TESTABILITY INFRASTRUCTURE

FOR SYSTEMS-ON-CHIP

by

MOHSEN NAHVI

B.Sc. University of Manchester Institute of Science and Technology, 1989
M.Sc. University of Manchester Institute of Science and Technology, 1990

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES
(ELECTRICAL AND COMPUTER ENGINEERING)

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA
June 2004

Abstract

Relying on external *automatic test equipment* (ATE) resources is insufficient for the new paradigm of billion-transistor core-based *System-on-Chip* (SoC) designs. Embedded testers that take over some functionality of these ATEs are increasingly deemed essential. To achieve high-quality test and reduce cost, these embedded infrastructures need to perform deterministic tests and exploit the advantages of *automatic test pattern generation* (ATPG) test vector sets. This thesis proposes an embedded testing infrastructure that leverages the potentials of the classical embedded testing in the form of *Built-in Self-Test* (BIST). However, unlike BIST, the methodology of this thesis is based on the conventional scan/ATPG approach. This novel methodology partitions test resources to embed the test application and test results analysis on-chip while keeping the ATPG test vector files off-chip. The proposed infrastructure was implemented on silicon and experimental area and test time results are reported. Using the methodology of this thesis, a high-quality deterministic test, with reduced overall test time through ideal multi-site testing, can be achieved.

Modular, flexible, and systematic test architectures are also deemed essential in SoC tests. The conventional testing paradigm requires a direct connection between a tester and the *circuit under test* (CUT). This arrangement undermines the modularity in the test

architecture by tightly coupling its elements. This thesis proposes to de-couple test data processing and communication to lower test cost. To that end, a novel systematic and indirect test architecture that is based on network-oriented protocols is proposed. In this new architecture, test stimuli and expected results for digital cores are formatted into new protocols and then encapsulated into packets. These packets are augmented with control and address bits allowing them to be autonomously transmitted to their destination through a switching infrastructure. Finally, embedded autonomous blocks at each core are used for applying the test and comparing the results. In this way, the methodology of this thesis facilitates test cycle automation and eliminates the need for control lines. This results in better utilisation of available resources. A first implementation of this new architecture and its area and test time impact are presented.

Table of Contents

List of Tables

List of Figures

Acronyms

ASIC	Application-Specific Integrated Circuit
ASM	Algorithmic State Machine
ATE	Automatic Test Equipment
ATPG	Automatic Test Pattern Generation
BIST	Built-in Self-Test
BSR	Boundary Scan Register
CAS	Core Access Switch
CTL	Core Test Language
CUT	Circuit under Test
DAST	Dedicated Autonomous Scan-based Testing
DFT	Design for Testability
DSM	Deep Sub-Micron
EARA	Embedded Autonomous Results Analyzer
EAS	Embedded Autonomous Sequencer
EDA	Electronic Design Automation
FSM	Finite State Machine
GALS	Globally Asynchronous and Locally Synchronous

IC	Integrated Circuit
I-IP	Infrastructure IP
ILP	Integer Linear Programming
IM	Interface Matching
I/O	Input/Output
IP	Intellectual Property
ITRS	International Technology Roadmap for Semiconductors
LAN	Local Area Network
LAS	Logical Address Space
LFSR	Linear Feedback Shift Register
LSB	Least Significant Bit
NIMA	Network-oriented Indirect and Modular Architecture
NoC	Network on Chip
PAS	Physical Address Space
PI	Primary Input
PLL	Phase Locked Loop
PO	Primary Output
SC	Scan Chain
SE	Scan-cell Element
SECT	Standard for Embedded Core Test
SI	Scan Input

SO	Scan Output
SoC	System-on-Chip
TAM	Test Access Mechanism
TAP	Test Access Port
TLM	TAP Link Module
TP	Test Pattern
TS	Test Select
UDL	User-Defined Logic
VLSI	Very Large Scale Integrated Circuit
WAN	Wide Area Network
WBR	Wrapper Boundary Register
WIR	Wrapper Instruction Register
WPC	Wrapper Parallel Control
WPI	Wrapper Parallel Input
WPO	Wrapper Parallel Output
WSC	Wrapper Serial Control
WSI	Wrapper Serial Input
WSO	Wrapper Serial Output

Acknowledgements

I am very grateful to my supervisor, Professor André Ivanov, for his friendship, support, guidance, and encouragement in the course of my graduate studies. I have been truly fortunate to have the pleasure of working with him for the past few years while conducting the research of this thesis. André has contributed a great deal to the manner in which I have conducted the work of this thesis, and to the development of my ability to write technical papers and be articulate when presenting my work. I thank him most sincerely for all his contributions to my professional life.

My sincere thanks go to my dear wife, Elham. Her love and support have always been unconditional and heart warming. She never ceases to amaze me with her ability to be so loving and caring. She has contributed so much to the successful completion of my graduate studies through her constant support, kindness, and love. Thank you Elham.

I am also grateful to Professor Resve Saleh and Dr. Steve Wilton who have helped and encouraged me as my supervisory committee. My thanks are also extended to the many great people working in the System-on-Chip (SoC) lab, and, in particular, to Roozbeh Mehrabadi and Victor Aken'Ova for their readiness to help in any way. I am also grateful to Doris Metcalf, and Shahab and Ameneh Ghoreishi.

Finally, I would like to thank Canadian Microelectronics Corporation (CMC), Micronet, PMC-Sierra, Genum Corporation, and the University of British Columbia for their financial support.

To Elham, for her love and kindness.

Chapter 1
Introduction

The semiconductor industry has witnessed an astonishing rate of growth in the past 40 years. The fundamental element contributing to this growth has been the continuous reduction in the size of the transistors on a chip. This trend has been doubling the number of transistors on an *integrated circuit* (IC) every eighteen months or so, a well-known trend referred to as Moore's law. According to the 2003 edition of International Technology Roadmap for Semiconductors (ITRS), this rate of growth could be sustained for at least fifteen more years, resulting in even more complex *very large scale integrated* (VLSI) circuits [1].

However, random failures in the process of fabricating an IC cannot be avoided and these failures result in defective chip circuitry. Hence, it is essential to test semiconductor ICs after fabrication. This post fabrication test is intended to guarantee,

with a high probability, the absence of possible defects in the chips. In this way, the post fabrication test verifies a chip's reliability before it is used in a system [2][3]. ICs are tested in different stages (sometimes referred to as *test insertions*). Based on the intended goals, test insertions can target the parametric, temperature stressing, logical, functional, and timing integrity of the integrated *circuit under test* (CUT). Testing VLSI circuits can be a very expensive and difficult process and will grow in its cost and complexity as ICs follow Moore's law [4]. *Design for testability* (DFT) techniques are, hence, essential to facilitate the testing of VLSI chips and reduce test cost [2][4]. In addition to certain design guidelines, which enhance the testability of a CUT, DFT techniques generally consist of added hardware or circuitry that improves *observability* and *controllability* of internal nodes of the CUT.

The focus of this thesis is on DFT techniques for digital circuits. In particular, this thesis proposes two effective and novel testability infrastructures for testing embedded digital blocks of *system-on-chip* (SoC) designs. These unique infrastructures combine the advantages of the current SoC test solutions in innovative ways in order to accommodate different trends and requirements in addressing the SoC test challenges. The solutions developed in this thesis lower test cost by partitioning of the test resources, performing embedded deterministic test, and enabling ideal multi-site parallel testing. Moreover, when compared to existing solutions, the proposed techniques described in this thesis exhibit modular, systematic, and flexible architectures that result in improved quality, productivity, and diagnostic capability of SoC post fabrication test. Since the focus of

this thesis is on DFT techniques for digital blocks of SoC designs, unless explicitly mentioned otherwise, this thesis uses the terms *test* and *DFT* as applied to digital components of an SoC design.

Section 1.1 of this chapter outlines the current test strategies. Section 1.2 motivates and presents the research goals of this thesis. Section 1.3 outlines the contribution of this thesis in terms of its original and novel work. Finally, Section 1.4 provides the organization of the rest of this thesis.

1.1 Current Test Strategies

There are two distinctive testing strategies for digital circuits: external testing of a chip using *automatic test equipment* (ATE); and embedded self-testing in the form of logic *Built-in Self-Test* (BIST).

In ATE-based testing, *automatic test pattern generation* (ATPG) tools are used to create deterministic test vector sets. Using the ATPG test vector sets, the ATE applies the test stimuli to the CUT and collects the test results to compare them with the expected values. Effectively, in the ATE-based testing strategy, the communication and the application of test data are closely coupled. In other words, the ATE acts as both a test source and a sink and it is assumed that the tester establishes a data path between itself and a core such that the tester has direct control of features of the core's DFT. Using this data path, the tester applies the test data to the core before it collects and observes the results. Figure 1-1 shows an example of a simplified test arrangement with six data lines

–also referred to as *test access mechanism* (TAM)– and two control lines. In Figure 1-1, the data lines are grouped into two TAMs such that the tester can have direct control of the scan chains of each core, while minimising test time.

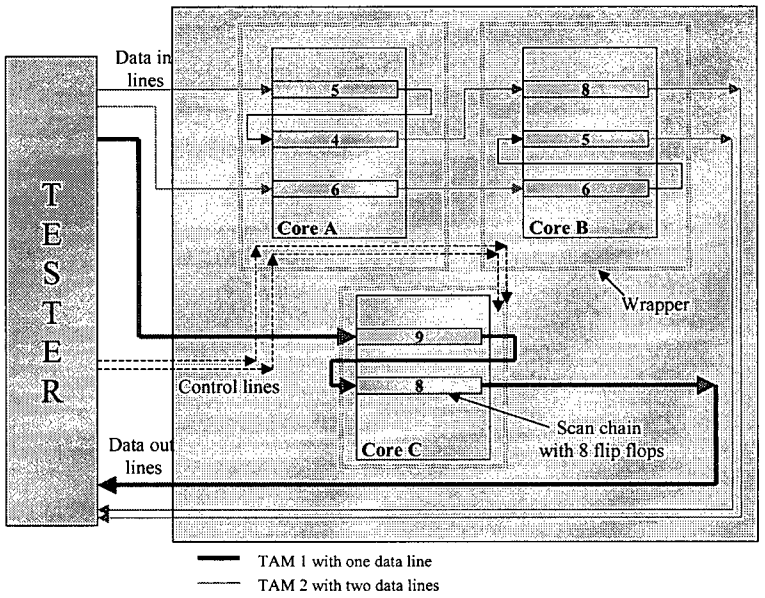


Figure 1-1: An example of a simplified SoC test architecture.

In contrast to ATE-based testing strategy, typical logic BIST techniques rely on pseudo-random test patterns. In a typical BIST arrangement, embedded blocks generate the test stimuli and apply them to the CUT. Test results are compacted into signatures and these signatures are compared to the expected ones. In effect, the BIST strategy can be considered as the opposite of the ATE-based strategy, as both the source and the sink of the test architecture are on-chip.

1.2 Research Goals

1.2.1 Embedded Deterministic Scan-based Testing

One goal of this thesis is the development of a new methodology for embedded deterministic testing of digital cores in SoC designs. In this thesis, this novel method is referred to as *dedicated autonomous scan-based testing* (DAST). With DAST, the goal is to leverage the potentials of logic BIST in SoC test, without paying the potential penalties of BIST. To that end, the DAST methodology uses embedded test resources that use ATPG test stimuli and expected results, as illustrated in Figure 1-2. Effectively, in the DAST methodology, DFT-tester resources are partitioned, keeping the ATPG test vector sets off-chip while embedding the control and the observation functions at the cores. As such, DAST is a novel embedded deterministic testing infrastructure, where both the test stimuli and the expected test results of the ATPG test vectors are used for on-chip ATPG-grade testing. In the unique methodology of DAST, the embedded resources require the communication of ATPG-based test stimuli and expected results through global interconnects that act as a communication link. This communication link can be any generic TAM or the novel on-chip network-oriented switching fabric developed in this thesis as outlined in Section 1.2.2.

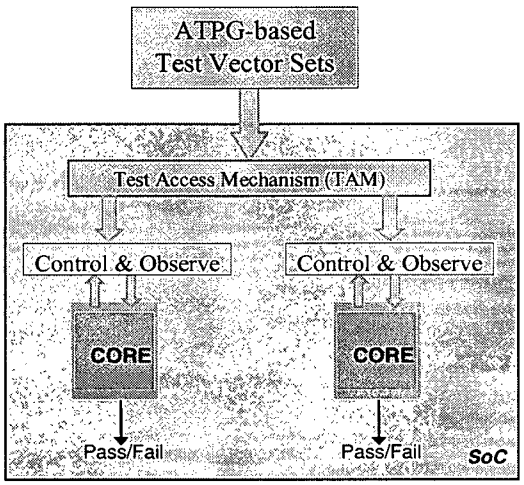


Figure 1-2: DAST concept.

The partitioning of the test resources in DAST lowers DFT-testers cost. This, in turn, reduces total production costs. However, unlike conventional BIST, the partitioning of the test resources in DAST does not require any changes to embedded cores and only requires minimal modification of the ATPG test flow. In this way, DAST results in flexible and modular test architectures that facilitate the reuse of test resources and achieving a highly productive test scheme.

DAST benefits from the advantages of both the BIST and the ATPG-based testing as, with DAST, ATPG test vector sets are used alongside embedded blocks. Hence, DAST enables testing to be performed at-speed and deterministically, ensuring high test quality with minimal test time. Moreover, contrary to the BIST methodology, the diagnostic information in DAST is not lost, as no compaction is implemented in the form of a signature.

Owing to the fact that both the test stimuli and the test results are sent to the cores, DAST, theoretically, achieves an ideal multi-site testing strategy for ATPG-based testing. Hence, given unlimited power for the external test resources, a tester, with a limited number of test channels that are equal to the number of the test pins of one chip, can test an unlimited number of chips in parallel. This ideal case is achieved as the same data set is sent to all chips, enabling sharing of test channels between all the chips.

1.2.2 Test Network-on-Chip

As a second objective, this thesis proposes and develops the architecture of a novel on-chip test network. In this thesis, this architecture is referred to as a *Network-oriented, Indirect and Modular test Architecture* (NIMA). In NIMA, the tester is assumed to lack direct control over a core's DFT and is de-coupled from the TAM, and the core and its wrapper. This de-coupling facilitates partitioning of an external tester into its external and embedded components, and enhances the modularity and scalability of the test architecture. In addition, NIMA presents the concept of a highly hierarchical, modular, and flexible test architecture that can address high-productivity requirements of testing SoC designs and, ultimately, help in lowering test cost. Figure 1-3 illustrates combined NIMA and DAST concepts as a block diagram.

In NIMA, test data, comprised of test stimuli and test results, is formatted into a new protocol and augmented with control bits. In this way, control lines are eliminated in NIMA and the TAM is only comprised of test signal lines that communicate both the test

data and the control bits. Hence, using NIMA, the effective test time is reduced, as the TAM lines are used more efficiently.

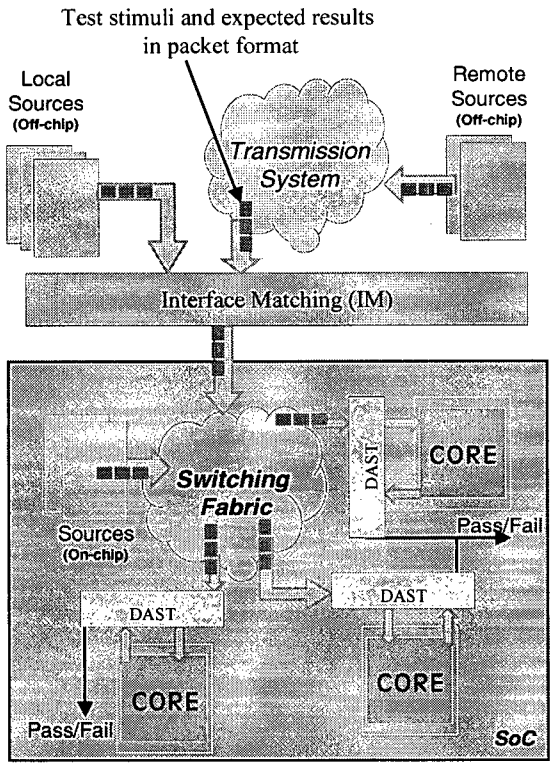


Figure 1-3: Combined NIMA and DAST concepts.

The reformatted test data and the control bits in NIMA are then encapsulated into packets such that the test data can be forwarded to its core autonomously. This autonomous behaviour of the NIMA architecture serves well in automating test cycles, a

requirement of high-productive test. Moreover, as no interaction from the tester is required in this step, a low-cost test template is achieved.

Finally, at a core, embedded autonomous blocks decode NIMA's packets and retrieve the test data. These embedded blocks apply the test stimuli to the core and compare the responses with expected results, identifying any error. This process can be handled using the DAST approach. In using autonomous embedded blocks, NIMA achieves the main requirements of SoC testing. More specifically, NIMA serves as a methodology that enables high quality at-speed testing.

Moreover, an important system-level design challenge identified in [4] is due to *system complexity*. It is predicted that this challenge will lead to forcing a focus on communication rather than computation in the next ten years. It is stated in [4] that:

> *"At 65 nm and below, communication architectures and protocols for on-chip functional processing units will require significant change from today's approaches. As it becomes impossible to move signals across a large die within one clock cycle or in a power-effective manner, or to run control and dataflow processes at the same clock rate, the likely result is a shift to asynchronous (or, globally asynchronous and locally synchronous (GALS)) design style. In such a regime, islands of self-timed functionality communicate via network-oriented protocols."*

Therefore, an underlying assumption of this thesis is that future SoC designs will include a switching fabric coupled with network-oriented protocols, for core interconnect. Hence, this thesis proposes the first methodology and test architecture that enables the reuse of such a fabric for testing cores.

Evidently, the concept of having a *network-on-chip* (NoC) that uses network-centric protocols and a switching fabric for data communication on a chip is very new. Most relevant works are geared towards the problem of core interconnection [5][6][7][8]. The motivations for many of these works are the non-scalability of global wire delays and their effect on global synchronization, the degradation of bus electrical performance with every attached unit, and the need for special error control mechanisms because of the unreliable transmission medium. Based on the open literature, no work on the subject of test architectures that use network-oriented approaches has been reported to date, with the possible exception of [9] and [10]. Reference [9] studies the impact of reusing an NoC for testing core-based systems, and reports the results of such reuse on test time based on two proposed scheduling mechanisms. When reusing an NoC, the work in [10] studies the effect of power consumption on the test time and test scheduling. Hence, the studies in [9] and [10] focus on the general concept of test data packetization and its impact on test time, and they do not address the methodology and the implementation of the test architecture required for NoC reuse. Therefore, this thesis is the first to propose, develop, and empirically validate the methodology and implementation of a test network on-chip

that can use its unique switching fabric or reuse other networks on-chip as they become available.

1.3 Contributions

In short, the principal contributions of this thesis are listed here.

- The DAST methodology of Chapter 3 is developed as a novel on-chip deterministic tester. In addition, DAST is developed such that it partitions the functionality of external testers into two distinctive operations of *data delivery* and *control/observation*. This partitioning enables embedding of the latter functionality at cores of an SoC design.
- The algorithms of Section 3.4.1 are developed to enable the full use of ATPG in the embedded testing methodology of DAST. The use of ATPG results in deterministic high-quality embedded testing in DAST, and the algorithms of Section 3.4.1 can be integrated into the design flow of a core to enhance its productivity.
- An efficient and simple implementation of DAST hardware is presented in Section 3.4.2. This implementation enables automatic and flexible DAST hardware development, as it uses hardware description language with generic parameters that can be set for any cores. Moreover, the implementation avoids design iterations, as it only requires the interface information of cores and does not alter cores or their functionality.

- DAST is characterized in terms of its test time models and these models are presented in Section 3.5.
- A novel on-chip test network is proposed and developed in Chapter 4. NIMA methodology of Chapter 4 is developed to enable a hierarchical systematic test architecture that can reuse an on-chip switching fabric with network-oriented protocols.
- NIMA is developed as a 3-layer communication network to enable modularity and scalability of test architectures. In this way, the design of NIMA facilitates test resource partitioning.
- A novel dynamic addressing mechanism to simplify the design of NIMA's network layer is presented in Section 4.4.3.
- A simple and efficient implementation of NIMA network layer is presented in Section 4.6.2, while DAST implementation is altered to work as the application layer of NIMA. To facilitate NIMA flexibility and the automation of the integration of NIMA in an SoC design flow, this implementation uses a hardware description language and generic values for different parameters of NIMA's design.
- NIMA is characterized in terms of its test time models of Section 4.7.2.
- The methodologies of NIMA and DAST are empirically validated to fit into a typical ASIC design flow, and NIMA and DAST are implemented on silicon. This

validation supports the fact that these methodologies can easily be integrated in a design flow.

1.4 Thesis Organization

Chapter 2 of this thesis, after the introduction of the SoC design methodology, reviews the test challenges in SoC designs. It then discusses the test trends and solutions in addressing these test challenges. Finally, Chapter 2 reviews the current test architectures in detail.

Chapter 3 reviews the concept of scan-based testing and identifies the components of scan-based testing, i.e., *data delivery* and *control/observation.* Chapter 3 then presents the concept of DAST in detail, provides the design flow, and presents an implementation for DAST. It also presents the experimental results when applying the DAST methodology to a number of benchmark circuits.

In Chapter 4, further motivation for NIMA is presented. Chapter 4 also provides NIMA's concept and architecture, and suggests an implementation of NIMA where DAST is integrated into the architecture. It also presents the experimental results of using the NIMA methodology in a number of benchmark circuits.

Finally, Chapter 5 concludes the thesis by summarizing the results and thesis contributions and provides direction for future work.

Chapter 2
Background and Motivations

This Chapter reviews the background and outlines the test challenges in SoC designs. The required test trends to address SoC test challenges are then discussed. Finally, current test architectures are reviewed and the research work of this thesis is motivated.

2.1 System-on-Chip Design Methodology

Table 2-1, an extract from [4], compares the number of transistors on a chip in the year 2001 to that predicted for the year 2016.

Table 2-1: ITRS Prediction for Number of Transistors on a Chip [4]

Year	2001	2016
High-volume Microprocessors Functions per chip at introduction (Million Transistors)	193	6,184
Application Specific ICs (ASIC) Maximum functions per chip at production (Million Transistors)	714	16,326

Using the large number of transistors available on a chip, designers have already managed to place an entire electronic system on a single chip. These are referred to as *systems-on-chip* or *system chips* for short. An example of an SoC is illustrated in Figure 2-1, where different embedded functional blocks (or embedded cores) of the system are conceptually shown [11]. As the number of the transistors increases, more complex systems, utilising hundreds of embedded cores, will be placed on a single chip. Apart from the availability of large transistor counts, there are other reasons behind this trend. One important reason is the market demand for portable products with increasing functionality [11][12], and another important reason is the need for a lower product cost [11][13].

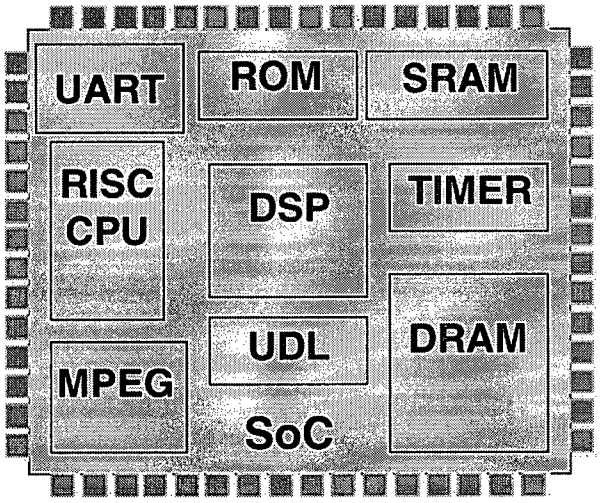


Figure 2-1: Example of a System-on-Chip [11].

The justification behind the first reason is that SoC designs are faster, more reliable, and require less power per function compared to systems on a board. This holds true as most of the communication is on-chip and, hence, there is no need for large and power-hungry drivers at the chip terminals [14]. In addition, the smaller footprint is more suitable for portable, low-weight, and low-cost products.

The rationale behind the comparatively lower cost of products using SoC chips is the fact that smaller and more compact products require fewer packages. Products with fewer packages, when compared to those with more packages, amount to comparatively simpler design process and integration. Hence, the overall product cost can be reduced significantly. In addition, simpler design process result in faster introduction of the product into the market and, hence, will generate more profit.

2.2 Reuse Paradigm

Figure 2-2, adopted from [15], illustrates potential design complexity and designer productivity. In Figure 2-2, the horizontal axis represents the year and the left hand side vertical axis shows the number of logic transistors per chip in million units. In addition, in Figure 2-2, the vertical axis on the right hand side represent a measure of designer productivity in the form of the number of thousand transistors each designer can design and verify per month in his/her designs. According to Figure 2-2, designers' productivity growth rate is equal to 21% each year. However, the number of logic transistors per chip follows Moore's law and increases at a rate of 58%. Owing to these different rates of growth, the gap between the productivity of designers and the number of available transistors is widening very rapidly.

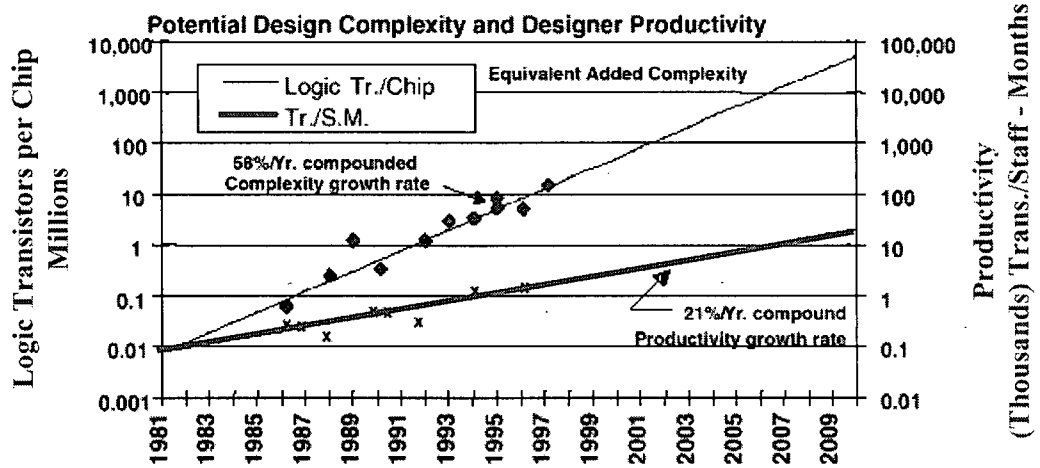


Figure 2-2: Potential design complexity and designer productivity [15].

It is becoming increasingly difficult to utilise all the transistors available on the chip effectively and still meet the time-to-market constraints. Nowadays, the very short life cycle of products compounds the problem further, as demand for a product might diminish before designers have a chance to introduce their version of the product [15][16][17][18].

To overcome the productivity gap and meet the stringent time-to-market constraints, it is not enough to put more manpower on the job. In fact, by virtue of the law of diminishing return, it is not economical. Hence, the only viable solution put forward so far is for designers to *reuse* in-house or externally acquired pre-designed/verified *intellectual properties* (IP) as embedded *cores* in SoC designs [4][12][13][17][18][19][20]. Effectively, SoC designs need to be *application-specific integrated circuits* (ASIC) that maximize reuse of IPs to improve design productivity.

The concept of reuse is not new to the semiconductor industry. In the typical design flow of systems-on-board, pre-fabricated ICs are reused. However, the difference in the case of SoC design methodology is the fact that cores to be used in SoC designs are not fabricated before they are used. Instead, cores in SoC design methodology come in some form of hardware description. Based on the method of their hardware description, there are three main categories for cores, i.e., *soft*, *firm*, and *hard* cores [18]. *Soft* cores are those functional blocks that are described in any synthesizable hardware description language such as VHDL or Verilog. *Firm* cores are typically gate-level descriptions that are targeted for a specific fabrication technology and can be optimized for speed, area,

power, etc. Finally, *hard* cores use the least flexible description method, and are the layout description –usually in a standard industry format such as GDSII– of optimized *firm* cores.

2.3 Test Challenges

In order to satisfy the reliability requirements of fabricated ICs, it is imperative to test these chips before they are shipped out to potential customers. However, there are many challenges in testing SoC chips, and many experts believe that testing SoC chips will be the bottleneck of future designs, if issues of DFT for SoCs are not addressed [4][21][22][23]. These challenges are the result of both the SoC design methodology itself [4][20][21][24][25][26], and the *deep sub-micron* (DSM) technology used to fabricate SoC chips [24][27][28][29]. This section summarizes these challenges.

2.3.1 Test Challenges in DSM Technology

The DSM fabrication process, shrinking chip geometries, the continuity of Moore's law, aggressive time-to-market and time-to-volume requirements, and finally, the need for lower total cost are all contributing to the test challenges in the DSM era. Based on different test requirements, some of the important test issues due to the DSM technology can be categorized into the following five groups [4][27][28][29][30].

High-quality Test

Very high-quality test, which achieves high fault coverage, is essential in manufacturing highly reliable chips. To guarantee high fault coverage, in addition to the stuck-at-fault

models, new fault models are needed to facilitate the detection of static and dynamic defects in the DSM fabrication processes [4][29][30]. Therefore, at-speed testing, using transition and path delay fault models in conjunction with scan chain(s), is becoming a requirement of chip testing [31]. However, in many cases, the performance of external testers lags behind the performance of chips they intend to test. In particular, external testers cannot operate at the high frequency required for at-speed testing of many CUTs and have insufficient accuracy for pin-to-pin timing requirements [31]. Hence, these testers lack the necessary accuracy and resources to handle at-speed testing and other new fault models. Therefore, new DFT methodologies and testing techniques are needed to facilitate at-speed testing. In addition, these new DFT methodologies and testing techniques should maintain the high quality of test and be able to use a variety of fault models.

Low-cost Test

With almost exponential reduction trend in ICs fabrication cost, chips test cost will soon equal that of their fabrication, if new test and DFT methods are not devised [4]. Today's monolithic testers are not designed to take advantage of the DFT used within a chip. Moreover, the design of these testers is such that they are able to perform a variety of different tests and support a broad range of ICs [27]. However, this amounts to very expensive testers with features that may not be needed or used for many CUTs or test insertions. It also results in the requirement of a large capital investment by companies for purchasing these multi-purpose testers. In addition, owing to the rapid advancement

of ICs, this large capital investment will be outdated within a few years, adding to the test cost. Thus, an important challenge facing the test community is finding solutions to reduce test cost through new DFT techniques, redesign of testers towards low-cost, reduced-functionality testers, and design of DFT-aware testers that take advantage of CUT DFT infrastructure.

Low-volume Test Data

To maintain high fault coverage, for complex designs, a larger test data volume is inevitable. In turn, larger test data sets require more memory in testers. In addition, with larger test data sets and limited number of tester channels, test time will increase. These problems are exacerbated by the slower rate of increase in the number of a chip's primary input/output (I/O) pins when compared to that of the number of transistors on a chip. Therefore, reduction in the test data volume to lower test time by either reducing the test data set or increasing the effective bandwidth is a problem in need of solution. Note that, using low-cost testers can mitigate the problems associated with test data volume as, in these cases, test time contribution to the overall cost is reduced.

High-productivity Test

To lower the cost and keep the time-to-market and the time-to-volume of products to their minimum, the DFT insertion, the test program development, and the test process should be hierarchical and automated as much as possible [29]. Hence, DFT techniques and test resources need to be modular and flexible to facilitate hierarchical and automated

test flow. Moreover, DFT circuitry should be seamless with the normal design flow and, thus, the DFT insertion process must avoid design iteration.

Test for Diagnosis

Yield improvements of new chips are essential in lowering production costs and achieving the time-to-volume [32]. Diagnostic information helps in the rapid yield ramp up. Thus, the test flow needs to include fault diagnosis as one of its objectives in order to avoid expensive off-line and special diagnosis testing processes. To this end, test resources and DFT techniques are needed to facilitate real-time yield analysis [28].

2.3.2 Test Challenges in SoC

The *IP-* or *core-based* design methodology is creating a new style of design with its own characteristics, requirements, benefits, and challenges. One particular challenge is how to test these core-based systems. Testing of SoC designs differs from system-on-board testing. Figure 2-3, adopted from [21], illustrates the differences in the design development and the test flow between systems-on-chip and systems-on-board. In a system-on-board, each component is designed, verified, fabricated, and finally tested. The system integrator uses these tested components, and performs testing of the board and the system with the assumption that the components are fault-free. However, as illustrated in Figure 2-3, in the SoC design methodology, cores are designed and verified, but are not pre-fabricated. Hence, these cores cannot be tested before their integration into the SoC. Therefore, the system integrator is responsible for all post-manufacturing tests of both the cores and the system that uses these cores.

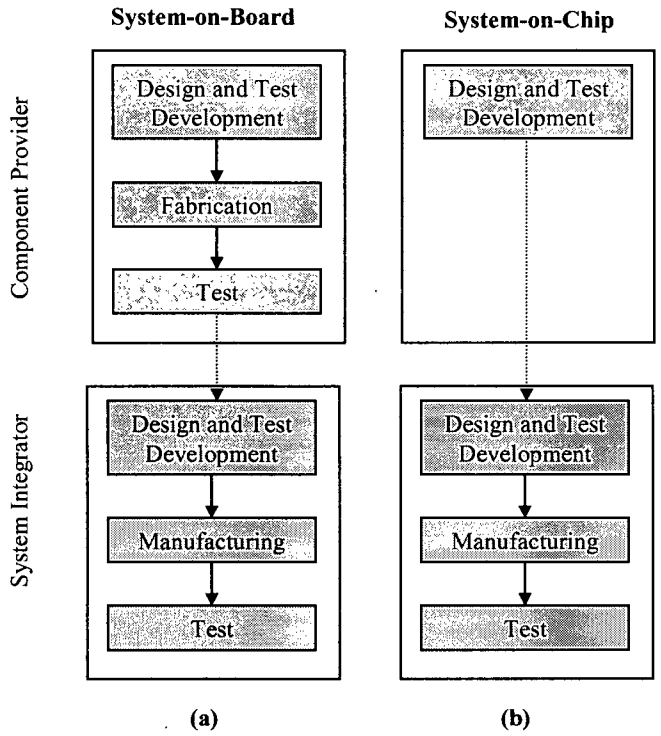


Figure 2-3: Design development and test flow for (a) System-on-Board, and (b) System-on-Chip [21].

The different test flows as explained above and illustrated in Figure 2-3, as well as having embedded cores with limited accessibility, result in unique challenges in SoC test. Moreover, stringent time-to-market and time-to-volume requirements, and the need for lower total cost also contribute to SoC test challenges. These challenges can be summarised as follows:

- Effective communication mechanisms for the test flow between different parties in the design of the SoC (note that the design of an SoC can span multiple groups in different companies which can be located in many different geographical locations);
- Accessibility of embedded cores from the primary input/output (I/O) of the chip;
- Testing different types of cores with compatible methodologies;
- Testing the *user-defined logic* (UDL) and the interconnects;
- Seamless integration of cores' DFT into the system DFT (modularity of the system test architecture);
- Having a test architecture allowing new cores to be integrated into the chip without incurring core design changes (scalability of the system test architecture);
- Designing the system DFT such that the chip will be reusable hierarchically (scalability of the system architecture).

2.4 SoC Test Trends

Test challenges as discussed in Section 2.3 call for new and advanced test methodologies for core-based SoCs. A few of the most important test trends that, according to the literature, help in addressing some or all of the challenges described in Section 2.3, are outlined in this section.

2.4.1 Test Reuse

The integration of IP blocks into an SoC results in a non-linear complexity growth for DFT and manufacturing test [4]. Hence, it is costly and inefficient to assign the design of embedded core DFTs and generation of test vectors to the system designer. It, however, is more practical to reuse pre-designed DFT schemes of the cores and their test vectors [4][18][21][24][25][33][34]. Core-based design with reuse methodology requires a new business model. In this new model, *core-providers* design and develop the IPs and *core-users* integrate them in an SoC environment. Hence, *core-providers* are responsible for the DFT techniques used in the core as well as being responsible for providing all the information to *core-users*. In this modular model, *core-users* treat individual core test programs as distinct components and integrate/schedule these components into a system test program with limited knowledge of the core's internal detail [21]. Separation of the tasks between *core-providers* and *core-users* introduces new challenges in the entire design flow. Hence, modularity and scalability of the test architecture are deemed essential to enhance the rapid development of designs and to avoid costly iterations between core-providers and core-users.

2.4.2 Systematic Test Architecture

To address core-based SoC testing challenges and enable test access to embedded cores, a more structured, systematic, and hierarchical approach than the traditional DFT is required [4]. Zorian *et al.* [21] proposed a generic test architecture consisting of three

components: *Source/Sink*; *Wrapper*; and *Test Access Mechanism* (TAM). Figure 2-4 illustrates this generic conceptual test architecture.

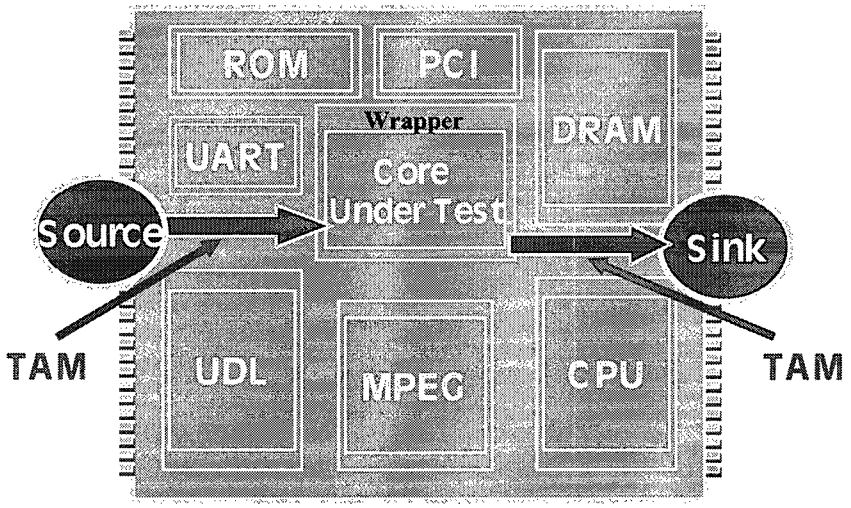


Figure 2-4: A generic conceptual test architecture [21].

In the model of Figure 2-4, the *source* applies the test stimuli to the core under test and the *sink* collects and performs an analysis of the test results with expected responses. In other words, the source *stores* and *controls* whereas the sink *stores* and *observes* in the process of testing a core. Independently, the source and the sink can either be on- or off-chip.

In addition, in the model of Figure 2-4, the TAM is the physical mechanism that connects the source and the sink with the core for communicating test data and control signals. TAM design determines how efficiently information is received and transmitted

from and to the outside world, and thus it affects, in part, the total test time, the complexity of test flow, and the test cost in general. Finally, the test wrapper is a shell around a core, which provides an interface between the core and its surroundings, isolates the core for test purposes, and helps in testing the interconnects. The wrapper is intended to enable "plug-and-play" cores such that cores, acquired from different providers, can be integrated into an SoC without any modifications.

As part of a wide variety of methods in addressing SoC test challenges, new *control* and *observation* mechanisms are required in SoC test [4]. Moreover, to facilitate a hierarchical test architecture, it is proposed that the test architecture needs to be separated from its behaviour [4]. In other words, it is proposed that the separation of the computation from the communication components of a test architecture is essential to address SoC test challenges. Finally, unification of the design and the test architecture flow is believed to be inevitable in SoC designs.

2.4.3 Test Resources Partitioning

With the growing complexity of ICs, more features have historically been added to external testers. This trend has transformed *automatic test equipment* (ATE) into expensive monolithic testers that conform to the idea of *one-solution-fits-all* [28][30]. In addition, many existing ATE are not designed to leverage the DFT features of the chips they test. DFT-aware testers are considered an important cost reduction factor in SoC test. In addition, the partitioning of ATE resources is emerging as an important solution to address test challenges in SoC. In this new model, the functionality of testers is broken

into different segments and distributed between different resources. For example, testing digital components, memory, and analog blocks of an SoC are performed in different ways and with different resources.

Trade-offs of test resource partitioning and features of low-cost testers are still the subject of debate and extensive research. However, it has been argued and shown that this division helps in creating much needed hierarchy, modularity, flexibility, and scalability in the test flow and the test resources, and ultimately, it helps to achieve higher test quality and reduce test cost [27][28][30][35][36]. As an example, it is known that almost every chip has *phase locked loop* (PLL) circuits for its clocking circuitry. Migrating clocking circuitry of ATE to chips and reusing on-chip PLL not only reduces ATE cost but also enables at-speed testing. As another example, reference [28] presents an economic study of using testers with reduced features in the test flow. In this study, the benefits of using lower cost testers in the early stages of the test flow is compared with the penalties of escaping certain defects due to the lower capabilities of these low-cost and reduced-feature testers. It is shown in [28] that, even in extreme cases, using low-cost testers results in a lower overall chip cost.

2.4.4 Multi-site Testing

Multi-site testing, or parallel testing, refers to concurrent testing of multiple similar chips on a single tester. Multi-site testing reduces overall test time and test cost. However, with the current ATE architecture, pin densities on the testers cannot easily increase to enable multi-site testing. One solution is to reduce tester pin functionality, enabling more

pins per tester [28][35]. This solution is the direct outcome of the test resource partitioning as described in Section 2.4.3. Therefore, multi-site testing is one of the main driving forces behind low-cost, low-feature, and DFT-aware testers [28].

In addition, new DFT methodologies are required to achieve higher degrees of parallelism in testing. These methodologies should significantly reduce the number of required test pins per IC, and, hence, increase the number of ICs that can be tested on a tester.

Another way to achieve testing parallelism in SoC designs is to simultaneously test multiple cores using new DFT techniques [37]. However, one limiting factor in such a solution is the power rating of the IC under test. This rating may limit the number of cores that can be tested concurrently and, hence, limit the effectiveness of the solution.

2.4.5 Test Data Compression

Compressing the test data is a clear way to address some of the challenges detailed in Section 2.3. There are many techniques in this domain [38][39][40][41][42][43]. The majority of them target *Built-in Self-Test* (BIST) as the primary method of the IC test. However, there are newer published techniques that use *automatic test pattern generation* (ATPG) test data sets alongside embedded blocks [44][45][46].

2.4.6 Embedded Testing

One important challenge for testing core-based SoC designs is accessing embedded cores from the system's primary inputs/outputs (I/O). This accessing proves challenging owing

to the limited number of primary I/O pins. According to the ITRS prediction, the maximum number of pins on the chips is to increase from about 3,070 in the year 2001 to 4,420 by the year 2016 [4]. However, the ratio of power/ground pins to the total pins on the chip can typically vary between 2:3 and 1:2 [4]. Moreover, with more complex IPs, there will be more test pins on each core. Hence, the combined effect is a shortage of available pins for testing in future chips. Multiplexing signal and test pins can mitigate this problem. However, increasingly there are high-speed signal pins on a chip that cannot be shared and, hence, the shortage of test pins to access embedded cores will still be a problem needing to be addressed.

In addition, testing core-based SoCs requires a level of complexity that makes relying on external *automatic test equipment* (ATE) insufficient [4]. As previously mentioned, the cost of ATE rises as the complexity of ICs increase. Moreover, ATEs can quickly become outdated compared to the chips that they test, and, for example, often lack the necessary resolution and accuracy to test new devices effectively and perform at-speed testing of these devices [4][23].

The limited number of chip I/Os, high tester cost, the need for at-speed testing, and the typically lagging technology of external testers call for specialized embedded support infrastructure blocks, suited for embedded testing [4][23]. Test resources, hence, need to be partitioned into their internal and external components such that embedded test components take over some functionality of the external ATE resources [4][47]. These embedded solutions are referred to as *infrastructure IPs* (I-IP) [32][48]. I-IP blocks do

not add to the main functionality of the chip. Rather, they facilitate post fabrication test, and can enhance chip's lifetime reliability [49]. It is imperative that such embedded solutions perform deterministic tests, use the benefits of the *automatic test pattern generation* (ATPG), and work in harmony with their specialized off-chip counterparts [4][29][44][50].

2.5 Current Test Architectures

This section overviews current test architectures. Section 2.5.1 reviews the standard wrapper being developed by the IEEE P1500 Working Group. Section 2.5.2 describes the general concept of *Built-in Self-Test* (BIST) architectures. Finally, Section 2.5.3 provides a comprehensive survey of other test architectures where external test equipment is used as the test *source* and the test *sink*.

2.5.1 Standard Wrapper

The IEEE P1500 Working Group is working towards a *Standard for Embedded Core Test* (SECT), to allow the automatic identification and configuration of testability features in integrated circuits containing embedded cores [51]. Towards that end, the P1500 aims at standardising a scalable architecture in the form of a wrapper around a core and defines a *Core Test Language* (CTL). However, other components of the test architecture, i.e., source, sink, and TAM will not be standardized. The scalable architecture, as illustrated in Figure 2-5, will provide a mandatory serial port and an optional parallel interface for testing a core.

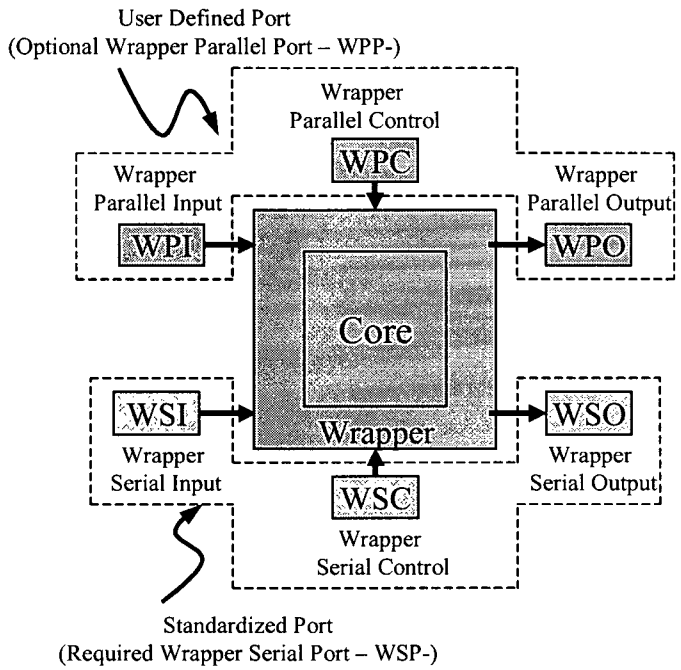


Figure 2-5: Block level overview of a P1500 wrapper [51].

In addition, the standard will provide a set of instructions enabling different modes, mandatory and optional, for the wrapper architecture. These instructions will be loaded into a *wrapper instruction register* (WIR) and control *wrapper boundary registers* (WBR) in front of the core's functional input/outputs. Figure 2-6 illustrates a conceptual view of the required P1500 wrapper architecture [51]. A *core-provider* will use the standard language to communicate to the *core-user* the internal, external, and pattern information for every test mode of a core. A *core-user*, however, is responsible for designing the source, sink, TAM, and the overall system DFT.

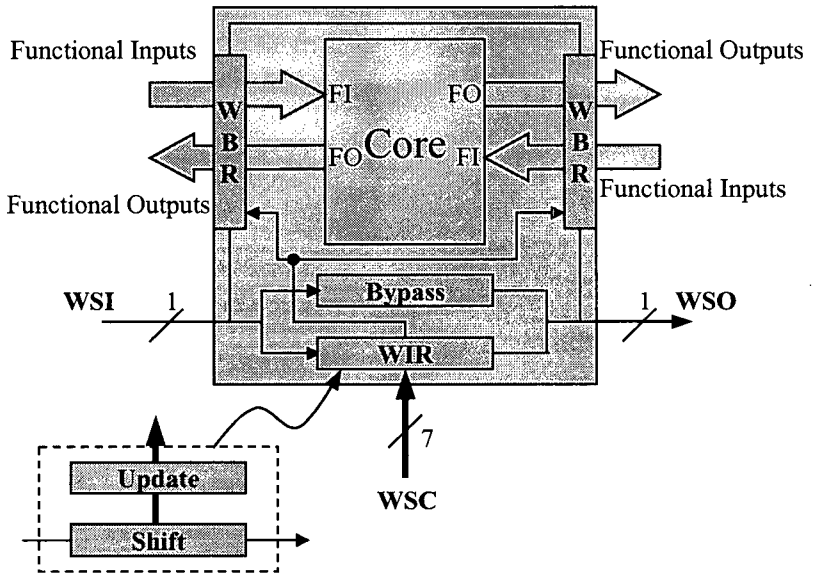


Figure 2-6: Conceptual view of the required P1500 wrapper architecture [51].

2.5.2 Built-in Self Test

Embedded testing architectures in the form of Built-in Self-Test (BIST) have been used to reduce test cost [2][52][53][54][55]. BIST facilitates in-field and at-speed testing and, hence, improves the quality of test. Moreover, BIST lowers the test cost by partitioning the test resources as outlined in Section 2.4.3. Memory BIST is now widely used to test the memory components of ICs. Moreover, using analog BIST results in reduced-function and low-cost external testers, and can potentially achieve better test quality. Finally, logic BIST is gaining more interest as a solution that can address many of the test challenges explained in Section 2.3. This thesis uses BIST to refer to logic BIST, as the

focus of this thesis's work is the development of new and novel DFT techniques for digital cores of SoC designs.

A block diagram of a typical BIST arrangement is illustrated in Figure 2-7. In a typical BIST arrangement, test patterns are either stored or generated on chip and are local to the CUT. These patterns are applied to the CUT under the control of a BIST controller. The outputs of the CUT are compacted during the entire process of test data application. Finally, the compacted output response is compared to reference signatures at the end of the test process, and a *pass/fail* signal is generated. The dominant technique in logic BIST is based on the STUMPS architecture [38][56], which uses multiple scan chains as the underlying DFT technique in the CUT (for an introduction to scan chains, see Section 3.2). Hence, this thesis assumes the use of scan chains in BIST.

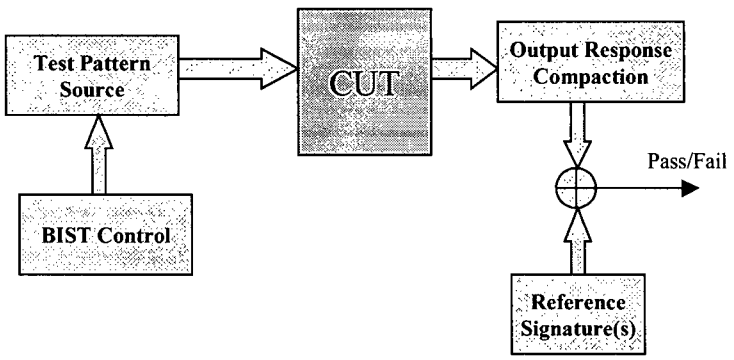


Figure 2-7: Block diagram of a typical BIST arrangement.

BIST can simplify the integration of a core's DFT into a system DFT in SoC designs and, hence, enable near-seamless reuse of test resources. In these techniques, both the test source and the sink are local to the CUT and, thus, the TAM amounts to local wires between BIST circuitry and the core. This local communication model simplifies the problem of test data transmission. However, on-chip storage of the entire deterministic test vectors, for each core, is not cost-effective or even practical in many cases.

To eliminate the need for on-chip storage of test data, typically, *linear feedback shift registers* (LFSR) are used to generate pseudo random patterns as the stimuli in the BIST arrangement. These test vectors are then applied to the CUT, and, in the simplest configuration of BIST, the test results are compacted and the final signature is compared to the signature of expected results. One problem with such pseudo-random pattern BIST is the potentially lower fault coverage, compared to that resulting from deterministic patterns obtained via *automatic test pattern generation* (ATPG). This lower fault coverage is mainly due to the linear dependency of the generated random patterns and the fact that some faults, such as path delay faults, are resistant to random patterns.

The potentially lower fault coverage associated with a BIST methodology can be mitigated, although not solved, in different ways. One method for smaller circuits is the introduction of a *phase shifter* between the LFSR and the CUT's scan chains[56]. This phase shifter reduces the dependency of the bits applied to the scan chains. Another method consists of inserting additional test points in the design [2][57][58]. However, test point insertion requires modification of the design and is therefore not desirable.

In BIST, phase shifters and test points cannot typically guarantee the same high fault coverage achievable using ATPG-based vectors. Other techniques have been proposed that target the pseudo-random pattern generator to mitigate the problem of low coverage [38]. These techniques include: weighted pseudo-random sequences [59] and multiple-polynomial LFSR with reseeding [60]. However, the complexity and cost factor of these solutions can be high and not desirable in many cases.

Another strategy to address the potential low fault coverage in BIST is known as mixed-mode BIST. Mixed-mode BIST uses deterministic test vectors as well as pseudo-random vectors [61]. As one example, the fault coverage can be increased by applying top-up test vectors from an external tester. However, it has been shown that for large industrial designs, the ATPG top-up pattern volume is 25-65% of a full ATPG test approach [62]. In another study, this ratio is reported as high as 70% [63]. This large ratio defeats many reasons for using BIST, and is not acceptable in many cases. Other techniques in mixed-mode BIST include: bit-flipping [64], bit-fixing [65], weighted random pattern generation [66], and *Sequence Generating Logic* [67]. In this latter work, no external deterministic pattern is applied. Rather, a block is placed between the LFSR and the CUT that changes the output of the LFSR into deterministic patterns. The potential problem with these techniques is the large area required for the BIST circuit. Reference [67] reports BIST hardware equal to about 5-15% of the total CUT area, for circuits of up to 100K gates that use 10,000 test patterns. It is, however, possible to

trade-off test quality, test time, and BIST area overhead [67], if lower test quality or longer test time is acceptable.

Compacting test responses and using a signature analyzer at the output of the CUT, result in additional challenges of BIST techniques. Diagnostics in BIST is more difficult than ATE-based testing as, in BIST, most of the information needed for diagnosis is lost during compaction of the test results into signatures. Therefore, diagnosis is only possible with multiple test runs that add to the test time and the complexity of the test process. Moreover, using signature analysis in BIST can mask some faults, caused by inherent aliasing in signature analyzer compaction schemes. Another problem with signature analyzers, and consequently in BIST, is the "zero-ing out", where the history of the circuit response is lost [2].

Another drawback with logic BIST techniques is in their very long test time requirement. In ATPG-based testing, the test vectors are generated and then compacted to be as few as possible. Moreover, the CUT's output response is compared with the expected response for each test vector, resulting in shorter test time for faulty chips. However, in BIST, owing to the random nature of the entire test vector set –or at least the initial set of test vectors in the more elaborate versions of BIST–, a large number of test vectors is needed for achieving an equivalent level of fault coverage. In addition, the comparison with the expected reference signatures is only possible at the end of the test process or, at best, at the end of intermediate sections of the test process [68][69]. Hence, the result of the test is only available after application of multiple test vectors in addition

to the vector that can identify a fault, resulting in even longer test time for faulty chips. The study in [62] shows that, typically, the test time is two to three times longer than the test time using ATPG.

Using pseudo-random patterns in BIST, results in additional challenges. These patterns can cause parts of the CUT to generate responses that are unpredictable ("X") in the BIST simulation, and no signature can be defined for these cases. It is imperative to prevent the propagation of these unknown ("X") values to the compaction circuit, as the signature will be corrupted otherwise. Moreover, it is necessary to prevent bus conflicts during the application of the pseudo-random patterns. Finally, BIST using pseudo-random patterns can cause a large power dissipation, as these patterns may result in large circuit switching [70][71].

As was presented in this section, logic BIST is a powerful test methodology that can address many problems facing the test community for testing core-bases SoCs. However, there are many challenges in the path of making logic BIST a viable test solution. These challenges were summarized in this section and some of the proposed improvements presented. Additional problems and challenges in BIST include: making the design BIST-ready; automating BIST insertion; and integrating BIST into the overall design flow with minimal impact [62]. This thesis, as explained in Section 1.2.1 and detailed in Chapter 3, builds on the strengths of BIST methodology, solves some of its potential shortcomings, and, hence, presents an alternative solution in addressing some of the different SoC test challenges described in Section 2.3.

2.5.3 *ATE-based Test Architectures*

ATE-based test architectures differ from BIST in the fact that both the source and the sink are an integral part of an external tester in the form of an ATE. This thesis uses the TAM arrangement in these architectures as a classifier. In SoC test architectures, test stimuli and test results are transported through the TAM. Hence, the TAM can be viewed as the communication link in the architecture. In addition to the TAM, extra control lines are used to properly set-up the TAM and core wrappers. Hence, there are two categories of lines in generic SoC test architectures: 1) data lines (referred to as TAM in this work) and 2) control lines.

Dedicated wires or existing functional interconnects can be used for the data lines [72][73][74][75]. While in [72], [73], and [74] the cores are modified such that each core has a *transparent* mode for testing, [75] uses the processor bus for test data transport. There are many proposed ATE-based test architectures in the literature. Based on the connection method between the chip pins and the core terminals, these can all be grouped into three main categories: multiplexer-, serial-, and bus-based connections. Most of these architectures suggest the use of a serial control mechanism to properly set-up the TAM and core wrappers.

In the first category, multiplexers are used to allow test access to the cores. The simplest method in this category is to multiplex the test pins to the primary I/Os such that a direct path is established during test [76]. A second method modifies the cores such that each core has a *transparent* mode for testing [72][73]. A recent third method

provides a transparent path based on modelling the TAM design as an *Integer Linear Programming* (ILP) problem, to minimise the overall test time and overhead area [74].

A number of test architectures in the serial-based category use the established IEEE 1149.1 standard [26][77][78]. Whetsel, in [79], uses a hierarchical structure by introducing a *Tap Link Module* (TLM). An improvement on the TLM is presented in [80], where the *Test Access Port* (TAP) of the 1149.1 standard is kept unchanged from its original form, and hence, simpler TLM controls are designed.

A number of different variations of the bus-based connection schemes have been reported. Varma *et al.* [33] suggest a structured architecture based on separate data and control buses. In their work, provision has also been made for using several such buses with different widths. To simplify the control mechanism in the TAM architecture and provide scalability of the architecture through hierarchy, a multilevel bus structure connected in a tree topology has been suggested [81]. Marinissen *et al.* suggested the TestRail architecture [22], where cores are connected in a daisy chain configuration and buses (or Rails) can have different widths, fan-in, and fan-out. In TestRail, each core can be bypassed if needed to access the next one in line and control is achieved via a serial connection.

In the bus-based category, different methods are suggested for accessing the cores from the buses. *Core Access Switches* (CAS) select P signals out of N bits of a bus and use the TestRail topology for the buses [82]. Whetsel has suggested an addressable architecture in [83]. In this latter architecture, each core is given an addressable *Test Port*, which can

serially be assigned with its appropriate address to provide an intelligent distributed control mechanism for connecting cores to buses. Finally, a time division multiplexing technique has been suggested in [84]. In the latter, using configurable and dedicated arbiters, cores autonomously assume the control of the bus. Additional issues in the bus-based connection scheme are the test architecture optimization and test scheduling. In order to minimize the test time and the number of required test pins for a given set of constraints, many studies have been reported that suggest different heuristics to optimally design test wrappers and assign the cores to the TAMs [85][86][87][88][89][90].

ATE-based test architectures benefit from a high test quality, as ATPG test vector sets are used. However, in the above test architectures, there is no distinction made between the communication and the application of test data. The close coupling between the communication and the application of test data works against many of the test trends detailed in Section 2.4. Therefore, novel test architectures are needed that, while utilising the quality of ATPG-based testing, conform to the test trends of Section 2.4 to a high degree. This thesis, as outlined in Section 1.2.2 and explained in Chapter 4, presents a methodology that helps in achieving systematic test architectures that benefit from test reuse, resources partitioning, and multi-site and embedded testing.

Chapter 3
Dedicated Autonomous Scan-Based Testing

3.1 Introduction

As discussed in Section 2.4.6, embedded testing is one of the most important trends in dealing with challenges of testing core-based SoC designs. Test resources need to be partitioned to move part of the tester's functionality onto a chip. This chapter presents the concept, and provides the implementation and experimental results of a *Dedicated Autonomous Scan-based Testing* (DAST) methodology [91][92], for testing embedded digital cores. The novelty in DAST is in the fact that core test stimuli and expected results, generated by ATPG, are pre-processed into a new test-data protocol. In doing so, all the control sequence of testing is transferred from the external ATE to a dedicated *Embedded Autonomous Sequencer* (EAS) block, associated with single or multiple cores

on an SoC. Moreover, *Embedded Autonomous Results Analyzer* (EARA) blocks [92] are used in synchronism with EAS blocks to deterministically analyze the test results and to compare them with expected results. Thus, using DAST, a simple data transmitter can be used in place of an external ATE. Hence, the complexity of the test flow can be significantly reduced and rendered much more scalable and portable between technology generations.

In essence, DAST divides the ATE functionality into its test data *communication* and test data *control/observation* components, and migrates the latter away from the ATE to place it on-chip at the periphery of the core in the EAS and EARA. Therefore, the former can be reduced to an inexpensive unintelligent block. Since DAST is based on the conventional scan/ATPG approach, it does not imply any compromise in terms of test-quality (coverage and fault models), ease of use, and broad applicability. Moreover, as the EAS and EARA blocks are external to a core, the impact on the design and the design flow is also minimal. Finally, as the test is performed on chip, at-speed test is feasible with minimal cost.

3.2 Components of Scan-based Testing

One of the most popular structural DFT techniques is scan design. Scan design increases internal circuit node controllability and observability of sequential circuits. In essence, scan design transforms the difficult task of sequential circuit test into the easier task of combinational circuit test. To achieve this transformation, regular D-type flip-flops in a sequential circuit, as shown in Figure 3-1a, are replaced with the scan flip-flops as shown

in Figure 3-1b. When the sequential circuit is in test mode, *Test Select* (*TS*) signals of the multiplexers are asserted to select the *Scan In* (*SI*) as the input of the scan flip-flops *(scan cells)*. In this way, flip-flops are connected in long chains to form shift registers. Using the created shift registers, all the scan cells can be set to desired states in the test mode. Similarly, the state of the internal flip-flops and nodes can be captured by the scan registers and shifted out serially, thereby providing increased observability [2][3].

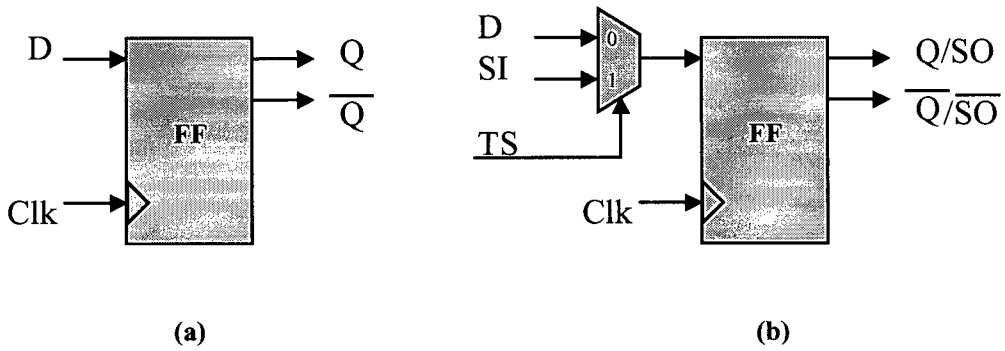


Figure 3-1: Block diagrams of (a) D-type flip-flop, and (b) Scan D-type flip-flop.

By nature, scan-based testing is a repetitive procedure. The steps in scan testing a core can be summarised by the generic waveforms in Figure 3-2. In Figure 3-2, the signal labels *PI*, *SI*, *CLK*, *ST_PO*, and *ST_SO* denote primary input values, scan input values, test clock, strobe primary outputs, and strobe scan outputs, respectively. In addition, *TS* represents the value of the test select signal.

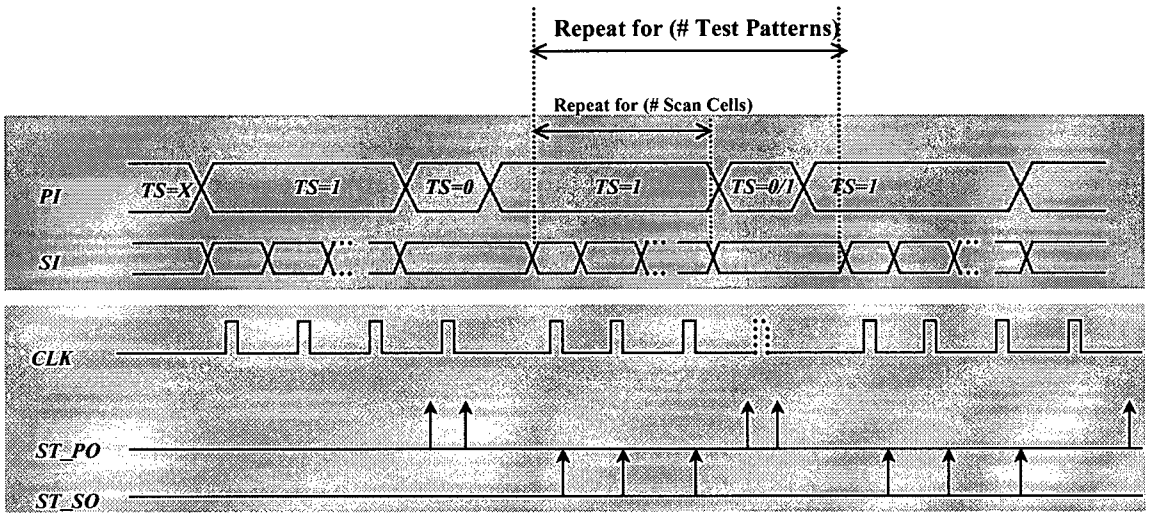


Figure 3-2: Generic scan test waveforms.

From Figure 3-2, a scan test procedure can be conceptually divided into two functions: a) *data delivery*, and b) *control/observation*. For a given test pattern, *data delivery* refers to the following steps:

- Load the test vector into the scan chain flip-flops through the scan inputs (*SI*);
- Apply and maintain non-scan portions of the test vector at the primary inputs (*PI*) of a core;
- Set *TS* to logic 1 and apply new values to *PI*.

For the same given test pattern, *control/observation* refers to the following steps/actions:

- Assert *CLK* to load-in test vector in the scan chain;

- Assert *ST_SO* to test for values at scan outputs (*SO*);
- Assert *ST_PO* on two different occasions to check consistency at the primary outputs (*PO*).

In the traditional methodology, the ATE is responsible for transforming raw test vector data into scan sequence waveforms of Figure 3-2. This thesis, however, introduces a simple hierarchy whereby the complex ATE scan functionality is replaced by a simple off-chip test-data compiler. As described in Section 3.3, the compiler only needs to marginally transform (into a new protocol) the raw test data from ATPG, embedding the capability to generate the complex scan sequences in the dedicated EAS and EARA blocks associated with specific cores.

3.3 DAST Concept

External ATEs need to perform both functions of data delivery and control/observation. This is one of the reasons for increasingly expensive and complex test flows and equipment. Given today's ample resources in terms of silicon, this thesis takes advantage of the repetitive nature of scan testing by incorporating EAS and EARA blocks dedicated to each core on an SoC. The EAS's function is to accept simple binary test data and to transform it to produce the specific waveforms required for applying the scan pattern to the specific target core. The EARA's function is to work in synchrony with the EAS block and compare the incoming expected test results to the test results at the output of the core. This essentially amounts to separating the two functions of the off-chip ATE, as

described in Section 3.2, and moving the control and observation onto the chip. That is, DAST replaces a hierarchically flat system by one with a simple hierarchy.

Similar to the case of ATE-based testing, to reduce the amount of memory needed in the off-chip block, the DAST methodology allows for the transformed test vectors to be compressed before sending them to the embedded blocks. The compressed test sets can then be decompressed on-chip at the EAS and EARA blocks.

Using the DAST methodology, both test stimuli and expected test results are sent to a core via a TAM. Thus, as far as scan testing is concerned, it is possible to replace the ATE by an unintelligent serial interface circuit block. This latter block is simply required to transmit test data at a designated clock rate to the embedded blocks of DAST. Upon receiving test data, an EAS block applies the test data to a core and generates the test clock. Simultaneously, and in parallel to the EAS operation, the EARA block collects the test results from the *primary input* and *primary output* pins of the core and compares them with the incoming expected test results. On the first occurrence of an inconsistency between the expected and actual results, a "sticky" no-go signal is generated by the EARA block, marking the detection of a fault. This idea is illustrated in Figure 3-3.

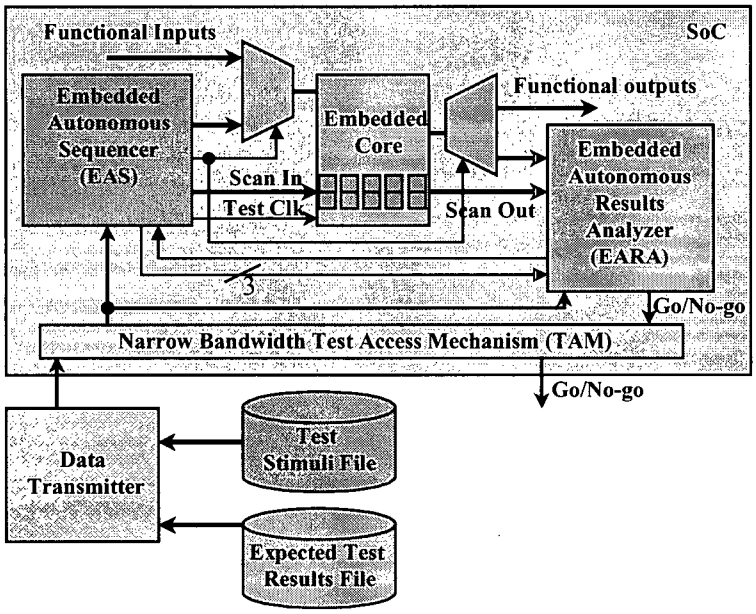


Figure 3-3: Concept of DAST as a block diagram.

The design flow incorporating the DAST methodology is shown in Figure 3-4. This flow is only partially different from a flow using conventional ATPG/Scan. Similar to the case of a conventional ATPG/Scan test flow, scan chains are assumed to be inserted automatically and/or manually before the test vector and expected test result files are generated by the ATPG. The function of EARA must be fully deterministic. Hence no unknown values, i.e., "X" values, can be allowed in the test stimuli and expected test results, as otherwise two lines are needed to encode the 3-valued data bits. In the specific case of this thesis, without loss of generality, all "X" values in the test vector file are replaced with the zero logic value. This modified test vector file is used to apply the test vectors to a gate-level model of the circuit, and real expected results are collected and

replaced with "X" values in the expected test results file. Using the EAS- and EARA-compilers, the test vector and the expected test results files are subsequently compiled into the test data and the expected test result protocols of EAS and EARA.

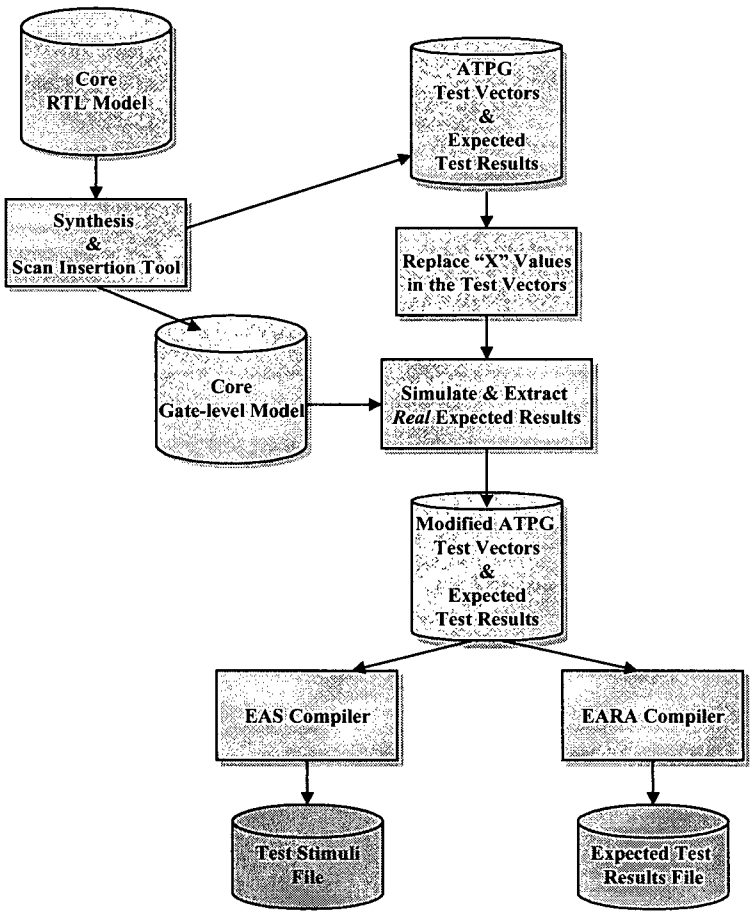


Figure 3-4: DAST design flow.

Both the EAS and EARA logic are generated in RTL VHDL or Verilog and placed out of a core under test. Therefore, EAS and EARA are generic, flexible, and non-intrusive to a core under test, and should be viewed as test *soft* IP. As a result, the DAST design is adaptable to the parameter changes of a core and scales with the fabrication technology. The EAS and EARA logic only depends on a core's primary input/output numbers, scan chains number/depth, and specific type of scan cells, used to implement the chains. They do not depend on a core's functionality. Both the EAS and EARA blocks also include bypass circuitry to allow the SoC's normal functional mode.

For the test flow, only a simple data transmitter is required to send the test stimuli and the expected result data onto the chip, as shown in Figure 3-3. The test control, waveform generation, and comparison to the real results are all done automatically by the EAS and EARA blocks.

3.4 Implementation

Assuming a simple serial connection for each of EAS and EARA block to the off-chip, the DAST methodology was implemented to enable the simple comparison of the performance and cost of the DAST methodology to the classical one. No specific TAM architecture was assumed for the development of DAST. However, a direct connection, as well as NIMA (as introduced in Section 1.2.2 and explained in Chapter 4), was used for implementation and verification.

3.4.1 EAS & EARA Compilers

The modification of the ATPG-generated file, as described in Section 3.3, is core-dependent. This modification takes a few seconds for simple cores to several minutes for more complex ones. After the modification of the ATPG-generated file, EAS and EARA compilers are needed to convert the modified ATPG-generated file into *test stimuli file* (EAS file) and *expected test results file* (EARA file), as shown in Figure 3-4. Two C programs were developed that take as inputs the modified ATPG test vectors and expected test results. These programs then insert three-bit op-codes to the beginning of each section of the test program to convert the test vectors and expected results into EAS and EARA protocols, respectively. The time taken for this protocol conversion is very short and is less than a few seconds on Sun Blade100 machines for cores of up to 100K gates. The following op-codes are used as simple instructions in the EAS and EARA:

- *Shift-PI-BSR*: shift into primary input boundary scan register;
- *Shift-SC*: shift into scan chains;
- *Shift-SC-BSR*: shift into scan input boundary scan register;
- *Assert-Clk*: assert the test clock;
- *Shift-PO-BSR*: shift into primary output boundary scan register;
- *Shift-SO-BSR*: shift into scan output boundary scan register.

The algorithm for the EAS compiler is shown in Figure 3-5. Based on the waveforms of Figure 3-2 and the intended operation of EAS blocks, the algorithm, in line 2,

identifies seven *action groups,* i.e., *Pattern boundary, PI data, Clocking, SI data, PO data, SC data,* and *SO data.* In lines 35, 39, 42, 46, 49, and 55, the necessary data for *primary input* (*PI*), clocking information, *scan input* (*SI*), *primary output* (*PO*), *scan chain* (*SC*) elements, and *scan out* (*SO*), are saved by the algorithm in temporary variables. In addition, proper flags for each action group are set at the same time.

```
1.  While not end of ATPG file then
2.  identify action_group;
3.  case action_group of
4.    Pattern boundary:
5.      if Shift_PI_BSR active {
6.        write Shift_PI_BSR op-code to EAS file;
7.        write PI data to EAS file;
8.      }
9.      if scan cycle {
10.       write Shift_SC op-code into EAS file when needed;
11.       for number of scan chains {
12.         for number of scan depth {
13.           write scan element data into EAS file;
14.         }
15.       }
16.     else { -- not scan cycle
17.       if Shift_SC_BSR active {
18.         write Shift_SC_BSR op-code to EAS file;
19.         write SI data to EAS file;
20.       }
21.       if Shift_PO_BSR active {
22.         write Shift_PO_BSR op-code to EAS file;
23.         if Shift_PI_BSR active {
24.           increase zero_padding counter by op-code and PI length;
25.         }
26.       }
27.       if Shift_SC_BSR active {
28.         increase zero_padding counter by op-code and SC length;
29.       }
30.       for difference of PO length and zero_padding counter {
31.         zero-pad EAS file;
32.       }
33.       write Assert_clk code into EAS file based on clocking information;
34.       break;
35.   PI data:
36.       save PI data;
37.       Shift_PI_BSR active;
38.       break;
39.   Clocking:
40.       save clocking information;
41.       break;
42.   SI data:
43.       save SI data;
44.       Shift_SC_BSR active;
45.       break;
46.   PO data:
47.       Shift_PO_BSR active;
48.       break;
49.   SC data:
50.       for number of scan chains {
51.         save scan element data;
52.         Shift_SC active;
53.       }
54.       break;
55.   SO data:
56.       for number of scan chains {
57.         Shift_SO_BSR active;
58.       }
59.       break;
60. End while not end of ATPG file
```

Figure 3-5: EAS compiler algorithm.

As shown in lines 4 to 34 of the EAS compiler algorithm, based on the previously set flags, the EAS file is updated in the case of *Pattern boundary* action group. In lines 5 to

8, the op-code for primary input data together with PI data is added to the EAS file. In addition, in lines 9 to 15, data for scan cycle is identified and added to the EAS file. In lines 16 to 20 the op-code and data for scan inputs are added. Finally, lines 21 to 34 identify the number of zero-padding bits needed to maintain the synchronization between EAS and EARA blocks. These lines add the padding bits based on the difference between *PO*'s length and those of the *PI* and *SC*.

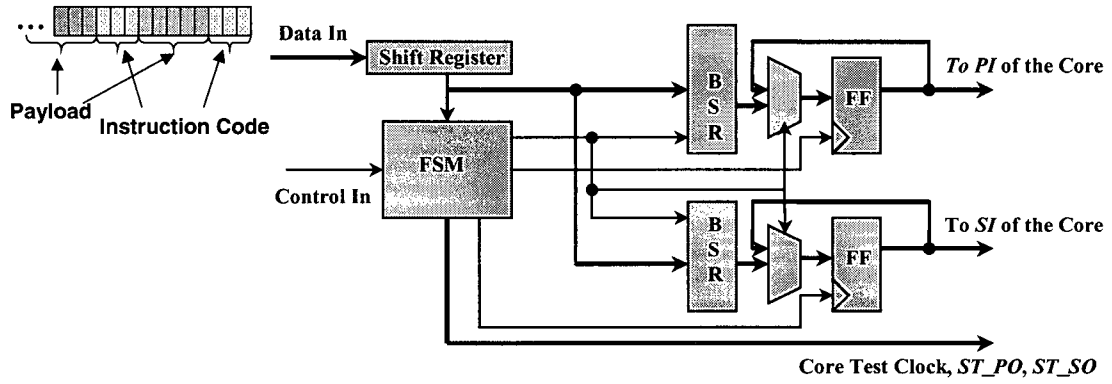
The algorithm for the EARA compiler is shown in Figure 3-6. Similarly to the case of the EAS compiler, and based on the waveforms of Figure 3-2 and the intended operation of EARA blocks, the algorithm, in line 2, identifies seven *action groups*. These action groups are: *Pattern boundary, PI data, Clocking, SI data, PO data, SC data,* and *SO data*. In addition, in lines 59, 62, 65, 68, 72, and 76, the necessary data for *PI*, clocking information, *SI*, *PO*, *SC* elements, and *SO*, are saved by the algorithm in temporary variables, and proper flags are set. Finally, in lines 4 to 58 the EARA file is updated. In this latter section, the algorithm, in lines 6 to 10, adds the op-code for PI data to the EARA file. Since no action is required in the EARA block for this period, 0 is added to the EARA file, to indicate this fact. Lines 11 to 20 handle the initialization part of the test vector when the EARA block is mainly dormant and, hence, it is only required to keep its synchronization with the EAS block. Lines 21 to 28 add the op-code and data for *SO*. Finally, in addition to updating the EARA file for *PO* data, lines 29 to 58 identify and add the zero-padding bits required in the EARA block to maintain the synchrony between EARA and EAS blocks.

```
1.  While not end of ATPG file then
2.    identify action_group;
3.    case action_group of
4.      Pattern boundary:
5.        if scan cycle {
6.          if Shift_PI_BSR active {
7.            for total number of PI + op-code length {
8.              add 0 to EARA file;
9.            }
10.         }
11.         if initial section of test vector {
12.           for total length of op-code {
13.             add 0 to EARA file when needed;
14.           }
15.           for total number of scan elements {
16.             for total number of scan chains {
17.               add 0 to EARA file;
18.             }
19.           }
20.         }
21.         else {-- not initial section of test vector
22.           write Shift_SO_BSR op-code into EARA file;
23.           for total number of scan chains {
24.             for total number of scan elements {
25.               wrtie SO data to EARA file;
26.             }
27.           }
28.         }
29.       else {-- not scan cycle
30.         if Shift_PO_BSR active {
31.           write Shift_PO_BSR op-code to EARA file;
32.           write PO data into EARA file;
33.           if Shift_PI_BSR active {
34.             increase zero_padding counter by op-code and PI length;
35.           }
36.           if Shift_SC_BSR active {
37.             increase zero_padding counter by op-code and SC length;
38.           }
39.           for difference of PO length and zero_padding counber {
40.             zero-pad EARA file;
41.           }
42.         else {-- Shift_PO_BSR not active
43.           if Shift_SC_BSR active {
44.             for total number of SC and op-code length {
45.               write 0 to EARA file;
46.             }
47.           }
48.           if Shift_PI_BSR active {
49.             for total number of PI and op-code length {
50.               write 0 to EARA file;
51.             }
52.           }
53.           for op_code length {
54.             write 0 to EARA file based on clocking information;
55.           }
56.         }
57.       }
58.       break;
59.     PI data:
60.       Shift_PI_BSR active;
61.       break;
62.     Clocking:
63.       save clocking information;
64.       break;
65.     SI data:
66.       Shift_SC_BSR active;
67.       break;
68.     PO data:
69.       save PO data;
70.       Shift_PO_BSR active;
71.       break;
72.     SC data:
73.       Set flag for initial section of test vector;
74.       Shift_SC active;
75.       break;
76.     SO data:
77.       for number of scan chains {
78.         save SO data;
79.         Shift_SO_BSR active;
80.       }
81.       break;
82. End while not end of ATPG file
```

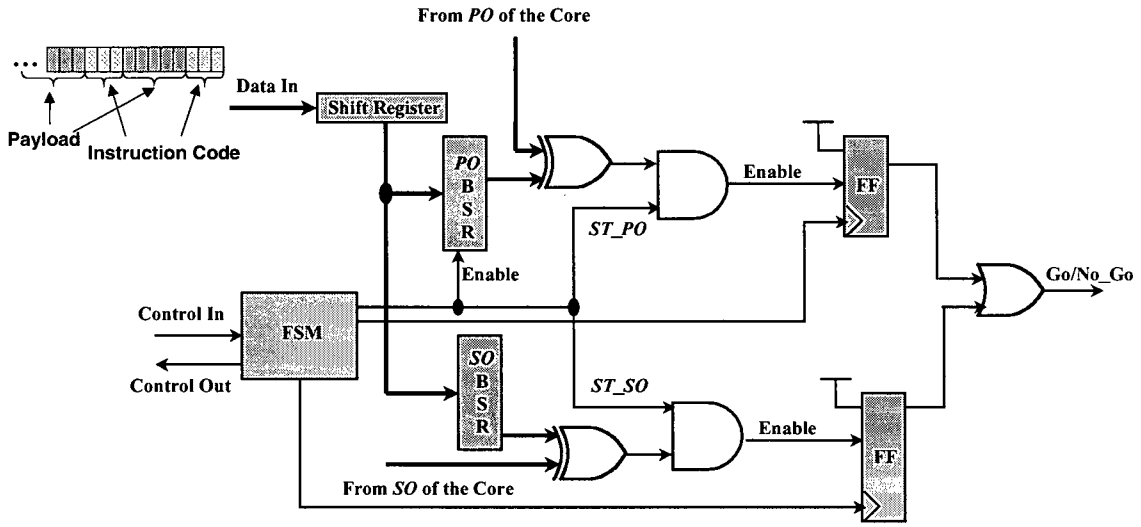
Figure 3-6: EARA compiler algorithm.

3.4.2 EAS & EARA Hardware

The hardware implementations of both EAS and EARA blocks are illustrated in Figure 3-7. These hardware implementations are extremely simple and compact, and require minimal design effort for any given core. Moreover, the modularity of the EAS and EARA design allows for their easy automation.



(a) EAS block.



(b) EARA block.

Figure 3-7: Hardware implementation of DAST components.

As shown in Figure 3-7a, in the EAS block, the incoming data is captured in a shift register until its *finite state machine* (FSM) block can decode the op-codes. Test data destined for the primary inputs, *PI*, are then sent to the appropriate *boundary scan registers* (BSR), and correct shift/capture signals are asserted by the FSM. The same is true for data destined for the scan inputs, *SI*, of the core. Upon receipt of the *Assert-Clk* op-code, the EAS toggles the core's test clock.

Similarly, in the case of EARA block, as shown in Figure 3-7b, the incoming data is captured in a shift register until its FSM block can decode the op-codes. Expected test results for the *primary outputs* are then sent to the *PO-BSR* and appropriate shift/capture signals are asserted by the FSM. The expected test results of the scan outputs, however, are always captured in the *SO-BSR*. Using XOR gates, the outputs of both the *PO-* and *SO-BSR* are always compared to a core's *primary output* and *scan output* values, respectively. However, using AND gates, the results of the comparison are gated by the *ST_PO* and *ST_SO* signals, respectively. The output of the AND gates then act as enable signals to two flip-flops such that if a mismatch is detected the Go/No-Go signal is asserted high.

The *Algorithmic State Machine* (ASM) for EAS's FSM block is given in Figure 3-8 as pseudo-code. After receiving the op-code in line 4, and based on the received op-code, the state machine of Figure 3-8 enters one of five different states. These states are: *Shift_PI_BSR, Shift_SC, Shift_SC_BSR, Assert_Clk,* and *Shift_PO_BSR*. In these states, the ASM generates necessary signals for the correct operation of the EAS block in

applying the test stimuli to the primary inputs and scan inputs as described in Section 3.2. In line 18, appropriate signals are asserted to synchronize the operation of EARA block with the EAS block. Finally, in lines 27 and 28, the state machine halts its operation until the EARA state machine calls for resumption of the process. This handshaking is needed to account for different length of scan-in and scan-out cycles of any given core.

```
1.  if valid data is present then
2.     while valid data and not end_of_op-code then
3.        while not end {
4.          get the op-code;
5.        }
6.        case op-code of
7.          Shift-PI_BSR:
8.                while not end {
9.                    shift to the primary input registers;
10.               }
11.               assert BSR update signals;
12.         Shift-SC:
13.               while not end {
14.                   shift to the scan chain;
15.                   if ready to assert the clock then
16.                       assert the clock;
17.               }
18.               assert signal to synch with EARA;
19.         Shift-SC_BSR:
20.               while not end {
21.                   shift to the scan chain input registers;
22.               }
23.               assert BSR update signals;
24.         Assert-Clk:
25.               assert the test clock;
26.         Shift-PO_BSR:
27.               call for EARA;
28.               wait for call from EARA;
29.       end case;
30.    end while valid data and not_end_of_op-code;
31. else wait;
```

Figure 3-8: EAS block *Algorithmic State Machine.*

The *Algorithmic State Machines* (ASM) for EARA's FSM block is given in Figure 3-9 as pseudo-code. Here, after receiving the op-code in line 3, and based on the received op-code, the state machine of Figure 3-9 enters in two different states, i.e., *Shift_PO_BSR* and *Shift_SO_BSR*. In these states, the ASM generates necessary signals for the correct operation of the EARA block to compare the incoming expected test results to the core primary and scan outputs. In line 9, the state machine halts its operation and sends calling signals to the EAS block before waiting for handshaking signals from the EAS ASM. Upon receiving the handshake signals from the EAS block, the ASM asserts primary output strobe signals in line 11. Finally, in line 13, the state machine halts its operation a second time for the EAS state machine to call for the resumption of the process. In this case, signals from the EAS block are used for comparing the core's scan outputs with the expected results.

```
1.  if synch EARA is present then
2.      if valid data is present and not ignore valid data then
3.          get the op-code;
4.          case op-code of
5.              Shift-PO_BSR:
6.                  while not end {
7.                      shift to the primary output registers;
8.                  }
9.                  call for EAS;
10.                 wait for call from EAS;
11.                 assert PO strobe signals;
12.             Shift-SO_BSR:
13.                 wait for signal from EAS for synchronization;
14.         end case;
15.     end if;
16.     else {
17.         wait for valid data;
18.     }
19. else {
20.     wait for synch EARA;
21. }
```

Figure 3-9: EARA block *Algorithmic State Machine.*

3.5 Experimental Procedure

Quantitative comparison of DAST performance with the existing solutions is not realistic, as DAST is the first solution that combines ATPG-based testing and BIST. Moreover, published works in this area mainly use proprietary designs and do not provide their results for a set of benchmark circuits that can be compared easily. Therefore, this thesis attempts to set precedence and reports the key characteristics of its developed work on a number of benchmark circuits. Moreover, the results of the work of this thesis are

compared with lower bound cases, where it is assumed that ATPG-based testing does not require any on-chip circuitry and that it results in the minimum testing time.

This section describes an SoC benchmark, UBC_SoC, developed for this work, and then provides details of the experimental procedure followed to evaluate the effectiveness of the DAST methodology, when applied to both UBC_SoC and ITC'02 SoC Benchmarks[1].

Experimenting and validating DAST by compiling empirical test time data, area, and power consumption trade-offs requires benchmark SoC circuits. One requirement for this benchmark is that it be described in RTL code so as to be able to insert scan chain(s), generate the test vectors, and implement the complete DAST flow.

Since benchmarks described in RTL for SoC studies are essentially non-existent at this time, UBC_SoC was generated. UBC_SoC constitutes of a series of benchmark circuits from the ITC99 Benchmarks [93], for which the RTL codes are readily available. UBC_SoC was created using Politecnico di Torino circuits (I99T) of ITC99 Benchmarks by changing the signal types from bit or integer to std_logic. UBC_SoC includes four cores: one instance of *b10* with a single scan chain; one instance of *b10* with three scan chains; one instance of *b15* with one scan chain; and one instance of *b15* with two scan chains. Figure 3-10 illustrates the UBC_SoC benchmark. The cores of UBC_SoC were subsequently wrapped using the guidelines of the IEEE P1500 standard as discussed in

[1] ITC'02 SoC Test Benchmarks are a set of circuits intended to help the research community for objective comparison of methods and tools for modular testing of core-based SoCs [97].

Section 2.5.1. Using Synopsys Design Compiler™, the circuits included in UBC_SoC were first synthesized in TSMC's 0.18 μm technology. Following this latter step, Synopsys Test Compiler™ was used to insert scan chain(s) in each circuit of UBC_SoC and generate the ATPG test stimuli and expected test results files of Figure 3-4. EAS and EARA compilers were then used to generate EAS and EARA files, as discussed in Section 3.4.1.

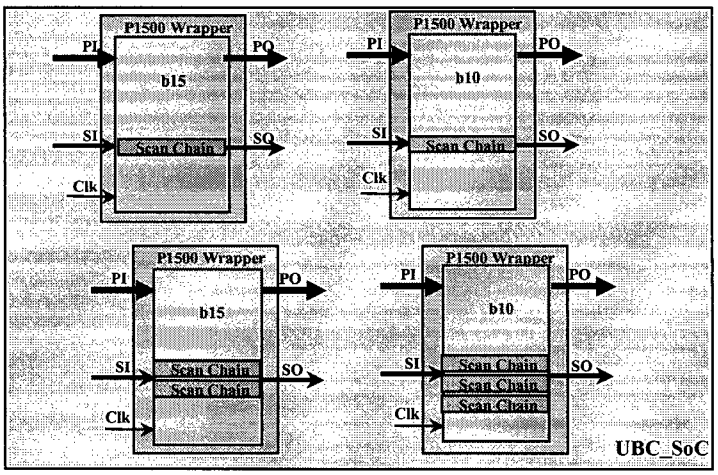


Figure 3-10: UBC_SoC Benchmark.

For each of the constituent cores of UBC_SoC, corresponding EAS and EARA blocks, as discussed in Section 3.4.2, were developed in VHDL. The gate-level representation of each core and its dedicated EAS and EARA RTL codes were then combined into a top VHDL module and these modules were added as core instances in the UBC_SoC

benchmark. In addition, to simplify the task at-hand, the same clock frequency for both the cores and their EAS and EARA blocks were used.

Separate test-benches were also developed in VHDL. These test-benches were used to simulate the application of test vectors to each core in the SoC and predict expected test time. Given that op-codes consist of three bits and that, typically, for each test pattern, both EAS and EARA sequence through several states, as illustrated in Figure 3-2 and described in Section 3.2, DAST test time models can be developed (see Appendix A). These test time models are defined for three different cases given by:

$$T_{M1\_DAST} = (9 + 2PI + SI \times SE) + TP(23 + 2PI + SI + SI \times SE) \quad (3\text{-}1)$$

iff $PO < PI+3$

$$T_{M2\_DAST} = (9 + 2PI + SI \times SE) + TP(20 + PI + PO + SI + SI \times SE) \quad (3\text{-}2)$$

iff $PI+SI+6 > PO \geq PI+3$

$$T_{M3\_DAST} = (9 + 2PI + SI \times SE) + TP(14 + 2PO + SI \times SE) \quad (3\text{-}3)$$

iff $PO \geq PI+SI+6$

where *PI*, *SI*, and *PO* are the numbers of primary inputs, scan inputs, and primary outputs as defined in Section 3.2. Moreover, *TP* is the total number of test patterns, *SE* is the maximum number of scan cells in the scan chain(s), and T_{M1_DAST}, T_{M2_DAST}, and T_{M3_DAST} are the test time models for DAST, in terms of clock cycles.

Only limited information is available for the ITC'02 SoC Benchmark circuits. In the second phase of the experiment, the models given in (3-1), (3-2), and (3-3) were used to predict the test time performance of DAST on the ITC'02 SoC Benchmarks, for which the test vector files are unavailable.

In addition, to compare DAST test time to that of a serial connection in a conventional external ATPG-based methodology, theoretical lower bound test-time models for the latter were developed (see Appendix B). Underlying assumptions of the latter models are that test data is applied to the core serially and that the results are observed serially. These models are given by:

$$T_{S1} = (2PI + SI \times SE) + TP(2PI + SI + SI \times SE) \tag{3-4}$$

$$iff \quad PO < PI$$

$$T_{S2} = (2PI + SI \times SE) + TP(PI + PO + SI + SI \times SE) \tag{3-5}$$

$$iff \quad PI+SI > PO \geq PI$$

$$T_{S3} = (2PI + SI \times SE) + TP(2PO + SI \times SE) \tag{3-6}$$

$$iff \quad PO \geq PI+SI$$

where T_{S1}, T_{S2}, T_{S3} are in clock cycles.

For each constituent core of the UBC_SoC benchmark, corresponding EAS and EARA blocks were synthesized in TSMC's 0.18 μm technology and area and power data were

collected. The data were collected using the *Area* and *Power Report* tools of Synopsys' Design Compiler™.

Moreover, using the parameters given for the cores (modules) in the ITC'02 SoC Benchmarks, corresponding EAS and EARA blocks were designed. These EAS and EARA blocks were synthesized as described earlier and their area requirements and power consumption collected with Design Compiler™. Finally, using the DAST test time models as given in Equations (3-1), (3-2), and (3-3), DAST test times for ITC'02 Benchmark modules were compiled.

3.6 Results

Table 3-1 reports total area and power for components of EAS blocks for the cores of the UBC_SoC Benchmark. In Table 3-1, b10_1SC, b10_3SC, b15_1SC, and b15_2SC represent cores of the UBC_SoC Benchmark, as shown in Figure 3-10, and refer to b10 with one scan chain, b10 with three scan chains, b15 with one scan chain, and b15 with two scan chains, respectively. In addition, in Table 3-1, PI_BSR, SI_BSR, Buffer_in, and FSM refer to primary inputs BSR, scan input BSR, buffer input, and the FSM blocks of the EAS blocks, as shown in Figure 3-7a. The data for the EAS block, excluding the mandatory components of the IEEE P1500 wrapper, appears in the Adj_EAS column. Here, for the different cores under consideration, the EAS blocks amount to a total area equal to approximately 300 to 350 2-input NAND gates.

Table 3-1: EAS Area and Power for the UBC_SoC Benchmark Cores

	Hardware Components of EAS for UBC_SoC								EAS for UBC_SoC	
	PI_BSR		SI_BSR		Buffer_in		FSM		Adj_EAS	
Circuit	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)
b10_1SC	2439	56	195	4	220	9	2765	51	2984	60
b10_3SC	2439	56	569	13	220	9	2695	49	2915	59
b15_1SC	7155	164	195	4	220	9	3399	60	3618	70
b15_2SC	7155	164	382	9	220	9	3362	61	3582	70

Table 3-2 reports total area and power for components of EARA blocks for cores of UBC_SoC Benchmark. In Table 3-2, PO_BSR, SO_BSR, Buffer_in, and FSM refer to primary outputs BSR, scan output BSR, buffer input, and the FSM blocks of the EARA blocks, as shown in Figure 3-7b. Again, similarly to the EAS blocks, the data for the EARA blocks, excluding the mandatory components of the IEEE P1500 wrapper, is given in the Adj_EARA column. Here, for different cores, the EARA blocks amount to a total area equalling that of 200 to 250 2-input NAND gates. Thus, per core, the total additional area for the DAST methodology in the UBC_SoC is minimal as it amounts to the equivalent of about 500-600 2-input NAND gates.

Table 3-2: EARA Area and Power for the UBC_SoC Benchmark Cores

	Hardware Components of EARA for UBC_SoC								EARA for UBC_SoC	
	PO_BSR		SO_BSR		Buffer_in		FSM		Adj_EARA	
Circuit	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)
b10_1SC	919	19	236	5	220	9	1638	39	1858	48
b10_3SC	919	19	459	12	220	9	1638	39	1858	48
b15_1SC	9314	205	236	5	220	9	2224	49	2443	58
b15_2SC	9314	205	350	9	220	9	2224	49	2443	58

Table 3-3 reports total area and power for components of EAS blocks for cores (modules) of the ITC'02 SoC Benchmarks. The notation used in Table 3-3 for components of the EAS blocks are identical to those used in Table 3-1. Here, for the different benchmarks under consideration, the EAS blocks amount to a total area equal to approximately 350 to 450 2-input NAND gates and are consistent with those for the UBC_SoC cores.

Table 3-3: EAS Area and Power for ITC'02 SoC Benchmark Modules

	Hardware Components of EAS for ITC'02 Benchmark Modules								EAS for ITC'02 Benchmark Modules	
	PI_BSR		SI_BSR		Buffer_in		FSM		Adj_EAS	
ITC'02 cores	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)
d281_m5	40335	925	1130	26	220	9	3582	62	3801	71
d695_m5	7155	164	6033	138	220	9	3655	63	3875	73
d695_m6	11676	268	3021	69	220	9	3541	61	3761	71
d695_m9	6594	151	6033	138	220	9	3639	63	3858	72
f2126_m1	67184	1539	1504	35	220	9	4037	68	4257	78
f2126_m2	16018	367	3021	69	220	9	4005	68	4224	78
f2126_m3	5651	130	195	4	220	9	3439	60	3659	70
g1023_m1	26235	601	2626	61	220	9	3696	64	3915	74
g1023_m2	41644	955	382	9	220	9	3370	60	3590	69
g1023_m4	27370	627	756	17	220	9	3541	62	3761	71
g1023_m10	56809	1301	195	4	220	9	3342	60	3561	70
h953_m1	21137	484	756	17	220	9	3744	64	3964	73
h953_m2	12799	294	382	9	220	9	3452	62	3671	71
h953_m8	6594	151	1504	35	220	9	3704	63	3923	73
p34392_m1	2813	65	195	4	220	9	3537	63	3757	72
p93791_m12	68144	1560	8668	199	220	9	4265	71	4484	80
p93791_m19	101478	2325	8278	190	220	9	4371	74	4590	83
t512505_m7	23024	527	195	4	220	9	3537	63	3757	72
t512505_m8	19999	458	569	13	220	9	3830	67	4049	76
t512505_m9	15457	354	195	4	220	9	3464	63	3683	72
t512505_m14	141898	3246	195	4	220	9	3724	66	3944	75
t512505_m15	76600	1755	195	4	220	9	3358	60	3578	69
t512505_m16	160510	3674	195	4	220	9	3578	63	3797	72
t512505_m23	18690	428	382	9	220	9	3875	68	4094	78
t512505_m24	34306	787	195	4	220	9	3724	66	3944	75
t512505_m31	39762	912	5265	121	220	9	4387	74	4606	84
u226_m7	18299	419	3769	87	220	9	3679	64	3899	73

Table 3-4 reports total area and power for components of EARA blocks for cores (modules) of the ITC'02 SoC Benchmarks. The notation used in Table 3-4 for components of the EARA blocks are identical to those used in Table 3-2. Here, for different benchmarks, the EARA blocks amount to a total area equalling that of 250 to 300 2-input NAND gates and are consistent with the results of the UBC_SoC cores. Thus, per core, the total additional area for the DAST methodology is minimal as it amounts to the equivalent of about 600-800 2-input NAND gates.

Table 3-4: EARA Area and Power for ITC'02 SoC Benchmark Modules

	Hardware Components of EARA for ITC'02 Benchmark Modules								EARA for ITC'02 Benchmark Modules	
	PO_BSR		SO_BSR		Buffer_in		FSM		Adj_EARA	
	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)
d281_m5	30016	654	785	20	220	9	2464	37	2683	46
d695_m5	40335	883	3744	95	220	9	2561	39	2781	48
d695_m6	20218	443	1915	49	220	9	2431	37	2651	46
d695_m9	42559	933	3744	96	220	9	2537	39	2756	48
f2126_m1	69550	1529	1012	26	220	9	2643	41	2862	51
f2126_m2	18409	404	1915	49	220	9	2374	37	2594	46
f2126_m3	2769	61	207	5	220	9	1976	31	2195	40
g1023_m1	36062	792	1695	43	220	9	2513	39	2732	48
g1023_m2	28358	616	329	9	220	9	2439	37	2659	46
g1023_m4	20633	452	549	14	220	9	2399	37	2618	46
g1023_m10	50023	1088	207	5	220	9	2610	39	2830	48
h953_m1	20218	443	549	14	220	9	2431	37	2651	46
h953_m2	11900	259	329	9	220	9	2285	35	2504	44
h953_m8	9176	202	1012	26	220	9	2228	35	2447	44
p34392_m1	12563	273	207	5	220	9	2301	35	2521	44
p93791_m12	10717	236	5395	136	220	9	2260	35	2480	44
p93791_m19	57463	1249	5167	130	220	9	2626	39	2846	48
t512505_m7	4858	107	207	5	220	9	2122	33	2342	42
t512505_m8	19527	427	443	12	220	9	2374	37	2594	46
t512505_m9	16108	354	207	5	220	9	2301	35	2521	44
t512505_m14	50544	1100	207	5	220	9	2659	39	2878	48
t512505_m15	17441	384	207	5	220	9	2391	37	2610	46
t512505_m16	117642	2577	207	5	220	9	2700	41	2919	51
t512505_m23	16364	359	329	9	220	9	2326	35	2545	44
t512505_m24	17031	375	207	5	220	9	2350	37	2569	46
t512505_m31	42006	921	3281	83	220	9	2630	39	2850	48
u226_m7	8489	187	2403	62	220	9	2204	35	2423	44

The relatively small area and power requirements of the DAST components prove that embedded deterministic testing of the DAST methodology is achievable with only a small area overhead in addition to the BSR area overhead of the P1500 wrapper. In addition, these small area requirements prove the flexibility and scalability of the DAST methodology, as the area overhead of the EAS and EARA blocks appear to be essentially circuit independent. In other words, from the data of Table 3-3, PI_BSR and SI_BSR blocks are the larges blocks of the EAS blocks. Similarly, from the data of Table 3-4, PO_BSR and SO_BSR blocks are the largest blocks of the EARA blocks. However, given that cores are wrapped by the IEEE P1500 standard wrapper, these BSR are part of the standard wrapper and can be used in the EAS and EARA blocks. Therefore, the area requirements of DAST appear to be relatively constant for different circuits, proving that DAST is scalable in terms of its area and power requirements.

To compare DAST area overhead with that of IEEE P1500, Adj_EAS, Adj_EARA, and IEEE P1500 BSR areas for ITC'02 Benchmarks modules are plotted in Figure 3-11. As evident from Figure 3-11, the total area overhead of DAST (the added top two sections of each column in Figure 3-11) is in many cases a small fraction of the P1500 BSR area (the bottom section of each column in Figure 3-11). Given that the P1500 wrapper is now considered as a required DFT infrastructure to test embedded core with external testers, the results prove that DAST provides a deterministic embedded test infrastructure with only minimal area overhead when compared to ATE-based testing.

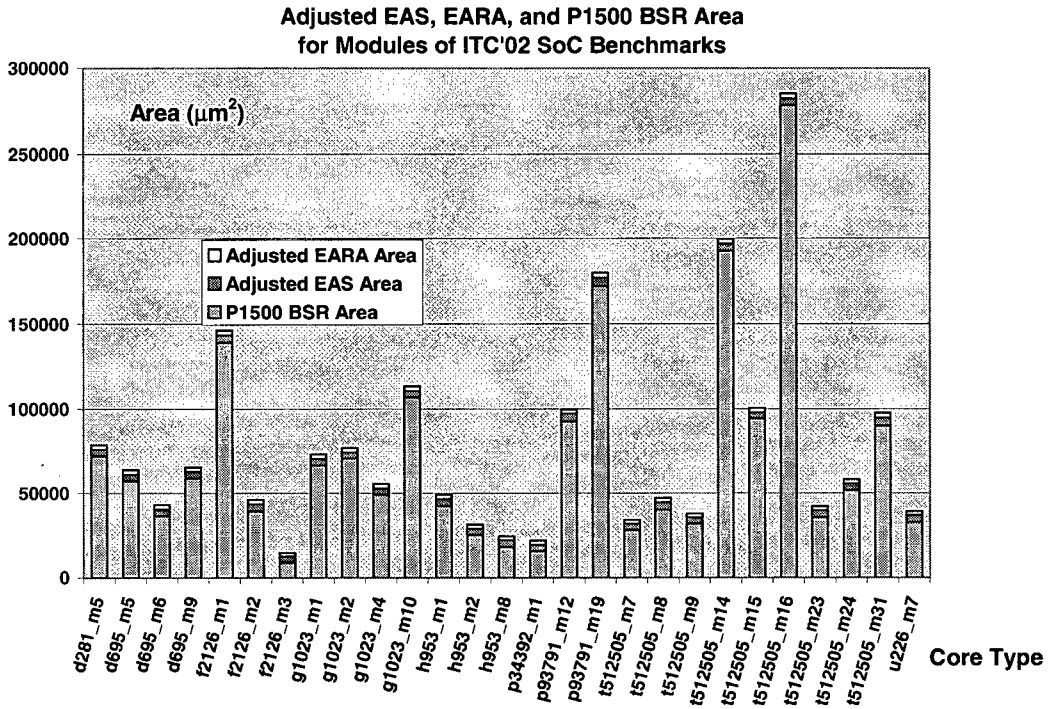


Figure 3-11: P1500 BSR and adjusted EAS and EARA areas for modules of ITC'02 Benchmarks.

The test time, using the DAST methodology, for the UBC_SoC cores is tabulated in Table 3-5. In this table *TP*, *SE*, *PI*, *PO*, *SI*, and T_{DAST} denote the number of test patterns, maximum number of flip-flops in the scan chain(s), functional input numbers, functional output numbers, scan input numbers and the simulated DAST test time (clock cycles), respectively. In the same table, T_S denotes the theoretical lower bound test time for a serial connection in a conventional external ATE-based approach, as given by Equations (3-4), (3-5), and (3-6).

Table 3-5: Simulated DAST Test Time (clock cycles) and its Test Time Models Prediction Values for Cores of UBC_SoC

	Circuit Characteristics					External Tester Time Model (Cycles)	DAST Test Time (Cycles)	DAST Time Model (Cycles)	% Error Between DAST Test Time & DAST Time Model
Circuit	SI	PI	PO	SE	TP	T_S	T_{DAST}	T_{M_DAST}	$100*(T_{M\_DAST} - T_{DAST})/T_{DAST}$
b10_1SC	1	13	6	17	52	2331	3492	3536	1.3%
b10_3SC	3	13	6	6	52	2488	3670	3693	0.6%
b15_1SC	1	38	70	449	556	328009	335780	335802	0.0%
b15_2SC	2	38	70	225	537	317356	324867	324883	0.0%

Moreover, Table 3-5 also includes T_{M_DAST} obtained from the DAST time models given by Equations (3-1), (3-2), and (3-3). From Table 3-5, these test-time models closely follow actual simulated test times.

As previously stated in Section 3.5, this thesis reports experimental results for a suite of benchmark circuits in order to provide a basis for quantitative comparison of future work in this area. In addition, since quantitative comparison with other works in this field is not practical, this thesis compares the results of its work with lower bound cases in ATPG-based external testing. This comparison is important because other works in the field target lower bound values for ATPG-based external testing. Therefore, Table 3-6 reports the estimated DAST test time based on Equations (3-1), (3-2), and (3-3), for the ITC'02 SoC Benchmark modules, and the percentage overhead compared to the lower bound conventional external ATE-based approach. Note that in Table 3-6, the bi-directional pins of a module are not included. Instead, these pins are counted in both the

primary inputs and outputs. From Table 3-6, the increase in test time with DAST is minimal and in many cases is less than 2% when compared to a lower bound serial connection in a conventional external ATE-based approach.

Table 3-6: Predicted Test Time (clock cycles) for ITC'02 SoC Benchmarks Modules in DAST Methodology

	Circuit Characteristics					External Tester Time Model (Cycles)	DAST Time Model (Cycles)	% Overhead Between DAST Time Model & External Time Model
ITC'02 cores	SI	PI	PO	SE	TP	T_S	T_{M_DAST}	$100*(T_{M\_DAST} - T_S)/T_S$
d281_m5	6	214	228	32	118	77084	78745	2.2%
d695_m5	32	38	304	45	110	226796	228345	0.7%
d695_m6	16	62	152	41	234	225420	228705	1.5%
d695_m9	32	35	320	54	12	30214	30391	0.6%
f2126_m1	8	356	529	1000	334	3034084	3038769	0.2%
f2126_m2	16	85	139	319	422	2276478	2282395	0.3%
f2126_m3	1	30	20	452	103	53351	55729	4.5%
g1023_m1	14	139	273	43	134	154712	156597	1.2%
g1023_m2	2	221	215	84	74	45898	47609	3.7%
g1023_m4	4	145	155	54	268	141474	145235	2.7%
g1023_m10	1	301	377	13	29	22858	23273	1.8%
h953_m1	4	112	152	348	341	579952	584735	0.8%
h953_m2	2	68	89	327	9	8278	8413	1.6%
h953_m8	8	35	69	189	305	504832	509111	0.8%
p34392_m1	1	15	94	806	210	209576	212525	1.4%
p93791_m12	46	361	80	93	391	1977986	1986988	0.5%
p93791_m19	44	538	437	100	210	1164676	1169515	0.4%
t512505_m7	1	122	36	514	608	462230	476223	3.0%
t512505_m8	3	106	147	1473	1025	4835456	4849815	0.3%
t512505_m9	1	82	122	530	195	151624	154363	1.8%
t512505_m14	1	751	381	1225	278	761111	767514	0.8%
t512505_m15	1	406	132	386	151	182247	185729	1.9%
t512505_m16	1	850	897	154	370	722614	727803	0.7%
t512505_m23	2	99	124	1372	532	1594686	1602143	0.5%
t512505_m24	1	182	129	1669	429	874619	884495	1.1%
t512505_m31	28	211	316	1550	3370	148431662	148478851	0.0%
u226_m7	20	97	64	54	76	99618	101375	1.8%

3.7 Summary and Conclusions

Relying on external ATE resources is insufficient for the new paradigm of billion-transistor core-based designs. The timing requirements needed for at-speed testing of many cores are beyond the capacity of external resources. Therefore, embedded testers that take over some functionality of external ATEs are increasingly deemed essential. However, to achieve high-quality test and reduce products cost, these embedded blocks need to perform deterministic tests and use the benefits of ATPG test vector sets generation.

Scan test sequence generation follows a simple protocol. A scan test sequence generation procedure can be viewed as constituted of two distinct parts: one of test *data delivery* and the other of test *data control and observation*. The test *data control and observation*, i.e., the generation of the specific scan vector sequencing tends to require much ATE resources for at-speed test and automated test application and flow. It has been shown that test resources partitioning results in lowering test cost. This chapter presented a methodology that essentially ports some of external testers capabilities onto the chip. This amounts to a methodology that is referred to as *Dedicated Autonomous Scan-based Testing* (DAST).

The implementation of DAST requires introducing a level of hierarchy between the ATPG and the embedded core under test. In DAST, a simple compilation of the raw test data is performed off-chip to append a few op-codes with the test patterns. The data and

associated op-codes are then transferred onto the chip and interpreted by an *Embedded Autonomous Sequencer* (EAS) and an *Embedded Autonomous Results Analyzer* (EARA).

With an area overhead equivalent to about 600-800 2-input NAND gates, DAST enables on-chip comparison of test results for an SoC designed with an ATPG/scan-based DFT methodology. Embedded results evaluation is enabled by sending the expected test results, in addition to the test stimuli, while EARA operates in synchrony with EAS.

DAST components were implemented in TSMC 0.18 μm for different cores of the ITC'02 Benchmarks and their area requirements were reported. In addition, and for comparison purposes, test time models for DAST scheme and the corresponding serial connection that one would expect to find in a conventional external ATE-based methodology, were developed. Per chip, the test time in DAST is marginally longer than the test time of a comparable serial ATE-based testing. However, multi-site testing in DAST is ideal. In other words, theoretically, unlimited number of chips can be tested concurrently on a single tester, as the expected test results are sent to the chip and the test comparison is performed on chip. Effectively, based on the total number of similar chips tested, test time is reduced significantly using DAST methodology when compared to typical ATE-based testing.

Clearly, there are trade-offs in the case of DAST with respect to the area overhead, test time, and reduced functionality of the external tester. Overall, the advantage of DAST is that ATPG/scan-based testing can be performed with minimal external components, providing a basis for SoC testing that is cost-effective, flexible, and portable. Moreover,

the partitioning of test resources in DAST facilitates a hierarchical test flow suited for automation of the entire test cycle. In addition, since the DAST scheme is based on the conventional scan/ATPG approach, DAST benefits from the same high test-quality, ease of use, and broad applicability characteristics associated with the conventional ATE-based scan testing methodology. Finally, benefiting from combining embedded testing with ATPG, DAST enables at-speed and in-field testing that is based on ATPG test vector sets and is deterministic.

Chapter 4
Network-Oriented Indirect and Modular Architecture for Test

4.1 Introduction

In ATE-based test architectures that are generally in use today, one implicit assumption is that a single tester, acting as both the source and the sink, is in direct control of the embedded cores and their DFT. While this assumption results in simple test protocols in many cases, it undermines the modularity in the generic test architecture by tightly coupling the elements of the test architecture, i.e., the source, sink, wrapper, and the TAM. Such a tester uses the total test bandwidth, where the bandwidth is defined to be the sum of the products of test pins and their maximum frequency of operation. In

addition, based on the test requirements of every core, the system integrator divides and fixes the bandwidth between different cores of the SoC. For ATE-based test architecture templates, the CUT is physically connected to the tester through a test-head and, to maintain timing requirements, a physical proximity between the chip and the tester is required.

In the example illustrated in Figure 1-1, there are eight test pins, of which, six pins are used for the test data and two pins are used for the test control. This pin arrangement is based on: (i) the number of test patterns for each core, assuming identical frequency of operation; and (ii) the total given number of test pins for the chip. In addition, the data lines are divided into two TAM groups, as shown in Figure 1-1, and the wrappers around the cores are designed to match the number of the core internal scan chains to the width of the TAM. The arrangement of Figure 1-1 requires a tester with eight test pins. If such a tester is unavailable in a test insertion, the TAMs need to be changed to accommodate the available tester's channels. Moreover, if after the design or in a subsequent design, any of the cores needs to be connected to a different TAM width from the widths given in Figure 1-1, the wrapper for that core would need to be changed.

As an additional example of the close coupling between the test elements of the test arrangement of Figure 1-1, the tester is forced to operate at a minimum frequency, as dictated by different delays in the *forward* and *return* paths of the TAMs. Here, the *forward* path refers to the data lines from the tester to the cores and the *return* path refers to the data lines from the cores to the tester.

From the above arguments, the diminished modularity in tightly coupled test architectures leads to a reduced flexibility in being able to modify the test architecture elements, as the modification of any part of the architecture requires subsequent changes in the other parts. For example, the addition of one extra core on a TAM can change the timing characteristics of the TAM and, hence, the operating frequency of the tester. By the same argument, the restricted modularity also results in a reduced flexibility in regards to implementing or integrating new schemes in the methodology. As chips become more complex, a high flexibility for implementing new schemes in the test methodology is key for keeping the overall test cost low and productivity high (see Section 2.3). Test architecture and methodology with such flexibility are considered to scale well with changes in the test design and flow, and are referred to be scalable in this thesis.

Moreover, with a tightly coupled test model, testing requires close proximity of a chip and the tester. Hence, in-field testing, i.e., testing the chip when in its target system, or remote-access testing, is virtually impossible or extremely costly. In addition, multiple testers cannot test a chip simultaneously. A multiple-tester arrangement is key if test resource pooling is required between multiple sites or companies to reduce test cost. Furthermore, using a multiple-tester arrangement can be cost-effective when cores test speeds vary significantly. In such a scenario, it is more cost-effective to reserve the high-speed tester's channels for fast cores and use low-speed channels for lower speed cores. A multiple-tester arrangement is in direct contrast to a test architecture where operating at

the lowest frequency is the fundamental mechanism used to solve the disparity between the speed of tester, I/O pins, the TAM, and the cores. Volkernik *et al.* in [35] have shown the cost benefit of bandwidth matching between a tester and cores.

As the first steps towards developing solutions addressing the issues discussed above, in this work a *Network-oriented, Indirect and Modular Architecture* (NIMA) is proposed where different testers can connect to a common switching fabric and send test data to the cores under test. NIMA is considered to be a special communication network that consists of hardware and software that allow the transportation of test stimuli and expected results from multiple sources to multiple cores. In addition, a switching fabric for the test architecture is proposed in this thesis. However, the NIMA architectural design is such that it can be migrated into a template, where the common switching fabric used for core interconnections is also used for testing the SoC.

The indirect methodology of NIMA breaks the coupling between the core, the TAM, and the tester by de-coupling *test-data processing* and its *communication*. In this thesis terminology, *test-data processing* refers to the functional behaviour of source/sink and wrapper, whereas, *test-data communication* refers to the interaction between source/sink and the wrapper. Hence, NIMA alleviates the previously discussed problems associated with tightly coupled test models. NIMA provides the basis for modular test design and programming. NIMA also enables single or multiple testers to send test data over a *local area network* (LAN) or a *wide area network* (WAN), such as the Internet, to an SoC in-field in order to test the chip in its target system. Finally, the control mechanism of the

test architecture is incorporated in the packets of NIMA. Hence, compared to typical test architectures that require separate data and control lines in their architectures, fewer test pins are required when using NIMA.

4.2 NIMA Concept

The key concept in the NIMA is establishing an indirect[2] digital communication path, from multiple sources to multiple destinations, through a switching fabric [94][95]. Testing ICs using the NIMA scheme requires that test stimuli and expected test results for cores are first compiled into new formats and then encapsulated into packets. These packets are subsequently augmented with control and address bits such that they can autonomously be transmitted to their destination through a switching fabric. Owing to the indirect nature of the connection, embedded autonomous blocks at each core are responsible for applying the test to the core and comparing the test results. To simplify the requirements of these embedded blocks, this thesis assumes that: 1) The packets can arrive at their destination cores with varying delays; 2) The packets arrive in their original order; 3) No packet is lost in the communication link.

The proposed NIMA is illustrated in Figure 4-1 as a block diagram, where IP cores, a switching fabric, on- and off-chip sources, *Embedded Autonomous Sequencers* (EAS), *Embedded Autonomous Results Analysers* (EARA), an *interface matching* (IM) block, and an optional transmission system are shown.

[2] The indirect property of NIMA refers to tester's lack of direct control over a core's DFT.

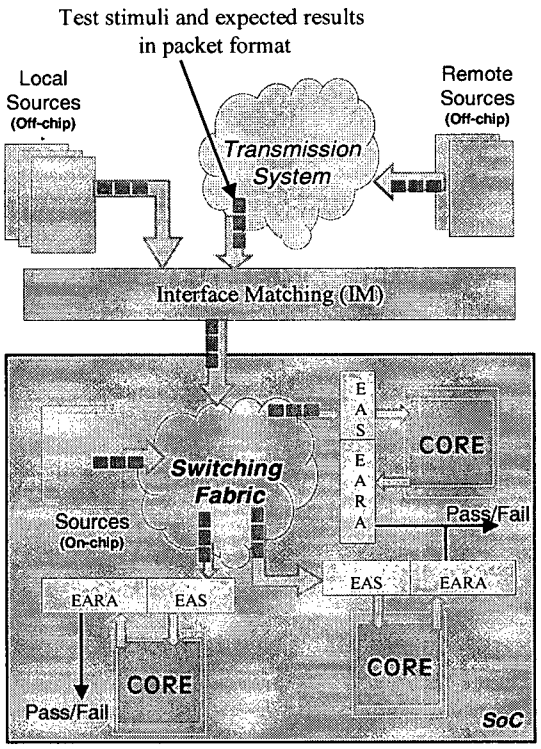


Figure 4-1: Conceptual representation of NIMA.

In NIMA, test stimuli and expected results are encapsulated into packets, hence, dedicated blocks at each core, indicated as EAS blocks in Figure 4-1, extract test stimuli from the incoming packets and apply them to their respective core. Moreover, in synchronism with the application of the test vectors by the EAS blocks, dedicated EARA blocks compare test results at the output of their respective core to the expected results within the incoming packets.

As illustrated in Figure 4-1, the connection between off-chip sources and the IM block can be direct and/or through a transmission system such as a WAN or a LAN. The IM block in Figure 4-1 receives test packets from different test sources with varying bandwidths and matches these bandwidths to those of the SoC, where the bandwidth is defined to be the product of the frequency and the number of the channels. As an example, consider the case in Figure 4-2 where four different sources send test data to the SoC. These four sources have five, three, two, and one channels, respectively. Moreover, these sources can send data at maximum frequencies of 50, 100, 100, and 150 MHz per channel, respectively (note that, there are no control lines associated with these sources, as control signals are embedded in the incoming packets). In addition, in Figure 4-2 three different groups of test input pins to the SoC can be identified. These three groups have 2, 2, and 1 channels with maximum frequency of 200, 100, and 150 MHz per channel, respectively. According to the sequence of the incoming packets, the IM block matches the total incoming bandwidth of 900 MHz to the total SoC bandwidth of 750 MHz.

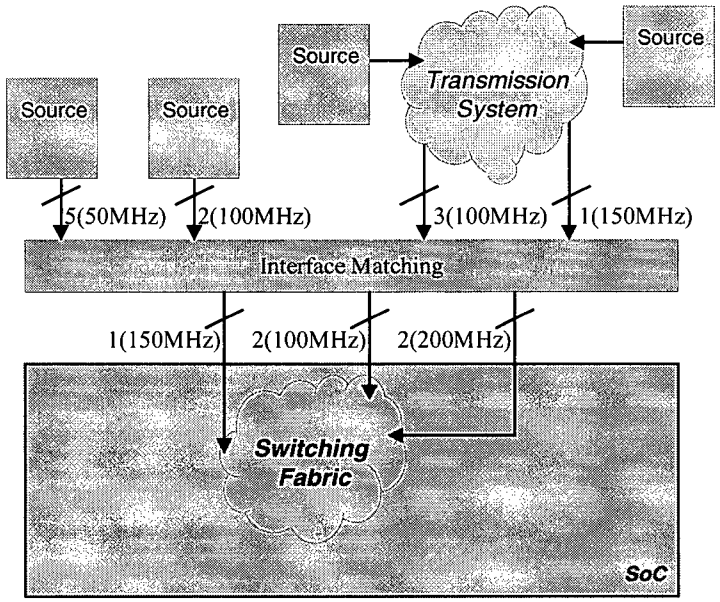


Figure 4-2: An example showing function of the IM block.

To help with partitioning of test resources, NIMA can be regarded as a special communication network, consisting of hardware and software that allow transportation of test stimuli and expected results from multiple sources to multiple cores. Regarding NIMA as a special communication network helps in using the cumulative knowledge in different communication networks, and to apply this knowledge for the specific case of a test network on-chip. To promote the modularity and simplicity of the design tasks, a layered architecture with a formal interface between each layer is the most established method to divide the functions implemented by communication networks [96]. Since the communication tasks involved in SoC testing are not as complex as those in other communication networks, this thesis uses a 3-layer model for NIMA that consists of a

Physical Layer, a *Network Layer*, and an *Application Layer,* as shown in Figure 4-3. In this model, the tasks in one element of the architecture deal directly with tasks in another element within the same layer through a *virtual* link such that modifications in other layers do not alter the protocol within a layer. The physical connection, however, is only in the vertical direction of the model except in the *Physical Layer*, where there is no virtual link but only a physical link.

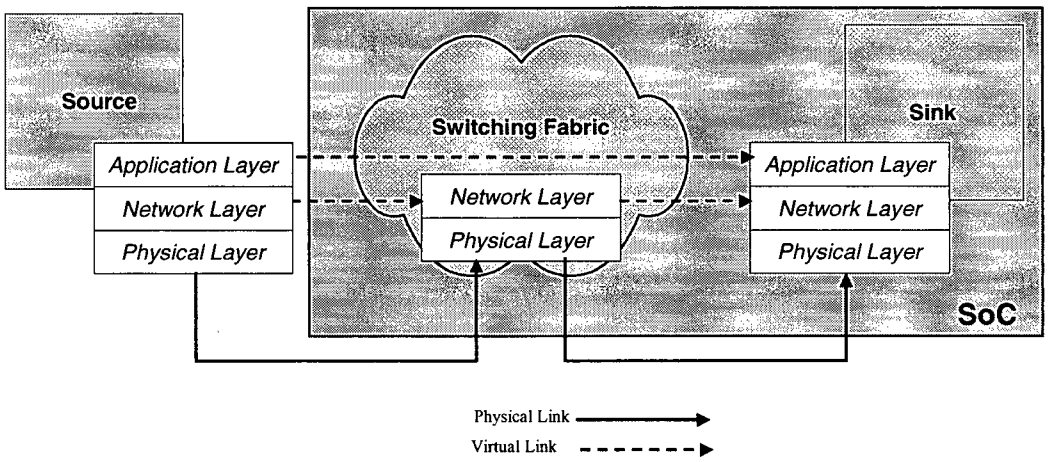


Figure 4-3: The conceptual 3-layer model in NIMA.

4.3 Physical Layer

The *Physical Layer* encompasses the actual interconnection medium. For contemporary ICs, the physical medium consists of the metal wires routed between the SoC blocks. This layer specifies the voltage level of the signals on the wires, the timing of the signal

events, the signalling techniques, and other physical properties of the link, such as protection measures against cross talk. Finally, the *Physical Layer* presents the data as a stream of 1's and 0's to the *Network Layer* in the present design of NIMA. This thesis assumes that the reliability of the physical link can be maintained in a chip, and hence no error detection mechanism is implemented. In addition, this thesis assumes a synchronous transmission in NIMA where a separate clock is provided next to the data bit stream. However, it is possible to encode the clock in the data and later extract it in the switching fabric [96].

Although electrical signals over metal wires are the dominant physical link in today's chips, it does not mean that NIMA cannot use a completely different physical connection, such as guided or unguided electromagnetic waves, in the future. One benefit of a layered approach in NIMA's design is the possibility of utilising new approaches with minimal design efforts as new techniques become available and when their uses are justified.

4.4 Network Layer

The *Network Layer* dictates the details regarding how data is transmitted across the network, the switching technologies used, and the network topology implemented. The design of the *Network Layer* in NIMA is greatly influenced by the packet format used. Hence, this section starts by describing the design of the packet format. Following the design of the packets, the design of switches in the fabric, the addressing mechanism, and the routing strategy in the *Network Layer* of NIMA are provided.

4.4.1 Packet Format

The most fundamental design issue is the packet format. The packet format in NIMA is illustrated in Figure 4-4.

Figure 4-4: NIMA packet format.

Here, a brief description of each field is provided.

Sync Word Field:

The *Sync Word* signals the beginning of the packet to a switch. The starting point of a packet within the incoming bit stream needs to be identified for the switches, as the communication link in NIMA can be idle for any length of time. The predefined pattern in *Sync Word* identifies a new packet to the switches.

Data Length Field:

Packets in NIMA can have varying data length to accommodate different requirements of cores in terms of test data length. The *Data Length* field identifies the length of the embedded data in the packets to the switches, and hence, enables the switches to switch entire packets to the proper output channel.

Address Length Field:

As discussed later in Section 4.4.3, a novel dynamic addressing mechanism is used to promote flexibility and scalability in seamless integration of hierarchical SoCs. A variable length *Address* field is used to achieve the dynamic addressing mechanism of Section 4.4.3. The *Address Length* defines the length of the *Address* field, and, in effect, acts as a counter to the number of *locations* in the *Address Field.*

Address Field:

The *Address* field holds a variable length of address bits. The details about this field are provided later in Section 4.4.3.

Data Field:

The *Data* field holds a variable number of data bits (alternatively referred to as the payload in this thesis). The payload includes test data as well as other related patterns for the *Application Layer.*

The sizes of the fields in NIMA packet format, in terms of number of bits, are denoted by S, L_D, L_A, A, and D as shown in Figure 4-4. For the *Address* field, A represents the number of locations in this field where each location contains n bits. This results in switches having 2^n output channels. In addition, the first four fields, identified in Figure 4-4 with a different colour/shading from the last field, together constitute the header of a packet. In the header of the packets, the relative positions of the *Data Length* and *Address Length* fields have been chosen arbitrary as shown in Figure 4-4.

4.4.2 Switches

Switches in NIMA have one input and 2^n output channels. To promote the scalability of the NIMA design, each channel in the switch is of width *2k* and is comprised of test-stimuli and test-results sub-channels. The black box diagram of a switch in NIMA, with *n=2,* is shown in Figure 4-5.

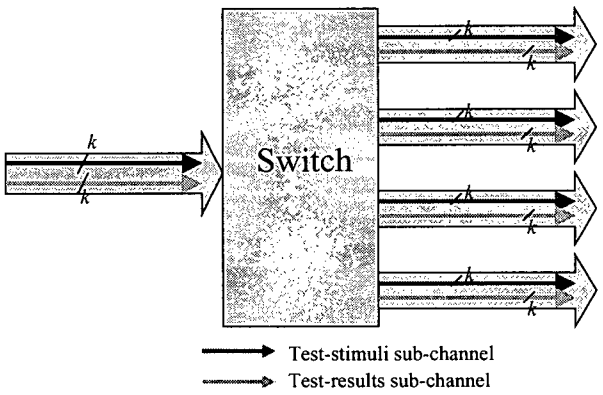


Figure 4-5: Black box diagram of a switch in NIMA with four output channels.

For switches in NIMA, the *least significant bit* (LSB) of the test-stimuli sub-channel is denoted as the *primary line* and only this *primary line* follows the format of NIMA packets. For this reason, the packet header is only defined in the *primary line*, denoted by *Line 0* in Figure 4-6. As illustrated in Figure 4-6 and Figure 4-7, with the header only defined in the *primary line* of the test-stimuli sub-channel, part of the switch channel bandwidth is reserved and thus not used. However, this arrangement results in a scalable

architecture where the value of k can range from one to its maximum available value without any change in the design of switches or NIMA's *Network Layer* protocol. That is, the same generic code for the switches can be used to instantiate the NIMA network for a particular design.

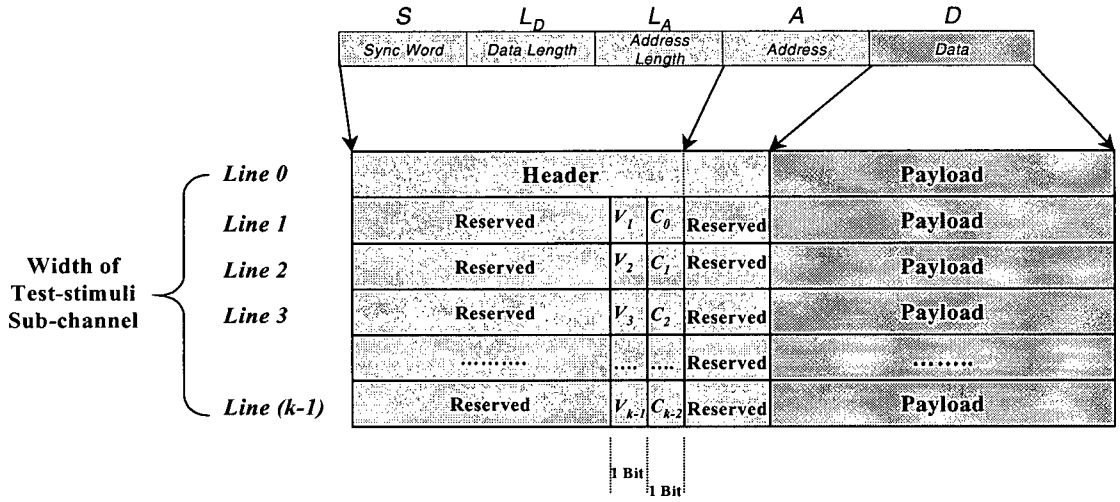


Figure 4-6: Occupancy of test-stimuli sub-channels in NIMA switches.

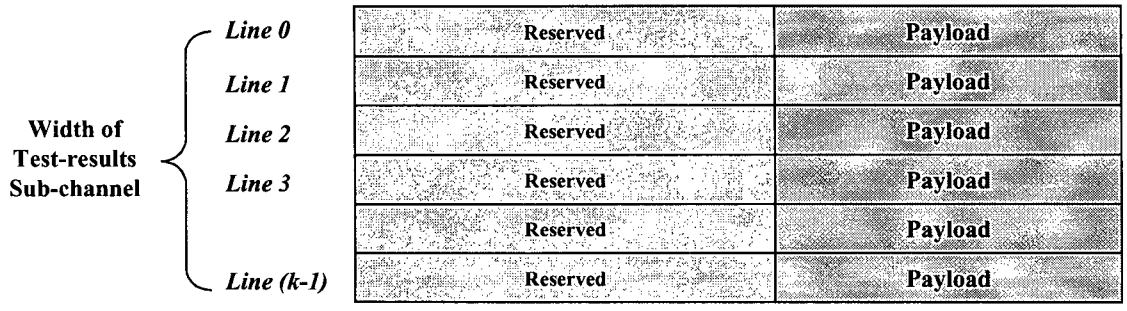


Figure 4-7: Occupancy of test-results sub-channels in NIMA switches.

Line 1 to *Line (k-1)* in both the test-stimuli and the test-results sub-channels can either hold valid or invalid payload in a given packet. The *V* flag bits shown in Figure 4-6 help the tasks in the *Application Layer* of NIMA to identify lines with valid payload where a line with valid payload is identified with logic 1. Note that, in terms of having valid or invalid payload data, the status of corresponding lines in the test-stimuli and the test-results sub-channels are identical. Hence, for example, if *Line 3* to *Line (k-1)* do not carry valid payload data in the test-stimuli sub-channels, the same lines do not carry a valid payload in the test-results sub-channels. In addition, note that if a packet is present on a switch's output channel, *Line 0* in both sub-channels carry valid payload data.

In addition to payload validity, the tasks in the *Application layer* require further payload information. As shown in Figure 4-8, the payload is divided into an array of k x D bits for some applications. However, certain payloads may not occupy the trailing bits of this array in all the k bits of the output sub-channels. Figure 4-8 shows an example for a design with $k=6$. In this case, a payload of length 81 bits is divided into an array of 14x6 bits, where an X denotes invalid bits in the array. As shown in Figure 4-6, *C* flag bits can be programmed in *Line 1* to *Line (k-1)* of the test-stimuli sub-channel to identify the invalid bits in the array.

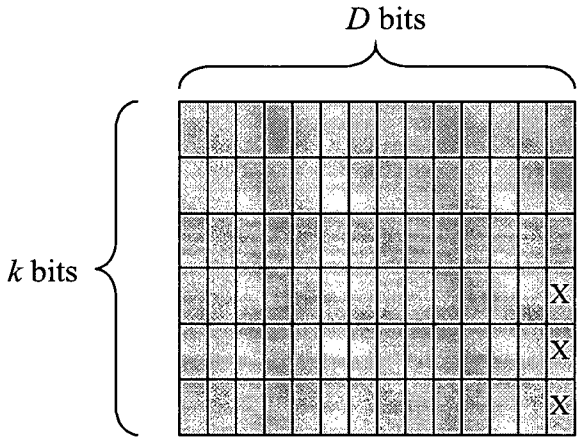


Figure 4-8: A typical payload array in sub-channels with invalid last bits marked by "X".

However, the *C* flag requires only $\lceil \log_2 k \rceil$ bits and is used as a binary number, $|C|$, in the *Application Layer*. This number only shows the status of the trailing bits in *Line 1* to *Line (k-1)* of the sub-channels, as *Line 0* in the sub-channels will always have a valid trailing bit in the existence of a packet. Thus, the number of lines with an invalid trailing bit is equal to $k-1-|C|$. For the case shown in Figure 4-8, the *C* flag will consist of three bits and is equal to *010,* indicating *6-1-2=3* lines having invalid trailing bits.

In addition, switches in NIMA can calculate the end of the packets before the last bit of the packet leaves the switch. Thus, theses switches can instantly look for the next incoming packet after the last bit of the previous packet. Therefore, packets in NIMA can be pipelined one after another without any gaps between them.

4.4.3 Dynamic Addressing Mechanism

Using the packet structure of Figure 4-4 and the packet format as explained in Section 4.4.1, the number of the bits in the *Address Field* is: $A \leq n \times 2^{L_A}$. Since the *Logical Address Space* (LAS) in NIMA is $LAS = 2^A$, therefore $LAS = 2^{n \times 2^{L_A}}$. If the values of n and L_A are chosen appropriately, the LAS can be very large, and hence enable the system designer to use partial addressing without significant disadvantages. This partial addressing enables seamless integration of new cores as well as design of hierarchical SoCs. Using variable length addressing, the entire LAS is essentially divided into 2^{L_A} hierarchical levels each comprised of 2^n pages. As an example, assume $n=2$ and $L_A=10$. For this case, $LAS = 2^{2048}$ and the LAS is divided into 1024 hierarchical levels each of 4 pages.

The value of A, the length of the *Address field*, need not be constant and can be chosen such that $N \leq 2^{n \times A}$, where N is the total number of cores to be accessed for testing. As an example, consider a system with 49 cores. For this example, the minimum value of A is three as $49 < 2^{2 \times 3}$. Thus, based on the configuration of the network, all the cores can be addressed with a minimum of three levels of switches. Now, assume that a later revision of the design requires the use of 73 cores instead of 49. For this case, the minimum value of A is four as $73 < 2^{2 \times 4}$. Similarly and based on the configuration of the network, all the cores for this new revision can be addressed with a minimum of four levels of switches. Hence, a configurable and modular architecture is provided, where

the size of the network scales and grows according to the number of the cores without any design change.

The *Physical Address Space* (PAS) is defined as the part of the LAS that is actually being used. The PAS can vary in size and it can be chosen to be the minimum size required. When new cores in later iterations/revisions are added to the SoC and if the PAS is all assigned, new levels of addresses can be introduced to increase the size of the PAS. Thus, new cores can be accommodated with no design modification of the *Network Layer*. This yields the advantage of a scalable TAM architecture between SoC design versions, referred to here as *design-version scalability*.

Moreover, if an existing SoC is used as an embedded core in a subsequent SoC, the PAS of the earlier SoC is assigned to the lowest part of the subsequent SoC's LAS. The PAS of the subsequent SoC is then continued from the next available hierarchical level of the 2^n pages. This creates another level of scalability: *multi-level scalability*. Again, using the *design-version scalability* feature, new cores in the subsequent SoC are assigned addresses in the PAS. *Design-version scalability* and *multi-level scalability* are illustrated conceptually in Figure 4-9.

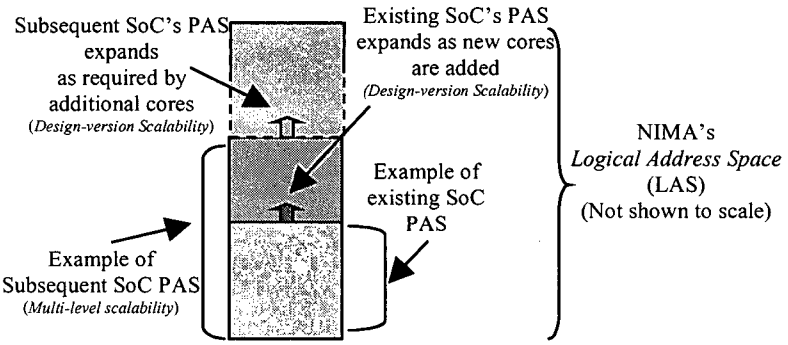


Figure 4-9: Address spaces for NIMA's *Network Layer*.

It is possible to connect each first-level switch to a primary I/O pin of the system chip. This effectively divides the entire chip space/address into subsections with uncorrelated networks that can be individually addressed and accessed. Hence, the *Effective Logical Addressing Space*, LAS_{Eff}, for an n-dimensional architecture can be sized to be: $LAS_{Eff} = m \times 2^{n \times 2^{L_A}}$, where m is the number of chip primary I/Os. Note that in general, $m \geq 2k$, where $2k$ is switches channels width, as defined in Figure 4-5.

A feature of the proposed dynamic addressing mechanism is the limiting of the latency of packets in the switches. To illustrate this, consider an alternative *Network Layer* where fixed address-length addressing mechanism is used. For such an example, assume that the packets use B bits in their address field. A switch in this hypothetic *Network Layer* can only decode the address, before routing the packets, when all the bits in the address field have been received. Thus, the latency introduced by each switch in this case is proportional to B. In the case of the NIMA, an address length pointer is used and hence, a switch can decode the address field as soon as a minimum number of address

bits are received, i.e., n bits in the proposed architecture (see Section 4.4.4). Hence, the introduced latency by a switch in NIMA is proportional to L_A+n.

In order to be able to address more than five hundred cores in the case of the fixed address-length architecture, B needs to be greater than nine. However, in the case of NIMA, L_A+n can be as low as five where $n=2$, $L_A=3$ and, hence, $500 < 2^{2\times 2^3}$. Moreover, the smaller latency in NIMA results in the need for smaller buffering when compared to a fixed address-length addressing mechanism. In addition, the used address bits in each NIMA switch can be taken out of the packet and discarded. This progressively shortens the length of the packets, and hence, the introduced latency, as packets move in the network.

4.4.4 Routing in the Switches[3]

To eliminate the need for maintaining routing information in the switches, the packets in NIMA are routed using source routing, i.e., implying that the route is predefined and hardwired. As described in Section 4.4.1, the *Sync Word* in the *primary line* identifies the beginning of an incoming packet to the switches. The switches in NIMA save $S + L_D + L_A$ and the first n bits in the *Address* field of a packet as defined in Figure 4-4. The switches then use the first n bits in the *Address* field of the packets to decode the

[3] In this thesis, the *Network Layer* is based on virtual indirect connections provided by switches. That is to say, to simplify NIMA, packets are sent in a particular order and are required to reach their destination in that order. Designs that are more elaborate will include out-of-order packet reception that is not considered in this work. Future studies can also look into the effects of out-of-order packet reception in terms of design complexities and cores' fault coverage in cases where packets are not put back in their original order at the cores.

destination of the packets and identify the output channel to which the packet will be forwarded. As soon as the output channel is determined, the packets are routed to that channel. This ensures a minimal number of buffers and a maximum of $S + L_D + L_A + n$ clock cycles delays in the switches. This technique is referred to as *cut-through* routing [96]. In *cut-through* routing, packets are forwarded to the router output as soon as the destination is known, without waiting for the tail of the packet to arrive. This technique reduces latency of the network and reduces the amount of required hardware in the form of buffers in the routers.

4.5 Application Layer

In NIMA *Application Layer*, the data is comprised of test vectors, test results, DFT and wrapper control signals that have to be converted to/from packets. The NIMA architecture allows arbitrary bit widths for switches sub-channels, k, such that packets in a sub-channel can be blocks of k-dimensional bit arrays instead of a one-dimensional bit stream (note $m \geq 2k$ where m is the total primary test pins). As a result, the cores that require the use of the test network must have the capability of scaling the packet payload into the bit width suitable for their application. Moreover, for packets wider than one bit, handshaking *ready* signals are required to indicate which bits are valid, as the data may not completely fill the entire payload array. This *ready* signal is essential to maintain a constant interface protocol between the *Network* and the *Application* layer. In effect, the information in the V and C flags as described in Section 4.4.2 can be used for the *ready* signal.

Since cores on an SoC are assumed to possibly have different origins, a standardised test interface or wrapper at each core is critical. Moreover, a standard language for the description of core test programs and DFT is also essential. As described in Section 2.5.1, one such standard, referred to as P1500 [51], is currently under development by the IEEE and is used in this thesis. Specific P1500 instructions are required to support full scan test as well as any other DFT strategies. Furthermore, a mechanism is needed to generate the P1500 control signals in the correct sequence at the core site. The mechanism needed to generate the P1500 control signals and the test programs based on *Core Test Language* (CTL) [51] fall in NIMA's *Application Layer*.

For cores with a P1500 wrapper, a set of P1500 wrapper control flags is devised in the *Application Layer* of NIMA. These control flags enhance the flexibility and scalability of NIMA such that NIMA remains independent of future modifications or improvements to the P1500 standard, as long as the interface between the *Application Layer* and the *Network Layer* is maintained. In the design of this thesis, these control flags are based on a set of wrapper control bits embedded in the payload that results in an additional hierarchy in the message that is to be packetized for each core. These bits serve as instructions for the P1500 control mechanism. Hence, any modifications to the P1500 wrapper will result in the modification of these bits or their interpretation in the *Application Layer*.

As shown in Figure 4-1, blocks of EAS and EARA from the DAST methodology of Chapter 3 are other *Application Layer* tasks. EAS blocks are responsible for receiving

test stimuli and applying them according to the instruction embedded in the payload. These instructions detail the data bits that must be applied to the pins of the core under test, and the timing at which they are applied. EARA blocks, on the other hand, receive the expected results and compare them to the output of the core. Using internal signalling, EARA blocks operate in synchrony with the EAS blocks to ensure the integrity of the test process.

Another task in the *Application Layer* is decompression of the incoming test data if compression is used on the original test data. Compression and decompression of test data can help toward reducing test time, and hence test cost, by reducing the required memory in the external tester and the number of bits needed for communication. There are many different compression and decompression techniques [40], and if used, their tasks fall within the *Application Layer* in NIMA.

Scheduling is another *Application Layer* task. There are many proposed schemes for SoC test scheduling in the literature [85][86][87][88][90], and the results of these works can be used for scheduling in NIMA. The problem definition in NIMA scheduling is similar to the above works, and is to determine when to send test packets for each core such that, while keeping within the bounds of any given constraints, the overall test time and the hardware complexity are minimised. For instance, the scheduler must prevent conflicts in the network resource and, at the same time, minimise the test time. During the test, activating all core DFT simultaneously may result in power dissipation that exceeds the chip junction or package heat tolerance. Hence, based on a given power

budget, the scheduler needs to control the activity of the DFTs [98]. Moreover, the test data from the NIMA network may arrive at faster rates than can be consumed by a core. Thus, a buffering scheme is required to ensure the core gets the data only when it is ready to accept it. Otherwise, the integrity of the test will not be maintained. To prevent buffer overflow, constraints must be applied to the time intervals between consecutive packets destined for the cores.

4.6 Implementation

To validate the concept, NIMA was implemented on the UBC_SoC platform of Section 3.5 and a chip was fabricated. The design and implementation flow follow a typical digital ASIC deign and fabrication flow that uses *electronic design automation* (EDA) tools. This section provides the details of the implementation for the three layers of NIMA's architecture.

4.6.1 Physical Layer

For the *Physical Layer*, the traditional metal interconnect was used. Automatic tools, as part of a standard ASIC design flow, routed the wires in the implementation of NIMA. The voltage on the wire is expected to swing between 0V and 1.8V, which is a typical range for the targeted 0.18 μm technology. The analysis suggested that the capacitive coupling between routed wires would not be severe enough to cause erroneous signal transition. Therefore, no additional hardware for cross-talk protection was implemented.

4.6.2 Network Layer

To mirror a real SoC with embedded cores, it was assumed that there is no direct access to the cores in UBC_SoC from its primary inputs/outputs (I/Os). Moreover, it was assumed that only three test pins were available in the platform. Based on these assumptions, the switching fabric was designed such that cores can be accessed through a maximum of two levels of switches. Figure 4-10 illustrates the interconnect architecture used for the switching fabric in UBC_SoC.

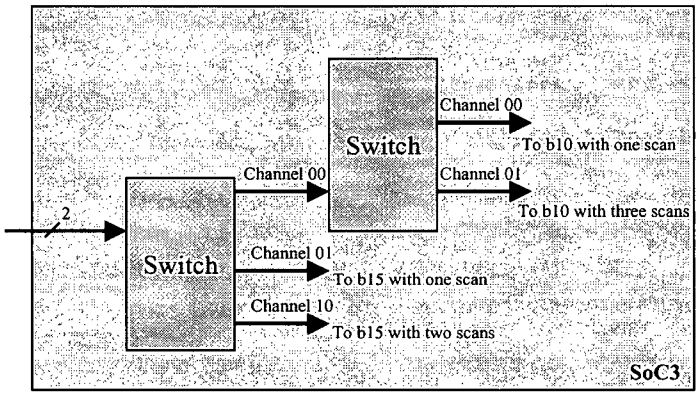


Figure 4-10: Interconnect architecture for the switch fabric in UBC_SoC.

Figure 4-11 illustrates a block diagram of the switches in the NIMA implementation. In Figure 4-11, incoming packets are first buffered. Subsequently, appropriate logic informs the *finite state machines* (FSM) of the detection of the *sync word* that, in turn, indicates the beginning of a packet. After the detection of a packet, the FSM provides necessary signals for the *Packet End* block such that it calculates the length of the packet.

The FSM also provides necessary signals for the *Last Switch Detection* block. Based on the signal from the *Last Switch Detection* block, if the switch is a last switch, i.e., the switch that is directly connected to cores, the switch only outputs the payload and not the entire packet. As detailed in Section 4.4.2, based on the valid or invalid status of each link in the output channels and the trailing bits in the payload array, necessary information is extracted from the packets to generate the *Ready* signal, as the processes in the *Application Layer* need the *Ready* signal to interpret the payload. Finally, the FSM provides necessary switching signals for the *De-Mux* block to route the packet, or the payload in the case of last switches, to the appropriate output channel.

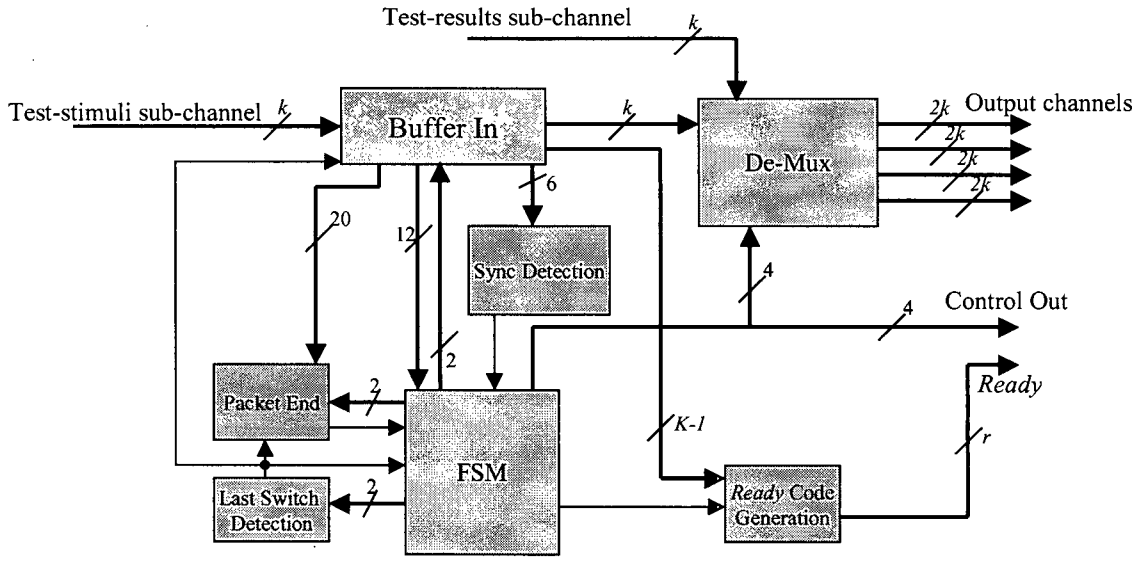


Figure 4-11: Block diagram of a switch in the NIMA implementation.

The switches in the *Network Layer* that process the packets header and eventually route packets, were written in VHDL RTL and synthesised with the following parameters: $S=6$, $L_D=10$, $L_A=6$, $n=2$, and $k=1$. The switches buffer the incoming data for a maximum of $S+L_D+L_A$ bits in shift registers. In the present implementation, this amounts to 22 bits. After detecting an incoming packet from a *Sync Word* in the *primary line*, switches use the first two bits in the *Address* field to decode the destination of the packet and identify the output channel to which the packet will be forwarded. In the present implementation, after the decision on the routing is made, the switch deletes these two bits from the address field, updates the *Address Length* field, and passes the packets to the appropriate output channel.

The *Ready* signal is of width $\lceil \log_2 k \rceil + k$ bits and is the concatenation of three parts. The LSB is a general flag showing if a valid packet is present on a switch's output channel. The next $k-1$ bits are a direct copy of the V flag bits in the test-stimuli sub-channel, as shown in Figure 4-6. The remaining bits are a copy of the C flag bits, as shown in Figure 4-6 and explained in Section 4.4.2.

4.6.3 Application Layer

The implementation of DAST, as given in Section 3.4, was used with minor modification to interface to the *Network Layer* of NIMA. The required modifications were mainly in the FSM blocks of Figure 3-7a and Figure 3-7b such that the control signals to these FSM could be driven from the control signals of a switch, as shown in Figure 4-11.

Moreover, pre-processing *C* programs were developed to construct NIMA *Network Layer*'s packets, and to encapsulate EAS and EARA data files into the packets payload. The pre-processing can be executed off-line with minimal computational effort. These programs accept as inputs the binary files of *Test Stimuli* and *Expected Test Results,* as illustrated in the Figure 3-4, and packetize the data in these files according to the test-stimuli and test-results sub-channels definitions given in Figure 4-6 and Figure 4-7. This packetizing task is performed by identifying sections of the data in the *Test Stimuli* and the *Test Results* files that can appear in the payload section of the packets such that the maximum length of the payload field is not violated and the *Application Layer* data is not corrupted. Subsequently, these programs add the appropriate packet header information such that the switches in the NIMA *Network Layer* can autonomously route the payload to the appropriate core.

The scheduling of the packets is trivial, as three test-pins are used in the present implementation of NIMA. Using three test-pins and assuming the same frequency for different layers of the test architecture implies that the rate of incoming data is always smaller than the rate of data that can be used at the cores. NIMA implementations with more test pins can use more sophisticated scheduling techniques.

4.7 Experimental Results

This section analyses and reports the area and the power requirements for the current implementation of switches in the *Network Layer.* In addition, to provide an understanding of the area and the power requirements for the current implementation of

NIMA *Application Layer*, it reports these for a selection of ITC'02 Benchmark Circuits modules. Finally, Section 4.7.2 presents the test times for cores of UBC_SoC. Using these empirical values, a test time model for the current implementation of NIMA is developed. This model is then used to predict the test times for a selection of ITC'02 Benchmark modules.

4.7.1 Area and Power Overhead

Using Synopsys Design Compiler™, the area and power requirements for the current implementation of the switches in the *Network Layer* are 12656 μm^2 and 3026 μW, respectively. Therefore, the area overhead of NIMA *Network Layer* is equivalent to about 1250 2-input NAND gates per switch.

The modifications to the design of DAST components to interface with NIMA switches are minimal. Hence, the area and power for components of EAS and EARA blocks for cores (modules) of the ITC'02 SoC Benchmarks are identical to those provided in Table 3-3 and Table 3-4, respectively. Thus, the total additional area for the present NIMA *Application Layer* is minimal as it amounts to the equivalent of about 600-800 2-input NAND gates.

4.7.2 NIMA's Test Time

The test time models given in Equations (3-1), (3-2), and (3-3) can be used to predict the required total data bits for EAS and EARA in NIMA (denoted as *DDM* in this thesis). This holds true, as the models are equal to the number of the clock cycles needed to apply

all the data bits and, hence, are equal to the total data bits for EAS and EARA (note that the sizes of EAS and EARA data files are equal).

In NIMA, all the data bits for EAS and EARA blocks are encapsulated in the payloads. Hence, the total number of packets needed to accommodate all EAS and EARA data bits can be modelled as *Packets* such that

$$Packets = \gamma \left\lceil \frac{DDM}{D} \right\rceil \tag{4-1}$$

where $\gamma > 1$ is an empirical calibrating coefficient[4], *DDM* denotes EAS and EARA data bits, and *D* represents the size of the payload in bits.

Finally, the test time model for any core in the present implementation of NIMA is denoted by T_{MN} and according to the following expression

$$T_{MN} = Packets \times (22 + A) + DDM \tag{4-2}$$

where *A* denotes the number of the address bits used for the core (for example, using Figure 4-10, $A=4$ for b10 with three and one scan chains and $A=2$ for b15 blocks). In addition, in Equation (4-2) the first part is equal to the total header bits in all the packets and *DDM* represents the total payload data for the core.

[4] Due to their structure, the EAS (or *Test Stimuli*) and EARA (or *Expected Test Results*) data files may not necessarily divide into complete *D* bits segments. In many cases, these segments, that represent the payloads, will be shorter than the maximum *D* bits allowed. The calibrating coefficient, γ, in Equation (4-1) is, hence, used to compensate for this fact.

For each core of UBC_SoC, Table 4-1 reports the simulated NIMA test time, T_N, for the implementation of UBC_SoC with *D=1000*. In addition, Table 4-1 reports the value of T_{MN} as given by Equation (4-2), where $\gamma=2$ and *D=1000*. Moreover, to compare NIMA test time to that of a serial connection in conventional test architectures, using Equations (3-4), (3-5), and (3-6), T_S is calculated and reported in Table 4-1. Finally, the last column in Table 4-1 reports the percentage error between the experimental NIMA test time, T_N, and the time model, T_{MN}, as given in Equation (4-2).

Table 4-1: NIMA Simulated Test Time and its Test Time Model Prediction Values for Cores of UBC_SoC (in clock cycles)

	Circuit Characteristics					External Tester Time Model (Cycles)	NIMA Test Time (Cycles)	Total Test Data Bits	NIMA Time Model (Cycles)	% Error Between NIMA Test Time & NIMA Time Model
Circuit	*SI*	*PI*	*PO*	*SE*	*TP*	T_S	T_N	*DDM*	T_{MN}	$100*(T_{MN} - T_N)/T_N$
b10_1SC	1	13	7	17	49	2199	3618	3335	3543	-2.1%
b10_3SC	3	13	6	6	52	2488	3714	3693	3901	5.0%
b15_1SC	1	38	71	449	1065	629940	686945	644859	675819	-1.6%
b15_2SC	2	38	72	224	1057	626268	672202	641075	671891	0.0%

Assuming no control lines are required in the conventional external test architectures, Table 4-2 reports the estimated NIMA test time based on the model captured by Equation (4-2) for the ITC'02 SoC Benchmark modules and the percentage overhead compared to conventional external test architectures. T_{MN} in Table 4-2 is reported using the maximum value of *D=1023*, assuming *A=2* for every core, and using the same value for γ as used in

Table 4-1. Note that in Table 4-2, the bi-directional pins of a module are not included. Instead, these pins are counted in both the primary inputs and outputs.

Table 4-2: Predicted NIMA Test Time for ITC'02 SoC Benchmarks Modules (in clock cycles)

	Circuit Characteristics					External Tester Time Model (Cycles)	Total Test Data Bits	NIMA Time Model (Cycles)	% Overhead Between NIMA Time Modle & External Tester Time Model
ITC'02 cores	*SI*	*PI*	*PO*	*SE*	*TP*	T_S	*DDM*	T_{MN}	$100*(T_{MN} - T_S)/T_S$
d281_m5	6	214	228	32	118	77084	78745	82441	6.9%
d695_m5	32	38	304	45	110	226796	228345	239097	5.4%
d695_m6	16	62	152	41	234	225420	228705	239457	6.2%
d695_m9	32	35	320	54	12	30214	30391	31831	5.4%
f2126_m1	8	356	529	1000	334	3034084	3038769	3181377	4.9%
f2126_m2	16	85	139	319	422	2276478	2282395	2389531	5.0%
f2126_m3	1	30	20	452	103	53351	55729	58369	9.4%
g1023_m1	14	139	273	43	134	154712	156597	163989	6.0%
g1023_m2	2	221	215	84	74	45898	47609	49865	8.6%
g1023_m4	4	145	155	54	268	141474	145235	152051	7.5%
g1023_m10	1	301	377	13	29	22858	23273	24377	6.6%
h953_m1	4	112	152	348	341	579952	584735	612191	5.6%
h953_m2	2	68	89	327	9	8278	8413	8845	6.8%
h953_m8	8	35	69	189	305	504832	509111	533015	5.6%
p34392_m1	1	15	94	806	210	209576	212525	222509	6.2%
p93791_m12	46	361	80	93	391	1977986	1986988	2080252	5.2%
p93791_m19	44	538	437	100	210	1164676	1169515	1224427	5.1%
t512505_m7	1	122	36	514	608	462230	476223	498591	7.9%
t512505_m8	3	106	147	1473	1025	4835456	4849815	5077383	5.0%
t512505_m9	1	82	122	530	195	151624	154363	161611	6.6%
t512505_m14	1	751	381	1225	278	761111	767514	803562	5.6%
t512505_m15	1	406	132	386	151	182247	185729	194465	6.7%
t512505_m16	1	850	897	154	370	722614	727803	761979	5.4%
t512505_m23	2	99	124	1372	532	1594686	1602143	1677359	5.2%
t512505_m24	1	182	129	1669	429	874619	884495	926015	5.9%
t512505_m31	28	211	316	1550	3370	148431662	148478851	155445619	4.7%
u226_m7	20	97	64	54	76	99618	101375	106175	6.6%

In addition to the expected NIMA test time, Table 4-2 reports the percentage overhead of NIMA test time when compared to lower bound conventional external test

architecture. Using lower bound values for conventional ATE-based testing methods compares NIMA to many other test strategies reported in the literature, such as [85][86][88][90], as these works propose different methods of achieving the lowest possible test time in ATE-based testing. From Table 4-2 and assuming no control lines are required for the conventional serial connection, the increase in test time with NIMA is always less than 10% when compared to a lower bound serial connection in a conventional external tester approach. Moreover, the results reported in Table 4-2 are independent of the result reported in [9] that reusing an NoC allows comparable test times with techniques that use exclusive TAMs. As shown in [9], with the assumption of having an NoC, the large number of chip's functional I/O pins can be used for post-fabrication test, resulting in low test time with no extra test pins.

4.8 Summary and Conclusions

In the current test architectures, no distinction is made between the *communication* and the *processing* of test data. In addition, an underlying assumption is the existence of a physical link that enables a single tester to directly control a core's DFT elements. While this assumption generally results in simple test protocols, this chapter identified a number of problems in terms of reduced modularity and flexibility, characteristic of such a direct-access template.

As one solution, this chapter proposed the concept of *Network-oriented, Indirect and Modular Architecture* (NIMA) for testing core-based SoCs. In NIMA, different testers can connect to a common switching fabric and send test data to the core(s) under test.

The indirect methodology of NIMA breaks the coupling between the core, the TAM, and the tester. That is, NIMA de-couples test-data *processing* and *communication*. In doing so, NIMA alleviates the problems associated with tightly coupled test architectures, and facilitates test resources partitioning for lowering test cost.

NIMA is developed such that test stimuli and expected results for cores are first compiled into new formats and then encapsulated into packets. These packets are subsequently augmented with control and address bits such that they are autonomously transmitted to their destination through a switching fabric. In this way, NIMA makes a better utilisation of the available resources, and eliminates the need for control lines in test architectures. Owing to the indirect nature of the connection in NIMA, embedded autonomous blocks at each core are used for applying the test pattern to a core and comparing the test results with the expected results. The autonomous test data communication, application, and comparison in NIMA facilitate automation of the entire test cycle resulting in lowering test cost.

This chapter also presented an implementation of NIMA on a simple SoC to validate its underlying concept. For each layer of NIMA's architecture, it provided the detailed design parameters used in the implementation. Finally, experimental results for a simple SoC and ITC'02 Benchmark Circuits were presented. It was observed that, in general, switches in NIMA require an area equal to about 1250 2-input NAND gates, and that, per switch, the *Application Layer* in NIMA adds an area equivalent to about 600-800 2-input NAND gates. In addition, the results predict that test time in NIMA increases less than

10% per chip, when compared to a lower bound in conventional test architectures. However, as the control mechanism is integrated in NIMA's packets, NIMA eliminates the need for control lines and, hence, requires fewer test pins when compared to current test architectures. Hence, assuming the same number of test pins for NIMA and the current test architectures, the test time in NIMA will be lower than the test time in other test architectures. In addition, multi-site testing is ideal in NIMA when DAST is used in the application layer. Hence, the test time is significantly reduced.

Chapter 5
Conclusions

5.1 Summary

Testing *Very Large Scale Integration* (VLSI) circuits can be an expensive and difficult process. *Design for testability* (DFT) techniques that facilitate testing of VLSI chips and reduce test cost are essential. With the paradigm shift towards core-based system design, new challenges arise. These new challenges call for new DFT methodologies.

The common practice of relying solely on external *automatic test equipment* (ATE) resources is insufficient for the billion-transistor core-based systems. Moreover, typical logic *Built-in Self-Test* (BIST) techniques use pseudo-random patterns. However, one major challenge with using pseudo-random patterns is the potentially lower fault coverage, compared to that resulting from deterministic patterns obtained via *automatic test pattern generation* (ATPG). In addition, BIST test time is generally two to three

times longer than that of ATPG-based testing. This longer test time is the result of the use of pseudo-random patterns and the compaction of the test responses in BIST. Additionally, compacting the test responses and using a signature analyzer at the output of the *circuit under test* (CUT), result in other drawbacks of BIST techniques, such as loss of diagnostic information, aliasing, and "zero-ing out". Therefore, embedded testers in the form of deterministic *infrastructure intellectual property* (I-IP) that take over some functionality of external ATEs are increasingly deemed essential. These I-IPs facilitate post fabrication testing, but do not add to the functionality of the chip. In other words, embedded deterministic I-IP is an enabling technology that facilitates ATPG-grade, in-field, and at-speed testing and, hence, improves the quality of test.

To leverage the benefits of embedded testing without the potential shortcomings of *Built-in Self-Test* (BIST) techniques, this thesis described a novel embedded test architecture with test resources that requires the communication of ATPG test data through on-chip global interconnects. This novel technique is referred to as *Dedicated Autonomous Scan-based Testing* (DAST). DAST is an improvement over BIST in two major ways. Firstly, it improves test time and test coverage with minimal test data, as it relies on ATPG test data and avoids compacting the test results. Secondly, DAST results in no loss of diagnostic information.

In addition, scan test sequence generation follows a simple protocol. A scan test sequence generation procedure was shown to constitute of two distinct parts: one of test *data delivery* and the other of test *data control and observation*. The test *data control*

and observation, i.e., the generation of the specific scan vector sequencing tends to use much of ATE resources. Therefore, DAST reduces test cost because it partitions the test resources and ports expensive test data control and observation responses from the ATE onto the test chip.

The implementation of DAST requires introducing a level of hierarchy between the ATPG and the embedded core under test. In DAST, a simple compilation of the raw test data is performed off-chip to append a few op-codes to the test stimuli and expected outputs. The data and associated op-codes are then transferred onto the chip and interpreted by an *Embedded Autonomous Sequencer* (EAS) and an *Embedded Autonomous Results Analyzer* (EARA) local to each core.

This thesis validated the DAST concept and, through physical implementation, showed that with an area overhead equivalent to about 600-800 2-input NAND gates, DAST enables embedded deterministic test of SoC chips. It was shown that, per chip, the test time in DAST is marginally longer than the test time for comparable ATE-based testing. However, DAST enables an ideal multi-site testing strategy. In this ideal multi-site testing strategy, theoretically, an unlimited number of chips can be tested concurrently on a single tester. This is feasible, as test responses in DAST are sent alongside the test stimuli, enabling these data to be sent to similar chips concurrently. Therefore, effectively, based on the total number of similar chips tested, test time is reduced significantly using the DAST methodology when compared to typical ATE-based testing.

In addition, this thesis categorised commonly-used test architectures into three groups based on the connection scheme between the chip pins and the core terminals. It was observed that, in general, the current common test practices assume the existence of a physical link that enables a single tester to directly control a core's DFT elements. While this assumption typically results in simple test protocols, a number of problems with such a direct-access template were identified. These problems include: reduced flexibility; the need for close proximity between the CUT and the tester in order to maintain timing requirements; impracticality or high cost of in-field or remote-access testing; and the impracticality of using multiple-testers to test an SoC. These problems are in addition to the reduced modularity in the generic test architecture. However, in order to address many new challenges, SoC test architectures require a more structured, systematic, modular, and hierarchical approach than the traditional DFT. Furthermore, the International Technology Roadmap predicts that the system complexity in chips using 65nm technology and beyond necessitate a focus on communication rather than computation. Based on this prediction, this thesis proposes that future chips include a switching fabric coupled with network-oriented protocols for connecting cores together. This thesis also proposes that new methodologies and test architectures, which allow reuse of this switching fabric for test purposes, will likely be needed to lower the test cost.

The issues outlined above motivated the development of the concept of *Network-oriented, Indirect and Modular Architecture* (NIMA) for testing core-based SoCs. In this

scheme, different testers can connect to a common switching fabric and send test data to the core(s) under test. The indirect methodology of NIMA decouples the core, the *test access mechanism* (TAM), and the tester. That is, NIMA handles test-data *processing* and *communication* separately. In doing so, NIMA alleviates the previously outlined problems associated with tightly coupled test architectures. Furthermore, the de-coupling of test-data *processing* and *communication* facilitates partitioning of external test resources into new embedded blocks and external low-cost and low-functionality components. This partitioning lowers tester cost and enhances the test productivity through the introduction of hierarchical, modular, and flexible test architecture.

NIMA is developed such that test stimuli and expected results for cores are first compiled into new formats and then encapsulated into packets. These packets are subsequently augmented with control and address bits such that they are autonomously transmitted to their destination through a switching fabric. Owing to the indirect nature of the connection, embedded autonomous blocks at each core are used for applying the test pattern to a core and comparing the test results with the expected results. In this thesis, the NIMA test architecture is considered as a special communication network consisting of hardware and software that allow the transportation of the test stimuli and expected results from multiple sources to multiple cores. It is predicted that regarding NIMA as a special communication network helps in using the accumulative knowledge in different communication networks. As one example, in this thesis, a 3-layered model

consisting of *Physical Layer*, *Network Layer*, and *Application Layer* for NIMA was developed.

This thesis presented an implementation of NIMA on a simple core-based design to validate its underlying concept. For each layer of NIMA's architecture, the detailed design parameters used in the implementation were presented. Finally, experimental results for a simple design and ITC'02 Benchmark Circuits were provided. It was observed that, in general, switches in NIMA require an area equal to about 1250 2-input NAND gates, and that the *Application Layer* in NIMA adds an area equivalent to about 600-800 2-input NAND gates. In addition, the results predict that test time in NIMA increases by a maximum of about 10% when compared to a lower bound test time in conventional test architectures. This increase in test time was predicted under the assumption that comparable conventional test architectures do not need or use any control TAM lines.

In summary, NIMA provides the basis for embedded, cost-effective, scalable, modular, and flexible test design and programming with small area overhead. NIMA, by integrating the control mechanism in the packets, eliminates the need for control lines and, hence, requires fewer test pins when compared to current test architectures. Therefore, assuming the same number of test pins for NIMA and the current test architectures, the test time in NIMA will be lower than the test time in other test architectures. Furthermore, NIMA facilitates the remote-test of an SoC by single or multiple testers. NIMA also enables the transmission of the test data to an SoC deployed

in the field when it is desired to test and monitor a chip in its target system. Finally, and equally important, NIMA serves in contributing towards the development of new test architectures that benefit from the reuse of a *network-on-chip* (NoC) interconnect template.

5.2 Future Work

The concepts of DAST and NIMA for testing core-based SoC designs were validated in this thesis. The implementations of these methodologies were given for a serial connection to off-chip resources. An interesting future research work would be the implementation and characterization of DAST and NIMA with multi-bit communication channels. Of particular interest in this case is the possibility and effect of a dynamically changing channel width to reduce idle time and, hence, lower test time.

Future work should also investigate other switching techniques in the *Network Layer* of NIMA. Investigation needs to focus on switches that can be used in an NoC environment where cores connect to each other through these switches. Further research work for improvement of the *Network Layer* design and implementation can investigate redundancy in routing and out-of-order packet delivery. Out-of-order delivery can be handled in two ways in the *Application Layer*. In the first option, the packets will be put back into their original order. This option could result in potentially large buffering mechanism that may not be acceptable. In the second option, the packets are used as they are received out-of-order. Investigations need to determine the impact of this latter scheme on the fault coverage and operation of the embedded blocks.

NIMA was developed with the assumption of a single frequency for the cores and the test architecture. Investigation is required to develop and characterize a more realistic case of having several clock-domains within the cores and throughout the chip. In addition, scheduling techniques that can handle a multiple clock-domain regime are also needed. Moreover, NIMA presents the opportunity to be easily integrated into a *globally asynchronous and locally synchronous* (GALS) methodology with significant benefits. Hence, more research work is required to develop NIMA to work in a chip based on GALS concept.

Finally, investigation into drastically different techniques, such as wireless connections, in the *Physical Layer* and their impact on the overall performance of SoC testing constitute other envisaged future work.

Bibliography

[1] The International Technology Roadmap for Semiconductors, 2003 Edition, http://public.itrs.net.

[2] M.L. Bushnell, and V.D. Agrawal, "Essentials of Electronic Testing, for Digital, Memory & Mixed-signal VLSI Circuits", *Kluwer Academic Publishers*, 2000, ISBN 0-7923-799-1-8.

[3] M. Abramovici, M.A. Breuer, and A.D. Friedman, "Digital Systems Testing and Testable Design", *Computer Science Press*, 1990.

[4] The International Technology Roadmap for Semiconductors, 2001 Edition, http://public.itrc.itrs.net.

[5] P.G. and A. Greiner, "A Generic Architecture for On-Chip Packet-Switched Interconnections", *Proc. Design, Automation and Test in Europe*, 2000, pp. 250-256.

[6] W.J. Dally and B. Towless, "Route Packets, Not Wires: On-Chip Interconnection Networks", *Proc. Design Automation Conference*, 2001, pp. 684-689.

[7] L. Benini and G. De Micheli, "Networks on Chips: A New SoC Paradigm", *IEEE Computers,* January 2002, pp. 70-78.

[8] P.P. Pande, C. Grecu, A. Ivanov, and R. Saleh, "Design of a Switch for Network on Chip Applications", *Proc. IEEE International Symposium on Circuits and Systems*, 2003, pp. 217-220.

[9] E. Cota, *et al.*, "The Impact of NoC Reuse on the Testing of Core-Based Systems", *Proc. IEEE VLSI Test Symposium*, 2003, pp. 128-133.

[10] E. Cota, L. Carro, F. Wagner, and M. Lubaszewski, "Power-aware NoC Reuse on the Testing of Core-based Systems", *Proc. IEEE International Test Conference*, 2003, pp. 612-621.

[11] E.J. Marinissen, Y. Zorian, R. Kapur, T. Taylor, and L. Whetsel, "Towards a Standard for Embedded Core Test: An Example", *Proc. IEEE International Test Conference*, 1999, pp. 616-627.

[12] M. Bernbaum and H. Sachs, "How VSIA Answers the SoC Dilemma", *IEEE Computers*, June 1999, pp. 42-49.

[13] E.J. Marinissen and Y. Zorian, "Challenges in Testing Core-Based System ICs", *IEEE Communication Magazine*, June 1999, pp. 104 –109.

[14] D. Chin, "Executing System on a Chip: Requirements for a Successful SoC Implementation", *IEEE Electron Devices Meeting*, 1998, IEDM98-3 – 8.

[15] The International Technology Roadmap for Semiconductors, 1999 Edition, http://public.itrs.net.

[16] A. Benso *et al.*, "HD2BIST: Architectural Framework for BIST Scheduling, Data Patterns delivering & Diagnosis in SoCs", *Proc. IEEE International Test Conference, 2000*, pp. 892 –901.

[17] D. Gajski *et al.*, "Essential Issues for IP Reuse", *Proc. IEEE ASP-DAC*, 2000, pp. 37-42.

[18] Y. Zorian, "Test Requirements for Embedded Core-Based Systems an IEEE P1500", *Proc. IEEE International Test Conference*, 1997, pp. 191-199.

[19] R.K. Gupta and Y. Zorian, "Introducing Core-Based System Design", *IEEE Design & Test of Computers*, December 1997 14(4), pp. 15-25.

[20] W. Ke and K. Truong, "Design With Testability for a Platform-Based SoC Design Methodology", *Proc. IEEE AP-ASIC*, 1999 pp. 307-310.

[21] Y. Zorian, E.J. Marinissen, and S. Dey, "Testing Embedded-Core Based System Chips", *Proc. IEEE International Test Conference*, 1998, pp. 130-143.

[22] E.J. Marinissen *et al.*, "A structured & Scalable Mechanism for Test Access to Embedded Reusable Cores", *Proc. IEEE International Test Conference,* 1998, pp. 284-293.

[23] Y. Zorian, "Testing the Monster Chip", *IEEE Spectrum*, July 1999, pp. 22-24.

[24] Y. Zorian, "System-Chip Test Strategies", *Proc. Design Automation Conference*, 1998, pp. 752-757.

[25] R. Chandramouli and S. Pateras, "Testing Systems on a Chip", *IEEE Spectrum*, Nov. 1996, pp. 42-47.

[26] L. Whetsel, "Core Test Connectivity Communication & Control", *Proc. IEEE International Test Conference*, 1998, pp. 303-312.

[27] R. Kapur, R. Chandramouli, and T.W. Williams, "Strategies for Low-Cost Test", *IEEE Design & Test of Computers*, November-December 2001, pp. 47-54.

[28] J. Bedsole, R. Raina, Al Crouch, and M.S. Abadir, "Very Low Cost Testers: Opportunities and Challenges", *IEEE Design & Test of Computers,* September-October 2001, pp. 60-69.

[29] J. Rajski, "DFT for High-Quality Low Cost Manufacturing Test", *IEEE Asian Test Symposium,* 2001, pp. 3-8.

[30] D&T Roundtable, "Test Resource Partitioning", *IEEE Design & Test of Computers*, July-September 2000, pp. 126-132.

[31] S. Pateras, "Achieving At-Speed Structural Test", *IEEE Design & Test of Computers*, September-October 2003, pp. 26-33.

[32] Y. Zorian, "Embedded Memory Test & Repair: Infrastructure IP for SOC Yield", *Proc. IEEE International Test Conference*, 2002, pp. 340-349.

[33] P. Varma and S. Bhatia, " A Structured Test Re-Use Methodology for Core-Based System Chips", *Proc. IEEE International Test Conference*, 1998, pp. 294-302.

[34] P. Harrod, "Testing Reusable IP- A Case Study", *Proc. IEEE International Test Conference*, 1999, pp. 493-498.

[35] E.H. Volkernik, A. Khoche, J. Rivoir, and K.D. Hilliges, "Modern Test Techniques: Tradeoffs, Synergies, and Scalable Benefits", *Journal of Electronic Testing: Theory and Applications*, vol. 19, 2003, pp. 125-135.

[36] A. Chandra and K. Chakrabarty, "Test Resource Partitioning for SOCs", *IEEE Design & Test of Computers,* September-October 2001, pp. 80-91.

[37] K. Arabi, "Logic BIST and Scan Test Techniques for Multiple Identical Blocks", *Proc. IEEE VLSI Test Symposium*, 2002, pp. 60-65.

[38] D. Das and N.A. Touba, "Reducing Test Data Volume Using External/LBIST Hybrid Test Patterns", *Proc. IEEE International Test Conference*, 2000, pp. 115-122.

[39] V. Iyengar, K. Chakrabarty, and B.T. Murray, "Deterministic Built-in Pattern Generation for Sequential Circuits", *Journal of Electronic Testing: Theory and Application*, vol. 15, August-October 1999, pp. 97-115.

[40] A. Chandra and K. Chakrabarty, "System-on-a-Chip Test-Data Compression and Decompression Architectures Based on Golomb Codes", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, No. 3, March 2001, pp. 355-368.

[41] H.G. Liang, S. Hellebrand, and H.J. Wunderlich, "Two-Dimensional Test Data Compression for Scan-Based Deterministic BIST", *Proc. IEEE International Test Conference*, 2001, pp. 894-902.

[42] I. Bayaraktaroglu and A. Orailglu, "Test Volume and Application Time Reduction", *Proc. Design Automation Conference*, 2001, pp. 151-155.

[43] P.T. Gonciari, B.M. Al-Hashimi, and N. Nicolici, "Improving Compression Ratio, Area Overhead, and Test Application Time for System-on-a-Chip Test Data Compression/Decompression", *Proc. Design, Automation and Test in Europe*, 2002, pp. 604-611.

[44] J. Rajski, M. Kassab, N. Mukherjee, and N. Tamarapalli, "Embedded Deterministic Test for Low-Cost Manufacturing", *IEEE Design & Test of Computers,* September-October 2003, pp. 58-66.

[45] J. Rajski *et al.,* "Embedded Deterministic Test for Low Cost Manufacturing Test", *Proc. IEEE International Test Conference*, 2002, pp. 301-310.

[46] T. Hiraide *et al.*, "BIST-Aided Scan Test – A New Method for Test Cost Reduction", *Proc. IEEE VLSI Test Symposium,* 2003, pp. 359-364.

[47] B. Nadeau-Dostie, "Design For At-speed Test, Diagnosis, and Measurement", *Kluwer Academic Publishers*, 2000.

[48] Y. Zorian, "What is Infrastructure IP?", *IEEE Design & Test of Computers*, May-June 2002, pp. 5-7.

[49] Y. Zorian, "Embedded Infrastructure IP for SOC Yield Improvement", *Proc. Design Automation Conference*, 2002, pp. 709-712.

[50] Y. Zorian, "Guest Editor's Introduction: What Is Infrastructure IP?", *IEEE Design & Test of Computers*, Infrastructure IP Special Issue, vol. 19, Issue 3 , May-June 2002, pp. 3-5.

[51] IEEE P1500 Standard for Embedded Core Test (SECT), http://grouper.ieee.org/groups/1500.

[52] E.J. McCluskey, "Built-In Self-Test Techniques", *IEEE Design & Test of Computers*, vol. 2, No. 2, Apr. 1985, pp. 21-28.

[53] P.H. Bardell, W.H. McAnney, and J. Savir, "Built-In Test for VLSI", *John Wiley & Sons Inc.*, 1987, ISBN: 0-471-62463-2.

[54] V.D. Agrawal, C.R. Kime, and K.K. Saluja, "A Tutorial on Built-in Self-Test, Part 1: Principles", *IEEE Design & Test of Computers,* March 1993, pp. 73-82.

[55] V.D. Agrawal, C.R. Kime, and K.K. Saluja, "A Tutorial on Built-in Self-Test, Part 2: Applications", *IEEE Design & Test of Computers,* June 1993, pp. 69-77.

[56] J. Rajski, N. Tamarapalli, and J. Tyszer, "Automated Synthesis of Phase Shifters for Built-In-Self-Test Application", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems,* vol. 19, No. 10. October 2000, pp. 1175-1188.

[57] N.A. Touba and E.J. McCluskey, "Test Point Insertion based on Path Tracing", *Proc. IEEE VLSI Test Symposium*, 1996, pp. 2-8.

[58] N. Tamarapalli and J. Rajski, "Constructive Multi-Phase Test Point Insertion for Scan-Based BIST", *Proc. IEEE International Test Conference*, 1996, pp. 649-658.

[59] F. Muradali, V.K. Agrawal, and B. Nadeau-Dostie, "A New Procedure for Weighted Random Built-In-Self-Test", *Proc. IEEE International Test Conference*, 1990, pp. 660-668.

[60] S. Venkataraman, J. Rajski, S. Hellebrand, and S. Tarnick, "An Efficient BIST Scheme Based on Reseeding of Multiple Polynomial Linear Feedback Shift Registers", *Proc. International Conference on Computer-Aided Design*, 1993, pp. 572-577.

[61] K. Chakrabarty, B. T. Murray, and V. Iyengar, "Deterministic Built-in Test Pattern Generation for High-Performance Circuits Using Twisted-Ring Counters", *IEEE Trans. on VLSI Systems*, vol. 8, No. 5, October 2000, pp. 633-636.

[62] G. Hetherigton *et al.*, "Logic BIST for Large Industrial Designs: Real Issues and Case Studies", *Proc. IEEE International Test conference*, 1999, pp. 358-367.

[63] R.W. Bassett *et al.*, "Low-Cost Testing of High-Density Logic Components", *IEEE Design & Test of Computers*, 1990, pp. 15-28.

[64] H.J. Wunderlich and G. Kiefer, "Bit-Flipping BIST", *Proc. International Conference on Computer Aided-Design,* 1996, pp. 337-343.

[65] N.A. Touba and E.J. McCluskey, "Altering a Pseudo-Random Bit Sequence for Scan Based BIST", *Proc. IEEE International Test Conference,* 1996, pp. 167-175.

[66] A. Jas, C.V. Krishna, and N.A. Touba, "Hybrid BIST Based on Weighted Pseudo-Random Testing: A New Test Resource Partitioning Scheme", *Proc. IEEE VLSI Test Symposium*, 2001, pp. 2-8.

[67] G. Kiefer, H. Vranken, E.J. Marinissen, and H.J. Wunderlich, "Application of Deterministic Logic BIST on Industrial Circuits", *Proc. IEEE International Test Conference*, 2000, pp. 105-114.

[68] Y. Wu and A. Ivanov, "Single-Reference Multiple Intermediate Signature (SREMIS) Analysis for BIST", *IEEE Trans. on Computers*, vol. 44, Issue 6, June 1995, pp. 817-825.

[69] Y. Wu and A. Ivanov, "Minimal Hardware Multiple Signature Analysis for BIST", *Proc. IEEE VLSI Test Symposium*, 1993, pp. 17-20.

[70] S. Wang and S.K. Gupta, "DS-LFSR: a BIST TPG for Low Switching Activity", *Proc. IEEE Trans.*, vol. 21, Issue 7, July 2002, pp. 842-851.

[71] S. Gerstendorfer and H.J. Wunderlich, "Minimized Power Consumption for Scan-Based BIST", *Proc. IEEE International Test Conference*, 1999, pp. 77-84.

[72] I. Ghosh, S. Dey, and N.K. Jha, "A Fast & Low Cost Testing Technique for Core-Based System-on-Chip", *Proc. Design Automation Conference*, 1998, pp. 542-547.

[73] I. Ghosh, N.K. Jha, and S. Dey, "A Low Overhead Design for Testability & Test Generation Technique for Core-Based Systems", *Proc. IEEE International Test Conference*, 1999, pp. 50-59.

[74] M. Nourani and C. Papachristou, "An ILP Formulation to Optimize Test Access Mechanism in System-on-Chip Testing", *Proc. IEEE International Test Conference*, 2000, pp. 902-1000.

[75] B. Mathewson, "Core Provider's Test Experience", *Presentation at IEEEP1500 Working Group Meeting*, June 1998, CA. U.S.A., http://grouper.ieee.org/groups/1500/pastmeetings.html#dac98.

[76] V. Immaneni and S. Raman, "Direct Access Test Scheme-Design of Block and Core Cells for Embedded ASICS", *Proc. IEEE International Test Conference*, 1990, pp. 488-492.

[77] N.A. Touba and B. Pouya, "Testing Embedded Cores using Partial Isolation Rings", *Proc. IEEE VLSI Test Symposium*, 1997, pp. 10-16.

[78] B. Pouya and N.A. Touba, "Modifying User-Defined Logic for Test Access to Embedded Cores", *Proc. IEEE International Test Conference*, 1997, pp. 60-68.

[79] L. Whetsel, "An IEEE 1149.1 Based Test Access Architecture for ICs with Embedded Cores", *Proc. IEEE International Test Conference*, 1997, pp. 69-78.

[80] D. Bhattacharya, "Hierarchical Test Access Architecture for Embedded Cores in an Integrated Circuit", *Proc. IEEE VLSI Test Symposium*, 1998, pp. 8-14.

[81] A. Benso *et al.*, "HD2BIST: Architectural Framework for BIST Scheduling, Data Patterns delivering & Diagnosis in SoCs", *Proc. IEEE International Test Conference*, 2000, pp. 892 –901.

[82] M. Benabdenbi and W. Maroufi, "CAS-Bus: A Scalable and Reconfigurable Test Access Mechanism for Systems on a Chip", *Proc. Design, Automation and Test in Europe*, 2000, pp. 141-145.

[83] L. Whetsel, "Addressable Test Ports, an Approach to Testing Embedded Cores", *Proc. IEEE International Test Conference*, 1999, pp. 1055-1064.

[84] Z.S. Ebadi and A. Ivanov, "Time Domain Multiplexed TAM: Implementation and Comparison", *Proc. Design, Automation and Test in Europe*, 2003, pp. 732-737.

[85] V. Iyengar and K. Chakrabarty, "System-on-a-Chip Test Scheduling With Precedence Relationships, Pre-emption, and Power Constraints", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, No. 9, Sept. 2002, pp. 1088-1094.

[86] E. Larsson, and Z. Peng, "An Integrated System-On-Chip Test Framework", *Proc. Design, Automation, and Test in Europe*, 2001, pp. 139-144.

[87] Y. Huang *et al.*, "Optimal Core Wrapper Width Selection and SOC Test Scheduling Based on 3-D Bin Packing Algorithm", *Proc. IEEE International Test Conference*, 2002, pp. 74-82.

[88] S.K. Goel and E.J. Marinissen, "Effective and Efficient Test Architecture Design for SoCs", *Proc. IEEE International Test Conference*, 2002, pp. 529 538.

[89] V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "Test Wrapper and Test Access Mechanism Co-Optimization for System-on-Chip", *Journal of ElectronicTesting: Theory and Applications*, vol. 18, 2002, pp. 211-230.

[90] W. Zou, S.M. Reddy, I. Pomeranz, Y. Huang, "SoC Test Scheduling Using Simulated Annealing", *Proc. IEEE VLSI Test Symposium*, 2003, pp. 325-331.

[91] M. Nahvi, A. Ivanov, and R. Saleh, "Dedicated Autonomous Scan-Based Testing (DAST) for Embedded Cores", *Proc. IEEE International Test Conference*, 2002, pp. 1176-1183.

[92] M. Nahvi and A. Ivanov, "An Embedded Autonomous Scan-Based Results Analyzer (EARA) for SoC Cores", *Proc. IEEE VLSI Test Symposium*, 2003, pp. 293-298.

[93] ITC99 Benchmarks: http://www.cerc.utexas.edu/itc99-benchmarks/bench.html.

[94] M. Nahvi and A. Ivanov, "A Packet Switching Communication-Based Test Access Mechanism for System Chips", *Proc. IEEE European Test Workshop*, 2001, pp. 81-86.

[95] M. Nahvi and A. Ivanov, "Indirect Test Architecture for SoC Testing", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems,* vol. 23, Issue 7, July 2004, pp. 1128-1142.

[96] W. Stallings, "Data & Computer Communications – Sixth Edition", *Prentice-Hall Inc.*, ISBN 0-13-084370-9, 2000.

[97] ITC'02 Benchmarks: http://www.extra.research.philips.com/itc02socbenchm.

[98] R. M. Chou, K. K. Saluja, and V. D. Agrawal, "Scheduling Tests for VLSI Systems Under Power Constraints", *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 5, No. 2, June 1997, pp. 175-185.

Appendix A:
DAST Test Time Models

This appendix derives Equations (3-1), (3-2), and (3-3):

$$T_{M1\_DAST} = (9 + 2PI + SI \times SE) + TP(23 + 2PI + SI + SI \times SE) \quad \text{(A-1)}$$

iff $PO < PI+3$

$$T_{M2\_DAST} = (9 + 2PI + SI \times SE) + TP(20 + PI + PO + SI + SI \times SE) \quad \text{(A-2)}$$

iff $PI+SI+6 > PO \geq PI+3$

$$T_{M3\_DAST} = (9 + 2PI + SI \times SE) + TP(14 + 2PO + SI \times SE) \quad \text{(A-3)}$$

iff $PO \geq PI+SI+6$

where *PI*, *SI*, and *PO* are the numbers of primary inputs, scan inputs, and primary outputs as defined in Section 3.2. Moreover, *TP* is the total number of test patterns, *SE* is the maximum number of scan cells in the scan chain(s), and T_{M1_DAST}, T_{M2_DAST}, and T_{M3_DAST} are the test time models for DAST, in terms of clock cycles.

Given the algorithms for the EAS and EARA compilers given in Figure 3-5 and Figure 3-6, careful investigations of typical test data sets reveal that, for a typical test pattern, EAS blocks receive D_{EAS_TP} bits of data:

$$D_{EAS\_TP} = 2(D_{PI}) + D_{SI} + 3(D_{Assert-Clk}) + D_{SC} + D_{Sync} \quad \text{(A-4)}$$

where D_{PI}, D_{SI}, D_{SC} are the intended data bits for the primary inputs, the scan inputs, and the scan chain(s), respectively. Moreover, $D_{Assert\text{-}Clk}$ represents the data bits of the *Assert-Clk* op-code, as used in the EAS compiler of Section 3.4.1. Finally, D_{Sync} denotes the number of bits needed for synchronizing DAST's EAS and EARA blocks.

Given the typical generic scan test waveforms of Figure 3-2, and based on lines 24 and 28 in the algorithm of Figure 3-5 and lines 34, 37 and 49 in the algorithm of Figure 3-6, the number of bits for synchronization, i.e., D_{Sync} can be obtained from:

$$D_{Sync} = ((D_{PO} - (D_{PI} + D_{SI})) + 1) + ((D_{PO} - D_{PI}) + 1) \quad \text{(A-5)}$$

where D_{PO} denotes the expected test data bits for the primary outputs. Note that D_{Sync} comprises of two terms and that each of these terms is always ≥1, as the number of the synchronization bits is always a positive integer.

As explained in Section 3.4.1, three bits are used for the *Assert-Clk* op-code, and both the EAS and the EARA compilers add three-bit op-codes to the beginning of each section of the test program. Therefore, the blocks of data for *PI*, *SI*, and the scan chains have three op-codes added to them by the EAS compiler. Hence, $D_{Assert\text{-}Clk}$=3, D_{PI}=*PI*+3, D_{SI}=*SI*+3, and D_{SC}=*SC*+3 (note that $SC = SI \times SE$). However, D_{PO} is simply equal to the

number of primary outputs, i.e., *PO*, as the synchronization is only needed for the number of primary outputs.

Replacing Equation (A-5) in Equation (A-4), and substituting for known values yields:

$$D_{EAS\_TP} = 2(PI+3)+(SI+3)+3(3)+(3+SI\times SE)+((PO-(PI+SI+6))+1)+((PO-(PI+3))+1) \quad \text{(A-6)}$$

If one clock cycle is needed for each bit in the data blocks, the test time in DAST is equal to the total number of bits received by the EAS block. Given that the EAS block receives initialization data for the primary inputs and the scan chain(s), before receiving the entire test pattern, DAST test time can be modeled to be:

$$T_{M\_DAST} = 2(D_{PI})+D_{SC}+TP(D_{EAS\_TP}) \quad \text{(A-7)}$$

where the first two terms represent the initialization bits and the last term denotes the total test pattern bits.

The last two terms of Equation (A-6) are conditional terms and, as explained for Equation (A-5), are always ≥1. Hence, to simplify Equation (A-6), three conditions can be identified as follows: $PO<PI+3$; $PI+SI+6>PO\geq PI+3$; and $PO\geq PI+SI+6$. Using these conditions, replacing Equation (A-6) in Equation (A-7), and after simplification, Equation (A-7) yields the DAST test times models of:

$$T_{M1\_DAST} = (9+2PI+SI\times SE)+TP(23+2PI+SI+SI\times SE) \quad \text{(A-8)}$$

iff $PO < PI+3$

$$T_{M2\_DAST} = (9 + 2PI + SI \times SE) + TP(20 + PI + PO + SI + SI \times SE) \quad \text{(A-9)}$$

iff $PI+SI+6 > PO \geq PI+3$

$$T_{M3\_DAST} = (9 + 2PI + SI \times SE) + TP(14 + 2PO + SI \times SE) \quad \text{(A-10)}$$

iff $PO \geq PI+SI+6$

Appendix B: Theoretical Test Time Models for Serial ATE-based Testing

This appendix derives Equations (3-4), (3-5), and (3-6):

$$T_{S1} = (2PI + SI \times SE) + TP(2PI + SI + SI \times SE) \quad \text{(B-1)}$$

iff $PO < PI$

$$T_{S2} = (2PI + SI \times SE) + TP(PI + PO + SI + SI \times SE) \quad \text{(B-2)}$$

iff $PI+SI > PO \geq PI$

$$T_{S3} = (2PI + SI \times SE) + TP(2PO + SI \times SE) \quad \text{(B-3)}$$

iff $PO \geq PI+SI$

where *PI*, *SI*, and *PO* are the numbers of primary inputs, scan inputs, and primary outputs as defined in Section 3.2. Moreover, *TP* is the total number of test patterns, *SE* is the maximum number of scan cells in the scan chain, and T_{S1}, T_{S2}, T_{S3} are the test time models of a serial ATE-based testing scheme, and are in clock cycles.

For lower bound analysis, assume that no time is required to set up the TAM and the wrapper of the CUT, and that one clock cycle is needed for each bit in the test data blocks. Careful investigations of typical generic scan test waveforms of Figure 3-2 and the test data sets reveals that, for a typical test pattern in a serial ATE-based testing scheme, the test time, i.e., T_{TP} is:

$$T_{TP} = 2(D_{PI}) + D_{SI} + D_{SC} + (D_{Po} - (D_{PI} + D_{SI})) + (D_{Po} - D_{PI}) \quad \text{(B-4)}$$

where D_{PI}, D_{SI}, D_{PO}, and D_{SC} are the intended data bits for the primary inputs, the scan inputs, primary outputs, and the scan chain(s), respectively. Moreover, the last two terms are always positive integer values. These two terms represent the differences between the number of test stimuli bits and the test response bits, and hence, denote the waiting time of the tester between finishing the application of the test stimuli and receiving all the test response bits.

Given that the CUT receives initialization data for the primary inputs and the scan chain(s) before receiving the entire test pattern, the test time for a serial ATE-based testing, i.e., T_S can be modeled to be:

$$T_S = 2(D_{PI}) + D_{SC} + TP(T_{TP}) \quad \text{(B-5)}$$

where the first two terms denote the initialization bits and the last term represents the total test pattern bits.

The lower bound values of D_{PI}, D_{SI}, D_{PO}, and D_{SC} are simply equal to *PI*, *SI*, *PO*, and $SC = SI \times SE$, respectively. Replacing these values and Equation (B-4) in Equation (B-5) yields:

$$T_S = (2PI + SI \times SE) + TP(2PI + SI + SI \times SE + (PO - (PI + SI)) + (PO - SI)) \quad \text{(B-6)}$$

The last two terms of Equation (B-4) are conditional that also appear in Equation (B-6). To simplify Equation (B-6), three conditions can be identified as follows: $PO<PI$; $PI+SI>PO\geq PI$; and $PO\geq PI+SI$. Using these conditions and after simplification, Equation (B-6) yields the test time models of a serial ATE-based testing scheme:

$$T_{S1} = (2PI + SI \times SE) + TP(2PI + SI + SI \times SE) \quad \text{(B-7)}$$

iff $PO < PI$

$$T_{S2} = (2PI + SI \times SE) + TP(PI + PO + SI + SI \times SE) \quad \text{(B-8)}$$

iff $PI+SI > PO \geq PI$

$$T_{S3} = (2PI + SI \times SE) + TP(2PO + SI \times SE) \quad \text{(B-9)}$$

iff $PO \geq PI+SI$