Improving System-On-Chip Test Networks For: Bandwidth, Security, and Power

Saurabh Gupta
Southern Methodist University, sgupta@smu.edu

Follow this and additional works at: https://scholar.smu.edu/engineering_compsci_etds

Part of the Computer and Systems Architecture Commons, Digital Circuits Commons, Hardware Systems Commons, and the Other Computer Engineering Commons

Recommended Citation
https://scholar.smu.edu/engineering_compsci_etds/5

This Dissertation is brought to you for free and open access by the Computer Science and Engineering at SMU Scholar. It has been accepted for inclusion in Computer Science and Engineering Theses and Dissertations by an authorized administrator of SMU Scholar. For more information, please visit http://digitalrepository.smu.edu.
IMPROVING SYSTEM-ON-CHIP TEST NETWORKS FOR:
BANDWIDTH, SECURITY, AND POWER

Approved by:

Dr. Jennifer Dworak
Computer Science and Engineering
Dissertation Committee Chairperson

Dr. Daniel Engels
Provost Office - SMU

Dr. Ping Gui
Electrical Engineering

Dr. Eric Larson
Computer Science and Engineering

Dr. Sukumaran Nair
Computer Science and Engineering

Dr. Kundan Nepal
School of Engineering
University of St. Thomas
IMPROVING SYSTEM-ON-CHIP TEST NETWORKS FOR:
BANDWIDTH, SECURITY, AND POWER

A Dissertation Presented to the Graduate Faculty of the
Lyle School of Engineering
Southern Methodist University
in
Partial Fulfillment of the Requirements
for the degree of
Doctor of Philosophy
with a
Major in Computer Engineering
by

Saurabh Gupta

M.S., Computer Engineering, Southern Methodist University, Dallas, TX, USA
B.E., Electronics Engineering, Mumbai University, India

May 19, 2018
Modern System-on-Chips (SoCs) provide benefits such as reduction in overall system cost, and size, increased performance, and lower power consumption. Increasing complexity of these Integrated Circuits (ICs) has resulted in a higher probability of manufacturing defects. Manufacturing defects can result in the faulty operation of a system. Thus, it is essential to test an IC after it is manufactured to detect any possible faults in it. These SoCs include on-chip embedded instruments that can be used for test, debug, diagnosis, validation, monitoring, characterization, configuration, or functional purposes. IEEE 1687 Std. (IJTAG) provides a standard interface for the reconfigurable access and control of on-chip embedded instruments. Fast, and accurate tests are very desirable to the IC manufacturers to reduce time to market for their products.

To overcome some of the challenges associated with modern SoC testing, we designed novel IJTAG based designs that improve the overall bandwidth, and security of the test network. Bandwidth improvements were done using our new broadcast network design, and Parallel-IJTAG network design. We evaluated previously proposed secure IJTAG designs for their susceptibility to side-channel attacks, and explored mitigation strategies. Applying test data in parallel to different SoC partitions may result in higher peak power consumption during test compared to the functional power specifications, thus causing test slowdowns and/or producing erroneous test results. We devised a new graph coloring algorithm which helps in designing systems that require lower peak power consumption during SoC testing.
# TABLE OF CONTENTS

LIST OF FIGURES ................................................................. viii
LIST OF TABLES ................................................................. xi
ACKNOWLEDGMENTS ............................................................. xii

CHAPTER

1. INTRODUCTION ............................................................... 1
   1.1. Embedded Instrumentation ........................................ 1
   1.2. Motivation of the Thesis ........................................... 3
   1.3. Thesis Contributions ................................................ 4
   1.4. Thesis Organization .................................................. 6

2. BACKGROUND ................................................................. 8
   2.1. IEEE Std. 1149.1 ...................................................... 8
   2.2. IEEE Std. 1687 ......................................................... 8
       2.2.1. Overview of IJTAG .......................................... 9
       2.2.2. Network Instruction Bit (NIB) ................................ 9
       2.2.3. Segment Insertion Bit (SIB) ................................ 10
   2.3. Locking Segment Insertion Bit (LSIB) ......................... 11

3. PRIOR WORK ................................................................. 17
   3.1. IEEE Std. 1687 ...................................................... 17
   3.2. Broadcasting Test Data ........................................... 18
   3.3. Previous Broadcast Network Designs ......................... 18
   3.4. Test Security ......................................................... 20
       3.4.1. Power Analysis Attacks ................................... 22

4. A PARALLEL IJTAG NETWORK ........................................... 23
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0.1</td>
<td>SIBs in Parallel-IJTAG network - I</td>
<td>23</td>
</tr>
<tr>
<td>4.0.2</td>
<td>Parallel-SIBs (P-SIBs) in Parallel-IJTAG network - II</td>
<td>27</td>
</tr>
<tr>
<td>4.0.3</td>
<td>Parallel-IJTAG network using Addressable Parallel-SIBs</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>Hybrid IJTAG Network</td>
<td>31</td>
</tr>
<tr>
<td>5.</td>
<td>PARALLEL-IJTAG SECURITY</td>
<td>33</td>
</tr>
<tr>
<td>5.1</td>
<td>Securing Parallel-IJTAG network</td>
<td>33</td>
</tr>
<tr>
<td>5.1.1</td>
<td>LSIBs in Parallel-IJTAG network - I</td>
<td>33</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Locking-Parallel-SIBs (L-PSIBs) in Parallel-IJTAG network - II</td>
<td>34</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Cost of a Guess</td>
<td>36</td>
</tr>
<tr>
<td>5.1.4</td>
<td>Securing instrument access in a hybrid IJTAG architecture</td>
<td>38</td>
</tr>
<tr>
<td>6.</td>
<td>EFFICIENT PARALLEL TESTING: A CONFIGURABLE AND SCALABLE BROADCAST NETWORK DESIGN USING IJTAG</td>
<td>42</td>
</tr>
<tr>
<td>6.1</td>
<td>Broadcast Network - Without Hierarchy</td>
<td>43</td>
</tr>
<tr>
<td>6.1.0.1</td>
<td>NIB as Partition Broadcast Control Bit (BCB)</td>
<td>43</td>
</tr>
<tr>
<td>6.1.0.2</td>
<td>Broadcast Segment Insertion Bit</td>
<td>44</td>
</tr>
<tr>
<td>6.2</td>
<td>Broadcast Network - With Hierarchy</td>
<td>49</td>
</tr>
<tr>
<td>6.2.0.1</td>
<td>Partition Broadcast Control Bit (BCB) with locked state</td>
<td>49</td>
</tr>
<tr>
<td>6.2.0.2</td>
<td>Cluster Broadcast Control Bit (BCB)</td>
<td>49</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Test Time Improvements</td>
<td>50</td>
</tr>
<tr>
<td>7.</td>
<td>NEW LSIB KEY LOGIC CIRCUITS AND ANALYSIS</td>
<td>54</td>
</tr>
<tr>
<td>7.1</td>
<td>Key Logic 1 – AND-tree</td>
<td>55</td>
</tr>
<tr>
<td>7.2</td>
<td>Key Logic 1b – AND-tree with randomized key bit order</td>
<td>58</td>
</tr>
<tr>
<td>7.3</td>
<td>Key Logic 1c – Gate-tree with Compiler Optimizations</td>
<td>60</td>
</tr>
<tr>
<td>7.4</td>
<td>Mitigate Power Analysis Attacks by Sharing Key Bit Flip-Flops Among Multiple LSIBs</td>
<td>63</td>
</tr>
<tr>
<td>7.5</td>
<td>Key Logic with Programmable LSIB-Keys</td>
<td>64</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.</td>
<td>JTAG architecture and interfacing of serial IJTAG network to JTAG</td>
<td>10</td>
</tr>
<tr>
<td>2.2.</td>
<td>Network Instruction Bit</td>
<td>11</td>
</tr>
<tr>
<td>2.3.</td>
<td>SIB design</td>
<td>12</td>
</tr>
<tr>
<td>2.4.</td>
<td>Locking SIB that prevents Update from Assert/Deassert of SIB_Select</td>
<td>13</td>
</tr>
<tr>
<td>2.5.</td>
<td>Example scan network with 1 LSIB and 6 key bits</td>
<td>14</td>
</tr>
<tr>
<td>3.1.</td>
<td>Broadcast and daisy scan example from the IEEE Std 1687, [1, p. 232]</td>
<td>19</td>
</tr>
<tr>
<td>3.2.</td>
<td>Broadcast design example with exclusive access from the IEEE Std 1687, [1, p. 231]</td>
<td>20</td>
</tr>
<tr>
<td>4.1.</td>
<td>JTAG architecture and interfacing a Parallel-IJTAG network to JTAG</td>
<td>24</td>
</tr>
<tr>
<td>4.2.</td>
<td>Simple Parallel SIBs - Flat architecture</td>
<td>25</td>
</tr>
<tr>
<td>4.3.</td>
<td>Serial SIBs - Flat architecture</td>
<td>26</td>
</tr>
<tr>
<td>4.4.</td>
<td>n-bit Parallel-SIB using single update cell for an n-bit Parallel-IJTAG network</td>
<td>28</td>
</tr>
<tr>
<td>4.5.</td>
<td>n-bit Addressable Parallel-SIB. Address=(10...1)_n</td>
<td>30</td>
</tr>
<tr>
<td>4.6.</td>
<td>Broadcast network using Addressable-PSIBs to reduce the length of the scan paths</td>
<td>31</td>
</tr>
<tr>
<td>4.7.</td>
<td>JTAG architecture and interfacing a serial and parallel IJTAG network to JTAG</td>
<td>32</td>
</tr>
<tr>
<td>5.1.</td>
<td>Locking-SIBs in a Parallel-IJTAG network</td>
<td>34</td>
</tr>
<tr>
<td>5.2.</td>
<td>Locking-SIB using single update cell in an n-bit Parallel-IJTAG network</td>
<td>35</td>
</tr>
<tr>
<td>5.3.</td>
<td>TAP controller states to open an LSIB with a guess key pattern in the network</td>
<td>37</td>
</tr>
</tbody>
</table>
5.4. Steps (clock cycles) required in an attempt to open an L-PSIB in a parallel network with its key logic in the serial network, using a guess pattern during a random brute force attack. .................................................. 40

6.1. New IJTAG based reconfigurable broadcast network - Without hierarchy . . . 44

6.2. Modified SIB design with SIB-Lock, pipelined UpdateEn, and forced shift cell reset for the new configurable broadcast network ......................... 45

6.3. SIB unlock-update operations, with and without pipelined UpdateEn . . . . 48

6.4. New IJTAG based reconfigurable broadcast network - With hierarchy . . . . 50

6.5. Sample IJTAG networks used in Table 6.2 ................................. 51

7.1. Key logic 1 – AND-tree. Key length = n-bits. Hamming weight = j ....... 56

7.2. Switching activity values and the corresponding key bit values for each bit location of the LSIB key, obtained by applying walking-1 patterns. Hamming weight of the LSIB key = 21. Pattern reference background = all zeros. Circuit = Key Logic 1 – AND-tree for a single LSIB key. ..................... 57

7.3. Switching activity values and the corresponding key bit values for each bit location of the LSIB key, obtained by applying walking-1 patterns. Hamming weight of the LSIB key = 21. Reference background = all zeros. Circuit = Key Logic 1b – AND-tree with randomized key bit order, for one LSIB key. 59

7.4. Key logic 1c – Compiler synthesized gate-tree. LSIB key length = 48-bits. Hamming weight = j ................................................................. 61

7.5. Switching activity values and the corresponding key bit values for each bit location of the LSIB key, obtained by applying walking-1 patterns. Hamming weight of the LSIB key = 21. Reference background = all zeros. Circuit = Key Logic 1c – Gate-tree with compiler optimizations, for one LSIB key. 62

7.6. Switching activity values and the corresponding key-bitsum values for each bit location of the LSIB keys, obtained by applying walking-1 patterns. Hamming weights of keys: LSIB1 = 32, LSIB2 = 13, LSIB3 = 37, LSIB4 = 30. Pattern reference background = all zeros. Circuit = Key Logic 1c – Gate-tree with compiler optimizations, for four LSIB keys. ............... 64

7.7. Key logic 3 – Key logic with programmable LSIB Unlock key. LSIB key length = 48-bits. Hamming weight = j .................................................. 65
7.8. Circuit to detect brute force attack on LSIB key logics with shared key bit flip flops, and gate UpdateEn signal to the Update cells holding LSIB key bits. Key length = \( n \)-bits. Hamming weights of keys: LSIB1 = \( j \), LSIB2 = \( k \), LSIB3 = \( l \), LSIB4 = \( m \), redundant key = \( z \).

8.1. Staggered Test Clocks.

8.2. Colored partitions on an SoC – Colors here represent the stagger values used for Shift-Clock Stagger Assignment.

8.3. Planar graph Obtained from the US Map.

8.4. New Planar Graph Coloring Algorithm for Shift-Clock Stagger Assignment.

8.5. US planar graph colored using the algorithm shown in Fig. 8.4. Case (a): No color set provided by the user. Smallest-last ordering used for this algorithm is given in Table 8.1.

8.6. Color utilization results for the US graph shown in Fig. 8.3 using our new graph coloring algorithm Fig. 8.4 (cases a and b), and using the smallest-last/SL coloring algorithm from literature (case c). Smallest-last ordering used for both the algorithms is given in Table 8.1. Case (a): No color set provided by the user. Case (b): New algorithm executed using 5 available colors. Case (c): SL coloring algorithm.
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1. Clock cycles required to access 32-bit TDR\textsubscript{a} in a serial and a 4-bit parallel network - Flat architecture</td>
<td>27</td>
</tr>
<tr>
<td>4.2. Overall Access Times (no. of clock cycles) - Flat architecture, three instruments. C=Concurrent, and S=Sequential instrument access</td>
<td>29</td>
</tr>
<tr>
<td>5.1. Expected time to unlock an L-PSIB in a hybrid (serial and parallel) network. (s=256, n=64, d=10), shift frequency=10MHz</td>
<td>41</td>
</tr>
<tr>
<td>6.1. Broadcast SIB States</td>
<td>46</td>
</tr>
<tr>
<td>6.2. No. of clock cycles required for read/write of test data using four different IJTAG network designs.</td>
<td>52</td>
</tr>
<tr>
<td>8.1. Smallest Last Ordering of the Vertices from the US Planar Graph in Fig. 8.3. Deletion degree (Del. Deg.) in the table shows the degree of the vertex when it was deleted from the graph during smallest-last ordering. Order number shows the order in which vertices get colored—from 0 to 50. DC is included; total vertices = 51.</td>
<td>76</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Professor Jennifer Dworak, for her constant support and patient guidance throughout my PhD program. This dissertation and the associated publications would not have been possible without Dr. Dworak’s extensive mentoring. Working with Dr. Dworak has been an incredible learning experience and it has greatly helped in improving my critical thinking and problem-solving abilities.

I would also like to thank the rest of my committee members: Professor Daniel Engels, Professor Suku Nair, Professor Ping Gui, Professor Eric Larson, and Professor Kundan Nepal for their constructive suggestions and insightful comments. I would also like to express my gratitude to Dr. David Matula for the very informative discussions on graph theory algorithms.

My appreciation also goes to my mentors and colleagues for their help during my internship at NVIDIA: Vinod Pagalone, Jon Colburn, Milind Sonawane, Jae Wu, Ran Wang, Bonita Bhaskaran, and Kaushik Narayanun.

I am thankful to my friends for motivating me throughout the course of my research. Finally, I would like to thank my family for their endless support and love.
Chapter 1
INTRODUCTION

Moore’s law [2] predicted that the number of transistors in the Integrated Circuits (ICs or chips) will double every 18 months. So far this prediction has been largely true because of the decreasing dimensions, referred to as feature size, of the transistors and interconnecting wires. The reduction in feature size has resulted from the improvements in the semiconductor design and manufacturing techniques. This has helped in designing increasingly complex Integrated Circuits (ICs) that can contain up to a few billion transistors. System-of-Chips (SoCs) are complex ICs that integrate major functional elements such as microprocessors, digital signal processors (DSPs), on-chip memory, peripheral interfaces, sensors, communication modules, etc [3]. While SoCs provide several benefits such as reduction in the overall system cost and size, increased performance, and lower power consumption they also pose challenges to the VLSI engineers in each phase of their design, verification, debugging, fabrication, and testing.

Due to unavoidable process variation or random localized manufacturing imperfections, it is impossible to fabricate each and every IC without any defects. The increasing number of transistors on a chip and reducing feature size has also resulted in a higher probability of manufacturing defects. These manufacturing defects can result in the faulty operation of a system. Also, nanoscale-devices are more prone to issues such as crosstalk, material aging, soft errors, and electro-migration. Thus, it is essential to test an IC after it is manufactured to detect any possible faults in it. These tests are performed at several stages of production, such as wafer level, package level, and final test before it is sent to the market.

1.1 Embedded Instrumentation

Complex SoCs contain hundreds of circuits embedded into them for a specific control or data-collection purpose, commonly referred to as instruments. These on-chip instruments
can be used for test (Design for Test), debug and diagnosis (Design for Debug/Diagnosis), validation (Design for Validation), monitoring, characterization, configuration, or functional purposes. Embedded instrumentation is cost-effective and efficient compared to using standalone, external instruments like oscilloscopes and logic analyzers to validate designs, verify the signal integrity on chip-to-chip interconnects and input/output (I/O) buses, test manufactured assemblies, and diagnose failures. Moreover, multi-core and multi-chip devices, and high volume of semiconductor devices that are integrated on circuit boards has made using external and modular instrumentation increasingly difficult. Some of these embedded instruments are listed below:

- **Design for Test (DFT):** Instruments like digital and analog converters, pattern matchers and generators, voltage and phase controllers, limit comparators, bit-stream comparators in I/O blocks of memory chips, integrated programmable pseudo-random pattern generators, and others are included to provide test coverage within chips and at the pins. Built in Self Test (BIST) instruments such as Logic BIST (LBIST), Memory BIST (MBIST), and I/O BIST are used during several test phases of a device.

- **Design for Debug/Diagnosis and Validation:** Instruments such as logic analyzer, bus analyzer, and virtual I/O allow to check any internal signal or node, including embedded hard or soft processors. Embedded instruments can also help to obtain eye diagram or other instrumentation plots that validate the performance of high-speed receivers. Other examples include trace buffers, shadow capturing of registers, and programmable random vector generators.

- **Monitoring:** Instruments can monitor, sequence, track, and margin multiple system voltages, adjust voltages according to pre-programmed limits, and store fault data for future analysis. Process monitors (used to identify systemic problems), voltage monitors (used to identify IR-drop problems), and temperature monitors, and memory error detectors are some more examples of such instruments.
1.2 Motivation of the Thesis

The increasing complexity of SoCs necessitates an increasing number of embedded instruments to be included in the chip and the board on which it is placed. The usual method of accessing and controlling these instruments through the IEEE Std. 1149.1 (Joint Test Action Group - JTAG) [4] is not scalable anymore [5]. The IEEE Std. 1687 (Internal-JTAG) [6] is an extension of the IEEE 1149.1 standard. IEEE 1687 focuses on the access and control of on-chip embedded instruments. IEEE 1687 allows access to these embedded instruments, connected in a network on the device, in different configurations by dynamically reconfiguring this network.

The instrument access network used in the IEEE Std. 1687 is a serial scan network. Previous works have shown ways to optimize the network design and reconfiguration strategies to reduce the time required to access instruments in the network. Unfortunately, instruments that require a large amount of test data and several accesses during test mode still result in long test times when the test data is shifted through the scan path serially. High bandwidth access to these instruments through parallel network can be useful in reducing the total access time. It is important to analyze the bandwidth improvements and security aspects of such a parallel network with respect to the serial scan network used in the IEEE Std. 1687.

To meet high performance requirements, System-on-Chips (SoCs) may include multiple replicated copies of functional embedded cores. Replicated copies of functional embedded cores and their associated test/debug embedded instruments require identical test data. Thus, the same test data needs to be shifted several times through the network for each instance of the replicated embedded cores. This results in long test data write times in a standard IJTAG scan network—because the data is shifted through the network serially. To solve this problem, a configurable and scalable broadcast network architecture using IJTAG is required.

As some of these embedded instruments can provide very detailed information regarding the internals of the chip it is important to secure access to these embedded instruments using IEEE 1687 network. The IEEE 1687 Standard offers new opportunities for securing scan networks. In [7], Dworak et al. introduced a method to secure access to particular scan segments—allowing fine-grained protection of different parts of the scan network where
different instruments could be protected by different keys (or sets of keys).

Previous work shows that brute force attacks on the secure instrument access mechanism that was introduced in [7] may take thousands of years based on the size of the key and some other parameters of the network. However, this method of securing access to the instruments might still be susceptible to side channel attacks such as power analysis attacks. A power analysis attack takes advantage of the fact that the amount of instantaneous power consumed by a circuit is typically data dependent and thus extract information regarding the target circuit by statistically analyzing several power traces of the circuit obtained by applying different sets of input data. Thus, it is important to study the susceptibility of these proposed secure designs (of instrument access) to power analysis attacks in an IEEE 1687 network.

As the SoCs become more complex, and larger in sizes the number of scan flip-flops have also increased along with the increasing number of embedded instruments. This has also resulted in a dramatic increase in the total test data volume and the number of test patterns. Thus, the number of test cycles required to shift in the test data has also increased. To reduce test times, this data is often scanned in and applied to the different SoC blocks in parallel. However, parallel testing can result in increased simultaneous switching activity in an SoC beyond its functional specifications. Thus, it is important to explore test design solutions to reduce the peak test power while shifting test data in parallel to multiple SoC partitions.

In the next sub-section, the contributions of this thesis regarding security and bandwidth optimization problems in the area of reconfigurable on-chip embedded instrument access networks are presented.

1.3 Thesis Contributions

- **Parallel-IJTAG Networks [8]**: We explored the design of a Parallel-IJTAG network using network elements that were previously designed for serial IEEE 1687 networks. We showed that the overhead of dynamically reconfiguring the networks is the same in both parallel and serial designs. We also calculated the cost of instrument access-times (number of clock cycles) in both the parallel and serial designs to show access time reductions in a parallel network.
Furthermore, we also improved designs of the network elements that were previously designed for serial networks – so that they are better suited for a parallel network and improve instrument access times.

**Secure Instrument Access in Parallel-IJTAG [8]:** We explored some of the security implications of parallel IJTAG networks. We showed how the security mechanisms that were introduced for serial networks can also be used in Parallel-IJTAG networks and investigated the cost and effectiveness of these designs in a Parallel-IJTAG network.

Despite the increased bandwidth of the instrument access network, the security mechanisms we introduced for serial network are effective in a parallel network. For example, the expected amount of time for successful random brute force attacks on this secure instrument access mechanism using keys of sufficient sizes is over 12,000 years.

**Broadcast Network Design Using IJTAG [9]:** To reduce the time required to apply identical test data to these replicated cores, we designed a novel broadcast network architecture that harnesses IEEE Std 1687 (IJTAG). Our architecture provides highly configurable broadcast/multicast and daisy modes, allowing one to selectively apply test data to any combination of embedded modules.

The broadcast network is also scalable, supports hierarchical network architectures, and can be easily interfaced with other IJTAG-compliant test architectures. The new broadcast network provides a trade-off between network reconfigurability and the programming overhead of the network reconfiguration bits. It saves up to 65-80% test time in our sample test data broadcast scenario when compared to serial and prior broadcast IJTAG networks. Compared to a serial network, our broadcast network requires only one extra reconfiguration bit.

**Susceptibility to Power Analysis Attacks [10]:** We first investigated the ability of an attacker to extract the correct key by simply observing the switching activity that occurs when various bit patterns are shifted through the chain. The purpose of this mechanism is to offer low cost secure access to the internal embedded instruments,
thus it is desirable to mitigate power analysis attacks on this mechanism without significantly adding area overhead.

- **Mitigating Power Analysis Attacks** [10]: Next, we modified these secure test network designs and showed that the modified design not only reduces area overhead by reducing the number of flip-flops required to hold the key bits, but also obfuscates the switching activity related to a single key.

- **Map Coloring to Reduce Test Power**: Finally, we studied methods to reduce peak test power while shifting parallel test data in multiple SoC partitions. One existing method attempts to solve this problem by staggering a single test clock into different shift-clock phases. These staggered shift-clock phases need to be assigned to the SoC partitions such that no two adjacent partitions receive the same staggered phase (no conflicts). This assignment problem has been described as *Shift-Clock Stagger Assignment* (SCSA) problem in the existing literature. However, previously proposed solutions do not guarantee a conflict free assignment even with 5 or 6 staggered clock phases. We designed a new graph coloring algorithm that offers conflict free SCSA results.

  - Our novel planar graph coloring algorithm results in a conflict free assignment of the staggered clock phases to the SoC partitions. This algorithm also balances the usage of all the available colors (staggered clock phases) while working within hardware design constraints. A planar graph coloring algorithm with these set of features wasn’t available in the literature until now. In some cases, our algorithm also uses fewer colors than an existing planar graph coloring algorithm from the literature.

1.4 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2, we present an overview of the relevant architectural features of the IEEE 1687 test network, previous work in LSIB based security, and concepts used throughout this thesis.

In Chapter 3, we discuss the past research that has been done for reconfigurable instrument access networks, and throughput optimization. It also presents a broad overview of
work related to securing scan chains and mitigating power analysis attacks.

Chapter 4 discusses parallel IJTAG architecture and explores design of different Parallel-SIBs. Chapter 5 extends the analysis to securing a parallel IJTAG network. Chapter 6 presents a configurable and scalable broadcast network design using IJTAG for efficient parallel testing of replicated embedded cores in SoCs.

Chapter 7 contains an evaluation of several key logic circuits for switching activity correlation, and further enhancements to LSIB based security.

Chapter 8 describes the Shift-Clock Stagger Assignment (SCSA) problem and details the previously proposed solutions. It also contains relevant background information useful for the discussion of our new graph coloring algorithm. Finally, the new graph coloring algorithm and its usefulness in solving the SCSA problem is discussed. Chapter 9 concludes the thesis.
Chapter 2
BACKGROUND

2.1 IEEE Std. 1149.1

The IEEE 1149.1 Standard [4], commonly known as the JTAG (Joint Test Action Group) standard, defines test logic that is designed to test the interconnects between chips on a single printed circuit board. The standard describes the Test Access Port (TAP) controller that consists of four signals: TMS (test mode select), TDI (test data in), TCK (test clock) and TDO (test data out), with an optional fifth signal, TRST (test reset). A TAP controller that dictates the operation protocol of capture-shift-update uses a defined state machine that allows instrument data to be captured and shifted out serially (a read operation) and new data to be shifted in serially and updated (a write operation). The input data is supplied serially through the TDI input pin and synchronized to the rising-edge of TCK, while the output data is serially shifted out through the TDO output pin and synchronized to the falling-edge of TCK.

Test data may take one of multiple paths through the JTAG register architecture since the standard architecture defines multiple register paths such as the instruction register, boundary scan register, bypass register, ID code register, and generic test data registers (TDRs). TCK in conjunction with TMS drives the FSM, and it is the FSM that determines on an initial level whether the scan path taken is either an instruction register (IR) scan or one of the defined data register (DR) scans – the specific DR taken depends on the encoding in the IR. In other words, TCK and TMS determine the state of the system.

2.2 IEEE Std. 1687

The IEEE 1687 standard [6] is an extension of the IEEE 1149.1 standard. IEEE 1687 focuses on the access and control of on-chip embedded instruments for device test, debug, and configuration. Instead of using different instructions in the IEEE 1149.1 Instruction
Register (IR) to access embedded instruments in different configurations, IEEE 1687 defines the use of scan data within the scan chain itself to dynamically reconfigure the chain and open access to new chain segments. Specifically, special scan cells that act as *Network Instruction Bits* (NIBs) are used to generate control signals and reconfigure the scan chain by placing an appropriate value in the update cell of the NIB during the UpdateDR (Update Data Register) cycle of the IEEE 1149.1 state machine. A NIB that can reconfigure the scan path is called a *Segment Insertion Bit* (SIB).

This distributed control approach allows arbitrary combinations of instruments to be accessed concurrently without requiring new instructions to be encoded for the instruction register. As such, it also enables easier design reuse when the combinations of instruments included in a new version of a design changes because the instruction decoding for the instruction register does not necessarily need to be redesigned to handle the changes.

2.2.1 Overview of IJTAG

Fig. 2.1 shows an overview of the JTAG architecture and the interfacing of the IEEE 1687 JTAG network to JTAG. Note that additional test data registers (TDRs) that don’t implement IJTAG may also be present and selected through placing an appropriate instruction in the instruction register (IR).

An IEEE 1687 test network provides a scalable plug-and-play interface for interfacing and accessing the on-chip embedded instruments. Moreover, the dynamic reconfiguration of the IEEE 1687 network through distributed control has significant advantages over a pure instructional-based approach. Design modifications to the test network, including the addition of new embedded instruments, is easier because the decoding of the IEEE 1149.1 instruction register does not need to change. It is also possible to access various combinations of hundreds of instruments without requiring a new instruction for each combination.

2.2.2 Network Instruction Bit (NIB)

The distributed control is obtained using special data-side scan path cells called Network Instruction Bits (NIBs), shown in Fig. 2.2. Control signals such as “local reset” and “deny” can be generated by placing correct values in the update cell of a NIB. For example, a local reset signal could be used to reset an individual instrument’s TDR. In contrast, conducting
a reset of some portion of the JTAG architecture is generally accomplished by passing the state machine through the Test-Logic-Reset (TLR) state - which has the bad side effect of resetting all JTAG chips on a board attached to the same active daisy chain. A “deny” signal could be used to prevent capture and update operations on a specific TDR to help with the debugging procedures. Implementing similar functionality using the IEEE 1149.1 JTAG would require a separate instruction for each instrument TDR on the scan path.

2.2.3 Segment Insertion Bit (SIB)

A Segment Insertion Bit (SIB) is a special type of NIB that is used in an IJTAG network to dynamically reconfigure the active shift path by adding (or removing) a scan segment to (or from) the active TDI-TDO shift path, as shown in Fig. 2.3. The SIB(_Select)* signal in Fig. 2.3 can be asserted or de-asserted by placing an appropriate value in the update cell of the SIB. When the SIB(_Select)* signal is asserted, it causes the ScanMux to select the input connected to the TDO2 signal, which is then fed to the SIB shift cell. Thus the “Extra scan segment” between TDI2 and TDO2 gets added to the active shift path. The SIB
is considered to be in “open” state during this mode. The $SIB_{Select}^*$ signal also allows the Capture, Shift, and Update control signals from TAP controller to propagate to the “Extra scan segment”. When the $SIB_{Select}^*$ signal is de-asserted, the active shift path shown consists of only the SIB cell between TDI and TDO. Thus, the “Extra scan segment” is bypassed. The SIB is considered to be in the “closed” state during this mode.

### 2.3 Locking Segment Insertion Bit (LSIB)

Unfortunately, while IEEE 1687 has significant advantages for embedded instrument access, if it is unsecured, it also provides significant opportunities for an attacker to access those embedded instruments and data as well (and using easily and readily available off-the-shelf HW and SW). An attacker can easily shift appropriate data into the scan chain to update each scan cell with a logic 1 or a logic 0, thus changing the length of the scan chain containing any SIBs and obtaining access to the data and instruments behind them.

For example, by accessing an unsecured IEEE 1687 network, attackers may obtain information related to encryption keys, IC identifiers, sensor data, trace buffers and hardware monitors. Attackers may be able to access memories through an MBIST (Memory BIST) engine or reconfigure the IC itself by setting appropriate configuration bits. Hence securing
access to embedded instruments in an IEEE 1687 network is critical to the integrity of the IC's data and functionality.

In [7], Dworak et al. introduce Locking Segment Insertion Bits (LSIBs) to secure access to segments in IEEE 1687. To open an LSIB, a preselected key must be present in the scan chain. This key is then used to enable the update signal for the LSIB. Thus LSIBs provide fine grained protection and secure access to different instruments using different keys or set of keys. The circuitry that checks the values placed in the key bits on the scan chain against the correct value of the LSIB’s key is referred to in this paper as the "key logic". Further
optimizations to LSIB based security in an IEEE 1687 network were discussed in [11–13]. One possible implementation of an LSIB is shown in Fig. 2.4.

An LSIB is a SIB that has been modified to prevent the SIB from “opening” or “closing” unless the conditions needed to “Unlock” the LSIB have been met. In this figure, an AND gate has been added so that an update operation (Update Data Register cycle of the TAP
controller) on the update cell can only be performed if the Unlock signal is set to a 1. In general, the value of the Unlock signal is determined by the values of other signals, such as the values placed in other scan cells in the IJTAG network.

Fig. 2.5. shows a 10-bit scan path segment with an LSIB (shown in orange) and six key bits (shown in blue) on the chain. The shift and update cells of each scan cell are shown separately. All the key bits are in the update cells of the scan cells.

Figure 2.5: Example scan network with 1 LSIB and 6 key bits.

The additional scan path between TDI2 and TDO2 can be inserted at any point in the normally accessible scan path since it is represented by a ScanMux which is not required to be co-located with its control (update) bit. In the example circuit in Fig. 2.5, the additional scan path is inserted between shift bit-2 and bit-3 next to the LSIB. The AND gates in the Fig. 2.5. constitute the key logic circuit. If the correct key value is placed in all the key bit locations shown in Fig. 2.5, then the key logic circuit generates the Unlock signal shown in
Fig. 2.4. The last AND gate, feeding into the update cell (orange color) of the LSIB, is the same as the AND gate in Fig. 2.4.

Thus, from the description of the logic shown in Fig. 2.4. and Fig. 2.5. we can see that to allow access to the additional scan path between the TDI2 and TDO2, two conditions are required to be met. First is to generate the Unlock signal, that allows the UpdateEn signal to the update cell of the LSIB, by placing the correct key value in the key-bit locations of the scan chain, so that the update cell can receive data from the shift cell. This key value is referred to in this paper as the “LSIB unlock key”, or simply “LSIB key”.

The second condition is to place appropriate value in the update cell, so that the update cell generates the Select* signal when the UpdateEn is asserted, consequently enabling access to the additional scan path. This appropriate value that should be placed in an LSIB’s update cell is referred to in this paper as the “LSIB update cell value”. The SIB (and consequently, the LSIB) should be made such that the timing separates these two conditions to becoming two separate DR operations – a SIB does not react to its own update such that it does a spurious Write upon opening or closing – some hardware designs delay the actual generation of the Select* signal by $\frac{1}{2}$ a TCK cycle.

So, these two operations, which open (or close) an LSIB, would need two Data-Register-Scans (DR-Scans), which means two passes through the data-side of the JTAG State-Machine. The first condition sets the keys, the second condition enables the LSIB to open or close, but the actual access and operation of the added scan path isn’t until the third pass through the DR-side of the FSM – a hacker will not receive feedback of success or failure until the third DR-Scan.

The time required for an attacker to unlock an LSIB increases based on the scan chain length and number of key bits. Depending on the length of the chain, the size of the key, and the scan shift frequency, it may take hundreds or thousands of years on average to successfully open an LSIB with a brute force attack that attempts to simply guess the correct key by shifting bits through the network.

Other variations of this approach are also possible. For example, one may also place the key bits in the Shift cells of other scan cells of the chain. This decreases the amount of time required for an authorized user to access the instrument. However, it potentially provides
less security because it is easier for an attacker who does not know the key to make opening
attempts with new key guesses more quickly. It may also make power analysis attacks easier.
However, to some extent, this reduction in security can be counteracted by adding additional
key bits. For example, it was shown in [7], that the expected amount of time for a successful
brute force attack on a 48-bit long LSIB key in a scan chain approximately 5000 bits long and
operating at a scan shift frequency of 10 MHz is on the order of thousands of years. Thus,
LSIBs provide a scalable method of securing access to the internal embedded instruments
in an IJTAG network, where overhead (both access time and key bits) can be traded for
increased security.
Chapter 3
PRIOR WORK

3.1 IEEE Std. 1687

Previous works have discussed methods to calculate the time required for instrument access and methods to reduce overall instrument access times in an IEEE 1687 network. Algorithms for automated design and optimization of 1687 networks were discussed in [14]. In [15], a method to calculate the overall access time (OAT) for a given IEEE 1687 network was presented. OAT calculations were performed on flat and hierarchical network architectures using two access schedules—a sequential schedule and a concurrent schedule. A methodology to evaluate the test cost of reconfigurable networks, according to the functional fault model introduced in [16], was presented in [17]. The problem of test scheduling to reduce test access-times while satisfying resource and power constraints was also discussed in [18], and hybrid test schedules (a combination of sequential and concurrent schedules) were explored. The hybrid test-scheduling method to reduce OAT while satisfying resource and power constraints that was proposed in [18] was further optimized in [19]. In [20], an efficient scalable-methodology for the delivery and localization of interrupts was presented based on hierarchical, multi-mode IJTAG networks.

The authors of [21] investigated the potential use of parallel test access in the context of 3D stacked ICs. As part of this work they discussed increasing the number of scan paths in an IJTAG network that used the “Remote Controlled Scan Mux Architecture” (RCSMA) for scan path reconfiguration. While RCSMA has advantages, it also requires a pass through the IR (Instruction Register) scan to reset the value of the Scan Control Unit each time before the scan chain can be reconfigured.

As the number of instruments that must be accessed concurrently in an IEEE 1687 network increases, the amount of data that must be shifted through the chain on every
Capture-Shift-Update cycle of the JTAG state machine increases as well. This can dramatically increase test time in a traditional IEEE 1687 scan network—especially if the instruments have long TDRs—because the data is shifted through the network serially from the test data in (TDI) port to the test data out (TDO) port.

To help address this issue, this thesis explores a Parallel IJTAG architecture that is reconfigured through the use of Parallel-SIBs (P-SIBs).

### 3.2 Broadcasting Test Data

In VLSI testing, the concept of broadcasting test data has been traditionally associated with applying the same test data to multiple internal scan chains. This was first proposed in [22] and [23]. Other scan architectures, such as Illinois scan have also used broadcast scan modes [24,25]. Several compression based test architectures, such as [26], [27], and [28] have used broadcast scans schemes. The conceptual modular architecture using a Test Access Mechanism (TAM) and a core test wrapper was published in [29]. A TAM and wrapper architecture for parallel testing of identical cores was discussed in [30]. Parallel testing of multiple identical cores using a broadcast-based TAM has also been discussed in [31–33]. Along with improving the TAM bandwidth, several other solutions to reduce test times and leverage wasted shift cycles for test time/quality improvements have been proposed [34–37].

The IEEE 1500 Standard [38] for embedded core-based integrated circuits defines a wrapper parallel port (WPP) as a parallel interface to the IEEE 1500 wrapper. This parallel access mechanism provides increased data bandwidth to the wrapper core. However, the IEEE Std 1149.1 [39] for board testing and the IEEE Std 1500 [38] do not provide broadcast scan mechanisms to apply identical test data to the replicated copies of embedded cores in an SoC. In contrast, broadcast network designs to apply identical data to multiple test instruments were briefly discussed in the IEEE Std 1687 [1, pp. 230-232]. In addition, a Parallel-IJTAG network architecture and some optimizations were discussed in [8], but a data broadcast mode was not included in that discussion.

### 3.3 Previous Broadcast Network Designs

An example broadcast network design from the IJTAG standard documentation is shown in Fig. 3.1. The network design shown in Fig. 3.1 supports both broadcast and daisy chain
modes. Here, the network can be operated in broadcast mode by asserting the broadcast control signal of the Network Instruction Bit (NIB). During the broadcast mode, all three instruments can be loaded simultaneously. However, there is no way to configure the broadcast mode such that some instruments remain bypassed during broadcast and only a subset of these instruments receive the broadcasted data. Also, there is no way to reconfigure the active shift path in daisy scan mode either.

![Broadcast and daisy scan example from the IEEE Std 1687, [1, p. 232]](image)

Another example broadcast network design from the IJTAG standard is shown in Fig. 3.2. This design can be used to broadcast test data to all three instruments in the network. It can also access and control exactly one instrument at a time by asserting Sel1, Sel2, or Sel3. However, it does not provide a daisy mode so that all the instruments can be read out concurrently in series. Similar to the design in Fig. 3.1, with this design there is no way
to configure the broadcast mode such that some instruments are bypassed during broadcast and only a subset of these instruments receive the broadcast data.

In contrast, our new broadcast network architecture provides configurable broadcast and daisy modes, and improves test data write times with very low additional area overhead.

### 3.4 Test Security

Historically, attackers trying to gain access to internal data and functionality of the chips and the board have used the JTAG port. For example, hackers have previously used the JTAG port to disable the Digital Rights Management (DRM) policy on an Xbox [40]. Unfortunately, more malicious attacks may also occur in which an attacker attempts to gain access to private data, debug instruments, test circuitry, state information, or configuration hardware by serially scanning in private or undocumented instructions or data into the
instruction register and scan chains.

As a result of successful scan chain-based attacks, methods of securing the scan chain and the JTAG ports have been developed. One particular method involves limiting scan chain access by using challenge response pairs [41–43]. As an example, in [42], Clark introduced a JTAG design that uses the SHA256 hash algorithm and a true random number generator to create challenge response pairs that limit access to chip internals via the scan chain. In [44], a security module and test control module are used to protect memory content. When the security module is in restricted mode, data encryption and decryption of the memory contents is disabled, the TDO pin is forced to a logic 1, and only limited access to the memory is permitted.

Scan chain attacks have also been successful in compromising cryptographic algorithm implementations. Yang et al. [45] show how scan chains can be used to mount side channel attacks on a hardware implementation of the Data Encryption Standard (DES). A scan chain-based side channel attack on a stream cipher was successfully shown by Mukhopadhyay et al. [46]. Similarly, Kamal and Youssef [47] describe the use of scan chain-based side channel attacks to retrieve the secret keys of the NTRUEncrypt cryptosystem. Additional scan chain-based attacks have been shown to extract the secret keys from ciphers [48].

To avoid scan chain-based attacks, it has been suggested that the scan chain be divided into segments whose access order may be reconfigured [49–51]. Reconfiguration may occur automatically when, for example, the first $k$ bits that are shifted into the scan chain do not correspond to the first $k$ bits of the key.

Additional security mechanisms have been proposed to further secure an IEEE 1687 scan chain network against attacks. LSIB based security proposed in [7] is discussed in Section 2.3. Liu and Agrawal [52] introduced a key generation mechanism that uses a linear feedback shift register (LFSR) to dynamically generate keys. A defined number of scan flip flops are used to create an LFSR that is used for key generation. In [53], Baranowski et al. propose a challenge-response protocol that uses secure segment insertion bits to reconfigure the length of the scan chain. All of these secret key methods make it harder for an attacker to use brute force attacks to gain access to hidden or secure segments of IEEE 1687 networks.
3.4.1 Power Analysis Attacks

To some extent, breaking key-based protection mechanisms in scan-based circuits is inherently different from breaking cryptographic keys. Unlike cryptographic analysis, breaking into a scan chain generally requires explicitly shifting data through the chain and performing capture and update operations until the desired behavior is seen. Thus, brute force attacks can be very difficult and time consuming.

However, these protection mechanisms may still be susceptible to side channel attacks, such as power analysis attacks. Kocher et al., [54] presented practical methods of performing power analysis on cryptographic circuits. One of the methods proposed to extract the secret keys using power analysis is called Simple Power Analysis (SPA). SPA can be used when a visual inspection of the circuit’s power trace can leak information regarding the secret key or the intermediate values propagating through the circuit.

In [55], two categories of countermeasures against power analysis were discussed: hiding and masking. Hiding involves design changes which makes power consumption random or uniform for all the input data values, and masking involves randomizing the intermediate values.

Several papers have proposed different methods of hiding and masking countermeasures [56,57]. To evaluate the robustness of these countermeasures in a circuit several simulation methods at different design stages from system level to post layout level have been proposed. High level of abstraction models were used to simulate leakage traces in [58], [59] proposes convolution-based framework to predict power consumption at each supply pad in the IC using post-layout design instead of performing the time consuming full-chip SPICE simulations, and [60] evaluates speed and accuracy of analog (SPICE) and digital (toggle count from VCD file) simulations for leakage analysis on post-layout netlists. Statistical model for subthreshold current while considering process variations have also been proposed [61].

Although previous work has investigated the ability of LSIB-based security approaches to withstand brute force attacks, the ability of an attacker to deduce a key from scanning intelligent patterns through the network and observing the switching activity was not explored.
Chapter 4
A PARALLEL IJTAG NETWORK

As discussed in Section 3.1 several methods of reducing SIB overhead and different instrument test scheduling algorithms have previously been proposed. However, even if the SIB overhead is reduced, the IJTAG networks discussed until now are limited due to their serial scan designs. Thus, the test time (number of clock cycles) depends on the size of the instrument’s TDR and the number of times an instrument is accessed. In this paper, to increase the bandwidth of the IJTAG network, we discuss two ways of designing a Parallel-SIB based distributed control network.

Fig. 4.1 shows a conceptual overview of the JTAG circuitry and the interfacing of a Parallel-IJTAG network to JTAG. The required Capture, Shift, and Update control signals to the instruction register and to the test data registers, including the Parallel-IJTAG network, are supplied from the TAP controller. The \(n\)-bit wide Parallel-IJTAG network is fed by TDI from the TAP and an \((n-1)\)-bit wide Parallel-TDI (P-TDI) bus. The \((n-1)\)-bit P-TDI and \((n-1)\)-bit P-TDO bus can be multiplexed with general purpose input/output (GPIO) pins on the chip such that during test mode the GPIO pins can be used for P-TDI and P-TDO. If enough GPIO pins cannot be multiplexed during the test mode then an alternative may be to supply the parallel data through SerDes input/output ports, if such ports are provided.

4.0.1 SIBs in Parallel-IJTAG network - I

The \(n\)-bit Parallel-IJTAG network is similar to a serial IJTAG network with multiple scan paths. In the simplest case, a \(k\)-bit TDR is divided into \(n\) registers (TDR segments) each of size \((k/n)\) bits. Each SIB in the serial IJTAG network is also replaced with \(n\) SIBs distributed over the parallel scan paths. If the length \((k)\) of the TDR and the width \((n)\) of the parallel network is such that \((k/n)\) is not an integer then the \(n\)-th TDR segment is padded with additional shift cells such that its length is equal to the other remaining \((n-1)\)
Figure 4.1: JTAG architecture and interfacing a Parallel-IJTAG network to JTAG.

TDR segments.

An $n$-bit Parallel-IJTAG network is shown in Fig. 4.2, and the corresponding serial IJTAG network with three SIBs is shown in Fig. 4.3. For the sake of simplicity, instead of showing TDI from the TAP and the additional $(n−1)$ P-TDIs for the parallel network separately, we labelled the entire $n$-bit input port as P-TDI. (This convention is followed in the rest of this paper.) These example network designs are based on the flat architecture. In this flat architecture, each SIB is in the active scan path regardless of its open or closed state. Thus, each additional SIB in this flat network contributes to the SIB programming overhead every time an instrument needs to be accessed through the network.

If a 4-bit Parallel-IJTAG network is designed based on the flat architecture shown in Fig. 4.2, then a 32-bit TDR$_a$ from the serial IJTAG network is divided into a set of four segments – TDR$_1$, TDR$_2$, TDR$_3$, and TDR$_4$. These four TDR segments are in separate scan paths connected to four different SIBs – SIB$_1$, SIB$_2$, SIB$_3$, and SIB$_4$. Thus the SIB$_a$ from the serial network is replaced by four single-bit SIBs in the 4-bit parallel network. Similarly,
other SIBs (SIB\textsubscript{b} and SIB\textsubscript{c} in this instance) and their respective TDRs (TDR\textsubscript{b} and TDR\textsubscript{c}) from the serial network can be implemented in this Parallel-IJTAG network.

Opening (or closing) SIBs requires scanning in their control bits through the scan path, which contributes to the SIB programming over head. Also, the JTAG state machine needs to be cycled in the sequence of Exit1-DR, Update-DR, Select-DR, Capture-DR, and Shift-DR states to perform update and capture operations on the TDRs between two shift operations. Thus, the instrument access-time in an IJTAG network depends on the SIB programming overhead, the TAP controller cycles, the instrument TDR length ($L$), and the number of times ($A$) the instrument needs to accessed during test.

To calculate the test time improvements in the Parallel-IJTAG network compared to the serial design, we calculate the SIB programming overhead and the number of clock cycles required to perform read/write operations on the instrument TDRs in the example flat network designs shown in Fig. 4.3 and Fig. 4.2. We assume the length of TDR\textsubscript{a} = 32-bits, and the Parallel-IJTAG network is assumed to have four parallel scan paths. We also assume that the instruments are accessed sequentially as opposed to concurrently in the order that

Figure 4.2: Simple Parallel SIBs - Flat architecture
the SIBs associated to these instruments appear in the scan path. Thus, SIB\textsubscript{a} in the serial design and in the parallel design (SIB\textsubscript{1}, SIB\textsubscript{2}, SIB\textsubscript{3}, and SIB\textsubscript{4}) is opened in the first scan operation by shifting in three bits in each scan path. Five more clock cycles are needed to apply UpdateDR and return to the ShiftDR through the JTAG state machine so that a new scan shift cycle can commence. (This is shown in Row 1 in Table 4.1). Once SIB\textsubscript{a} is opened, the active serial scan path will include the 32 bits of TDR\textsubscript{a}, and the four active scan paths of the parallel network will include the 8 bits of TDR\textsubscript{1}, TDR\textsubscript{2}, TDR\textsubscript{3}, and TDR\textsubscript{4}. Thus, the next scan operation in the serial network will shift in three bits for the SIBs and 32 bits for the TDR\textsubscript{a}. On the other hand, the parallel network will shift in three bits for the SIBs and 8 bits for the TDRs in each of the four scan paths. During each of these scan operations, the previous output vector from the instrument can be scanned out while scanning in a new input vector. Hence to access an instrument 5 times this scan operation will be performed 5 times in total (Rows 2 and 3). Finally, an additional scan operation would be required to scan out the final output vector and close SIB\textsubscript{a} (Row 4). The number of clock cycles required to perform these operations in both the serial and Parallel-IJTAG networks is shown in Row 5. Accessing the other instruments sequentially would require this process to be repeated again for each instrument. Concurrent access would involve fewer iterations of this procedure, but the shift cycles would be longer to accommodate multiple TDRs being present on the chain simultaneously.
Table 4.1: Clock cycles required to access 32-bit TDR\textsubscript{a} in a serial and a 4-bit parallel network - Flat architecture

<table>
<thead>
<tr>
<th>Operation</th>
<th>Serial IJTAG # clock cycles</th>
<th>4-bit Parallel IJTAG # clock cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open SIB\textsubscript{a}</td>
<td>3(SIBs) + 5(TAP)</td>
<td>3(SIBs) + 5(TAP)</td>
</tr>
<tr>
<td>Scan out previous output vector + Scan in new input vector (TDR\textsubscript{a})</td>
<td>(3 + 32) + 5</td>
<td>(3 + 8) + 5</td>
</tr>
<tr>
<td>Repeat previous operation 4 more times</td>
<td>[(3 + 32) + 5]*4</td>
<td>[(3 + 8) + 5]*4</td>
</tr>
<tr>
<td>Scan out final output vector</td>
<td>(3 + 32) + 5</td>
<td>(3 + 8) + 5</td>
</tr>
<tr>
<td></td>
<td>∑=248</td>
<td>∑=104</td>
</tr>
</tbody>
</table>

As shown in Table 4.1, the SIB programming overhead for the Parallel-IJTAG and serial network remains the same. However, the instrument access-times (number of clock cycles) decreases in the Parallel-IJTAG network depending on the number of parallel scan paths available. Table 4.2 shows the Overall Access Times (OAT) for three instruments in both parallel and serial networks based on flat architecture using both concurrent (C) and sequential (S) test schedules. The calculation for the OAT values is based on the discussion in [15]. We can see that the OAT for instruments in a parallel network reduces drastically as compared to the serial network, for both concurrent and sequential schedules. The total number of single bit SIBs required in an \( n \)-bit simple Parallel-IJTAG network increases by a factor of \( n \).

4.0.2 Parallel-SIBs (P-SIBs) in Parallel-IJTAG network - II

In the previous subsection the \( n \)-bit Parallel-IJTAG network design was obtained by simply adding \( n \) serial scan paths in parallel to the IJTAG network. The SIB programming overhead and the number of clock cycles required to shift SIB data remained the same as compared to the corresponding serial IJTAG network design. However, the area overhead contributed by the SIBs increased by a factor of \( n \). The \( n \)-bit Parallel-SIB shown in Fig. 4.4 reduces this SIB area overhead by using only one update cell and \( n \) shift cells.

To open this Parallel-SIB, first an appropriate value is scanned in the shift-1 cell and
Figure 4.4: $n$-bit Parallel-SIB using single update cell for an $n$-bit Parallel-IJTAG network
Table 4.2: Overall Access Times (no. of clock cycles) - Flat architecture, three instruments. C=Concurrent, and S=Sequential instrument access

<table>
<thead>
<tr>
<th>TDR length $L_a$, $L_b$, $L_c$</th>
<th>TDR Access $A_a$, $A_b$, $A_c$</th>
<th>Parallel Network $n=4$</th>
<th>$n=8$</th>
<th>$n=16$</th>
<th>Serial Network $n=1$</th>
<th>$n=1$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>C S C S C S C S</td>
<td></td>
<td></td>
<td>C S</td>
<td></td>
</tr>
<tr>
<td>32, 48, 64</td>
<td>5, 4, 6</td>
<td>284 372 174 262</td>
<td>119 207</td>
<td>944 1032</td>
<td></td>
<td></td>
</tr>
<tr>
<td>128, 64, 256</td>
<td>4, 3, 2</td>
<td>464 520 256 312</td>
<td>152 208</td>
<td>1712 1768</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32, 16, 64</td>
<td>6, 4, 8</td>
<td>300 396 190 286</td>
<td>135 231</td>
<td>960 1056</td>
<td></td>
<td></td>
</tr>
<tr>
<td>128, 256, 512</td>
<td>2, 2, 1</td>
<td>576 616 304 344</td>
<td>168 208</td>
<td>2208 2248</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

then an UpdateEn signal is applied. The update-1 cell generates the Select* signal which is applied to the muxes of the $n$ shift cells in the parallel scan paths. The TDRs connected to these shift cells are also enabled and now become part of the active parallel scan paths.

The SIB programming overhead of the parallel network designed using this Parallel-SIB design is the same compared to the previous parallel design or existing serial IJTAG network designs. Thus, the bandwidth improvements are the same as in the previous design. However, this design requires $(n – 1)$ fewer update cells for each Parallel-SIB compared to the previous parallel network.

4.0.3 Parallel-IJTAG network using Addressable Parallel-SIBs

In the P-SIB design discussed in the previous sub-section, the bit from the shift-1 cell is fed to the update-1 cell directly to generate the Select* for all of the parallel SIBs. Another way to design this P-SIB is such that the bits from some or all of the shift cells are AND’ed together in complemented or uncomplemented form before feeding the result to the update-1 cell. In this way, the Select* signal will be generated only if all of the shift cells in the Parallel-SIB contain the correct values during update operation. Thus, the parallel SIB is now addressable. The P-SIB shown in Fig. 4.5 is an addressable PSIB which opens when the bits 10...1 are shifted into the shift cells of the PSIB followed by an update operation.

One of the uses of addressable PSIBs can be in designing broadcast networks such as that shown in Fig. 4.6. When the address of the target PSIB is broadcast on all three parallel
Figure 4.5: \( n \)-bit Addressable Parallel-SIB. Address=\((10...1)_n\)
paths, the PSIB with the correct address opens and allows access to the instrument’s TDR. This network design can be useful in cases when the number of pins available for PTDI are limited and no two instruments need to be accessed concurrently. It also reduces the SIB programming overhead during each instrument access because the instruments are not connected to the scan-path serially.

Figure 4.6: Broadcast network using Addressable-PSIBs to reduce the length of the scan paths

4.1 Hybrid IJTAG Network

The test architecture shown in Fig. 4.1 includes only the Parallel-IJTAG network. Even though the parallel design provides high bandwidth access to the embedded instruments, it requires more SIBs compared the serial network and adds to the complexity of routing control signals to the additional SIBs.
Fig. 4.7 shows a conceptual overview of the JTAG circuitry and the interfacing of a serial and Parallel-IJTAG network to JTAG. This architecture can provide the designer with opportunities to make tradeoffs between the instrument access times and the area of the test network. Instruments that require high bandwidth or the instruments that need to be accessed several times during test can be included in the parallel network. Similarly, instruments with smaller TDRs or instruments that do not require frequent access during test can be placed in the serial network. Additional parallel and/or serial TDRs could also be included if desired.

Figure 4.7: JTAG architecture and interfacing a serial and parallel IJTAG network to JTAG
5.1 Securing Parallel-IJTAG network

Securing access to the embedded instruments in a serial IJTAG network using Locking-SIBs was shown to be effective against brute force attacks in [7]. In the following subsections we explore designs which use LSIBs to secure instrument access in a Parallel-IJTAG network.

5.1.1 LSIBs in Parallel-IJTAG network - I

As shown in Fig. 2.5, in a serial IJTAG network the key bits required to unlock the LSIB are present in the serial scan path. These key bits are present in the update cells (or shift cells) of other SIBs in the scan path or in the update cells (or shift cells) of scan cells specially inserted in the scan path to hold key bits. The output from these key bit holding cells is fed to a tree of AND and NOT gates (referred to as Key Logic in this paper) which generates the LSIB Unlock signal. In an \( n \)-bit Parallel-IJTAG network these \( j \) key bits are spread over \( j \) scan cells in \( n \) parallel scan paths. Fig. 5.1 shows an \( n \)-bit Parallel-IJTAG network with Locking-SIBs.

The \( n \)-bit Parallel-IJTAG network shown in Fig. 5.1 is similar to the one in Fig. 4.2, but SIB\( a \) in the network is replaced with LSIB\( a \). This LSIB\( a \) consists of \( n \) single-bit LSIBs (LSIB\( 1 \), LSIB\( 2 \), LSIB\( 3 \), to LSIB\( n \)), with one LSIB in each of the \( n \) parallel scan paths providing secure access to its own subcomponent of TDR\( a \). The unlock signal for the LSIBs in the parallel scan path is generated by the key logic circuit when the correct key bit values are placed in the update cells of the key-bit holding registers. (In this case, \( n \) update cells from SIB\( b \) and SIB\( c \) are used as key bits. In an actual implementation the key bits can be located anywhere in the scan path.)
5.1.2 Locking-Parallel-SIBs (L-PSIBs) in Parallel-IJTAG network - II

In the previous subsection, an $n$-bit LSIB with a $j$-bit key in a Parallel-IJTAG network was described by replacing the single-bit SIBs in the parallel scan paths by single-bit LSIBs. As in the case of the Parallel-SIBs described in Section 4.0.2, we can design an $n$-bit Locking-PSIB that consists of only one update cell and $n$ shift cells. Fig. 5.2 shows an $n$-bit Locking-PSIB that can be used in an $n$-bit Parallel-IJTAG network.

In a Parallel-IJTAG network designed using this Locking-PSIB and the Parallel-SIBs from Section 4.0.2, the key bits are scanned in the shift registers of the scan cells. The key bits scanned in these shift registers are fed to the key logic circuit that generates an unlock signal. This unlock signal gates the UpdateEn signal to the single update cell present in the Locking-PSIB. If an appropriate value has been shifted into the shift-1 register of the Locking-PSIB at the same time that the key logic has asserted Unlock, the update1 register will assert the Select* signal when the UpdateEn signal is applied. This Select* signal is
Figure 5.2: Locking-SIB using single update cell in an n-bit Parallel-IJTAG network
fed to the muxes of all the parallel components of the Locking-PSIB and adds the \((k/n)\)-bit TDR segments to their respective parallel scan paths.

5.1.3 Cost of a Guess

The previous two subsections showed ways to provide secure access to the embedded instruments in a Parallel-IJTAG network. We now discuss the cost of guessing a \(j\)-bit key in an \(n\)-bit Parallel-IJTAG network of length \(m\)-bits. Compared to the serial network in which each key bit is scanned in sequentially, in an \(n\)-bit Parallel-IJTAG network \((j/n)\) key bits can be scanned in one shift operation. Due to this, it might appear that the LSIB security in a parallel network may be easily defeated using brute force attacks that attempt to access the network by trying random keys. However, a Parallel-IJTAG network using the standard JTAG controller still needs to go through the JTAG FSM’s states for each key guess. In this subsection, we show how this affects the cost of applying guess keys through the Parallel-IJTAG network.

The key logic circuits in both the Parallel-IJTAG network and the serial network work the same way. The outputs from the key bit holding registers are applied to the key logic circuit in parallel, and if the correct key is applied then the key logic generates an unlock signal for the LSIBs (or L-PSIBS). Thus, the amount of time required to make a simple key guess during a random brute force attack in both networks depends simply on the length of the chain and the number of clock cycles required to traverse the JTAG state machine.

As discussed earlier, the SIB programming overhead for the Parallel-IJTAG network is same as in the serial network. Thus, the length of the two networks designed with same SIB hierarchy when all the SIBs are closed will be equal. In a serial network, when a SIB is opened to access a \(k\)-bit TDR, the length of the active scan path increases by \(k\)-bits. In the Parallel-IJTAG network, when an \(n\)-bit SIB is opened to access a \(k\)-bit TDR, the lengths of all the active parallel scan paths increase by \((n/k)\)-bits. Thus the length of an \(n\)-bit parallel network when all of the SIBs are open would depend on the length \(k\) of the TDRs and the number of \(n\) parallel scan paths.

To investigate the time required to access an instrument hidden behind an L-PSIB, we first assume that the length of the parallel scan paths is \((m\)-bits\) long, which is ideally longer
than the \((j/n)\)-bits on the scan path that correspond to the key and the LSIB bit that the attacker is trying to open. We also assume that the Locking-PSIB discussed in Section 5.1.2 is used in the network and that the key bits are present in the shift cells (as opposed to update cells). The TAP controller state transitions that are required in an attempt to open an LSIB with a random guess key scanned into the network is shown in Fig. 5.3

An UpdateDR should be performed after the guess key pattern is scanned in the network. If the guess key is correct then the key logic circuit would generate an L-PSIB unlock signal and the UpdateDR operation will open the L-PSIB if the correct value was simultaneously

![TAP controller states diagram](image-url)
placed in the shift-1 cell of the L-PSIB. This will add \((n/k)\)-bits of the TDR segments to the parallel scan paths. This UpdateDR operation is followed by cycles required to check if length of the chain has changed indicating that the L-PSIB is now open. Checking the length of the chain involves shifting a deterministic pattern of length \(d\) through the chain of expected length \(m\). Because the transition from UpdateDR to ShiftDR includes an intermediate CaptureDR state, any previously scanned in guess pattern may now be overwritten if the corresponding shift cells of the key bits are also designed to capture data. The deterministic pattern of length \(d\) is now shifted in, followed by the next guess key pattern, which is shifted through the length \(m\) of the parallel scan paths. Thus, the attacker would need to shift for \((m + d)\) cycles for each guess key. The cost of applying a guess key, attempting to open the L-PSIB, and checking if the chain length is changed is thus:

\[
Cost_{guess} = (5 + m + d) \text{ cycles}
\]

For an L-PSIB key of length \(j\)-bits the expected number of guess key patterns required is \(2^{j+1}\). The “plus 1” is for the correct value that must be clocked into the LSIB’s update cell. This equation also assumes that the chain is sufficiently longer than the key that even seemingly different random key guesses by an attacker who does not know the key bit locations may be identical in the key bits themselves. Even relatively small key sizes can lead to long brute force attack times. For example, the expected amount of time for a successful brute force attack on average for a key size of 56 bits in 256 bit long parallel scan chains with a shift clock frequency of 10MHz is over 60000 years.

5.1.4 Securing instrument access in a hybrid IJTAG architecture

As discussed in the previous section, the amount of time required to open an L-PSIB using guess keys depends on the size of the key and the length of the parallel scan paths. The length of the parallel network and its corresponding serial network is the same if all the SIBs are closed. However, as mentioned earlier, the length of the parallel network does not increase by the same amount as that of the serial network when a SIB is opened to allow access to an instrument’s TDR. Thus at any given time in a parallel network, if some of the SIBs are open, then the length of the parallel scan paths will be smaller than the length of the corresponding serial network designed with the same SIB hierarchy. As the number of
parallel scan paths in the Parallel-IJTAG network increases, and as the number of L-PSIBs that are open increases, this difference in lengths between the two networks will increase as well.

The hybrid serial and parallel network discussed in Section 4.1 provides further opportunities to mitigate the brute force attacks on L-PSIBs in a parallel network as the number of parallel scan paths increases. Because the hybrid architecture consists of both the serial and parallel networks, the key logic circuits of L-PSIBs from the parallel network can be moved to the serial network. Thus, these key logic circuits can be fed with key bits from the serial scan path. Moreover, these key logic circuits for the L-PSIBs that are moved to the serial network can share the same input registers as the other existing key logic circuits designed for LSIBs in the serial network.

In this design, an L-PSIB would be unlocked by first selecting the TDR containing the serial network with the JTAG state machine. If we assume that the attacker is starting in the SelectDR state of the JTAG state machine, Fig. 5.4 shows the subsequent steps that would be followed by an attacker trying to open an L-PSIB in a hybrid network.

First, if the serial chain length is \( s \), then leaving SelectDR, shifting a key guess into the serial network, applying UpdateDR, and returning to SelectDR requires \( s + 5 \) clock cycles. Next, the attacker must shift to the parallel network. This requires changing the instruction in the JTAG instruction register. Entering the IR-Scan half of the state machine takes one clock cycle. If the length of the instruction register is \( i \), then leaving SelectIR, shifting in the instruction, applying UpdateIR, and returning to SelectDR requires \( i + 5 \) cycles. Next, the attacker tries to get the correct value in the Update cells of any L-PSIBs in the hope that the key was set correctly and an L-PSIB will open. A new DR-scan begins and all 1’s are shifted into the parallel network. If the parallel network is of length \( m \), this requires \( m + 5 \) cycles. This is followed by a second DR-scan that attempts to check the length of the chain to see if an L-PSIB was opened successfully. This consists of shifting in a distinctive pattern of \( d \) bits followed by an \( m \)-bit pattern of all 0’s and applying UpdateDR. This requires \( m + d + 5 \) cycles. Next, assuming the attacker was unsuccessful at opening an L-PSIB with the all 1’s attempt, he can use the 0’s already in the chain to attempt to get the correct value in the L-PSIB’s update cell and once again check the length of the chain. This requires an
Data Register (DR) scan – shift guess pattern in the \( s \)-bits long serial network

Instruction Register (IR) scan – switch to the parallel network

Data Register (DR) scan – shift all 1s to open an L-PSIB in the \( m \)-bits long parallel network

DR scan – shift a \( d \)-bit deterministic pattern (check length), and shift all 0s to open an L-PSIB in the \( m \)-bits long parallel network

DR scan – shift a \( d \)-bit deterministic pattern (check length) in the \( m \)-bits long parallel network

If unsuccessful – Run Instruction Register (IR) scan – switch to the serial network to shift next guess

\( (s+5) \) cycles

1 cycle DR to IR

\( (i+5) \) cycles

\( (m+5) \) cycles

\( (m+d+5) \) cycles

\( (m+d+5) \) cycles

1 cycle DR to IR

\( (i+5) \) cycles

Figure 5.4: Steps (clock cycles) required in an attempt to open an L-PSIB in a parallel network with its key logic in the serial network, using a guess pattern during a random brute force attack.
additional $m + d + 5$ clock cycles. If the attacker is still unsuccessful, he will need to once again switch to the serial network to begin a new guess. This requires $1 + i + 5$ clock cycles. The process can now begin again with a new key guess. Thus, the total cost of a guess is:

$$Cost_{\text{guess}} = (32 + s + 3 \times m + 2 \times d + 2 \times i) \text{ cycles}$$

Because the attacker is now able to deterministically try both a logic 1 and a logic 0 in the L-PSIB’s update cells (as shown in boxes 5 and 6 of Fig. 5.4), the expected number of random guesses on average is $2^k$ (assuming that the key size is much smaller than the serial scan path length and thus an attacker without detailed knowledge of the key locations cannot deterministically avoid repeatedly using the same key values in his guesses).

As shown in Table 5.1, this approach provides more security than the previous version due to the extra passes through IR-Scan and the greater likelihood of being able to hide the locations of the key bits. In this case, when $s = 256, n = 64,$ and $d = 10$, and the clock frequency is 10 MHz, the expected time to open an L-PSIB with a 56-bit key is approximately 118,000 years on average.

Table 5.1: Expected time to unlock an L-PSIB in a hybrid (serial and parallel) network. $s=256, n=64, d=10$, shift frequency=10MHz

<table>
<thead>
<tr>
<th>L-PSIB Key – j-bits</th>
<th>48</th>
<th>56</th>
<th>64</th>
<th>72</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time t years</strong></td>
<td>9.28E1</td>
<td>2.38E4</td>
<td>6.08E6</td>
<td>1.56E9</td>
<td>3.99E11</td>
</tr>
</tbody>
</table>
As discussed in Section 3.3, the IEEE Std 1687 describes two example broadcast networks. However, those networks broadcast the test data to all the instruments in the network. Thus, a careful decision has to be made during design regarding which instruments should be collected together in a broadcast group. A reconfigurable broadcast mode would give more flexibility during the design phase as well as during the actual testing phase. A network that provides reconfigurable broadcast (and daisy) modes and supports hierarchical broadcast networks and that has a low increase in the area overhead compared to a serial IJTAG network is desirable. The new broadcast network was designed keeping these design features in mind; using IJTAG also makes the architecture scalable. The new IJTAG broadcast network can be interfaced to the JTAG circuitry similar to the interfacing of IJTAG network shown in Fig. 2.1.

Some of the major features of this architecture are the following:

- Reconfigurable broadcast and daisy chain modes.

- Common test data can be broadcasted to any combination of instruments in the IJTAG network while the remaining instruments stay bypassed. SIBs do not change their open/close states during the broadcast mode.

- Switching the network from broadcast mode to daisy mode results in a fixed length daisy mode shift path configuration, each time. The resulting daisy chain configuration after exiting the broadcast mode does not depend on the prior broadcast mode configuration or the test data shifted during the last broadcast cycle.

- When this network is switched from broadcast to daisy mode, the resulting shift path
is of the shortest length possible in the network. All the SIBs in the network reset to their closed state when switching from broadcast to daisy mode. This reduces the amount of time required to reconfigure the shift path for the next daisy mode operation that follows the previous broadcast mode operation.

- The broadcast network architecture is scalable. During the design phase, the designer does not have to identify the groups of clients that would receive broadcast data.

- The new broadcast network architecture supports hierarchy while maintaining the above mentioned feature set.

- Similar to the traditional IJTAG network, the new broadcast network architecture can interface with the TAP controller. Thus it can be used in any IEEE 1149.1-compliant test architecture.

In the subsequent subsection we begin by describing our broadcast network design with single level of hierarchy. This is followed by design improvements required to support hierarchical broadcast networks.

### 6.1 Broadcast Network - Without Hierarchy

Fig. 6.1 shows a conceptual view of the new IJTAG broadcast network. The example network is part of a cluster which is divided into three partitions—A, B, and C. Each partition consists of a TDR and its corresponding SIB that is designed to support broadcast modes. Compared to a regular serial IJTAG network with flat-architecture [8,14], the design consists of an extra ScanMux per TDR. The broadcast_TDI signal can be routed along with other control signals.

#### 6.1.0.1 NIB as Partition Broadcast Control Bit (BCB)

The partition Broadcast Control Bit (BCB) at the cluster level is implemented using a NIB. The 1-bit partition BCB-Out signal from the partition BCB feeds the select signal of the scan-muxes in each partition. When the partition BCB asserts the BCB-Out signal, the network switches to broadcast scan mode. When the BCB-Out signal is de-asserted, the network switches back to daisy (or serial) scan mode. The BCB-Out signal from the
partition BCB is also applied to the modified SIBs in the network. With reference to the SIBs, the BCB-Out signal is referred to as SIB-Lock signal in this paper.

6.1.0.2 Broadcast Segment Insertion Bit

The modified SIB design used in the broadcast network of Fig. 6.1 is shown in Fig. 6.2. Compared to the SIB design in Fig. 2.3, the broadcast SIB has some additional features—a locked SIB state, a shift cell with a feedback path forced to zero during broadcast mode, and a pipelined UpdateEn. In terms of area overhead, the broadcast SIB requires two extra logic gates, and a pipeline flip-flop when compared to the SIB design in Fig. 2.3. These SIB design features that enable efficient broadcast mode operations are described subsequently.

Locked SIB State: As mentioned earlier, the partition BCB-Out signal also serves as the SIB-Lock signal for the broadcast SIB shown in Fig. 6.2. Based on value of the SIB-Lock
signal and the value in the update cell, the broadcast SIB shown in Fig. 6.2 can be in one of four different states. These four SIB states and the corresponding SIB-Lock and update cell values are given in Table 6.1.

During daisy (i.e. serial) mode, the SIB-Lock remains deasserted and the broadcast SIB works similarly to the SIB shown in Fig. 2.3. When the SIB is closed (update cell = 0), the corresponding TDR is bypassed. When the SIB is open (update cell = 1), the corresponding TDR is active and connected to the active shift path in daisy scan mode. Before switching to broadcast mode, in the last daisy shift operation each SIB is placed in an appropriate
Table 6.1: Broadcast SIB States

<table>
<thead>
<tr>
<th>SIB State</th>
<th>SIB_Lock</th>
<th>Update cell</th>
<th>TDR</th>
<th>Scan mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closed and unlocked</td>
<td>0</td>
<td>0</td>
<td>Bypass</td>
<td>Daisy_Byp</td>
</tr>
<tr>
<td>Open and unlocked</td>
<td>0</td>
<td>1</td>
<td>Active</td>
<td>Daisy</td>
</tr>
<tr>
<td>Closed and locked</td>
<td>1</td>
<td>0</td>
<td>Bypass</td>
<td>Broadcast_Byp</td>
</tr>
<tr>
<td>Open and locked</td>
<td>1</td>
<td>1</td>
<td>Active</td>
<td>Broadcast</td>
</tr>
</tbody>
</table>

open or closed state, depending on the TDRs that need to receive the identical test data during broadcast.

During broadcast mode the SIB-Lock signal remains asserted. This prevents the UpdateEn signal from reaching the update mux of the SIB and thus update operations cannot be performed on the SIB. In other words, during the broadcast mode, the SIB does not change its open/closed state. Without a lock, during the broadcast mode, all the broadcast SIBs will switch between their open/closed states. Once the target TDRs have received the broadcasted test data, the network can be switched back to daisy scan mode to reconfigure the SIBs. To reduce the number of clock cycles that would be required to switch from broadcast scan mode to daisy scan mode and reconfigure all the SIBs for the next broadcast/daisy operation, we made two more changes to the SIB design—during broadcast mode the shift cell’s feedback path is forced to zero and the UpdateEn signal to the SIB’s update mux is pipelined.

*Shift cell’s feedback path is forced to zero:* When the network is switched from broadcast to daisy mode, the SIBs that were open during the broadcast mode will add their corresponding TDRs to the active shift path in daisy scan mode. Thus, the resulting shift path will consist of all the TDRs that were receiving the broadcasted test data along with all of the broadcast SIBs in the network. Unfortunately, any network reconfiguration, after
switching from broadcast to daisy mode, would require shifting the new SIB configuration data through the TDRs connected to the active shift path. This would result in longer network reconfiguration overhead after each broadcast to daisy mode switch (depending on the number and length of the TDRs that were receiving the broadcasted data.)

To avoid this, the SIBs are designed such that the feedback path of the shift cell is forced to a zero during broadcast mode. Thus, when the broadcast mode ends and the SIBs are unlocked, an UpdateDR operation would close all the SIBs in the network resulting in minimal active shift path length. The network would now require fewer clock cycles to reconfigure it for the next daisy scan mode operation.

However, it is important to note that, the IJTAG standard did not consider these type of shift cell design modifications. Hence, an IJTAG compliant software might not be aware of the forced reset of the shift cells. Thus, to maintain compliance with the IJTAG Std. and the software, the last broadcast pattern should contain the broadcast SIB bit set to 0. Even though, explicitly setting this broadcast bit to 0 is not required for the circuit to function correctly. This is because, switching from broadcast mode to daisy mode will automatically reset all the broadcast SIBs. Another solution is to inform the software about the forced reset of the broadcast SIBs after switching from broadcast to daisy mode.

**Pipelined UpdateEn signal:** The network is switched from broadcast mode to daisy mode by shifting a zero in the BCB and performing an UpdateDR. This UpdateDR operation resets the BCB-Out/SIB-Lock signal, as shown by the red arrows in Fig. 6.3. As mentioned earlier, the shift cells of all the SIBs contain a zero at this point. Now, after applying the first UpdateDR to end the broadcast mode, a second UpdateDR will place a zero in the update cells of all the broadcast SIBs. This will close all the SIBs to produce a minimal length shift path. However, a second consecutive UpdateDR operation would require cycling through the TAP controller’s Capture-Shift-Update states, denoted as *TAP C* in the top half of Fig. 6.3.

To avoid these additional cycles, we have inserted a single flip-flop in the path of the UpdateEn signal of all the broadcast SIBs in the broadcast network. The first UpdateDR operation resets the BCB and deasserts the SIB-Lock signal. The corresponding timing diagram is shown in the bottom half of Fig. 6.3. Now, when the delayed UpdateEn signal
Figure 6.3: SIB unlock-update operations, with and without pipelined UpdateEn
from the UpdateDR operation reaches the broadcast SIBs in the next clock cycle, the SIB-lock signals for the broadcast SIBs are already zero—unlocking those SIBs. Thus, the delayed UpdateEn signal propagates to the update cell and allows the zero currently in the shift cell to be placed in the update cell on the next falling clock edge. This closes all the broadcast SIBs. As a result, a single UpdateDR operation resets the BCB and closes all the SIBs in two consecutive clock cycles. This, avoids the need to perform two consecutive UpdateDRs as required in a design without pipelined UpdateEn signals to the broadcast SIBs.

Since the TAP controller has to cycle through its state machine for the Capture-Shift-Update (CSU) signals, the delayed UpdateEn signal to the broadcast SIBs should be able to update the broadcast SIBs before any other CSU signals arrive at the broadcast SIB or the corresponding TDR.

6.2 Broadcast Network - With Hierarchy

Fig. 6.4 shows a conceptual view of the new IJTAG broadcast network with two levels of hierarchy. The example network is a part of the chip level test network, which is divided into three clusters—I, II, III. Each cluster may further consist of the cluster level broadcast network (partition BCB and partitions A, B, C) shown in Fig. 6.1 along with the additional broadcast SIB and scan mux per cluster as shown in Fig. 6.4. The broadcast SIBs (I, II, and III) in Fig. 6.4 operate in a similar manner as the broadcast SIBs shown in Fig. 6.1.

6.2.0.1 Partition Broadcast Control Bit (BCB) with locked state

For the hierarchical broadcast network to operate as intended, it is important to add locked states to the partition BCB of every cluster in the network. This is similar to the locked state of the broadcast SIBs we discussed earlier. Further optimizations similar to the broadcast SIB design, such as delayed UpdateEn and forced shift cell reset, can also be used.

6.2.0.2 Cluster Broadcast Control Bit (BCB)

The cluster BCB at the chip level is implemented using a NIB similar to the partition BCB that we showed earlier in Fig. 6.1. The control signal from the cluster BCB generates cluster BCB-Out and cluster SIB-Lock. The control signal also functions as the partition BCB-Lock for the partition BCBs in the clusters.
6.2.1 Test Time Improvements

The new broadcast network (shown in Fig. 6.1 without hierarchy and in Fig. 6.4 with hierarchy), provides a trade-off between network reconfigurability and SIB (and NIB) programming overhead. To evaluate this trade-off and test time improvements, we compare four IJTAG networks using 8 32-bit TDRs shown in Fig. 6.5—a serial IJTAG network in (a), prior broadcast networks in (b) and (c), and finally (d) based on our new broadcast network (from Fig. 6.1). These 8 TDRs are part of two embedded cores. Because the new network provides reconfigurable broadcast/daisy scan modes, these TDRs from the two embedded cores do not need to be grouped in clusters in Fig. 6.5d as compared to Fig. 6.5 (b), and (c). The total number of (SIBs+NIBs) in the networks shown in Fig. 6.5 (a), (b), (c), and (d) are 8, 4, 10, and 9 respectively.

The number of clock cycles required to perform read/write operations on the instrument
(a) Serial IJTAG network (flat-structure) with 8 TDRs of size 32-bits each

(b) IJTAG network with two clusters of broadcast networks based on Fig. 3.1

(c) IJTAG network with two clusters of broadcast networks based on Fig. 3.2

(d) IJTAG network based on the new broadcast network in Fig. 6.1

Figure 6.5: Sample IJTAG networks used in Table 6.2
Table 6.2: No. of clock cycles required for read/write of test data using four different IJTAG network designs.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Fig. 10a</th>
<th>Fig. 10b</th>
<th>Fig. 10c</th>
<th>Fig. 10d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daisy_all</td>
<td>282</td>
<td>274</td>
<td>398</td>
<td>284</td>
</tr>
<tr>
<td>Daisy_byp</td>
<td>154</td>
<td>274</td>
<td>210</td>
<td>156</td>
</tr>
<tr>
<td>Broadcast_all</td>
<td>282</td>
<td>86</td>
<td>101</td>
<td>53</td>
</tr>
<tr>
<td>Broadcast_byp</td>
<td>154</td>
<td>274</td>
<td>210</td>
<td>53</td>
</tr>
</tbody>
</table>

TDRs are reported in Table 6.2. The reported values are calculated using the same method that was used in [8,14]. For each of the test cases the number of clock cycles are reported assuming that the network was in reset state before the read/write operation. Thus, the initial state of all the SIBs and NIBs was assumed to be closed. The test cases were chosen to evaluate the networks in four different scan configurations—all 8 TDRs are in daisy scan mode (Daisy_all), some TDRs are in daisy scan mode and remaining are bypassed (Daisy_byp), all TDRs receive identical data (Broadcast_all), and finally some TDRs receive identical data and remaining are bypassed (Broadcast_byp).

**Daisy_all**: This test case involves reading out data from all the TDRs. As mentioned earlier, the IJTAG network in Fig. 6.5b has the lowest number of (SIBs+NIBs). Thus, it requires the lowest test time for a Daisy_all operation. However, it is important to note that the network clusters in Fig. 6.5b do not provide any reconfigurable daisy modes.

**Daisy_byp**: This test case involves reading out data from TDRs 1, 2, 5, and 6. The networks in Fig. 6.5 (a) and (d) can reconfigure the network in daisy mode such that only these four TDRs are active and the remaining TDRs are bypassed. The network in Fig. 6.5d contains one extra shift bit (BCB) in the chain compared to Fig. 6.5a. This extra bit needs to be shifted in the initial network reconfiguration step and while reading the TDRs 1, 2, 5, and 6. Thus, the new broadcast network requires two extra clock cycles compared to the serial network. The network in Fig. 6.5b, does not contain any SIBs within the broadcast clusters and thus it cannot reconfigure the active shift path during daisy scan mode. Also, the network in Fig. 6.5c does not support any daisy scan mode and can only access one TDR.
at a time. Thus, both of these prior broadcast networks require 42% and 25% more clock cycles, respectively.

Broadcast_all: This test case involves broadcasting data to all the TDRs. Now, even though the network in Fig. 6.5b has low SIB programming overhead, having two separate broadcast clusters results in 38% more test clock cycles compared to the new broadcast network in Fig. 6.5d. Also, in Fig. 6.5b, without clusters, spanning multiple TDRs from different embedded cores would result in very long daisy scan mode operations.

Broadcast_byp: This test case involves broadcasting test data to TDRs 1, 2, 5, and 6. In this scenario, our new broadcast network can broadcast the identical test data to TDRs 1, 2, 5, and 6 by using its reconfigurable broadcast mode while keeping the remaining TDRs bypassed. However, other networks do not provide this feature and will need to shift data in daisy scan mode. Thus, in this case our new network reduces test time by 65-80% clock cycles.
Chapter 7
NEW LSIB KEY LOGIC CIRCUITS AND ANALYSIS

In [7], key bit locations were assumed to be present in the shift cells of the scan chain. This gives an advantage to the attacker because the attacker can perform power analysis attack without needing to bring the JTAG state machine out of the Shift DR state. If the key bits are in the update cell, then the attacker would need to perform an Update DR after shifting in the guessed key bits. Thus, to increase the time that an attacker would need to investigate the network and the switching activity of the network with respect to various input patterns, it is better to place the key bits in the update cells as shown in Fig. 2.5.

It was shown in [7] that a key of length 48-bits can provide reasonable protection to the key logic circuit from brute force attacks based on the scan chain length and the scan shift frequency. For a scan chain length of 5000 bits and scan shift frequency of 10 MHz and no hierarchy (finding keys behind SIBs or LSIBs), the average amount of time required to find the correct 48-bit key through brute force attacks was shown to be in the order of thousands of years. Thus, in this paper, we chose to study keys 48 bits in length, although, our results are applicable to keys of other lengths as well.

Our experimental circuits consist of a 48-bit scan path segment where the update cell of every scan cell corresponds to the key bits of the LSIB key. The output of these update cells is fed to various key logic designs. These key logic designs are meant to check the key bit values in the updates cell against one or more than one possible correct keys. Each circuit was synthesized to a gate-level netlist using 90nm design libraries. The VCD (Value Change Dump) files generated after every simulation were processed to obtain the required switching activity data. We assume that the attacker is able to measure dynamic switching power corresponding to this switching activity of the circuit with good accuracy. We also assume the same switching power for both $1 \rightarrow 0$ and $0 \rightarrow 1$ transitions in the circuit. Finally, we assume that the update cells holding the key bits are not feeding data to any
other circuits. Thus the assumptions made for our analysis represent the “best case” scenario for the attacker.

To evaluate the susceptibility of key logic circuits to simple power analysis attacks we used two different sets of input patterns, referred to in this paper as “walking-1" and “walking-0" patterns. For an n-bit LSIB key $K_n$ each set of walking-1 and walking-0 patterns contains $n$ different input patterns, each $n$-bits in length. In a set of $n$ walking-1 patterns each pattern is of hamming weight one; the first input pattern has a single bit set to a one in bit location 0, second pattern in bit location 1 and so on. Similarly, the set of walking-0 patterns can be defined such that each pattern is a complement of the corresponding walking-1 pattern. Thus, for an LSIB with 48-bit unlock key, 48 patterns constituting walking-1 (walking-0) patterns were applied.

To measure the switching activity of the target circuit corresponding to a walking-1 (walking-0) pattern an attacker would need an appropriate background against which the attacker would measure the difference in switching activity. We used switching activity of the target circuit corresponding to an “all 0s” (or “all 1s”) pattern as a suitable background against which the switching activity of the circuit for each of the walking-1 (or walking-0) patterns was measured. Thus a pattern of all 0s (all 1s) was applied before applying each walking-1 (walking-0) pattern. We measured the cumulative switching activity of the key logic circuit for the clock cycle in which the circuit processed an input pattern. The attacker would be interested in the power trace generated due to this switching activity. We evaluate this switching activity data gathered using different attack patterns to assess the susceptibility of the key logic circuits to power analysis attacks.

7.1 Key Logic 1 – AND-tree

This implementation is just a simple array of AND gates as the name suggests. The array of AND gates is fed with the 48-bit data from the update cells holding the key bits. The inverted output (QN) of the flip-flops from the update cells, that are supposed to hold key bits whose correct value should be equal to a logic zero, is fed to the AND gate tree. Thus, if the correct value of the key bits is placed in the update cells then the AND tree will generate an Unlock signal. For an n-bit key of hamming weight $j$, there will be $j$ update
cells feeding their non-inverted output to the AND gate tree, and $n-j$ update cells feeding their inverted output to the AND gate tree. Fig. 7.1 shows an AND-tree based key logic for an $n$-bit LSIB key.

![Key logic 1 – AND-tree](figure)

The output of the update cells is applied to the AND gate tree in the same order as they appear in the scan chain. Although the figure shows AND gate tree implemented using two-input AND gates, AND gates of different fan-in sizes can be used as well.

We investigated susceptibility of this key logic circuit for LSIBs in an IEEE 1687 network by using 1000 different instances of the circuit. Each circuit has a unique 48-bit long key value that was randomly chosen with varying hamming distances. Switching activity of each circuit was collected using walking-1 patterns, resulting in 48 data points for the 48 bit locations of the LSIB key. Fig. 7.2 shows an example plot of the switching activity data obtained by applying walking-1 patterns to one of the AND-tree based key logic circuit.

We analyzed the switching activity data obtained using the walking-1 patterns to determine if it is possible for the attacker to deduce the embedded LSIB key. On comparing the switching activity values against each key bit value, following observations were made:

1. Pairs of the consecutive bit locations $[n-(m+1)]$ and $[n-(m+2)]$ in key Kn where $m$ ranges from $(0, 2, 4, \ldots, n-2)$, have same key bit value if the switching activity value
Figure 7.2: Switching activity values and the corresponding key bit values for each bit location of the LSIB key, obtained by applying walking-1 patterns. Hamming weight of the LSIB key = 21. Pattern reference background = all zeros. Circuit = Key Logic 1 – AND-tree for a single LSIB key.
for the two locations is equal.

(a) All such consecutive pairs of key bits which belong to the subset of bit locations with lowest switching activity value are set to a logic one in the LSIB key

(b) All remaining pairs are set to a logic zero in the LSIB key

2. Pairs of the consecutive bit locations \([n - (m + 1)]\) and \([n - (m + 2)]\) in key \(K_n\) where \(m\) ranges from \((0, 2, 4, \ldots, n - 2)\), have different key bit values if the switching activity values for the two locations are not equal. The bit location corresponding to the higher switching activity value is set to a logic zero and the bit location corresponding to the lower switching activity value is set to a logic one.

Using this correlation information between the switching activity data and the corresponding key bit for each bit location of the LSIB keys, we programmatically extracted correct value of the keys for all 1000 such key logic circuits implemented for single LSIB keys.

It is important to note that even though the AND gate tree in this example consisted of two-input AND gates, similar analysis can be done for an AND gate tree designed using gates with higher fan-in values. For example, if the AND-tree is implemented using 3-input AND gates, then similar analysis can be done by obtaining switching activity data using “walking-pair-of-1’s” patterns. In this case values of the key bits can be deduced by comparing switching activity value for three consecutive bit locations in the key.

7.2 Key Logic 1b – AND-tree with randomized key bit order

This key logic is a variation on the design of the Key Logic 1 – AND-tree described previously. In the previous design the switching activity values corresponding to the consecutive bit locations could be correlated to the correct values of the key bits. This is because, in the previous design the output of the update cells is applied to the AND gate tree in the same order as they appear in the scan chain. To prevent this relationship between the consecutive bit locations of the key, we modified the design by randomizing the order in which the output of the update cells holding the key bits is applied to the AND gate tree.

We investigated susceptibility to power analysis attacks of this modified key logic design
using the same 1000 keys as used in the previous investigation. Switching activity of each circuit was collected using the walking-1 patterns, resulting in total 48 data points for the 48 bit locations of the LSIB key. Fig. 7.3 shows an example plot of this switching activity data for AND-tree based key logic circuit with randomized key bit order.

Figure 7.3: Switching activity values and the corresponding key bit values for each bit location of the LSIB key, obtained by applying walking-1 patterns. Hamming weight of the LSIB key = 21. Reference background = all zeros. Circuit = Key Logic 1b – AND-tree with randomized key bit order, for one LSIB key.

We analyzed the switching activity data obtained using walking-1 patterns to determine if it is possible for the attacker to deduce the embedded LSIB key for this key logic design. The key bits were divided into separate sets based on their corresponding switching activity
values obtained. Key bits corresponding to equal switching activity values were placed in same groups, marked as group I, II, and III in Fig. 7.3. The key bit values were recovered using following procedure:

1. Group-I (pink box): Find key bit locations set to a logic zero in the LSIB key–In the walking-0 patterns, set one bit-location from Group-I to a logic zero and walk a zero through rest of the bit locations from Group-I. The pair of bit locations that produces highest switching activity value, during this walking-0 test, is set to a logic zero in the LSIB key. Repeat this step for all remaining bit locations in Group-I. During this process, bit locations set to a logic 1 in the LSIB key, will produce lower switching value than the original switching value of the bit locations in this group. Thus, key bit values for all the bit locations in the Group-I can be obtained.

2. Group-II (blue box): Find key bit locations set to a zero in the LSIB key–The process is same as Group-I.

3. Group-III (green box): Find key bit locations set to a logic one in the LSIB key–In the walking-1 patterns, set one bit-location from group-III to a logic one, and walk a one through rest of the bit locations in group-III. The pair of bit locations which produces highest switching activity, during this walking-1 test, is set to a logic one. Repeat this step for every bit location in group-III. During this process, bit locations set to a logic zero in the LSIB key will not change their switching activity value.

Thus, the correct key for this key logic circuit can be recovered when the key logic is implemented for a single LSIB key. Even though this investigation and analysis was done for key logic with two-input AND gates, the same procedure is also applicable for AND gates with higher fan-in values. For example, if 3-input AND gates are used, then using walking-pair-of-1’s and walking-pair-of-0’s patterns the LSIB key can be recovered by following the procedure just discussed.

7.3 Key Logic 1c – Gate-tree with Compiler Optimizations

The previous key logic circuits were designed manually and synthesized using a suitable standard cell library without any compiler optimizations for area or power. We re-synthesized
the 1000 circuits from “Key Logic 1a – AND-tree” using the same standard cell library while keeping the compiler optimizations enabled. An example synthesized circuit obtained using area optimizations is shown in Fig. 7.4.

![Diagram of Key Logic 1c – Compiler synthesized gate-tree. LSIB key length = 48-bits. Hamming weight = \( j \)](image.png)

We investigated susceptibility of this key logic design to power analysis attacks by using the same 1000 keys used in the previous investigations. Switching activity of each circuit was collected using both the walking-1 and walking-0 patterns. Fig. 7.5 shows an example plot of the data obtained by applying the walking-1 patterns.

From Fig. 7.5 it can be observed that an attacker can deduce the key bits that are set to a logic zero in the LSIB key by using the bit locations that show switching activity values higher than a certain threshold (marked with a dotted green line in this example). Similarly, using the switching data obtained by applying the walking-0 patterns, an attacker can deduce the key bits that are set to a logic one. Thus, by using the walking-1 and walking-0 patterns, an attacker can recover most of the key bit values. The bit locations that give the same switching activity for both the walking-1 and walking-0 patterns, can be recovered using brute force attacks quickly. The average coefficient of determination \( (R^2) \) between the switching activity values when walking-1 patterns are applied and the LSIB’s key bit values for this design is 0.54.
Figure 7.5: Switching activity values and the corresponding key bit values for each bit location of the LSIB key, obtained by applying walking-1 patterns. Hamming weight of the LSIB key = 21. Reference background = all zeros. Circuit = Key Logic 1c – Gate-tree with compiler optimizations, for one LSIB key.
7.4 Mitigate Power Analysis Attacks by Sharing Key Bit Flip-Flops Among Multiple LSIBs

The key logic circuits discussed until now were implemented to generate an Unlock signal for a single LSIB. If this approach is used, then every LSIB will have its separate key logic circuit that utilizes a separate set of update cells for key bits. Another way to implement this key logic based security for LSIBs is to share the same set of update cells (holding key bits) among key logic circuits of multiple LSIBs. Thus, this set of update cells feeds its output to multiple key logic circuits, or in other words – key logic circuits from multiple LSIBs share the same input key bit flip-flops.

This can also be considered as designing a single key logic circuit with multiple embedded LSIB keys that generates separate Unlock signals for each of the multiple LSIBs. This obviously reduces the number of update cells required to hold the key bits. Since output from the same set of update cells is fed to the key logic circuits of multiple LSIBs, placing any new key value in these update cells will cause the key logic circuits to switch simultaneously. Simultaneous switching of multiple key logic circuits will naturally obfuscate the switching activity of any individual key logic.

To evaluate susceptibility of the key logic circuits towards simple power analysis attacks, we implemented designs with four key logic circuits sharing the same key bit flip-flops. 1000 such circuit instances were generated with unique 48-bit LSIB keys of varying hamming weights and synthesized using standard cell library. Compiler optimizations flags similar to Key Logic 1c were enabled during synthesis. Switching activity data was gathered by applying the walking-1 patterns to all the 1000 instances of the circuits with four keys. Due to the simultaneous switching of multiple key logic circuits, the switching activity value for each walking-1 pattern depends on the bit values of all four LSIB keys at that binary location. We refer to the sum of bits at each of the 48 bit locations of the keys as “key-bitsum”.

An example bar plot with the switching activity data, obtained by applying walking-1 patterns to a circuit with four LSIB keys, plotted against the key-bitsum values for each key bit location is shown in Fig. 7.6. In this case the switching activity has no direct correlation to the key-bitsum values or the bit values for the individual LSIB keys. The average coefficient of determination ($R^2$) between the switching activity values and the key-bitsum values for
7.5 Key Logic with Programmable LSIB-Keys

The key logic circuits discussed in previous sections had their LSIB keys embedded in their design. This embedded key was effected due to the inverted and non-inverted outputs of the update cell flip-flops fed to the AND gate tree. If the LSIB key is same for every die of an IC, then it provides an advantage to the attacker. An attacker can obtain several
copies of the target IC and then implement a distributed guided brute force attack on them simultaneously to recover the LSIB key. This would reduce the time required to obtain the key by a huge factor, since the effort of applying all the generated guess key patterns will be divided among multiple ICs in time. To avoid this scenario, the LSIB keys should be made different for each die. Another option is to make those keys programmable through EEPROM. A key logic circuit that provides an option to use different LSIB keys for each die or use programmable keys through EEPROM is shown in Fig. 7.7.

![Figure 7.7: Key logic 3 – Key logic with programmable LSIB Unlock key. LSIB key length = 48-bits. Hamming weight = j](image)

7.6 Key Logic with Brute Force Attack Detection

Fig. 7.8 shows an example circuit that detects if a wrong LSIB key is placed in the update cells (update cells that hold key bits), and based on this detection prevents the UpdateEn signal to these Update cells and thus locking these Update cells. The UpdateEn signal to the key bit Update cells is disabled until a reset signal is applied to the device. It is important that an authorized user can use the IEEE 1687 network without applying a correct key (causing one of the LSIBs to open) each time. To avoid this, a redundant key logic circuit
is added to the design which shares the same input data (output from the key bit flip-flops) as the other four key logic circuits as show in Fig. 7.8. Thus, with this design an authorized user can use the IEEE 1687 network without inadvertently locking the Update cells holding the key bits by using the correct key for the redundant key logic.

![Figure 7.8: Circuit to detect brute force attack on LSIB key logics with shared key bit flip flops, and gate UpdateEn signal to the Update cells holding LSIB key bits. Key length = \( n \)-bits . Hamming weights of keys : LSIB1 = \( j \), LSIB2 = \( k \), LSIB3 = \( l \), LSIB4 = \( m \), redundant key = \( z \).](image)

The output of the key logic circuits sharing the same key bit flip flops is fed to an OR gate, and the output of the OR gate is used to gate the UpdateEn signal to the Update cells holding key bits. Thus, if any of the five correct keys (4 LSIB keys and one redundant key) is placed in the key bits, it will prevent the Update cells from getting locked. When an attacker applies wrong guess keys during a brute force attack, this design will lock the Update cells. Thus, forcing the attacker to reset the circuit each time resulting in much longer time for a successful brute force attack, including brute force attacks guided using the information.
obtained through power analysis attacks.
As discussed earlier, increasing SoC complexity has resulted in a dramatic increase in the number of scan flip-flops. Thus, resulting in a large test pattern volume. To reduce test times, modern DFT architectures support shifting test data in parallel to different SoC partitions. However, this comes at a cost. Shifting test data in parallel results in excessive simultaneous switching activity—well beyond the functional specifications of the design.

To reduce peak test power, an SoC can be divided into different blocks and tested separately [62]. One common solution is to include multiple clock domains, separate test clocks, and multiple free-running clocks in the SoC for testing [63,64]. A problem that arises here is that multiple test clocks also result in complex test-clock distribution designs. Each test clock would also require an extra package pin. As discussed in [65], adding extra package pin for each test clock is very difficult, especially in 2.5D/3D ICs [66,67].

One potential solution to the problem of using multiple test clocks without increasing extra package pin for each test clock is to apply a single test clock to the SoC and derive multiple test clocks inside each block [68]. However, if all the derived test clocks are in phase then it will result in the simultaneous toggling of all the clock domains. This will again cause a large number of scan flip-flops to toggle simultaneously when the test data is scanned in. Thus, resulting in a very high peak test power compared to the functional mode specifications. Higher than functional mode peak power will cause excessive Power Supply Noise (PSN), and potentially corrupt flip-flop states—resulting in test failures. This also results in circuit slowdowns during test.

The above mentioned potential problem that may arise due to the use of multiple test clocks (shift clocks in this case) that are derived from a single test clock can be resolved by staggering the derived clocks during shift cycles [65]. Thus if all of the derived shift clocks have different duty cycles and/or clock phases, the simultaneous switching of scan flip-flops
during shift can be reduced. The shift clocks of different phases derived from the single test clock are referred to as stagger values in the literature. These values are assigned to different SoC blocks such that no two adjacent blocks receive the same stagger value. Thus, the scan chains in any two adjacent blocks should not toggle simultaneously if the stagger value assignment is done correctly. A new problem that arise here is that the number of staggered values (derived shift clocks) is bounded due to the number of possible/available clock phases. However, an SoC may contain several hundreds of blocks/partitions. This problem of assigning small number of stagger values to a large number of SoC blocks to minimize toggle activity during test data scan was first studied in [65], and referred to as *Shift-Clock Stagger Assignment* (SCSA).

In [65], a heuristics based approach was proposed to perform faster but sub-optimal SCSA compared to their own mathematical model that produced optimal assignments and required higher computational times. However, this heuristics based method resulted in conflicts and in certain cases neighboring partitions received the same stagger values. Also, their mathematical model based programs did not run to completion in certain test cases involving larger designs with many partitions. In this chapter we discuss our own novel approach to solve the SCSA problem that provides an optimal assignment without requiring higher computational times compared to the methods proposed in in [65]. The remainder of this chapter is organized as follows. Section 8.1 discusses prior work related to shift-clock staggering and SCSA. Section 8.2 gives background information regarding map coloring and describes how SCSA problem is related to the map coloring problem. Section 8.3 discusses the new planar graph coloring algorithm for SCSA, and finally Section 8.4 presents experimental results.

### 8.1 Prior Work

As described in [65], SoC designs use a single input clock from the Automatic Test Equipment (ATE) as the shift clock. This ATE clock is applied to each block inside the SoC. A clock divider in each block generates local test clocks. Test patterns to the de-compressors and from the compressors of each SoC block are driven in series using the ATE clock. Inside each block, the scan chains between the de-compressor and compressor logic
are shifted using the locally generated clocks. Staggering of the shift clock at each block level was proposed in [65]. These stagger values should be generated locally in each block such that no two adjacent blocks use that same stagger value (conflict free). This is the Shift-Clock Stagger Assignment (SCSA) problem. Thus, no two adjacent blocks sharing the same power rail will toggle simultaneously when the decompressed test data is shifting through their internal scan chains. An example of the different staggered clock phases is shown in Fig. 8.1.

Previously proposed solutions used integer linear programming (ILP) to obtain optimal SCSA results. However, the assignment does not always produce conflict free results. These conflicts were resolved based on the length of the shared boundary between two adjacent blocks. If the shared boundary's length between two adjacent blocks was below a certain threshold then the algorithm proceeded with the assignment even if the two blocks received the same stagger value. Another drawback of this solution is that it takes large computational time to execute. In certain cases with very large designs (500 to 600 SoC blocks) and 0 threshold value the ILP method didn’t terminate when assigning 8 different stagger values. In medium sized designs (60 to 70 SoC blocks) with 0 threshold value, the ILP method finished executing. For example, SoC with 62 blocks resulted in 10 conflicts when assigning 4 different stagger values to the SoC blocks.

Another previously proposed solution is based on heuristics. This solution reduces CPU time by more than 3-4 times. However, it results in higher conflicts when assigning stag-
ger values as compared to the ILP method. SoC with 62 blocks and 0 threshold value resulted in 13 conflicts when assigning 4 different stagger values using the heuristics method. Also, because this solution is based on heuristics that involves choosing partitions randomly for SCSA, it produced a different SCSA result on each execution with different amount of conflicts. Thus, it would require to be executed a few times until an acceptable SCSA is obtained.

Thus, a solution that provides conflict free SCSA results without using excessive computation time is required. We designed a new planar graph coloring algorithm that provides guaranteed conflict free assignments when using 6 stagger values with 0 threshold. For medium to small sizes, our new algorithm can provide conflict free assignment using 4-5 stagger values and 0 threshold. In the subsequent sections we discuss some relevant background information regarding map coloring followed by the description of our new planar graph coloring algorithm.

8.2 Background: Planar Map Coloring

As discussed earlier, the SCSA problem involves assigning a limited number of the available stagger values to the SoC blocks such that no two adjacent blocks receive the same stagger value. This is similar to the classical map coloring problem in which different regions on a map are to be colored using a limited set of colors such that no two neighboring regions are assigned the same color. For the SCSA problem, the set of different stagger values is similar to the set of colors in the map coloring problem. Fig. 8.2 shows a colored partition level layout of an SoC. Here each color represents a different stagger value. In the rest of this text we will use the terms colors and stagger values interchangeably.

Planar map coloring problem has been studied by several generations of mathematicians and computer scientists. The most important planar map coloring theorem is called the four-color theorem. According to the four-color theorem: "The regions of any simple planar map can be colored with only four colors, in such a way that any two adjacent regions have different colors" [69–71]. However, SCSA problem does not limit to four colors and neither does it try to use the fewest possible colors (stagger values). For SCSA the challenge is use all available colors (stagger values) to color the map without conflicts. Another desirable
Figure 8.2: Colored partitions on an SoC – Colors here represent the stagger values used for Shift-Clock Stagger Assignment

feature would be to balance the usage of all the available colors. This is to make sure that an almost equal number of non-neighboring partitions on the SoC shift test data at the same time. Thus, this will also prevent a scenario where a single color has been used more than the other available colors causing excessive number of partitions to switch simultaneously during shift.

To solve the SCSA problem we have designed a new planar graph coloring algorithm. Before introducing the new algorithm we first discuss how a planar map relates to a planar graph. A graph consists of points (vertices) and lines (edges). The vertices on the graph are connected to each other using edges. Simply, a graph is considered to be planar if it’s possible to draw the graph such that no two edges cross over each other. Every planar map can be drawn as a planar graph using the concept of duality.

If every region on a planar map is drawn as a vertex and if the shared boundaries between any two regions on the map is represented using an edge between their two corresponding vertices then the resulting planar graph is the dual of the map. Now, any coloring operation
that’s done on the vertices of this planar graph will also be applicable to the original planar map. It is important to note that in our analysis we have also considered a shared point between two regions on the map as a valid shared boundary, and used it to draw edges while generating our planar graphs from maps.

The planar graph shown in Fig. 8.3 is obtained from the US map using the duality concept. The actual state names corresponding to the vertex labels used in Fig. 8.3 are given in Table 8.1. More details about this table are given in the following subsection.

8.2.1 Sequential Coloring

*Sequential coloring* is a popular method of solving graph coloring problems. Sequential coloring is a two step process. The first steps involves determining a specific coloring sequence of the vertices in the graph. The second step involves a greedy coloring process to color these ordered vertices. As discussed in [72], several sequential coloring algorithms such as Largest First (LF), Smallest Last (SL), and Saturation Largest First (SLF)/DSATUR have been proposed in the literature. Each of these algorithms provide different trade-offs between
run-time, memory requirement, and coloring results.

The Smallest Last (SL) coloring algorithm guarantees conflict free coloring of a planar graph in at most six colors [73,74]. Also, this algorithm can be further enhanced to provide a linear time 5-coloring of any planar graph [75]. The basic principle of SL coloring algorithm is that the vertices with few neighbors should be colored as late as possible. The SL algorithm optimally colors trees, cycles, unicyclic graphs, wheels, complete bipartite graphs, Johnson’s graphs and Mycielski’s graphs as reported in [72]. As discussed earlier, this sequential algorithm is also a two step process. The first step is to generate an ordered set of the vertices to determine the sequence in which they will be colored. Second step involves greedy coloring these ordered vertices. The ordering used here is called smallest-last ordering. This ordering makes sure that the vertices with the most potential to create conflicts during greedy coloring are colored first, and the vertices with fewer neighbors (thus, fewer conflicts) get colored as late as possible. Because the SL algorithm provides optimal coloring of planar graphs and uses at most 6 (or in certain cases 5), we chose smallest-last ordering for our new planar graph coloring algorithm. We now introduce the procedure for the smallest-last ordering of the vertices.

In a graph, the number of neighbors of a vertex is called the degree of the vertex. The smallest-last ordering involves sequentially “deleting” vertices from the graph starting with the vertex of the smallest degree. Each time a vertex is removed from the graph the degrees of all its neighbors are updated (decreased by 1). The deleted vertex is stored in a separate set. The process is terminated once all the vertices in the graph are deleted. At the end of this process, the set containing all the deleted vertices from the graph forms the ordered set of vertices used during coloring. This set represents the smallest-last ordering of the vertices.

As an example, in Table 8.1 we provide an ordered set of the vertices from the planar graph (of the US map) shown in Fig. 8.3. In this table, the order number denotes the ordering of the vertices. Thus, according to this table, the vertex ‘v42 Tennessee’ is first in the ordered set and the vertex ‘v11 Hawaii’ is last in the ordered set. Deletion degree (Del. Deg.) in the table denotes the degree of the vertex when it was being deleted from the graph. Thus, we can see from the table that the vertices with the smallest degrees (v11 Hawaii and v1 Alaska) were deleted first. Because these two vertices were not connected to any other
vertices in the graph, degrees of other vertices remained unchanged upon their deletion. This was followed by the deletion of the vertex 'v19 Maine' of the smallest degree in the graph, and upon its deletion the degree of its neighbor 'v29 New Hampshire' was decreased by one. This process was repeated until all the vertices in the graph were deleted and added to this table. In the next section we discuss our new planar graph coloring algorithm that uses the ordered set of vertices obtained using smallest-last ordering.

8.3 New Planar Graph Coloring Algorithm for Shift-Clock Stagger Assignment

The challenges associated with the graph coloring problem for Shift-Clock Stagger Assignment are a bit different than the problems associated with the classical sequential coloring problems. Graph coloring algorithms are designed to minimize the number of unique colors that are required to color the graph without conflicts. These algorithms are not bounded by hardware design constraints such as power. Also, these algorithm do not attempt to balance the number of disjoint vertices that are assigned the same color. Considering these limitations of the existing algorithms and challenges associated with hardware design we worked on a new graph coloring algorithm. These are some of the salient features of our algorithm when compared to the SL coloring algorithm and the existing SCSA algorithms in the literature:

- If provided by the user during execution, our algorithm uses all the available colors (stagger values) during the coloring procedure.

- The utilization of all the available colors (stagger values) is kept balanced. That is, if a color is assigned to \( m \) vertices in the graph, then the algorithm assigns each of the remaining other \( (n - 1) \) colors to \( (m \pm 1) \) vertices from the graph as well. Naturally, the \( \pm 1 \) value depends on whether the total number of vertices \( v \) in the graph is divisible by the total number of \( n \) colors or not.

- If the user does not provide the total number of available colors then the algorithm tries to minimize the number of unique colors required to color the graph. Thus, in accordance with the SL coloring algorithm this number is at most going to be 6 for planar graphs. However, the utilization of all the colors is still kept balanced.
Table 8.1: Smallest Last Ordering of the Vertices from the US Planar Graph in Fig. 8.3. Deletion degree (Del. Deg.) in the table shows the degree of the vertex when it was deleted from the graph during smallest-last ordering. Order number shows the order in which vertices get colored—from 0 to 50. DC is included; total vertices = 51.

<table>
<thead>
<tr>
<th>Order No.</th>
<th>Del Deg.</th>
<th>Vertex Label</th>
<th>Order No.</th>
<th>Del Deg.</th>
<th>Vertex Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
<td>v42 Tennessee</td>
<td>26</td>
<td>7</td>
<td>v43 Texas</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>v25 Missouri</td>
<td>27</td>
<td>6</td>
<td>v33 North Carolina</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>v17 Kentucky</td>
<td>28</td>
<td>7</td>
<td>v10 Georgia</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>v15 Iowa</td>
<td>29</td>
<td>6</td>
<td>v0 Alabama</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>v27 Nebraska</td>
<td>30</td>
<td>7</td>
<td>v24 Mississippi</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>v41 South Dakota</td>
<td>31</td>
<td>6</td>
<td>v18 Louisiana</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>v50 Wyoming</td>
<td>32</td>
<td>6</td>
<td>v22 Michigan</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>v5 Colorado</td>
<td>33</td>
<td>6</td>
<td>v34 North Dakota</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>v3 Arkansas</td>
<td>34</td>
<td>6</td>
<td>v37 Oregon</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>v36 Oklahoma</td>
<td>35</td>
<td>6</td>
<td>v4 California</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>v16 Kansas</td>
<td>36</td>
<td>6</td>
<td>v32 New York</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>v44 Utah</td>
<td>37</td>
<td>5</td>
<td>v30 New Jersey</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>v31 New Mexico</td>
<td>38</td>
<td>6</td>
<td>v8 Delaware</td>
</tr>
<tr>
<td>13</td>
<td>8</td>
<td>v2 Arizona</td>
<td>39</td>
<td>6</td>
<td>v21 Massachuset</td>
</tr>
<tr>
<td>14</td>
<td>8</td>
<td>v12 Idaho</td>
<td>40</td>
<td>5</td>
<td>v45 Vermont</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
<td>v28 Nevada</td>
<td>41</td>
<td>5</td>
<td>v29 New Hampshire</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>v13 Illinois</td>
<td>42</td>
<td>5</td>
<td>v6 Connecticut</td>
</tr>
<tr>
<td>17</td>
<td>8</td>
<td>v46 Virginia</td>
<td>43</td>
<td>4</td>
<td>v39 Rhode Island</td>
</tr>
<tr>
<td>18</td>
<td>7</td>
<td>v48 West Virginia</td>
<td>44</td>
<td>4</td>
<td>v40 South Carolina</td>
</tr>
<tr>
<td>19</td>
<td>7</td>
<td>v20 Maryland</td>
<td>45</td>
<td>4</td>
<td>v47 Washington</td>
</tr>
<tr>
<td>20</td>
<td>8</td>
<td>v38 Pennsylvania</td>
<td>46</td>
<td>4</td>
<td>v7 DC</td>
</tr>
<tr>
<td>21</td>
<td>8</td>
<td>v35 Ohio</td>
<td>47</td>
<td>4</td>
<td>v9 Florida</td>
</tr>
<tr>
<td>22</td>
<td>7</td>
<td>v26 Montana</td>
<td>48</td>
<td>2</td>
<td>v19 Maine</td>
</tr>
<tr>
<td>23</td>
<td>7</td>
<td>v14 Indiana</td>
<td>49</td>
<td>0</td>
<td>v1 Alaska</td>
</tr>
<tr>
<td>24</td>
<td>6</td>
<td>v23 Minnesota</td>
<td>50</td>
<td>0</td>
<td>v11 Hawaii</td>
</tr>
<tr>
<td>25</td>
<td>7</td>
<td>v49 Wisconsin</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• In some test cases our algorithm used fewer unique colors when compared to the SL algorithm.

• As compared to the SCSA method discussed in [65], our algorithm provides conflict free assignments when using 5-6 stagger values for larger (600+ partitions) designs and 4-5 stagger values for smaller (50+ partitions) designs.

During the coloring procedure, the set of unique colors is always ordered based on the number of times a color is assigned to the vertices in the graph. Based on this parameter the colors are arranged in an ascending order. Thus, the color that is assigned to the fewest vertices is first in this ordered set of colors and the color that is assigned to the most vertices is last in this order. We simply refer to this as the ordered set of colors in this chapter. Steps involved in the new planar graph coloring algorithm are as follows:

**Input:** Smallest-last Ordering of the vertices (ordered vertex set), ordered set of unique colors

**Output:** Colored planar graph with balanced utilization of all the colors

**while** Number of vertices in the ordered vertex set != 0 **do**

- Current vertex \( v \) = Pick the first vertex from the ordered set of vertices;
- Current color \( c \) = Pick the first color from the ordered set of colors;

  **while** True **do**
  - if If any neighbor of the current vertex \( v \) is colored with \( c \) already **then**
    - New current color \( c \) = Pick the next color from the ordered set of colors;
  - else
    - Assign the current vertex \( v \) with the picked color \( c \);
    - BREAK;
  - end

end

- Ordered set of colors = Re-order the set of colors based on the number of times each color is used for vertex coloring;
- Remove the current vertex \( v \) from the ordered set of vertices;

end

Figure 8.4: New Planar Graph Coloring Algorithm for Shift-Clock Stagger Assignment
The above algorithm can be modified to handle the condition where the set of colors are not provided. In this case the algorithm will add a new color to the set of unique colors each time when it cannot color a vertex with the existing set of colors without conflict. This change will be similar to the existing smallest-last coloring algorithm from the literature.

### 8.4 SCSA Results Using New Planar Graph Coloring Algorithm

As discussed earlier, the Shift-Clock Stagger Assignment problem for low power SoC testing is same as the map coloring problem with some additional hardware constraints. So to evaluate the efficiency of the new graph coloring algorithm we used the US map as a sample test case. The smallest-last ordering shown in Table 8.1 for the graph in Fig. 8.3 was provided as an input to the algorithm in Fig. 8.4. We ran the C++ implementation of our algorithm with the following two configurations: (a) zero colors provided by the user, and (b) 5 colors provided by the user. We also ran the C++ implementation of the smallest-last coloring algorithm [73,74] on this graph, this is discussed under case (c).
New graph coloring algorithm for SCSA

Case (a)

| 13 | 12 | 13 | 13 |

Case (b)

| 10 | 10 | 10 | 10 | 11 |

Smallest-Last Coloring from literature

Case (c)

| 19 | 13 | 14 | 5 |

Figure 8.6: Color utilization results for the US graph shown in Fig. 8.3 using our new graph coloring algorithm Fig. 8.4 (cases a and b), and using the smallest-last/SL coloring algorithm from literature (case c). Smallest-last ordering used for both the algorithms is given in Table 8.1. Case (a): No color set provided by the user. Case (b): New algorithm executed using 5 available colors. Case (c): SL coloring algorithm

Case (a): The algorithm ran to completion and required only 4 colors to color the entire map. This is optimal coloring and in accordance with the four-color theorem discussed earlier. The coloring result for this case is shown in Fig. 8.5. As we can see in Fig. 8.5, the graph is optimally colored using four-colors without any conflicts. Colors red, green, and orange were used to color 13 vertices each. Blue color was used to assign colors to 12 vertices on the graph. This is because the total number of 51 vertices is not completely divisible by 4 available colors.

Thus, the utilization of all the four colors (red, blue, green, and orange) is balanced and optimal. Color utilization for this test case is also shown in Fig. 8.6.

Case (b): The program was executed again but this time the user provided 5 total available colors as an input along with the ordered vertex set shown in Table 8.1. Color utilization for this test case is shown in Fig. 8.6. As we can see in Fig. 8.6, for case (b), utilization of all the 5 colors (red, blue, green, orange, and brown) is balanced.

Case (c): In this test the US graph in Fig. 8.3 was colored using the smallest-last (SL)
The SL coloring algorithm uses greedy coloring and picks the first available color to color a vertex. This results in the over-utilization of the first color in the set of colors and the utilization gradually decreases for the other colors in the set. Thus, as seen in Fig. 8.6, for case (c), the red color (first color in the set) is used 19 times, followed by the blue, green, and orange colors utilized for 13, 14, and 5 times respectively. If this algorithm is used for the Shift-Clock Stagger Assignment problem, then a much higher number of partitions will shift test data for the first stagger value and very few partitions will shift test data using the fourth stagger value. Even though this algorithm resolves SCSA conflicts and prevents neighboring partitions from shifting test data at the same time, it does not balance the overall switching activity during shift operations across the SoC partitions.
Chapter 9
CONCLUSION

This research focused on improving SoC test networks for bandwidth, security, and power. We explored IJTAG test network designs for bandwidth and security improvements. To reduce peak power consumption during test we also proposed a novel graph coloring algorithm for Shift-Clock Stagger Assignment. In general, this algorithm also provides some useful feature enhancements over classical sequential graph coloring algorithms from the literature.

For bandwidth improvements, we have introduced multiple types of parallel SIBs and parallel IJTAG networks optimized for bandwidth, area, and/or security. We have shown that parallel SIB-based networks can significantly increase the test bandwidth with no increase in the SIB programming overhead (i.e. the clock cycles needed to shift through closed SIBs while accessing an instrument). Additional overhead can be saved by reducing the number of Update cells used for the Parallel-SIB design to 1. When test I/O is limited, a broadcast mode with addressable SIBs can add significant test bandwidth and lower SIB overhead at the cost of disjoint instrument access.

Furthermore, for bandwidth improvements, we introduced a new IJTAG based broadcast network architecture to improve the test data write times for multiple identical embedded cores. We have shown that the new broadcast network is scalable and it does not require careful design time considerations regarding the instruments that should be collected together in a broadcast group. Our Broadcast Control Bit (BCB) and modified broadcast SIB designs enable efficient hierarchical broadcast networks. We made further optimizations to our broadcast SIB design to reduce the overhead required to program the SIBs and reconfigure the network when switching from broadcast to daisy modes. Compared to a serial IJTAG network, our design requires only one additional scan-mux, two extra logic gates and an extra delay flip-flop per TDR in the network. In our sample test scenario we showed that by making use of its reconfigurable broadcast mode our broadcast network dramati-
cally reduced test time up to 65-80% clock cycles compared to IJTAG based serial and prior broadcast networks.

The security of parallel IJTAG networks protected by locking parallel SIBs was also investigated. Although one might intuitively think that security would be compromised with the ability to scan key data into the network in parallel, the number of guesses required still grows exponentially with key size. Shorter chain lengths in the parallel case (and thus a smaller cost of a guess) when some of the SIBs are open can be counteracted with slightly larger keys. In addition, when all of the SIBs are closed, the chain lengths of both networks are identical, and the cost of a guess is thus the same as well. However, more cells are active on the chain in the parallel case, providing more possible key bit locations when all SIBs are closed. We showed that a 56-bit key in a 256-bit long parallel scan chain shifted at 10 MHz requires over 60,000 years on average for a successful brute force attack. Distributing the keys and LSIBs across a parallel hybrid network increases the time required even further to 118,000 years on average.

We also showed how combinational key logic circuit for a single LSIB may be defeated with simple power analysis attacks. However, sharing key bits among multiple LSIBs can obfuscate the power signature of the individual LSIB keys while reducing the number of key bit holding flip-flops required. Specifically, we showed that combining 4 LSIB keys into a single key logic circuit that would generate 4 separate Unlock signals for 4 different LSIBs provides good obfuscation of the switching data (avg. $R^2$ of 0.05) and reduces the required number of key bit Update cells by a factor of 4, as compared to 4 separate key logic circuits designed for a single LSIB (avg. $R^2$ of 0.54). These changes can be easily implemented and evaluated by a designer at an early design phase. In the future, we will explore methods such as differential power analysis attacks on the LSIB key logic.

Finally, to reduce peak test power (due to the simultaneous switching of scan flip-flops in neighboring partitions) while shifting parallel test data through the internal scan chains of multiple SoC partitions, we worked on a novel solution for the Shift-Clock Stagger Assignment (SCSA) problem. We identified the SCSA problem as a map coloring problem and designed a novel planar graph coloring algorithm to solve it. Our algorithm provides several improvements over the existing SCSA solutions in the literature. Compared to the existing
solutions, our algorithm provides guaranteed conflict free assignments using $6/5$ stagger values (colors) regardless of the number of partitions in the SoC layout. The utilization of the stagger values is balanced. Thus, it also results in an equal number of partitions switching during scan shift. In some cases, our algorithm requires fewer unique colors to color a planar graph compared to the smallest-last coloring algorithm in the literature. Also, our algorithm can accept an arbitrary number of unique colors (less than or equal to the number of vertices to be colored), and utilize all of them in a balanced manner.

The bandwidth, security, and peak test power improvements and designs discussed in this dissertation are not interdependent on each other. Almost all of the proposed solutions can be independently implemented in an SoC test design. However, these solutions are not mutually exclusive either. If required, all or a combination of the designs/solutions from this work can be implemented in an SoC design. For example, the new IJTAG based broadcast design can be implemented at chip/partition level. Test network security can be improved using the secure IJTAG mechanisms discussed in this work. Finally, peak test power consumption while shifting test data in parallel to multiple SoC partitions can be reduced using our novel graph coloring algorithm for SCSA.


