is minimized while constraints like TAM bandwidth, maximum TSV limit TSV_{max} and maximum test power P_{stack} are not violated.

The problem of the fixed 3D and flexible 2D TAM architecturetest case will become clearer with help of the illustrative examples given in figure3.16. Assuming two hypothetical 3D SoC with two dies, TAM bandwidth N_{TAM} equal to 70 wires and maximum TSV limit TSV_{max} for die 2 being 50. As fig 3.16 (a) and (b), $N_{TAM}$is divided in two partitions (one for each die) with individual widths as$N_{TAM}(1)$= 20 and $N_{TAM}(2)$= 50. Since, die 2 has received TAM width equal to 20 so the TSV used will be 40 which is less than TSV_{max}. As shown in figure 3.16, the $N_{TAM}(1)$ and $N_{TAM}(2)$ are further partitioned into two buses at individual dies. Consequently, either test rail (as shown in figure 3.16 (a)) or test bus (as shown in figure 3.16 (b)) architecture can be used to optimize the TAM architecture at 2D die level.

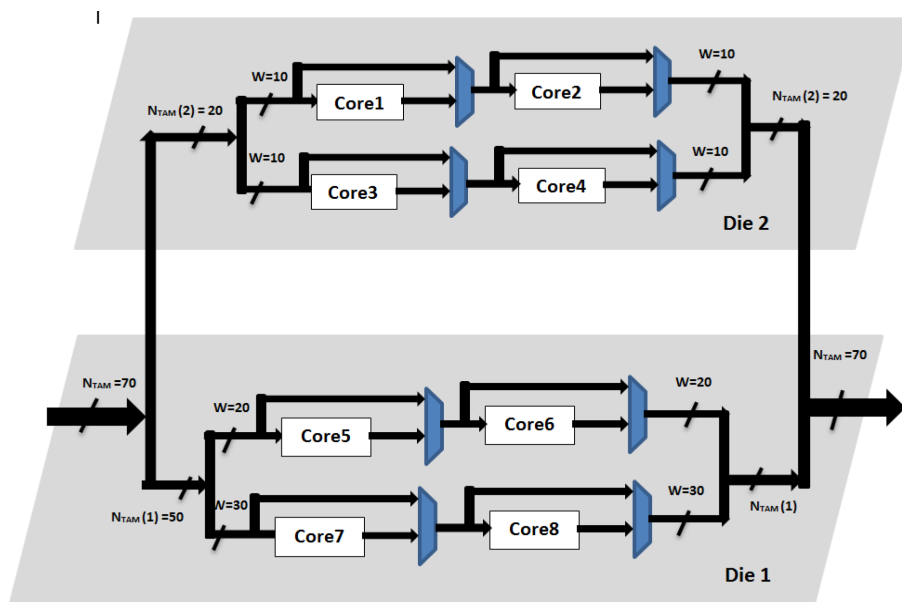


Figure 3.16 (a): Example of 3D SoCs with hard inter and flexible Intra die TAM test architecture with 2D dies supporting Test Rail TAM Architectures

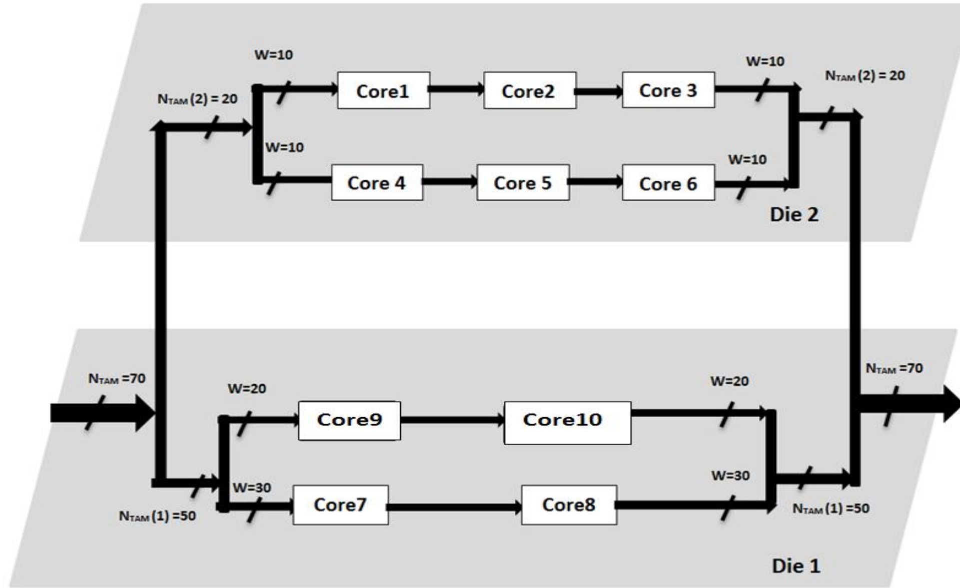


Figure 3.16: (b) Example of 3D SoCs with hard inter and flexible Intra die TAM test architecture with 2D dies supporting Test Bus TAM Architectures

3.4.2 Test Solution for 3D SoCs with fixed but soft inter and flexible intra TAM architectures.

The solutions to the problem $P_{HH}$of fixed but soft inter and flexible intra TAM architecture design can be developed by sequentially addressing the problem of test architecture development for 2D dies and stacked 3D SoC. 2D dies being composed of 2D cores need to have optimized test wrappers and TAM architecture so as to optimize its test time. Once a test solution for the individual die is made which gives optimized result for any value of TAM width, it can be further extended for 3D structure as well. For optimization of the wrapper design and the test time of each core $T.T_{core}(i,j)$, Best Fit Decreasing algorithm [21] and modified FAT wrapper designs[38] are opted for flat cores and hierarchical cores as per the description of the problems.

Initially, two sets are preparedto calculate the test time for different TAM widths. Out of these, set one includes the test times for various cores in correspondence to different TAM widths and set two containsthe sets of power compatible cores that can be scheduled together on the basis of their power consumption (This is done to ensure that the total power dissipation at any moment of testing to be below the global limits).

The first objective is to minimize the test time for all cores lying on individual dies. The TAM partitions allotted to individual dies can be assumed to be different test buses or test rails which serve TAM architecture present at the 2D die. Assuming the TAM allotted to the

die be divided in N_{BUS}number of partitions. Development of TAM architecture at 2D die level can employ Test Bus or Test Rail approaches. The designing TAM architectures at the 2D SoC can be done by following four steps sequentiallynamely: a)TAM architecture initialization b)TAM optimization stage 1 c)TAM optimization stage 2 d) TAM optimization stage 3 and e) TAM optimization for 3D stack. The description of each of these is as follows.

3.4.2.a TAM architecture initialization

Using this approach, a raw solution for the TAM partitioning and the assignment of the cores is done. Figure 3.17shows the flow chart for creating the initial solution for the 2D SoC available on the dies. Based on the available TAM bandwidth, the cores from the sorted list *S1* are assigned to the different TAM wires. TAM width assignment to each core is done on the basis of the comparison between the number of TAM wires and the number of cores N_{core}(i,j). If the number of TAM wires is less than the number of cores then they are allotted only to the cores with larger number of $TV_{\mathrm{core}}(i,j)$while, if the number of wires are larger than the number of cores then, the free wires are progressively added to the cores with larger test time. However, if the number of test wires and cores are same then the algorithm just finishes there itself.

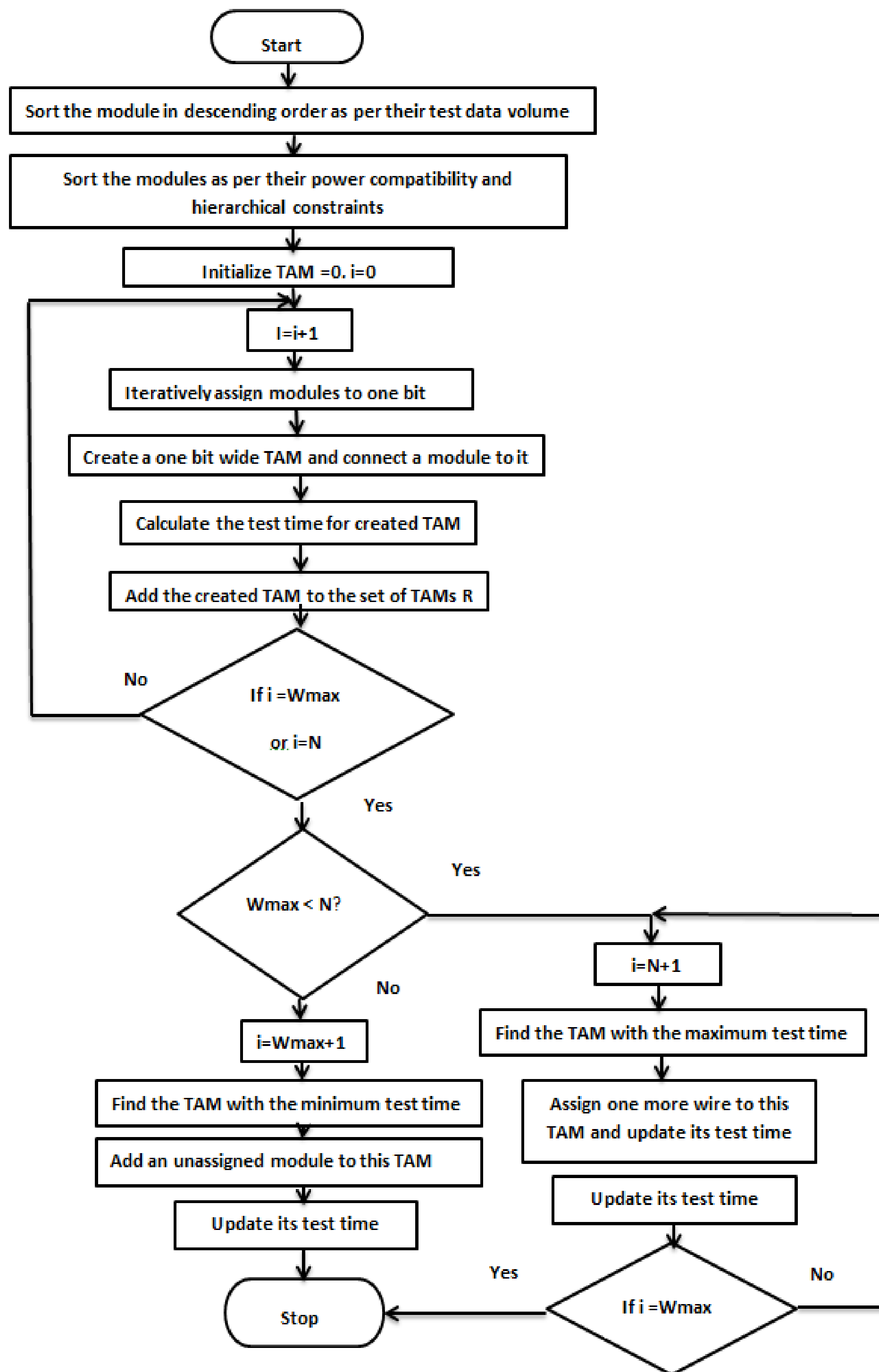


Figure 3.17 Flowchart for performingTAM architecture initialization

3.4.2.b TAM optimization stage 1

This step is performed to redistribute the TAM wires such that the TAMs taking minimum time are merged with other TAM wires to reduce the maximum test time among all TAM combinations. The flow chart of this step is as shown in the figure 3.18.

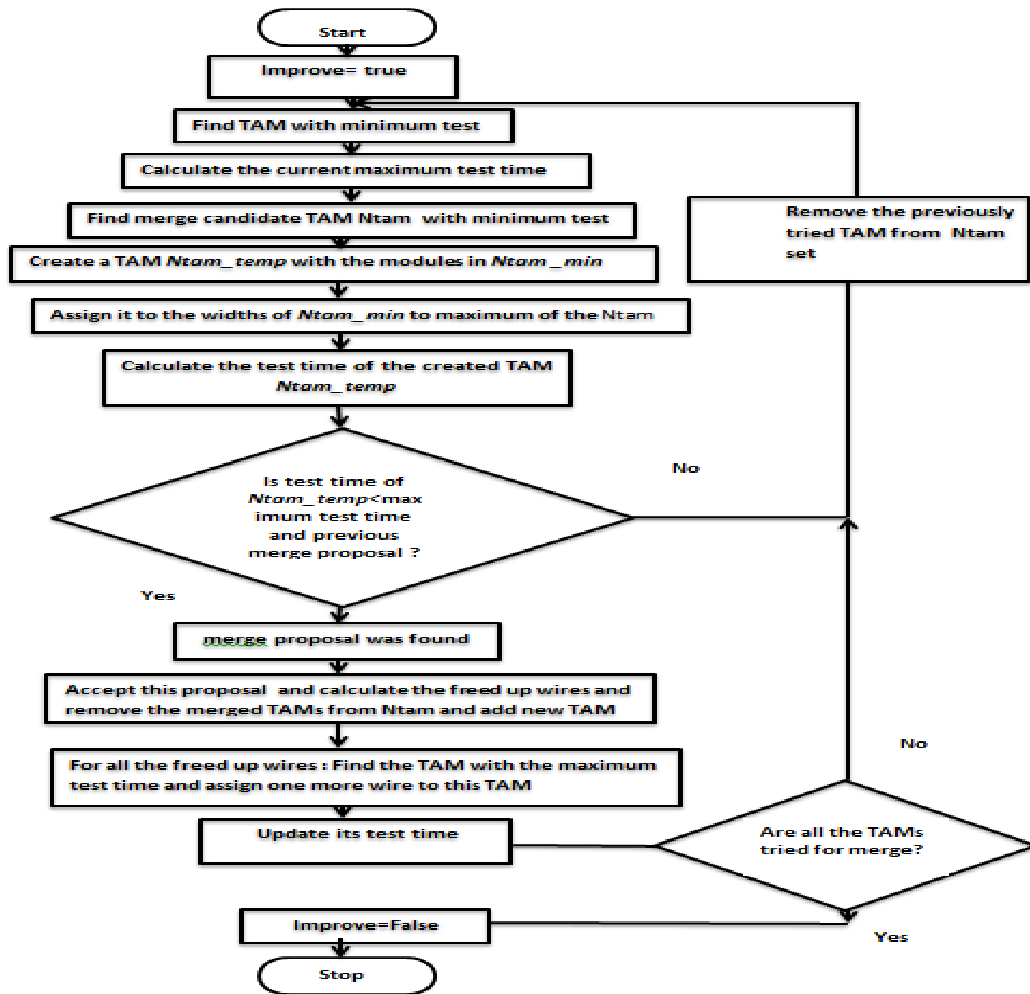


Figure 3.18 Flowchart for TAM optimization stage 1

The algorithm starts with the task of finding the TAM with least test time (TAM_{min}). Cores allotted to TAM with the minimum test time are merged with one of other TAMsTAM_{min_2}(having its test time less than current T_{max}) such that the resultant test time after merging doesn't exceed the T_{max}. Herein, the TAM allotted to the merged set of cores is chosen to be the maximum of the $TAM_{min}$and TAM_{min_2}.While the freed up wires are redistributed among the rest of the TAMs to minimize the overall test time. The search and corresponding redistribution of TAMs is done till no further improvement can be obtained.

3.4.2.c TAM optimization stage 2

This step (as shown in figure 3.19) further redistributes the TAMs such that the overall test time can be reduced. This is done by merging the TAMs with longest test time with other TAMs such that the resultant test time can be minimized. Hence, this step tries to reduce the critical value (i.e.longest test time T_{max}). If a change in T_{max}value is observed then such a proposal is accepted else, the TAM (TAM_{max})with maximum test time $T_{max}$is removed from

the list of subsequent investigation. The algorithm further tries to merge the TAMs left after removal of the $TAM_{max}$with some other TAMs in an iterative fashion such that that the merged width is still small and capable of freeing up some test wires. For instance, a TAM *t* with width *width(t)* that lies between the range of max (*width*(TAM_{min}), *width(t)*) and min (*width*(TAM_{min})+ *width(t)*) is selected such that its resultant test time is minimum. The freed up wires are redistributed among others TAMs to reduce the overall test time.

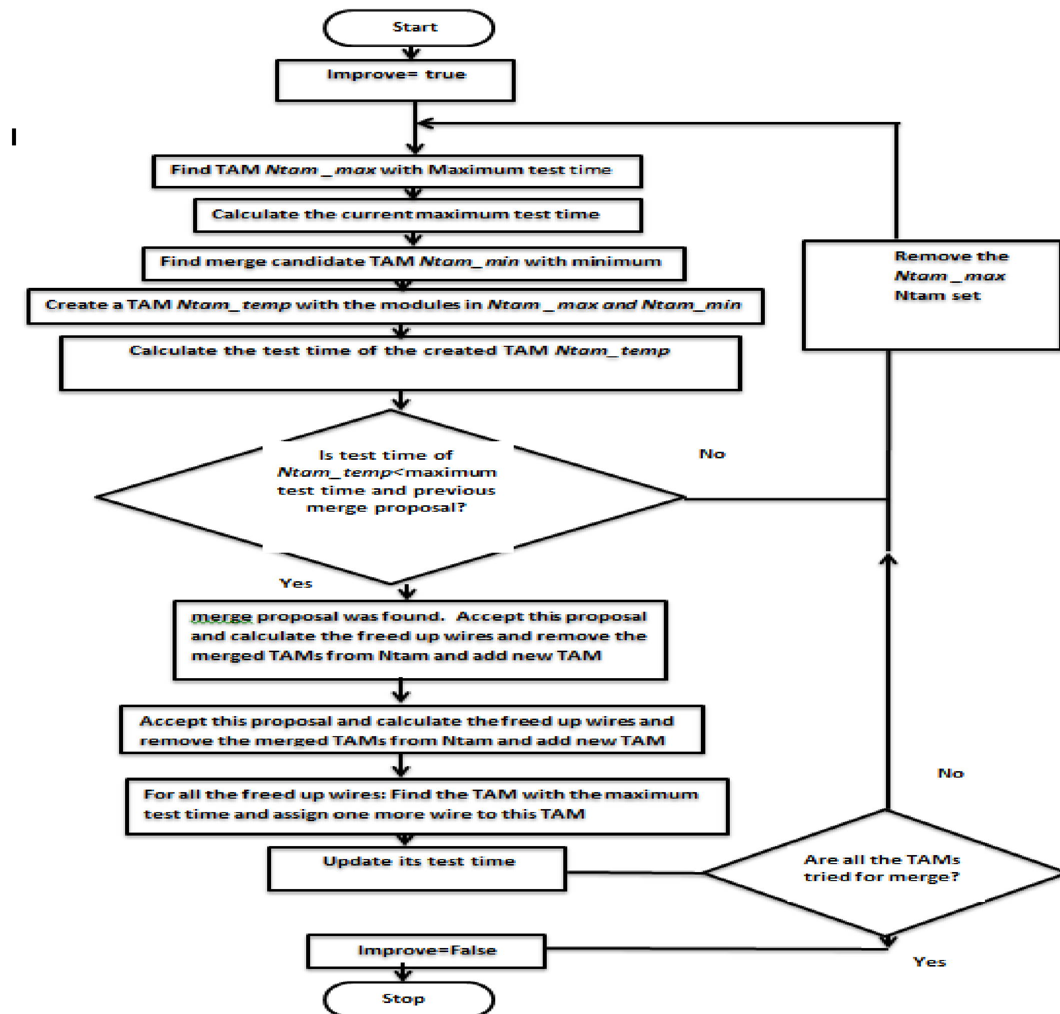


Figure 3.19 Flowchart for TAM optimization stage 2

3.4.2.d TAM optimization stage 3

This step is performed to redistribute the bottleneck cores from the TAMs taking maximum time with the ones having lesser time. The redistribution and rearrangement of TAM wires is done until the resultant Testing time is as optimized as possible. Figure 3.20 shows the flowchart to perform the same. The algorithm tries to find out the critical TAMs. The cores with smallest test time are investigated for possible insertion into the other TAMs such that

the test time doesn't exceed the current max. This way it reduces the critical test time alongwithbalancing the core assignment among various TAMs.

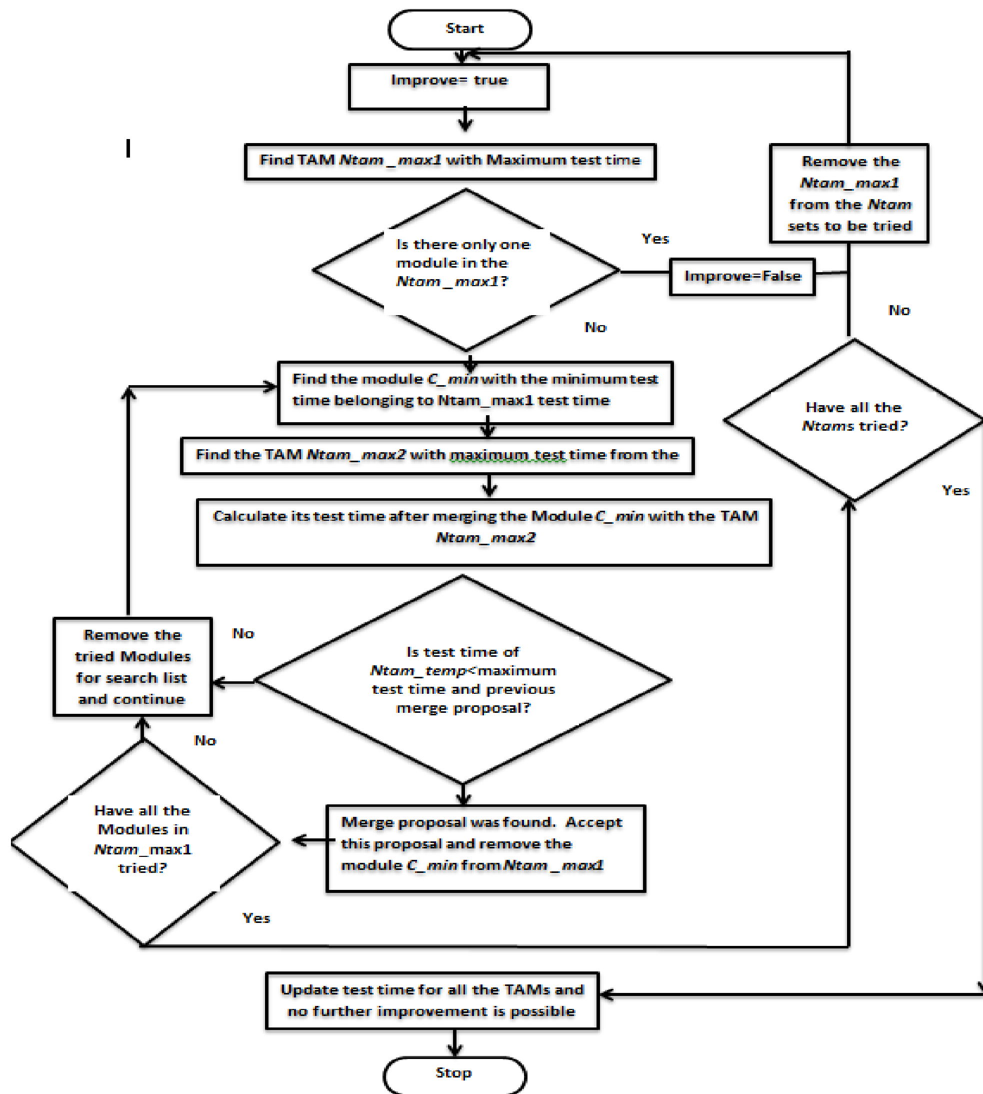
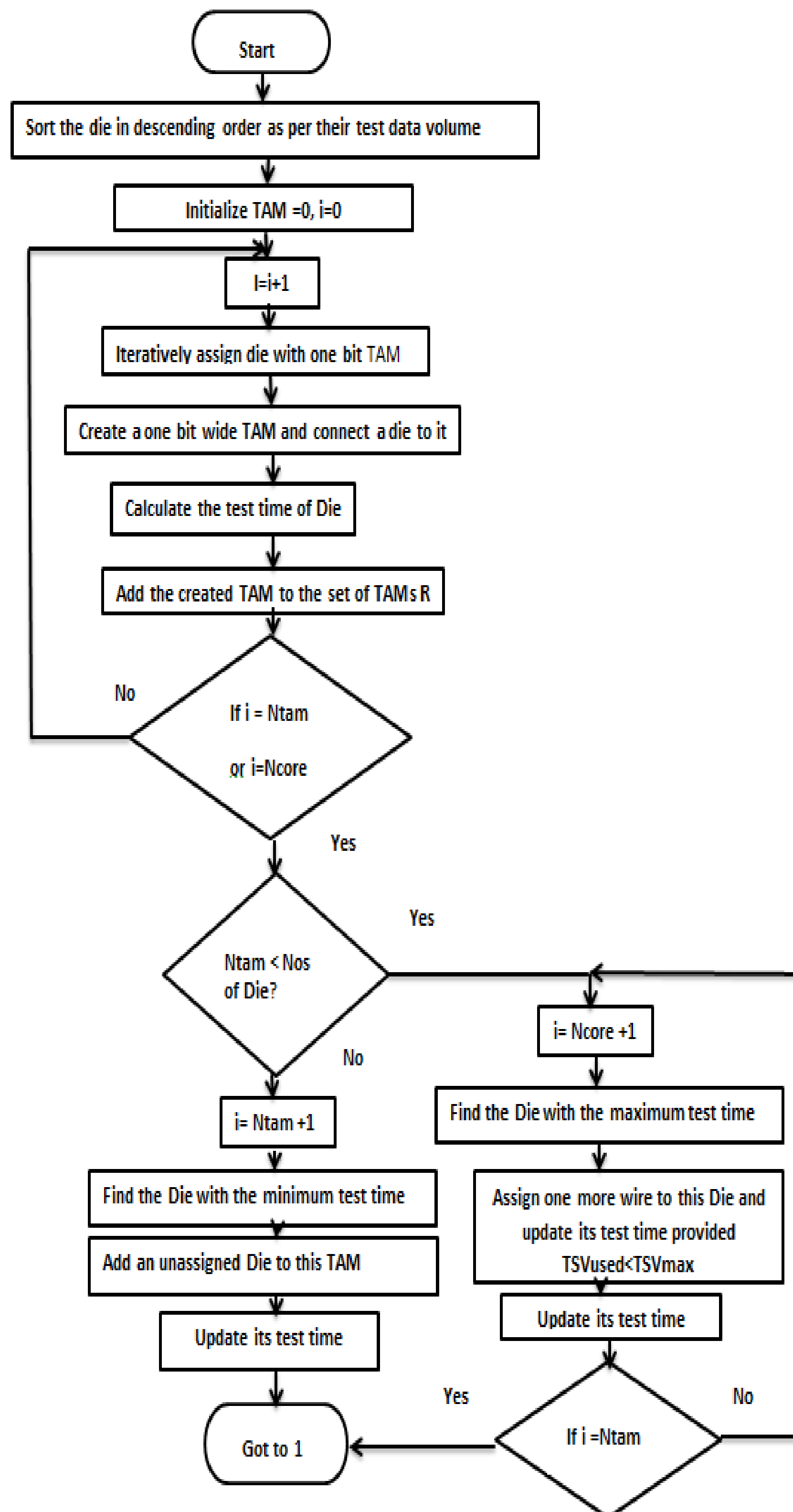


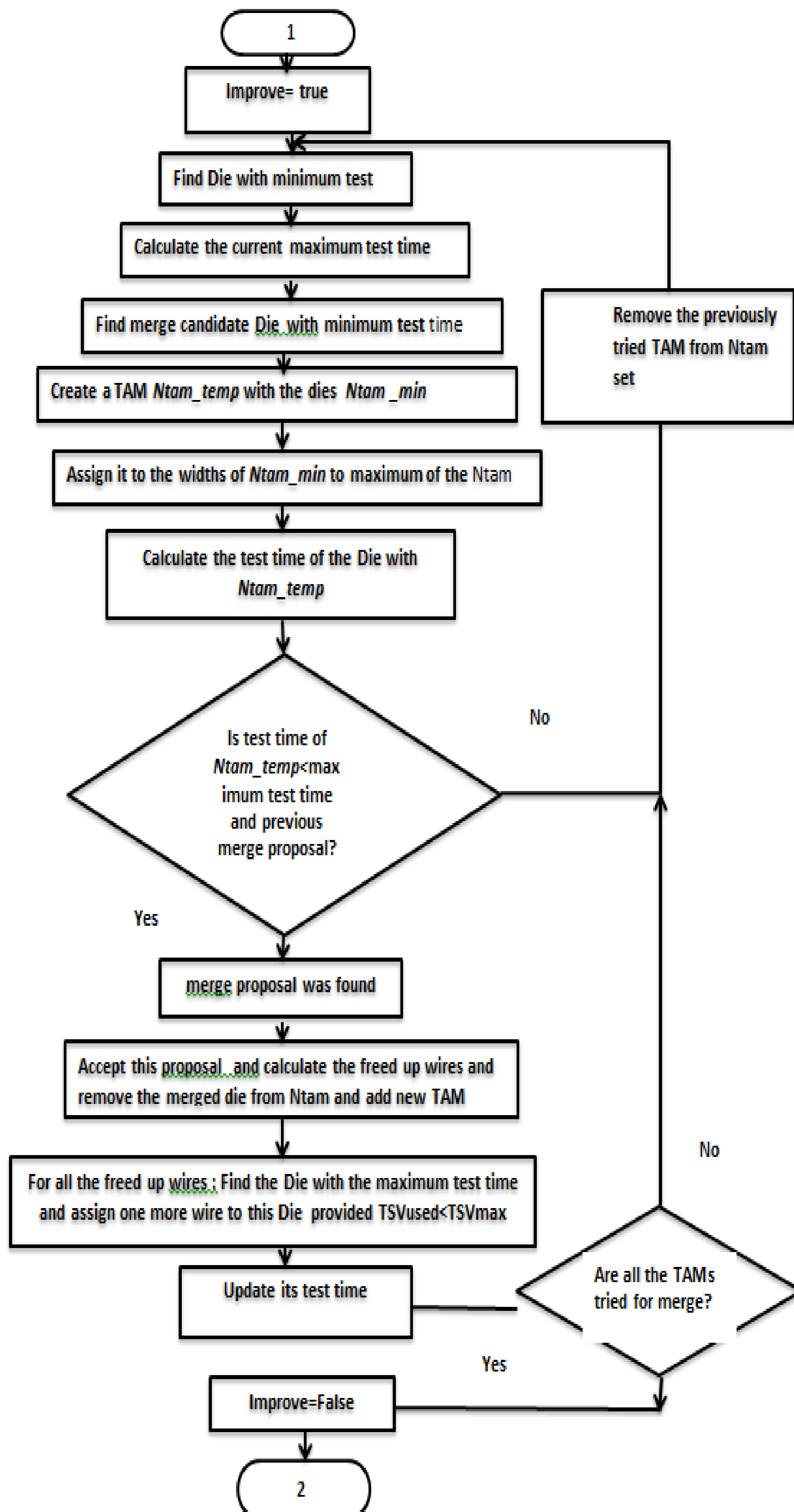
Figure 3.20Flowchart for TAM optimization stage 3

3.4.2.eTAM optimization for 3D stack

Once the 2D TAM solutions are decided, the final step of the global test optimization is done to decide the test schedules of the various 2D dies such that the test time can be minimized. Provided the limitation of fixed TAM width allotted to each die, this step merely tries to optimize the test time deciding on which dies should be tested in parallel or serial with each other. The idea is to test the multiplexed dies in serial while the others in parallel. Hence the test time of the multiplexed dies gets added up to the maximum of the test time of the dies being tested in parallel. The flowchart for optimization of the 3D stack is shown in figure 3.21.



Contd. Figure 3.21



Contd. Figure 3.21

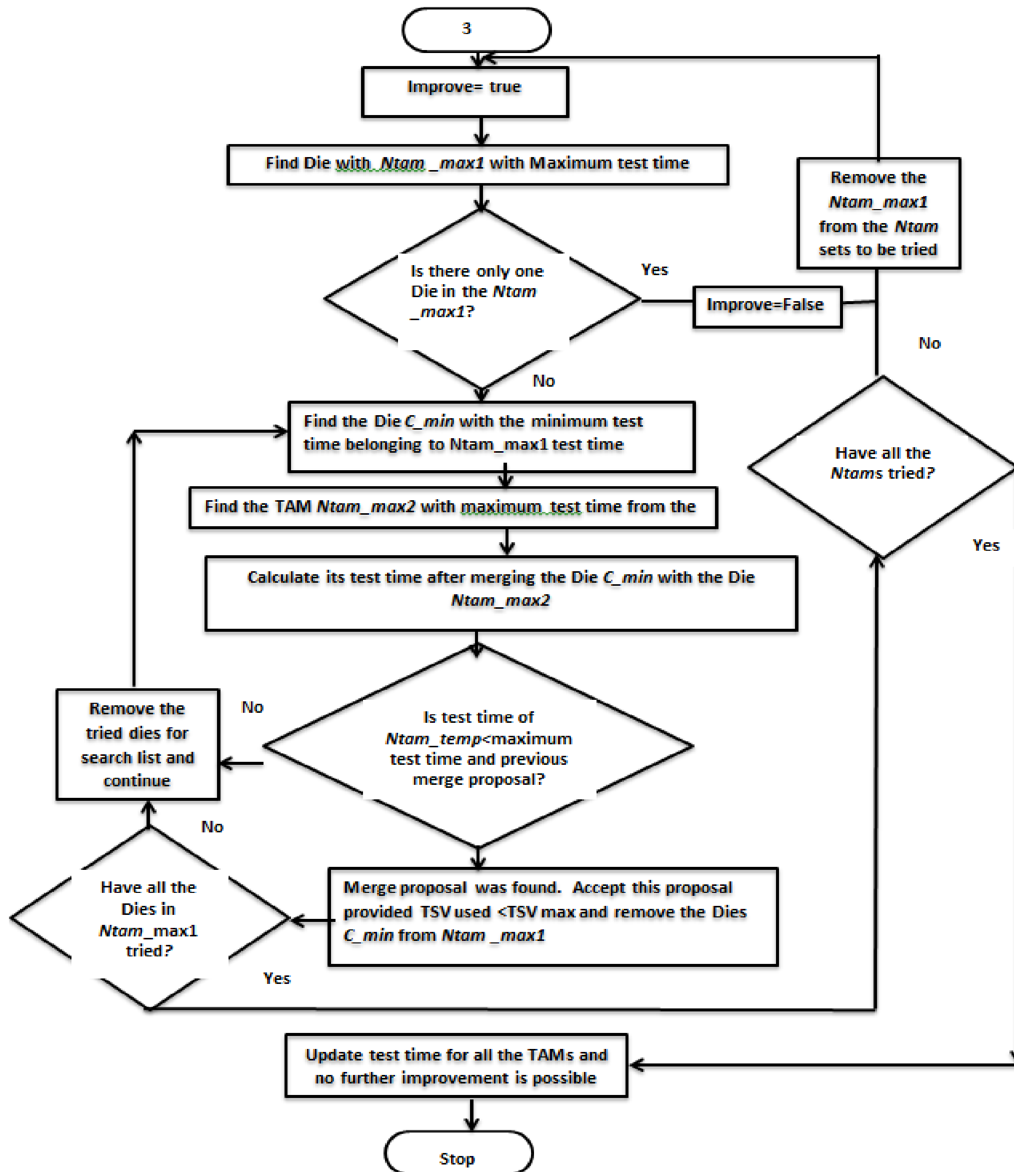


Figure 3.21 Flow chart for 3D test optimization

In case if the test integrator has to optimize the tests of individual dies and global 3D stack with a restriction of TAM partitions to be dedicated to different dies only then, the solution for 2D dies can be made by utilizing the sequential steps shown in figs. 3.17-3.20 followed by the 3D optimization step shown in flowchart given in figure 3.21. 3D optimization tries to redistribute the TAM wires among the various dies such that that the overall test time can be reduced. An attempt is made to assign more wires to the dies whose test time is high in comparison to others.This algorithm tries to find optimized wires at each die level by assigning minimal wires to each die and then finding optimized die time using 2D algorithms described earlier.During the time of scheduling the cores, a check is made that the cumulative

test power of the cores being tested in parallel doesn't exceed the maximum power limit. The same is deliberately not shown as a part of the flowchart to simplify it for understanding.

3.4.3Test solution development for 3D SoCs with flexible inter andflexible intra TAM architecture

The test solution to the problem P_{SS}can be derived by integrating the benefits of using fork and merge TAM architecture approach at the 2D die and 3D stack level. Since no limitation on the TSV number and TAM wires between different dies is taken into consideration so the solution needs to keep a check that at any point of time the available TSV usage doesn't exceed beyond the limits set by the TSV_{max}. Meanwhile, the test time is also reduced. An example of this kind of 3D TAM structure has been shown in figure1.10.

Herein, the solution to this kind of test problem incorporates the power, core hierarchy constraints. To avoid the concurrent scheduling of cores whose summative test power can exceed the maximum power limit, compatibility sets are initially prepared. Test wrappers to cater to the Flat and hierarchical cores are designed differently.Several trade-offs exist between the various resources as the optimization for one can lead to violation of other. For example: if the TSV limit and TAM are flexible then the infrastructure design can be highly optimized in reference to test time since TAM wires can be allotted as per the pareto optimal width selection to achieve minimum possible test time. On the other hand, if TAM and TSV limits are fixed then test time will be compromised. TSV usage minimization would become the prime objective which further dominates the TAM bus allocation to various dies. Cores test scheduling at die and stack levels then have to be done while satisfying various constraints for test time optimization.

In the proposed approach, the test time reduction is taken as the prime objective. Optimum widths for different cores are normally selected as per the pareto optimal points. TAM wires being soft are allowed to move between different dies and allotted to the different cores. Here number of TSV usage is taken care of by assigning the cores lying on one die to same TAM wires as much as possible. However, if required for test time reduction they are allowed to move up. So in a broader way this test solution design becomes more analogous to its 2D counterpart. The only difference is that the placement of various cores on the different dies guides the core selection in TAM allotment. The flowchart for the proposed solution is as shown in figure 3.22.

The algorithm initially works to find the ideal TAM width $width_p$for each core such that its test time is only a small percentage of what would have been if it was given maximum TAM

width available. The cores are allotted $width_p$ as long as the available TAM width is enough to support it. In case, the available width is less than the needed width then heuristics are used to insert the cores in the idle time. Each time a core finishes its test time, the freed TAM wires are added to the list of the available TAM width.

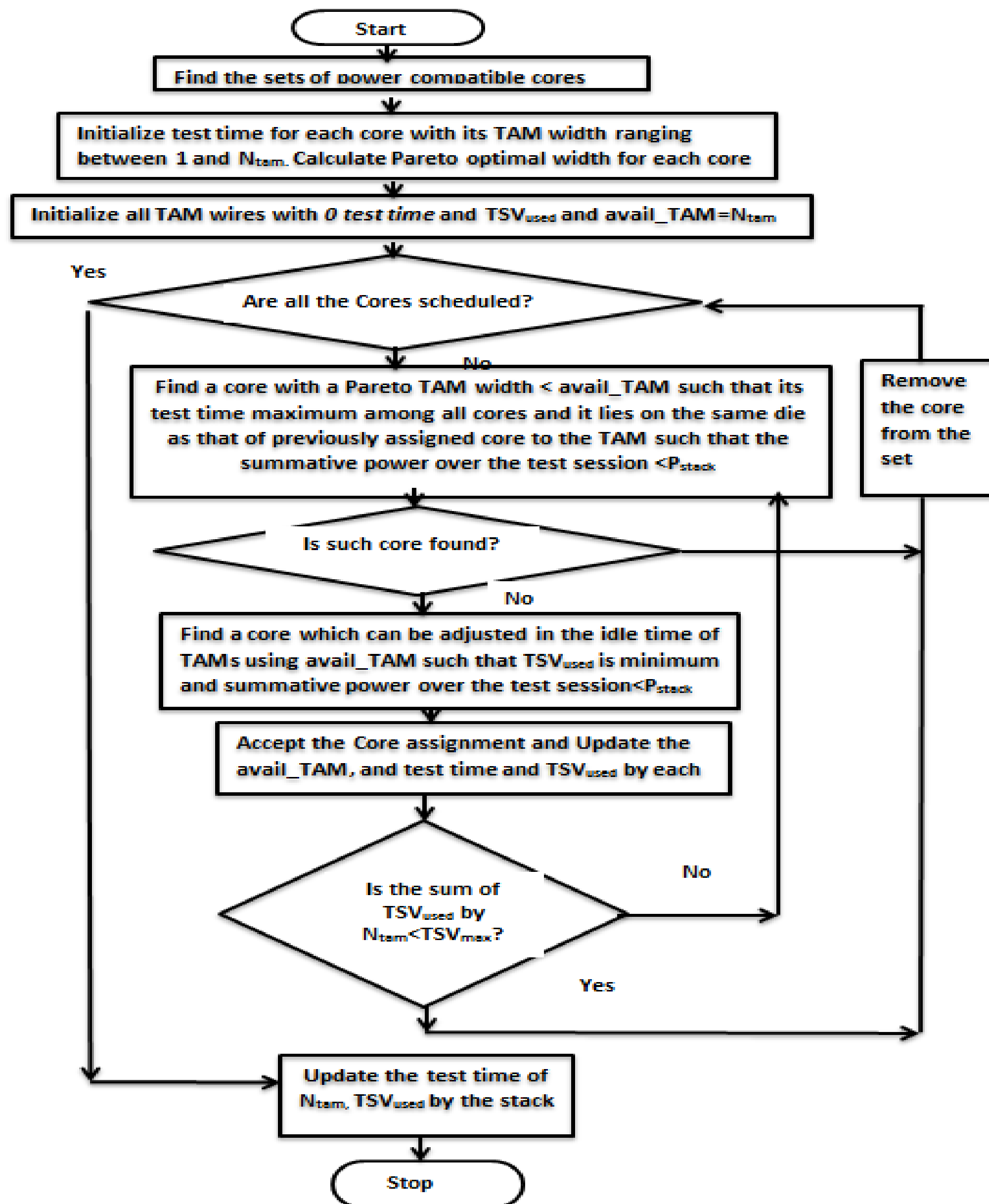


Figure 3.22Flowchart for 3D SoCs with flexible inter and fork and merge at intra TAM level

Cores with longest test time are allotted the TAM wires if the available TAM width is more than its Pareto optimal width and lying on the same die. In case the Core is the last to be scheduled then the available TAM is allotted the maximum TAM width directly irrespective of the pareto optimal width. On the other hand, if the core in question is not the last then the

algorithm attempts to widen up the TAM width of the cores with longest test time so that it can be reduced and the difference between the completion time of the cores being tested in parallel can be reduced. The processes are repeated till all the cores are scheduled efficiently.

3.5Summary

This chapter presented five different techniques for the test data compression. Out of these, ABMTC attempts to reduce the test power in combination with the test data compression. Test wrapper design for the fine grain partitioned 3D cores is proposed that minimizes the testing time of the core in addition to minimization of the TSV usage in forming up the wrapper chains. The 3D test solution developed for coarse grain partitioned SoC considers different test scenarios. The next chapter presents the results obtained after application of the test data compression techniques. Different test parameters have been used to show the effectiveness of the work.

4. RESULTS OF TEST DATA VOLUME COMPRESSION

4.1 Introduction

In order to validate the efficiency of the proposed techniques, comparisons are done with respect to results reported by previous techniques. The test sets generated by Mintest ATPG for Six large ISCAS' 89 benchmark circuits are taken as input and fed to the various compression algorithms. The quality of the various test data compression techniques can be gauged on the basis of four parameters namely: test data compression efficiency; on chip decoder area overhead, test application time and average /peak power dissipation.

4.2 Results and observations

The comparison between the proposed techniques and that of the previous works is as follows:

4.2.1 Test data compression efficiency

It is a metric used to find out the effectiveness of the compression achieved by the application of the encoding scheme. Assuming uncompressed test data volume to be represented by T_D and amount of the test data bits left after application of the compression technique be represented by $T_{E.}$ The compression efficiencycan be calculated as per the equation 4.1.

$$compression\ efficiency\ (\%) = \frac{\text{uncompressed data } (T_D) - \text{compressed data } (\mathrm{T_E})}{\text{uncompressed test data}} * 100 \quad (4.1)$$

More the value of compression efficiency more effective the technique would be. In order to validate the efficiency of the proposed technique, comparisons are done with previous techniques using benchmark circuits.

The statistics of occurrence of inter and unique cases are shown in the graph given in figure 4.1. As can be seen the occurrence of Count 0 (unique cases) is very high as compared to the other count values (inter cases). Therefore, it becomes essential to encode such blocks with lesser number of bits. This is done by analyzing the blocks at sub block level for the occurrence of special eight *(00/01/10/11/0U/1U/U0/U1/half_comp and half_inv_comp)* cases.

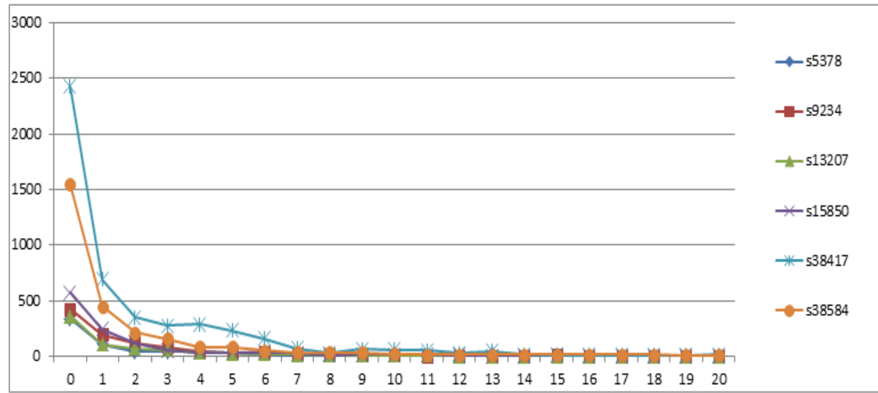


Figure 4.1Frequency of occurrence of unique and compatible blocks for block size =10

Another conclusion that can be drawn from graph is that the frequency of occurrence of one to eight number of blocks is high for almost all the benchmark circuits. Hence in order to encode them with less number of bits, the group one covers two to eight number of count. The graph presented in the figure 4.2 shows the statistics of occurrence of the special 8 cases among the unique cases. The unique patterns (having unique and non-compatible values at sub block levels) need to be kept as such in the compressed data. As evident from the graph, the occurrence of such blocks is quite high. In order to avoid redundant bits, such patterns are only preceded by the detect bit and prefix bits. Irrespective of the block sizes, the redundant bits are always kept to be two bits only.

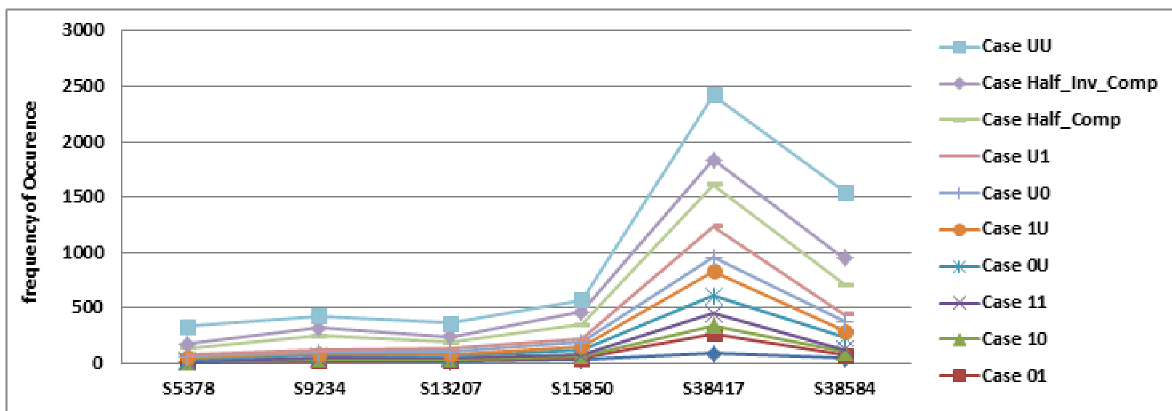


Figure 4.2 Frequency of occurrence of the special 8 cases among the unique cases for various benchmark circuit with block size =10 bits.

Graph in figure 4.3 presents the normalized values of the various special cases among the inter group one, two and three. As can be seen out of all special cases, the occurrence of *all zeroes/all ones/ zero-one/ one-zero, half_comp and half_inv_comp* is more as compared to the unique combinations. Therefore, only compatible and inverse compatible sub block cases

have been chosen out of all the eight special cases. The consideration of only two cases on one hand cover four (*all zeroes/all ones/ zero-one/ one-zero*) other cases while on other hand saves unnecessary bits which could have resulted in attempt to include other (0U/1U/U0/U1) cases. All *UU/0U/1U/U0/U1* and *UU* combinations are hence encoded as unique cases only.

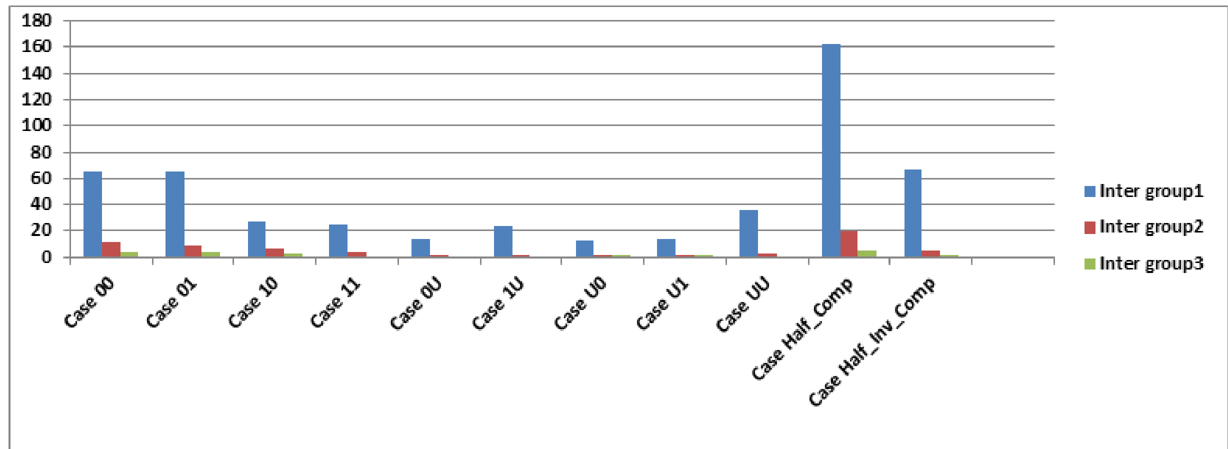


Figure 4.3 Frequency of occurrence of normalized values of the various special cases among the inter group one, two and three.

The compression efficiency results in percentage (using eq. 4.1) corresponding to different benchmark circuits for different block sizes using HBMTC are presented in table 4.1. The best cases are presented in column 15 for each case.

Table 4.1Compression efficiencies (in percentage) under different block sizes for different benchmark circuits

	Block Size (in bits)												
Benchmark circuit	**8**	**10**	**12**	**14**	**16**	**18**	**20**	**22**	**24**	**26**	**28**	**30**	**32**	**Best CR%**
S5378	**61.90**	**63.36**	**64.44**	**64.66**	**65.47**	**63.93**	**61.87**	**61.23**	**62.28**	**59.69**	**58.18**	**63.14**	**59.96**	**65.47**
S9234	**66.68**	**67.28**	**66.79**	**64.71**	**63.64**	**63.86**	**61.71**	**60.20**	**59.21**	**57.52**	**55.90**	**55.84**	**51.37**	**67.28**
S13207	**83.89**	**84.72**	**86.10**	**87.09**	**87.70**	**88.11**	**88.55**	**88.18**	**88.72**	**88.88**	**88.46**	**88.70**	**88.41**	**88.88**
S15850	**74.94**	**75.96**	**76.57**	**76.21**	**76.05**	**75.64**	**75.66**	**74.70**	**73.81**	**74.24**	**72.72**	**71.70**	**68.37**	**76.57**
S38417	**65.36**	**65.74**	**66.02**	**65.47**	**64.51**	**64.68**	**63.98**	**62.22**	**62.69**	**62.12**	**60.76**	**60.58**	**59.68**	**66.02**
S38584	**74.94**	**76.41**	**76.92**	**77.25**	**76.86**	**77.12**	**76.42**	**76.11**	**75.16**	**75.11**	**74.76**	**73.99**	**73.83**	**77.25**

The results corresponding to different encoding schemes and the proposed HBMTC are presented in table 4.2. The comparisons are made between the proposed techniques and other

schemes like Golomb [25], FDR[71] , EFDR [72], 9C [74], BM[64], CCPRL [76], 2^n-PRL [77], BM-8C [78].

Row 10 of table 4.2 represents the improvement obtained in application of HBMTC in comparison to various techniques mentioned in the respective columns. As can be seen the average improvement is obtained in using the HBMTC in comparison to previous techniques. The efficiency can be improved further if small codes are used for covering the all zeroes and all ones patterns.

Table 4.2Comparison of compression efficiencies (in percentage) achieved between proposed compression schemes and the various other techniques

	Test data compression technique							
Benchmark circuit	**Golomb**	**FDR**	**EFDR**	**9C**	**B.M**	**2^n-PRL**	**B.M-8C**	**HBMTC**
S5378	37.11	48.02	53.67	51.64	54.98	54.94	58.56	65.47
S9234	42.25	43.6	48.66	50.94	51.19	57.72	57.49	67.28
S13207	79.74	81.3	82.49	82.31	84.89	88.1	87.52	88.88
S15850	62.82	66.22	68.66	66.38	69.49	64.29	73.69	76.57
S38417	28.37	43.26	62.02	60.63	59.39	58.33	59.92	66.02
S38584	57.17	60.93	64.28	65.53	66.86	72.44	71.66	77.25
Average	51.24	57.22	63.29	62.38	64.46	65.97	68.14	73.58
% Improvement in HBMTC	22.34	16.36	10.29	11.2	9.12	7.61	5.44	

Such a type of technique increased the achievable compression beyond the results reported by BM-8C but the lack of ability to dynamically vary the *block_size* is still a limitation.

Figure 4.4 represents the frequency of occurrence of inter block and unique cases for test vector files of various benchmark circuits. A fixed block size equal to 8 for all the benchmark circuits has been taken just as an example and may not give the best efficiency.

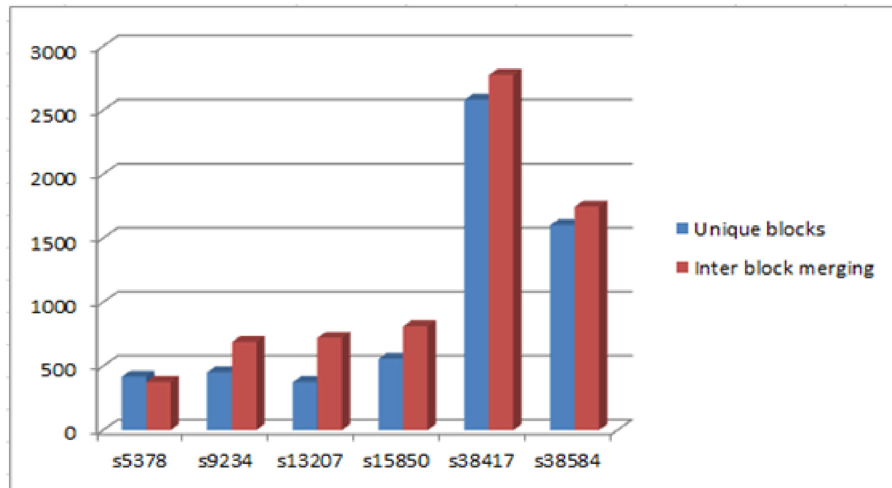


Figure 4.4 Frequency of occurrence of inter block merging and unique case for different benchmark circuits (for fixed block size =8)

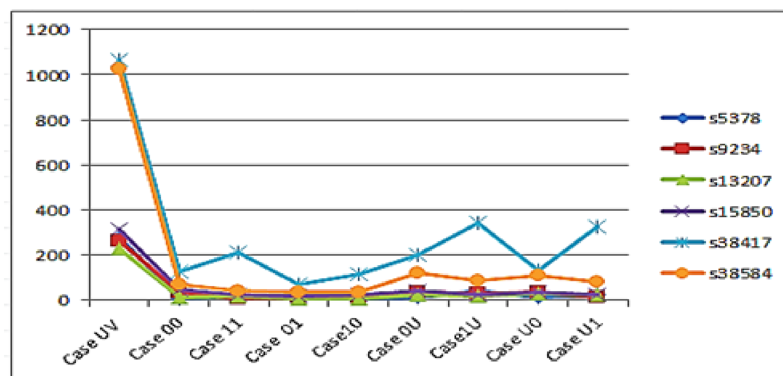


Figure4.5Frequency of occurrence of special eight cases for different benchmark circuits with block size = 8 bits

It may be observed that if the frequency of occurrence of compatible cases is always more than that of unique case then, the overhead of extra detect bit for the inter cases may reduce the compression efficiency. Hence, to take advantage of SCCPRL, unique cases should be more.

On analysing the unique cases further, blocks may be found to have occurrence of special 8 cases. Figure 4.5 provides the frequency of occurrence of special 8 cases within the unique pattern blocks. So if these eight special cases are encoded with lesser number of bits then better compression can be achieved. Same is being done in OSCCPRL.

The results corresponding to different encoding schemes and the proposed OSCCPRL are presented in table 4.3. The compression efficiency percentage can be obtained using equation 4.1. The comparisons are made between the proposed techniques and other schemes like Golomb [25], FDR [71], EFDR[72], 9C [74] , BM [64], CCPRL[76], 2^n-PRL[77], BM-8C

[64]in terms of the compression efficiency in percentage. Theresults for 10C are presented in columns 11 of table 4.3. The compression results presented in the table 4.3 include the best obtained for block (*K)* equal to: 16, 12, 14, 12, 18 and 10 bits for benchmark circuits: s5378, s9234, s13207, s38417, s15850 and s38584 respectively.

Table4.3Comparison of compression ratio of OSCCPRL with other test-independent compression techniques in terms of compression efficiency %

Benchmark circuit	Golomb [14]	FDR [15]	EFDR [18]	9C [24]	B.M [32]	CCPRL [24]	2^n-PRL [21]	B.M-8C [23]	10C	SCCPRL	OSCCPRL
s5378	37.11	48.02	53.67	51.64	54.98	61.08	54.94	58.56	52.40	60	71.15
s9234	42.25	43.6	48.66	50.94	51.19	62.95	57.72	57.49	51.30	52.19	75.93
s13207	79.74	81.3	82.49	82.31	84.89	90.06	88.1	87.52	80.64	92.1	94.14
s15850	62.82	66.22	68.66	66.38	69.49	76.32	64.29	73.69	64.72	79.34	85.91
s38417	28.37	43.26	62.02	60.63	59.39	64.61	58.33	59.92	58.45	70.15	73.23
s38584	57.17	60.93	64.28	`65.53	66.86	75.38	72.44	71.66	64.75	77	85.62
Average	51.24	57.22	63.29	62.38	64.46	71.73	65.97	68.14	62.04	71.80	80.99
% Improvement in SCCPRL	20.55	14.57	8.51	9.42	7.34	0.06	5.83	3.66	9.76		
% Improvement in OSCCPRL	29.75	23.77	17.7	18.62	16.53	9.26	15.03	12.86	12.86	9.19	

It may be noted that the 10C does provide a benefit of 1-2 % only in some of the benchmark circuits. Based upon the statistics, it could be concluded that the fall in efficiency occurs for the benchmarks with more number of *don't care* bits. Filling of such don't care bits can help in increasing run ofcases like *all zeroes* etc. *All zeroes* being encoded by using only one bit in 9C gives it an advantage of better compression. At the same time, this result can be improved if variable length 10C approach (similar to the approach used in 9C) is used. The results of SCCPRL are presented in column number 11, the block count length (*R)* is chosen to be 6, 5, 6, 5, 4, 6 (in bits) for the circuits s5378, s9234, s13207, s15850, s38417 and s38584 respectively. However, the count code is kept to be 3 bits in all cases. Row 9 of table 4.3represents the improvement obtained in application of SCCPRL in comparison to various techniques mentioned in the respective columns. As can be seen, SCCPRL shows a very little advantage of over that of CCPRL. Effectiveness of the SCCPRL technique in improving the compression efficiency can be more pronounced if the frequency of occurrence of the unique

cases is high. In other words, the overhead of extra detect bit in cases of inter block merging can be offset by the advantage provided by reduction of redundant bits. The same is done in OSCCPRL by selection of Intra block merging over Inter. Row 10 of table 4.3 represents the improvement obtained in application of OSCCPRL in comparison to various techniques mentioned in the respective columns. The block count length (*R)* is chosen to be 6, 5, 6, 5, 5, 6 (in bits) for the circuits s5378, s9234, s13207, s15850, s38417 and s38584 respectively. The fixed block_ size value (chosen for categorization of Intra block merging or unique cases) achieved for best cases are 16, 20, 48, 20, 12, 24 for the circuits s5378, s9234, s13207, s15850, s38417 and s38584 respectively. However, the count code is kept to be 3 bits in all cases since the frequency of occurrence of merging of blocks is not substantial beyond seven.

4.2.2 Decoder area overhead

This metric decides the extra burden on the SoC integrator on the basis of the on chip decoder area that is needed to retrieve the original test data. For the necessary comparison, the circuit under test and the decoder associated with the compression technique have been implemented using same platform.

4.2.2.a Comparison of the decoder area

The hardware overhead of the decoder of HBMTC and OSCCPRL (modeled using Verilog HDL and synthesized using Encounter RTL compiler from Cadence with 1.8 V, TSMC 180 nm CMOS standard cell library) is presented in table 4.4. The full-scan ISCAS'89 benchmark circuits are synthesized with single scan-chain. The comparison is made with respect to various other compression techniques.

It can be concluded that the area overhead is comparable to recently proposed BM-8C [78], CCPRL [76], techniques but, it is larger as compared toFDR [71] and 9C [74]. If there is a limitation in terms of the available silicon area then decision to employ either HBMTC or OSCCPRL techniques should be made after weighing the trade-offs of decoder area and needed test data compression.

Table 4.4 Comparison of decompression area overhead achieved between proposed compression schemes and the various other techniques.

Benchmark circuit	CCPRL	BM-8C	9C	FDR	HBMTC	OSCCPRL
S5378	9.6	12.8	8.2	7.8	12.5	10.7
S9234	7.3	9.7	6.2	5.9	9.2	8.3
S13207	3.5	5.8	3.7	3.5	5.10	4.2
S15850	3.7	5.9	3.8	3.6	4.5	3.9
S38417	1.8	2.3	1.5	1.4	2.2	2.2
S8584	1.9	2.5	`1.6	1.5	2.5	2.5

At the deep submicron level, being just a fraction of the overall larger SOCs design, such overhead may sometimes be acceptable.

4.2.3 Test application time

As mentioned in the section 1, the compressed test bits are fed to the on chip decompressor at ATE clock frequency (f_{ATE}). The FSM recognizes the codewords and generates the actual test data which is further applied to the circuit under test at circuit frequency (f_{CUT}) . For synchronization fate is selected to be an integral multiple of the f_{CUT}. Let the frequency ratio be $\alpha = f_{CUT} / f_{ATE}$. Assuming the compressed test data has N codewords C1−CN and each codeword has a length of Wi (i = 1, 2, ……,N). Let αmax =$\max_{2<i<N}$ (Hi − 1 /Wi) where Hi-1 is the length of decompressed test data for the codeword Ci-1. If α ≥ αmax, minimum TAT can be calculated as [76]:

$$\mathrm{TATmin} = \sum_{i}^{N} \mathrm{Wi} \quad + ([\max(\mathrm{Hn} - (\alpha - 2)])/\alpha \qquad (4.2)$$

If α < αmax, the ATE will be stalled several cycles to wait for the CUT to apply the decompressed test data, which occurs when the time consumed for ATE to send the codeword C_i to FSM is shorter than the time consumed for the CUT to apply the decompressed test data of the previous codeword C_{i-1}. Then, the total TAT will be calculated as [76]:

$$\mathrm{TAT} = \mathrm{TATmin} + \sum_{i=2}^{N} \{\max(\mathrm{Hi} - 1 - \mathrm{wi} * \alpha, 0\}/\alpha \qquad (4.3)$$

Table 4.5 presents the comparison of Test application time achieved between proposed compression schemes and the various other techniques in $f_{ATE.}$

Table4.5 Test application time calculations

Circuits	α	FDR	EFDR	BM	BM-8C	HBMT	OSCCPRL
S5378	2	24,933	17,075	16,018	15,088	14100	13,760
	4	16,803	13,172	12,239	11,191	10196	9,194
	6	15,259	12,096	11,183	10,348	9824	7,821
	8	14,039	11,652	10,899	10,089	9100	7,390
S9234	2	42,039	26,129	26,336	24,281	23644	21,753
	4	29,206	21,424	20,828	18,410	16128	14,008
	6	26,675	20,557	19,762	17,278	15124	11,852
	8	24,086	20,318	19,436	16,921	12766	11,030
S13207	2	1,16,101	88,487	88,045	87,319	85334	84,339
	4	70,361	52,711	50,784	49,730	47244	44,585
	6	57,089	41,898	39,177	38,138	37233	31,454
	8	48,358	36,946	33,768	32,326	30284	25,243
S15850	2	65,020	46,076	46,076	44,110	43224	41,080
	4	42,270	32,517	32,084	29,553	25234	24,317
	6	36,732	28,798	28,216	25,522	20337	19,071
	8	32,362	27,172	26,518	23,673	18346	16,710
S38417	2	1,86,261	1,04,569	1,09,180	1,06,725	99789	93,895
	4	1,23,700	75,614	80,273	79,074	69164	61,539
	6	1,13,451	68,212	73,286	71,564	60354	52,008
	8	1,10,521	65,509	70,202	69,069	56446	48,719
S38584	2	1,79,530	1,19,849	1,18,844	1,13,821	110013	10,5343
	4	1,18,628	86,320	83,255	76,161	74567	62,212
	6	1,04,630	78,066	73,953	66,048	58632	48,632
	8	93,260	74,955	70,692	61,908	52438	42,518

The TAT values are calculated as per eqns. 4.2 and 4.3 for different benchmark circuits. Results are computed for different α valuesas presented in table 4.5. Results are compared with the TAT obtained for other compression techniques like FDR[71], EFDR [72] , BM[12], BM[64], BM-8C [64]. As evident from the table, proposed HBMTC and OSCCPRL offers much reduced Test application time.

4.2.4.Scan-in Test Power

The total power consumption in scan-based testing is not only based on the number of transitions in test set but also on relative position of where the transition occurs [38]. One

common metric used to estimate the test power is the weighted transitions metric (WTM). The WTM is strongly correlated to the switching activity in the internal nodes of CUT during scan-shift operation. In [97]it is shown experimentally that scan vectors with higher WTM dissipate more power in CUT.

The WTM for the scan-in test stimuli $t_j$can be determined by

$$WTM_J = \sum_{i=1}^{l-1}(l-i)(t_{j,1} \; xor \; t_{j,i+1}) \qquad (4.4)$$

where$l$ is the scan-chain length and $t_j = t_{j,1}$, $t_{j,2}$, $t_{j,3}$ …….. $t_{j,l}$is the scan vector with $t_{j,1}$scanned in before $t_{j,2}$and so on.

The average power P_{avg} and the peak-power P_{peak} in scan-in mode for a test set $T_D = t_1$, t_2 , t_3 …… t_n can be estimated as

$$P_{avg} = \frac{\sum_{j=1}^{n}\sum_{i=1}^{l-1}(l-1)(t_{j,i} \; xor \; t_{j,i+1})}{n} \qquad (4.5)$$

$$P_{peak} = \max_{j \in (1,2,\ldots n)} \sum_{i=1}^{l-1}(l-i)(t_{j,i} \; xor \; t_{j,i+1}) \qquad (4.6)$$

Eqs. (4.4) - (4.6) show that, reducing the test vector's transition and the weight (*l-I*) are the key factors for reducing the average and peak-power. The same equations can be used to estimate also the average and peak-powers in scan-in mode.Tables 4.6 and 4.7 present the results so obtained in comparison to the previously proposed FDR [71] and EFDR [72] techniques.

Table 4.6Comparison of Scan-in peak power transitions between ABMTC scheme and various other techniques

Benchmark circuit	Mintest	FDR	EFDR	ABMTC
S9234	17494	12994	12062	12200
S13207	135607	101127	97613	97500
S15850	100228	81832	63494	63160
S38417	683765	505321	404654	402447
S38584	572618	234233	479547	463468
Average	301942	187101	211474	207755

Table 4.7Comparison of Sc-an-in average power transitions between ABMTC scheme and various other techniques.

Benchmark circuit	Mintest	FDR	EFDR	ABMTC
S9234	14630	5692	3469	3324
S13207	122031	12416	8016	7888
S15850	90899	20742	13394	13834
S38417	601840	172665	117834	13244
S38584	535875	136634	89138	89650
Average	273055	69630	46370	25588

As evident from the tables 4.6 and 4.7,both the scan-in peak and average power are reduced by employing the ABMTC technique.

4.3 Summary

The test data compression is a very promising technique to reduce the test data volume and challenges of test application time. The results of the five different test data compression techniques have been presented in this chapter. Out of these, 10C,SCCPRL, HBMTC and OSCCPRL techniques try to improve over the previous encoding techniques such that the test data compression can be increased. ABMTC attempts to decrease the switching power dissipation occurrence on the scan chains in addition to improvement of test data compression. In the techniques, the blocks of fixed length are chosen and tried for inter block merging. If the blocks are found to be unique then the intra level compression is done by iteratively dividing the blocks lengths in halves. Hence, the encoding is done at inter and intra block level using less number of bits. As per the simulation results, application of the HBMTC on various ISCAS'89 benchmark circuits show the compression efficiency to be increased by 20-50 % in comparison to the previously proposed techniques.

Simulation results of the application of the OSCCPRL on various ISCAS'89 benchmark circuits show the compression efficiency to be improved to an average of 80 %. The areas overhead of decompressor architecture as designed for OSCCPRL and HBMTC are found to be comparable to the earlier techniques. Being very small, their inclusion should be feasible when manufacturing is done at the deep submicron level. As evident from the results, the test application time is also found to be reduced by 27 -30%.

As per the simulation results of the application of the ABMTC on various ISCAS'89 benchmark circuits, it can be seen that the average compression efficiency is increased by 2-

20 % in comparison to the previously proposed techniques. The average and peak test powers are also found to be reduced by employing this technique. The decompressor architecture of ABMTC can be designed similar to that of OSCCPRL and HBMTC techniques and is expected to need a comparable silicon overhead.

The next chapter presents the proposed scheme for efficient development of test architecture for 3D SoCs.

5. RESULTS OF INTEGRATED TEST ARCHITECTURE FOR 3D SoC

5.1 Introduction

Optimized test wrapper design for fine grain partitioned 3D cores and test solution for coarse grain partitioned 3D SoCs have been proposed in chapter 3.The objective of both the approaches is to reduce the test time such that the TSV constraint is not violated. As discussed in chapter 3, for the fine grain partitioned SoCs, the cores are 3D which means their functional elements can lie on different dies. Therefore, the test wrapper design algorithm works on the optimization of wrapper chains such that the test time of individual cores can be optimized. Once the test wrappers are designed, an optimized 2D TAM architecture can be utilized to reduce the overall test time of the SoC. In case of coarse grain partitioned SoCs, the cores are still 2D but, their placement can be done on any of the stacked dies. Therefore, to ensure the testability of each component of the system, test elevators need to be used to appropriately route the test data between the various dies. Section 5.2 presents and discusses the results obtained for the test wrapper design for 3D cores. Section 5.3 presents the results for optimization of test plan for 3D SoCs assuming the cores to be 2D. The chapter concludes with a summary of the work.

5.2 Test wrapper design for fine grain partitioned 3D system on chip

The following section presents the results obtained for test wrapper design for fine grain partitioned SoC. To show the effectiveness of the proposed techniques, different ITC'02SoCbenchmark circuits [140] have been utilized due to unavailability of benchmarks for 3D SoCs.

5.2.1 Experimental Set up

Various SoC cores that have chosen for experimentation include:core7 of d281, core 4 and core13 of p93791. Out of the SoC benchmarks so selected for the experimentation d281 is developed by Duke University and has got 9 modules 2931 number of functional inputs and 882 functional outputs while p93791 and p22810 are developed by Philips research and Philips semiconductors. The detailed information of every benchmark is available in[140]. The algorithm is implemented in a high level language. Since the benchmark circuits are in 2D form so in order to make them compatible with the 3D circuit design, the placement of all the components of various cores is done using random enumeration. For every random

assignment of the components, different results are obtained for test time and TSV utilization. Therefore, minimum of fifty iterations are utilized to normalize the results obtained.

5.2.2. Results and observations

Tables 5.1 and 5.2 present the results as obtained for core 7 of d281. To show the effectiveness of the work, the comparisons have been made with the results presented in [95] and [96].

Table 5.1 Test Wrapper results for core 7 of SOC d281 in comparison to [94]

WC	TSV_{max}	Shortest Wrapper [94]	Proposed Shortest WC	TSV_{used} proposed	CPU Time (in secs)
2	12	NA	965	12	.02
	14	NA	977	11	.05
	16	1633	967	10	.04
	18	1064	967	12	.07
3	12	NA	592	11	.1
	14	NA	624	12	.08
	16	1347	624	14	.09
	18	752	624	8	.04
4	12	NA	500	12	.08
	14	NA	492	13	.07
	16	1347	522	13	.1
	18	611	522	9	.12
5	12	NA	480	11	.02
	14	NA	456	14	.05
	16	1347	456	15	.01
	18	486	450	15	.08

Table 5.2Test Wrapper results for core 7 of SOC d281 in comparison to [96]

WC	TSVmax	Longest Wrapper [96]	Proposed longest WC	TSVused [28]	TSVused proposed	CPU Time (in secs)
2	12	1129	1163	10	12	.02
	14	1223	1151	11	11	.05
	16	1223	1161	10	10	.04
	18	1223	1161	10	11	.07
3	12	710	770	12	11	.1
	14	710	755	14	12	.08
	16	710	755	14	14	.09
	18	816	755	16	8	.04
4	12	532	532	12	12	.1
	14	532	565	14	13	.07
	16	532	565	16	13	.1
	18	532	565	18	9	.12
5	12	426	430	12	12	.02
	14	426	430	14	14	.05
	16	426	432	16	15	.1

Table 5.3Test Wrapper results for core 4 of SOC p93791 in comparison to [94]

WC	TSV_{max}	Shortest Wrapper [94]	Proposed Shortest WC	TSV_{used} proposed	CPU Time (in secs)
2	12	NA	68	10	.05
	14	132	62	12	.05
	16	87	60	15	.07
	18	79	64	16	.04
3	12	NA	43	10	.1
	14	69	40	13	.08
	16	51	38	15	.09
	18	51	38	15	.04
4	12	NA	35	11	.05
	14	59	35	13	.07
	16	59	30	15	.1
	18	43	30	15	.12
5	12	NA	34	10	.04
	14	59	32	10	.03
	16	59	32	12	.08
	18	43	30	11	.09

Table 5.4 Test Wrapper results for core 4 of SOC p93791 in comparison to [96].

WC	TSV_{max}	Longest Wrapper [96]	Proposed longest WC	TSV-used [96]	TSV used proposed	CPU Time (in secs)
2	12	85	85	11	10	.05
	14	88	91	8	12	.05
	16	91	93	12	15	.04
	18	80	89	11	16	.07
3	12	53	58	12	12	.02
	14	53	59	13	14	.08
	16	52	58	14	15	.09
	18	73	56	15	15	.04
4	12	40	44	12	11	.02
	14	39	44	13	13	.07
	16	40	44	14	15	.10
	18	46	42	18	15	.12
5	12	31	42	11	12	.02
	14	33	40	12	14	.03
	16	32	40	14	14	.07
	18	36	38	15	11	.01

In all the tables presented above, column one represented the number of wrapper chains that have been formed as a part of the wrapper design. Column 2 represents the maximum limit for TSVs that can be utilized in each wrapper design. It may be concluded that where the algorithm in [94] is unable to form the wrapper chains (as shown in few of the cases) on being bounded by the TSV constraint, the solution can be obtained using the proposed algorithm. As shown in the tables 5.1-5.4, the results obtained seem to be quite motivating for the adoption of the proposed approach. It may be noted that the values of the TSV used and lengths of the wrapper chains can vary when used for practical scenario wherein actual placement of individual components is given. The results presented in columns four of tables 5.1-5.4 show the average of WC lengths as per the randomization approach used for the

placement of individual components which is done to cope up with the unavailability of 3D SoCs.

5.3 Test architecture for coarse grain partitioned 3D SoC

Optimization of the 3D test architecture is done to minimize the test time while ensuring the constraints set by the available TSV, test bandwidth, power etc. are not violated. Two practical scenarios of the 3D test are presented in this section that include cases of flexible inter and intra TAMand fixed inter and flexible Intra TAM architecture designs. The following section presents the results and observations for the test solution development for coarse grain partitioned3D SoC

5.3.1 Experimental Set up

To show the effectiveness of the proposed 3D test architecture methods, several ITC' 02 [140]SoC test Benchmarks circuits have been used including d265, p93791 and p22810 etc. For comparison with the previously proposed approaches, two hypothetical benchmarkscenarios have been preparedto create 3D like structure similar to what have been used in [28] and [45]. The algorithms have been written in C++. Algorithms presented in figures 3.15-3.20 have been used to determine the test architecture and test time for the various dies. The results for different test scenarios are as discussed below. Since, the die details of each core of the SoC under test have been assigned randomly so the results are compared with the lower bound of the test time presented in [82].

5.3.2 Results and observations

Simulation results obtained on application of the proposed solutions for two different test cases of test architecture optimization for3D SoCs have been presented in this section.

5.3.2 aResults for 3D test design with fixed but soft inter and flexible intra die TAM partition design

This case considers the different TAM partitions $N_{tam}(i)$ to be allotted to the different dies. As described in section 3.4.2, the number of TAM partitions have been made equal to the number of the dies. At each die, optimization of the TAM architecture is done for available width of TAM. Here, Test bus and Test rail architecture have been investigated to optimize the test time of individual dies. Results for three different SoCs are presented in tables 5.1 - 5.13 for thermal (power) unaware and the thermal(power) aware cases.Columns 2 of tables 5.8 to 5.16 presents the size of fixed TAM allotted to the dies, Column 3 represents the lower bound of the test time of the SoC. Column 3 represents the results reported in [28].

Corresponding to each case, the results have been reported with and without the power constraint described as thermally aware and unaware. As described in section 1.3.3, the individual cores assigned to each test bus needs to be tested sequentially while the testing of the cores lying on different test rails can either be done serially or in parallel. The test time calculations can be done by using the eqns. 3.10 and 3.11. In all the cases of the fixed inter die TAM, the TAM allotment to individual dies is done such that the TSV utilization is always maintained below that of the maximum values. A maximum TSV limit is chosen to be 60/80.

Table 5.5 Test time calculation for benchmark d695 with two dies and two partitions for thermally / power unaware cases.

N_{tam}	$N_{bus} = N_{die} = 2$	Thermally unaware [28]	Lower bound	Test time using proposed Algorithm	
				Test bus at 2D dies	Test rail at 2D dies
16	(7,9)	44225	40951	43875	44668
32	(11,21)	28524	20482	27372	29840
48	(21,27)	25761	13659	23223	23406
56	(23,23)	23256	11709	27249	21418
64	(23,41)	23232	10247	19875	21968

Table 5.6 Test time calculation for benchmark d695 with two dies and two partitions for thermally / power aware cases.

N_{tam}	$N_{bus} = N_{die} = 2$	Thermally unaware [28]	Lower bound	Test time using proposed Algorithm	
				Test bus at 2D dies	Test rail at 2D dies
16	(5,11)	44815	40951	44662	43364
32	(16,16)	26578	20482	27522	21626
48	(16,32)	20314	13659	19685	18599
56	(24,32)	19457	11709	20741	19275
64	(18,32)	44815	40951	44662	43364

Tables 5.5 and 5.6 show the comparison between the test times calculated by application of the proposed heuristic approach and one presented in [28] for the SoC benchmark circuit d695. The partitions used to obtain the various results are presented in columns 2 of each table. As evident the proposed technique gives comparable results in each case which are closer to the lower bound values too.

Tables 5.7 and 5.8 show the comparison between the test times calculated by application of the proposed heuristic approach and one presented in [28] for the SoC benchmark circuit

p22810 for thermally unaware and thermally aware cases. The partitions used to obtain the various results are presented in columns 2 and 3 of each table.

Table 5.7 Test time calculation for benchmark p22810 with three dies and three partitions for thermally / power unaware cases.

N_{tam}	N_{bus} =N_{die} =2	Thermally unaware [28]	Lower bound	Test time using proposed Algorithm	
				Test bus at 2D dies	Test rail at 2D dies
16	(7,9)	44225	5 19 509	421000	449793
32	(11,21)	28524	2 83 012	295835	313184
48	(21,27)	25761	139823	233550	232671
56	(23,23)	23256	119848	188687	182065
64	(23,41)	23232	104868	218687	209214

Table 5.8 Test time calculation for benchmark p22810 with two dies and two partitions for thermally / power aware cases.

N_{tam}	N_{bus} =N_{die} =2	Thermally aware [28]	Lower bound	Test time using proposed Algorithm	
				Test bus at 2D dies	Test rail at 2D dies
16	(6,10)	324902	5 19 509	543263	515537
32	(15,17)	270379	2 83 012	359079	312604
48	(19,29)	265452	139823	296420	146166
56	(19,37)	245234	119848	296937	238882
64	(19,45)	245234	104868	296343	238732

In each case, no fall in the test time is obtained once the bottleneck cores are encountered in the critical TAMs(the ones having higher test time).

Tables 5.9 and 5.10show the comparison between the test times calculated by application of the proposed heuristic approach and one presented in [28] for the SoC benchmark circuit p93791 for thermally unaware and thermally aware cases. Column 2 represents the different TAM partitions as allotted to different dies.

Table 5.9 Test time calculation for benchmark p93791 having four dies and four partitions for thermally / power unaware cases.

N_{tam}	$N_{bus} = N_{die} = 4$	Thermally unaware [28]	Lower bound	Test time using proposed Algorithm	
				Test bus at 2D dies	Test rail at 2D dies
16	(1,4, 4,7)	2179527	1746657	1822349	1886418
32	(1,10, 10,11)	944807	873334	957629	943241
48	(4,13, 13,18)	713347	582227	650511	727459
56	(1,7, 22,26)	664447	499053	593619	623169
64	(1,13, 22,28)	604553	436673	593619	645205

Table 5.10 Test time calculation for benchmark p93791 having four dies and four partitions for thermally / power aware cases.

N_{tam}	$N_{bus} = N_{die} = 4$	Thermally unaware [28]	Lower bound	Test time using proposed Algorithm	
				Test bus at 2D dies	Test rail at 2D dies
16	(1,1,2,12)	4023112	1746657	1931248	1768248
32	(1,1,4,26)	3695454	873334	1149163	1149163
48	(4,13,1,30)	1903700	582227	1084737	967940
56	(4,1,25,26)	1121150	499053	692199	640106
64	(1,10,25,28)	995522	436673	580444	571388

In the tables 5.5-5.10, it may be noted that the placement of the various cores will create a difference in the test time calculations of the entire SoC as the TAM partitions and the corresponding core assignment to them will vary.For doing the comparison with the results of [28], results have been computed for same TAM widths. It may be noted that better results could be achieved if the inter die TAM is optimized too.

5.3.2 bResults for flexible inter and intra TAM

Tables 5.11 - 5.13 present the results obtained for the SoCs d265, p93791 and p22810 respectively. The pseudo algorithm given in section 3.4.3 is used to design the TAM infrastructure wherein TAM wires are allowed to traverse the different dies of the SoC. Since the TAM architecture is taken to be flexible at the inter and intra die level so the TAM wires are not restricted to serve individual dies only. Assignment of the cores to the freed up TAM wires is done on the basis of the number of the available TAM wires and the pareto widths of the cores which have not tested. In order to reduce the TSV utilization, the TAM wires which have reached a certain die are investigated for their allotment to the cores lying on the same die. In case, no such core is found then, a heuristic is used to assign the cores which either leads to minimum increase in test time or TSV utilization. This problem doesn't limit the

number of TSVs that can be used between various dies and hence the test time can be optimized more effectively. However, a global limit of 200 TSVs is chosen to limit the solution space. Also, a maximum value of the peak power is taken up as constraint which helps in the selection of the cores whose tests can be scheduled concurrently. Columns 3-5 of the tables 5.11-5.13 represent the maximum values of the power that have been used as constraints while deciding the test solution for the complete stack. This problem becomes similar to the case taken up in [141] and hence the same power values have been chosen to be similar to the ones used in it.

Table 5.11Results of d695 for case P_{SS} for cores spread over 3 dies

N_{TAM}	Lower bound	Test time for P_{stack} =1500	Test Time for P_{stack} =2500	Test time for P_{stack} =3000	CPU time average
16	40951	44 175	44 175	44 175	0.1
24	27305	35 460	35460	35 460	0.3
32	20482	29 549	23 845	22 547	1.0
40	16388	29 242	22 491	19 262	2.0
48	13659	29 242	21 181	16 233	1.0
56	11709	29 242	21 181	16 233	1.1
64	10247	29 242	21 181	16 233	2.0

Table 5.12 Results of p22810 for case P_{SS} for cores spread over 3 dies

N_{TAM}	Lower bound	Test time for P_{stack} = 5000	Test time for P_{stack} = 8000	Test time for P_{stack} =10000	CPU time average (in Secs.)
16	419446	5 19 509	5 19 509	5 07 271	2.5
24	279644	3 52 373	3 68 160	3 64 674	5
32	209734	2 83 012	3 02 757	2 71 377	5
40	167787	2 39 544	2 04 678	2 17 118	4.5
48	139823	2 14 155	1 96 854	1 94 672	6
56	119848	2 65 284	2 03 198	1 50 589	10
64	104868	2 52 474	1 95 527	1 34 777	12.0

Table 5.13 Results of p93971 for case P_{SS} for cores spread over 3 dies

N_{TAM}	Lower bound	Test time for P_{stack} = 15000	Test time for P_{stack} = 20000	Test time for P_{stack} = 25000	CPU time average (in Secs.)
16	1746657	23 38 734	23 47 950	23 47 950	1.6
24	1164442	15 36 517	13 70 947	13 70 947	6.4
32	873334	13 72 852	12 36 118	11 06 710	14
40	698670	13 61 319	10 87 035	9 21 465	46.4
48	582227	7 63 032	6 97 736	6 95 331	65
56	499053	7 63 032	6 97 736	6 12 733	70
64	436673	7 25 080	6 50 218	5 81 539	82

The solution can be used to test the complete stack in minimum possible time however, it doesn't allow the partial stack or independent testing of individual dies and interconnect testing (which is major problem of test bus approach) at both inter and intra die level. Being a complete stack solution, it may provide the benefit of achieving best possible solution in terms of test time but, at the same time, it doesn't allow the leverage to reuse pre-existing TAM structures available on the dies.

To demonstrate the consideration of multiple test insertion which supports the pre, partial and post bond testing, results of the proposed technique are compared with the ones reported in [45].For experimentation, a hypothetical 3D SoC as used in [45] is considered which is formed by stacking different dies consisting of independent SoCs. Figure 5.1 presents the 3D SoC signifying the SoCs held at individual dies.

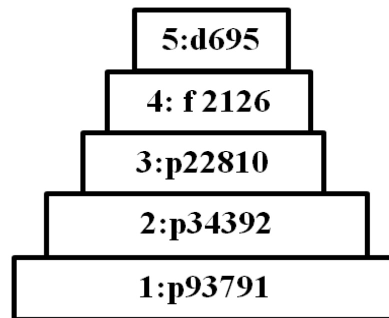


Figure 5.13D SoC formed by stacking multiple dies having independent SoCs

The test time so obtained after the insertion of multiple test schedules mentioned in columns 7-10 are presented in the column 5 of table 5.14. The TAM widths allotted to different dies are given in column 11. It may be noted that the sign '||' and ',' refers to cases when the dies are being tested in parallel or serial with respect to each other. As described above, the TAM architectures have been optimized to obtain the minimum time for the SoCs placed at different 2D dies. The proposed solution as discussed in section 3.4.2 is applied to the SoC model shown in figure 5.1 which allows the optimization of test partitions to be allotted to different dies and adoption of test bus/ test rail at die level. Column 1 presented 4 different test cases. Out of these, results presented for the cases P_{FS}^{H} and P_{MTS}^{H} have been taken from [45]. P_{FS}^{H} represents the optimized results for only final post bond stack test while P_{MTS}^{H} signify the multiple test insertion case with hard dies each having a constraint of maximum TSV limit for individual dies [45]. The test cases P_{HH} and P_{ss}presents the results obtained after application of the proposed solutions presented in section 3.4.2.

Table 5.14 Test time calculation for 3D SoC with multiple test schedules in comparison to results of [45]

Techniques used	Test case	TSV_{max}	W_{max}	Test Length	Red (%)	Test Sch. (1-2)	Test Sch. (1-2-3)	Test Sch. (1-2-3-4)	Test Sch. (1-2-3-4-5)	No. of test pins used per die
Test Length of P_{FS}^{H}	1	70	49	21254500	-	1||2	1,2||3	1||4, 2||3	1||5,2||3,4	30,25,25,20,15
	2	70	50	19091000	-	1||2	1,2||3	1,2,3||4	1||2,3||4	30,25,25,20,15
	3	70	69	10819000	-	1||2	1||2||3	1||2||3,4	1||3||4,2	30,25,25,20,15
	4	70	70	10819000	-	1||2	1||2||3	1||2||3.4	1,2||3||4	30,25,25,20,15
Test Length of P_{MTS}^{H}	1	70	49	12901800	39.30	1||2	1||2||3	1||3||4,2	1||2||3,4||5	30,25,25,20,15
	2	70	50	11042700	42.16	1||1	1,2||3	1||2,3||4	1,2||5,3||4	30,25,25,20,15
	3	70	69	9554160	11.69	1||2	1||2||3	1||3||4,2	1||2||3,4||5	30,25,25,20,15
	4	70	70	8959880	17.18	1||2	1||2||3	1,2||3||4	1||4||5,2||3	30,25,25,20,15
Proposed P_{HH}	1	70	49	12155656	42.81	1||2	1||2||3	1||3||4,2	1||2||3,4||5	30,25,25,20,15
	2	70	50	12370170	35.20	1||12	1,2||3	1||2,3||4	1,2||5,3||4	30,25,25,20,15
	3	70	69	9757794	9.80	1||2	1||2||3	1||3||4,2	1||2||3,4||5	30,25,25,20,15
	4	70	70	9557644	11.65	1||2	1||2||3	1,2||3||4	1||4||5,2||3	30,25,25,20,15
Proposed P_{SS}	1	70	49	11755913	44.69	1||2	1,2||3	1,2||3||4	1,2||3||4||5	35,25,20,10,14
	2	70	50	10616341	44.39	1||2	1||2||3	1||2,3||4	1,2||3,|4||5	35,25,20,10,14
	3	70	69	9417953	12.95	1||2	1||2||3	1||3,2||3	1||2,3||4||5	35,25,20,10,14
	4	70	70	9217305	14.80	1||2	1,2||3	1,2||3||4	1||4||5, 2||3	35,25,20,10,14

Column 6 of table 5.10 shows the reduction obtained after application of the proposed techniques with respect to the results presented in [45] for the case of multiple test insertion with hard dies. For the necessary comparison, the TAM widths allotted to each dies are chosen to be same as that of the values chosen in [45].

Experiments have been performed on d695, p22810 and p93791 to determine the effect of the core placement and TAM bandwidth on the TSV usage while optimizing the test time to near optimal with power constraint.In order to develop a general test solution that performs power optimization in addition to test time, three different floorplans: a) Nice b) uniform and c) random are considered for carrying out the simulation.

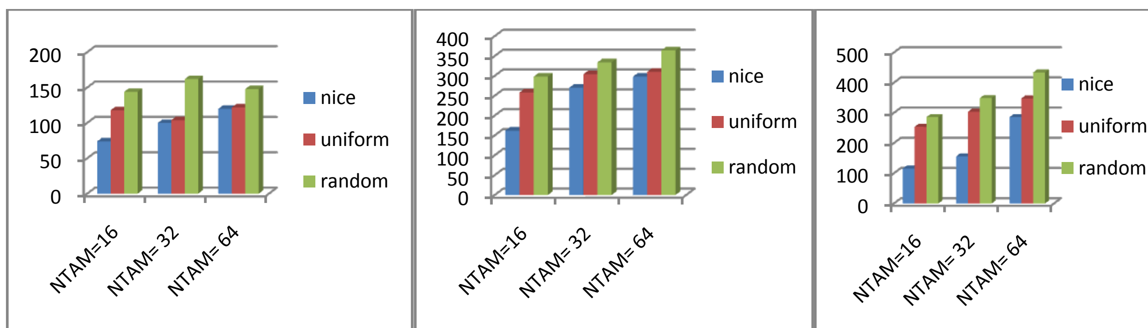


Figure 5.2 Graphical comparison of TSV usage corresponding to different placement of cores for a) d695 and b) p22810 and c) p93791

Results for different TAM widths and TSV*used* combinations for the three placement criteria are presented in figure 5.2 for d696, p22810 and p93791 SoC circuits.Nice floorplan

corresponds to the case wherein the cores which dissipate more power during the test are placed at the bottom die. Nice placement allows keeping the power hungry cores closer to the heat sink. To reduce the overall test time, bottom die is allotted larger number of TAM wires. It allows testing the big-power hungry cores faster and enabling more concurrency later on. Also it reduces the TSV usage since lesser number of TAM wires need to move up and down. Uniform placement corresponds to spreading the power hungry cores uniformly over the different dies helps ensuring that all the heat dissipation does not lead to local hotspots. Random assignment considers the worst case wherein the cores are placed randomly on different cores irrespective of their power budge.

5.4Summary

3D SoCs offer the advantage of realization of a complex SoC using much reduced footprint.The vertical interconnects have led to the reduced delay and power issues. This technology drift has brought new design and test challenges. Therefore, the test infrastructure design needs to be optimized with additional constraints on top of what existed for 2D counterparts. Simulation results for the proposed optimal test wrapper design for fine grain partitioned 3D SOCs have been presented in this chapter. The results show better balancing of wrapper chains with TSV utilization not exceeding the maximum available TSV limit.

Results for proposed test architecture development of 3D SoCs have presented that considering two different test cases. The results are presented for thermal unaware and thermal aware cases. Test lengths obtained after application of proposed techniques when compared to other previous techniques showed comparable results. In many cases, the obtained results are found to be better optimized. Theresults as obtained for P_{SS} test case which considers a complete stack test with major objectivesof providing a time and power optimized solution are also presented. Experimental results for multiple test insertions (which supports partial and full stack tests) for two different test scenarios which considered fixed/flexible inter TAM and flexible intra TAM are also presented. The results obtained showed an improvement of around 10-40 %. Experimental results show that inspite of having a tight constraints the approaches proposed result in a near optimum test time.

6. CONCLUSIONS AND FUTURE SCOPE

6.1 Conclusions

To keep pace with the ever growing demand of the market, semiconductor industry has gone through tremendous advancements in the design and fabrication technologies. Embedded core based system on chip design has proven to be a favorable solution to cope up with the enormous design efforts needed to realize the system. Pre verified and predesigned IP cores brought from either within the same company or outside vendor are integrated on a common base to realize the system on chip. Such a design technology offers the advantages like improved performance, reduced power, increased density along with providing a platform to contain a heterogeneous mix of logic blocks. All the benefits got enhanced by the recent developments in the semiconductor industry that includes formation of stacked 3D structures and network on chip. Introduction of fabrication technology in the vertical dimension led to the development of 3D SoC. Multiple 2D dies are stacked together to realize a complete 3D SoC with much higher component density and reduced footprint. Vertical interconnections between the various dies are provided using micro bumps and through silicon vias. Another technology advancement that has benefited the semiconductor industry is the inclusion of network on chip. Both the vertical interconnects and Network on chip have reduced the wiring crisis to large extent.

The evolution in the fabrication and design technology has brought along many test challenges. Some of the major challenges includes: multifold increase in test data volume; Highly uncorrelated test data which when transported between the ATE/ SoC periphery to the circuit under test leads to huge switching activity; deeply embedded components becoming difficult to access as the difference between number of on chip components and SoC pins is increasing etc. To ensure the quality of the system, an efficient test plan is required which may help in reducing the test cost. The work proposed in this thesis, includes test data compression techniques and development of an integrated test solution for 3D SoCs.

Out of the five test data compression techniques: 10C, SCCPRL, OSCCPRL and HBMTC reduces the amount of test data by utilizing the inter and intra block merging approaches. Another variant ABMTC attempts to reduce the peak and average test power by employing the hamming distance based test vector reordering in addition to block merging. The results obtained for the proposed techniques corresponding to various ISCAS'89 benchmark circuits show their effectiveness in achieving the desired objectives. The area requirements of the

decoders designed for OSCCPRL and HBMTC techniques present how with a little on chip design overhead, one can manage to achieve increased test data compression. More the test data compression lesser will be the amount of ATE memory and time needed for its transportation between ATE and SoC periphery. However, the price one has to pay is the area overhead of the on chip decoder.

The fine grain partitioned 3D SoCs comprising of 3D cores require an adaptation in the test wrapper chain formation. The various core elements spread over multiple dies need to be inserted in the wrapper chains for the appropriate test data delivery. A test wrapper design proposed as a part of this work performs the optimization of the test time for a given TAM width. The results obtained for different cores of the ITC'02 benchmark circuits show its comparison with respect to other previously proposed techniques.

The integrated test solution developed for 3D SoCs optimizes the test time while addressing the various constraints set by Test bandwidth, maximum TSV limit, test power etc. The solution considers two practical test scenarios and solves them in a two steps process. It optimizes the test solution for 2D dies for a given test bandwidth constraint followed by the optimization of 3D stack test for available TAM and TSV limits. The techniques are tested using various ITC'02 benchmark circuits. The results obtained are found to be comparabletoprevious techniques and hence show the effectiveness of the work.

6.2 Future scope

Test cost reduction being a critical component in the overall cost of the system needs to be addressed efficiently by employing various techniques some of which have been explored in this work. Though, the proposed techniques of test data compression optimize the test time by reducing the amount of test data and ATE memory but they still lag in the test power reduction. The overhead of decoder design may sometimes be a burden for the designer. More work needs to be done in the test data compression domain which should not only optimize the test time but also reduce the switching activity by some means. Amalgamation of the frequency directed encoding techniques and other block merging based encoding techniques can be done to further reduce the test data. Statistical encoding techniques can be further explored to reduce the test data. On chip memory and look up tables can be utilized to reduce the decoder overhead. Broadcast based test vector application techniques can be combined with various compression techniques to support multisite testing and save test time. The 3D test wrapper design can be adapted as per the hierarchical details of the cores. A model to estimate the number of TSVs that can reduce the test time of the 3D cores can be

developed which may guide the design and test engineers for reserving the TSVs for test purpose. Similarly, the proposed 3D test solution can be upgraded to cover the testing of the TSVs lying between the various dies. The various optimization algorithms like Ant colony optimization, bee colony optimization, genetic etc. can be explored to reduce the computational time of the 3D test solution.

NoC based SoC testing approach has better prospects to reduce the test cost by reusing the on chip network. Various network topologies, IP mapping techniques and evolutionary algorithms can be worked upon to proficiently reduce the test time. Techniques must be developed which should ensure the testability of network components and the core logics. The test solutions should also reduce the test power which is yet another major concern after test time.

References

[1] R. Saleh et al, "System-on-chip: Reuse and integration.," *Proceedings of the IEEE,* pp.. 1050-1069, 2006.

[2] R.R Tummala, VK Madisetti, "System on chip or system on package?," *IEEE Design & Test of Computers,* vol. 16, no. 2, pp. 48-56, 1999.

[3] B. A. Pierre, *Reuse methodology manual: for system-on-a-chip designs*, Springer Science & Business Media, 2012.

[4] R. Rajsuman, *System-on-a-chip: Design and Test,* Artech House, Inc., 2000.

[5] J.A. Davis et al, "Interconnect limits on gigascale integration (GSI) in the 21st century," *IEEE,* vol. 89, no. 3, pp. 305-324, 2001.

[6] R. S. Patti, "Three-dimensional integrated circuits and the future of system-on-chip designs.," *IEEE,* vol. 94, no. 6, pp. 1214-1224, 2006.

[7] P. Ramm et al, "3D System-on-Chip technologies for More than Moore system," *Microsystem technologie,* vol. 16, no. 7, pp. 1051-1055, 2010.

[8] K. Banerjee et al ,"3-D ICs: A novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration," *IEEE,* pp. 602-633, 2001.

[9] Y. Xie et al, "Design space exploration for 3D architectures," *Journal on Emerging Technologies in Computing Systems (JETC),* vol. 2, no. 2, pp. 65-103, 2006.

[10] A. Hemani et al, "Network on chip: An architecture for billion transistor era.," In *Proc. of the IEEE NorChip Conference*, Denmark, pp. 117-124, 2000.

[11] L. Benini and Giovanni De Micheli, "Networks on chip: a new paradigm for systems on chip design.," In *Proc. of Design, Automation and Test in Europe Conference and Exhibition,*, Europe, pp. 46-51,2002.

[12] J.Nurmi, "Network-on-chip: A new paradigm for system-on-chip design," In *Proc. ofInternational Symposium on System-on-Chip*, Tampere, pp. 117-124, 2005.

[13] W. J. Dally and B. Towles, "Route packets, not wires: on-chip inteconnection networks.," in *Proc. of38th annual Design Automation Conference (DAC '01)*, New York, USA, 2001.

[14] M. P. Kumar, S. Murali, and V. Kamakoti,"Network-on-Chips on 3-D ICs: Past, Present, and Future," *IETE Technical review*, vol 29, no. 4,pp. 318-335, 2012.

[15] M.A. Breuer and A. Friedman,*Digital system testing and testability design,* 1990.

[16] V.D. Agrawal, *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*,Springer Science & Business Media, 2004.

[17] H. H.S. Lee and K. Chakarbarty, "Test challenges for 3D integrated circuits," *IEEE Design & Test of Computers,* vol. 26, no. 5, pp.711-722, 2009.

[18] E. J. Marinissen, "Challenges in testing TSV-based 3D stacked ICs: test flows, test contents, and test access," in *Proc. ofIEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 544-547, 2010.

[19] "International Technology Roadmap for Semiconductors," 2012. [Online]. Available: http://www.public.itrs.net..

[20] Y. Zorian, E. J. Marinissen and S. Dey, "Testing embedded-core based system chips," in *Proc. ofInternational Test Conference*,pp. 130-143, 1998.

[21] V. lyengar, K. Chakrabarty and E. J. Marinissen, "Wrapper/TAM co-optimization, constraint-driven test scheduling, and tester data volume reduction for SOCs," in *Proc. of39th. IEEE Design Automation Conference*,pp. 685-690, 2002.

[22] P.K. Nag. et al., "Modeling the economics of testing: a DFT perspective.," *Design & Test of Computers ,* vol. 19, no. 1, pp. 29-41, 2002.

[23] E. H. Volkerink et al "Test economics for multi-site test with modern cost reduction techniques," in *Proc. of20th IEEE VLSI Test Symposium*, pp. 411-416, 2002.

[24] A. Sehgal, V. Iyengar and K. Chakrabarty, "SOC test planning using virtual test access architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 12, no. 12, pp. 1263-1276, 2004.

[25] K.Chakrabarty et al "System-on-a-chip test-data compression and decompression architectures based on Golomb codes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 20, no. 3, pp. 355-368,March 2001.

[26] N. A. Touba, "Survey of test vector compression techniques," *IEEE Design & Test of Computers ,* vol. 23, no. 4, pp. 294-303, 2006.

[27] E. Larsson and Z. Peng, "An integrated system-on-chip test framework," in *Proc. of IEEE conference on Design, automation and test*, Europe,pp. 138-144, 2001.

[28] X. Wu et al, "Test-access mechanism optimization for core-based three-dimensional SOCs," *Microelectronics Journal,* vol. 41, no. 10, pp.601-615, 2010.

[29] A. M. Amory et al, "A scalable test strategy for network-on-chip routers," in *Proc. ofIEEE International Test Conference*, pp. 9, 2005.

[30] E. Cota, L. Carro, and M. Lubaszewski, "Reusing an on-chip network for the test of core-based systems.," *ACM Transactions on Design Automation of Electronic Systems (TODAES)* , vol. 9, no. 4, pp. 471-499, 2004.

[31] E. J. Marinissen, S. K. Goel and M. Lousberg, "Wrapper design for embedded core test," in *Proc. ofInternational test conference*, pp. 911-920, 2000.

[32] E. J. Marinissen et al,"A structured and scalable mechanism for test access to embedded reusable cores," in *Proc. ofinternational test conference*,pp. 284-293, 1998.

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

[34] H. Rahaman, D. K. Das, and B. B. Bhattacharya, “An Adaptive BIST Design for Detecting Multiple Stuck-Open Faults in CMOS Complex Cell,” *IEEE Transactions on Instrumentation and Measurement,* vol. 57, no. 12, pp.2838-2845, December 2008.

[35] F. DaSilva et al, "Overview of the IEEE P1500 Standard," in *Proc. ofInternational test conference*, pp. 988-997, 2003.

[36] "IEEE P1500 Standard for Embedded Core Test," [Online]. Available: http://grouper.ieee.org/groups/1500/.

[37] A. Sehgal et al, "IEEE P1500-compliant test wrapper design for hierarchical cores.," in *Proc. ofInternational test conference*, pp. 1203-1212, 2004.

[38] S.K.Goel et al, "Testing of SoCs with hierarchical cores: common fallacies, test access optimization, and test scheduling," *IEEE Transactions on Computers* , vol. 58, no. 3, pp. 409-423, 2009.

[39] J. Aerts and E. J. Marinissen "Scan chain design for test time reduction in core-based ICs," in *Proc. ofInternational test conference*, pp. 448-457, 1998.

[40] V. Iyengar , K. Chakrabarty and E. J. Marinissen, "On using rectangle packing for SOC wrapper/TAM co-optimization," in *Proc. of20th international VLSI Test Symposium*, pp. 253-258, 2002.

[41] M. L. Flottes, J. Pouget and B Rouzeyre,"Sessionless test scheme: Power-constrained

test scheduling for system-on-a-chip," in *Proc. ofProceedings of the 11th IFIP on VLSI-SoC*, pp. 105-110, 2001.

[42] K. Chakrabarty, "Test scheduling for core-based systems using mixed-integer linear programming," *IEEE Transactions on computer-aided design of integrated circuits and systems,* vol. 19, no. 10, pp. 1163-1174, 2000.

[43] E. Larsson and H. Fujiwara, "Power constrained preemptive TAM scheduling," in *Proc. of. Seventh IEEE European Test Workshop,* pp. 119-126, 2002.

[44] V. Iyengar and K. Chakrabarty, "Precedence-Based, Preemptive, and Power-Constrained Test Scheduling for system on chip", in *Proc. Of 19th IEEE VLSI Test Symposium,* pp.368-374, 2001.

[45] B. Noia,K. Chakrabarty and E. J. Marinissen, "Optimization methods for post-bond testing of 3D stacked ICs," *Journal of Electronic Testing,* vol. 28, no. 1, pp. 103-120, 2012.

[46] E. J. Marinissen, J. Verbree and M. Konijnenburg, "A structured and scalable test access architecture for TSV-based 3D stacked ICs," in *Proc. of28th IEEE VLSI Test Symposium (VTS)*,pp. 269-274, 2010.

[47] E. J. Marinissen et al,"3D DfT architecture for pre-bond and post-bond testing," in proc. *IEEE International 3D Systems Integration Conference (3DIC)*, pp.1-8, 2010.

[48] F.Yuan, L. Huang and O. Xu,"Re-examining the use of network-on-chip as test access mechanism," in *ConfProc of Conference on Design, automation and test*, Europe, pp. 808-811, 2008.

[49] M. Amde, "Asynchronous on-chip networks," in *Proc. Of IEE Computers and Digital Techniques*, vol 152, no. 2,pp. 273-283, 2005.

[50] M. Krstic, "Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook," *IEEE Design & Test of Computers,* vol. 24, no. 5, pp. 430-441, Sept.-Oct 2007.

[51] H. J. Yoo, *Low-power NOC for high-performance SOC design*, CRC Press, 2008.

[52] P. P. Pande et al, "Performance Evaluation and Design Trade-offs for Network-on-Chip Interconnect Architectures," *IEEE Trans. Computers,* vol. 54, no. 8, pp. 1025-1040, August 2005.

[53] P. Guerrier, A. Greiner, "A generic architecture for on-chip packet-switched

interconnections, " in *Proc ofConference and Exhibition on Design, Automation and Test*, paris, pp. 273-283, 2000.

[54] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *IEEE transactions,* vol. 83, no. 10, pp. 1374-1396, 1995.

[55] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys (CSUR)* , vol. 38, no. 1, pp. 1-51, June 2006.

[56] H. C. Cooke et al, *Surviving the SOC Revolution: A Guide to Platform-Based Design*, Bostan: Kluwer Academic Publishers, 1999.

[57] Wang et al, *System-on-chip test architectures: nanometer design for testability* Morgan Kaufmann, 2010.

[58] P. Girard, "Survey of low-power testing of VLSI circuits," *IEEE Design and test of computers,* vol. 19, no. 3, pp. 82-92, 2002.

[59] R. M. Chou, K. K. Saluja and V. D. Agrawal, "Scheduling tests for VLSI systems under power constraints," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 5, no. 2, pp. 175-185., 1997.

[60] M. Ishida, D. S. Ha, and T. Yamaguchi, "COMPACT: A hybrid method for compressing test data.," in *Proc.of 16th IEEE VLSI Test Symposium*, pp.355-368, 1998.

[61] R. Sankaralingam, R. R. Oruganti, and N.A. Touba, "Static compaction techniques to control scan vector power dissipation," in *Proc of18th IEEE VLSI Test Symposium*, pp. 35-50, 2000.

[62] W.-D. Tseng, "Scan chain ordering technique for switching activity reduction during scan test," *in Proc.ofIEEE Proceedings-Computers and Digital Techniques*,vol 152, no 5, pp.609-617, 2005.

[63] U. S. Mehta, K. S. Dasgupta, N. M. Devashrayee, "Hamming distance based reordering and columnwise bit stuffing with difference vector: A better scheme for test data compression with run length based codes," in *Proc. of23rd International Conference on VLSI Design,*pp. 33-38, 2010.

[64] A. H. El-Maleh, "Efficient test compression technique based on block merging," *IET Computers & Digital Techniques* , vol. 2, no. 5, pp. 327-335, 2008.

[65] A. Jas, J. G. Dastidar, and N.A. Touba, "Scan vector compression/decompression using

statistical coding," in *Proc. of17th IEEE VLSI Test Symposium*,pp. 114-120, 1999.

[66] A. Jas et al, "An efficient test vector compression scheme using selective Huffman coding," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 22, no. 6, pp. 797-806, 2003.

[67] K. Xrysovalantis, E. Kalligeros, and D. Nikolos, "Optimal selective Huffman coding for test-data compression," *IEEE transactions on computers,* vol. 56, no. 8, pp. 1146-1152, August 2007.

[68] X. Kavousianos, E. Kalligeros and D. Nikolos,"RL-huffman Encoding for Test Compression and Power Reduction in Scan Applications," *ACM Trans. Des. Autom. Electron. Syst.,* vol. 10, no. 1, pp. 91-115, January 2005.

[69] M. H. Tehranipouret al, "Mixed RL-Huffman encoding for power reduction and data compression in scan test," in *Proc. ofIEEE International Symposium on Circuits and Systems*, vol. 2, pp. II-681, 2004.

[70] V. Iyenger, K. Chakrabarty and E. J. Marinissen, " Integrated wrapper/TAM co-optimization, constraint-driven test scheduling, and tester data volume reduction for SOCs," in *Proc. of39th Design automated conference (DAC)*, pp. 685-690, 2002.

[71] A. Chandra and K. Chakrabarty, "Frequency-directed run-length (FDR) codes with application to system-on-a-chip test data compression," in *Proc. of19th IEEE VLSI Test Symposium*, pp. 42-47, 2001.

[72] A. H. El-Maleh and R.H. Al-Abaji, "Extended frequency-directed run-length code with improved application to system-on-a-chip test data compression," in *Proc. of9th International Conference on Electronics, Circuits and Systems*, pp. 449-452, 2002.

[73] A. Chandra and K.Chakrabarty, "Reduction of SOC test data volume, scan power and testing time using alternating run-length codes," in *Proc. of39th annual Design Automation Conference*, pp. 673-678, 2002.

[74] M. Tehranipoor, M. Nourani, and K. Chakrabarty "Nine-coded compression technique for testing embedded cores in SoCs.," *IEEE transactions on very large scale integration (VLSI) systems,* vol. 13, no. 6, pp. 719-731, 2005.

[75] X. Ruan and R. Katti "An efficient data-independent technique for compressing test vectors in systems-on-a-chip," in *Proc. ofIEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, pp. 6, 2006.

[76] H. Yuan et al, "Test data compression for system-on-a-chip using count compatible pattern run-length coding," *Journal of Electronic Testing,* vol. 30, no. 2, pp. 237-242., 2014.

[77] L. J Lee et al., "2n pattern run-length for test data compression.," *IEEE Trans Comput Aided Des Integr Circuits Syst,* vol. 31, no. 4, p. 644–648, 2012.

[78] T.B. Wu, H.Z. Liu and P.-X. Liu, "Efficient test compression technique for SoC based on block merging and eight coding," *Journal of electronic testing,* vol. 29, no. 6, pp. 849-859, 2013.

[79] S. Sivantham. et al., "Enhancement of test data compression with multistage encoding", *Integration, the VLSI Journal,* vol. 27, no. 4, pp. 499-509, 2014.

[80] V. Iyengar, K. Chakrabarty and E. J. Marinissen, "Test access mechanism optimization, test scheduling, and tester data volume reduction for system-on-chip", *IEEE Transactions on Computers,* vol. 52, no. 12, pp. 1619-1632, 2003.

[81] K. Chakrabarty, "Optimal Test Access Architectures for system on chip," *ACM Transactions on Design Automation of Electronic Systems (TODAES),* vol. 6, no.(1), pp. 26-49, 2001.

[82] S. K. Goel and E. J. Marrinissen, "SOC Test Architecture Design for Efficient Utilization of Test Bandwidth", *ACM Trans. Des. Autom. Electron. Syst.,* vol. 8, no. 4, pp. 399-429, oct 2003.

[83] [Online]. Available: http://grouper.ieee.org/groups/1500/.

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

[85] E. J. Marinissen et al, "On IEEE P1500’s standard for embedded core test," *Journal of electronic testing,* vol. 18, no. 4, pp. 365–383, 2002.

[86] V. Iyengar, K. Chakrabarty and E. J. Marinissen, "Test Wrapper and Test Access Mechanism Co-Optimization for System-on-Chip," *Journal of electronic testing ,* vol. 18, no. 2,pp. 213-230, April 2002.

[87] Q. Xu and N. Nicolici, "Wrapper design for testing IP cores with multiple clock domains," in *Proc. of Design, Automation and Test in Europe Conference and Exhibition*, pp. 416-421, Europe, 2004.

[88] S. Koranne, "A novel reconfigurable wrapper for testing of embedded core-based SOCs

and its associated scheduling algorithm," in *Frontiers in Electronic Testing*, springer, pp. 51-70,2005.

[89] S.Koranne, "Design of reconfigurable access wrappers for embedded core based SoC test," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* , vol. 11, no. 5, pp. 955-960, 2003.

[90] E. Larsson and Z. Peng , "A Reconfigurable Power Conscious Core Wrapper and its Application to System-on-Chip Test Scheduling," *Journal of electronic testing,* vol. 24, no. 5, pp. 497–504, October 2008.

[91] N. Nicolici and B. M. Al-Hashimi, "Multiple scan chains for power minimization during test application in sequential circuits," *IEEE Transactions on Computers,* vol. 51, no. 6, pp. 721-734, 2002.

[92] T. Moriyasu and S. Ohtake, "A method of one-pass seed generation for LFSR-based deterministic/pseudo-random testing of static faults," in*Proc. of 16th Latin-American Test Symposium (LATS)*, pp. 1-6, Puerto Vallarta, 2015.

[93] A. Sehgal. et. al., "Hierarchy-aware and area-efficient test infrastructure design for core-based system chips," in *Proc. of The Design Automation & Test in Europe Conference,*pp. 1-6, Munich, 2006.

[94] K. Kim and K. K. Saluja,"Low-Area Wrapper Cell Design for Hierarchical SoC Testing.," vol. 25, no. 6, pp. 347, 2009.

[95] B.Noiaand K. Chakrabarty "Test-wrapper optimisation for embedded cores in through-silicon via-based three-dimensional system on chips," *IET Computers & Digital Techniques* , vol. 5, no. 3, pp. 186-197, 2011.

[96] Y.Q. Cheng et al., "TSV Minimization for Circuit—Partitioned 3D SoC Test Wrapper Design.," *Journal of computer science and technology,* vol. 28, no. 1, pp. 119-128, 2013.

[97] S. K. Roy et al, "Optimizing test wrapper for embedded cores using TSV based 3D SOCs," in *Proc. of IEEE Computer Society Annual Symposium on VLSI (ISVLSI),* pp. 31-36, Chennai, 2011.

[98] V. lmmaneni and S. Raman "Direct access test scheme-design of block and core cells for embedded ASICs.," in *Proc. of International Test Conference*,pp. 488-492, Washington, DC, 1990.

[99] N. A. Touba and B. Pouya, "Using partial isolation rings to test core-based designs," *IEEE Design & Test of Computers,* vol. 14, no. 4, pp. 52-59, 1997.

[100] L. Whetsel, "An IEEE 1149.1-Based Test Access Architecture for ICs with Embeddded Cores.," in *Proc. of IEEE International Test Conference,* pp. 69, Washington, DC, 1997.

[101] B.Pouya and N. A. Touba, "Modifying user-defined logic for test access to embedded cores," in *Proc. of International test conference*, pp. 60-68, washington DC, 1997.

[102] P. Harrod, "Testing reusable IP-a case study," in *Proc. of International Test Conference*, pp. 493-498, Atlantic City, 1999.

[103] F. P. M. Beenker et al, "Macro testing: unifying IC and board test," *Design & Test of Computers,* vol *3,* no.6*,* pp. 26-32, 1986.

[104] F. P. M. Beenker et. al., "Macro testability; The results of production device applications," in *Proc. ofInternational Test Conference*, 1992, p232.

[105] I. Ghosh, N. K. Jhat and S. Dey "Low overhead design for testability and test generation technique for core-based systems-on-a-chip," in *Proc. ofInternational Test Conference*, pp. 50-59, Washington, DC, 1997.

[106] I. Ghosh, S. Dey, and N. K. Jha, "A fast and low-cost testing technique for core-based system-chips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 19, no. 8, pp. 863-877, 2000.

[107] K. Chakrabarty , R. Mukherjee and A. Exnicios, "Synthesis of transparent circuits for hierarchical and system-on-a-chip test," in *Proc. ofInternational Conference on VLSI Design*, pp. 431-436, Bangalore, 2001.

[108] T. Yoneda et al, "Area and time co-optimization for system-on-a-chip based on consecutive testability," in *Proc. ofInternational test conference*, pp.415-422, 2003.

[109] S. Hwang, J. A Abraham, “Test data compression and test time reduction using an embedded microprocessor”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 11, no. 5, pp. 853-862, 2003

[110] M. Sugiharat, H.Date and H. Yasuura, "A novel test methodology for core-based system LSIs and a testing time minimization problem," in *Proc. of International test conference*, pp. 465-472, Washington DC, 1998.

[111] V. Iyengar, K. Chakrabarty and E. J. Marinissen, "Efficient Wrapper/TAM co-

optimization for large SOCs," in *Proc. of Conference and Exhibition On Design, Automation and Test in Europe*, pp 491, 2002.

[112] S. Koranne, "On test scheduling for core-based SOCs," in *Proc. of7th Asia and South Pacific Design Automation Conference and 15h International Conference on VLSI Design*, pp 505, Bangalore, 2002.

[113] S. Koranne, "Formulating SoC test scheduling as a network transportation problem," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* , vol. 21, no. 12, pp. 1517-1525, 2002.

[114] S. K. Goel. and Erik. Jan. Marinissen, "Cluster-based test architecture design for system-on-chip," in *Proc. of20th IEEE VLSI Test Symposium*, pp. 259-264, 2002.

[115] S. K. Goel. and E. J. Marinissen, "A novel test time reduction algorithm for test architecture design for core-based system chips," in *Proc. ofThe Seventh IEEE EuropeanTest Workshop*, pp. 7-12, 2002.

[116] C.Giri,S. Sarkar and S. Chattopadhaya "A genetic algorithm based heuristic technique for power constrained test scheduling in core-based SOCs," in *Proc. ofIFIP International Conference on Very Large Scale Integration*, pp. 320-323, Atlanta, GA, USA, 2007.

[117] W. Zou et al ,"SOC test scheduling using simulated annealing," in *Proc. of21st VLSI Test Symposium*pp. 325-330, 2003,.

[118] J.-H. Ahn and S. Kang, "SoC test scheduling algorithm using ACO-based rectangle packing.," in *Proc. ofInternational Conference on Intelligent Computing.*, pp. 665-660, Berlin Heidelberg, 2006.

[119] Yu Huang et al., "Resource allocation and test scheduling for concurrent test of core-based SOC design," in *Proc. of10th Asian Test Symposium*, pp. 265-270, Kyoto, 2001.

[120] Yu Huang et al., “Static pin mapping and SOC test scheduling for cores with multiple test sets ,” in *Proc. ofInternational Symposium on quality Electronic Design*, pp. 99-104, 2003.

[121] V. Iyengar et al ,"Test resource optimization for multi-site testing of SOCs under ATE memory depth constraints," in *Proc. ofIEEE International Test Conference*, pp. 1159-1168, 2002.

[122] Y. Huang. et al., "Optimal core wrapper width selection and SOC test scheduling based

on 3-D bin packing algorithm," in *Proc. ofInternational Test Conference*, pp. 74-82, 2002.

[123] H. M. Harmanani and H. A. Salamy, “A simulated annealing algorithm for system-on-chip test scheduling with Power and precedence Constraints,”*International Journal of Computational intelligence and applications*,vol 6, no. 4, pp. 511-530, 2006.

[124] E. Larsson, Z. Peng and G. Carlsson "The design and optimization of SOC test solutions," in *Proc. ofIEEE/ACM International Conference on Computer Aided Design*, pp. 523-530, San Jose, CA, USA, 2001.

[125] C.P. Su and C.W. Wu, "A graph-based approach to power-constrained SoC test scheduling," *Journal of Electronic Testing ,* vol. 20, no. 1, pp. 45-60, 2004.

[126] D. Zhao and S. Upadhyaya, "Power constrained test scheduling with dynamically varied TAM," in *Proc. ofVLSI test synmposium*, pp. 273-278, 2003.

[127] Q. Xu and N. Nicolici, "On reducing wrapper boundary register cells in modular SOC testing," in *Proc. ofInternational test conference*, pp. 622-631, 2003.

[128] V. Iyengar et al, "Design and optimization of multi-level TAM architectures for hierarchical SOCs," in *Proc. of21st VLSI Test Symposium*, pp 299-304, 2003.

[129] L. Jiang, L. Huang and Q. Xu, "Test architecture design and optimization for three-dimensional SoCs," in *Proc. ofDesign, Automation & Test in Europe Conference & Exhibition*, pp. 220-225, Nice, 2009.

[130] L. Jiang et al, "Layout-driven test-architecture design and optimization for 3D SoCs under pre-bond test-pin-count constraint," in *Proc. ofIEEE/ACM International Conference on Computer-Aided Design-Digest of Technical Papers*, pp. 191-196, San Jose, CA, 2009.

[131] C.Y. Lo et al, "SOC test architecture and method for 3-D ICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 29, no. 10, pp. 1645-1649, 2010.

[132] E. J. Marinissen. and Y. Zorian, "Testing 3D chips containing through-silicon vias," in *Proc. ofInternational Test Conference*, pp. 1-11, Austin, TX, 2009.

[133] S. K. Millican and K. K. Saluja, "Optimal Test Scheduling of Stacked Circuits under Various Hardware and Power Constraints," in *Proc. of28th International Conference on VLSI Design*, pp. 487-492, Bangalore, 2015.

[134] B. SenGupta, U. Ingelsson, E. Larsson, "Scheduling tests for 3D stacked chips under power constraints.," *Journal of electronic testing* , vol. 28, no. 1, pp. 121-135, 2012.

[135] W.D.Tseng and L.J. Lee, "Test data compression using multi-dimensional pattern run-length codes," *Journal of Electron Test,* vol. 22, no. 6, pp. 393–400, 2011.

[136] L. J. Lee,W. D. tseng and R. B. lin, "An internal pattern run-Length methodology for slice encoding," *ETRI Journal,* vol. 33, no. 3, pp. 374–381, 2011.

[137] C. Liu et al, "Test scheduling for network-on-chip with BIST and precedence constraints," in *Proc. ofInternational Conference on Test*, pp. 1369-1378, 2004.

[138] C. Liu and V. Iyengar "Test scheduling with thermal optimization for network-on-chip systems using variable-rate on-chip clocking.," in *Proc. ofConference on Design, Automation and Test in Europe*, pp. 6, Germany, 2006.

[139] X. T. Tran et al, "Design-for-test approach of an asynchronous network-on-chip architecture and its associated test pattern generation and application," *IET computers & digital techniques* , vol. 3, no. 5, pp. 487-500, 2009.

[140] E.J. Marinissen et al, "A set of benchmarks for modular testing of SOCs," in *Proc. ofIEEE International Test Conference (ITC)*, pp. 519-528, Baltimore, MD, 2002.

[141] J. Pouget, E. Larsson, and Z. Peng, "Multiple-constraint driven system-on-chip test time optimization," *Journal of electronic testing,* vol. 21, no. 6, pp. 599-611, 2005.

[142] P. P. Pande et al, "Design, synthesis, and test of networks on chips," *IEEE Design & Test of Computers,* vol. 22, no. 5, pp. 404-413, 2005.

[143] H. Yi, S.Park, and S. Kundu, "On-chip support for NoC-based SoC debugging," *IEEE Transactions on Circuits and Systems,* vol. 57, no. 7, pp. 1608-1617, July 2010.

[144] X.T.Tran et al, "A DFT Architecture for Asynchronous Networks-on-Chip," in *Proc. ofEleventh IEEE European Test Symposium*, pp. 219-224, Southampton, 2006.

[145] E. Cota et al, "The impact of NoC reuse on the testing of core-based systems," in *Proc. of21st VLSI Test Symposium*, pp. 128-133, 2003.

[146] E. Cota et al, "Power-aware NoC Reuse on the Testing of Core-based Systems," in *Proc. ofInternational Test Conference,*pp. 612-621, 2003.

[147] F. A. Hussin, T. Yoneda, and H. Fujiwara, "Optimization of NoC wrapper design under bandwidth and test time constraints.," in *Proc. of12th IEEE European Test Symposium*, pp. 35-42, Freiburg, 2007.

[148] G. Mali et al, "Non-preemptive test scheduling for Network-on-Chip(NoC) based systems by reusing NoC as TAM," in *Proc. of2010 IEEE Asia Pacific Conference on Circuits and Systems*, pp. 268-271, Kuala Lumpur, 2010.

[149] J.H. Ahn and S. Kang, "Test scheduling of NoC-based SoCs using multiple test clocks," *ETRI journal* , vol. 28, no. 4, pp. 475-485, 2006.

[150] C. Liu et al, "Power-aware test scheduling in network-on-chip using variable-rate on-chip clocking," in *Proc. of23rd IEEE VLSI Test Symposium*, pp. 349-354, 2005.

[151] É. Cota and C. Liu, "Constraint-driven test scheduling for NoC-based systems.," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 25, no. 11, pp. 2465-2478., 2006.

[152] K. Manna, S. Singh, S. Chattopadhyay, I. Sengupta, "Preemptive Test Scheduling for Network-on-Chip Using Particle Swarm Optimization," in *Proc. ofVLSI Design and test*, pp. 74-82, 2013.

LIST OF PUBLICATIONS

- H. Vohra and A. Singh,” Optimal Selective count compatible Runlength Encoding for SOC Test data compression,” *Journal of Electronic testing*, vol. 32, no. 6, pp. 735-747,2016.
- H. Vohra and A. Singh, “Survey of system on chip modular test Approach,” *Journal of VLSI design tools and technology*, vol 6, no. 3, pp. 56-70, 2016.

List of communicated papers

- H. Vohra and A. Singh, “Hierarchical block merging based test data compression technique,” Communicated to *IET Computers and digital techniques*(under review).
- H. Vohra and A. Singh, “Test architecture optimization algorithms for coarse grain partitioned three dimensional System on chip” *IET journal of software*