Chapter 1

Introduction

THE MONOLITHIC INTEGRATED circuit (IC) was reportedly conceptualized in 1952 by the British radar engineer Geoffrey Dummer. Jack S. Kilby of Texas Instruments is accredited for devising the first IC later in 1958. In 1961, Robert Noyce of Fairchild Semiconductor Corp. was awarded the first patent for development of an IC. This circuit contained one transistor, three resistors, and one capacitor. Ever since, there had been a tremendous growth in the semiconductor technology leading to diminishing transistor size. Historically, the trend of growth of microchip technology has been adhering to the Moore's Law [1], which states: 'microchips will gain a factor of two in speed every 1.5 years'. A contemporary IC can have multi million transistors packed into a single silicon substrate. This era of high density ICs is known as Very Large Scale Integration (VLSI).

Advancement in the semiconductor technology has led to decrease in the minimum feature size of the transistors, for which there has been several benefits. Some of the benefits are: increase in speed of operation of chips, reduction in system sizes, increased functionality within a single device, and reduction in the price. However, designing such highly dense circuits from scratch becomes a nuisance. At this time, the need for having modular designs and reuse of existing design modules were felt. The system-on-chip (SoC) technology evolved as a means to tackle the problem of integrating extensive functionalities onto a single silicon substrate, as per demand of the contemporary market. The characteristic which singularizes a SoC technology from that of an application-specific integrated-circuit (ASIC) technology (full chip is custom made) is inherent in the course undertaken dur-

ing the chip design flow. During design, reusable functional design components, called *cores*, either developed in-house or acquired from various core vendors, are assembled and systematized by the SoC integrator to finally produce a single SoC design [2].

1.1 Motivation

Re-usability of the cores hastens the SoC design process. However, not much relief to the process of testing is observed. Rather it procreates new challenges in testing [53]. The reusable cores are pre-designed and pre-verified modules, but are not tested. This is owing to the fact that testing is the process of validating the correct operation of the chip only after the chip has been manufactured. Since the cores are design components and not manufactured modules, testing is left up to the core users. As the complexity of the SoC increases so does the gate to pin count ratio of the chip, which, in effect, lowers the controllability and observability of the SoC internals from the SoC terminals. Thus test access is one major challenge in SoC testing. Another challenge is the huge test time. Since a SoC is a large circuit, the number of test vectors needed to test it is also large. As the size of the test set increases so does the test time. Moreover, if the test vector source and test response sink is an external device such as an Automatic Test Equipment (ATE) then the problem of test time is aggravated. The non-semiconductor connectivity between the ATE channels and chip test pins slows down the test process. Since chips are produced in millions, increase in the test time of a single chip leads to increase in the overall test time. However, the market demands quick production, that includes testing. Therefore, reduction in SoC test time is another major challenge in SoC testing, in order to conform to the market requirements. The test time can be effectively handled by increasing parallelism in testing. Parallelism can be brought about at the ATE end by increasing the number of chips that can be tested in parallel by a single ATE. The ATEs being very expensive devices, increasing concurrency by merely increasing the number of ATEs in a test environment is not practical. The cost of an ATE increases with the increase in the ATE pin count. Therefore, increasing the number of ATE pins in not an appropriate solution. Taking all these problems into account, an obvious way of achieving concurrency at the ATE is by reducing the test pin count of the chips to be tested by the ATE.

Once parallelism in testing at the ATE end is established, the next stride would be to reduce test time of each chip. One way of achieving this goal is by testing a number of cores within a single SoC concurrently by planning an appropriate schedule. Other than that, finding out the shortest test set that meets the desired constraints can also aid in reducing test time at the chip level. In sequential circuits, serial shift registers known as scan-chains, are popularly used to control and observe the internal states. The process of serially loading/unloading test data into the scan-chains is one of the major consumers of test time. If this test application time could be reduced by some means then considerable amount of test time would be saved. More concurrency at the ATE along with a reduced test time of each chip can bring about a substantial reduction in the overall test time of all the chips.

Even though the cores are pre-verified (i.e., the designs are validated) by the core vendors, it remains as a task of the core users to verify that the complete SoC, after integration of all the cores, meets the timing requirements in accordance with the SoC specification. One way of tackling this problem is by completely flattening the final SoC design and then running some standard commercially available static timing analysis (STA) tools. Another approach is by conserving the core level hierarchy inherent in SoC designs, through the use of efficient timing models or abstractions of the corresponding cores. The high complexity and the core diversities make timing verification of an entire flattened SoC design a tedious process. Modular timing verification using timing models of cores can help to speed-up the overall process. The objective of the entire thesis is a faster time-to-market of each chip. This is achieved by targeting the reduction in time to design test structures, reduction in test time, and reduction in time to perform timing verification of the full SoC design.

1.2 Contributions

In this thesis two broad aspects of the SoC development process, namely, testing and static timing verification, have been separately studied. One of the main objectives of this thesis is to reduce the cost of test for system-on-chips. This reduction in test cost is brought about by targeting the following aspects of testing:

- Reduction in test cost by reducing the time to design Design-For-Test (DFT) structures for a particular SoC. DFT is the portion of the chip that aids in transporting test data within the chip, and also sometimes is involved in generating test inputs and analyzing test responses.
- Reduction in test cost by reduction in test time. This is made possible in the following ways:
  - ✠ Increasing the number of chips to be tested concurrently by the ATE.
  - ✠ Reducing the test data volume for each chip.
  - ✠ Reducing the test application time.

One more objective of this thesis is to reduce the time taken to verify the complete SoC for achieving timing closure.

The main contributions of the thesis are summarized below:

- **The design and implementation of a reusable BIST core.** The complete testing of the SoC for stuck-at faults in logic cores and interconnects is done using a single centralized BIST structure, delivered in the form of a core. This centralized core has the following properties:
  - ✠ The core has a cellular automata (CA) based test-pattern-generator (TPG) and a CA based test-response-compactor (TRC). It also contains a test controller, and some registers to store seeds and CA rules.
  - ✠ The core is generalized and hence is re-usable.
  - ✠ The core can test itself.
  - ✠ The role of the two CA registers for test generation and test compaction are interchanged during logic core testing and interconnect testing. Therefore, no separate hardware is required for the two types of testing.

  The single core for full SoC testing has the following advantages:
  - ✠ By making the core re-usable considerable reduction in the time to design test structures is achieved.

- ✠ Reduction in test hardware compared to SoCs with each core having a dedicated BIST (with some increase in routing area). This is due to lesser number of TPGs and TRCs.
- ✠ Increased flexibility and easy integration of test structures to SoC.
- ✠ When logic cores come with dedicated BIST structures, the fault coverage as desired by the target process in which the reusable core is to be integrated might not match with that of the coverage as provided by the core vendor. If the coverage achievable by the BIST in core is lower than that desired after integration, then the SoC does not meet the aspired quality. If the coverage is higher, then the test cost such as performance, test time, area, or power may become higher [8]. Therefore, though it demands more effort from the SoC test designer to design BIST for cores, yet it is often helpful in terms of performance and quality.

- **A seed selection algorithm to find the best test set.** A new seed selection algorithm is presented. It tries to find the best seed from a given number of runs of a known PRPG. The best seed is one that produces the shortest sequence having the desired fault coverage. The algorithm has the following properties:
  - ✠ On running the algorithm for a complete PRPG cycle, the best single-seeded sequence of the PRPG may be obtained.
  - ✠ The algorithm to find the best seed proceeds concurrently with the pseudo-random pattern generation. Hence this is a single-pass algorithm.
  - ✠ The technique is applicable equally to both combinational (test-per-clock) and fully scanned (test-per-scan) sequential circuits. It is independent of the size and number of scan chains.
  - ✠ Performance of the proposed algorithm has been compared with conventional seed selection algorithms on ISCAS [3] combinational and sequential benchmark circuits. The proposed algorithm has been observed to outperform the existing methods with respect to the test lengths (sequence lengths).

Once the novelty of the algorithm is established, it is applied on various synthetic SoC environments built out of benchmark circuits from ISCAS as cores.

- **A new design-for-test for sequential cores for reduction in test application time.** A mechanism of concurrently loading and unloading all the flip-flops in a sequential CUT, during testing with pseudo-random patterns, has been proposed. An expander and a compactor constitutes the DFT. The DFT has the following properties:

  - ✠ A core having the proposed DFT has either no or very short scan chains.
  - ✠ The number of test inputs is not increased; consequently, the size of the part of the TPG leading to the test inputs remains the same.
  - ✠ At the boundary, inside the CUT, is an $n : m$ expander that expands the n-bit test inputs to m-bits (where, $n << m \leq 2^n - 1$), and thus can load m flip-flops in parallel. This reduces the test application time to the CUT considerably.
  - ✠ The total test time of each core is reduced by factors, depending on the amount of reduction in the length of the largest scan-chain.
  - ✠ One constraint of the method is that not every test pattern can be loaded into the internal flip-flops, if this expander is used. However, since only pseudo-random testing is done, every pattern is not required to achieve the desired fault coverage.
  - ✠ The affectiveness of the proposed approach is notably significant in pseudo-random built-in self-test based testing, as can be observed through the high fault coverage found from the experimental results on the ISCAS'85 and ISCAS'89 benchmark circuits.

- **A hierarchical static timing verification method for SoC.** A study of the several timing issues that may arise during integration of cores in SoC, has been presented. It also brings out the various check points where timing needs to be validated for modular SoC static timing verification. A bottom-up hierarchical approach of verifying the system level timing of a SoC is presented. The following points characterize the approach:

- It is a divide and conquer approach which often leads to a much faster system level validation.
- The timing abstractions of the cores are assumed to be given by core vendors. The interconnection delays of the SoC may be extracted from the post layout timing information, generated in the form of standard delay format (SDF) file.
- In the lowest level of the hierarchy are the cores and the glue logic. Glue logic are considered as modules like cores and timing abstractions are generated for each such module.
- The cores and glue-logic modules are then considered to be black boxes and are further clustered according to data-propagation distance. Each of these clusters are timing verified concurrently. This continues till clustering results in a single module, this is the top level module. Verifying this top level module for timing results in the verification of the entire SoC for timing closure.
- Such concurrent verification approach results in a reduction of verification time by factors, as has been shown by experimental results.

1.3 Organization of the thesis

The rest of the thesis is structured as follows:

Chapter 2 gives some background information and preliminary concepts of modular SoC design, timing verification and test.

Chapter 3 reports some related works to present the state-of-art in connection to the thesis.

Chapter 4 describes how a centralized BIST core, namely *coreBIST*, can be used to act as a test source and sink to all logic cores, for stuck-at fault coverage. It provides the design and implementation results of the BIST core. The corresponding TAM design is shown.

Chapter 5 gives a seed selection algorithm, namely the *slideBox* algorithm, to reduce the pseudo-random test sequence length of a given PRPG, without compromising on the fault-coverage.

Chapter 6 describes a test-expander design that expands an n-bit test vector to m-bits,

where, $n \leq m \leq 2^n - 1$. The expander is composed of functions of XOR gates. It aids in loading m number of internal flip-flops concurrently with just n number of test input ports. The test application time is, therefore, much reduced. This chapter also describes the design of a compactor that compacts responses from m scan-chains to k number of output ports.

Chapter 7 at first explores the requirements for a hierarchical system level timing verification of a core-based design, such as a SoC, and then an algorithm for hierarchical timing verification is proposed. Timing paths of functional structures and test structures are analyzed separately. The timing abstractions of the cores are assumed to be given by core vendors. Apart from the inherent core-level hierarchy in an SoC, the depth of the hierarchy in verification is increased by a method of clustering the interface timing paths based on their timing dependencies. Each of these clusters are timing verified concurrently. This continues till clustering results in a single module. This is the top level module. Verifying this top level module for timing results in the verification of the entire SoC for timing closure. Such concurrent verification approach results in a reduction of verification time by factors.

Chapter 8 concludes the thesis and discusses some possible directions of future work.