Scan Chaining And Test Scheduling In An Integrated Scan Design System

Sridhar Narayanan

CENG Technical Report 94-33

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California 90089-2562
(213)740-4469

December 1994

SCAN CHAINING AND TEST SCHEDULING IN AN INTEGRATED SCAN DESIGN SYSTEM

by

Sridhar Narayanan

A Dissertation Presented to the

FACULTY OF THE GRADUATE SCHOOL

UNIVERSITY OF SOUTHERN CALIFORNIA

In Partial Fulfillment of the

Requirements for the Degree

DOCTOR OF PHILOSOPHY

(Electrical Engineering)

December 1994

Copyright 1994 Sridhar Narayanan

Dedication

To my wonderful parents for their undying love, patience and support.

Acknowledgements

It is with a mixed sense of pride and relief that I pen these words of gratitude to the people who have greatly helped me in completing this dissertation. First and foremost, I would like to express my sincere thanks to my advisor and committee chairman, Professor Melvin Breuer. His invaluable experience and insight greatly enhanced the quality of my work. In addition his constant encouragement and guidance motivated me to perform to the best of my abilities. Incidentally he has also inculcated in me a healthy respect for the use of commas as will be evidenced by the paucity of such punctuation in this document. I have greatly benefited from my numerous discussions with Dr. Sandeep Gupta on a wide range of topics ranging from delay testing to the OJ Simpson case. I would like to thank him for his useful suggestions and critique of my work, and for taking the time and trouble to plow through this thesis. I would also like to thank Dr. Ming-Deh Huang, Dr. Lloyd Welch and Dr. Massoud Pedram for serving on my dissertation and qualifying committees. My heated discussions with Dr. Sarma Sastry always provided me with fresh perspectives on graduate life; I thank him for his enthusiasm and personal interest in my work.

In my years at USC the interaction and feedback from the different members of the Test Group have always been fruitful and extremely useful. In particular I would like to thank the following people for their support and friendship during my tenure as a graduate student; Rajesh, Charlie, Deb, Gopal, Ishwar, Rajeshwari, Lempel, Lin, Amit, Rajiv, Shin-lee and Ping-Xu. The wonderful office staff in the department has always ensured the smooth flow of any non-technical matters related to my graduate study. I would like to express my sincere appreciation to all of them and in particular to Mary Zittercob and Lucille Stivers for their patience and kindness. With regard to funding through the course of my graduate life, I would like to acknowledge both the fellowship provided by the Graduate School at

USC, and financial support provided by the Defense Advanced Research Project Agency through Contract No. JFBI90092 (monitored by the Federal Bureau of Investigation).

The travails of a doctorate extend beyond the frontiers of the university and do in fact impinge quite a bit on one's personal life. In this regard, I have been most fortunate in having a bunch of extremely considerate and interesting friends. I will refrain from explicitly listing names (for fear of missing out anyone) but just register my heartfelt gratitude to all of them for being an integral part of my life. Throughout these fun-filled and occasionally frustrating years of graduate life, the one constancy has been my friend Ram. His fervor and enthusiasm for life, and his friendship and advice have gone a long way in inspiring me. The days we spent together here at USC will be ones I will cherish for a long time to come.

Contents

Dedication ii

Acknowledgements iii

List Of Tables x

List Of Figures xii

Abstract xv

1 Introduction 1

1.1 Overview of Scan Design 1

1.2 Scan Design Costs 3

1.3 Motivation and Goals 5

1.4 Outline of Thesis 7

2 Background 9

2.1 Compact Test Sets 9

2.2 Combining Test Generation/Application 10

2.3 Scan Chaining 11

2.3.1 Single and Multiple Scan Chains 11

2.3.2 Reconfiguring the Scan Chain 13

2.4 Test Scheduling 14

3 Overview of Scan Chaining 17

3.1 Introduction 17

3.2 Circuit Model 18

3.3 Test Application Schemes 22
3.3.1 Combined and Overlapped schemes 22
3.3.2 Flush and Minimum Shift Policies 24
3.4 A Classification of the Chain Methodologies 26
3.5 Reducing Test Time and Test Volume 30
3.6 Summary 32

4 Ordering Registers in a Single Scan Chain **33**
4.1 Introduction 33
4.2 Configuring the Scan Chain 34
4.3 Model of the Optimization Problem 36
4.4 Analysis of Optimal Configurations 39
4.4.1 Complexity Results 39
4.4.2 Restricting the Search Space 41
4.5 An Implicit Enumeration Algorithm 47
4.5.1 Bounds on Test Time 47
4.5.2 Details of the Algorithm 49
4.5.3 Incorporating Routing Constraints 52
4.6 Case Study 53
4.7 Summary 56

5 Multiple Scan Chains **58**
5.1 Introduction 58
5.2 Main Concepts 59
5.2.1 Test Application with Multiple Chains 60
5.2.2 Equal Length and Optimal Chains 62
5.3 Model of the Problem 64
5.4 Restricting the Search Space 67
5.5 A Polynomial Time Algorithm 70
5.5.1 Properties of Optimal Configurations 70
5.5.2 Dynamic Programming Technique 73
5.5.3 Complexity Analysis 77
5.6 Experimental Results 77
5.6.1 Influence of Design Characteristics 77

5.6.2 Case Study 80
5.7 Extensions to the Algorithm 81
5.7.1 Routing Area Constraints 82
5.7.2 Unequal Length Registers 82
5.7.3 Minimum Shift Policy 84
5.8 Summary 87

6 A Reconfigurable Scan Chain **88**
6.1 Introduction 88
6.2 Main Concepts 89
6.2.1 Reconfiguring a Single Scan Chain 89
6.2.2 Tradeoffs in Test time and Area Overhead 91
6.3 Optimal Scan Chain Configurations 94
6.3.1 Model of Optimization Problem 94
6.3.2 Statement of Optimization Problems 95
6.4 Flush Shift Policy 98
6.5 Minimum Shift Policy 102
6.5.1 Problem Analysis 102
6.5.2 Placement of Bypass Multiplexers 104
6.5.3 Ordering the Registers 108
6.5.3.1 Restricting the Search Space 108
6.5.3.2 Neighborhood Search Technique 110
6.6 Experimental Results 114
6.6.1 Case Studies 114
6.6.2 Influence of design characteristics 118
6.7 Summary 124

7 Asynchronous Multiple Scan Chains **125**
7.1 Introduction 125
7.2 Main Concepts 127
7.2.1 A Comparison of Chain Configurations 127
7.2.2 Use of Asynchronous Chains 130
7.3 Two-Phase Test Application Scheme 132
7.4 Model of Problem 135

7.5 Configuring the Scan Chains 136
7.6 Enhancing the Test Application Scheme 140
7.7 Experimental Results 143
7.8 Increasing the Number of Mode Signals 146
7.9 Summary 149

8 Test Scheduling for Scan Designs **150**
8.1 Introduction 150
8.2 A Generalized Circuit Model 152
8.3 Comparing BIST and scan scheduling 155
8.4 A Framework for Test Scheduling 157
8.5 Classification of Scheduling Techniques 159
8.6 Minimum Shift Policy 164
8.6.1 Importance of the scan chain configuration 164
8.6.2 Configuring the Scan Chain 170
8.6.3 Iterating between scheduling and chaining 174
8.7 Experimental Results 176
8.7.1 A Case Study 176
8.7.2 Discussion 177
8.8 Summary 181

9 Integrating to a Scan Design System **182**
9.1 Introduction 182
9.2 System Overview 183
9.3 Design Estimators 188
9.3.1 Motivation 188
9.3.2 Characteristics of the Design Estimators 190
9.3.3 Estimating the Test Application Time 191
9.4 Experimental Results 202
9.5 Summary 206

10 Conclusion and Future Work **208**
10.1 Scan Chain Methodologies 208
10.1.1 Configuring a Single Scan Chain 209

10.1.2 Multiple scan chains 210
10.1.3 A Reconfigurable Scan Chain 211
10.2 Scan Chaining and Test Scheduling 212
10.2.1 Asynchronous Multiple Scan Chains 213
10.2.2 Test Scheduling for Scan Designs 214
10.3 Integrating to a Scan Design System 215

Appendix **217**

Reference List **221**

List Of Tables

3.1 Summary of test sessions with minimum shift policy 25

4.1 Seven scan designs of the *USC Data Path-I* 55
4.2 Results for scan designs of the *USC Data Path-I* 56

5.1 Summary of test sessions for example design 62
5.2 Three Variations of Circuit 2 79
5.3 Comparison of test times with multiple scan chains 79
5.4 Test times for three scan designs of the *USC Data Path-I* 81

6.1 Summary of test sessions for both shift policies 91
6.2 Eight scan designs of *USC Data Path-I* and *AR-Filter Element* 116
6.3 Results for eight scan designs (Flush policy) 117
6.4 Neighborhood search with five seeds (Minimum shift policy) 119
6.5 Results for eight scan designs (Minimum shift policy) 120
6.6 Five sets of values for f_i and γ_i 121

7.1 Comparing multiple scan chain methodologies - Cases 1 and 2 144
7.2 Comparing multiple scan chain methodologies - Case 3 145
7.3 Results for a full scan design of the *USC Data Path-I* 147

8.1 Test schedules classified by scheduling discipline and shift policy . . . 157
8.2 Optimal schedule with parallel testing for both shift policies 162
8.3 Summary of test schedule for different order of scan registers 167
8.4 Parameters for scan registers in example design 171
8.5 Partial test schedule used to configure scan chain 172
8.6 Optimal test times for different scheduling techniques 178

9.1 Upper and lower bounds on test time for chain methodologies 201
9.2 Nine scan designs of the *USC Data Path-I* 203
9.3 Design costs for range of chain methodologies 207

List Of Figures

2.1 Division of the scan chain into high frequency and low frequency groups 14

3.1 (a) Unpartitioned scan design. (b) Partitioned design with disjoint kernels. (c) Partitioned design with non-disjoint kernels. 19

3.2 Example design with a single scan chain. 22

3.3 Classification of chain methodologies. 27

3.4 (a) Multiple scan chains.(b) Reconfigurable scan chain. 29

4.1 (a) Example scan design. (b) Optimal scan chain. (c) Worst-case ordering of registers. 35

4.2 Scan chain configuration with register parameters. 37

4.3 (a) Circuit topology with pure drivers and receivers. (b) Design with independent finite state machines. (c) Pipelined circuit topology. . . . 43

4.4 (a) Precedence graph for example design. (b) Precedence graph assuming equal length registers. 46

4.5 Search tree organization for example design. 50

4.6 *USC Data Path-I*. 54

5.1 (a) Example design. (b) A four-chain configuration for the design. . . 61

5.2 Two different chain configurations for example design. (a) Equal length chains (test time = 4508 clock cycles). (b) Optimal four-chain configuration (test time = 3312 clock cycles). 63

5.3 (a) Four-chain configuration with flip-flop and chain parameters. (b) Transformed configuration. 66

5.4 (a) Sorted list of 10 flip-flops. (b) Optimal three-chain configuration. (c) Dynamic programming table. 74

5.5 Circuit 2. 78

5.6 (a) Example design. (b) Optimal 2-chain configuration with flush policy. (c) Optimal 2-chain configuration for the minimum shift policy. 85

6.1 (a) Example design.(b) Reconfigured scan chain. 90
6.2 Optimally reconfigured chains for (a) flush policy, (b) minimum shift policy. 92
6.3 (a) Parameters of registers. (b) Transformed configuration. 94
6.4 (a) Optimal configuration with 2 multiplexers as generated by algorithm. (b) Alternate optimal solution with 2 multiplexers. 101
6.5 Optimal configurations for the minimum shift policy. (a) no multiplexer, (b) a single multiplexer, and (c) two multiplexers. 103
6.6 Maximal register sequences for scan chain ordering. 105
6.7 Neighborhood search with deterministic seed. 113
6.8 Portion of *AR-Filter Element*. 115
6.9 Savings in test time with the flush policy. 122
6.10 Savings in test time with the minimum shift policy. 123

7.1 (a) Scan flip-flop design. (b) Circuit with two finite state machines. . 126
7.2 Comparing multiple chain configurations. 129
7.3 Example circuit design. 130
7.4 (a) Circuit example. (b) Four-chain configuration with two chain groups. 133
7.5 Optimal asynchronous four chain configuration. 134
7.6 Two chain asynchronous configuration. 141

8.1 (a) Example scan design. (b) Internal structure of design. 153
8.2 (a) TCG of example scan design. (b) Modified TCG for minimum shift policy. 154
8.3 Optimal test times versus frequency of occurrence for 1000 orderings. 165
8.4 (a) Modified scan chain for design. (b) TCG of design. 166
8.5 (a) Precedence graph independent of partial schedule. (b) Modified precedence graph based on partial schedule. 172
8.6 (a) Example design with multiplexer. (b) Example design with a fanout point. 179

9.1 Flowchart of the SIESTA scan design system. 184
9.2 Search tree model of design space. 189
9.3 (a) Partial design of two compatible kernels. (b) Partial design of four incompatible kernels. (c) TCG of design. (d) TIG of design. 192

9.4 Area-test time tradeoffs with the combined scheme. 204
9.5 Area-test time tradeoffs with an optimal single scan chain. 205

Abstract

Scan design techniques reduce the complexity of test generation for sequential circuits. This is done by modifying the storage elements to act as a shift register in test mode. However a major drawback of scan design is the large test application times incurred in shifting test data in and out of a circuit. Reducing test application time can decrease the overall costs of testing and lead to shorter design cycles. The primary objective of this thesis is to minimize the test application overheads for partitioned serial scan designs. To achieve this we focus on the two main issues of scan path chaining and test scheduling. An optimal test application scheme is derived for a particular class of scan designs. Based on this scheme and the concept of a *shift policy*, we study different ways to organize the scan registers. With a single scan chain we show how an effective *ordering* of registers in the chain can lead to large reductions in test time. The popular approach to *multiple scan chaining* is to make all chains have equal length. We allow the chains to be of different lengths and show that this can lead to lower test times. A novel scan chain organization based on a *reconfigurable* chain is investigated in this work. Multiplexers are used to bypass segments of a scan chain to reduce the total shifting time. By enhancing the test control strategy in using multiple chains, we introduce the concept of *asynchronous* multiple chains. For each technique we develop optimal chaining algorithms and provide ways to tradeoff test time with design costs such as pin count, routing and logic overheads, and test control complexity. For a more general class of designs, a two-phase approach is required to minimize the total test time. In concert with an organization of the scan registers, efficient test scheduling strategies are developed to exploit parallelism in the test process. The integration of the various techniques to a scan design system provides a framework to analyze design tradeoffs in incorporating scan testability. A designer is provided with a range of options to efficiently traverse the design space and generate optimized testability solutions.

Chapter 1

Introduction

With the rapid growth in the number of circuits fabricated on a single device, the cost of testing becomes significant and occupies an increasing proportion of the total cost of design and fabrication. In progressing up the design hierarchy from chips to boards and then onto systems, testability can degrade unless careful attention is paid to every aspect of testing at all levels of the design. In this context the importance of Design-for-Testability (DFT) techniques cannot be over-emphasized [1]. Scan design encompasses a range of structured DFT techniques that ease the problem of test generation for sequential circuits by modifying some or all storage elements in the circuit. Every scan methodology will increase the costs of certain aspects of a design while decreasing other costs. For example the costs incurred in the area overhead of full scan designs will decrease the costs of debugging and diagnosing circuit failures. The main objective of our research is in developing an integrated scan design system that provides for efficient exploration of the design search space while optimizing the associated design costs. In the following sections, we provide a brief overview of scan design techniques and discuss the associated design costs.

1.1 Overview of Scan Design

DFT techniques were originally conceived to reduce the cost of testing by introducing testability criteria early in the design cycle. In particular structured DFT techniques provide well-defined design rules to enhance the *controllability* and *observability* of

internal nodes in a circuit. These testability attributes are useful in reducing the costs of deterministic test generation. Examples of structured DFT techniques are Level-Sensitive Scan Design (LSSD) [2], Scan Path [3] and Built-in Logic Block Observation (BILBO) [4]. The latter is an example of a built-in DFT technique in which test data is both generated and analyzed on the chip by appropriately modifying registers to act as pattern generators and signature analyzers. On the other hand, scan DFT techniques such as LSSD and Scan Path require test data to be stored outside the circuit and shifted in through one or more I/O pins. Our main emphasis in this research is on the various issues and costs as related to scan DFT techniques.

Scan design techniques reduce the complexity of test generation for sequential circuits by modifying the storage elements (latches or flip-flops) to act as observation and/or control points during the test process. This is normally done by adding logic to the storage elements to permit them to be connected into one or more shift chains. We will use the term *scan flip-flop* to refer to a storage element modified in this way. Test data can be shifted in and out of the scan flip-flops through I/O pins that provide access to the ends of the shift chains. A large number of variations on this basic idea has been proposed and investigated in the literature. Some of the criteria used in classifying this wide range of scan methodologies are presented below.

- *Selection of scan flip-flops* - In full scan designs all storage elements are modified and connected into shift chains. This reduces the test generation problem for sequential circuits to a purely combinational one. Note that the added logic in modifying all storage elements increases the total area and might affect system performance through the introduction of additional circuit delays. Partial scan techniques attempt to reduce the area overhead and performance degradation associated with full scan by only selecting a subset of the storage elements to be made scannable [5, 6, 7, 8]. However the complexity of test generation on the resultant circuit depends on both the number and choice of storage elements that are scanned.

- *Organization of storage elements* - The standard approach to organizing the scan flip-flops assumes the form of one or more scan chains. The scan flip-flops in each chain are accessed by means of scan-in and scan-out pins which form

the end-points of the chain. This requires test data to be serially shifted in and out of the circuit. A number of non-serial approaches to organizing the storage cells have been investigated. For example the scan flip-flops can be arranged as a random-access bit-addressable memory [9]. Similarly in a technique termed as *parallel scan*, every scan element is provided with its own scan chain [10]. In both cases every scan flip-flop can be controlled and observed in a single clock cycle. However these non-serial scan approaches lead to large area overheads and excessive pin requirements.

- *Fault model* - The classical single stuck-at fault has been the primary fault model used in both combinational and sequential test generation. A single test pattern applied at the primary input and present state lines of a sequential circuit is sufficient to sensitize and propagate a fault to either the primary outputs or next state lines. However the increased demand for quality in VLSI designs is motivating the use of more comprehensive fault models such as transistor stuck-open and delay faults. The requirement of a two-pattern test to detect such faults increases the complexity of test generation and adds an entirely new perspective to the scan design problem [11].

- *Design of storage cells* - The design of the modified storage element is a critical aspect of a scan design since it can have a significant effect on the area overheads and timing behavior of the final design. A range of designs that employ different types of storage elements can be found in [1]. In addition two-pattern testing might require the use of more complex scan cell designs that can simultaneously store two bits of test data [12].

1.2 Scan Design Costs

The incorporation of scan testability will increase the cost of a design in several ways. It is important to reduce these design costs as much as possible without compromising on the quality of the testing process. We provide a list of the main design costs incurred in using serial scan DFT techniques.

- **Area overhead** - The added logic in modifying the storage elements will lead to an increase in the circuit area. The actual increase will depend on both the design of the scan cell and the number of modified storage elements. In configuring the scan chain(s), the routing overheads and additional logic in the form of multiplexers will also contribute to the total circuit area. Additional hardware might also be required to control and clock the scan chains in the test mode.

- **Performance degradation** - Additional delays are introduced at the inputs to scan storage elements and this might cause a degradation in the circuit's performance under normal operation. A slower system clock rate may be required if the selected scan elements lie in critical timing paths.

- **Test application overheads** - The use of scan design techniques leads to high test application costs because test data (both input patterns and output results) needs to be serially shifted in and out of a circuit. In addition to the storage requirements for test data, the large test application times can contribute to the overall cost of testing. In the next section we will elaborate on the importance of reducing both these overheads.

- **I/O pin count** - To access the scan path(s), pins used for scan-in/scan-out purposes are required on the chip package. In addition, one or more pins will also be needed for control signals such as a test clock or a test mode line. Note that additional area overheads will be incurred in the routing of the test mode lines and test clocks, and if the pins are multiplexed with existing system pins.

- **Test generation complexity** - In full scan techniques the test generation problem is reduced to that of test generation for combinational logic. Test pattern generation for combinational circuits can be performed more efficiently as compared to sequential circuits. The complexity of test generation is thus an important component of the costs associated with partial scan techniques. In spite of being a non-recurring cost in the design cycle, the costs of test pattern generation and fault simulation do influence the testing costs, especially with current emphasis in obtaining high fault coverage for non-conventional fault models such as delay faults.

To achieve a good scan design it is necessary to minimize as many of these overheads as possible. The SIESTA scan design system being developed at USC provides a framework to address the different issues in scan design. The main goal of the system is to provide a range of optimized testability solutions. In the next section we outline the underlying motivation and goals of our research.

1.3 Motivation and Goals

Structured scan selection methodologies coupled with efficient test pattern generation have eased the test generation burden for complex digital systems. However the current approaches do not form a complete solution to the testing problem. In particular for scan designs, the test application overheads manifest in test data storage and test application time can dominate the overall costs of testing. The following two points highlight the importance of reducing these overheads from a practical perspective.

1. The huge data volumes associated with scan testing stress the limits of existing test equipment and may call for re-loading the test buffers during testing [13]. Such re-loading severely affects the throughput of expensive test equipment and results in intolerable costs. Any reductions in test application time can be effectively translated into a concomitant reduction in test equipment. In addition, for maintenance testing, the duration of the test affects the overall costs since the system might need to be shut down during the test process.

2. In today's increasingly competitive electronics industry, great demands are placed on fast turnaround times for designs. Reductions in the test time of a single device are cumulative over the entire volume of such devices. In particular for high-volume productions, this reduction can lead to a tremendous savings in cost and decreased time to market. Note that stringent demands on product quality dictate that any reduction in test data volume and test time does not lead to a reduced fault coverage and related defect levels.

The main focus of our research is to reduce the test application overheads for serial scan designs. However in a scan design environment, any reductions in the test

application overheads may influence other design costs such as I/O pins, logic and routing area overheads. It is important to provide appropriate tradeoffs to a circuit designer in satisfying any testability goals and design constraints. In this context we consider a unified approach dealing with the two main issues of scan chaining and test scheduling. Both these issues form an essential component of the scan design process, particularly in designs that are subdivided into a number of partitions or *kernels*. We use the term *kernel* to refer to a portion or partition of the circuit treated as the basic primitive in both test generation and test application. The potential savings in the overall testing costs as a result of partitioning a scan design will be illustrated in a subsequent chapter.

Scan Chaining: This specifies the manner in which the scan flip-flops in a design are to be configured. In applying tests to the different circuit partitions, the scan chain organization has a significant impact on the total test application time. The scan chaining phase exploits characteristics of the design such as the test lengths of the kernels and the number and roles of scan flip-flops used in testing each kernel. The primary goal of scan chaining is to efficiently configure the scan flip-flops to minimize the total test time, while satisfying any design constraints on pin count, routing and logic area, and control complexity. To determine an appropriate chain configuration, we consider design issues such as the number of scan chains, the assignment and order of scan flip-flops in the chains, and the amount of shifting required to test a kernel.

Test Scheduling: In scan designs certain resources such as scan registers and switches are used in testing two or more kernels in the design. Alternatively the scan flip-flops might be configured into a chain used in testing multiple kernels. The need to share resources in testing the design might lead to conflicts that preclude all kernels from being simultaneously tested. In such cases it is necessary to adopt a two-phase approach to minimizing the test time. In concert with an appropriate organization of the scan flip-flops, we need to exploit potential parallelism in testing kernels concurrently. Although test scheduling is viewed as a post-process to scan chaining, the underlying chain organization plays a key role in achieving optimal

solutions. This adds a completely new perspective to the test scheduling problem, significantly different from previous approaches developed for BIST designs [14, 15].

The integration of the chaining and scheduling subsystem to a scan design system provides a framework to analyze design tradeoffs in incorporating scan testability. The wide range of techniques embodied in the subsystem leads to a huge design search space. In an environment where a designer is evaluating the suitability of alternative test strategies for a design, it is unwise to spend vast resources to accurately evaluate each of these alternatives. Accurate design estimators can provide quick answers to the "what-if" questions that may exist, providing efficient mechanisms for making trade-off decisions. By effectively pruning the design search space, a designer is guided to a set of optimized testability solutions.

1.4 Outline of Thesis

In the next chapter we briefly overview previous work in reducing test application overheads for scan designs. We point out the drawbacks of previous attempts especially with regard to issues in scan chaining and test scheduling. The main body of this thesis can be broadly divided into two parts. In the first part we concentrate on aspects that deal solely with scan chaining. In other words, the test application time is primarily determined by the manner in which the scan registers are organized into a configuration. In Chapter 3 we present a formal circuit model for a class of partitioned scan designs. For this circuit model we derive an optimal test application scheme and introduce the concept of a *shift policy.* This forms the backdrop to a classification of the chain methodologies. Based on this classification we individually analyze three different chain configurations in the subsequent three chapters. In Chapter 4 we show how significant reductions in test time can be obtained by effectively ordering the registers in a single scan chain. The popular approach to configuring multiple scan chains is to make all chains have equal length. In Chapter 5 we allow the chains to be of different lengths and show how this can lead to lower test times. In both the above cases, we provide optimal algorithms to configure the chain(s) and indicate ways to incorporate design constraints. A novel

chain configuration based on inserting multiplexers is introduced in Chapter 6. We develop associated optimization techniques and show how reconfiguration provides an effective mechanism to tradeoff between test time and area overhead.

In the second part of the thesis, we deal with aspects that require a coordinated approach to both test scheduling and scan chaining to optimize the overall test time. By enhancing the test control strategy in employing multiple scan chains, larger savings in test time can be obtained. This forms the basis for a new scan organization referred to as *asynchronous* multiple chains. The associated chaining and scheduling aspects of this organization are studied in Chapter 7. In Chapter 8 we extend the class of partitioned scan designs to allow for resource conflicts in testing kernels in the design. We classify different scheduling techniques to motivate the need for a two-phase approach in optimizing test time. Based on this observation we develop a unique algorithm that iterates between scan chaining and test scheduling in minimizing the total test time. In Chapter 9 we provide an overview of the SIESTA scan design system and put in perspective the role of the scan chaining and test scheduling subsystems in the overall design flow. We briefly discuss the use of design estimators to efficiently explore the scan design search space. Using the scan design system, we incorporate scan testability for some example datapath circuits and study the associated design costs. We conclude in Chapter 10 with some achievements of this dissertation and a list of open problems to stimulate further research.

Chapter 2

Background

In this chapter we present a brief overview of previous work related to reducing the test application overheads for scan designs. We classify these approaches based on the main idea used to achieve the desired savings.

2.1 Compact Test Sets

Generating a compact set of tests for a design without compromising on the fault coverage can reduce both the test volume and the test application time. In this context there have been attempts in both static compaction performed after the generation of all test patterns [1], and dynamic compaction done separately for each test pattern [16]. A set of faults is said to be *independent* if no two faults in the set can be detected by the same test pattern. Hence an independent fault set of minimum cardinality is a lower bound on the size of any deterministic test set generated for a circuit. This forms the basis of the approach presented in [17] to generate minimal test sets for combinational circuits. Pomeranz et al. also employ this idea to order the faults prior to test generation and combine this with a set of heuristics for test compaction and dynamic line justification in deriving compact test sets [18]. Static compaction techniques have also been investigated for stuck-open test sets [19] and for test sequences derived for sequential circuits [20]. The key point to note is that the methodologies developed in our research are independent of the manner in which test patterns are generated for the individual kernels. In other words the test time

reductions in using compact test sets are cumulative to the savings obtained as a result of efficient chaining and scheduling.

2.2 Combining Test Generation/Application

The main idea in these approaches is the use of a new technique and/or hardware configuration in the application of tests. However to obtain reductions in test time, the test generation process must be constrained to conform to the test application strategy. In [21] the test time for full scan designs is reduced by limiting the number of shift operations. A hybrid approach of sequential and combinational test generation is used in generating tests. The final test set consists of parallel test sequences in addition to test patterns that need to be individually shifted into the scan chain. The approach is extended in [22] by ordering the faults prior to test generation and the use of more intricate ways to employ the full scan capability in generating test sequences. Both the above approaches are circuit-independent in that specific characteristics of a design are not exploited in reducing the test time. More importantly the savings greatly depend on the capabilities and sophistication of the test generators.

In [23], a combination of scan techniques and parity testing is used to reduce the costs of both test generation and test application. To reduce scan operations in observing faults, an exclusive-OR tree is added at inputs/outputs of the scan flip-flops. For a given test pattern, if an error is propagated to an odd number of flip-flops, the error can be observed without scanning out the contents of the flip-flops. Control of the flip-flops without a scan operation is achieved by constraining the process of test generation. Whenever possible test patterns are generated such that the next state outputs for a given pattern correspond to the present state inputs for the subsequent pattern. However this might increase the number of patterns needed to test the design. In addition for circuits that are not parity-testable, it is not clear whether the proposed technique will reduce the overall test time. For designs with a large number of flip-flops, the EXOR tree will also result in significant area overhead. There are a number of other similar approaches that use special hardware

structures in combination with constrained test generation to reduce the overall test application time [24, 25].

For a given design, with any of the above techniques, it is difficult to predict *apriori* if any savings in test time can be obtained. The specific requirements in the test generation phase also place great emphasis on the abilities of the test generator. Constraining the test generation might result in an increased number of patterns, offsetting the desired objective of reducing the test application overheads. In contrast our methodologies only specify that test generation be performed individually on the different kernels and place no constraints on the manner in which test patterns are generated. As a consequence, the testing of a kernel in a design could be enhanced with any of the above techniques and still fall within the purview of our analysis.

2.3 Scan Chaining

We now turn our attention to previous studies directly related to the main research ideas of this dissertation. As in our work, many of these approaches employ a similar paradigm of minimizing the test time given a set of pre-determined test patterns for a scan design consisting of one or more partitions. In this section we briefly discuss prior studies that focus on issues in scan chaining; in the next section we take a look at approaches dealing with test scheduling.

2.3.1 Single and Multiple Scan Chains

Bhawmik et al. employ a knowledge-based approach to configuring both single and multiple scan chains [26]. The circuit is partitioned into independent networks referred to as a *single-DP net*, a *chain net* or a *graph net*. Each of these networks reflect different structural topologies of scan registers and *data processors*, i.e., blocks of random logic. Embodied within their system is a set of heuristic design rules dealing with the problems of determining the total number of chains to be configured and assigning and ordering registers within each of these chains. The ordering of registers within a chain is based on counting the number of data processors tested by each register in the chain. In this way, a range of heuristic strategies is used to

determine a scan chain organization that satisfies constraints on routing area and test time. The approach however suffers from a few drawbacks. The knowledge-based paradigm requires extensive designer interaction in estimating the values of different heuristic cost measures. No attempts are made to generate optimal solutions for any given design cost such as test time or routing area. An important point overlooked in the approach is the number of test patterns required to test each data processor in the circuit. As our studies have revealed, this can have a significant impact on the total test application time.

In [27], the test time is reduced by breaking up the scan path into multiple chains with the scan in/out ports of each chain multiplexed with existing system I/O pins. The scan flip-flops are distributed evenly among the chains in an attempt to minimize the routing area overhead. However constraining the chains to be of equal length does not minimize the total test time. We will clearly show that unequal length chain configurations can often lead to greater reductions in the test application time. Lee and Shin propose an interesting variation to the use of multiple scan chains in a methodology termed as *partial parallel scan* [28]. The technique is suitable for partial scan designs and involves partitioning the latches and/or flip-flops in the circuit into disjoint scan groups. All storage elements within a scan group can be controlled and observed in a single clock cycle by means of suitably multiplexing their inputs and outputs with primary I/O. Control inputs are also required to select the appropriate scan group. In applying tests, since only a subset of the scan groups may need to be controlled and observed, the test application time is reduced. However the additional area and performance overheads manifest in the routing area, tri-state buffers and large fanouts from scan-in/scan-out pins limit the general applicability of the methodology. The complexity of optimizing the test time using partial parallel scan is also shown to be NP-hard and heuristic techniques are adopted to tackle the problem.

The problem of ordering flip-flops in a single scan chain to minimize the test time has been studied by Gupta and Breuer [29]. They employ a test application scheme referred to as the *overlapped scheme* and provide a heuristic procedure based on bipartite graph matching to order the flip-flops in the chain. The output of the procedure is an ordering of flip-flops and a number reflecting the degree of confidence

in the optimality of the solution. The only optimal solutions ever obtained by the procedure fall into a small class of chain orderings whose optimality is independent of the test lengths of the kernels. Three main drawbacks of the approach are (1) the test lengths of the kernels are not considered in configuring the scan chain, (2) the scan storage elements are restricted to be flip-flops leading to increased routing overheads since flip-flops from a single register might lie in different parts of the scan chain, and (3) the heuristic procedure in most cases generates sub-optimal solutions. In our work we formally prove the optimality of the overlapped scheme for a class of partitioned scan designs. In addition we address the above drawbacks in optimally configuring a single scan chain.

2.3.2 Reconfiguring the Scan Chain

The large test times in scan designs primarily arises from the need to shift in and shift out huge volumes of test data from a device. The problem assumes even greater significance as scan tests migrate from the chip to the board level and finally onto the system level. One of the features of the IEEE 1149.1 boundary scan standard [30] is the incorporation of a *bypass* mode that permits the boundary scan chain to be reconfigured. This provides the capability to bypass any chip to reduce the shifting time in testing other chips, glue logic or interconnect on the board. In [31], a procedure is given to optimally configure the boundary scan chain to test clusters of non boundary scan devices on a mixed-technology board. The boundary scan chain provides the means to access the inputs and outputs of the non boundary scan devices. An algorithm with time complexity polynomial in the number of boundary scan chips is designed to order the boundary scan registers to minimize the testing time for the non boundary scan clusters. One of the novel chain configurations introduced in our research is based on reconfiguration and can be viewed as a migration of the bypass mode to the chip level. However at the chip level the capability to bypass every scan register will lead to unacceptable area overheads. As will be shown in Chapter 6, a clear tradeoff exists between the test time and the area overhead in terms of the number of bypass multiplexers.

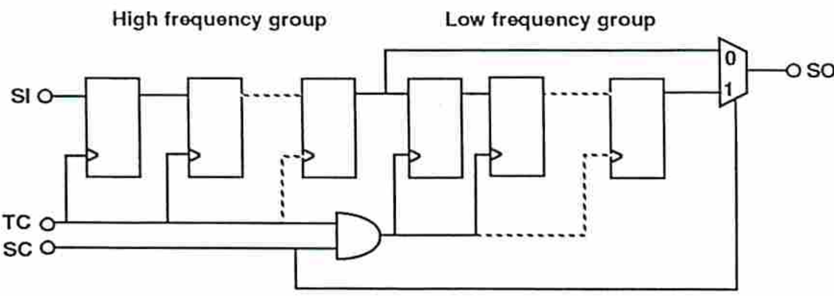


Figure 2.1: Division of the scan chain into high frequency and low frequency groups

Morley and Marlett use a multiplexer to dynamically reconfigure the scan chain and reduce the shifting requirements for partial scan designs [32]. The tests for a partial scan design consists of a set of test sequences. The test patterns within each sequence are analyzed to identify bit columns that change either frequently or infrequently, and the multiplexer is used to divide the scan flip-flops into two groups: a high frequency and low frequency group as shown in Fig. 2.1. In the figure *SI* and *SO* represent the scan-in/scan-out pins, *TC* denotes the test clock and *SC* represents the shift control signal whose value determines whether the entire chain or just the high frequency group is in shift mode. During testing the need to update the test pattern in the low frequency group is obviated in certain cases, leading to a reduction in the test time. However the fault coverage of the circuit might decrease since partial shifting also reduces observation of test results. Enhancing the test set to make up for this loss in fault coverage might offset the original reductions in test time. It is difficult to determine *apriori* if the methodology will result in any savings for a given design since it is highly dependent on the test sequences and thus indirectly on the sequential test generator used to generate the sequences. In our work we generate a reconfigurable scan chain based on the structural properties of a scan design without compromising on the fault coverage of the test set.

2.4 Test Scheduling

There exists a number of approaches in the literature dealing with the test scheduling problem for circuits designed with built-in self-test (BIST) capabilities [14, 15, 33].

For very large BIST designs, the test time can be excessively high if the tests of the various parts of the circuit are executed one after another. Reductions in test application time are achieved by carefully scheduling the tests of different logic blocks so as to exploit any available parallelism. The model of the scheduling problem used in most of these approaches is the *test compatibility graph* (TCG). Each node of the TCG represents the test event of a block of logic C_i and edges between nodes represent compatibility relations between the blocks. In other words, if an edge exists between nodes C_i and C_j, there exists no resource conflicts between the two blocks and hence the tests for the two blocks can be applied concurrently. Resource conflicts in BIST designs can be manifest in a number of ways, e.g., sharing of test pattern generators (TPG) and/or signature analyzers (SA) or I-path conflicts between the test plans of the respective blocks of logic [34]. In addition the nodes of the TCG may be weighted by their respective test lengths. The problem of generating an optimal test schedule is now reduced to enumerating the cliques of the TCG and generating a minimal cover of the weighted graph.

Since the general problem has been shown to be NP-complete, a number of variations on this basic procedure have been studied to improve the optimality of solutions and reduce the computation costs. In [14], the complementary graph (TIG) of the TCG is employed and an optimal test schedule for the case of kernels with equal test lengths is reduced to a minimum coloring of this graph. In addition a number of different scheduling disciplines are discussed for designs consisting of kernels with unequal test lengths. Chen [15] and Jones et al. [33] extend these scheduling disciplines to further optimize the overheads of test application for BIST designs. However the scheduling of tests for scan designs reveals some fundamentally different aspects of scheduling as compared to BIST designs. We will discuss the main differences in Chapter 8. At this point it is sufficient to note that since shifting time dominates the test application time for scan designs, the underlying scan chain organization plays an important role in the test scheduling process.

Since the cost of test application is a major factor in scan designs, the need to exploit potential parallelism in testing various kernels assumes even greater importance. Surprisingly there have been very few previous studies on the test scheduling

problem in the context of scan designs. Gupta considers test scheduling as a post-process to scan chaining [35]. The scan chain organization is assumed to be fixed and the scheduling of tests is done based on this given organization. The scheduling algorithm considers kernels one at a time and incrementally adds the tests of a kernel to a partial schedule to minimize the overall test time. An exhaustive branch-and-bound procedure is designed to generate an ordering of kernels such that the test time of the resultant schedule is minimized. No mention is made as to how the chain configuration is generated. In addition, the solutions obtained by the branch-and-bound procedure are optimal contingent on the fact that the scheduling of kernels is carried out incrementally.

Oostdijk et al. also employ a sequential approach in configuring the chain(s) prior to scheduling the tests for the kernels in a design [36]. The scan registers are organized into one or more chains based on a heuristic weight function evaluated for each register. A pessimistic cost function that considers the shifting in of test data and shifting out of test results as independent operations is used in generating the chain configuration. This scan chain configuration is then used as an input to the test scheduling phase. Resource conflicts between kernels are modeled by the test incompatibility graph (TIG) and using heuristics similar to those adopted in [14], a minimum coloring of the TIG is obtained to generate disjoint sets of tests that must be applied sequentially. In a post-processing technique, the test schedule is further optimized by considering pairs of test sets that can be *overlaid*, i.e., while shifting in test patterns for one set of tests, the results of another set of tests are shifted out. As we will show in our analysis, use of the TIG in a form similar to that used in BIST scheduling constrains the search space of feasible test schedules. Due to the close interaction between scan chaining and test scheduling, a sequential approach to the two phases will not lead to globally optimal solutions.

Chapter 3

Overview of Scan Chaining

3.1 Introduction

The testing costs for a design can be broadly divided into the costs of test generation and the costs of test application. The related design costs associated with scan techniques mainly consists of the area overhead and potential performance degradation in modifying the storage elements and the one-time cost of test pattern generation. The cost of test application on the other hand is primarily determined by the test application time and the test volume. Scan chaining thus plays a vital role in the scan design process since the organization of the scan flip-flops greatly determines the total test application time. Secondary design costs linked to test application such as routing area overheads, pin count and the test control complexity are also influenced by the scan chain configuration. In this chapter we lay the groundwork to study an entire range of scan chain methodologies for partitioned scan designs. This will serve as an introduction to the first part of this thesis dealing solely with scan chaining. We decouple the scan chaining aspect of reducing test time from the test scheduling aspect by restricting our attention to a class of partitioned designs. For this class of designs an optimal test application scheme is derived. In conjunction with this scheme we introduce the concept of a *shift policy* that exploits structural properties of a design to both reduce the shifting time and provide design cost tradeoffs. These basic ideas and definitions provide the framework to classify the different scan chain organizations, which are discussed in detail in the subsequent three chapters.

In the second part of this thesis consisting of Chapters 7 and 8, we make use of some concepts introduced in this chapter. However for the methodologies considered in these two chapters, it is not possible to decouple the scan chaining problem from the associated test scheduling problem. For example in using asynchronous scan chains even with a restricted circuit model, the optimization of test time is a function both of the configuration of the registers as well as the manner in which tests are applied to the design. Similarly when we extend the circuit model in Chapter 8, both aspects of test scheduling and scan chaining need to be considered in minimizing the overall test time.

3.2 Circuit Model

As a first step in introducing the scan chaining problem we briefly consider the benefits in partitioning a scan design. This will motivate a class of partitioned designs that will serve as the circuit model in our chaining analysis. To cope with the testing problem for complex VLSI circuits, it is widely acknowledged that one has to partition the circuit into independently testable blocks. The partitioning of a design could be performed manually by designers, based on their knowledge of the functionality of circuit modules. Alternatively automatic partitioning techniques that exploit both the structure and function of circuit blocks, can be used to subdivide a design. For example in a full scan design of a sequential circuit, the scan path provides complete access to the combinational portions of the circuit. This combinational logic can be partitioned into a set of kernels, or disjoint portions of logic that can be tested independently. Each kernel is essentially a maximal region of connected combinational logic [37]. Similarly there exist a number of other approaches to generate disjoint kernels for both full and partial scan designs [26, 35, 38, 39]. For example in [35], *size-based* partitioning divides the kernel obtained by partial scan into a number of smaller disjoint kernels by scanning additional registers. Test generation can be carried out as a separate process on each of the kernels. In general partitioning the scan design can realize a number of potential advantages.

- Since test generation is carried out on smaller pieces of logic, the overall test generation cost is reduced

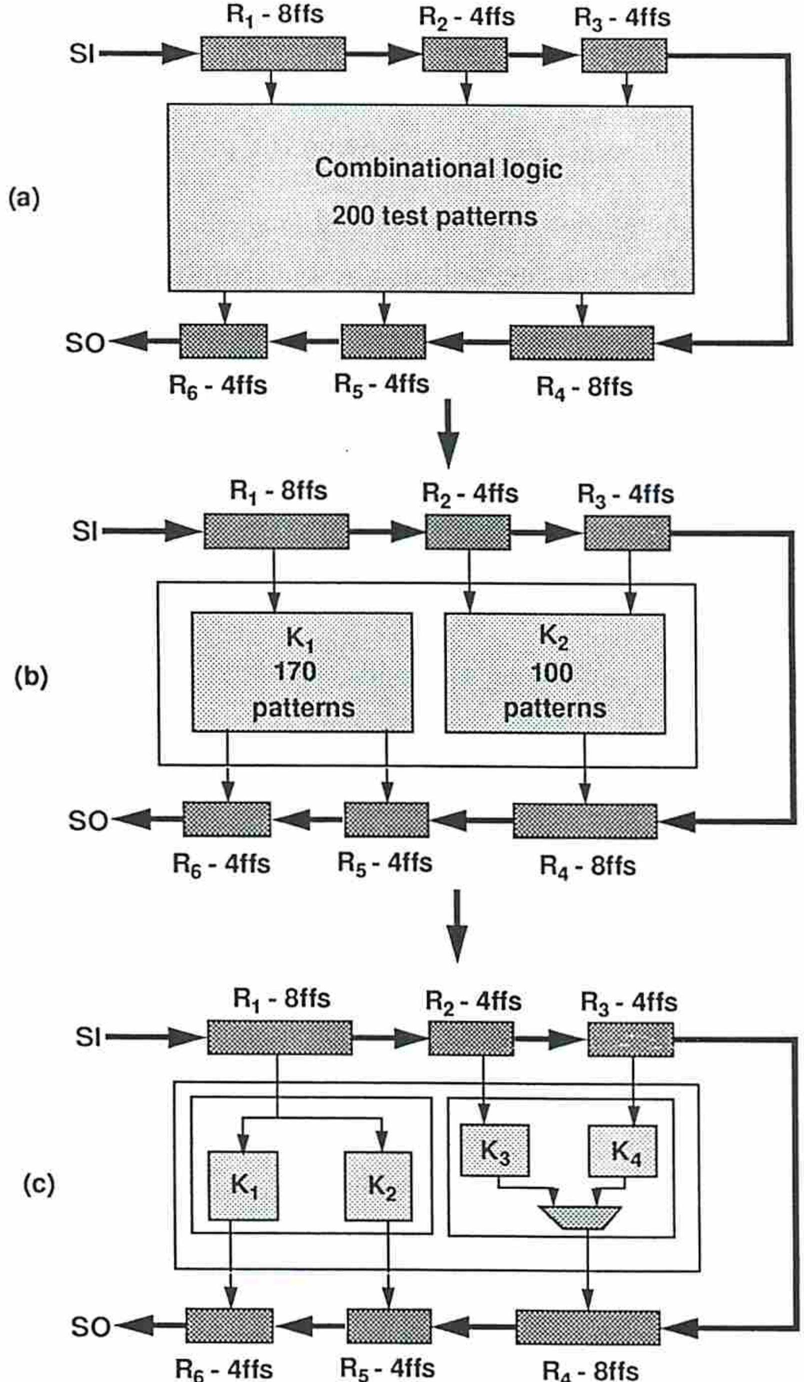


Figure 3.1: (a) Unpartitioned scan design. (b) Partitioned design with disjoint kernels. (c) Partitioned design with non-disjoint kernels.

- As we will show in our analysis, by applying tests for the different kernels concurrently, the test application overheads can be minimized.

- In an environment that requires redesign of certain partitions in the circuit, test generation need only be re-done on the partitions affected by the change.

- Since both the application of tests and analysis of responses are done individually for each kernel, the fault diagnosis capabilities of the testing process are enhanced.

- Test compaction can be done more efficiently on a partitioned block of logic.

- We can associate with each kernel a unique test length as well as a set of scan flip-flops and primary I/O involved in testing the the kernel. This knowledge can be exploited to reduce the shifting time during the application of tests.

To illustrate these advantages, consider the full scan design shown in Fig. 3.1(a) consisting of a single combinational kernel and six scan registers. The primary inputs/outputs are not shown in the figure. A **register** is defined as a collection of one or more homogeneous flip-flops with all flip-flops driven by the same clock and control signals. The number of patterns to fully test the combinational kernel is equal to 200. To apply each pattern, we shift it into the scan chain for 32 clock cycles equal to the number of flip-flops in the chain, before applying a normal clock to propagate the pattern through the logic and latch the results. While shifting out the results, the next test pattern is simultaneously shifted into the chain. The total test application time is thus given by $200(32 + 1) + 32 = 6632$ clock cycles, where the final 32 clock cycles are to shift out the results of the last pattern. As shown in Fig. 3.1(b), the combinational kernel can be partitioned into two smaller disjoint kernels by clustering maximally connected logic [37]. Each of the two kernels can be handed individually to an ATPG system, leading to a reduced cost in test generation as compared to the single kernel. Note that the number of patterns to test either of the two kernels cannot be greater than 200. More importantly, for the partitioned design with the same configured scan chain, the total test time can be reduced to 2346 clock cycles, a savings of 65% over the unpartitioned case. The details in deriving this test time will be explained shortly.

More refined partitioning techniques can be used to further divide disjoint kernels into smaller subkernels [35, 40, 41]. As shown in Figure 3.1(c), such techniques might exploit the existence of fanout points and/or *switches* such as multiplexers and busses. Test generation can also be performed individually on each of these smaller kernels. However in testing the design, resource conflicts manifest in the use of the multiplexer or scan register, preclude every subset of kernels from being concurrently tested. For such cases a combination of both scan chaining and test scheduling will be required to optimize the test time. This will form the focus of our discussion in Chapter 8. For our scan chaining analysis, we restrict our attention to designs as in Fig. 3.1(b) consisting of disjoint kernels in which every subset of the kernels can be concurrently tested.

A scan design is thus modeled by (1) an ordered set $KSEQ = \{K_1, K_2, \ldots, K_n\}$ of n kernels, each of which may be combinational or sequential; (2) a set of M scan registers $S = \{R_1, R_2, \ldots, R_M\}$; and (3) a set of primary inputs/outputs. Test generation is done individually on each kernel in $KSEQ$. For kernel K_i $(1 \leq i \leq n)$, let W_i denote its *test length*, i.e., the number of patterns required to fully test the kernel. The kernels in $KSEQ$ are ordered in terms of non-decreasing test lengths, i.e., $W_1 \leq W_2 \leq \ldots \leq W_n$. Let $WSEQ = \{W_1, W_2, \ldots, W_n\}$ denote the ordered set of test lengths. In addition for each kernel we associate a subset of scan registers and primary I/O that serve to apply test patterns and receive test results from the kernel. Since every subset of the kernels in $KSEQ$ can be concurrently tested, every scan register and primary I/O can apply tests to and/or receive results from at most one kernel. Hence we characterize each register R_k $(1 \leq k \leq M)$ by the following three parameters.

- *length* l_k - number of scan flip-flops in R_k.
- *driver weight* d_k - test length of kernel fed by R_k.
- *receiver weight* r_k - test length of kernel feeding R_k.

For example in the scan design shown in Fig. 3.2, the parameters for register R_3 are equal to $l_3 = 2, d_2 = 50$ and $r_2 = 300$.

As pointed out earlier our chaining techniques are independent of the manner in which tests are generated for the individual kernels. In our analysis we will assume all

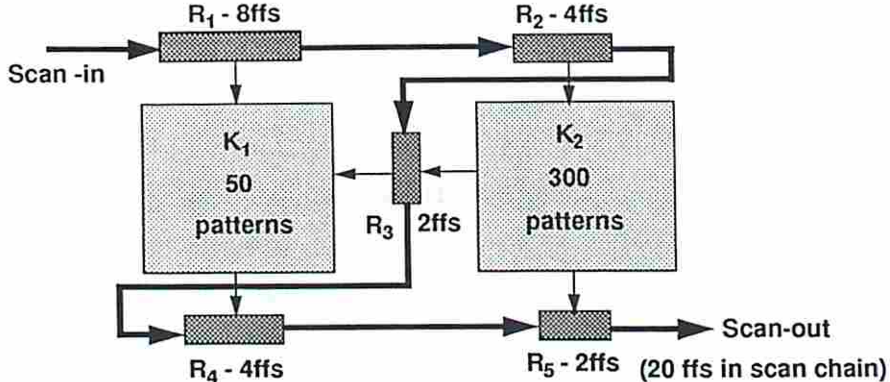


Figure 3.2: Example design with a single scan chain.

kernels in $KSEQ$ are combinational. Minor modifications might be required when dealing with sequential kernels. With a balanced sequential kernel, d clock cycles will be required to propagate test data through the kernel as opposed to a single clock cycle for a combinational kernel [35]. The value of d is given by the *sequential depth* of the kernel. A number of other variations in applying tests to the kernels can be accommodated within our analysis. For example a single test pattern can be scanned and held in the flip-flops and the primary inputs varied while clocking the kernel a number of times [42]. The main concepts in reducing test time with combinational kernels can be extended to deal with these test application modes for sequential kernels.

3.3 Test Application Schemes

3.3.1 Combined and Overlapped schemes

Consider the full scan design shown in Fig. 3.2. The design consists of two combinational kernels K_1 and K_2 with test lengths of 50 and 300, respectively. The number of flip-flops within each register is shown and all five registers are connected into a single scan chain. This circuit will be used to illustrate and define the test application schemes and shift policies.

The traditional test scheme would merge the test sets for the two kernels into a single test set of length 300 and apply these tests to both kernels simultaneously. Each test pattern is applied by shifting it into the scan chain for 20 clock cycles, where 20 represents the number of flip-flops in the chain. A single pulse of the normal clock is then applied to propagate the pattern through the kernels. To observe the results, the scan chain is placed in shift mode for 20 clock cycles while simultaneously shifting in the next pattern. This test application scheme, referred to as the **combined** scheme, in effect treats both kernels in the design as a single kernel with a test length of 300. The total test time is equal to $300(20 + 1) + 20 = 6320$ clock cycles, where the final 20 clock cycles are to shift out the results of the last pattern. A drawback of the combined scheme is that no information about the design is used in the application of tests.

To fully exploit information about the different kernels and scan registers in a circuit, we employ a more efficient test application scheme referred to as the **overlapped** scheme [1]. In the overlapped scheme, the application of tests is divided into a set of **test sessions**. In each test session a fixed number of test patterns are applied to a subset of kernels until the test set of the kernel with least test length is exhausted. For our example the overlapped test scheme will consist of two test sessions TS_1 and TS_2. In the first session TS_1, 50 test patterns are applied to both kernels until the test set of K_1 is exhausted. The remaining 250 test patterns are applied to K_2 in the second test session TS_2.

Before generalizing the overlapped scheme, we simplify our circuit model in the following way. Two or more kernels with *equal* test length are merged into a single kernel. Every scan register and primary I/O associated with any of the original kernels is now associated with the merged kernel. This simplification will not affect the test application time with any of our chaining methodologies. Hence we can assume that $KSEQ$ contains n kernels of strictly increasing test lengths. In the first test session, W_1 patterns are applied to all kernels until the test set of K_1 is exhausted. Similarly in the second session, $(W_2 - W_1)$ test patterns are applied to kernels $K_2, \ldots, K_n$ until the test set of K_2 is exhausted. The overlapped scheme will thus consist of n test sessions $\{TS_1, TS_2, \ldots, TS_n\}$ and is summarized in the following procedure. Define $W_0 = 0$.

Overlapped Test Scheme

For $i = 1$ to n do:

{

In session TS_i

Apply $W_i - W_{i-1}$ test patterns to kernels $K_i, \ldots, K_n$.

}

The overlapped scheme only specifies the subset of kernels tested in each session and the number of patterns applied to these kernels. To complete the definition of each session, we need to specify the **chain cycle** employed in the session.

Definition The **chain cycle** of a test session is defined as the number of clock cycles used to shift in a test pattern while simultaneously shifting out the results of the previous pattern.

To determine the total test application time with the overlapped scheme, we need to evaluate the chain cycles for each session. The chain cycle of a session is determined by two factors, (1) the *shift policy*, and (2) the scan chain configuration.

3.3.2 Flush and Minimum Shift Policies

For a given chain configuration, the shift policy specifies the amount of shifting used in each session of the overlapped scheme. It reduces the shifting requirements in applying tests by exploiting information about the scan registers that need to be accessed in the session. We deal with two different shift policies, i.e., the **flush** policy and the **minimum shift** policy.

Under the flush policy, the chain cycle of a session is given by the minimum number of clock cycles to **flush** the chain(s) that contain the registers used in the session. A scan chain is said to be flushed if the number of clock cycles for which it is placed in shift mode is equal to the length of the chain. The length of the chain is measured in terms of the number of flip-flops. For our example design, consider the second session TS_2. To apply 250 patterns to K_2, only registers R_2, R_3 and R_5 need to be accessed. Since there only exists one scan chain in the design. the chain cycle of the session is given by the length of the chain, i.e., 20 clock cycles. For the design,

Test session	Kernels tested	No. of patterns	Drivers	Receivers	Chain cycle	Test time
TS_1	K_1, K_2	50	R_1, R_2, R_3	R_3, R_4, R_5	14	750
TS_2	K_2	250	R_2	R_3, R_5	12	3250

Table 3.1: Summary of test sessions with minimum shift policy

the total test time with the flush policy is thus equal to 50(20 + 1) + 250(20 + 1) + 20 = 6320 clock cycles. In designs with a single scan chain, the overlapped scheme with the flush policy is equivalent to the combined scheme. The flush policy will prove useful when dealing with organizations containing two or more scan chains, e.g., multiple scan chains or a reconfigurable scan chain. Note that the combined scheme implicitly employs a flush policy since each pattern is shifted for the entire length of the scan chain.

The minimum shift policy more accurately characterizes the roles of a register used in a particular session. For example in the first session, registers R_1, R_2 and R_3 are involved in applying tests to both kernels while R_3, R_4 and R_5 are used to capture test results. We refer to registers involved in applying test patterns as *drivers* and registers involved in receiving test results as *receivers*. A register used both as a driver and receiver in a session is referred to as a *driver-receiver*. The chain cycle of the session is given by the minimum number of clock cycles to shift data into the drivers while simultaneously shifting out results from the receivers. For example in TS_1, 14 clock cycles are required to shift a pattern into the drivers R_1, R_2 and R_3 and 8 clock cycles are required to shift out the results from the receivers R_3, R_4 and R_5. The chain cycle is given by the maximum of these two numbers, i.e., 14 clock cycles. Similarly the chain cycle employed in the second session TS_2 is equal to 12, since a minimum of 12 clock cycles is required to shift test data into R_2 while simultaneously shifting out results from R_3 and R_5. The total test time is thus equal to 50(14 + 1) + 250(12 + 1) + 14 = 4014 clock cycles. This represents a 36.4% reduction in test time over the combined scheme. For the circuit in Fig. 3.2, the details of the overlapped scheme with the minimum shift policy are summarized in Table 3.1.

The test application scheme determines the manner in which the total test time is computed. The overlapped scheme in conjunction with a shift policy exploits information about the kernels and their respective test lengths, the scan registers used in testing each kernel, and the organization of the scan registers, to minimize the overall test application time. For a design satisfying our circuit model and given any scan chain configuration, it can be shown that the overlapped scheme represents an optimal test scheme. In other words the test time with the overlapped scheme will be less than or equal to the test time with any other test application scheme. A formal proof of this is deferred to Chapter 8 after introducing some concepts and notation in test scheduling. In all subsequent discussion we will therefore assume the overlapped scheme is used to apply tests.

The total test time under the overlapped scheme is equal to the sum of the durations of each test session. We will use the notation CC_i to denote the chain cycle of session TS_i $(1 \leq i \leq n)$. The duration of session TS_i is equal to $(W_i - W_{i-1})(CC_i + 1)$. Hence the total test time is given by

$$\mathcal{T} = \sum_{i=1}^{n}(W_i - W_{i-1})(CC_i + 1) + CC_1$$

Note that the final term CC_1 represents the time required to observe the results of the last pattern and to synchronize between sessions. Since the addition of 1 introduces a constant term equal to W_n to the total test time, the above expression can be simplified as shown below.

$$\mathcal{T} = \sum_{i=1}^{n}(W_i - W_{i-1})CC_i + CC_1 \tag{3.1}$$

This equation will serve as the objective function to be used in organizing the registers into the various scan chain configurations.

3.4 A Classification of the Chain Methodologies

Based on the definitions of the test application schemes and shift policies, we can now classify the various scan chain methodologies. Fig. 3.3 shows the range of

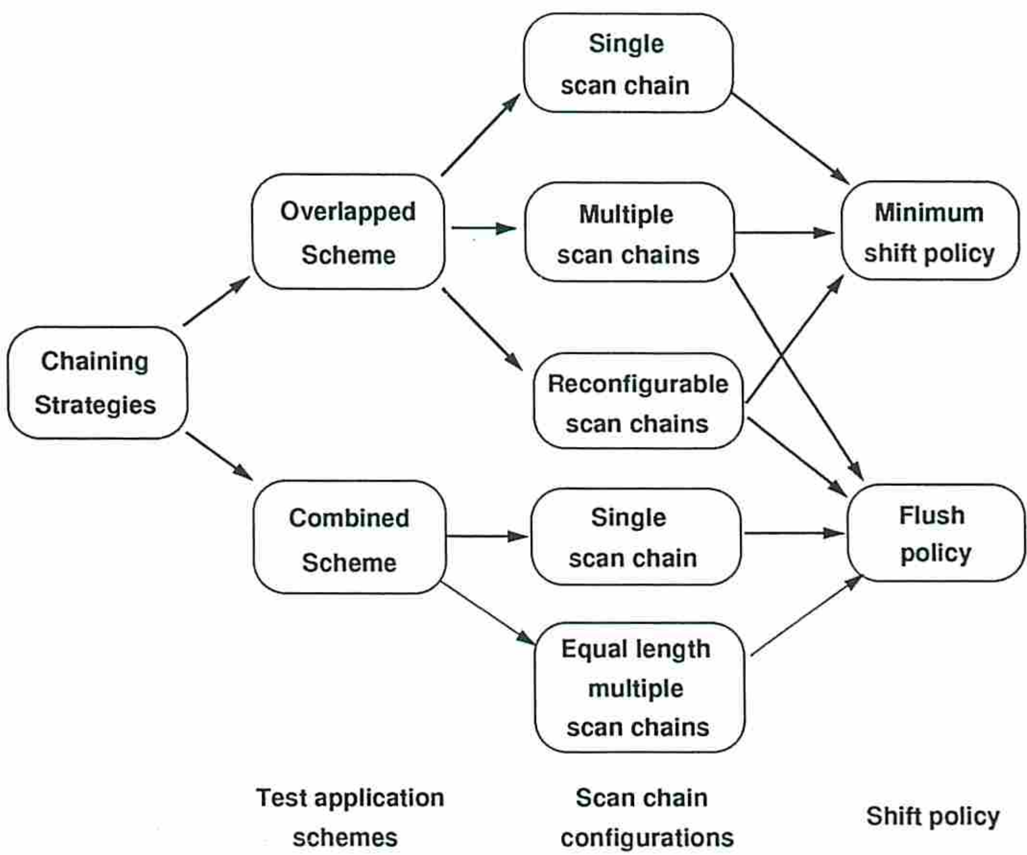


Figure 3.3: Classification of chain methodologies.

methodologies in the proposed classification. The methodologies are classified based on three criteria: (1) the test application scheme; (2) the scan chain configuration; and (3) the shift policy. With a single scan chain, the combined scheme with a flush policy represents the conventional approach to applying tests. However as shown by our simple example, use of the overlapped scheme with a minimum shift policy can reduce the overall test time. In fact even greater savings in test time can be obtained by effectively ordering the registers in the chain; this will form the focus of our discussion in the next chapter.

In a similar fashion the overlapped test application scheme with both shift policies can be utilized for other scan chain configurations. Multiple scan chains are obtained by breaking up a single chain into a number of smaller chains as shown in Fig. 3.4(a). Since each chain is equipped with its own scan-in and scan-out pins, data can be shifted in parallel to greatly increase the bandwidth of test data. The conventional approach to using multiple chains is to make all chains of equal length, and employ the combined scheme with a flush policy to apply tests. However a more effective way to use multiple chains is to employ the overlapped scheme with a configuration that permits the chains to be of unequal lengths. Issues such as the number of chains, the assignment and order of registers in the chains, and design cost tradeoffs with routing area and pin count will be dealt with in Chapter 5. The third scan configuration dealing with a reconfigurable scan chain is based on inserting multiplexers at selected positions in a single chain, as shown in Fig. 3.4(b). This permits the realization of multiple scan chains that can be used in a serial fashion. The different aspects of this organization will be discussed in Chapter 6.

In the overlapped scheme, recall that the chain cycle of a session is a function of both the shift policy and the scan chain organization. For a given shift policy, the primary goal of the scan chaining techniques is to minimize the chain cycles of each session and hence the value of the objective function as given in equation 3.1. Note that the optimality of the overlapped scheme is contingent on the assumption that a unique chain cycle is used in each test session. As we will show in Chapter 7, if two or more chain cycles are used in a session, the overlapped scheme no longer represents an optimal way to apply tests. With asynchronous multiple scan chains, it is not possible to determine *apriori* an optimal test application scheme. This in

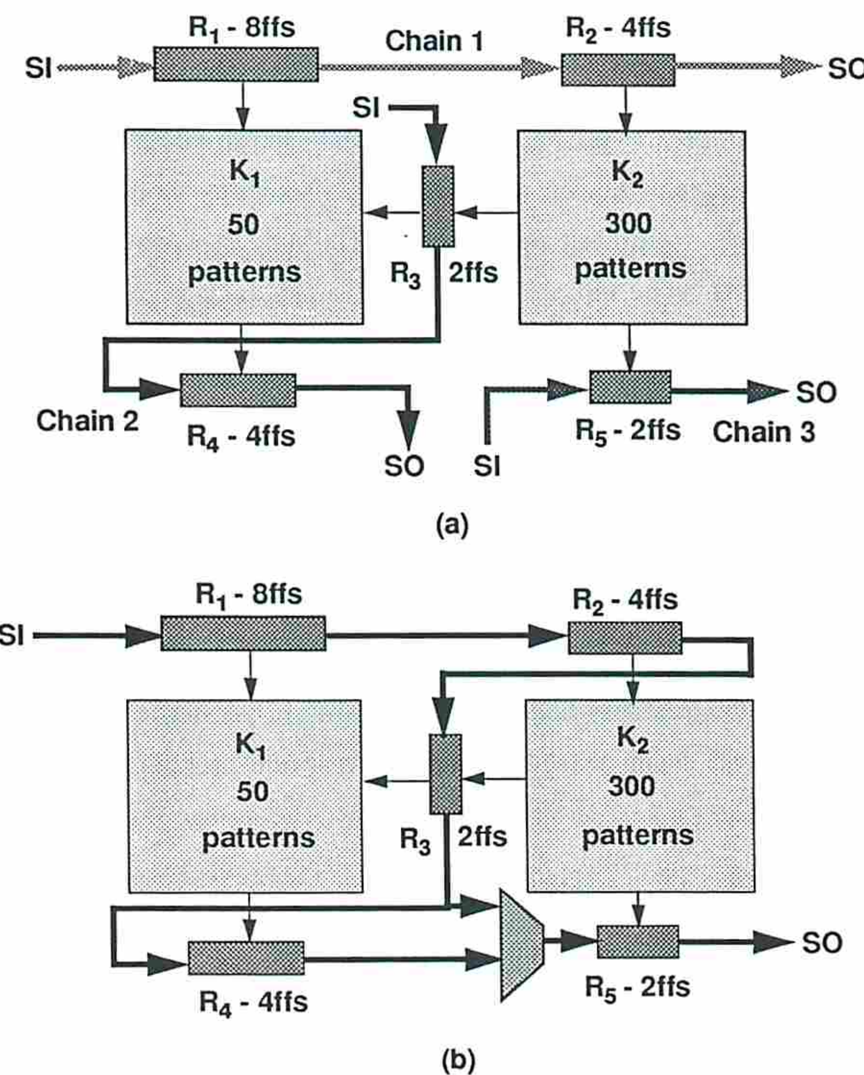


Figure 3.4: (a) Multiple scan chains.(b) Reconfigurable scan chain.

turn explains why asynchronous multiple chains are not included in the classification of Fig. 3.3.

3.5 Reducing Test Time and Test Volume

In Section 1.3 we showed the importance of reducing both test application time and test data storage from a practical perspective. To more clearly illustrate this point, let us consider a hypothetical device that requires 100000 deterministic test patterns. Assume each pattern needs to be shifted into a scan chain of length 500 flip-flops. On test equipment operating at 100MHz, the total tester time for the scan tests would thus be equal to 0.5 seconds. In such a case, reducing test time by say 50% would not significantly impact the overall costs. However the chaining and scheduling methodologies developed in this thesis can reduce the shifting time for deterministic test patterns as well as pseudorandom test patterns. If the number of pseudorandom patterns required for the device is an order of magnitude greater, i.e., 10^6, the total tester time would increase to 5 seconds. Similarly with increasing device complexities, if the number of flip-flops in the chain increased from 500 to 5000, the test time would correspondingly increase to 5 seconds.

It is difficult to quantify the monetary gains in reducing the test time for scan patterns from say 5 seconds to 2 seconds on test equipment. This would depend on factors such as the cost of the test equipment, the equipment operating costs, and the depreciation in the value of the equipment. Taking into account these factors, let us assume the cost of a single second on test equipment translates to 10 cents. For a device that costs about 200-300 dollars, reducing three seconds of test time would not significantly affect the overall economics. On the other hand for a device costing about 2-3 dollars and produced in large volumes, reducing the test time by three seconds could potentially translate to a 10% profit margin. Note however that tester time is a combination of time for scan tests as well as other parametric and functional tests. Hence one should consider the percentage of the total tester time occupied by scan tests in the overall cost savings. As with test equipment, it is difficult to quantify the savings in cost in reducing the down-time of systems for maintenance testing.

An equally important role in the test application costs is played by the test data volume. Reducing test volume can potentially reduce both the number of buffer reloads on test equipment, and the random access memory (RAM) costs for standalone testing. In certain cases a 50% reduction in test volume might be more important than a corresponding reduction in test time. With regard to test data shifted into the scan chain(s), any reductions in test time translate to an equivalent reduction in the volume of data that needs to be stored. In addition savings in test volume are also obtained for data that needs to be applied at the primary inputs and observed at the primary outputs. Recall that the overlapped scheme exploits information about the different kernels in a design. In particular the testing of the design progresses in stages such that one or more kernels are fully tested in every session. As a result we save on the volume of test data for the primary inputs/outputs (PI/POs) of kernels that are fully tested in each session.

To provide an estimate of the savings in test data at the PI/POs, consider a design of n kernels. Let $\mathcal{P}$ denote the sum of the number of primary inputs and outputs of the design. For an average case analysis let us assume each kernel is tested by $\mathcal{P}/n$ PI/POs. With the combined scheme the volume of test data for the PI/POs is given by $\mathcal{P}W_n$. Under the overlapped scheme, the savings in test data for kernel K_1 is equal to $(W_n - W_1)\mathcal{P}/n$. In general for kernel K_i the savings in test data is given by $(W_n - W_i)\mathcal{P}/n$. The total savings in test data for the design is given by

$$\mathcal{P}/n(nW_n - W_n - \sum_{i=1}^{n-1} W_i)$$

If we assume $\sum_{i=1}^{n-1} W_i = \gamma W_n$ where $0 < \gamma < n-1$, the savings in test volume over the combined scheme is given by the following expression.

$$\text{Percentage savings} = (1 - \frac{\gamma + 1}{n}) \times 100$$

Consider the case of a design with two kernels, i.e., $n = 2$. As γ varies from $n-1$ to 0, the corresponding savings in test volume vary from 0 to 50%. Similarly for a design with $n = 10$, as $\gamma \rightarrow 0$ the savings in test volume can be as large as 90%. It is important to keep the total test volume below a critical upper bound. For a given tester, if the test volume exceeds the associated upper bound for the tester,

the entire test data cannot fit into the primary memory of the tester. Thus a tester reload process must take place, and this is a time consuming and expensive task.

3.6 Summary

For the class of designs under consideration, the overlapped scheme provides for an optimal way to apply tests. The concept of a shift policy exploits information about the scan registers and the chain organization to reduce shifting requirements in applying tests. Based on these ideas, we have laid the framework to study a range of chain configurations. The most significant aspect of these chain methodologies is the tremendous savings in test time that can be obtained over conventional techniques. In addition the methodologies provide ways to tradeoff the associated design costs so as to enhance the efficiency of incorporating scan testability.

Chapter 4

Ordering Registers in a Single Scan Chain

4.1 Introduction

In this chapter we study the first of our three chain configurations dealing with a single scan chain. The conventional approach to testing designs with a single chain is by means of the combined scheme. In this regard the combined scheme does possess two advantages. Since the entire scan chain is placed in shift mode for each pattern, the test time is independent of the order of registers in the chain. Hence full flexibility is afforded to a designer to minimize the routing area in ordering the registers in the chain. The second advantage of the combined scheme arises from the uniformity of the testing scheme. Since the same sequence of operations are performed throughout the test process, the test control complexity is minimal. However its main drawback is that a large percentage of the test application time is wasted in the shifting process.

As shown in the previous chapter the test time can be reduced by using the overlapped scheme in concert with a minimum shift policy. For a given scan chain, the minimum shift policy reduces the chain cycle of each session and hence the overall test time. In this chapter we focus on further reducing this test time by modifying the order of registers in the scan chain. The idea basically arises from observing that certain scan registers are more frequently accessed than others as part of the test process. The concept of "frequently used" registers will be a recurring theme in all our chaining techniques. Intuitively the frequently used registers correspond to

those registers involved in testing kernels with larger test lengths. Any disparity in the usage of registers can be exploited in generating the chain configuration. For the case of a single scan chain, this corresponds to placing frequently accessed registers close to the scan-in/scan-out pins.

We present a structured methodology to configure a single scan chain that minimizes the test application time under the overlapped scheme. As a first step we derive a mathematical model of the problem to optimally order the registers in the chain. This model is used to establish the NP-completeness of the problem before delving into some analysis and algorithmic techniques to generate optimal solutions. We indicate ways to tune the algorithm to take into account any routing constraints, and conclude with some implementation results highlighting the advantages of the proposed methodology.

4.2 Configuring the Scan Chain

Consider the full scan design shown in Fig. 4.1(a) reproduced from the previous chapter. The chain cycles in sessions TS_1 and TS_2 are equal to 14 and 8, respectively. The total test time is thus equal to $50(14 + 1) + 250(8 + 1) + 14 = 4014$ clock cycles. The chain cycles in each session depend on the order of registers in the chain. To illustrate this we modify the ordering of registers as shown in Fig. 4.1(b). Note that the three registers involved in testing K_2 are placed such that R_2 is at the beginning of the chain, and R_3 and R_5 are at the end of the chain. As a result the chain cycle employed in the first session increases from 14 to 18, and the chain cycle in the second session decreases from 12 to 4. Since a greater number of test patterns are applied in the second session, the total test time decreases to 2218 clock cycles. This represents a reduction of 64.9% over the combined scheme and a reduction of 44.7% over the scan chain in Fig. 4.1(a). The ordering of registers in fact represents an optimal single chain configuration in terms of minimizing the test time. Conversely consider the order of registers as shown in Fig. 4.1(c), in which R_5 is placed at the beginning of the chain. Since it is used as a receiver in both test sessions, the chain cycles in both sessions will be equal to 20, i.e., the length of

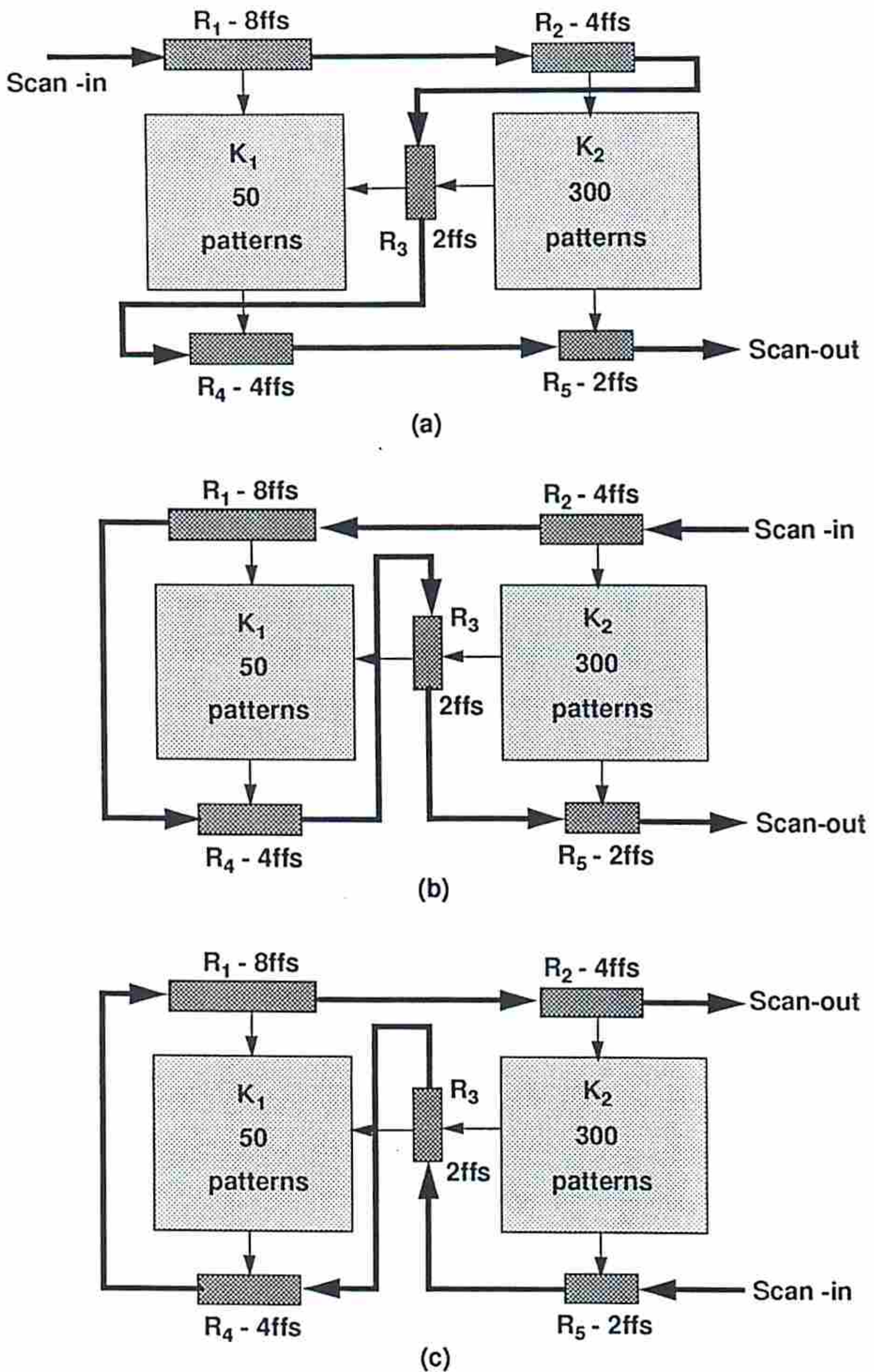


Figure 4.1: (a) Example scan design. (b) Optimal scan chain. (c) Worst-case ordering of registers.

the scan chain. This represents a worst-case ordering of the registers where the test time with the minimum shift policy is equal to that of the combined scheme.

The goal of our methodology is to provide efficient ways to generate optimal single chain configurations as in Fig. 4.1(b). The large reductions in test time are clearly a function of both the test lengths of the kernels, and the roles of the scan registers in the design. In particular the test lengths of the kernels influence the optimal scan chain ordering. To illustrate this, let us assume the test length of K_2 in our design is equal to 60. If the scan chain in Fig. 4.1(b) is used to test the circuit, the total test time will be equal to 1018 clock cycles. An optimal chain however is constructed by ordering the registers in the sequence R_2, R_1, R_3, R_4, R_5 from the scan-in to the scan-out pin. The associated test time is equal to 854 clock cycles. One of the main drawbacks of previous approaches [29, 26] is that the test lengths of the kernels are not taken into account in configuring the chain.

With the overlapped scheme the savings in test time are obtained at the cost of a more restricted scan path ordering. The area overhead, if any, resulting from constraining the order of registers only involves the routing of single-bit wires between adjacent registers in the chain. In addition the algorithmic techniques to be developed can be tailored to handle routing constraints.

4.3 Model of the Optimization Problem

In this section we develop a mathematical framework to deal with the problem of generating an optimal ordering of the scan registers. To evaluate the chain cycle of a session we require additional definitions that relate the registers in the circuit to their roles in the session. The scan chain of Fig. 4.1(a) is shown in Fig. 4.2 with the parameters of every scan register indicated above the figure. Note that these parameters are a function only of the circuit topology and independent of the scan chain configuration.

Consider register R_3 in the figure. Since its driver weight is equal to 50, it will be used as a driver only in TS_1. Similarly since its receiver weight is 300, it will be used as a receiver in sessions TS_1 and TS_2. More generally, a register with driver

Parameters independent of scan chain configuration

	R_1	R_2	R_3	R_4	R_5
Length	8	4	2	4	2
Driver Weight	50	300	50	0	0
Receiver Weight	0	0	300	50	300
Position	1	2	3	4	5
Driver Distance	8	12	14	18	20
Receiver Distance	20	12	8	6	2

SI → R_1 → R_2 → R_3 → R_4 → R_5 → SO

Parameters dependent on scan chain configuration

Figure 4.2: Scan chain configuration with register parameters.

(receiver) weight equal to W_i $(1 \leq i \leq n)$ will be involved in testing kernel K_i and hence used as a driver (receiver) in every session from TS_1 to TS_i. A register with receiver (driver) weight equal to 0 is referred to as a a *pure* driver (receiver). In Fig. 4.2, R_1 and R_2 are pure drivers, and R_4 and R_5 are pure receivers.

The single scan chain to be configured can be represented by a sequence of M slots numbered $1, 2, \ldots, M$ to which registers are to be assigned. We adopt the convention that the first slot represents the location of the leftmost scan register closest to the scan-in pin and the Mth slot represents the location of the rightmost scan register closest to the scan-out pin. The scan path chaining problem basically involves the assignment of a scan register to each slot such that the test time is minimized, i.e., a one-to-one mapping between the elements of the set S and the M slots. Let us assume a given permutation π of the scan registers in the M slots. Every scan register R_k will have a unique position p_k in the scan chain, where p_k represents the slot to which R_k is assigned. The following additional two parameters can be extracted for every scan register R_k.

1. Driver distance $Ddist_k$ - this represents the number of clock cycles required to shift a single bit from the scan-in pin to the rightmost flip-flop in R_k, i.e.,

$$Ddist_k = \sum_{\forall j \ni p_j \leq p_k} l_j.$$

2. Receiver distance $Rdist_k$ - this represents the number of clock cycles required to shift a single bit from the leftmost flip-flop in R_k to the scan-out pin, i.e.,

$$Rdist_k = \sum_{\forall j \ni p_j \geq p_k} l_j.$$

The position p_k of a scan register R_k as well as its driver (receiver) distance $Ddist_k$ ($Rdist_k$) can be evaluated given a fixed permutation π. For the scan chain in Fig. 4.2, the positions, driver distances and receiver distances of every register are shown.

Consider the registers used as drivers in session TS_i. By definition the driver weights of each of these registers must be greater than or equal to W_i. The minimum number of clock cycles required to shift test patterns into these registers is equal to the driver distance of the register closest to the scan-out pin. Similarly if we consider all registers used as receivers in the session, the receiver distance of the register closest to the scan-in pin will determine the minimum number of clock cycles required to shift out test results.

We create a set of the driver distances of all registers used as drivers in TS_i and a similar set of receiver distances for the registers used as receivers. The chain cycle CC_i is equal to the maximum number in the union of these two sets, i.e.,

$$CC_i = max(\{Ddist_k | d_k \geq W_i\} \cup \{Rdist_k | r_k \geq W_i\}).$$

We use the term *drive* (*receive*) cycle of a session to refer to the maximum number in the set of driver (receiver) distances evaluated for the session. Hence the chain cycle of a session is equal to maximum of its drive and receive cycles. Let us illustrate the evaluation of the chain cycle in test session TS_2 for the scan chain shown in Fig. 4.2. The set of driver distances is $\{12\}$, i.e., the driver distance of R_2, and the set of receiver distances is $\{8, 2\}$, i.e., the receiver distances of R_3 and R_5, respectively.

The maximum number in the union of these two sets is 12 which represents the chain cycle of TS_2.

In this way given any scan chain, the driver and receiver distances of every scan register can be computed to determine the chain cycle in each test session. These can then be used in equation 3.1 to determine the value of the objective function. To simplify the optimization procedures, we will neglect the constant term CC_1 in equation 3.1 in all subsequent analysis. For a given scan chain π, we denote the value of its objective function as $Obj(\pi)$. The complexity of determining $Obj(\pi)$ is dominated by the evaluation of the chain cycles for the respective test sessions. The worst case time complexity to evaluate the chain cycles is of order $O(Mn)$.

4.4 Analysis of Optimal Configurations

4.4.1 Complexity Results

The model developed in the previous section is used to formulate and analyze the scan register sequencing problem. The search problem deals with finding a scan chain configuration that minimizes the objective function. The corresponding decision problem is formulated and shown to be NP-complete by a simple reduction.

Scan Register Sequencing (SRS)

Given a positive integer K, the set $WSEQ = \{W_1, W_2, \ldots, W_n\}$, the set $S = \{R_1, R_2, \ldots, R_M\}$ of M scan registers and for each $R_k \in S$ $(1 \leq k \leq M)$, a length $l_k \in Z^+$, a driver weight $d_k \in WSEQ$ and a receiver weight $r_k \in WSEQ$, does there exist a one-to-one mapping $\pi : S \rightarrow \{i | 1 \leq i \leq M\}$ such that $Obj(\pi) \leq K$?

Theorem 1 *SRS is NP-complete.*

Proof It is easy to see that $SRS \in NP$ since a nondeterministic algorithm need only guess at an ordering of the scan registers and check in polynomial time whether the objective function is less than or equal to K. We transform the known NP-complete problem *Partition* to our problem [43].

Partition

Given a finite set A with cardinality p and a "size" $s(a) \in Z^+$ for each $a \in A$, is there a subset $\hat{A} \subseteq A$ such that $\sum_{a \in \hat{A}} s(a) = \sum_{a \in A - \hat{A}} s(a)$?

The basic units of the Partition instance are the individual elements $a \in A$ and let $B = \sum_{a \in A} s(a)$. In transforming Partition to SRS, let the local replacement for each $a \in A$ be a single register R_k with $l_k = s(a), d_k = W_1$ and $r_k = W_1$, where W_1 is a constant. We augment the set S with an additional register R_{p+1} with $l_{p+1} = 1, d_{p+1} = W_2$ and $r_{p+1} = W_2$, where $W_2 > W_1$. Hence in the corresponding instance of SRS, the set $WSEQ$ is equal to $\{W_1, W_2\}$, the set S is equal to $\{R_1, R_2, \ldots, R_{p+1}\}$ and let $K = W_1(B+1) + (W_2 - W_1)(\lfloor B/2 + 1 \rfloor)$. Clearly this instance can be constructed in polynomial time from the Partition instance.

In the above instance, the test is carried out in two sessions. Since every register in the set S is used as a driver-receiver in the first session, every permutation of the registers will require a chain cycle CC_1 equal to $(B+1)$. The first session will contribute an amount equal to $W_1(B+1)$ to the objective function. Any feasible mapping π must therefore ensure that the chain cycle utilized in the second session is equal to $(\lfloor B/2 + 1 \rfloor)$. In the second session, only register R_{p+1} will be used as a driver-receiver.

If B is an odd integer, a feasible mapping cannot be constructed in which case the desired subset for the Partition instance also cannot exist. However, if B is an even integer, every feasible mapping would require the driver and receiver distances of R_{p+1} to be equal to $(B/2 + 1)$. Hence the problem of ordering the registers to satisfy the bound K is reduced to the problem of dividing the remaining p registers into two disjoint subsets, such that the sum of the lengths of the registers in each subset is equal to $B/2$. A feasible mapping can then be constructed by placing the registers in each subset on either side of register R_{p+1}. However this can be done if and only if there is a subset $\hat{A} \subseteq A$ such that $\sum_{a \in \hat{A}} s(a) = B/2 = \sum_{a \in A - \hat{A}} s(a)$.

Thus the desired subset $\hat{A}$ exists for the instance of Partition if and only if a feasible one-to-one mapping exists for the corresponding instance of SRS. □

The decision problem derived from a search problem is no harder than the search problem, and since the decision problem has been shown to be NP-complete, the

search problem is NP-hard. The number of possible scan chain orderings is equal to $M!$. To efficiently search this solution space, we exploit certain properties of optimal solutions that reduce the overall computational complexity.

4.4.2 Restricting the Search Space

Consider registers R_1 and R_2 in the scan chain shown in Fig. 4.2. Let us interchange their positions in the chain, i.e., assign R_2 to slot 1 and R_1 to slot 2. Note that the driver and receiver distances of the remaining three registers are unchanged. As a result of the interchange, the chain cycle of TS_1 is unaffected since both R_1 and R_2 are used as drivers in this session. However the chain cycle of TS_2 decreases since the driver distance of R_2 decreases from 12 to 4. This simple example illustrates a property that can be generalized to either (1) two adjacent registers in the scan chain, or (2) two registers of equal length, provided their driver and receiver weights satisfy certain conditions.

Lemma 1 - *Adjacency Interchange Property*
Consider any two registers R_α and R_β where $d_\alpha \leq d_\beta$ and $r_\alpha \geq r_\beta$. In an optimal scan chain let $p_\alpha < p_\beta$. The positions of the two registers can be interchanged without increasing the value of the objective function if either $l_\alpha = l_\beta$ or $p_\alpha = p_\beta - 1$.

Proof The value of the objective function depends only on $WSEQ$ and the chain cycles of the sessions. It is sufficient to show that the chain cycle of any test session cannot increase. In interchanging the positions of the two registers, the driver and receiver distances of every other scan register in the chain remain the same. Since $d_\alpha \leq d_\beta$, R_β will be used as a driver in every test session in which R_α is used as a driver. Hence the interchange will not affect the drive cycle for any of these sessions. The driver distance of R_j decreases as a result of the interchange. Hence the drive cycle of every session in which R_j is solely used as a driver cannot increase. Similarly since $r_\alpha \geq r_\beta$, the receive cycle of any test session cannot increase as a result of the interchange. □

Note that the adjacency interchange property cannot be invoked on registers R_3 and R_4 in the scan chain of Fig. 4.2. Both the driver and receiver weights of R_3 are greater than the corresponding weights of R_4. The next lemma is a direct

consequence of the adjacency interchange property and is referred to as a *dominance* property in classical scheduling theory [44].

Lemma 2 If there exists k registers ($k \geq 1$) with driver (receiver) weights equal to W_n and receiver (driver) weights equal to 0, these registers can be assigned to the first (last) k slots in an optimal chain.

Proof Consider an optimal ordering of the scan registers. By successive use of the adjacency interchange property, each of the k pure drivers (receivers) can be interchanged with any register assigned to the slot on its immediate left (right). Hence the k registers can be made to occupy the first (last) k slots in the chain without increasing the value of the objective function. □

For the example in Fig. 4.2, the dominance property can be used to assign register R_2 to slot 1 and register R_5 to slot 5 of the scan chain. The adjacency interchange property provides ways to limit the size of the search space in generating an optimal solution. Its use in restricting the search space however greatly depends on the topology of the circuit, the number of kernels in the design, and the roles played by the different registers. To illustrate this, let us consider the three different circuit topologies shown in Fig. 4.3. In the three designs, all registers are assumed to be of equal length, i.e., l flip-flops.

Case 1 - The circuit in Fig. 4.3(a) consists of n kernels and all registers in the design are either pure drivers or pure receivers. The adjacency interchange property can be used to configure an optimal scan chain in time complexity polynomial in the number of registers. Consider registers R_1 and R_2 in the design. Since the driver weight of R_1 is greater than the driver weight of R_2 and their receiver weights are both equal to 0, R_1 should always be placed in a lower-numbered slot as compared to R_2. By invoking the adjacency property on every pair of registers, an optimal scan chain can be configured in the following manner. The n pure drivers are assigned to the first n slots in the chain and the n pure receivers to the last n slots. Among the drivers, the registers are ordered in terms of decreasing driver weights from slots 1 to n. Similarly, the receivers are ordered in terms of increasing receiver weights from slots $n+1$ to $2n$. Note that this solution is independent of the test lengths of the kernels provided $W_n \geq W_{n-1} \geq \ldots \geq W_1$. The optimality of the solution is

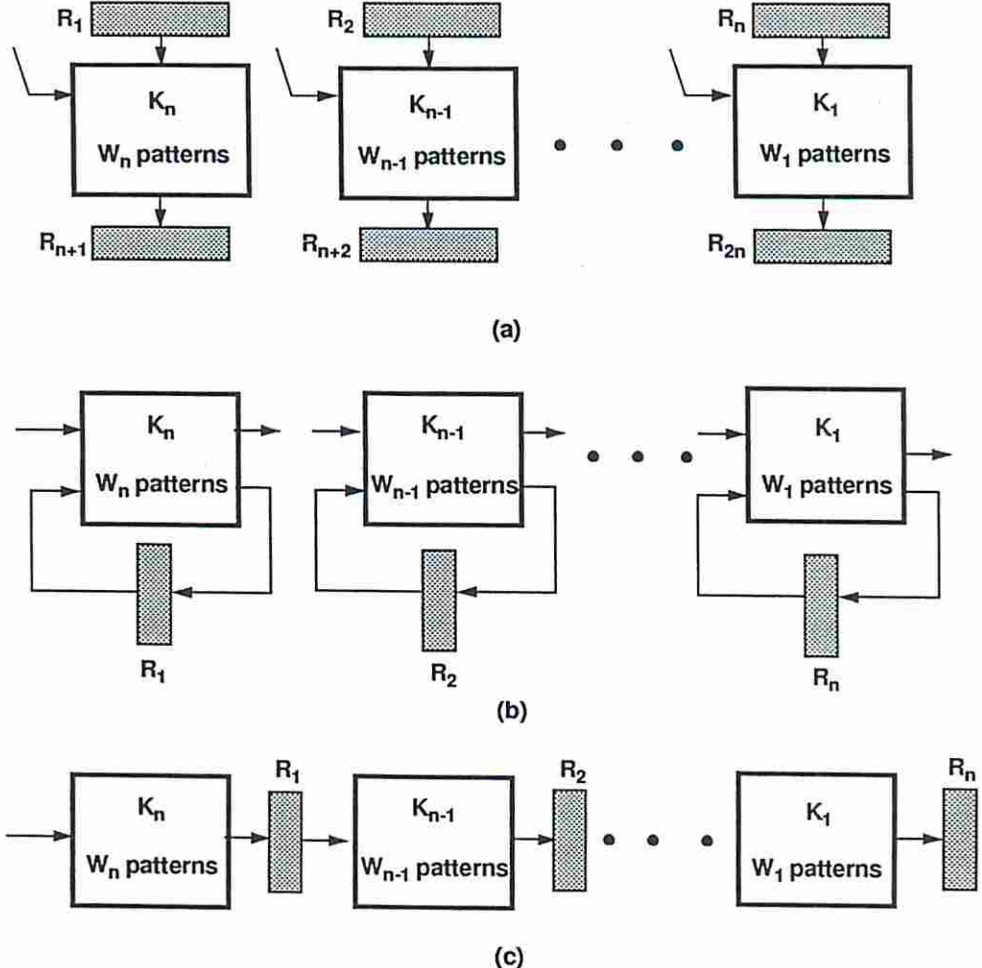


Figure 4.3: (a) Circuit topology with pure drivers and receivers. (b) Design with independent finite state machines. (c) Pipelined circuit topology.

also independent of the lengths of the scan registers, i.e., all registers need not be of equal length.

Case 2 - For the circuit in Fig. 4.3(b), all registers are driver-receivers and each of them is involved in testing a single kernel. Consider registers R_1 and R_2. Since both the driver and receiver weights of R_1 are greater than the corresponding weights for R_2, the adjacency property cannot be invoked to determine their relative positions in the chain. This observation is true for every pair of registers in the design. However all registers in the design are driver-receivers and each is involved in testing a single kernel. This property can be exploited to configure an optimal scan chain based solely on the *weights* of the registers. The weight of a register is equal to the maximum of its driver and receiver weights. Consider any session of the overlapped test scheme. To minimize the chain cycle of the session, the "best" way to order the registers is to place the registers used in the session in slots close to the middle of the chain. The remaining unused registers in the session are then placed in slots on either side of these registers. For example with $n = 4$, let us consider the third session in which $(W_3 - W_2)$ patterns are applied to kernels K_3 and K_4. To minimize the chain cycle of the session, registers R_1 and R_2 should be placed in slots 2 and 3, respectively, and registers R_3 and R_4 should be placed in slots 1 and 4, respectively. In general an optimal scan chain is obtained by placing register R_1 with weight W_n in slot $\lceil \frac{n}{2} \rceil$ of the scan chain. Registers R_2 and R_3 with weights W_{n-1} and W_{n-2}, respectively, are then placed in slots on either side of this register. In this fashion the two largest weight registers that have not been assigned are placed in slots on either side of registers that have already been placed. As in Case 1 the optimality of this solution is independent of the values of the test lengths for the different kernels. However the ordering of registers is not guaranteed to be optimal if the lengths of all registers are not equal.

Case 3 - For the pipelined circuit topology in Fig. 4.3(c), all registers are driver-receivers except for register R_n. However every register except for R_n is involved in testing two kernels. The adjacency interchange property provides no reduction in the search space. Consider registers R_1 and R_2 in the design. Since both the driver and receiver weights of R_1 are greater than the corresponding weights for R_2, it is not possible to specify their relative positions in the chain. This is true for every

pair of registers in the design. In the case of $n = 2$, the size of the search space consists only of two orderings. For both orderings, the chain cycle of the first session will be equal to $2l$. Hence to minimize the chain cycle of the second session, register R_1 should be placed in the second slot of the chain. In other words, an optimal scan chain is always given by R_2, R_1 in order from scan-in to scan-out. For values of n greater than 2, the ordering of registers in an optimal scan chain depend on the respective test lengths of the kernels. For example consider the case of $n = 3$. If $W_3 = 150, W_2 = 100$, and $W_1 = 50$, an optimal scan chain is given by R_3, R_1, R_2 in order from scan-in to scan-out. On the other hand, if W_3 assumes the value of 250, an optimal ordering of registers is given by R_3, R_2, R_1. With any ordering of registers in the chain, the chain cycle of the first session will always be equal to $3l$. Assume R_3 is assigned to either slot 2 or slot 3 in a scan chain. The position of the register can be interchanged with the register assigned to slot 1 without increasing the chain cycles of the second and third sessions. Hence in an optimal scan chain, R_3 must always occupy the first slot in the chain. However the positions of R_1 and R_2 in the remaining two slots depend on the test lengths of the kernels. If $(W_2 - W_1) \geq (W_3 - W_2)$, then in an optimal scan chain R_1 and R_2 should be placed in slots 2 and 3, respectively. The positions of the two registers must be interchanged in an optimal chain if $(W_2 - W_1) < (W_3 - W_2)$. Similarly for values of n greater than 3, the test lengths of the kernels play an important role in determining an optimal scan chain.

For more general circuit topologies, recall that the adjacency interchange property can be conditionally invoked in two ways; the first in deciding the order of two adjacent registers in the chain, and the second in specifying the relative order of two equal length registers. We will employ the former in the course of configuring the scan chain. The latter, in concert with the dominance property, can be used in a pre-processing step to induce a partial order on the set of scan registers. For two equal length registers R_i and R_j, if $d_i \leq d_j$ and $r_i \geq r_j$, there exists an optimal scan chain in which $p_i > p_j$. The relation $p_i > p_j$ can be modeled in the form of a directed arc from R_j to R_i. In this way a directed acyclic graph, referred to as the *precedence graph*, can be constructed. The vertices of the graph represent scan

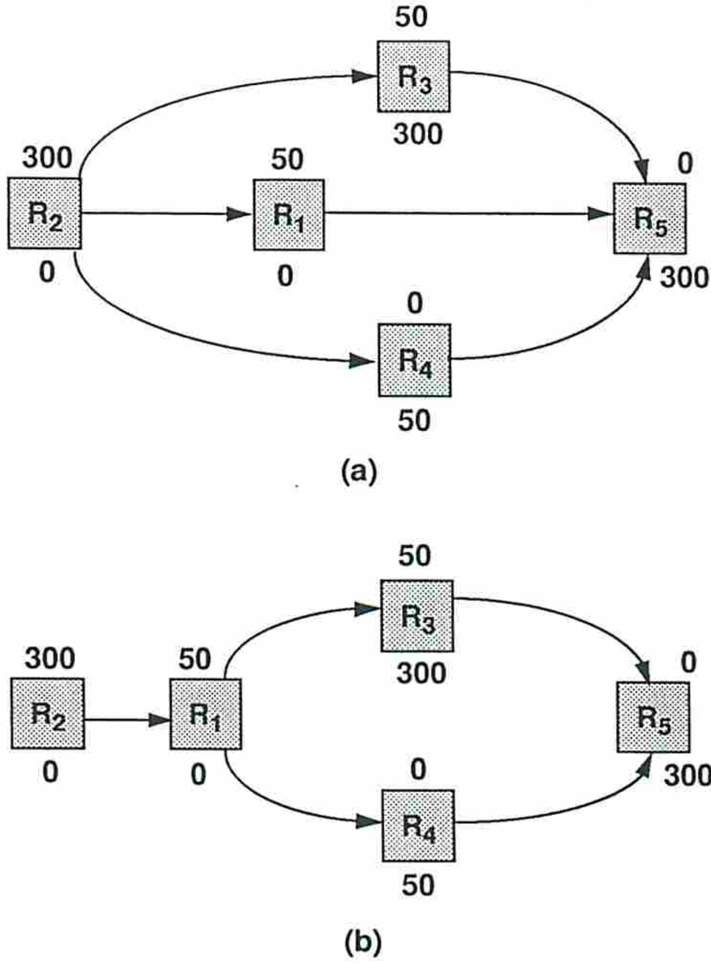


Figure 4.4: (a) Precedence graph for example design. (b) Precedence graph assuming equal length registers.

registers and the edges represent the order of sequencing the registers in an optimal configuration. Since the relation is transitive, we only retain the longest path between any two registers in the precedence graph.

For our example design the precedence graph is shown in Fig. 4.4(a). The numbers above and below each register denote its driver and receiver weights, respectively. Note that all arcs in the graph are a result of employing the dominance property. To illustrate the use of the adjacency interchange property, let us assume all the registers are of equal length. As a result a directed arc can now be introduced between R_1 and R_3, and between R_1 and R_4. The resulting precedence graph

is shown in Fig. 4.4(b). The search space for an optimal configuration is now reduced to the set of *topological orderings* of the precedence graph. In the graph of Fig. 4.4(b), the two possible topological orderings are R_2, R_1, R_3, R_4, R_5 and R_2, R_1, R_4, R_3, R_5.

The structure of the precedence graph is clearly dependent on the characteristics of the input data. It is possible for the precedence graph to contain only a single topological ordering in which case enumeration is entirely avoided. In general the precedence graph can provide a substantial reduction in the search space for an optimal configuration.

4.5 An Implicit Enumeration Algorithm

All the properties outlined in the previous section will be used in designing an implicit enumeration algorithm to generate an optimal scan chain. Before presenting the details of the algorithm, we derive lower and upper bounds on the test time. These bounds will serve a two-fold purpose. Firstly they are used to restrict the amount of explicit enumeration in the process of generating an optimal solution. More importantly the bounds can also be used in estimating the test time prior to invoking the algorithm. Their use in the latter context will be illustrated in Chapter 9 when dealing with the complete scan design system.

4.5.1 Bounds on Test Time

Consider the assignment of a register with driver (receiver) weight equal to W_n to the Mth (first) slot in the scan chain. The chain cycle of every session from TS_1 to TS_n will be equal to the length of the scan chain. Any permutation of the remaining registers in the available slots will lead to a worst case ordering which achieves the upper bound on the test time. In effect, the overlapped scheme with the minimum shift policy is reduced to the combined scheme with such an ordering of the registers.

To derive a lower bound on the test time, recall that the scan chain configuration determines the chain cycles of the sessions. To minimize the duration of a session, we need to minimize the chain cycle of the session. Since the roles played by a register might vary in different sessions, a scan chain ordering that minimizes the

chain cycle in one session need not minimize the chain cycle in the other sessions. To derive a lower bound, we will therefore determine the least possible chain cycle of each session separately. The following definition is reproduced from [35].

Definition For a given test session TS_i ($1 \leq i \leq n$), the **minimum chain cycle** MCC_i is the minimum number of clock cycles, over all possible orderings of the scan chain, required to shift in a new test pattern while shifting out the results of the previous test.

The minimum chain cycle is the lowest possible chain cycle that can be used in a session assuming the registers in the chain can be ordered solely to minimize this chain cycle. Consider our example design in Fig. 4.1. In session TS_1, a minimum of 14 clock cycles is required to shift test patterns into R_1, R_2 and R_3 which act as drivers in the session. Similarly a minimum of 8 clock cycles is required to shift out results from the receivers R_3, R_4 and R_5. The minimum chain cycle that can be employed in the session is 14. This is achieved by assigning the registers R_1, R_2, R_3, R_4, R_5 in sequence to slots 1 to 5 of the scan chain. In session TS_2, R_2 plays the role of a driver, and R_3 and R_5 are receivers. The minimum chain cycle that can be employed in the session is equal to 4, i.e, R_2 is assigned to slot 1, and R_3 and R_5 to slots 4 and 5, respectively.

To extend this idea to the general case, let L_d, L_r and L_{dr} respectively denote the sum of the lengths of the drivers, receivers and driver-receivers used in a session. Let L denote the sum of the lengths of all scan registers. In [35] expressions are obtained for the minimum chain cycle of a session assuming every scan register to be a single flip-flop. We extend the expressions to unequal length registers and derive the following general rule to determine the minimum chain cycle (MCC) of a session.

$$MCC = \begin{cases} max(L_d, L_r) & \text{if } L_{dr} = 0 \\ max(L_d, L_r, \lceil \frac{L-L_{dr}}{2} \rceil) + L_{dr} & \text{if } L_{dr} > 0 \end{cases} \quad (4.1)$$

For the case when L_{dr} is greater than 0 and $max(L_d, L_r, \lceil \frac{L-L_{dr}}{2} \rceil)$ is equal to $\lceil \frac{L-L_{dr}}{2} \rceil$, the above rule only generates a lower bound on the minimum chain cycle of the session. Evaluation of the minimum chain cycle in this case can be shown to

be an NP-hard problem. The above formula can be used to evaluate the minimum chain cycle of every test session. A lower bound on the objective function is then obtained by replacing CC_i with MCC_i in equation 3.1. Note that the minimum chain cycle of a session is independent of the scan chain configuration and only depends on the parameters of the scan registers.

4.5.2 Details of the Algorithm

We utilize an implicit enumeration technique called *best-first* search to generate an optimal chain configuration. Best-first search employs a tree organization of the solution space and uses bounding functions to curtail the enumeration process. We illustrate the algorithm by configuring an optimal scan chain for our example design. As a pre-processing step, equation 4.1 is used to evaluate the minimum chain cycle of every test session. For our example the minimum chain cycles of TS_1 and TS_2 are equal to 14 and 4, respectively. The first step in the algorithm is to employ the dominance property to assign any registers that satisfy the conditions of the lemma. In our example we assign R_2 to slot 1, and R_5 to slot 5 of the scan chain. This partial configuration forms the root of the search tree. Fig. 4.5 shows the organization of the search tree. The branching process from the root involves the assignment of the remaining registers to slot 2 of the scan chain. In doing so we restrict the possible candidates for the next slot to registers that satisfy the precedence graph. With the precedence graph of Fig. 4.4(a), three nodes are generated at level 1 of the search tree as shown in Fig. 4.5. On the other hand, if we were to use the precedence graph in Fig. 4.4(b), only a single node would be generated at level 1. The leaf nodes in the tree correspond to configurations in which all registers have been assigned to slots in the chain.

We evaluate a cost function for every node generated in the search process. This cost function will be used both as a means of directing the search as well as pruning branches of the search tree. Consider the partial configuration $R_2, R_1, _, _, R_5$ generated at level 1 of the tree. We evaluate its cost function by considering the roles of the registers in the partial configuration for each test session. In TS_1, R_1 and R_2 will be used as drivers and R_5 as a receiver. The partial configuration will therefore

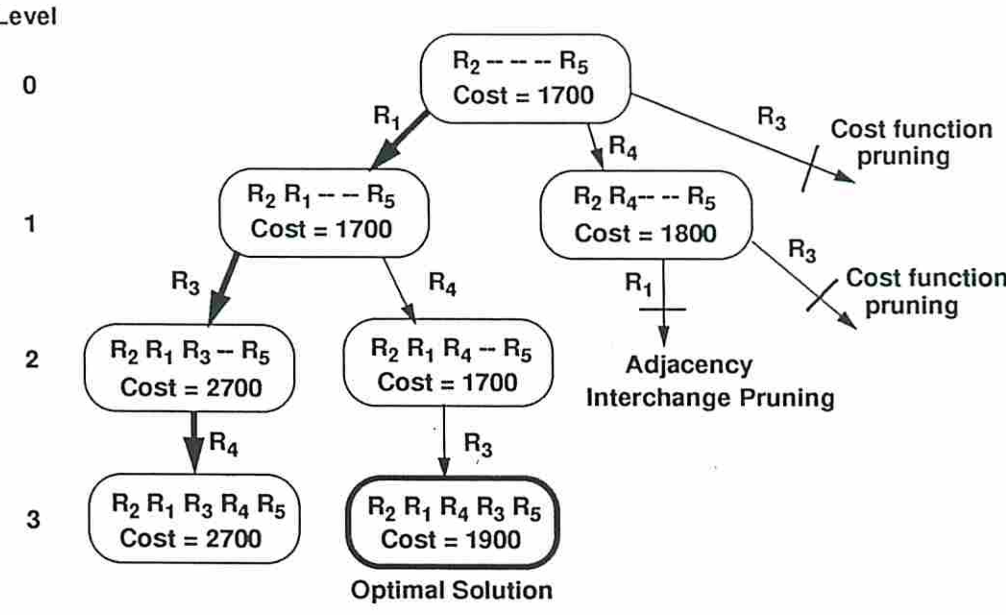


Figure 4.5: Search tree organization for example design.

require a minimum of 12 clock cycles to shift in test patterns while simultaneously shifting out test results. We compare this number with the minimum chain cycle evaluated for the session, i.e., 14. The value of CC_1 evaluated for this node is set to the maximum of the two numbers. Similarly in TS_2, the chain cycle evaluated from the partial configuration is 4, which is equal to MCC_2. Using these chain cycles, the cost function of the node is calculated by using equation 3.1, i.e., 50(14)+250(4) = 1700. The value of the objective function for any complete chain configuration derived from this node will be greater than or equal to 1700. Hence the cost function represents a lower bound on any solution generated from this node.

The search process initially traverses down a path in the search tree in a depth-first fashion to generate an initial solution. The value of the objective function for this solution is used as an initial bound to prune the search tree. In our example this corresponds to the leftmost path in the tree and the leaf node R_2, R_1, R_3, R_4, R_5 has an associated cost of 2700. At every subsequent stage in the search process, a node with minimum cost function is chosen to be expanded. In the case of a tie, the node at a higher level in the tree is chosen. To curtail the enumeration process,

we employ two pruning techniques. The first involves the use of the adjacency interchange property. Consider the node $R_2, R_4, _, _, R_5$ in the search tree of Fig. 4.5. Expanding this node will involve the placement of registers R_1 or R_3 in slot 3 of the scan chain. We can avoid the generation of the node $R_2, R_4, R_1, _, R_5$ since R_1 and R_4 can be interchanged to decrease the value of the cost function. Hence an alternate path exists in the search tree in which R_1 is assigned before R_4. The second pruning technique curtails the generation of nodes with cost function greater than the current best solution. Since the cost function of the node $R_2, R_4, R_3, _, R_5$ is greater than the current best of 2700, it is not generated. The search terminates when the node with minimum cost selected for expansion is a leaf node. In our example, the expanded nodes and their cost functions are shown in Fig. 4.5. Note that out of a possible 120 (5!) different chain configurations, we only generate two leaf nodes in evaluating an optimal scan chain. The optimal configuration corresponds to the leaf node R_2, R_1, R_4, R_3, R_5 with an associated cost equal to 1900, i.e., the chain cycle in TS_1 is equal to 18 and the chain cycle in TS_2 is equal to 4. A pseudocode of the algorithm is given below.

Generate-Optimal-Chain (Inputs: $S, WSEQ$)
begin

1. Evaluate minimum chain cycle for each test session.
2. Employ dominance property to initialize root node.
3. Generate initial solution with root node.
4. *Best-solution* = cost function of initial solution.
5. Initialize *List-of-nodes* with root node as single element.
6. While *List-of-nodes* is not empty
   - (a) Remove *Current-node* from head of *List-of-nodes*.
   - (b) If *Current-node* is a leaf node
     return *Current-node*.
   - (c) Else for each unassigned register R_i in *Current-node*
     - If (precedence graph satisfied) and (no adjacency-interchange pruning)
       Assign R_i to next slot and evaluate cost function.
     - If (no cost-function pruning)
       Insert *new-node* in sorted *List-of-nodes*.

If *new-node* is a leaf node, update *Best-solution.*

end

The above algorithm will also be used as part of the two-phase approach developed in Chapter 8 when dealing with the combined scan chaining and test scheduling problem.

4.5.3 Incorporating Routing Constraints

Recall that the routing overheads in configuring the scan chain only involve single-bit wires between adjacent registers in the chain. This would be true in the case of silicon compilers where registers are modeled as macro functions. All flip-flops in the register will typically be placed adjacent to each other. On the other hand, for standard cell and gate array placement styles, flip-flops belonging to a single register might be randomly placed in the physical layout. In employing the above algorithm, each flip-flop can be treated as a single-bit register. Additionally routing constraints can be incorporated in the algorithm depending on the manner in which these constraints are specified. For example a register might be constrained with regard to its position in the scan chain. It might have to be placed either before or after one or more registers in the chain. Such constraints could be modeled by adding or deleting arcs from the precedence graph. Alternatively if two registers need to be placed adjacent to each other in the chain, they can be merged and treated as a *composite* register in configuring the chain. Note that such a register might violate our circuit model. However the algorithm could be modified to take into account such composite registers in generating optimized solutions. If placement information about the physical location of the scan registers is made available, wire length can also be used to model routing constraints. For example an upper bound can be imposed on the wire length between any two adjacent registers in the chain. In our algorithm, while expanding a node, the possible candidates for the next available slot can be restricted to those registers that lie at a distance less than the upper bound.

4.6 Case Study

By employing the single chain methodology, large reductions in test time are obtained when the design consists of multiple kernels with different test lengths; such properties are common in designs such as data path and signal processing circuits. We utilize the *USC Data Path-I* shown in Fig. 4.6 to illustrate our methodology on a range of full and partial scan designs generated by SIESTA. The circuit consists of 7 combinational blocks, 7 registers of width 8 bits and 3 registers of width 16 bits. The number of gates within each combinational block is shown in the figure. The inputs in dotted lines to each block represent control inputs. This example will serve as a benchmark for all three of our chain methodologies.

By employing full scan, each combinational block can be treated as a kernel. SIESTA also generates partial scan designs by selecting an appropriate subset of the registers to be made scannable [6]. The kernels in these partial scan designs consist of *balanced* sequential structures composed from the combinational blocks in the circuit. For example, if registers R2, R4, R5, R8, and R10 are made scannable, two sequential kernels are obtained. The first kernel is the pipelined structure composed from the ALU, R6, Complement, R7, MUX1, R3 and MUX2. The second kernel is composed from MUX3, R1, Barrel Shifter, R9 and Multiplier. Application of a single test pattern to these kernels requires the pattern to be held in the scan registers for a number of clock cycles equal to the length of the pipeline. Table 4.1 shows seven different full and partial scan designs for the *USC Data Path-I*. The test lengths column denotes the number of test patterns required to test each kernel in the design.

To highlight the influence of design characteristics on the final test time, minor modifications are introduced to the design. In Design 5, the ALU is replaced by a larger combinational block that requires 500 test patterns to fully test it. In Designs 6 and 7, registers of appropriate width are introduced at the control inputs and primary inputs/outputs of the circuit. This represents a possible scenario in a boundary scan environment. Note that this has the effect of introducing pure drivers and receivers in the circuit. In Design 6 the entire combinational logic is treated as a single kernel, while in Design 7 each combinational block is treated as a kernel.

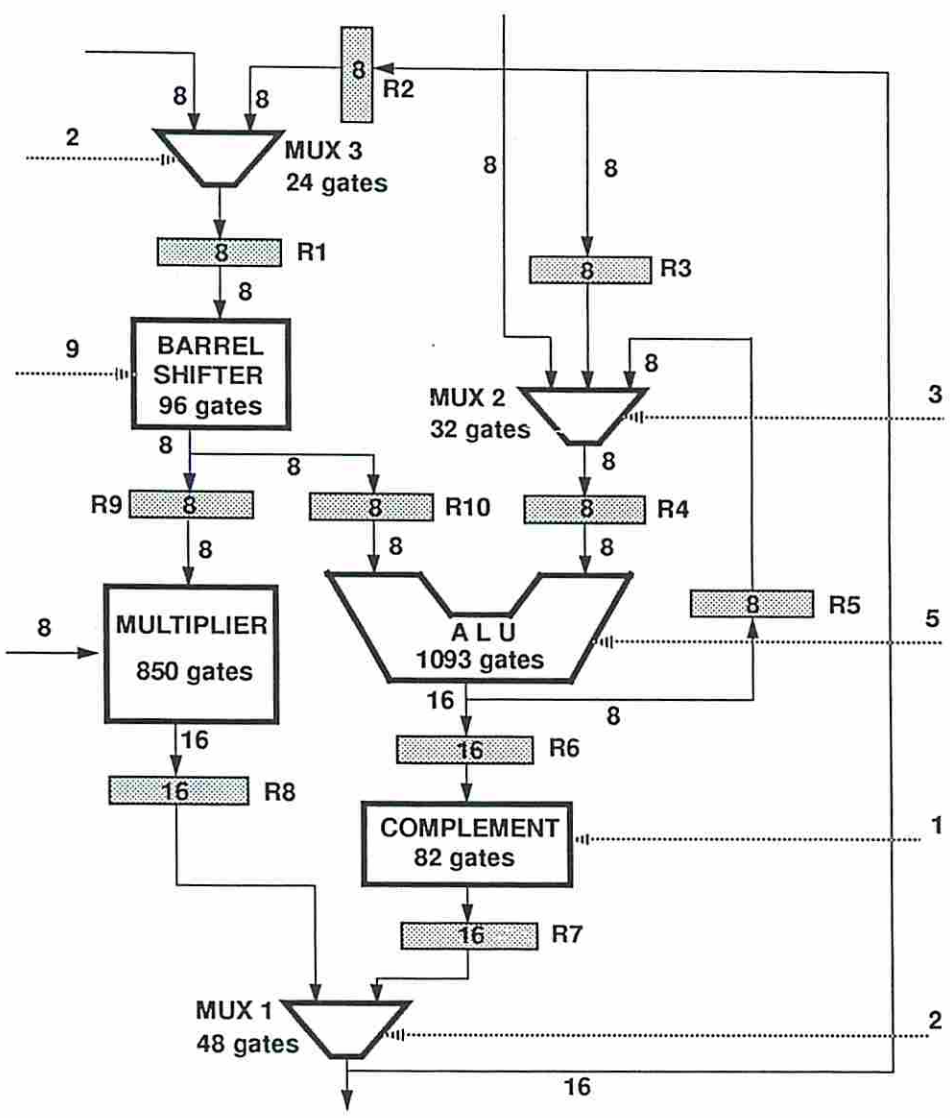


Figure 4.6: *USC Data Path-I.*

Design index	Type of scan	No. of kernels	Test lengths	No. of scan registers	Scan registers
1	Full	7	10,11,14,17 18,135,179	10	R1,R2,R3,R4,R5 R6,R7,R8,R9,R10
2	Partial	4	17,29,141,179	7	R2,R3,R4,R5,R6,R9,R10
3	Partial	4	21,29,135,179	7	R2,R4,R5,R6,R8,R9,R10
4	Partial	2	229,579	5	R2,R4,R5,R8,R10
5	Full	7	10,11,14,17 18,135,**500**	10	R1,R2,R3,R4,R5 R6,R7,R8,R9,R10
6	Full	1	221	19	All registers including primary I/O and control
7	Full	7	10,11,14,17 18,135,179	19	All registers including primary I/O and control

Table 4.1: Seven scan designs of the *USC Data Path-I*

Table 4.2 shows the results of applying the single chain algorithm to the seven scan designs in Table 4.1. The test times are in terms of the number of clock cycles. For the overlapped scheme, an optimal scan chain is configured and a minimum shift policy is used. The percentage reduction column represents the savings in test time over the combined scheme. The percentage of nodes column represents the ratio of the number of nodes created in the search process over the total search space.

Note the effect of replacing the ALU in Design 1 with a larger combinational block in Design 5; the reductions in test time increase from 59% to 70%. Even with a single kernel in Design 6, knowledge of the pure drivers and receivers is exploited to obtain a test time savings of 10%. In Design 7, by treating each combinational block as a kernel, we obtain a reduction of 70% in test time. The reduced number of nodes generated with respect to the total search space highlights the efficiency of the algorithm.

The implicit enumeration algorithm is limited by the computational requirements for designs containing a large number of registers. For example Design 7 required an hour of CPU time on a SPARC-1 workstation to generate an optimal solution. One way to reduce the computation time is to restrict the amount of enumeration during the search process. For example an upper bound can be imposed on the number of candidates for the next available slot at the lower levels of the search tree.

Design index	Test time (combined)	Test time (overlapped)	% red.	% nodes
1	18899	7803	58.71	0.01
2	11878	8182	31.12	0.4
3	13497	7689	43.03	0.7
4	30156	21756	27.86	8.3
5	52604	15828	69.91	0.01
6	35297	31745	10.06	0.0
7	28619	8687	69.6	16.3

Table 4.2: Results for scan designs of the *USC Data Path-I*

Alternatively heuristic techniques can be adopted that might potentially generate suboptimal solutions. To address the computational costs, a heuristic technique based on *neighborhood search* will be presented in Chapter 6 in the context of a reconfigurable scan chain. As we will see, the technique can also be used to configure a single scan chain in the absence of any reconfiguration.

4.7 Summary

We have presented an efficient methodology to configure a single scan chain. For the example case study, reductions as large as 70% are obtained over the combined scheme. The potential savings in test time are theoretically unbounded since it greatly depends on the test lengths of the kernels and the roles of the scan registers. Note that the methodology adds no extra hardware to the circuit and does not preclude the use of test compaction techniques to further reduce test time. We have furnished a proof showing the scan register sequencing problem to be NP-complete under the assumption that all registers are not of the same length. Detailed analysis and optimization techniques have been provided to optimally configure the scan chain. An interesting problem emerging from the analysis is to characterize the complexity of scan register sequencing when all registers are of equal length. Intuitively it seems as difficult as the unconstrained problem. Recall that the *max*

function is required to evaluate the chain cycle of a session. The complexity in deriving optimal solutions primarily arises from the non-linearity of the *max* function.

To address concerns regarding routing area, the implicit enumeration algorithm can be suitably modified to incorporate additional design constraints. The proposed methodology, in particular, can provide great dividends for scan designs composed of multiple kernels with a large variance in test lengths.

Test session	Kernels tested	Number of patterns	Chains active	Chain cycle	Test time
TS_1	K_1, K_2, K_3	30	C_1, C_2, C_3, C_4	10	330
TS_2	K_2, K_3	70	C_1, C_3, C_4	10	770
TS_3	K_3	400	C_3	8	3600

Table 5.1: Summary of test sessions for example design

while other chains are applying data to the circuit under test. This *asynchronous* operation of the chains can lead to savings both in the test time and pin count.

5.2.2 Equal Length and Optimal Chains

If a single scan chain is used to test the circuit in Fig. 5.1(a), the chain cycle used in each session is equal to 32 and hence the total test time is 500(32 + 1) + 32 = 16532 clock cycles. In Fig. 5.2, two alternative four-chain configurations are shown for the circuit. The configuration in Fig. 5.2(a) has four chains of equal length. The chain cycle used in each session will be equal to 8 and the total test time is equal to 500(8+1) + 8 = 4508 clock cycles. Consider the four-chain configuration shown in Fig 5.2(b). Test sessions TS_1 and TS_2 will employ a chain cycle of 12 since chains C_1 and C_2 are involved in testing K_1 and K_2. However in test session TS_3, only chains C_3 and C_4 are used and hence a chain cycle of 4 is sufficient to apply the remaining 400 patterns to K_3. The total test time is equal to 30(12+1) + 70(12+1) + 400(4+1) + 12 = 3312 clock cycles. A 26% reduction in test time is obtained over the configuration in Fig. 5.2(a). The reduction is a direct consequence of placing flip-flops that are frequently used, i.e., flip-flops involved in testing K_3, in the two short chains C_3 and C_4. In fact the configuration shown in Fig. 5.2(b) represents an optimal four-chain configuration in terms of minimizing the test time.

Depending on the number of available I/O pins, further reductions in test time are possible by increasing the number of scan chains. In the limiting case, this would lead to full parallel scan providing access to every scan flip-flop in a single clock cycle. However the severe pad limitations, huge number of storage elements in current VLSI designs and the associated routing overhead make the use of full

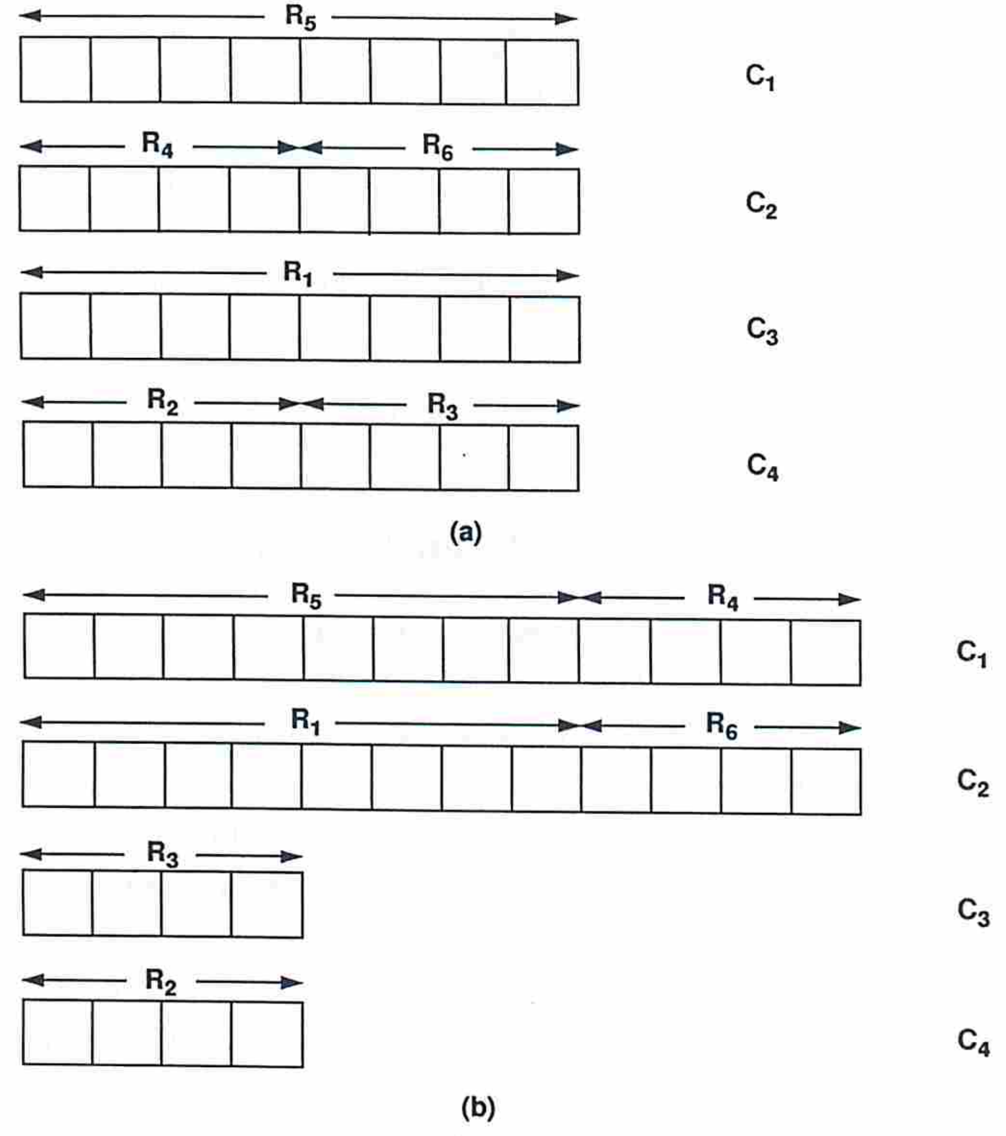


Figure 5.2: Two different chain configurations for example design. (a) Equal length chains (test time = 4508 clock cycles). (b) Optimal four-chain configuration (test time = 3312 clock cycles).

parallel scan impractical. Given the number of chains to be configured as an input design parameter, the goal of our approach is to generate optimal configurations as in Fig. 5.2(b). The test time improvements over configurations with equal length chains are clearly dependent on the number and test lengths of the kernels in the circuit as well as the number of flip-flops involved in testing each kernel. If the circuit contains a single kernel, an optimal configuration will consist of equal length chains. At the other end of the spectrum, consider a circuit consisting of two kernels with test lengths in the ratio of 50:1. Let the number of flip-flops involved in testing the kernels be in the ratio of 1:50. By placing the flip-flops involved in testing the kernel with the larger test length in one or more separate chains, reductions as large as 90% are possible by configuring optimal chains as opposed to equal length chains.

The simple division of flip-flops into equal length chains permits an arbitrary assignment and ordering of the flip-flops in the chains. In configuring optimal chains with the flush policy, we constrain the assignment of the flip-flops to the chains but place no restrictions on the order of flip-flops within each chain. Hence we retain the flexibility to arbitrarily order the flip-flops within a chain to minimize routing overhead. Although our main objective will be to minimize the test time, we will indicate ways to incorporate routing constraints in the optimization techniques.

5.3 Model of the Problem

Let k denote the number of scan chains to be configured and let N denote the total number of scan flip-flops, i.e., $N = \sum_{j=1}^{M} l_j$. Let r_m $(1 \leq m \leq N)$ denote a single flip-flop and $\mathcal{F} = \{r_1, r_2, \ldots, r_N\}$ denote the set of N scan flip-flops. All flip-flops belonging to a particular register inherit the driver and receiver weights of the register. With the flush policy however, we do not distinguish the role of a register as being a driver or receiver. Hence for each flip-flop r_m we can replace the two weights with a single weight parameter w_m given by the maximum of its driver and receiver weights. For example in the design of Fig 5.1, all 4 flip-flops in register R_2 will have a weight parameter equal to $max(100, 500)$, i.e., 500. A flip-flop r_m with weight parameter $w_m = W_i$ will be used in every test session from TS_1 to TS_i.

Intuitively the weight parameter characterizes the frequency of usage of the flip-flop in the test application process.

The optimization problem is to determine an assignment of the scan flip-flops to the chains so as to minimize the chain cycles in each session. Recall that the chain cycle of a session is determined by the chains that are active in the session. We therefore need to relate the scan chains to their roles in a session.

Definition A *multiple scan chain configuration* of k chains, denoted by $\mathcal{MC}$, is defined as a k-way disjoint partition of the flip-flops in the set $\mathcal{F}$ and is represented by an ordered set of scan chains $\{C_1, C_2, \ldots, C_k\}$. Each scan chain is a subset of the flip-flops in $\mathcal{F}$ such that $C_1 \cup C_2 \ldots \cup C_k = \mathcal{F}$ and $C_i \cap C_j = \emptyset$, where $1 \leq i, j \leq k$ and $i \neq j$.

In addition we characterize a chain C_j $(1 \leq j \leq k)$ by means of the following two parameters.

- $l(C_j)$: the length of the chain in terms of the number of flip-flops.
- $W(C_j)$: the weight of the flip-flop with maximum weight parameter in the chain, i.e.,

$$W(C_j) = \max_{r_m \in C_j} (\text{weight parameter of } r_m).$$

Without loss of generality, we assume the chains in $\mathcal{MC} = \{C_1, C_2, \ldots, C_k\}$ to be ordered in terms of non-increasing lengths, i.e., $l(C_1) \geq l(C_2) \geq \ldots \geq l(C_k)$. The four-chain configuration $\{C_1, C_2, C_3, C_4\}$ in Fig. 5.1(b) is shown again in Fig. 5.3(a) and the weight parameters of the flip-flops and the lengths and weights of each of the chains are displayed. Analogous to a flip-flop, if a chain has weight equal to W_i, it will be active in every session from TS_1 to TS_i. For example in Fig. 5.3(a), since the weight of C_1 is equal to 100, it will be active in sessions TS_1 and TS_2.

The chain cycle CC_i is equal to the length of the longest chain active in session TS_i. This corresponds to the length of the longest chain with weight greater than or equal to W_i. Hence the following expression can be used to evaluate the chain cycle of test session TS_i.

$$CC_i = max[\, l(C_j) | 1 \leq j \leq k \text{ and } W(C_j) \geq W_i \,].$$

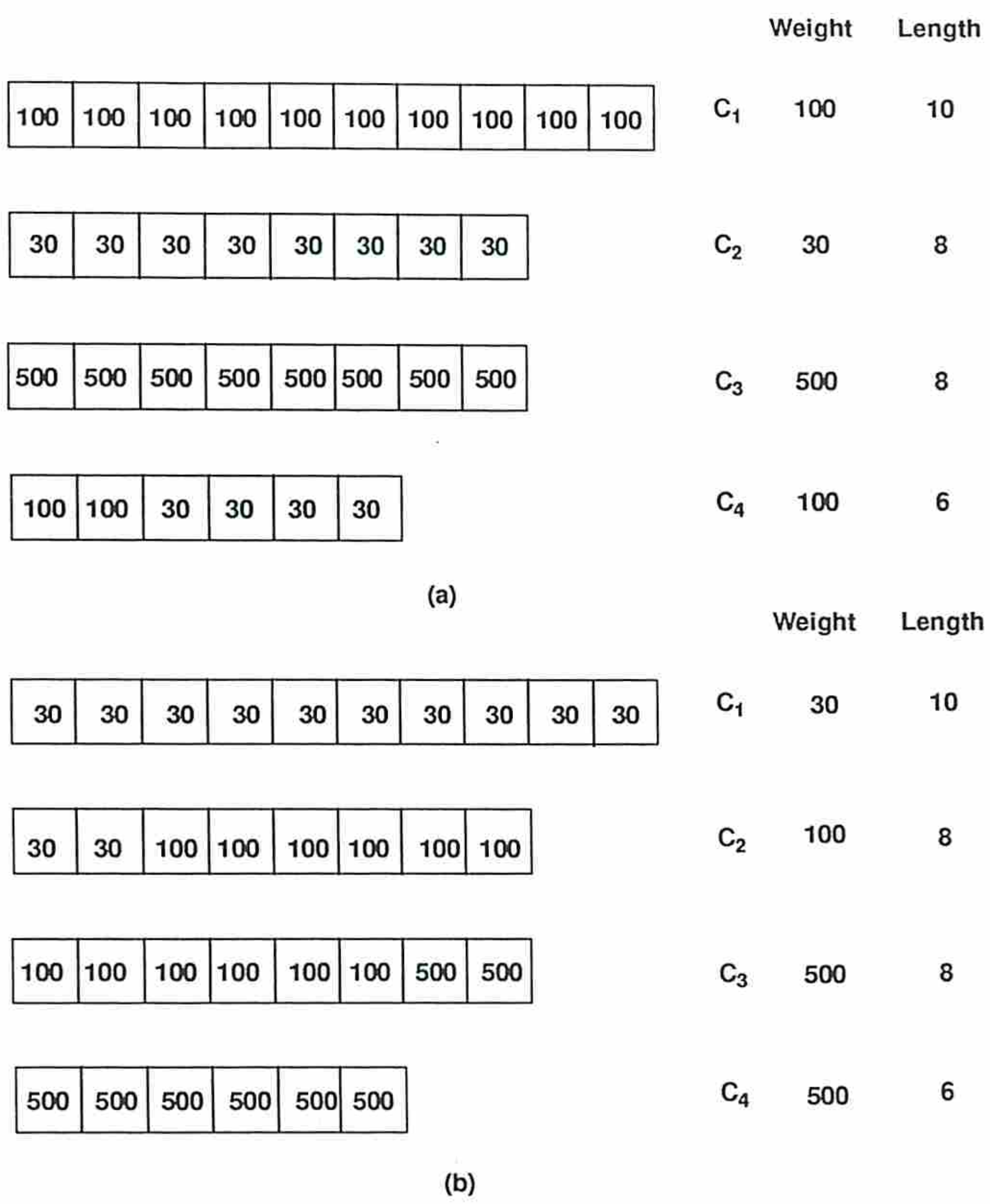


Figure 5.3: (a) Four-chain configuration with flip-flop and chain parameters. (b) Transformed configuration.

The chain cycles can be used in equation 3.1 to determine the total test time. In the next section we characterize the structure of an optimal chain configuration to provide an alternative means of evaluating the total test time.

5.4 Restricting the Search Space

The search problem to optimally configure multiple scan chains can be formally stated as follows.

Multiple scan chain problem (MSC): Given the set $\mathcal{F} = \{r_1, r_2, \ldots, r_N\}$ of N scan flip-flops with $w_m \in WSEQ$ ($1 \leq m \leq N$) and given that k chains are to be configured, find a multiple chain configuration $\mathcal{MC} = \{C_1, C_2, \ldots, C_k\}$ such that the test time with the flush policy is the minimum over all k-chain configurations.

By assumption, $l(C_1) \geq l(C_2) \geq \ldots \geq l(C_k)$. Since each flip-flop can be placed in any of the k chains, the size of the search space is approximately equal to k^N. However it is possible to reduce the search space by considering a limited set of multiple chain configurations that satisfy a certain property. We first illustrate the property by means of the configuration shown in Fig. 5.3(a). Without changing the length of any chain, let us reassign the flip-flops among the chains so that $W(C_4) \geq W(C_3) \geq W(C_2) \geq W(C_1)$. Fig. 5.3(b) shows the transformed configuration in which every flip-flop in C_{j+1} has weight greater than or equal to every flip-flop in C_j ($1 \leq j \leq 3$). As a result of this transformation the total test time decreases. Since all four chains are active in TS_1, the chain cycle in the first session is unchanged. However in TS_2, only chains C_2, C_3 and C_4 are active in the new configuration and hence the chain cycle decreases from 10 to 8. In TS_3, chains C_3 and C_4 are active and the chain cycle remains equal to 8. The reassignment of flip-flops among the chains can be done for any multiple chain configuration without increasing the test time. To prove this in the general case, we make use of the following definition.

Definition A chain **dominates** a test session if it determines the chain cycle of the session, i.e., the chain is the longest among all chains active in the session.

Lemma 3 Consider any multiple chain configuration $\mathcal{MC} = \{C_1, C_2, \ldots, C_k\}$. Without changing the length of any chain, rearrange the flip-flops in the chains so

that $W(C_1) \leq W(C_2) \leq \ldots \leq W(C_k)$. A new configuration $\mathcal{MC}_{new}$ is generated in which every flip-flop in C_{j+1} has weight greater than or equal to the weight of every flip-flop in C_j $(1 \leq j < k)$. The test time of the resulting configuration is less than or equal to the test time of $\mathcal{MC}$.

Proof It is sufficient to prove that the chain cycle of any test session cannot increase as a result of the transformation. Note that the reassignment of flip-flops preserves the length of each chain in the configuration. Since all chains are active in the first test session TS_1, the chain cycle of this session is unchanged. Consider any test session TS_i $(2 \leq i \leq n)$. In $\mathcal{MC}$ let us assume chain C_α dominates the session. In other words, all flip-flops of weight greater than or equal to W_i will lie in chains C_γ such that $\alpha \leq \gamma \leq k$. In the transformed configuration $\mathcal{MC}_{new}$, these flip-flops will still be assigned to chains of index greater than or equal to α. Hence the chain cycle of the session will be less than or equal to $l(C_\alpha)$. □

The above lemma provides a way to significantly reduce the search space for an optimal configuration. We need only consider configurations in which every flip-flip in chain C_{j+1} is of weight greater than or equal to the weight of every flip-flop in C_j $(1 \leq j < k)$. This intuitively satisfies our notion of assigning the frequently used flip-flops, i.e., flip-flops with large weight factors, to shorter chains. Hence we can restrict our attention to multiple chain configurations $\{C_1, C_2, \ldots, C_k\}$ such that $l(C_1) \geq l(C_2) \geq \ldots \geq l(C_k)$, and $W(C_1) \leq W(C_2) \leq \ldots \leq W(C_k)$.

To simplify the explanation of the algorithm to be developed, we provide an alternative method to evaluate the test time solely based on the weights and lengths of the chains in the configuration. Consider the transformed configuration shown in Fig. 5.3(b). Recall that $WSEQ = \{30, 100, 500\}$. In TS_1, chain C_1 will dominate the session and hence the chain cycle of the session is equal to 10. The contribution of the session to the total test time is equal to 30(10), i.e., $W(C_1)l(C_1)$. Similarly chain C_2 will dominate test session TS_2 and the duration of the session will be $(100 - 30)8$, i.e., $(W(C_2) - W(C_1))l(C_2)$. Finally, chain C_3 will dominate TS_3 and the session will have duration equal to $(500 - 100)8$, i.e., $(W(C_3) - W(C_2))l(C_3)$. Note that chain C_4 will not dominate any session but without affecting the total test time we can add the expression $(W(C_4) - W(C_3))l(C_4)$ which evaluates to 0 since $W(C_4)$ is equal to $W(C_3)$. To account for the time to shift out the final results, the

length of C_1, i.e., 10 is added to the total test time. Hence the weights and lengths of the chains in the configuration are sufficient to evaluate the total test time. This analysis is stated in a generalized form below.

Lemma 4 Given a multiple chain configuration $\mathcal{MC} = \{C_1, C_2, \ldots, C_k\}$, the total test time can be evaluated from the weights and lengths of the chains and is given by the following expression. Define $W(C_0) = 0$.

$$\mathcal{T} = \sum_{j=1}^{k} [W(C_j) - W(C_{j-1})] l(C_j) + l(C_1). \tag{5.1}$$

Proof In $\mathcal{MC}$, consider the weight of C_1. If $W(C_1) = W_i$, every test session from TS_1 to TS_i will employ a chain cycle equal to $l(C_1)$ and the total duration of these sessions will be $W(C_1)l(C_1)$. Consider chain C_2 in the configuration. By assumption, $W(C_2) \geq W(C_1)$. If $W(C_2) = W_i$, chain C_2 will not dominate any test session and its contribution in the above equation will evaluate to 0. Else if $W(C_2) = W_j$, where $j > i$, every session from TS_{i+1} to TS_j will employ a chain cycle equal to $l(C_2)$. The contributions of these sessions will be $(W(C_2) - W(C_1))l(C_2)$. A similar argument can be used to show that the duration of every session from TS_{j+1} to TS_n is obtained by evaluating the expression $(W(C_\alpha) - W(C_{\alpha-1})l(C_\alpha)$ for each of the remaining chains C_α $(3 \leq \alpha \leq k)$. The final term $l(C_1)$ denotes the time to flush the final result from all scan chains. □

Note that the summation in equation 5.1 iterates through the *chains* in the configuration as opposed to the *sessions* of the test scheme as in equation 3.1. All information required to calculate the test time is embodied in the weight factors of the N flip-flops used to configure the k chains. Since we will only consider configurations that satisfy the conditions of Lemma 3, equation 5.1 can be used to evaluate the test time of every such configuration.

5.5 A Polynomial Time Algorithm

5.5.1 Properties of Optimal Configurations

Let the N flip-flops in the set $\mathcal{F}$ be sorted in terms of non-increasing weights. The problem of generating k-chain configurations can be viewed in terms of placing $k-1$ partitions ("dividers") in this sorted list. To illustrate the idea, consider the sorted list of 10 flip-flops shown in Fig. 5.4(a). The weight parameters of all the flip-flops are shown. We use this example to construct an optimal configuration of 3 chains denoted by $\{C_1, C_2, C_3\}$. The chains are configured in top-down fashion starting with chain C_3. Since C_3 is the shortest chain in the configuration, the placement of the first partition is restricted to the first three positions in the list. This results in three possible lengths for C_3, i.e., 1, 2 and 3. For each of these choices, we can evaluate the contribution of C_3 to the total test time. For example, let $l(C_3)$ be equal to 2. The weight of C_3 will be 100 and irrespective of the assignment of the remaining 8 flip-flops to chains C_2 and C_1, the weight of chain C_2 will be 60. Hence the contribution of C_3 can be evaluated as $(100-60)2 = 80$. For each of the possible lengths for C_3, we can now determine the corresponding lengths for C_2. For example if $l(C_3)$ is equal to 2, the possible lengths for C_2 are 2, 3 and 4. This corresponds to placement of a partition in positions 4, 5 and 6 in the sorted list. This constraint on the length of C_2 follows from the fact that $l(C_2)$ must be less than or equal to $l(C_1)$ and greater than or equal to $l(C_3)$. In this way, we can enumerate every configuration and explicitly evaluate the test time for each of them. The optimal configuration in our example corresponds to the three chains shown in Fig. 5.4(b). The test time is evaluated by using equation 5.1, i.e., $50(4) + (60-50)4 + (100-60)2 + 4 = 324$. Note that in the process of enumeration we only consider configurations that satisfy Lemma 3.

The drawback of explicitly enumerating every possible configuration is that it results in an algorithm with time complexity exponential in the number of chains. To optimally configure k chains with N flip-flops, the algorithm has worst case complexity of order $O(N^k)$. *Dynamic programming* [45] can be used to dramatically reduce the complexity of the algorithm. To justify the use of dynamic programming, we exploit two properties that arise from the above enumerative algorithm.

The first property is observed by characterizing the structure of an optimal configuration. In the example of Fig. 5.4(a), let us assume we know the length of chain C_3 is equal to 2 in an optimal configuration. The contribution of chain C_3 can be evaluated and the problem is now reduced to configuring the remaining 8 flip-flops into 2 chains, with the additional constraint that each of these chains must be of length greater than or equal to 2. We can treat this as an independent subproblem. The test time contribution of the two chains obtained by solving this subproblem is then added to the contribution of chain C_3 to generate a solution to the original problem. Extending this to the general case, if we know the length of chain C_k in an optimal configuration, the contribution of chain C_k can be evaluated irrespective of how the remaining $N - l(C_k)$ flip-flops are assigned to the $k-1$ chains. The problem is now reduced to configuring the remaining $k-1$ chains using the $N-l(C_k)$ flip-flops while ensuring that every chain is of length greater than or equal to $l(C_k)$. The optimal test time is then obtained by adding the contribution of C_k to the contributions of these $k - 1$ chains. *The key observation is that the contributions of these $k - 1$ chains to the total test time should be minimized.* In other words, the subproblem of configuring the $k - 1$ chains, each of length at least equal to $l(C_k)$, should be optimally solved. Else, if there exists an alternative way of configuring the $k - 1$ chains whose contribution is less, we can add chain C_k to this configuration to generate a solution whose test time is less than the optimum: a contradiction. Thus an optimal solution to our problem contains within it optimal solutions to subproblem instances: one of the hallmarks of the applicability of dynamic programming.

We introduce the following notation to define the test time of an optimal configuration recursively in terms of the optimal solutions to subproblems. Let $\mathcal{P}(A, \beta, \gamma)$ represent the problem of generating an optimal multiple chain configuration, where A represents the set of flip-flops used to configure the chains, β represents the number of chains to be configured, and γ represents the lowest possible length of a chain in the configuration. The test time of an optimal configuration obtained by solving $\mathcal{P}(A, \beta, \gamma)$ is denoted by $\mathcal{T}_{opt}(A, \beta, \gamma)$. Since our overall problem is to assign the flip-flops in the set $\mathcal{F}$ to k chains with the lowest possible length of any chain in the configuration equal to 1, we denote it by $\mathcal{P}(\mathcal{F}, k, 1)$. To compute $\mathcal{T}_{opt}(\mathcal{F}, k, 1)$, let us assume we know the length of chain C_k in an optimal configuration. The

contribution of C_k could then be evaluated and the total test time, divided into two components, would be equal to

$$(W(C_k) - W(C_{k-1}))l(C_k) + \mathcal{T}_{opt}(\mathcal{F} - C_k, k-1, l(C_k)).$$

Note that $\mathcal{F} - C_k$ signifies the set difference operation. The above equation assumes a known value for the length of C_k. Since C_k is the shortest chain in a configuration, its length can only vary from 1 to $\lfloor \frac{N}{k} \rfloor$. Thus the optimal test time is given by

$$\mathcal{T}_{opt}(\mathcal{F}, k, 1) = \min_{1 \leq l(C_k) \leq \lfloor \frac{N}{k} \rfloor} \{ (W(C_k) - W(C_{k-1}))l(C_k) + \mathcal{T}_{opt}(\mathcal{F} - C_k, k-1, l(C_k)) \}.$$

Note that $W(C_{k-1})$ in the above expression depends on the value of $l(C_k)$, while $W(C_k)$ is independent of $l(C_k)$. In the example of Fig. 5.4(a), if $l(C_3)$ is equal to 1, $W(C_2)$ is equal to 100 and if $l(C_3)$ is equal to 2 or 3, $W(C_2)$ is equal to 60. However in all three cases the weight of C_3 is equal to 100.

Similarly, we can recursively define the optimal solutions to subproblems. For a general subproblem $\mathcal{P}(A, \beta, \gamma)$, let the cardinality of the set A be equal to α. The possible lengths for the chain C_β can vary from γ to $\lfloor \frac{\alpha}{\beta} \rfloor$. In each case we can evaluate the contribution of C_β irrespective of how the remaining chains are configured. This is added to the contribution of an optimal solution to $\mathcal{P}(A - C_\beta, \beta - 1, l(C_\beta))$ and the configuration with minimum test time is chosen as the optimal. For our example, consider the subproblem of using the 8 lowest-weight flip-flops to configure two chains of length greater than or equal to 2. The possible lengths for chain C_2 are equal to 2, 3 and 4. In each case, we add the contribution of C_2 to the contribution from an optimal solution to the single-chain subproblem consisting of the remaining flip-flops. Among these three cases, the configuration with minimum test time is chosen as the optimal. Note that evaluation of a subproblem consisting of a single chain is trivial. In our example, if $l(C_2)$ is equal to 4, C_1 will consist of the remaining 4 lowest-weight flip-flops and its contribution is evaluated as $50(4) + 4 = 204$. The final term equal to 4 represents the time to shift out the final results, i.e., the length

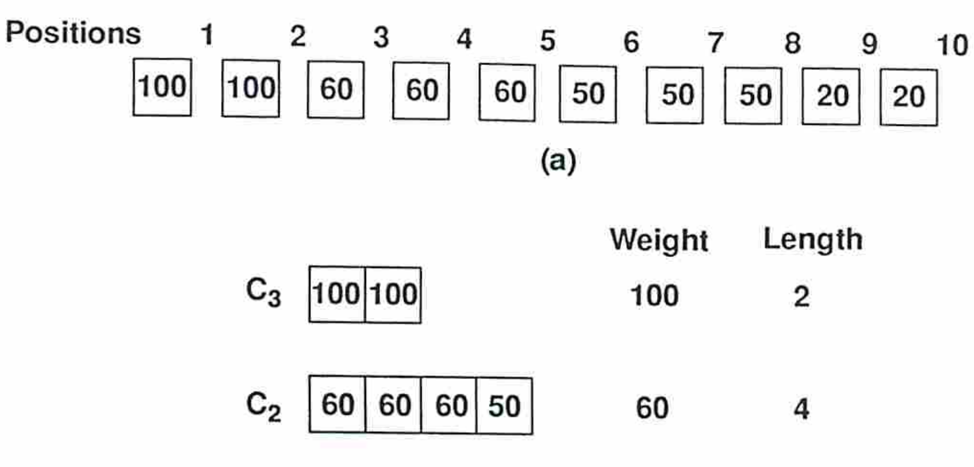
of the longest chain in the configuration. The termination condition of shifting out the final results is taken into account when solving single-chain subproblems.

The second property to be observed in using dynamic programming is that there are a relatively few subproblems to be solved. In the process of exhaustively enumerating every possible configuration, each subproblem is encountered many times. *The exponential complexity arises from repeatedly solving the same subproblem.* For example, the subproblem of configuring a single chain with the 5 lowest-weight flip-flops is solved twice in the process of enumeration, i.e., with $l(C_3) = 1$ and $l(C_2) = 4$, and again with $l(C_3) = 2$ and $l(C_2) = 3$. Dynamic programming takes advantage of this property of overlapping subproblems to reduce the computational costs of generating an optimal configuration.

5.5.2 Dynamic Programming Technique

In essence, dynamic programming calculates the optimal solutions to all subproblems once and stores these answers in a table. The optimal solution to $\mathcal{P}(\mathcal{F}, k, 1)$ is then easily evaluated by utilizing these stored values. We will illustrate the main concepts by configuring an optimal three-chain configuration for the 10 flip-flops shown in Fig. 5.4(a). The table shown in Fig. 5.4(c) is used to implement the dynamic programming technique. The flip-flops are sorted in order of non-decreasing weight and are shown above the table.

In contrast to the top-down approach used in the enumerative algorithm, we employ a bottom-up approach to configuring the chains. The table is filled by first considering all single-chain subproblems involving chain C_1. The first row in the table is used to store optimal solutions to these subproblems. Recall that C_1 will always contain the lowest-weight flip-flops. Since it is the longest chain in any configuration, its length can vary from $\lceil \frac{10}{3} \rceil$ to 8 (at least one flip-flop must exist in each of the remaining two chains). For each of these lengths, the contribution of C_1 is evaluated and stored in the appropriate entry. For example, if $l(C_1) = 5$, then the weight of C_1 is equal to 50 and its contribution to the total test time is $50(5) + 5 = 255$. This information is stored as a 2-tuple (255,5) where the first value denotes the



C1 | 50 | 50 | 20 | 20 | 50 | 4

(b)

Sorted List of 10 flip-flops

Flip-flop weights	20	20	50	50	50	60	60	60	100	100
Chains	1	2	3	4	5	6	7	8	9	10
C_1				4 ? ? C_1 C_2 C_3 (204, 4)	5 ? ? C_1 C_2 C_3 (255, 5)	6 ? ? C_1 C_2 C_3 (366, 6)	7 ? ? C_1 C_2 C_3 (427,7)	8 ? ? C_1 C_2 C_3 (488,8)		
C_1 , C_2							4 3 ? C_1 C_2 C_3 (234,3)	4 4 ? C_1 C_2 C_3 (244,4)	5 4 ? C_1 C_2 C_3 (455, 4)	
C_1 , C_2, C_3										4 4 2 C_1 C_2 C_3 (324, 2)

Table Entry (A, B) - A : test time of the partial configuration
B : length of the shortest chain in the partial configuration

(c)

Figure 5.4: (a) Sorted list of 10 flip-flops. (b) Optimal three-chain configuration. (c) Dynamic programming table.

test time contribution of C_1 and the second denotes the length of C_1. This 2-tuple is stored in the appropriate entry in the first row of the table.

Each entry in the second row of the table corresponds to subproblems involving the chains C_1 and C_2. The number of flip-flops used in configuring two chains must be at least 7 (if not chain C_3 would be of a greater length than either C_1 or C_2) and at most 9 since C_3 must contain at least one flip-flop. Let us study how the table entry enclosed in a bold rectangle is derived; for this entry, we need to assign the 8 lowest-weight flip-flops to the two chains. Let A denote the set of these 8 lowest-weight flip-flops. Since 8 flip-flops are being assigned to chains C_1 and C_2, the length of chain C_3 must be equal to 2. Hence the subproblem that needs to be optimally solved is denoted by $\mathcal{P}(A, 2, 2)$. The length of chain C_2 can assume values of 2, 3 and 4 and the corresponding lengths for chain C_1 are 6, 5 and 4. We consider the three cases separately to determine the optimal solution to the subproblem. In each case, we add together the contribution of chain C_2 and the appropriate value in the first row of the table. For example, if $l(C_2)$ is equal to 4, the test time is evaluated as $(60 - 50)4 + 204 = 244$. Note that the contribution of C_1 is directly obtained from the first row of the table. Among the three cases, the solution with least test time corresponds to the 2-tuple (244,4) where 244 denotes the optimal test time and 4 denotes the length of C_2 in the optimal configuration. This 2-tuple is stored in the highlighted table entry. The arrows from each entry in row 2 point to the single-chain subproblems used in evaluating the optimal solution for the respective table entry.

Note that we only needed to solve a single subproblem $\mathcal{P}(A, 2, 2)$ for the highlighted table entry in the second row since the length of chain C_3 was known to be equal to 2. However in the general case each subproblem of the form $\mathcal{P}(A, 2, \gamma)$ is optimally solved and stored in the table entry. The value of γ varies from the lowest possible length of C_2, i.e., 1 to its greatest possible length, i.e., 4. For example, in evaluating $\mathcal{P}(A, 2, 1)$, the length of chain C_2 can assume values of 1, 2, 3 and 4 and the optimal solution is obtained by evaluating each of the four cases and choosing the one with minimum test time. However, in evaluating $\mathcal{P}(A, 2, 4)$, chain C_2 can only assume a single length of 4 and the optimal solution corresponds to the test time of this single configuration.

Finally, the third row in the table will possess a single entry in the last column for distributing all 10 flip-flops among the three chains. For each of the three possible lengths of C_3, i.e., 1, 2 and 3, the contribution of C_3 is added to the appropriate value stored in the 2-tuples in the second row, as shown by the arrows. For example, if $l(C_3)$ is equal to 2, the test time of the configuration is evaluated as $(100-60)2+244 = 324$. The optimal test time corresponds to the minimum of the three possible cases and the optimal configuration is easily derived from the table. In our example, the optimal configuration and test time are shown in the single entry in row 3 of the table.

The following procedure represents the dynamic programming technique used to generate an optimal k-chain configuration given the set $\mathcal{F}$ of N scan flip-flops.

Generate-Optimal-Configuration (Inputs: $k, \mathcal{F}$)

1. Sort the N flip-flops in order of non-decreasing weight.
2. Evaluate optimal solutions to required single-chain subproblems and store the 2-tuples in the appropriate entries in the first row of the table.
3. For row $\beta = 2$ to k
   (a) Generate the lower and upper bounds on the number of flip-flops to be distributed in β chains $\{C_1, \ldots, C_\beta\}$. Denote these bounds as col_{min} and col_{max}, respectively.
   (b) For $\alpha = col_{min}$ to col_{max}
      i. Let A denote the set of α lowest-weight flip-flops.
      ii. For $\gamma = \lfloor \frac{\alpha}{\beta} \rfloor$ down to 1
         - Let $l(C_\beta) = \gamma$.
         - Add contribution of C_β to appropriate entry in previous row.
         - If $(\gamma < \lfloor \frac{\alpha}{\beta} \rfloor)$, compare with test time of $\mathcal{P}(A, \beta, \gamma + 1)$.
         - Store test time of $\mathcal{P}(A, \beta, \gamma)$ and length of C_β as a 2-tuple.
4. The optimal test time is stored in row k and column N and the optimal configuration is derived from the table.

5.5.3 Complexity Analysis

Given N flip-flops to be optimally configured into k chains, the worst case time complexity of the above algorithm is polynomial in the number of flip-flops. Sorting the N flip-flops using any standard sorting algorithm is clearly upper bounded by a time complexity of $O(NlogN)$. To apply the dynamic programming procedure, a table with k rows and N columns is used. Consider a table entry in row β $(2 \leq \beta \leq k)$ of the table. To optimally evaluate all subproblems $\mathcal{P}(A, \beta, \gamma)$, the length of chain C_β is varied from the maximum possible value of γ to 1. In each case computing the test time is of constant order since it involves a single addition of the contribution of C_β to a stored value in the previous row. The number of different possible values for γ in any table entry is upper bounded by $\lfloor \frac{N}{2} \rfloor$. Hence updating a single table entry is in the worst case of order $O(\lfloor \frac{N}{2} \rfloor)$. Since there are k rows in the table and at most N table entries per row, the worst case time complexity of the algorithm is $O(kN^2)$.

5.6 Experimental Results

5.6.1 Influence of Design Characteristics

The savings in test time by configuring optimal chains as compared to equal length chains greatly depends on the characteristics of the circuit under test. In particular the test lengths of the kernels, and the number of flip-flops involved in testing each kernel affect the respective test times. To illustrate this we utilize the circuit shown in Fig. 5.5. The circuit has six combinational kernels and thirteen registers with each register labeled with the number of flip-flops it contains. The primary inputs/outputs at each kernel are not shown in the figure.

A full scan methodology is used in testing the circuit. We use three variations of this basic circuit by permuting six test lengths in different ways among the kernels. For each of the three cases, Table 5.2 shows the test length and number of flip-flops associated with each kernel. Based on the definition of the weight parameter, if a

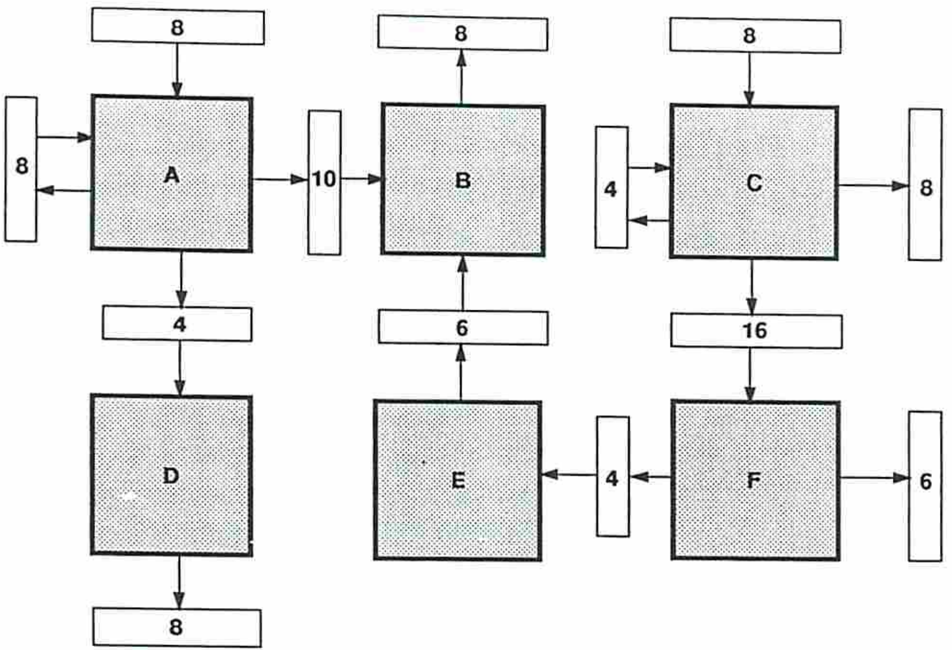


Figure 5.5: Circuit 2.

register is shared by two kernels, all flip-flops in the register are associated with the kernel having the greater test length.

In Table 5.3, while varying the number of scan chains from 1 to 10, the test times for both equal length and optimal chain configurations are shown for each of the three cases. The test times are in terms of the number of clock cycles. The percentage reduction in all three cases represents the savings in the test time of an optimal configuration as compared to equal length chains. For each case, if the kernels are listed in decreasing order of test lengths, the corresponding list of the number of flip-flops associated with each kernel is (10,12,18,22,20,16) for Case 1, (24,12,36,16,4,6) for Case 2 and (36,30,14,10,8,0) for Case 3. An interesting observation is that the test time difference between an optimal and equal length chain configuration decreases as the number of flip-flops associated with kernels of large test lengths increases. In Case 1, the total number of flip-flops associated with the kernels of test length 200 and 500 is 20% of the total number of flip-flops and the average test time reduction is 34%. However, for Case 2 and Case 3, when the flip-flop ratio increases to 36% and 67% respectively, the average test time reduction drops to 20% and 5%, respectively.

	CASE 1		CASE 2		CASE 3	
	No. of test patterns	No. of flip-flops	No. of test patterns	No. of flip-flops	No. of patterns	No. of flip-flops
Kernel A	40	16	60	16	200	30
Kernel B	80	18	500	24	80	14
Kernel C	50	20	80	36	500	36
Kernel D	200	12	200	12	50	8
Kernel E	500	10	50	4	40	0
Kernel F	60	22	40	6	60	10

Table 5.2: Three Variations of Circuit 2

No. of chains	Test time (equal length)	Case 1		Case 2		Case 3	
		Test time (optimal)	% red.	Test time (optimal)	% red.	Test time (optimal)	% red.
1	49598	49598	0	49598	0	49598	0
2	25049	15896	36.54	20642	17.59	23762	5.14
3	17033	11096	34.86	13082	23.20	16934	0.58
4	13025	8018	38.44	10562	18.91	12092	7.16
5	10520	6518	38.04	8051	23.47	10052	4.45
6	9017	5506	38.94	6791	24.69	8282	8.15
7	7514	4786	36.31	5981	20.40	7196	4.23
8	7013	4259	39.27	5261	24.98	6296	10.22
9	6011	3839	36.13	4721	21.46	5693	5.29
10	5510	3539	35.77	4316	21.67	5234	5.01

Table 5.3: Comparison of test times with multiple scan chains

To further illustrate the influence of design characteristics on the configuring of optimal chains, let us consider the three circuit topologies in Fig. 4.3 introduced in Chapter 4. Without loss of generality, let us assume the lengths of all registers in Fig. 4.3(a) are equal to $l/2$ flip-flops, and the lengths of all registers in Figs. 4.3(b) and (c) are equal to l flip-flops. In addition assume that the sets $WSEQ$ are identical for the three designs. The *key* point to note is that an optimal configuration of k chains configured for any one of the three designs can be used to configure an optimal k-chain configuration for the other two designs. This follows from the observation that with the flush policy the test time of an optimal configuration depends only on the weights of the flip-flops, and the number of flip-flops for each of the different weights. In all three designs, the number of flip-flops with weight W_i $(1 \leq i \leq n)$ is equal to l.

Recall that in configuring a single chain for the design in Fig. 4.3(c), the values of the test lengths of the kernels influenced the optimal ordering of registers. Similarly the structure of an optimal k-chain configuration for any of the designs depends on the relative number of patterns to be applied to each kernel. To illustrate this let us consider the problem of configuring three optimal chains for the case of $n = 2$ kernels. Since the number of flip-flops associated with both K_1 and K_2 is equal to l, an optimal 2-chain configuration will consist of two equal length chains. In configuring 3 chains, the optimal configuration will depend on the ratio $\beta = \frac{W_2}{W_1}$. It can be shown that for $\beta \leq 3$, an optimal configuration will consist of three equal length chains with the longest chain being of length $\lceil \frac{2l}{3} \rceil$. For values of $\beta > 3$, an optimal configuration consists of three chains $\{C_1, C_2, C_3\}$. Chain C_1 is of length l and only contains flip-flops of weight W_1. Chains C_2 and C_3 are each of length $\frac{l}{2}$, and only contain flip-flops of weight W_2. Similarly for designs with n greater than 2, the values of the kernel test lengths play an important role in determining an optimal chain configuration.

5.6.2 Case Study

As with a single scan chain, the advantages in optimally configuring multiple chains can be fully appreciated when the design consists of multiple kernels with varying

test lengths. To illustrate this we make use of three scan designs of the *USC Data Path-I*. Design 1 represents the basic design shown in Fig. 4.6 in which each combinational block is treated as a kernel. In Design 2 the ALU is replaced by a larger combinational block that requires 500 test patterns, and in Design 3 we modify Design 1 by introducing registers of appropriate width at the primary input/outputs and control inputs. Table 5.4 compares the results of configuring optimal chains with equal length chains for each of the three designs. The number of chains vary from 2 to 6 and the test times are in terms of the number of clock cycles. Note the savings in test time for all three designs.

Index	No. of chains	Test time (equal length)	Test time (optimal)	Percentage reduction
Design 1	2	9539	9539	0
	3	6479	6091	5.99
	4	4859	4437	8.68
	5	3959	3515	11.21
	6	3419	3013	11.87
Design 2	2	26552	23804	10.35
	3	18035	16504	8.49
	4	13526	11368	15.95
	5	11021	8944	18.85
	6	9518	7507	21.13
Design 3	2	15119	14267	5.64
	3	10259	8130	20.75
	4	7739	6037	21.99
	5	6299	5033	20.10
	6	5219	4209	19.35

Table 5.4: Test times for three scan designs of the *USC Data Path-I*

5.7 Extensions to the Algorithm

In this section we indicate ways to extend the algorithm to deal with (a) routing area constraints, (b) unequal length registers as the basic storage elements, and (c) the use of the minimum shift policy in applying tests.

5.7.1 Routing Area Constraints

In circuit design environments, the standard practice is to configure equal length chains after the physical placement and routing of circuit modules. The dynamic programming algorithm can be used to logically configure the scan chains prior to the physical layout process, and permit the placement and routing programs to optimize the layout. Since the optimal solution only constrains the assignment of flip-flops to scan chains, the layout system is provided with the flexibility to order the flip-flops within a chain such that the routing area is minimized. In an optimal configuration, the positions of any two flip-flops of equal weight can be interchanged without increasing the test time. The optimality is also not affected when flip-flops are interchanged between any two chains provided the weight of either chain does not increase.

The algorithm can also be modified to incorporate routing constraints in configuring the chains. For example, if certain flip-flops need to be placed adjacent to each other in a chain, they can be treated as a single element in the algorithm. The evaluation of test times for partial configurations containing these elements will need to be appropriately modified. In the algorithm the flip-flops are initially ordered in terms of non-decreasing weight parameters. This ordering can be modified to reflect constraints dealing with routing area. For example if two or more flip-flops are physically close to each other in the layout, they can also be made to lie close to each other in the initial ordering. This will increase the probability of them being assigned to the same chain. Modifying the order of flip-flops might however compromise on the optimality of the final solution.

5.7.2 Unequal Length Registers

Although we have assumed single-bit flip-flops in our discussion, the dynamic programming algorithm can also be used in optimally configuring registers provided they are all of equal length. In evaluating the test time of a partial chain configuration, we use a scaling factor of l where l represents the length of each register. The difficulty arises when dealing with registers that are not all of the same length. If the set $\mathcal{F}$ of N scan flip-flops is replaced with the set S of M scan registers,

the MSC optimization problem is as hard as the problem of configuring two equal length chains. This corresponds to dividing S into two disjoint subsets such that the lengths of registers in both subsets are equal, i.e., equivalent to the *Partition* problem used in the previous chapter. A formal proof of the NP-hardness of the MSC problem can be derived based on this analogy.

To more clearly illustrate the complexity of the problem, consider a scan design containing three registers R_1, R_2 and R_3 with weight parameters equal to 300, 300, and 200, and lengths equal to 5, 7, and 8, respectively. By breaking down the registers into flip-flops, an optimal 2-chain configuration will consist of two equal length chains, each of length 10. The associated test time is equal to 300(10 + 1) + 10 = 3310 clock cycles. Note that in configuring the two chains we only need to break up a single register. On the other hand, retaining the registers an indivisible elements, an optimal 2-chain configuration will correspond to assigning R_3 to the first chain, and R_1 and R_2 to the second. The corresponding test time is equal to 3913 clock cycles. Note that both the weight and length of the first chain are less than the corresponding values for the second chain. Hence the search space for an optimal solution cannot be restricted to configurations that satisfy the conditions of Lemma 3.

To generate a k-chain configuration with registers, let us start with an optimal configuration assuming the registers are broken down into flip-flops. Let B denote the number of registers whose flip-flops are assigned to different chains. From the structure of the dynamic programming algorithm, a register will only be divided if its flip-flops are assigned to two or more consecutive chains. Note that the flip-flops of at most one register can lie in any two consecutive chains. Hence in a k-chain configuration, flip-flops from at most $k - 1$ registers will be assigned to different chains, i.e., B is upper bounded by $k - 1$. If B is equal to 0, the configuration will correspond to an optimal solution. Else the flip-flops belonging to the "divided" registers can be appropriately reassigned among the chains. For example, if we assume that flip-flops from a register lie in at most two chains, the flip-flops can be reassigned so that the register wholly lies in one of two chains. Hence in the worst case, 2^B different configurations will be considered to determine a solution that minimizes the test time.

Given an optimal solution based on flip-flops, a designer can specify the amount of reassignment to tradeoff between the test time and the routing overhead in breaking up a register across two chains. The above heuristic approach to automatically generate a configuration based on registers might lead to a suboptimal solution. A more detailed analysis needs to be done to estimate the amount of deviation from an optimal solution.

5.7.3 Minimum Shift Policy

The use of the flush policy leads to increased flexibility in configuring the chains since the test time is independent of the order of flip-flops within each chain. However greater savings in test time can be obtained if the overlapped scheme is used with a minimum shift policy. The drive cycle and receive cycle can be evaluated for each active chain in a session. The maximum of the drive and receive cycles of all active chains will determine the chain cycle of the session. To minimize the chain cycles and hence the overall test time, the goal in configuring k chains is to determine (1) an assignment of the flip-flops to the chains, and (2) an ordering of the flip-flops within each chain. By imposing an ordering of flip-flops within a chain, less flexibility is afforded in minimizing the routing overheads in configuring the chains. A designer must therefore weigh the savings in test time against the possible effects on the routing area overhead.

Minimizing the test time with the minimum shift policy is a more complex optimization problem than with the flush policy. One possible approach to configuring the chains is to divide the problem into two parts; the flip-flops are first assigned to the chains prior to ordering them within each chain. Consider the following heuristic approach to generate a multiple chain configuration for the minimum shift policy.

Multiple-Chain-Heuristic (Inputs: $\mathcal{F}, k, WSEQ$)

1. Invoke *Generate-Optimal-Configuration*($\mathcal{F}, k$) to optimally assign the N flip-flops to k chains.
2. For each chain C_j ($1 \leq j \leq k$)
   - Determine driver and receiver weights for each flip-flop.

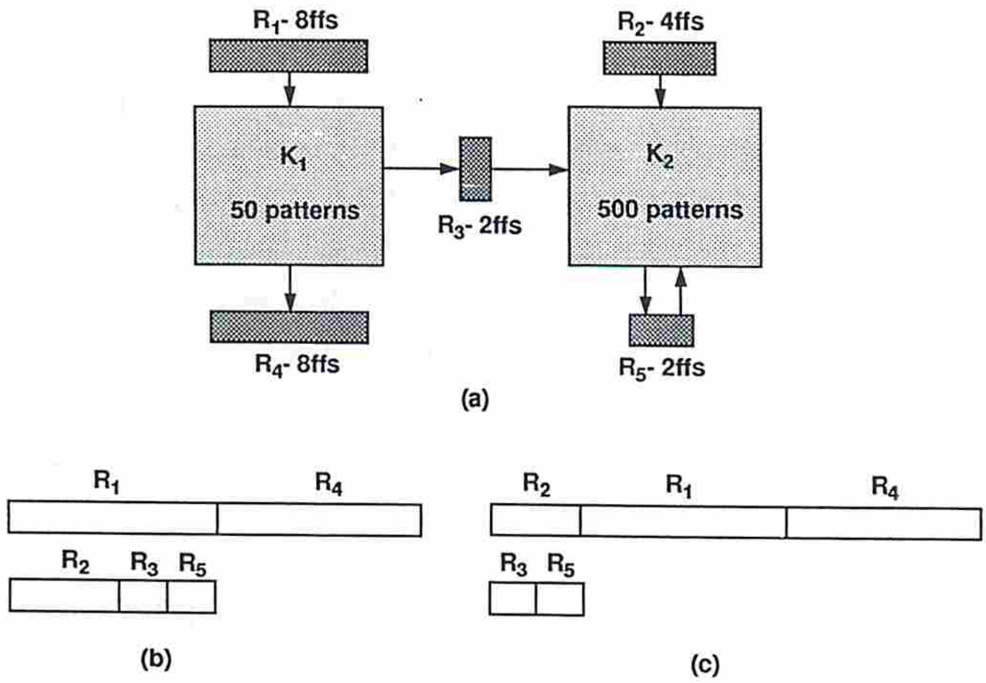


Figure 5.6: (a) Example design. (b) Optimal 2-chain configuration with flush policy. (c) Optimal 2-chain configuration for the minimum shift policy.

- Optimally order the flip-flops in the chain using the procedure *Generate-Optimal-Chain* (flip-flops in C_j, $WSEQ$).
- Determine drive and receive cycle for each session of overlapped scheme using chain C_j.

3. The chain cycle CC_i of session TS_i ($1 \leq i \leq n$) is given by the maximum of the drive and receive cycles among all chains.

4. The "optimal" test time is evaluated using equation 3.1.

In the above procedure the flip-flops in a given chain are treated independent of the flip-flops in the other chains. In other words the flip-flops in the chain are assumed to be the only scan elements in the design. Using their driver and receiver weights, the flip-flops are optimally ordered within the chain based on the procedure determined for a single chain in Section 4.5. However the two-step approach used in the procedure might not lead to an optimal solution. To illustrate this, consider the scan design shown in Fig. 5.6(a) consisting of two combinational kernels and five scan registers. In Fig. 5.6(b), a two-chain configuration as obtained from the dynamic programming algorithm is shown. This represents an optimal configuration with the flush policy. Note that no register is broken up in configuring the chains. The registers can now be optimally ordered within each chain as shown in the figure. Using the minimum shift policy, the chain cycles of sessions TS_1 and TS_2 are equal to 8 and 8, respectively. The total test time is thus equal to 50(8 + 1) + 450(8 + 1) + 8 = 4508 clock cycles. However an optimal two-chain configuration for the minimum shift policy is shown in Fig. 5.6(c). The chain cycles of the two sessions are equal to 12 and 4, respectively, leading to a total test time of 2912 clock cycles.

A more detailed study of the problem needs to be carried out. In particular certain properties of optimal configurations should be identified to constrain the total search space. The two-step approach can be used as a viable heuristic although more analysis needs to be done to gauge its effectiveness. The design of an efficient algorithm to configure multiple chains for the minimum shift policy is a subject for future study.

5.8 Summary

We have presented a comprehensive and structured methodology to efficiently configure multiple scan chains. The main idea is to assign flip-flops involved in testing kernels with larger test lengths to shorter chains. A polynomial-time algorithm has been derived to obtain a configuration that minimizes the test application time. Results clearly show the advantages of the design strategy over configuring equal length scan chains. The efficiency of the algorithm permits a designer to easily evaluate optimal test times for any number of chains, and investigate the tradeoffs between pin count and test time.

generate optimal solutions. Implementation results will be used to compare the test times of equal length and optimal chains. Extensions to the minimum shift policy and to the use of registers as indivisible entities will be discussed at the end of the chapter.

5.2.1 Test Application with Multiple Chains

Consider the circuit shown in Fig. 5.1(a) consisting of three combinational kernels K_1, K_2 and K_3 and six scan registers. The test lengths of the kernels are shown in the figure. The scan flip-flops in each of the registers are assigned to the four scan chains C_1, C_2, C_3 and C_4 as shown in Fig. 5.1(b). Note that two of the flip-flops in register R_4 are assigned to chain C_1 and the other two to chain C_4. Given a chain C and a kernel K, if any flip-flop in C is used to either apply test patterns or receive test results from K, we say that C is *involved* in testing K. Hence chains C_1, C_2 and C_4 are involved in testing K_1; C_1, C_3 and C_4 are involved in testing K_2; and C_3 is involved in testing K_3. Using the four chains we employ the overlapped scheme with a flush policy to apply tests to the respective kernels.

For example, in the first test session TS_1, 30 patterns are applied to all three kernels. Since each of the four chains is involved in testing at least one kernel, all four chains are used and are said to be "active" in the session. To apply each test pattern in the session, the corresponding input patterns for the kernels are merged and shifted simultaneously into the four chains, while at the same time the previous test results are shifted out. Since the longest active chain C_1 has 10 flip-flops, a minimum of 10 clock cycles is required to entirely flush all four chains. The chain cycle of the session is thus equal to 10. Note that all the chains operate in a synchronized manner and hence must all follow the chain cycle of the current session.

The details of the overlapped scheme for the example design are summarized in Table 5.1. The total test time is equal to $330 + 770 + 3600 + 10 = 4710$ clock cycles where the last 10 clock cycles are required to synchronize between sessions and flush out the final results. Since all chains in a session operate synchronously, a single control pin on the chip is sufficient for shift enable purposes. In Chapter 7 we use an additional control pin to permit certain scan chains to be shifting data

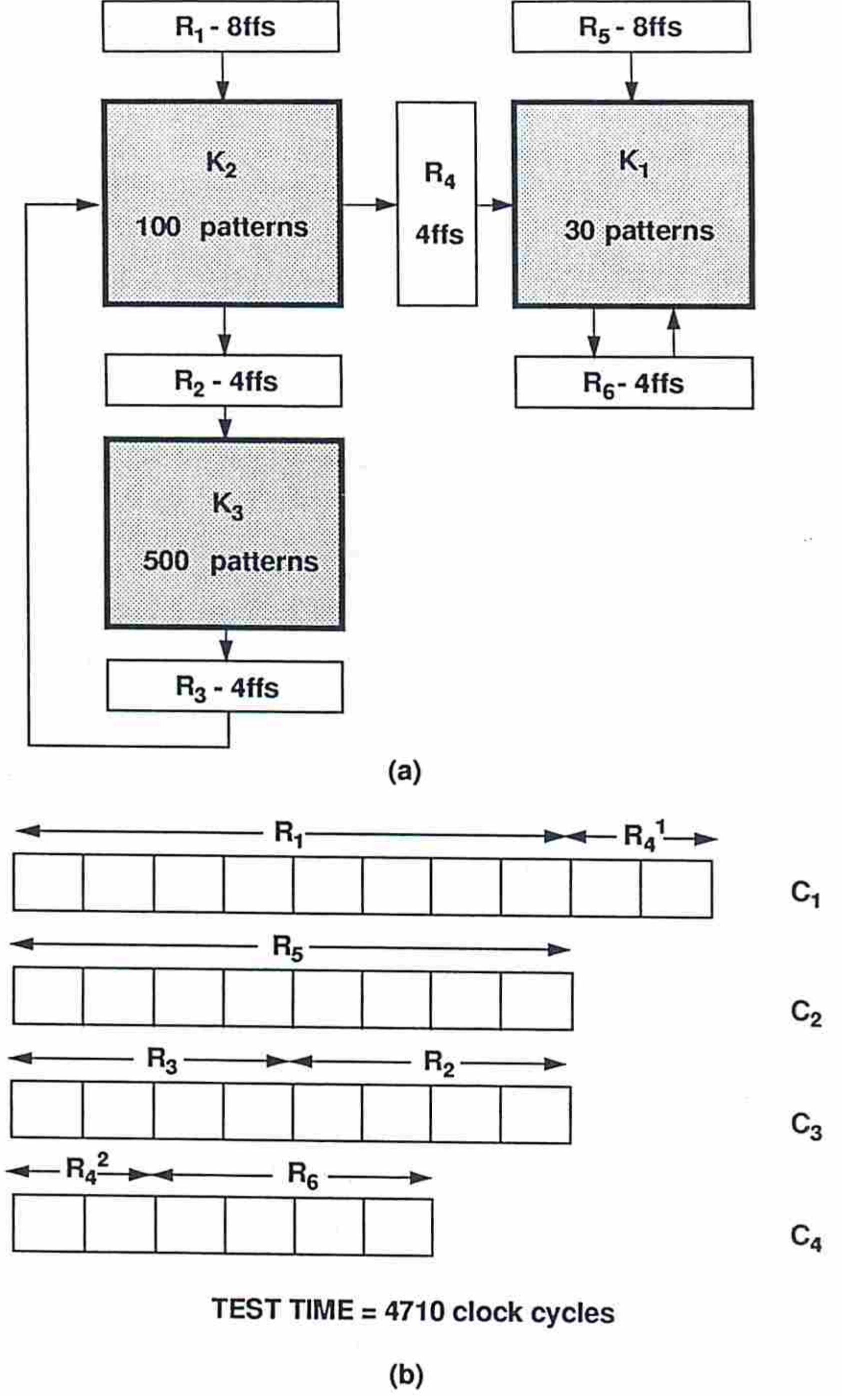


Figure 5.1: (a) Example design. (b) A four-chain configuration for the design.

Chapter 6

A Reconfigurable Scan Chain

6.1 Introduction

In this chapter we study our third chain configuration which deals with the idea of reconfiguring a single scan chain. The IEEE 1149.1 boundary scan standard [30] provides a uniform framework for testing boards. It mandates that all chips on a board be accessed through a test bus that only provides a single test data input and output port. This severely constrains the bandwidth of test data to a device and precludes the use of multiple scan chains operating in parallel. The main idea in our approach is to employ multiplexers to permit the realization of multiple scan chains that can be used in a serial fashion. The multiplexers are used to bypass registers that are not frequently accessed in the test application process. As we will show, this can significantly reduce the overall test application time. An analogy can be drawn between our approach, and the *bypass* mode associated with the 1149.1 boundary scan standard [30]. The bypass mode operating at the board level, permits any chip to be bypassed to reduce shifting time in testing other chips, glue logic or interconnect on a board.

A number of important issues arise in using multiplexers to reconfigure the scan chain. At the device level, the introduction of multiplexers will impact the area overhead in terms of both the logic for the multiplexers and additional routing overheads. Hence it is important to efficiently introduce a minimal number of multiplexers to reduce the overall test application time. Appropriate tradeoffs must be provided to

a designer depending on the relative importance of the design costs of area overhead and test time. In this regard use of the overlapped scheme with the two shift policies can provide different degrees of tradeoffs in the design costs. For each shift policy, we will discuss in detail optimization techniques to minimize both the number of added multiplexers and the associated test time. The advantages of reconfiguration will be illustrated for a wide range of partitioned circuits.

6.2 Main Concepts

We briefly highlight the salient features in reconfiguring a scan chain and compare the design costs in using the two shift policies. Based on the models introduced in the previous chapters and some additional notation to model the multiplexers, we formally define the optimization problems that will be analyzed for each shift policy.

6.2.1 Reconfiguring a Single Scan Chain

Consider the circuit shown in Fig. 6.1(a) consisting of six scan registers and three combinational kernels K_1, K_2 and K_3 with test lengths of 20, 100 and 500, respectively. The number of flip-flops within each scan register is shown and all six registers are connected into a single scan chain consisting of 32 flip-flops. Under the overlapped scheme with a flush policy, the test time is equal to $500(32 + 1) + 32 = 16532$ clock cycles. Similarly with the minimum shift policy, the chain cycle in all three sessions is equal to 28, leading to a total test time of 14528 clock cycles.

With both shift policies, the key point to note is that test time is wasted in shifting data through registers that are not accessed in a session. For example in TS_3, it is sufficient to shift patterns only into registers R_1, R_2 and R_3 involved in testing K_3. As described in Chapter 4, ordering the registers in the chain provides one way to exploit this information to reduce the shifting time. Reconfiguration provides an additional degree of freedom to further reduce the shifting time.

To illustrate the idea, consider the scan configuration shown in Fig. 6.1(b) which is identical to the one in Fig. 6.1(a) except for the introduction of the *2-to-1* multiplexer at the scan-out pin. By switching the control signal on the multiplexer two

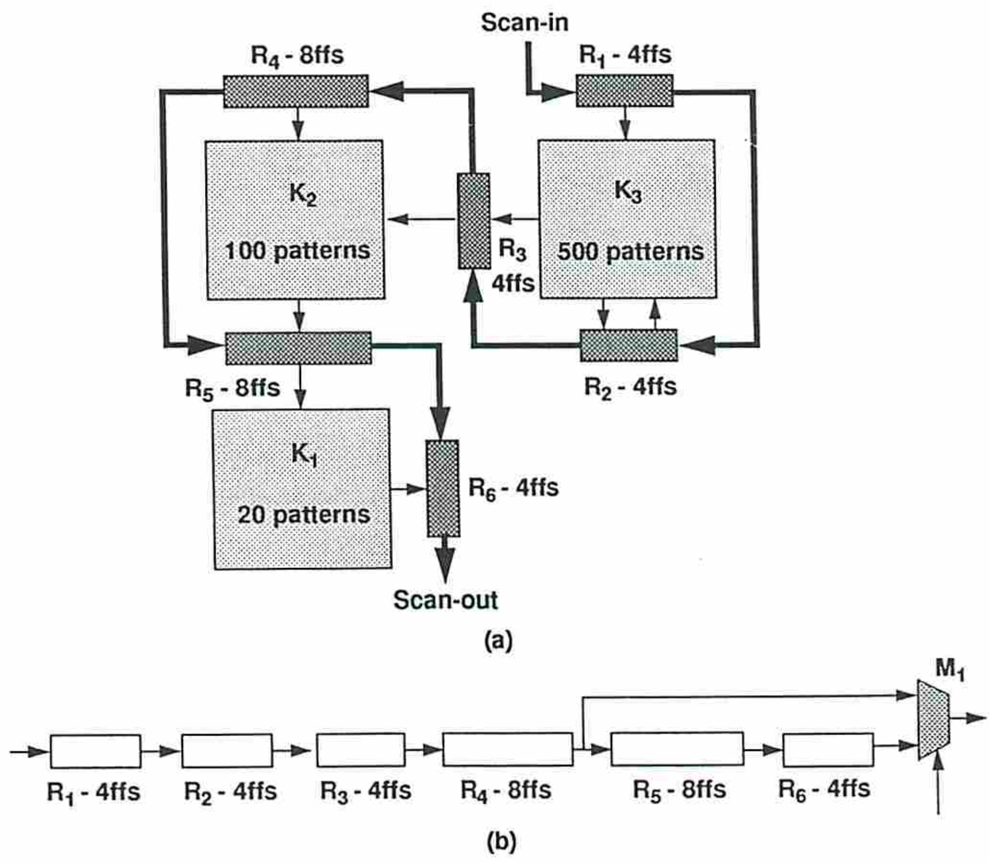


Figure 6.1: (a) Example design.(b) Reconfigured scan chain.

Test session	Kernels tested	Number of patterns	Flush policy		Minimum shift policy	
			Chain cycle	Test time	Chain cycle	Test time
TS_1	K_1, K_2, K_3	20	32	660	28	580
TS_2	K_2, K_3	80	32	2640	28	2320
TS_3	K_3	400	20	8400	16	6800

Table 6.1: Summary of test sessions for both shift policies

different serial scan chains can be realized. The first will consist of all six registers as in Fig. 6.1(a) and the second made up of registers R_1, R_2, R_3, R_4 from scan-in to scan-out. For both shift policies, the chain consisting of all six registers will be used in sessions TS_1 and TS_2. Hence the chain cycles in these two sessions remain the same as with the scan chain in Fig. 6.1(a).

However in the third session, since only registers R_1, R_2 and R_3 need to be accessed, it is sufficient to employ the shorter chain for both policies. With the flush policy the chain cycle of TS_3 reduces from 32 to 20. Hence the total test time decreases to 11732 clock cycles, a savings of 29% over the combined scheme. With the minimum shift policy, a chain cycle of 16 is sufficient to shift a pattern into R_1 and R_2 while simultaneously shifting out results from R_2 and R_3 to the scan-out pin. The test time thus reduces to 9728 clock cycles, a reduction of 41% over the combined scheme, and a savings of 33% over the non-reconfigured scan chain. For each shift policy, a summary of the test sessions using the reconfigurable scan chain of Fig. 6.1(b) is shown in Table 6.1.

Note that for both shift policies, we have neglected the time required to reconfigure the scan chain. In a boundary scan environment, this will entail a fixed number of clock cycles to shift an instruction into the *instruction register* [30] for setting up the multiplexer control signals. We will neglect this constant control overhead for reconfiguring the scan chain in all subsequent discussion.

6.2.2 Tradeoffs in Test time and Area Overhead

To fully illustrate the advantages of reconfiguring the scan chain, let us evaluate the maximum possible reductions in test time that can be achieved with each of

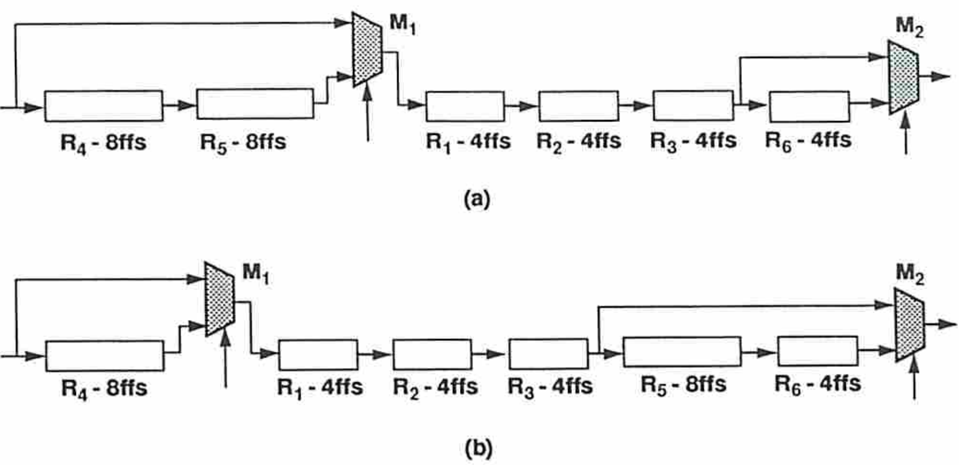


Figure 6.2: Optimally reconfigured chains for (a) flush policy, (b) minimum shift policy.

the shift policies. For our example circuit, since the first session will require the use of all six registers, the minimum chain cycle that can be used with the flush policy is equal to 32. In the second session, since only registers R_1, R_2, R_3, R_4 and R_5 involved in testing K_2 and K_3 need to be flushed, a minimum chain cycle of 28 is sufficient. Similarly in the third session, a minimum chain cycle of 12 is sufficient to flush R_1, R_2 and R_3. The scan configuration shown in Fig. 6.2(a) reconfigured with two *2-to-1* multiplexers achieves these lower bounds on the chain cycles. Note that the control signal for M_2 is toggled after TS_1 and similarly the control signal for M_1 is toggled after TS_2. The corresponding test time is equal to $20(32 + 1) + 80(28 + 1) + 400(12 + 1) + 32 = 7812$ clock cycles, a reduction of 52.7% over the combined scheme.

Consider the scan configuration shown in Fig. 6.2(b) which achieves the lower bounds on the chain cycles with the minimum shift policy. In the first session, the chain consisting of all six registers is used. A chain cycle of 28 is needed to shift patterns into R_4, R_1, R_2, R_3 and R_5 while shifting out results from R_1, R_2, R_3, R_5 and R_6. Employing the same chain in the second session, 20 clock cycles are needed to shift patterns into R_4, R_1, R_2 and R_3 while shifting out results from R_2, R_3 and R_5. In the third session, by toggling the control signals on both multiplexers, a chain cycle of 8 is sufficient to shift patterns into R_1 and R_2 while shifting out results from

R_2 and R_3. The test time is given by 20(28 + 1) + 80(20 + 1) + 400(8 + 1) + 28 = 5888 clock cycles, a reduction of 64.3% over the combined scheme. Intuitively the large reductions in test time arise from the capability to bypass the less frequently accessed registers in the testing process, i.e., those registers solely involved in testing K_1 and K_2.

The goal of our approach is to generate optimally reconfigured scan chains as in Figs. 6.2(a) and 6.2(b) which minimize the test time given a *fixed* number of multiplexers. Note that the savings in test time depend on the number of kernels, their relative test lengths and the distribution of registers involved in testing them. In reconfiguring a scan chain for both shift policies, the following issues must be considered: (1) the savings in test application time; (2) the logic overheads of the added multiplexers; and (3) the routing overheads in constraining the order of registers in the chain. To efficiently tradeoff the logic overheads with the test time, it is important to obtain the maximum possible savings with the addition of every multiplexer. This will form the main focus of our attention in the subsequent three sections.

With regard to constraining the order of registers in the chain, the area overhead is minimal since it only involves the routing of a single wire between registers. In addition the flush policy, while providing substantial reductions in the test time, retains the flexibility to minimize routing overheads in configuring the chain. For example in Fig. 6.2(a), the test time is independent of the order of registers in certain sections of the scan chain, e.g., the registers R_1, R_2 and R_3 can be placed in any order among themselves. Use of the minimum shift policy with an optimally reconfigured chain provides the greatest reductions in the overall test time. However this policy imposes more constraints on the order of registers in the chain. In developing the optimization techniques we will indicate ways to retain some of the flexibility in ordering registers, possibly by trading off reductions in test time.

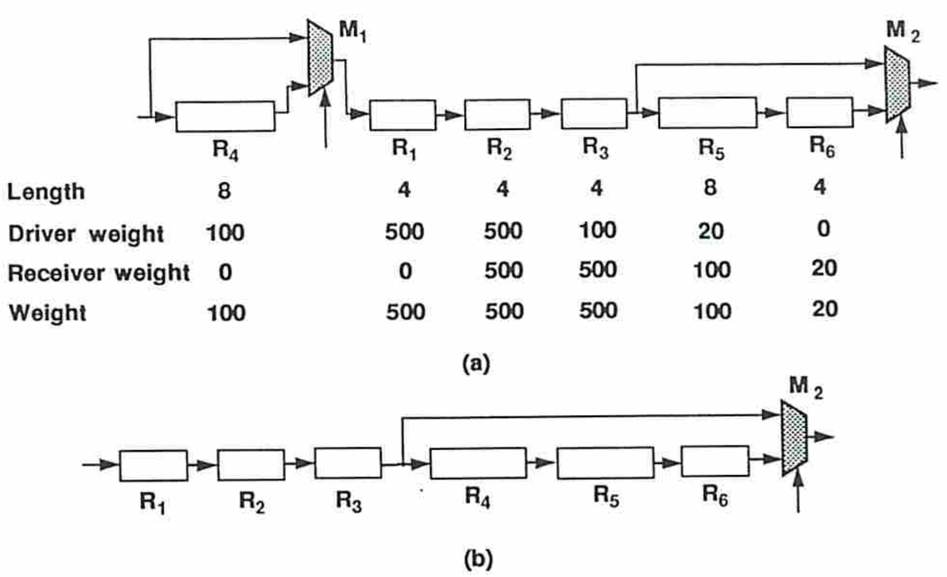


Figure 6.3: (a) Parameters of registers. (b) Transformed configuration.

6.3 Optimal Scan Chain Configurations

6.3.1 Model of Optimization Problem

In this section we lay the groundwork to deal with the problem of generating optimally reconfigured chains for both shift policies. We adopt the notation developed in Chapter 4 to model a scan chain as a sequence of slots numbered $1, 2, \ldots, M$. Any permutation of registers in the slots represents a feasible scan chain ordering.

The configuration of Fig. 6.2(b) is shown again in Fig. 6.3(a) with the parameters of the registers. Recall that the weight of a register is given by the maximum of its driver and receiver weights. As in Chapter 5, we will use the notation w_p to refer to the weight parameter of register R_p ($1 \leq p \leq M$). We reconfigure a scan chain by inserting a set of *2-to-1* multiplexers that permit certain segments of the chain to be bypassed. Consider the bypass segment shown in Fig. 6.3(a) consisting of the registers R_5 and R_6. Since these registers occupy slots 5 and 6 in the chain, we model the segment by the ordered pair (5,6). Any multiplexer introduced to reconfigure the chain is associated with a bypass segment (f, l) where f and l represent, respectively, the *first* and *last* slots of the segment bypassed by

the multiplexer. If j ($j > 1$) multiplexers bypass different segments having the same final slot, we replace the j *2-to-1* multiplexers by a single *j+1-to-1* multiplexer. Each input to this *j+1-to-1* multiplexer will be referred to as a *bypass point*. The insertion of multiplexers generates a number of serial scan chains where each chain represents a distinct sequence of scan registers from scan-in to scan-out. Without loss of generality, we will assume there are k serial scan chains denoted by the set $C = \{C_1, C_2, \ldots, C_k\}$ where the value of k depends on both the number of multiplexers and their respective bypass segments. These chains will be used to determine the chain cycle of every test session for both shift policies. Let the chains in C be ordered in terms of non-increasing lengths. For the scan configuration in Fig. 6.3(a), $C = \{C_1, C_2, C_3, C_4\}$ where $C_1 = \{R_4, R_1, R_2, R_3, R_5, R_6\}$, $C_2 = \{R_1, R_2, R_3, R_5, R_6\}$, $C_3 = \{R_4, R_1, R_2, R_3\}$ and $C_4 = \{R_1, R_2, R_3\}$.

For both shift policies, the chain cycle of session TS_i ($1 \leq i \leq n$) is determined by the *shortest chain in C containing all registers of weight greater than or equal to W_i*. With the flush policy, the chain cycle is equal to the length of this chain in terms of the number of flip-flops. For the minimum shift policy, based on the positions of the drivers and receivers in the chain, the drive cycle and receive cycle can be evaluated. The chain cycle of the session is given by the maximum of these two values. Hence given the set C of serial scan chains, the chain cycles for each test session can be evaluated and used in equation 3.1 to determine the total test time. The final term CC_1 in equation 3.1 will be neglected in all subsequent analysis.

6.3.2 Statement of Optimization Problems

For each shift policy let us initially evaluate lower bounds on the test times. As in Chapter 4, we evaluate the least possible chain cycle that can be employed in each test session. The lower bounds on the chain cycles with a reconfigurable scan chain are different from the corresponding bounds with a single scan chain. With a single chain the registers that are not accessed in a session will influence the chain cycle of the session. Test data might still have to be shifted through these registers, and this will increase the overall shifting time. On the other hand, with a reconfigurable scan

chain, these registers can be bypassed to restrict the shifting of data only through registers accessed in the session.

Flush policy - In session TS_i, every register with a weight parameter greater than or equal to W_i will be used. Hence the *minimum chain cycle* MCC_i that can be employed in the session is equal to the sum of the lengths of all registers R_p such that $w_p \geq W_i$. In the circuit of Fig. 6.1(a), the minimum chain cycles of the three sessions are equal to 32, 28 and 12, respectively.

Minimum shift policy - Every register R_p with either $d_p \geq W_i$ and/or $r_p \geq W_i$ will be used in session TS_i. Consider a serial scan chain consisting only of these registers in which all drivers are placed before all receivers in sequence from the scan-in to scan-out pin. For example, in the circuit of Fig. 6.1(a), the registers R_1, R_2 and R_3 will be used in session TS_3. We configure a hypothetical chain consisting of these three registers with the drivers, i.e., R_1 and R_2, placed before the receivers, i.e., R_2 and R_3. The minimum chain cycle that can be employed in the session is thus equal to 8. In general the minimum chain cycle of session TS_i is the maximum of two quantities, (1) the sum of lengths of all registers R_p such that $d_p \geq W_i$, and (2) the sum of lengths of all registers R_p such that $r_p \geq W_i$.

The test time with a reconfigurable scan chain depends on (a) the order of registers in the chain, (b) the number of multiplexers used to reconfigure the chain, and (c) the positions at which the multiplexers are placed. Given the kernels in $KSEQ$ and the set S of scan registers involved in testing them, a statement of the optimization problem can be done in two ways for each of the shift policies.

(1) *Determine an ordering of scan registers for inserting a minimum number of multiplexers to realize the lower bounds on the chain cycles for each test session of the overlapped scheme.*

(2) *Given that j multiplexers are to be used for reconfiguration, determine an ordering of scan registers and the positions of the multiplexers to minimize the test application time.*

Note that the first problem is a special case of the second problem. By treating it as a separate problem, we can gain insight into the complexity of deriving optimal solutions as well as highlight the maximum savings in test application time. A scan

chain reconfigured with j multiplexers is said to be optimal if the associated test time is less than or equal to the test time of every other scan configuration with j multiplexers. The size of the search space is given by the the number of different configurations that can be generated by M scan registers and j bypass multiplexers. The number of different scan chain orderings is equal to $M!$. For a given ordering of registers, the number of ways to insert a single multiplexer is equal to the number of different bypass segments of the scan chain. This can be shown to be equal to $\mathcal{P} = M(M+1)/2 - 1$. The number of different ways in placing j multiplexers is thus equal to ${}^{\mathcal{P}}C_j$. Hence the total number of scan configurations is equal to ${}^{\mathcal{P}}C_j * M!$. Each of these configurations can generate a maximum of 2^j different serial chains by appropriately controlling the multiplexers. Since the total number of possible solutions is exponential in both the number of scan registers and bypass multiplexers, we need to restrict the search space in developing efficient solutions.

Under the overlapped scheme, the registers used in session TS_{i+1} are a subset of the registers used in session TS_i $(1 \leq i < n)$. This observation can be used to restrict the number of ways in placing multiplexers to reconfigure the scan chain.

Lemma 5 Consider an optimal configuration containing two multiplexers M_1 and M_2 that bypass segments (f_1, l_1) and (f_2, l_2), respectively, such that $f_1 < f_2$, $l_1 < l_2$ and $f_2 \leq l_1$. We can modify the positions of one of the multiplexers to bypass the segment (f_1, l_2) without increasing the total test time.

Proof Assume M_1 is used to bypass the segment (f_1, l_1) in session TS_α and M_2 used to bypass the segment (f_2, l_2) in session TS_β $(\alpha < \beta)$. Hence every register in the segment (f_1, l_1) will not be required in any session after TS_α. The segment of the chain bypassed by M_2 can be modified to include the registers in the segment $(f_1, f_2 - 1)$ without increasing the test time. Similarly if M_2 is used in a session prior to the use of M_1, we can modify the position of M_1 to include the segment $(l_1 + 1, l_2)$. □

The above lemma provides a sufficient condition on any two serial chains used in applying tests with the overlapped scheme. For an optimally reconfigured scan chain, consider two serial chains C_α and C_β used in two consecutive sessions TS_i and TS_{i+1}, respectively. Since the registers bypassed in a session will not be required

in a subsequent session, the chain C_β must be a subset of the chain C_α. The next lemma is a direct consequence of this observation and will prove useful when dealing individually with each shift policy.

Lemma 6 Given an optimally reconfigured scan chain, let k different serial chains be employed in applying tests using the overlapped scheme. The minimum number of bypass multiplexers to generate these k different chains is equal to $k-1$. □

The above two lemmas are valid for both shift policies. In the next two sections we will individually analyze reconfiguration techniques for both the flush and minimum shift policies.

6.4 Flush Shift Policy

With the flush policy the order of registers in a serial scan chain does not affect the chain cycle of a session. Consider the scan configuration shown in Fig. 6.3(a) consisting of four serial chains $\{C_1, C_2, C_3, C_4\}$. In applying tests with the flush policy, chain C_1 will be used in sessions TS_1 and TS_2 and chain C_4 in session TS_3. We can transform this configuration into a simpler one by placing the registers in C_4 in any order starting from the scan-in pin, and then placing the registers in $(C_1 - C_4)$ (where $-$ denotes the set difference operator) in any order up to the scan-out pin. A single multiplexer is then introduced to bypass the segment (4,6). The resultant configuration is shown in Fig. 6.3(b). The chain cycles of all three sessions are unchanged and the number of multiplexers is reduced by one.

The following lemma generalizes this idea by restricting both the order of registers and the style of reconfiguration in generating an optimal solution for the flush policy. To simplify its explanation we assume the M scan registers are broken down into their constituent flip-flops. Similar to our multiple chain methodology, these flip-flops will be used as the basic elements to be ordered in the scan chain. The assumption is made only for ease of explanation and we will show how the final results are valid for both equal and unequal length registers.

Lemma 7 To generate an optimally reconfigured chain with N scan flip-flops and j multiplexers, it is sufficient to consider a configuration with (a) the scan flip-flops

placed in order of non-increasing weight parameters from the scan-in to scan-out pin, and (b) a single *j+1-to-1* multiplexer introduced at the scan-out pin.

Proof Consider any optimal scan configuration satisfying Lemma 5 with $(j+1)$ different serial scan chains used in applying tests. Let us denote these chains as $\{C_1, C_2, \ldots, C_j, C_{j+1}\}$. From Lemma 5 it follows that $C_{j+1} \subset C_j \subset \ldots \subset C_2 \subset C_1$. Recall that if a serial chain is used in session TS_i, it must contain every flip-flop of weight greater than or equal to W_i. We transform this configuration to one that satisfies the statement of the lemma without increasing the number of bypass multiplexers. Place the flip-flops in the sets $C_{j+1}, (C_j - C_{j+1}), \ldots, (C_2 - C_3), (C_1 - C_2)$ in order from scan-in to scan-out. Introduce a *j+1-to-1* multiplexer at the scan-out pin with its j bypass points inserted between each of these sets. Rearrange the flip-flops in the chain in order of non-increasing weight from scan-in to scan-out without disturbing the positions of the bypass points. By this transformation, the length of every serial chain is preserved and the maximum weight of the flip-flops in each of the $(j+1)$ chains does not decrease. Hence the chain cycle of each test session cannot increase. From Lemma 6, we note that a minimum of j multiplexers is required to obtain $(j+1)$ different serial scan chains. □

Given N scan flip-flops and j multiplexers, we only need to consider a single ordering of the scan flip-flops and $(N-1)$ possible positions for the bypass points of the *j+1-to-1* multiplexer. The size of the search space has thus been reduced to ${}^{N-1}C_j$ different configurations. The next lemma restricts the positions at which the bypass points of the *j+1-to-1* multiplexer can be placed.

Lemma 8 In an optimally reconfigured scan chain, the possible candidates for the bypass points of the *j+1-to-1* multiplexer can be restricted to positions between two adjacent flip-flops in the chain having *different* weight parameters.

Proof Assume that in an optimal configuration, there exists a bypass point between two adjacent flip-flops r_α and r_β, each of equal weight W_i. Move the bypass point one flip-flop to the left closer to the scan-in pin so that it lies at the input to r_α. As a result the chain cycle of every session from TS_1 to TS_i is unchanged, and the chain cycle of every session from TS_{i+1} to TS_n will either decrease or remain the

same. Note that if a bypass point already exists at the new position, we can remove the original bypass point without affecting the total test time. □

Corollary To achieve the lower bounds on the chain cycles with the flush policy, a minimum of λ multiplexers is required, where λ =(number of different weights of the scan flip-flops) -1. □

The above corollary provides the solution to the first of our problems by introducing a *λ+1-to-1* multiplexer at the scan-out pin with a bypass point placed between every pair of adjacent flip-flops with different weights. In general the number of different weights for the scan flip-flops will be less than or equal to n, the number of kernels in the design. Hence the second problem of generating an optimal configuration given j multiplexers is in the worst case reduced to choosing one of ${}^{n-1}C_j$ configurations. Explicitly enumerating each configuration would lead to an algorithm with worst case time complexity $O(n^j)$. Although exponential in the number of multiplexers, the algorithm will be efficient for most scan designs provided the number of kernels is not excessively large. Note that for the cases of $j = 1, 2, (\lambda - 2)$ and $(\lambda - 1)$, the complexity of the algorithm is either linear or quadratic in the number of kernels. For every other case we have designed a more efficient algorithm based on *dynamic programming* (see Appendix I). Given j multiplexers, the worst case time complexity of the dynamic programming algorithm is equal to $O(jn^3)$.

We need to highlight two points about the optimal solutions generated by our algorithm. Recall that in our analysis we assumed that scan registers are broken down into individual flip-flops. However in an optimal configuration, the bypass points are restricted to lie between flip-flops of different weights. Since all flip-flops in a register have the same weight, the algorithm will retain registers as indivisible entities in generating optimal solutions. With regard to the routing overheads in reconfiguring the scan chain, a great degree of flexibility is afforded both in the ordering of registers and in the placement of bypass multiplexers. A configuration generated by our algorithm can be used to generate alternate optimal solutions that provide for different ways of inserting bypass multiplexers. An optimal configuration with 2 multiplexers for our example design can be obtained from the configuration in Fig. 6.3(b) by replacing the *2-to-1* by a *3-to-1* multiplexer with the second bypass point placed between R_5 and R_6. The resulting configuration is shown in Fig. 6.4(a).

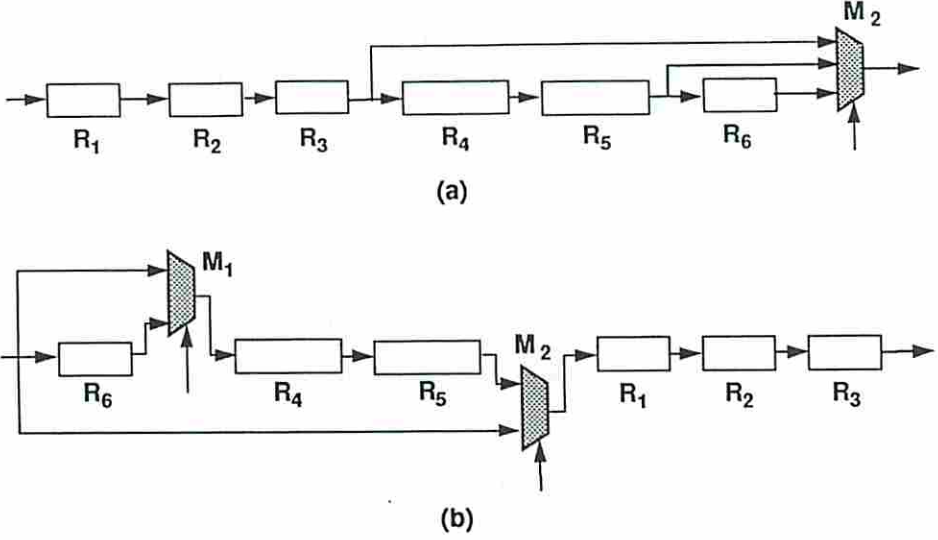


Figure 6.4: (a) Optimal configuration with 2 multiplexers as generated by algorithm. (b) Alternate optimal solution with 2 multiplexers.

The scan configurations shown in Fig. 6.2(a) and Fig. 6.4(b) represent alternate optimal solutions that also achieve the lower bounds on the chain cycles with 2 multiplexers. In both these configurations, we can realize the same three serial chains of the original configuration by appropriately controlling the multiplexers.

In general an optimal solution with j multiplexers divides the registers into $(j+1)$ disjoint segments with boundaries defined by the scan-in/scan-out pins, and the bypass points of the *j+1-to-1* multiplexer. Treating each of these segments as a single entity, we can permute them in any order within the chain and appropriately insert j multiplexers to generate an alternate optimal solution. For our example design, an optimal solution with 2 multiplexers consists of the three segments $\{R_1, R_2, R_3\}$, $\{R_4, R_5\}$ and $\{R_6\}$ from the scan-in to scan-out pin. The configuration in Fig. 6.2(a) reverses the order of the first two segments while the configuration in Fig. 6.4(b) reverses the order of all three segments.

Lemma 9 Consider any optimal solution with a *j+1-to-1* multiplexer at the scan-out pin. Every permutation of the $(j+1)$ disjoint segments in the chain can be used to derive an alternate optimum by appropriately inserting j multiplexers.

Proof Denote the disjoint segments in the optimal configuration as $S_{j+1}, S_j, \ldots, S_1$ from the scan-in pin to the scan-out pin. Any alternate optimum must realize the

same serial scan chains as in the original configuration. However the order of registers within the chains can be different. Consider any permutation of the $(j+1)$ segments. Insert the first multiplexer to bypass the segment S_1 in the given permutation. The two settings of this multiplexer will realize two of the serial scan chains in the original configuration. The second multiplexer is used to bypass the segment S_2. If S_1 and S_2 lie adjacent to each other, the position of this multiplexer can be modified to bypass both segments. Continuing in this fashion, the αth multiplexer $(1 \leq \alpha \leq j)$ is used to bypass segment S_k and any adjacent segments that have already been bypassed. Each of the $j+1$ serial chains can be realized by appropriately setting the control signals on the j multiplexers. □

Hence to optimally insert j multiplexers, a total of $(j+1)!$ different configurations can be considered. For each of these configurations, the order of registers within each segment does not affect the total test time. Given this range of optimal solutions, a designer has the flexibility to choose one that minimizes the routing overheads in configuring the chain.

6.5 Minimum Shift Policy

6.5.1 Problem Analysis

In Chapter 4 we showed that the problem of ordering registers to minimize the test time is NP-hard. We can consider a less constrained version of this problem, i.e., to determine an ordering of registers that satisfies the minimum chain cycles of each session. The minimum chain cycles correspond to those evaluated in Section 6.3.2, although we can equivalently use the ones derived in Chapter 4. Note that a solution that satisfies the minimum chain cycles will also minimize the test time, although the converse might not be true. The less constrained version represents a special case of our first optimization problem restated as a *decision* problem.

Satisfy-Chain-Cycles (SCC)

Does there exist an ordering of scan registers with the insertion of j $(j \geq 0)$ multiplexers to achieve the minimum chain cycles for each test session?

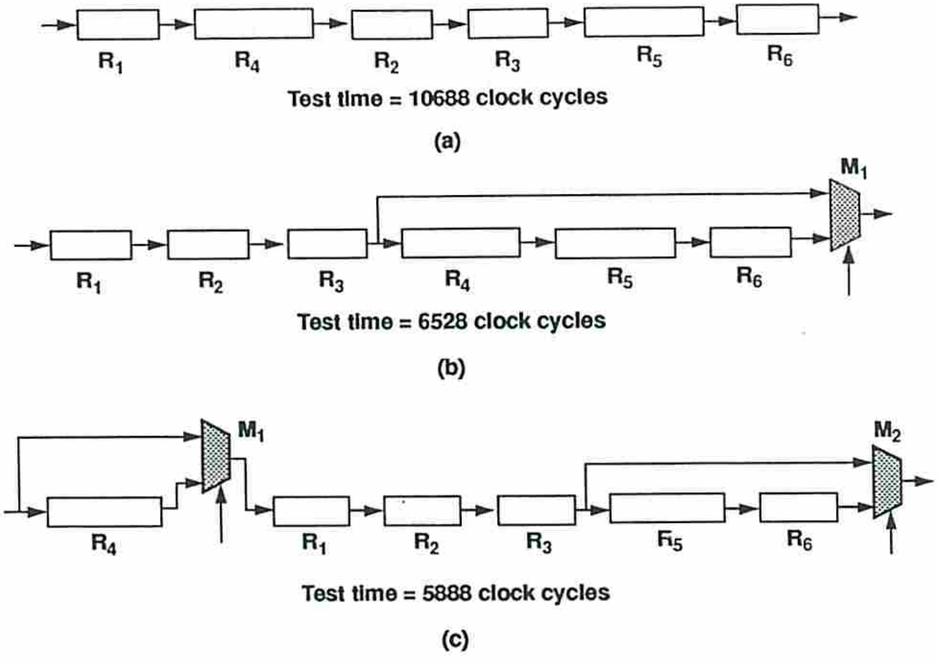


Figure 6.5: Optimal configurations for the minimum shift policy. (a) no multiplexer, (b) a single multiplexer, and (c) two multiplexers.

Theorem 2 *SCC* is *NP*-complete.

Proof Consider the case where no bypass multiplexers are added to the scan chain, i.e., $j = 0$. We transform the known NP-complete problem *Subset Sum* [43] to this case of *SCC*. A detailed proof is given in Appendix II. □

Since our second optimization problem is as difficult as *SCC*, its *NP*-hardness follows from the above proof. The main contribution to the complexity of *SCC*, as seen from the proof in Appendix II, arises from the variations in the lengths of the registers. In fact, if all registers are of equal length, a polynomial-time algorithm for *SCC* with $j = 0$ can be obtained from the solution to a similar problem, *Sequencing with Release Times and Deadlines* [46]. However even with equal length registers, the problem of *minimizing* the test time with j ($j \geq 0$) multiplexers is complicated by the non-linearity of the *max* function used in determining the chain cycles.

Before discussing the solution techniques, we make one final observation about optimizing test time with the minimum shift policy. Unlike the flush policy, there

exists a close interaction between the ordering of registers and the placement of bypass multiplexers. With the flush policy the ordering of scan registers in the chain is initially fixed. For any given value of j, the problem is then reduced to inserting the multiplexers for this given ordering of registers. In other words decomposing the problem in this way did not affect the optimality of the final solution. However for the minimum shift policy, the ordering of the registers depends on the number of multiplexers to be inserted in the chain. For example in Fig. 6.5, we show three optimal scan configurations for the circuit of Fig. 6.1(a) with $j = 0$, 1 and 2 multiplexers, respectively. Note that the ordering of registers is different in all three cases.

Based on the intractability of the problem, it becomes essential to employ heuristic techniques to efficiently generate optimal solutions. Since our first optimization problem is a limiting case of the second problem, we concentrate on the latter in developing our solution techniques. Given j multiplexers to be used for reconfiguration, the complete problem is decomposed into two sub-problems: (1) enumerate a subset of all possible orderings of the registers, and (2) for each ordering, optimally insert j multiplexers to minimize the test time. The best solution among these scan configurations is chosen as the "optimal" solution.

6.5.2 Placement of Bypass Multiplexers

Let us initially analyze the second sub-problem in the above heuristic algorithm. Consider an ordering of registers for our example circuit as shown in Fig. 6.6. The driver weight, length and receiver weight of each register are displayed above, within and below the register, respectively. The different segments in the chain that can be potentially bypassed are shown below the figure. Note that a register with weight equal to 500 cannot be part of any bypass segment since the register will be used in every test session. We can prune this set of potential bypass segments into a smaller set of segments to serve as candidates for placing bypass multiplexers. This is done by considering the weight parameters of the registers in each segment. For example both the segments (5,5) and (5,6) contain register R_5 with weight 100, and hence they can only be bypassed at the beginning of session TS_3. Since the segment

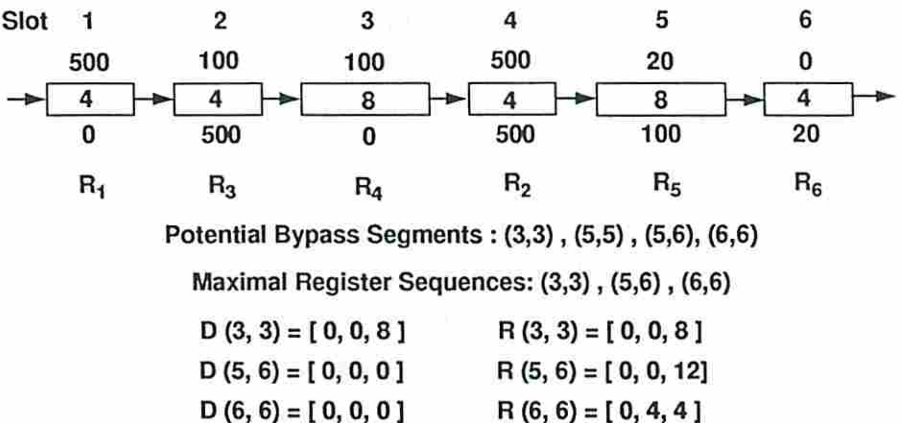


Figure 6.6: Maximal register sequences for scan chain ordering.

(5,5) is subsumed by (5,6), we can exclude the former as a candidate for placing a multiplexer.

Definition For a given ordering of registers in the chain, a *maximal* register sequence in session TS_i ($2 \leq i \leq n$) corresponds to a segment of adjacent registers that satisfies two properties: (1) every register in the segment has weight less than W_i, and (2) the segment is not a subset of any other segment satisfying the first property.

For example in the scan chain of Fig. 6.6, the segments (3,3) and (5,6) are maximal register sequences in the third session TS_3. It is sufficient to consider only the maximal register sequences in each session as candidates for placing bypass multiplexers. This follows as a consequence of Lemma 5 and the definition of a maximal sequence. For any ordering of registers in the chain, the next lemma provides an upper bound on the total number of maximal register sequences over all sessions.

Lemma 10 Given any ordering of the M scan registers, an upper bound on the total number of maximal register sequences is given by $M - N(W_n)$, where $N(W_n)$ is equal to the number of registers with weight equal to W_n.

Proof A register with weight equal to W_n cannot be included in a bypass segment. Consider any maximal register sequence (f_α, l_α) for test session TS_i ($2 \leq i \leq n$). Any other maximal sequence for the same session must be such that both segments are disjoint. Similarly a maximal register sequence (f_β, l_β) for a subsequent test

session must either be disjoint with (f_α, l_α) or such that $f_\beta \leq f_\alpha$ and $l_\beta \geq l_\alpha$. In any case, the maximal sequence (f_β, l_β) must contain at least one additional register not included in (f_α, l_α). Hence the total number of maximal register sequences for all sessions must be less than or equal to $M - N(W_n)$. □

To evaluate all maximal register sequences for a given ordering, we require $n-1$ passes (for test sessions TS_2 to TS_n) of the M registers in the chain. In each pass the maximal register sequences for the corresponding session are determined. For the example in Fig. 6.6, the first pass will identify segment (6,6) and the second pass will identify segments (3,3) and (5,6). Let us evaluate the reduction in test time if a multiplexer is used to bypass the segment (3,3). The segment (3,3) is a maximal register sequence in TS_3 and hence it will be bypassed prior to the start of session TS_3. Since the maximum driver weight among all registers to the right of this segment is 500, the drive cycle of session TS_3 will be decreased by 8 clock cycles. Similarly since the maximum receiver weight of all registers to the left of the segment is 500, the receive cycle of TS_3 will also reduce by 8. Hence with each maximal register sequence (f_k, l_k) we associate two arrays each of length n, a driver array $D(f_k, l_k)[1, \ldots, n]$ and a receiver array $R(f_k, l_k)[1, \ldots, n]$. The value of $D(f_k, l_k)[i]$ ($R(f_k, l_k)[i]$) represents the decrease in the drive (receive) cycle for test session TS_i if a multiplexer is used to bypass the segment (f_k, l_k). For the scan chain ordering in Fig. 6.6, the arrays for each maximal register sequence are shown.

Lemma 11 Consider any two maximal register sequences (f_α, l_α) and (f_β, l_β). If two multiplexers are used to bypass both segments, the resulting decrease in the drive cycle of test session TS_i is equal to (1) $D(f_\alpha, l_\alpha)[i] + D(f_\beta, l_\beta)[i]$ if both segments are disjoint, else (2) $max\ (D(f_\alpha, l_\alpha)[i], D(f_\beta, l_\beta)[i])$. An analogous result is valid for the receive cycle of TS_i. □

Consider the segments (5,6) and (6,6) for our example ordering of registers. The receiver arrays for the two segments are [0, 0, 12] and [0, 4, 4] respectively. If multiplexers are used to bypass both these segments, the decrease in the receive cycles of TS_1, TS_2 and TS_3 will be 0, 4 and 12, respectively. If a third multiplexer is now used to bypass segment (3,3), the overall reductions in the receive cycles of the three sessions will be 0, 4 and 20, respectively. In this way, given any j maximal

register sequences and their associated driver and receiver arrays, Lemma 11 can be used to evaluate the cumulative reductions in the drive and receive cycles for each test session. Given any permutation π of the registers in the chain, the following procedure optimally inserts j multiplexers to minimize the total test time.

Insert-Bypass-Muxes (Inputs: π, j)
begin

1. Let $\mathcal{P}$ = set of all maximal register sequences.
2. For each test session TS_i $(1 \leq i \leq n)$
   (a) Evaluate drive and receive cycles for non-reconfigured chain.
   (b) Generate maximal register sequences for session TS_i.
   (c) Evaluate driver and receiver arrays for each maximal sequence.
   (d) Add the maximal register sequences to $\mathcal{P}$
3. Let *Best-test-time* denote current best solution.
4. Initialize *Best-test-time* $= \infty$.
5. For every subset of $\mathcal{P}$ with cardinality j
   (a) Invoke Lemma 11 to obtain cumulative reductions in drive and receive cycles.
   (b) Evaluate *Resulting-test-time* for chosen subset of bypass segments.
   (c) If (*Resulting-test-time* < *Best-test-time*)
      - *Best-test-time* = *Resulting-test-time*.
      - Save chosen subset of bypass segments.
6. Return *Best-test-time* and corresponding j bypass segments.

end

The worst-case time complexity in generating all maximal register sequences, and evaluating their driver and receiver arrays is of order $O(Mn)$. Given any j of these segments, determining the resultant test time is of order $O(jn)$ by successive application of Lemma 11. To optimally insert j multiplexers, every subset of the maximal sequences with cardinality j will need to be explicitly enumerated. By Lemma 10, the number of such subsets is upper bounded by an order $O(M^j)$. Although exponential in j, practical limitations in the logic overheads of reconfiguring the chain will impose an upper bound on the number of multiplexers that can be introduced. As our experimental results suggest, 2-3 multiplexers are sufficient in most cases to achieve the desired reductions in test time.

6.5.3 Ordering the Registers

6.5.3.1 Restricting the Search Space

To complete the description of our heuristic procedure, we turn our attention to generating an appropriate subset of all orderings of registers in the chain. For each ordering, the procedure *Insert-Bypass-Muxes* is invoked with the desired value of j. To restrict the search space of all orderings, it is possible to exploit certain properties of the registers. In particular the *adjacency interchange* property described in Chapter 4 can be used to determine the relative positions of two registers. With a reconfigured scan chain however, we need to impose an additional constraint in employing this property.

Consider two registers R_α and R_β such that $d_\alpha \leq d_\beta$ and $r_\alpha \geq r_\beta$. In an optimal single scan chain, if $p_\alpha < p_\beta$, we can interchange the positions of the two registers if either $l_\alpha = l_\beta$, or $p_\alpha = p_\beta - 1$. As shown in Chapter 4, this will not increase the chain cycle of any test session. However in an optimally reconfigured scan chain, interchanging the positions of the two registers will modify the order of registers in the different serial scan chains. This in turn might increase the chain cycle of one or more sessions. For example consider R_3 and R_4 in the optimal configuration shown in Fig. 6.5(b). The two registers satisfy the conditions of the adjacency interchange property. By interchanging their positions the chain cycle of the third

session increases, since the longer scan chain consisting of all six registers will need to be used in the session.

Therefore the adjacency interchange property as stated in Chapter 4 is not directly applicable to a reconfigured scan chain. However the property can be applied if we ensure that the two registers involved in the interchange have equal weight parameters. In other words, the two registers are used in the same number of sessions. This idea is generalized in the following lemma.

Lemma 12 In an optimally reconfigured scan chain, consider two registers R_α and R_β that satisfy the conditions of the adjacency interchange property. The positions of the two registers can be interchanged provided they are of equal weight W_i $(1 \leq i \leq n)$.

Proof Both R_α and R_β will only be used in sessions TS_1 to TS_i. If $l_\alpha = l_\beta$, the chain cycle of every session from TS_1 to TS_i cannot increase. Similarly the drive and receive cycles for every serial scan chain used in sessions TS_{i+1} to TS_n will remain unchanged. Consider the case when $l_\alpha \neq l_\beta$. Since the two registers lie in adjacent slots, interchanging their positions will not increase the chain cycle of any session from TS_1 to TS_i. From our definition of a maximal register sequence, either both or neither of the registers will be part of a serial chain employed in any session from TS_{i+1} to TS_n. Hence interchanging their positions will have no effect on the chain cycles of these sessions. □

Note that the above lemma is independent of the number of multiplexers used to reconfigure the chain, and their positions in the configuration. The ordering of registers with any number of multiplexers must therefore satisfy the conditions of the lemma. Similar to the use of a precedence graph in Chapter 4, the lemma can be used in a pre-processing step to restrict the search space in ordering the registers. It induces a precedence relation on every pair of registers with equal weight provided they are also of equal length. However in the case of registers with weight equal to W_n, a precedence relation can be generated independent of the lengths of the registers. This generalizes the *dominance property* introduced in Chapter 4.

Lemma 13 Consider any two registers R_α and R_β of weight W_n, where $d_\alpha \leq d_\beta$ and $r_\alpha \geq r_\beta$. In an optimally reconfigured scan chain, R_β can be placed in a slot

closer to the scan-in pin than R_α. In addition, if there exists a register R_θ with $d_\theta = W_n$ (0) and $r_\theta = 0$ (W_n), the register can be placed in slot 1 (M) in an optimal configuration.

Proof Consider an optimal configuration in which R_α and R_β are placed in slots p_α and p_β respectively, and let $p_\alpha < p_\beta$. Since both registers cannot be in any segment bypassed by a multiplexer, the serial scan chain employed in every session must include both registers. Consider the case when $d_\alpha = W_n = d_\beta$ and $r_\alpha \geq r_\beta$. Insert R_β in position p_α by moving every register along with any bypass multiplexers in positions p_α to $p_{\beta-1}$ one slot to the right. For any serial chain used in a session, the effect of this change will be to increase the number of clock cycles required to access every shifted register from the scan-in pin by an amount equal to l_β. *However the drive cycle of the session cannot increase since it must have been at least equal to the number of clock cycles required to access R_β in its original position at p_β.* Hence the total test time cannot increase as a result of the change in the position of R_β. A similar proof can be constructed for each of other three cases, i.e., $r_\alpha = W_n = r_\beta$, $d_\alpha = W_n = r_\beta$ and $r_\alpha = W_n = d_\beta$. Finally note that a similar argument furnishes a proof for inserting a register R_θ with $d_\theta = W_n$ (0) and $r_\theta = 0$ (W_n) in slot 1 (M) without increasing the chain cycle of any session. □

Based on the above two lemmas, we can consider a restricted set of orderings for the circuit in Fig. 6.1(a), i.e., R_1 can be placed in slot 1, R_2 must precede R_3 and similarly R_4 must precede R_5. Hence the number of possible orderings is reduced from 720 (6!) to 30. In general the reduction in the search space provided by the above two lemmas greatly depends on the relative weights and lengths of the registers in the design.

6.5.3.2 Neighborhood Search Technique

Rather than investing our efforts in generating a single optimal ordering, we will emphasize the efficient generation of "good" orderings that attempt to minimize the overall test time. Our main motivation for this is to satisfy any constraints imposed by routing area on the order of registers in the chain. Generating a number of solutions with minimal test time will increase the probability of satisfying such

constraints while providing for an effective mechanism to tradeoff routing area with test time. In addition for designs with a large number of registers, effective heuristics will require much less computation time as compared to enumerative techniques. For example the computational requirements of the implicit enumeration technique in Chapter 4 limit the applicability of the algorithm to designs with less than about 20 registers.

We employ a neighborhood search technique commonly used in sequencing algorithms [44]. The basic elements in neighborhood search are the concept of a neighborhood of a sequence and a mechanism for generating neighborhoods. The generating mechanism is a method of transforming one sequence, referred to as the *seed*, into a collection of related sequences. The following procedure outlines the three main steps to generate a scan chain reconfigured with j multiplexers.

Neighbourhood-Search (Inputs: S, j)

1. Generate an initial seed (ordering of registers in the chain) and optimally insert j multiplexers to evaluate the test time.

2. Optimally insert j multiplexers for each sequence in the neighborhood of the seed. If no sequence has a test time lower than the seed, stop the search.

3. Select the sequence in the neighborhood with the least test time and initialize it as the seed. Return to 2.

The above procedure provides great flexibility to generate a number of locally optimal solutions by appropriately choosing both the seed and the neighborhood structure. An intuitive approach to generate a "good" seed is to place frequently accessed drivers (receivers) close to the scan-in (scan-out) pin and driver-receivers close to the middle of the scan chain. To quantify the role of a register R_p, a cost function is assigned to the register given by the difference of its driver and receiver weights, i.e., $(d_p - r_p)$. A large positive (negative) value of the cost function for a register suggests it is predominantly used as a driver (receiver) while a value close to zero signifies its dual role as a driver and receiver. The seed is generated by ordering the registers in non-increasing values of the cost function from scan-in to

scan-out. In the case of two or more registers with equal value, the ordering is arbitrarily decided. Note that the ordering of registers in this seed satisfies the precedence relations specified in Lemmas 12 and 13. For the circuit of Fig. 6.1(a), the initial seed is obtained by placing the registers in the order $R_1, R_4, R_2, R_6, R_5, R_3$ from scan-in to scan-out. We use this seed in Fig. 6.7 to illustrate the neighborhood search technique in generating an optimal configuration with a single multiplexer.

The proofs in Lemmas 12 and 13 provide two obvious ways to perturb a seed in generating a neighborhood structure; (1) a pairwise interchange of every two registers in the chain, and (2) insertion of a register between every pair of registers. Note that both mechanisms individually generate neighborhoods of size $O(M^2)$. Based on this observation, we divide the total search process into two phases. In Phase I the positions of every two registers are interchanged if they do not violate the precedence relations established in Lemmas 12 and 13. For example in Fig. 6.7, as a result of Lemma 13, R_2 and R_3 are not interchanged in the course of searching the neighborhood. In Phase II the locally optimal solution at the end of Phase I is used as a seed for the second neighborhood mechanism obtained by inserting every register between every pair of registers. As in the previous case, we only consider insertions that satisfy the precedence relations among the registers. For our example circuit, the initial seed with a single multiplexer is shown in Fig. 6.7. The solution at the end of Phase I is obtained by interchanging R_2 and R_4. In Phase II three neighborhoods are searched before converging to the locally optimal solution shown in Fig. 6.7. The ordering of registers in this configuration is obtained by inserting R_3 between R_2 and R_4, followed by the insertion of R_5 between R_4 and R_6.

As with all neighborhood techniques, there is in general no way to predict whether a locally optimal solution is also a global optimum. In our example, explicit enumeration of all 30 orderings will confirm that our final solution is also globally optimal. To increase the probability of converging to a global optimal, the neighborhood search procedure can be repeatedly invoked on a number of randomly generated seeds, each of which satisfy the precedence relations of Lemmas 12 and 13. In addition we can also employ the best ordering as obtained for $j-1$ multiplexers as a seed for the case of inserting j multiplexers. The best solution after invoking the neighborhood search on all these seeds is chosen as the optimum. To evaluate the

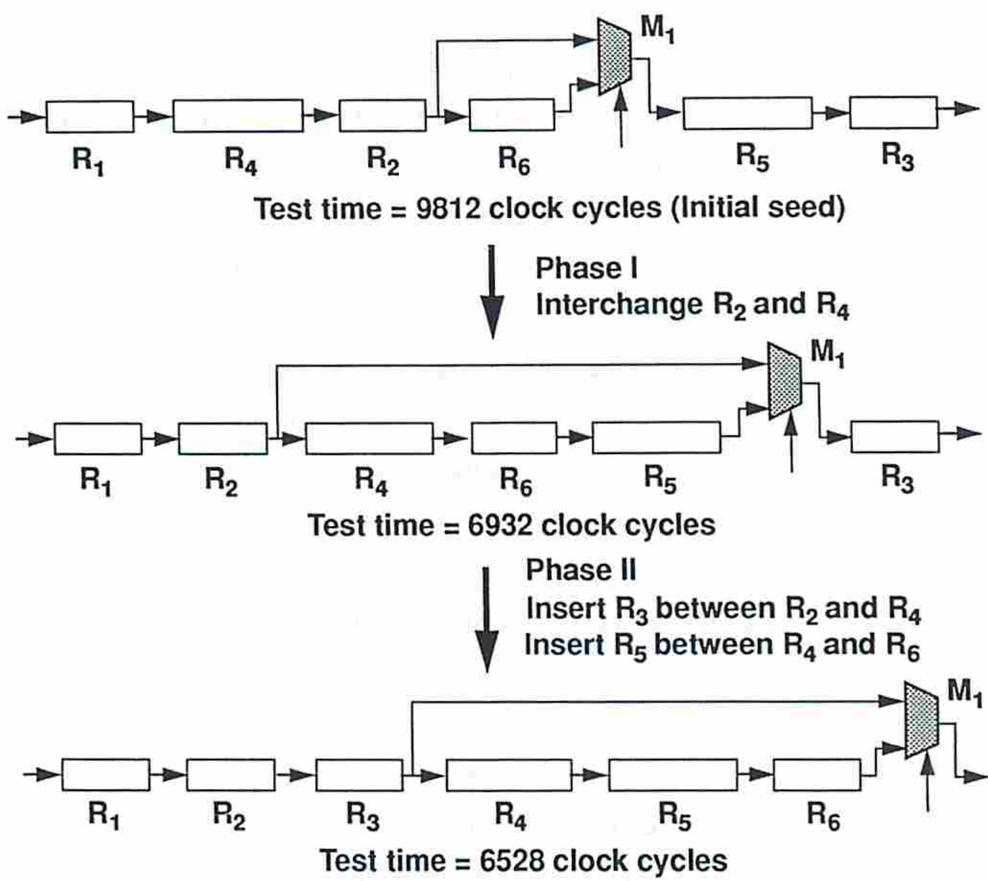


Figure 6.7: Neighborhood search with deterministic seed.

quality of the final solution, we compare the result with the lower bound on test time under the minimum shift policy.

If placement information about the physical locations of the scan registers is available, routing constraints can be incorporated in the search technique. For example a cost function of the total wire length can be used both to constrain the initial seed as well as exclude certain sequences generated in the neighborhood search.

6.6 Experimental Results

6.6.1 Case Studies

To illustrate our results we employ two data path circuits, (1) the *USC Data Path-I* and (2) a pipelined implementation of the *AR-Filter Element* generated by the ADAM system [47]. A portion of the *AR-Filter Element* is shown in Fig. 6.8. This circuit contains 16 multipliers, 12 adders and 72 8-bit registers. Due to the large degree of parallelism in the circuit, there exists many cascades of registers directly feeding each other as shown in the figure. Any such cascade consisting solely of registers and the wires between them are classified as *trivial* kernels. The chosen filter implementation is a feed-forward circuit. We obtain partial scan designs by appropriately scanning registers that partition the entire circuit into smaller portions of logic. Each portion of logic is then handed individually to a test pattern generation system. The details in generating these partial scan designs will be discussed in Chapter 9 when dealing with the integrated scan design system.

Table 6.2 shows the details of four scan designs of the *USC Data Path-I* and four designs of the *AR-Filter Element*. In each case the scan selection methodology, the number of kernels, their test lengths, the maximum sequential depth among all kernels, and the number of scan registers are shown. For the *AR-Filter Element*, the test lengths for the adder and multiplier are equal to 16 and 69, respectively. The test length of the trivial kernel consisting of a single wire is equal to 2. Note that the number of different test lengths is less than the number of kernels since there exists multiple instances of the same kernel in each design. Designs 3, 4, 6, 7 and 8 represent boundary scan versions of the original circuits in which registers are

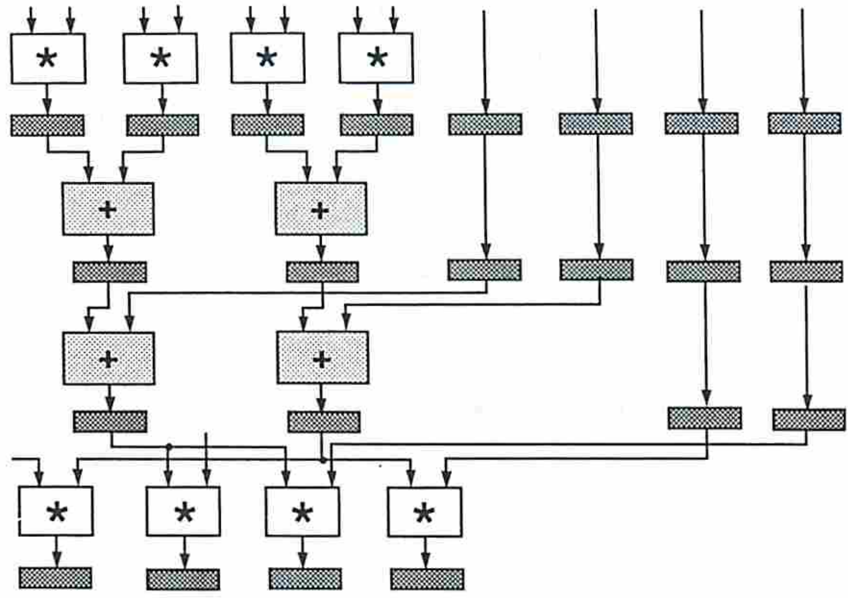


Figure 6.8: Portion of *AR-Filter Element.*

introduced at all primary inputs/outputs and at all control inputs. In these designs, except for Design 6, partial scan is also employed.

In Table 6.3 we compare the test times (in clock cycles) of the combined scheme and the overlapped scheme under the flush policy. The number of multiplexers j is varied from 1 to λ, where λ is the minimum number required to achieve the lower bounds on the chain cycles. The % red. column represents the savings in test time over the combined scheme and the % incr. column denotes the deviation from the lower bound on test time under the flush policy (shown in the second column).

In Table 6.4, for the minimum shift policy, we show the results of the neighborhood search technique and multiplexer insertion algorithm for each of the eight designs. Three different random seeds and two deterministic seeds are used. Det. 1 employs the static ordering as outlined in Section 6.5.3 and Det. 2 uses the best ordering obtained with $j-1$ multiplexers as a seed for inserting j multiplexers. The CPU times, obtained on a SPARC II workstation, denote the total time required for all five seeds. For each design, the number of bypass multiplexers is increased, until the resultant test time with at least one of the seeds is within 1-2% of the lower bound under the minimum shift policy. The seed Det. 2 achieves this requirement

Design index	Type of scan	No. of kernels	Test lengths	Max. seq. depth	No. of scan registers (ffs)
Scan designs of the *USC Data Path-I*					
1	Full	7	10,11,14,17 18,135,179	0	10 (104)
2	Full	7	10,11,14,17 18,135,**500**	0	10 (104)
3	Partial	4	17,29,141 179	1	17 (126)
4	Partial	4	21,29,135 179	2	17 (134)
Scan designs of the *AR-Filter*					
5	Full	24 + 36 trivial	2,16,69 109	0	72 (576)
6	Full	24 + 36 trivial	2,16,69 109	0	106 (856)
7	Partial	20 + 8 trivial	2,16,23 69,109	6	66 (520)
8	Partial	24 + 8 trivial	2,16,69 109	5	68 (552)

Table 6.2: Eight scan designs of *USC Data Path-I* and *AR-Filter Element*

in most of the designs and in general the deterministic seeds provide better solutions than the random seeds. The quality of the local optima is largely independent of the seed for designs with a small number of registers, highlighting the *strength* of the neighborhood mechanisms. However in Designs 4-8 which contain a large number of registers, the mechanisms are unable to sufficiently perturb the random seeds to reduce test time even with the addition of more multiplexers.

In Table 6.5 the best solution obtained with the five seeds is compared with both the test time of the combined scheme, and the lower bound on test time under the minimum shift policy, in a format similar to Table 6.3. From Tables 6.3 and 6.5 we see that the savings in test time with the minimum shift policy are on average 25-35% more than with the flush policy. For both shift policies note the effect of increasing the test length of the *ALU* by comparing the test time reductions for Designs 1 and 2. The advantage of bypassing the less frequently used registers becomes more

Design index	Test time (lower bound)	Test time (combined)	Overlapped scheme			
			No. of muxes	Test time	% red.	% incr.
1	11371	18899	j = 1	12459	34.08	9.57
			j = 2	11403	39.66	0.28
			j = 3	11371	39.83	0.0
2	24532	52604	j = 1	29244	44.41	19.21
			j = 2	24564	53.30	0.13
			j = 3	24532	53.36	0.0
3	16430	23038	j = 1	18500	19.70	12.60
			j = 2	16562	28.11	0.80
			j = 3	16430	28.68	0.0
4	14151	24657	j = 1	15799	35.92	11.65
			j = 2	14391	41.64	1.70
			j = 3	14151	42.61	0.0
5	24365	63469	j = 1	30733	51.58	26.14
			j = 2	26925	57.58	10.51
			j = 3	24365	61.61	0.0
6	37125	94269	j = 1	50373	46.56	35.68
			j = 2	42053	55.39	13.27
			j = 3	37125	60.62	0.0
7	36031	57963	j = 1	43687	24.63	21.25
			j = 2	37431	35.42	3.89
			j = 3	36087	37.74	0.16
			j = 4	36031	37.84	0.0
8	36069	61374	j = 1	45285	26.21	25.55
			j = 2	36965	39.77	2.48
			j = 3	36069	41.23	0.0

Table 6.3: Results for eight scan designs (Flush policy)

significant with increasing disparity in the test lengths of the kernels. Observe that in most cases less than 3 bypass multiplexers are sufficient to converge to test times within 3% of the lower bounds for both shift policies. The tremendous savings in test time, coupled with the minimal impact on area overhead with the insertion of 1-3 multiplexers, shows that reconfiguration is both an effective and practical design methodology. In Table 6.5 for five of the designs, the test time reduces to within 10% of the lower bound even with no reconfiguration; this highlights the significant role played by the ordering of registers in optimizing test time. However a salient feature of these five designs is the absence of driver-receivers solely involved in testing kernels of larger test length, e.g., R_2 in the circuit of Fig. 6.1(a). Due to their dual roles in the minimum shift policy, such registers will generally be placed in the middle of the chain. Hence multiplexers will be required to access them when the larger kernels are being individually tested. Compare, for example, the test times of the configurations in Fig. 6.5(a) and Fig. 6.5(c). We will elaborate on this observation in the next subsection.

6.6.2 Influence of design characteristics

Although our two case studies demonstrate test time savings as large as 75% over the combined scheme, the potential advantages of reconfiguration can be appreciated in a more general setting. Consider a circuit having two kernels K_1 and K_2 with test lengths of W_1 and W_2 such that $W_2 > W_1$. Let L represent the total number of scan flip-flops in the circuit and assume l_1 of these flip-flops are *solely* involved in testing K_1. Under the combined scheme the total time spent in shifting patterns is equal to W_2L. For the overlapped scheme with the flush policy, the addition of a single multiplexer will lead to a shifting time of $W_1L + (W_2 - W_1)(L - l_1)$. The percentage savings in test time is given by the following expression.

$$
\begin{aligned}
\text{\% Savings} &= \frac{W_2L - W_1L - (W_2 - W_1)(L - l_1)}{W_2L} \times 100 \\
&= \frac{(W_2 - W_1)l_1}{W_2L} \times 100
\end{aligned}
$$

Design index	No. of muxes	Test time (clock cycles)					CPU time (in secs)
		Det. 1	Random 1	Random 2	Random 3	Det. 2	
1	j = 0	7803	8739	7803	7859	7803	3.6
	j = 1	7715	7771	7715	7715	7715	
	j = 2	7715	7771	7715	7659	7659	
2	j = 0	15828	15828	15828	15884	15828	3.5
	j = 1	15740	15828	15828	15884	15740	
	j = 2	15740	15684	15732	15884	15684	
3	j = 0	14152	14156	14248	14156	14152	47.7
	j = 1	11924	11828	14248	14156	11924	
	j = 2	10948	10948	11220	10900	10948	
	j = 3	10708	10948	11220	10900	10708	
	j = 4	10708	10948	11220	10900	10576	
4	j = 0	9417	9417	9417	9417	9417	20.1
	j = 1	9009	9417	9417	9417	9025	
	j = 2	9009	9417	8785	8657	8633	
5	j = 0	20525	20525	20525	21805	20525	1122
	j = 1	16717	20525	20525	21805	16717	
	j = 2	16717	20525	16717	21805	16493	
6	j = 0	31013	31293	31405	32421	31013	2801
	j = 1	26085	31293	31405	32421	26309	
	j = 2	26085	31293	26001	32421	26001	
7	j = 0	26727	27175	28791	26727	26727	605
	j = 1	25383	27175	28791	26727	25383	
	j = 2	25383	27175	28791	25215	25271	
8	j = 0	25925	26205	26501	26037	25925	564
	j = 1	25029	26205	26501	26037	25253	
	j = 2	25029	26205	26501	26037	24945	

Table 6.4: Neighborhood search with five seeds (Minimum shift policy)

Design index	Test time (lower bound)	Test time (combined)	Overlapped scheme			
			No. of muxes	Test time	% red.	% incr .
1	7659	18899	j = 0	7803	58.71	1.88
			j = 1	7715	59.18	0.73
			j = 2	7659	59.47	0.0
2	15684	52604	j = 0	15828	69.91	0.92
			j = 1	15740	70.08	0.36
			j = 2	15684	70.18	0.0
3	10480	23038	j = 0	14152	38.57	35.04
			j = 1	11828	48.66	12.86
			j = 2	10900	52.69	4.01
			j = 3	10708	53.52	2.18
			j = 4	10576	54.09	0.92
4	8633	24657	j = 0	9417	61.81	9.08
			j = 1	9009	63.46	4.35
			j = 2	8633	64.99	0.0
5	16493	63469	j = 0	20525	67.66	24.45
			j = 1	16717	73.66	1.36
			j = 2	16493	74.01	0.0
6	25641	94269	j = 0	31013	67.10	20.95
			j = 1	26085	72.33	1.70
			j = 2	26001	72.42	1.40
7	25051	57963	j = 0	26727	53.89	6.69
			j = 1	25383	56.21	1.33
			j = 2	25215	56.50	0.65
8	24809	61374	j = 0	25925	57.76	4.50
			j = 1	25029	59.22	0.89
			j = 2	24945	59.36	0.55

Table 6.5: Results for eight scan designs (Minimum shift policy)

	K_1	K_2	K_3	K_4	K_5	K_6	K_7
	$f_1 = 0.01$	$f_2 = 0.2$	$f_3 = 0.4$	$f_4 = 0.5$	$f_5 = 0.75$	$f_6 = 0.9$	$f_7 = 1.0$
1	$\gamma_1 = 0.143$	$\gamma_2 = 0.143$	$\gamma_3 = 0.143$	$\gamma_4 = 0.143$	$\gamma_5 = 0.143$	$\gamma_6 = 0.143$	$\gamma_7 = 0.143$
2	$\gamma_1 = 0.01$	$\gamma_2 = 0.02$	$\gamma_3 = 0.04$	$\gamma_4 = 0.06$	$\gamma_5 = 0.12$	$\gamma_6 = 0.25$	$\gamma_7 = 0.5$
3	$\gamma_1 = 0.5$	$\gamma_2 = 0.25$	$\gamma_3 = 0.12$	$\gamma_4 = 0.06$	$\gamma_5 = 0.04$	$\gamma_6 = 0.02$	$\gamma_7 = 0.01$
	$f_1 = 0.01$	$f_2 = 0.02$	$f_3 = 0.7$	$f_4 = 0.8$	$f_5 = 0.9$	$f_6 = 0.95$	$f_7 = 1.0$
4	$\gamma_1 = 0.02$	$\gamma_2 = 0.02$	$\gamma_3 = 0.03$	$\gamma_4 = 0.05$	$\gamma_5 = 0.1$	$\gamma_6 = 0.2$	$\gamma_7 = 0.58$
5	$\gamma_1 = 0.8$	$\gamma_2 = 0.1$	$\gamma_3 = 0.03$	$\gamma_4 = 0.02$	$\gamma_5 = 0.02$	$\gamma_6 = 0.02$	$\gamma_7 = 0.01$

Table 6.6: Five sets of values for f_i and γ_i

$$= \gamma(1-f) \times 100 \text{ where } \gamma = \frac{l_1}{L} \text{ and } f = \frac{W_1}{W_2}$$

To derive a similar expression for the minimum shift policy, assume that the flip-flops involved in testing a kernel are equally divided among the three roles of driver, receiver and driver-receiver. For the two kernel circuit, two multiplexers will be required to achieve the lower bound on test time with the minimum shift policy. The percentage savings over the combined scheme can be shown to be equal to $\frac{1}{3}(1+2\gamma) - \frac{2}{3}f\gamma$. For both policies note that as $\gamma \to 1$ and $f \to 0$, the percentage savings approaches 100%. Reconfiguration provides the greatest dividends when one kernel has a test length much smaller than the other but utilizes a greater percentage of the total scan flip-flops. For the given distribution of roles for a kernel's flip-flops, the savings with the minimum shift policy can be as large as 33% over the flush policy.

For our two data path circuits the results suggest that in most cases a maximum of 1-3 multiplexers are sufficient to achieve the desired reductions in test time. To clearly understand the variation in the overall savings with the addition of more bypass multiplexers, consider a circuit having n kernels with the sets $KSEQ$ and $WSEQ$ as defined in Chapter 3. Let l_i represent the number of flip-flops involved in testing K_i ($1 \leq i \leq n$), and let $L = \sum_{i=1}^{n} l_i$. By optimally inserting $n-1$ multiplexers, the percentage savings in test time with the flush policy over the combined scheme is given by

$$\sum_{i=1}^{n} \gamma_i(1-f_i) \times 100 \text{ where } \gamma_i = \frac{l_i}{L} \text{ and } f_i = \frac{W_i}{W_n}.$$

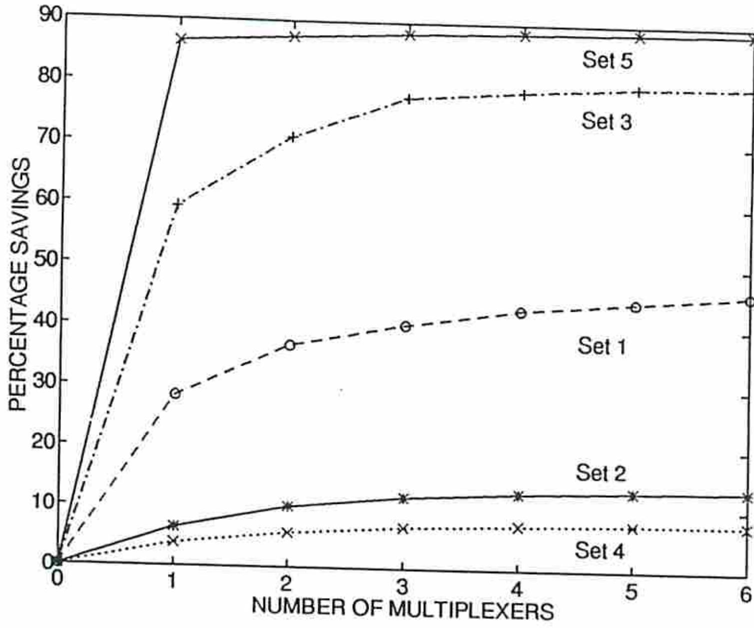


Figure 6.9: Savings in test time with the flush policy.

In Table 6.6 for the case of $n = 7$ kernels, we show five sets of values for f_i and γ_i. For each set, the graph in Fig. 6.9 plots the optimal savings in test time with the flush policy as the number of multiplexers is varied from 1 to 6. Consider the first set of values in which all flip-flops in the circuit are equally apportioned among the seven kernels. The addition of the first multiplexer provides 62% of the total possible savings and the curve highlights the "diminishing" returns with the insertion of every additional multiplexer. For the same values of f_i, Set 2 considers circuits in which a greater percentage of the flip-flops are involved in testing kernels of larger test length. The overall savings is reduced and the curve is more linear with the first multiplexer only providing 47% of the total possible savings. Set 3 reverses the distribution of the flip-flops among the kernels as compared to Set 2. The corresponding curve saturates after the addition of 3 multiplexers and the first multiplexer provides 74% of the total savings. Sets 4 and 5 employ values for f_i and γ_i that are more skewed both in the distribution of test lengths and flip-flops. The corresponding curves highlight two ends of the spectrum in terms of both the rate of saturation and the overall savings in test time.

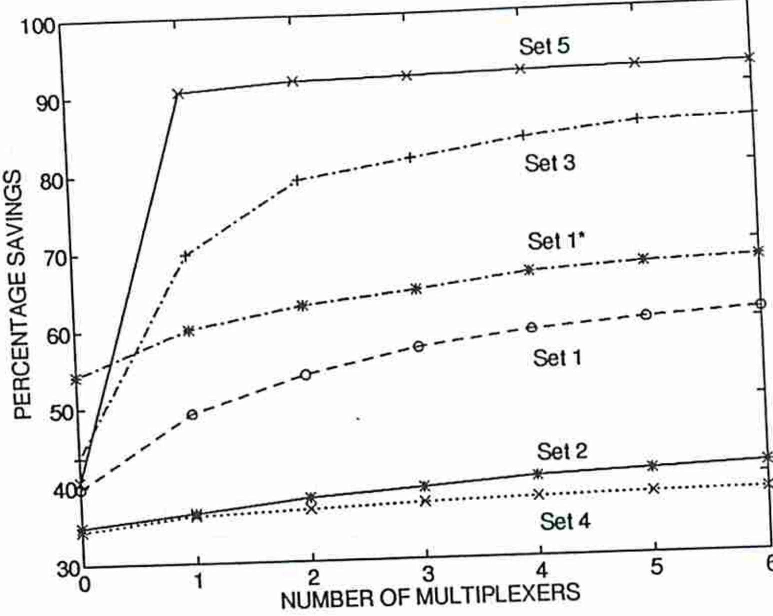


Figure 6.10: Savings in test time with the minimum shift policy.

A similar analysis can be performed for the minimum shift policy by assuming as in the two-kernel case that a kernel's flip-flops are equally divided among the three different roles. Fig. 6.10 plots the savings in test time for the five sets of values in Table 6.6 as the number of multiplexers is varied from 1 to 6. The general behavior of the plots is similar to that obtained with the flush policy. In particular note the large savings even in the absence of any reconfiguration. With the minimum shift policy, the savings in test time depend not only on the values of γ_i and f_i but also on the different roles played by the flip-flops associated with a kernel.

To illustrate this we modify the distribution in Set 1 by restricting flip-flops involved in testing the kernels with the largest two test lengths to only assume the roles of driver and receiver. Set 1* in Fig. 6.10 show the resultant savings in test time for this case. Observe both the faster rate of saturation and the difference in savings for any value of j as compared to Set 1. In the case of Set 1, the driver-receivers involved in testing the larger kernels will be placed in the middle of the chain since they play dual roles in a majority of the sessions. To achieve the lower bounds on the chain cycles in sessions solely devoted to testing the larger kernels, it is essential

to access these registers from both the scan-in and scan-out pins. The initial 1-2 multiplexers introduced to reconfigure the chain will be utilized for this purpose. However in the case of Set 1*, placing the drivers and receivers of the larger kernels at the ends of the chain obviates the need for these bypass multiplexers.

6.7 Summary

We have presented a structured and highly unique methodology of reconfiguring the chain to reduce the test time for scan designs. The basic idea is to bypass registers that are not frequently accessed in the test process. Both the flush and minimum shift policies provide a designer with different reconfiguration strategies to satisfy any constraints on test time and area overhead. It is essential for a designer to study the tradeoffs between test time and the logic and routing overheads in configuring the chain, prior to choosing a reconfiguration strategy. Detailed analysis and solution techniques have been outlined to optimize the test time for both policies. The algorithms provide effective ways to tradeoff test time with the logic overheads of the bypass multiplexers and any routing area constraints. We clearly highlight the potential advantages of reconfiguration to obtain tremendous savings in test time. Our experimental results on two data path circuits demonstrate significant reductions at the expense of 1-3 bypass multiplexers.

The previous three chapters have illustrated the wide range of chaining methodologies that can be used for a given scan design. Each methodology provides different tradeoffs in the associated design costs of the test application time, routing and logic area overhead, pin count, and control complexity. In Chapter 9 we will use different scan designs of an example data path circuit to compare the design costs of the various methodologies.

Chapter 7

Asynchronous Multiple Scan Chains

7.1 Introduction

In Chapter 5 we showed that breaking up a single scan chain into multiple chains can lead to large reductions in the test time. In employing these chains, a single *test mode signal* controls the operation of the scan flip-flops. Consider the design of a scan flip-flop as shown in Fig. 7.1(a) [1]. In this design T is a test mode control signal that determines which of the two data inputs is enabled. In the *normal* mode of operation ($T = 0$), data is latched in from the functional circuitry through D_1. In *test* mode ($T = 1$) data can be shifted in via D_2 from the previous flip-flop in the chain. The use of a single mode signal imposes a synchronization constraint in the operation of multiple chains, i.e., the chains are required to shift data simultaneously and apply this data to the circuit in the same clock cycle. In an optimal multiple chain configuration, recall that all scan chains might not be of the same length. For those sessions in the overlapped scheme in which all chains are active, test data can be shifted more quickly into the shorter chains as compared to the longer ones. However the use of a single mode signal forces the shorter chains to wait until the longer chains are flushed before data is applied to the kernels under test.

In this chapter we introduce the idea of **asynchronous** multiple chains in which groups of scan chains can operate independent of each other. This is achieved by using an additional mode signal to control the scan flip-flops in the design. Asynchronous multiple chains can provide greater reductions in the test application time

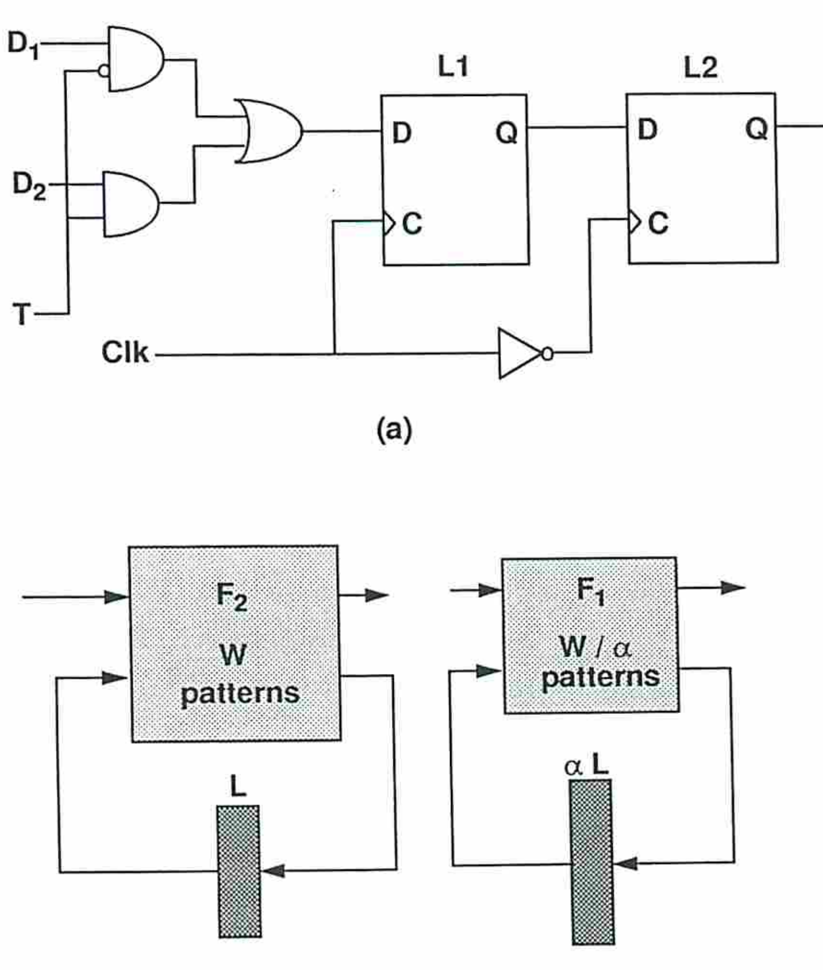


Figure 7.1: (a) Scan flip-flop design. (b) Circuit with two finite state machines.

as compared to the methodology in Chapter 5. In essence we exploit both the spatial parallelism in dividing a single chain into multiple chains, and the temporal parallelism in running these chains independent of each other. As we will illustrate in the implementation results, asynchronous chains also provide interesting design tradeoffs between the I/O pin count and test time.

7.2 Main Concepts

7.2.1 A Comparison of Chain Configurations

As with our multiple chain methodology we will assume (a) the scan registers in a design are broken down into flip-flops, and (b) a flush shift policy is used in the application of tests. In addition we will restrict our attention to designs that satisfy the circuit model outlined in Chapter 3. To illustrate the advantages of asynchronous multiple chains, consider the circuit in Fig. 7.1(b) consisting of two finite state machines K_1 and K_2. Assume a full scan methodology is used. The number of patterns and number of scan flip-flops used to test each kernel is shown in the figure. For K_1 (K_2), the values are given by W/α (W) and αL (L), respectively. Let us assume $\alpha \geq 1$, and W/α and αL are integers. We compare the test times spent in shifting patterns with a flush policy for different chain configurations. Each configuration employs a single test mode signal.

- *Single scan chain* - A single pattern is shifted into the scan chain for $L(1+\alpha)$ clock cycles before being applied to the circuit. Two pins are needed for scan-in and scan-out, and an additional pin for the test mode signal.
  Total shifting time $= WL(1+\alpha)$. *Pin count* $= 3$.
- *k equal length scan chains* - Breaking up a single chain into k equal length chains reduces the test time by a factor of k.
  Total shifting time $= W\lceil\frac{L(1+\alpha)}{k}\rceil$. *Pin count* $= 2k+1$.
- *2 optimal scan chains* - Using the dynamic programming algorithm, we can configure two optimal chains for the circuit. If $\alpha \leq 2$, an optimal configuration consists of two equal length chains. Else if $\alpha > 2$, the test time with the

overlapped scheme is minimized by assigning each kernel's flip-flops to separate chains. In the latter case, W/α patterns are applied to both machines by shifting in each pattern for αL clock cycles. The remaining $W - W/\alpha$ patterns are then applied to K_2 by shifting in each pattern for L clock cycles.

$$\textit{Total shifting time} = \begin{cases} W\lceil \frac{L(1+\alpha)}{2} \rceil & \text{if } \alpha \leq 2 \\ WL(2 - \frac{1}{\alpha}) & \text{if } \alpha > 2 \end{cases} \qquad \textit{Pin count} = 3.$$

- *Reconfigurable scan chain* - By optimally inserting a single multiplexer, each pattern is shifted for $L(1+\alpha)$ and L clock cycles in the first and second sessions, respectively.

  $\textit{Total shifting time} = 2WL. \qquad \textit{Pin count} = 3.$

- *Full parallel scan* - The largest reductions in test time can be obtained if each flip-flop in the circuit is provided with its own scan chain. The number of available pins on a device might however limit the applicability of this methodology.

  $\textit{Total shifting time} = W. \qquad \textit{Pin count} = 2L(1+\alpha) + 1.$

Instead of two synchronous chains, let us configure two asynchronous chains by assigning each machine's flip-flops to separate chains and providing each chain with an independent mode signal. This allows both machines to be tested concurrently and independent of each other. Hence with a single additional pin, the total shifting time reduces to $max(WL, (W/\alpha)(\alpha L)) = WL$ clock cycles, i.e., the test time is independent of α. In Fig. 7.2, as the value of α varies from 1 to 10, we compare the test times of 2, 3, 4 and 10 equal length chains, 2 optimal chains with a single mode signal, an optimally reconfigured chain, and 2 asynchronous chains. The test times are normalized with respect to the test time for a single scan chain under the combined scheme. As α increases, note that asynchronous chains provide savings in both test time and pin count. In addition the test time-pin count product with two asynchronous chains is less than the corresponding value with full parallel scan for $\alpha > 2$.

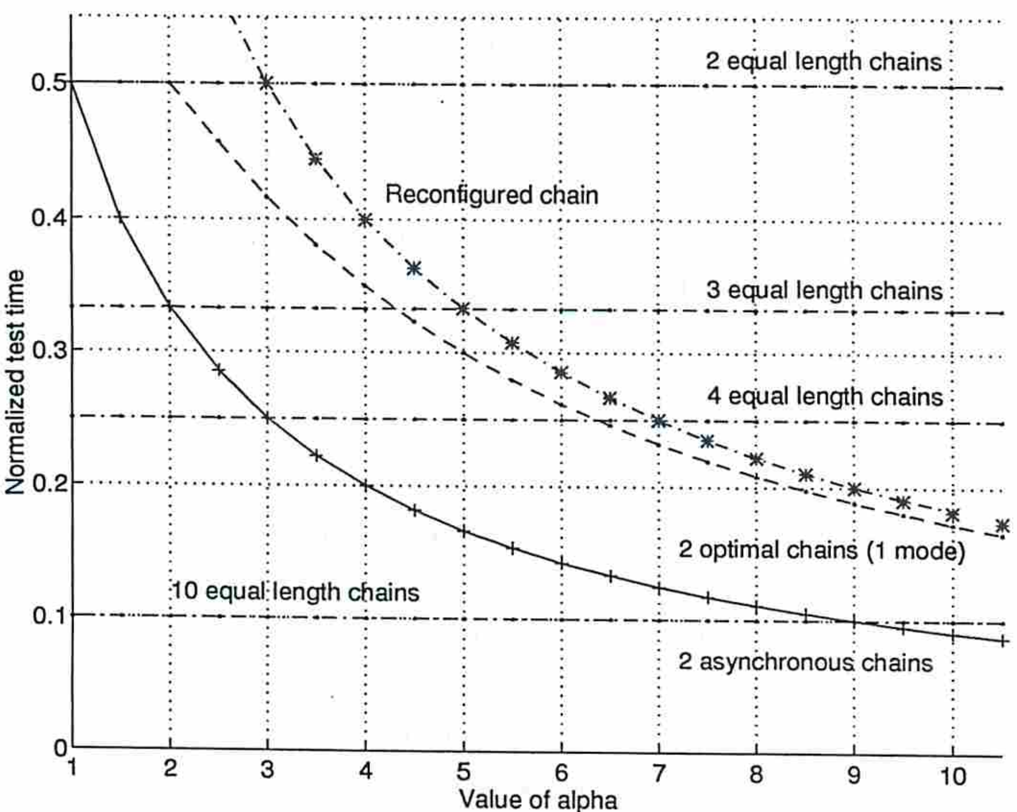


Figure 7.2: Comparing multiple chain configurations.

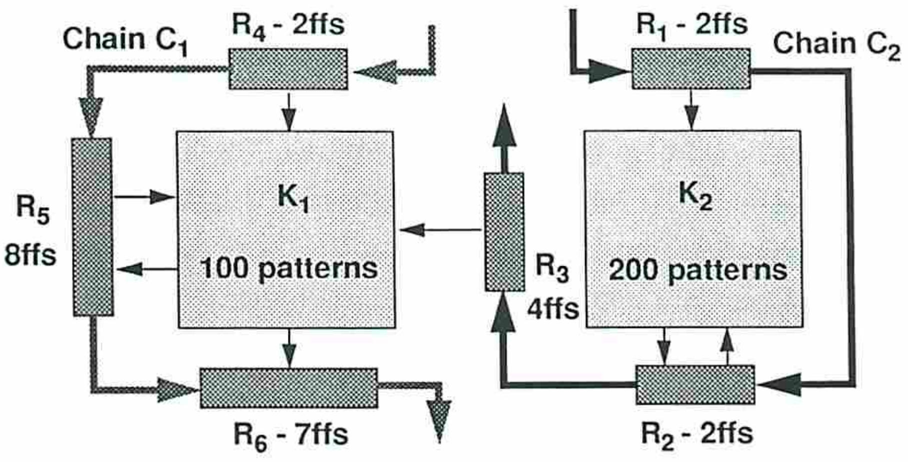


Figure 7.3: Example circuit design.

7.2.2 Use of Asynchronous Chains

Our simple example circuit illustrates the basic idea in employing asynchronous scan chains. However to fully appreciate their advantage, we need to *minimize* the associated test time in using asynchronous chains for a given design. In our previous three chain methodologies, an optimal test application scheme was determined prior to configuring the chains. As a result the overlapped scheme provided an objective function that was used in determining an optimal chain configuration. With asynchronous chains however, the overlapped scheme does not represent an optimal way to apply tests. The use of a single chain cycle in each session would defcat the purpose of two or more mode signals to control the chains. More importantly, with asynchronous chains, it is not possible to decouple the problem of configuring the scan chains from the associated problem of scheduling tests for the different kernels.

To illustrate this, consider the example design shown in Fig. 7.3 consisting of two kernels and six scan registers. The registers are configured into two chains C_1 and C_2, and each chain is provided with a separate mode signal. The **shift cycle** of a chain is defined as the number of clock cycles used to shift in a pattern and/or shift out results. Since the flush policy is used, the shift cycle of a chain must be greater than or equal to its length in terms of the number of flip-flops. An optimal way to apply tests with the configuration of Fig 7.3 is to employ a shift cycle of

8 for chain C_2 while simultaneously using a shift cycle of 16 for chain C_1. In this way a single pattern is applied to K_1 for every two patterns applied to K_2. The total test time is thus equal to 200(8 + 1) + 8 = 1808 clock cycles. Consider a modified chain configuration in which the register R_4 is assigned to chain C_2 instead of C_1. The lengths of C_1 and C_2 are now equal to 10 and 14, respectively. An optimal test scheme will now be equivalent to the overlapped scheme. In the first session, 100 patterns are applied to both kernels by employing a shift cycle of 14 for both chains. The remaining 100 patterns are then applied to K_2 by employing C_2 with a shift cycle of 10. The key point to note is that the chain configuration *determines* the manner in which tests are optimally applied to the circuit. With no knowledge of the scheme used in applying tests, it becomes difficult to formulate an objective function in configuring the scan chains. This close interaction between the scan chaining and test scheduling phases makes the problem of minimizing test time extremely complicated. A similar situation is encountered in Chapter 8 in configuring a single scan chain for a more generalized circuit model.

We adopt an approach that decomposes the problem of minimizing test time into two interrelated subproblems.

1. *Given any chain configuration of k scan chains with two mode signals, determine an efficient test scheme to organize the application of tests to the kernels while exploiting the asynchronous operation of the chains.*

2. *Based on this test scheme, determine both an assignment of the scan flip-flops to k chains and a distribution of two mode signals among the chains so as to minimize the associated test time.*

We will restrict our attention to the use of two mode signals in controlling the scan chains. The approach can be extended to more than two signals although the additional benefits in doing so greatly depend on the structure of the circuit under test. We employ a test application scheme referred to as the *two-phase* overlapped scheme. This is basically an extension to the overlapped scheme and will be discussed in the next subsection. The optimization techniques to configure the chains minimize the test time for this given scheme. The use of the two-phase overlapped scheme imposes certain restrictions on the manner in which the chains are employed. For

a given chain configuration, these restrictions can be relaxed to further reduce the overall test time. This post-processing step to enhance the test scheme will be discussed at the end of the chapter.

7.3 Two-Phase Test Application Scheme

Consider the circuit shown in Fig. 7.4(a) consisting of four combinational kernels K_1, K_2, K_3 and K_4 with test lengths of 20, 80, 200 and 500 respectively. The scan registers used in testing these kernels are configured into four scan chains as shown in Fig. 7.4(b).

Definition A **chain group** is a set of scan chains controlled by a single mode signal.

In our example, by distributing two mode signals among the chains, two chain groups are formed: $CG_1 = \{C_1, C_2\}$ and $CG_2 = \{C_3, C_4\}$. The test application scheme is divided into two phases, a synchronous phase and an asynchronous phase. Within each phase tests are applied to the respective kernels in the form of test sessions. In the synchronous phase we do not exploit the existence of chain groups and assume a single mode signal controls all chains. For our example the synchronous phase will consist of two sessions TS_1 and TS_2. In TS_1, 20 patterns are applied to all four kernels until K_1 is fully tested. For each test pattern in the session, the corresponding bit streams are simultaneously shifted into the four chains for a total of 24 clock cycles before activating the normal clock to propagate the pattern through the kernels.

In the second session TS_2, 60 patterns are applied to kernels K_2, K_3 and K_4 until K_2 is fully tested. All four chains are used and the chain cycle of the session is 24. Hence the total test time for the synchronous phase is equal to 20(24 + 1) + 60(24 + 1) = 2000 clock cycles. At this point, the two chain groups become "decoupled", i.e., every kernel yet to be fully tested only requires chains from a single chain group. Hence in the asynchronous phase the two chain groups are operated independently. Test patterns are applied to kernels tested solely by each chain group in the form of test sessions. In our example 120 patterns are applied

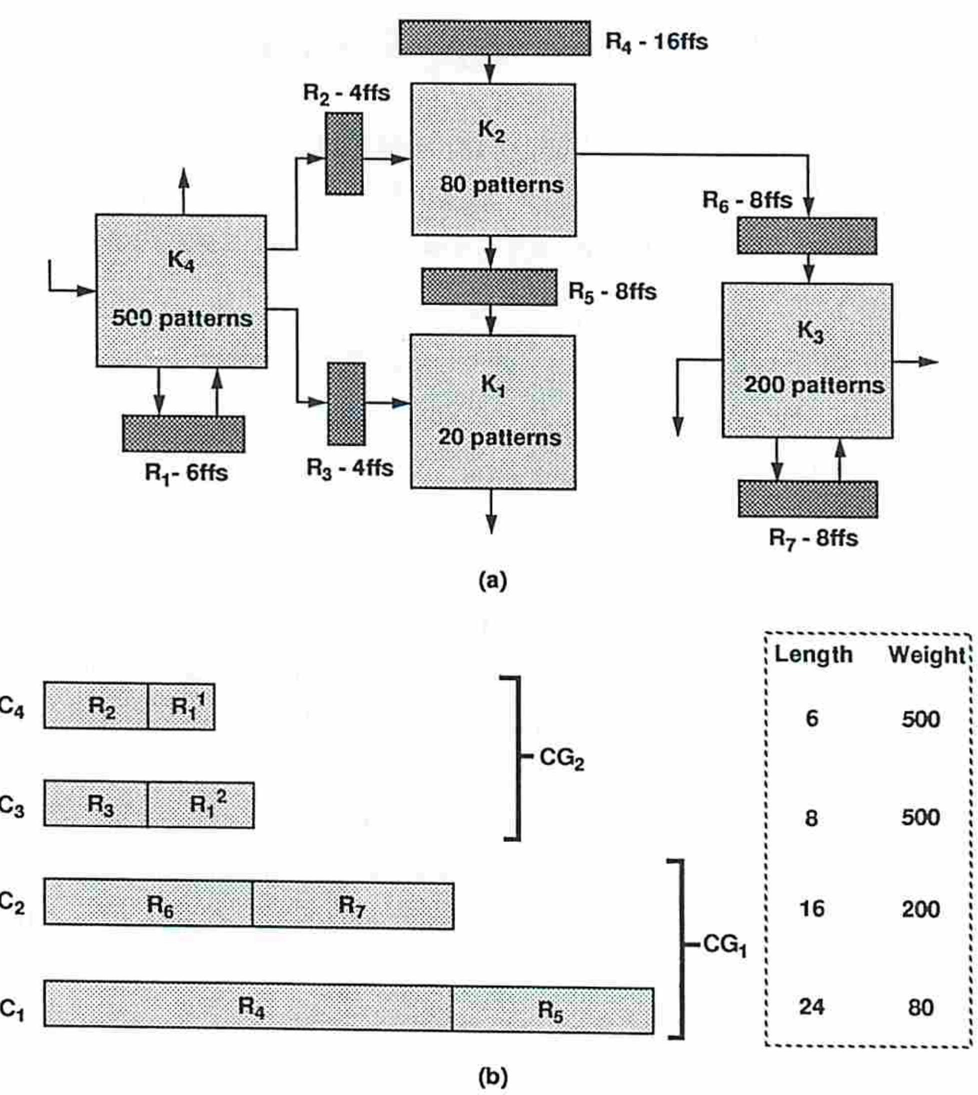


Figure 7.4: (a) Circuit example. (b) Four-chain configuration with two chain groups.

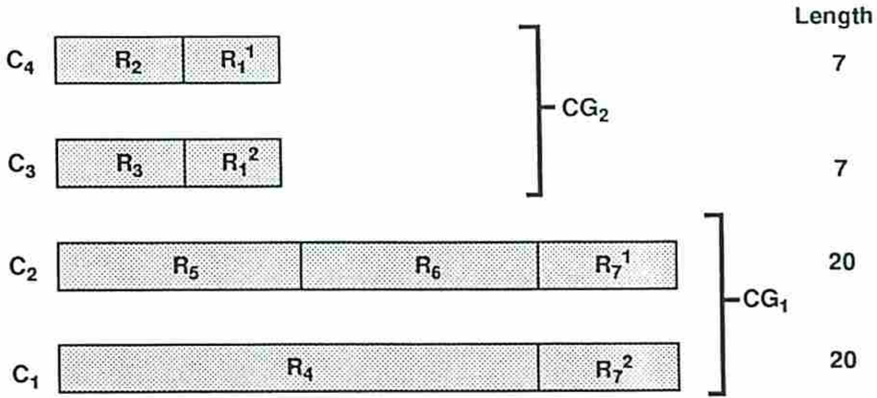


Figure 7.5: Optimal asynchronous four chain configuration.

to K_3 in session TS_1^1 using chain C_2. The chain cycle of the session is equal to 16. Simultaneously 420 patterns are applied to K_4 in session TS_1^2 using chains C_3 and C_4 and the chain cycle is equal to 8. The test time for the asynchronous phase is thus equal to $max(120(16+1), 420(8+1)) = 3780$ clock cycles. The total test time under the two-phase test application scheme is equal to 2000 + 3780 = 5780 clock cycles.

If a single mode signal controlled all four chains, the testing of kernels K_3 and K_4 will be completed in two additional sessions as per the overlapped scheme. The total test time increases to 6740 clock cycles. Hence a savings of 14.2% is obtained in using two mode signals to control the chains. For a configuration with 4 equal length chains and a single mode signal, the total test time would be equal to 500(14 + 1) = 7500 clock cycles. The four chain configuration with 2 mode signals achieves a savings of 25% in the total test time.

Additional savings in test time can be obtained if the chain configuration is modified. Consider the four chain configuration with 2 mode signals shown in Fig. 7.5. With this configuration, the test time under the two-phase scheme is equal to 20(20 + 1) + 60(20 + 1) + max(420(7 + 1), 120(20 + 1)) = 5040 clock cycles, a 33% reduction over four equal length chains. We focus on the problem of generating such configurations that minimize the test time under the two-phase test scheme.

7.4 Model of Problem

As a first step, let us generalize the two-phase overlapped scheme to a circuit of n kernels characterized by the sets $KSEQ$ and $WSEQ$. As in Section 5.3, let N and k denote the number of scan flip-flops and the number of scan chains, respectively. Let m denote the number of test mode signals, which we will assume is equal to 2.

Definition For a given asynchronous chain configuration, the *synchronizing* kernel K_α is the kernel with the largest test length in $KSEQ$ that is tested by at least one chain from both chain groups. The test length of K_α is referred to as the *synchronizing* weight W_α.

Based on the value of W_α, we generate two disjoint ordered subsets of $WSEQ$ denoting the test lengths of kernels solely tested by each chain group in the asynchronous phase, i.e., $W_{CG_1} = \{W_{i_1}, W_{i_2}, \ldots, W_{i_\beta}\}$ and $W_{CG_2} = \{W_{j_1}, W_{j_2}, \ldots, W_{j_\gamma}\}$ where $\alpha < i_1 < i_2 < \ldots < i_\beta \leq n$ and $\alpha < j_1 < j_2 < \ldots < j_\gamma \leq n$. In the four chain configuration of Fig. 7.4(b) the synchronizing kernel is K_2, the synchronizing weight is 80, $W_{CG_1} = \{200\}$ and $W_{CG_2} = \{500\}$. The two-phase overlapped scheme consists of α sessions, $TS_1, \ldots, TS_\alpha$ in the synchronous phase and two concurrent set of test sessions in the asynchronous phase, i.e., β sessions, $TS_1^1, \ldots, TS_\beta^1$ using the chains in CG_1 and γ sessions, $TS_1^2, \ldots, TS_\gamma^2$ using the chains in CG_2. The two-phase overlapped scheme is summarized below.

Two-Phase Overlapped Scheme (Inputs: $KSEQ, WSEQ, W_\alpha, W_{CG_1}, W_{CG_2}$)

<u>*Synchronous Phase*</u>

begin

  Set $W_0 = 0$

  For $p = 1$ to α

    In session TS_p, apply $(W_p - W_{p-1})$ patterns to kernels $K_p, \ldots, K_n$.

end

<u>*Asynchronous Phase*</u>

  Set $W_{i_0} = W_{j_0} = W_\alpha$.

parallel-begin

  For $p = 1$ to β

In session TS_p^1, apply $(W_{i_p} - W_{i_{p-1}})$ patterns to kernels $K_{i_p}, \ldots, K_{i_\beta}$.

For $q = 1$ to γ

In session TS_q^2, apply $(W_{j_q} - W_{j_{q-1}})$ patterns to kernels $K_{j_q}, \ldots, K_{j_\gamma}$.

parallel-end

We adopt the notation of Section 5.3 to characterize the weights of the flip-flops, and the weights and lengths of the chains in the configuration. The parameters of the chains are shown in Fig. 7.4(b). The chain cycle CC_p of a session TS_p in the synchronous phase is equal to the length of the longest chain in both chain groups of weight greater than or equal to W_p. Similarly in the asynchronous phase, the chain cycle of session TS_p^1 (TS_p^2) is equal to the length of the longest chain in chain group CG_1 (CG_2) of weight greater than or equal to W_{i_p} (W_{j_p}). Given the weights and lengths of the chains in each chain group, the chain cycle of each session can be evaluated. The total test time is given by the following expression. Assume $W_0 = 0$ and $W_{i_0} = W_{j_0} = W_\alpha$.

$$
\begin{aligned}
TT = \; & \textstyle\sum_{p=1}^{\alpha}(W_p - W_{p-1})(CC_i + 1) \\
& + max\,(\, \textstyle\sum_{p=1}^{\beta}(W_{i_p} - W_{i_{p-1}})(CC_p^1 + 1),\; \textstyle\sum_{q=1}^{\gamma}(W_{j_q} - W_{j_{q-1}})(CC_q^2 + 1))
\end{aligned}
$$

7.5 Configuring the Scan Chains

The problem of configuring the chains to minimize the test time is more complicated than with the overlapped scheme. The complexity arises in the independent operation of the chains in the asynchronous phase, which requires the *max* function in evaluating the total test time. Let $\mathcal{C}(N, k, m)$ denote a scan chain configuration obtained by assigning N scan flip-flops to k chains with m mode control signals distributed among the chains. To efficiently search the space of all possible configurations, we divide the problem into two subproblems: (1) generate all two-way disjoint partitionings of the N flip-flops, and (2) for each two-way partitioning, optimally configure the flip-flops into k chains. For a given two-way partitioning, all flip-flops within a partition are controlled by the same mode signal. Note that dividing the problem in this way ensures that every possible k-chain configuration is

considered. However explicitly enumerating the search space would result in excessive computational costs. We discuss two ways to reduce the overall complexity of the search process. Let us initially consider the second subproblem.

Let the N scan flip-flops be divided into two disjoint partitions P_1 and P_2 of N_1 and N_2 flip-flops, respectively. Based solely on this partitioning, we can determine the synchronizing weight W_α and the number of patterns to be applied in each session, both in the synchronous and asynchronous phases. Assume that k_1 chains are to be configured with the flip-flops in P_1 and k_2 chains with the flip-flops in P_2, where $k_1 + k_2 = k$.

Within each partition we can restrict the manner in which the flip-flops are to be configured into chains. For any two chains within a chain group, all flip-flops in the shorter chain must be of weight greater than or equal to the weight of flip-flops in the longer chain. This idea is similar to that used in Lemma 3 in Section 5.4. Note however that it can only be applied to chains within a chain group. The following lemma restates Lemma 3 in the context of an asynchronous chain configuration.

Lemma 14 Consider any optimal $\mathcal{C}(N, k, m)$ configuration. Within each chain group, we can redistribute the flip-flops among the chains so that for any two chains C_i and C_j in the group, if $l(C_i) \leq l(C_j)$, every flip-flop in C_i is of weight greater than or equal to the weight of every flip-flop in C_j. The test time does not increase with this transformation.

Proof Since flip-flops are only rearranged within the chain groups, the synchronizing weight, the number of test sessions and the number of patterns applied in each session are unchanged in both phases. The reassignment of flip-flops preserves the lengths of chains in both chain groups. For a test session in either of the two phases, let K_δ denote the kernel with least test length being tested in the session. Since the chain cycle of the session is equal to the longest chain of weight greater than or equal to W_δ, the transformation cannot increase the chain cycle of the session. □

The problem of configuring the chains within a chain group is similar to the problem of generating an optimal configuration for the overlapped scheme. It is this similarity that is exploited in the above lemma to reduce the amount of explicit enumeration. For partition P_1, Lemma 14 allows us to configure the k_1 scan chains

in the following manner. The flip-flops in P_1 are sorted in order of non-increasing weight factors in an array $A[1, \ldots, N_1]$. The shortest chain in the partition can vary in length from 1 to $\lfloor \frac{N_1}{k_1} \rfloor$ and in each case must contain the largest weight flip-flops starting from $A[1]$. For any given length x of this shortest chain, the length of the next shortest chain can vary from x to $\lfloor \frac{N_1 - x}{k_1 - 1} \rfloor$ and contain the largest weight flip-flops starting from $A[x+1]$. Continuing recursively in this fashion, we ensure that the next shortest chain to be configured is as long as the previous chain and does not exceed the length of a chain to be subsequently configured. It is thus sufficient to enumerate a total of $O(N_1^{k_1})$ configurations to satisfy the conditions of Lemma 14. Note that $k_1 \ll N_1$. Similarly $O(N_2^{k_2})$ configurations need to be enumerated for partition P_2.

The above approach to configuring the chains within a chain group is similar to the algorithm outlined at the beginning of Section 5.4. A natural question that arises is whether we can use the dynamic programming approach to further reduce the amount of enumeration. Recall that one of the properties used in the dynamic programming algorithm is the ability to evaluate the test time contribution of one or more chains independent of the rest of the configuration. In an asynchronous chain configuration, the test time contribution of a chain depends on both the synchronous and asynchronous phases. Based on Lemma 14, its contribution in the asynchronous phase can be determined similar to the overlapped scheme. It is however difficult to evaluate its contribution in the synchronous phase with no knowledge of the lengths and weights of chains in the other chain group. In particular Lemma 14 is not applicable across the chain groups. Conversely if the synchronizing weight is equal to 0, the dynamic programming algorithm can be independently applied to each chain group to minimize the overall test time.

For a given two-way partitioning, the number of possible values for k_1 and hence k_2 is equal to $k - 1$. By using Lemma 14 within each partition, the size of the search space is equal to $O(k * N_1^{k_1} * N_2^{k_2}) \approx O(kN^k)$. However this is a worst case assumption. The following two lemmas can be used to further prune the search space in configuring the chains.

Lemma 15 Consider any two chains C_i and C_j in different chain groups. If $W(C_i) \leq W(C_j)$ and $W(C_i) \leq W_\alpha$, it is sufficient to consider configurations in which $l(C_i) \geq l(C_j)$.

Proof In an optimal $\mathcal{C}(N,k,2)$ configuration, assume there exists two chains C_i and C_j in different chain groups such that $W(C_i) \leq W(C_j), W(C_i) \leq W_\alpha$ and $l(C_i) < l(C_j)$. Note that C_i will only be used in the synchronous phase of the test scheme. Move C_i to the same chain group as C_j and redistribute flip-flops among the two chains such that $l(C_i) \geq l(C_j)$. As a result the chain cycle of every session both in the synchronous and asynchronous phases can only decrease or remain the same. □

Lemma 16 Within a chain group, if two chains are of equal weight, they should also be of equal length (± 1).

Proof Since both chains are of equal weight, they will be used in the same set of test sessions. Redistribute flip-flops among the chains such that they are of equal length (± 1). For those test sessions in which both chains are used, the corresponding chain cycles cannot increase. □

Let us now consider the first subproblem of generating all two-way disjoint partitionings of the N scan flip-flops. Note that in the worst case the number of such partitionings is equal to 2^N. However in every case, we can restrict all flip-flops of weight W_n and W_{n-1} to lie in a single partition.

Lemma 17 In an optimal $\mathcal{C}(N,k,2)$ configuration, all flip-flops of weight W_n and W_{n-1} should lie in a single, though not necessarily the same chain group.

Proof If flip-flops of weight W_n are in both chain groups, the synchronizing weight will be equal to W_n. This reduces the two-phase overlapped scheme to a single phase scheme with no asynchronous mode of operation. A similar result is valid for flip-flops of weight W_{n-1}. □

From a routing point of view we would also like to retain all flip-flops belonging to a register within a single chain group. Based on this and Lemma 17, we adopt a heuristic kernel-oriented approach to partitioning the flip-flops into two groups. All flip-flops of the same weight are assigned to a single partition. As a result in the

worst case, a total of 2^n different partitions will be analyzed, where n is equal to the number of kernels. The following pseudo-code outlines the algorithm to generate an "optimal" $\mathcal{C}(N, k, 2)$ configuration for the two-phase overlapped scheme.

Configure-chains (Inputs: $KSEQ, N, k$)
begin

1. Initialize *Best-time* $= \infty$.
2. For every two-way partitioning of the scan flip-flops.
   (a) Evaluate W_α, W_{CG_1} and W_{CG_2}.
   (b) For $k_1 = 1$ to $k - 1$
      i. Sort the N_1 flip-flops in partition P_1.
      ii. Configure k_1 scan chains and check condition in Lemma 16.
      iii. Sort the N_2 flip-flops in partition P_2.
      iv. Configure $(k - k_1)$ chains and check conditions in Lemmas 15 and 16.
      v. For every generated $\mathcal{C}(N, k, 2)$ configuration
         - If Test time $\mathcal{TT}$ < *Best-time*.
         - Update *Best-time* and store current configuration.

end.

7.6 Enhancing the Test Application Scheme

A drawback with the two-phase overlapped scheme is that the existence of two mode signals is not exploited in the synchronous phase. Given a chain configuration generated by our algorithm, we can consider more complex test schemes to further reduce the test time. For our example circuit consider the asynchronous two chain configuration shown in Fig. 7.6 which minimizes test time under the two-phase scheme. The test time is equal to $20(40 + 1) + 60(40 + 1) + max(420(14 + 1), 120(40 + 1)) = 9580$ clock cycles, a 32% reduction over two equal length chains. Note that to test kernels K_1 and K_2, both the chains C_1 and C_2 are required.

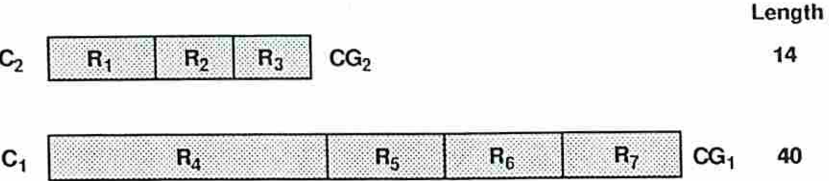


Figure 7.6: Two chain asynchronous configuration.

Rather than associating a single chain cycle with a test session, we characterize a test session by means of two shift cycles, one for each chain group. Similar to the shift cycle of a chain, the shift cycle of a chain group is given by the number of clock cycles the chains are placed in shift mode before toggling the mode signal. Consider the following test session in which the chains are shifted independent of each other while still synchronizing on the normal clock. A shift cycle of 14 is used for C_2 so that a pattern is applied to kernel K_4 every 15 clock cycles. For chain C_1, a shift cycle of 44 clock cycles is used. In the 45th clock cycle, test results are simultaneously latched into both chains. Hence for every pattern applied to kernels K_1, K_2 and K_3 in the session, three patterns are applied to K_4. By considering such test sessions, we can increase the rate of testing for certain kernels while still synchronizing on the normal clock to apply patterns to kernels tested by both chain groups.

We will use the above example to illustrate the enhanced test scheme. For our two chain configuration, the enhanced test scheme can potentially consist of three different sessions, (1) TS_1 in which both chains operate at a shift cycle of 40, (2) TS_2 in which chain C_1 operates at a shift cycle of 44 and chain C_2 at a shift cycle of 14 (as described above), and (3) TS_3 in which chain C_1 operates at a shift cycle of 40 and chain C_2 at a shift cycle of 14 (asynchronous mode). Let X_1, X_2 and X_3 respectively denote the number of clock cycles employed in each of the three sessions. An integer linear program can be formulated to minimize the total test time.

$$\textit{minimize}\ X_1 + X_2 + X_3$$
$$\text{subject to}$$
$$\lfloor \tfrac{X_1}{41} \rfloor + \lfloor \tfrac{X_2}{15} \rfloor + \lfloor \tfrac{X_3}{15} \rfloor \geq 500$$
$$\lfloor \tfrac{X_1}{41} \rfloor + \lfloor \tfrac{X_2}{45} \rfloor + \lfloor \tfrac{X_3}{41} \rfloor \geq 200$$
$$\lfloor \tfrac{X_1}{41} \rfloor + \lfloor \tfrac{X_2}{45} \rfloor \geq 80$$

The objective function to be minimized is the sum of the three decision variables. A constraint inequality is needed to model the testing requirements for each kernel. For example the first inequality denotes the contribution of each test session in testing K_4. In sessions TS_1, TS_2 and TS_3, a pattern is applied to K_4 every 41, 15 and 15 clock cycles, respectively. The sum of the patterns applied in all three sessions must exceed the test length of K_4, i.e., 500. Similarly the second and third inequalities model the testing requirements for K_3 and K_2, respectively. Note that an inequality is not required for K_1 since any combination of the test sessions that apply 80 patterns to K_2 will apply the required 20 patterns to K_1. By removing the floors in the constraint inequalities and solving the integer program, we get a solution of $X_1 = 1394, X_2 = 2070$ and $X_3 = 4920$. Hence in TS_1, 34 patterns are applied to all three kernels; in TS_2, 46 patterns are applied to K_2 and K_3 while simultaneously applying 138 patterns to K_4; and in TS_3, 120 patterns are applied to K_3 and 328 patterns to K_4. The total test time is equal to 8384 clock cycles, a reduction of 12.5% over the test time with the two-phase scheme.

The enhanced test scheme can be generalized to any chain configuration generated for the two-phase scheme. Given a chain configuration we can enumerate all potential test sessions. This is done by considering every pair of chains, one from each chain group. As shown in our example, each pair of chains can give rise to three potential sessions. Consider for example two chains C_i and C_j from chain groups CG_1 and CG_2, respectively. Assume $l(C_i) \leq l(C_j)$. The shift cycles of the three sessions that can be generated by these chains are respectively given by (1) $l(C_i)$ for both chain groups; (2) $l(C_i)$ for CG_1, and $\lceil \frac{l(C_j)}{l(C_i)+1} \rceil (l(C_i) + 1) - 1$ for CG_2; and (3) $l(C_i)$ for CG_1, and $l(C_j)$ for CG_2. In this way we can enumerate the set of test sessions to be considered in the optimization process. Note that multiple instances of sessions that employ the same pair of shift cycles can be pruned from the set. For each session TS_i a variable X_i, denoting the number of clock cycles employed in the session, is introduced in the integer program. For each kernel we determine both the set of test sessions in which it is tested and the rate at which patterns are applied to the kernel in the session. Based on this, a constraint equation is introduced in the integer linear program for each kernel.

In solving the integer program, the values of X_i that are greater than zero will determine the test sessions to be used in the final test scheme. Note that the order in which these sessions are used does not affect the total test time. The computational costs in enhancing the test scheme is determined by the number of variables in the integer program. In the worst case this number is of order $O(k^2)$.

7.7 Experimental Results

As with our previous methodologies, the savings in test time with asynchronous chains depend on the test lengths of the kernels, and the number of scan flip-flops involved in testing each kernel. An additional factor that influences the test time is the degree of sharing of the flip-flops between kernels. For example in the circuit of Fig. 7.1(b), there exists no sharing of flip-flops between the two kernels. A reduced degree of sharing provides greater opportunity to run the scan chains in the asynchronous mode. To illustrate this we employ the example circuit used in Section 5.6 and consider the same three variations of the circuit as shown in Table 5.2.

In Tables 7.1 and 7.2 we compare the test times of different multiple chain methodologies as the number of chains vary from 2 to 10. The methodologies shown in the table include: TT (eq. lgth) - equal length chains with a single mode signal; TT (1 mode) - optimal chains with a single mode signal; TT (2-ph) & (enh.) - asynchronous chains with 2 mode signals under the two-phase scheme and enhanced test scheme, respectively. The chain configuration for both asynchronous schemes is generated by the algorithm in Section 7.5. For each methodology we show both the test time in clock cycles and the percentage savings with respect to the test time for equal length chains. We summarize a few observations from the results in the two tables.

• The proportion of scan flip-flops associated with kernels of test length 200 and 500 increases from 22% in Case 1 to 67% in Case 3. Increasing the number of frequently accessed flip-flops reduces the ability to confine them to shorter chains. As a result the average savings with optimal chains and a single mode signal reduces from 36% for Case 1 to 5% for Case 3. A similar trend is also observed with the "asynchronous"

No. of chains	Test time (eq. lgth.)	Case 1			Case 2		
		TT (1 mode)	TT (2-ph)	TT (enh.)	TT (1 mode)	TT (2-ph)	TT (enh.)
1	49598	-	-	-	-	-	-
2	25049	15896 36.54%	15820 36.84%	11500 54.09%	23168 7.51%	23100 7.78%	22500 10.17%
3	17033	11096 34.86%	8500 50.10%	8446 50.41%	14114 17.14%	13420 21.21%	11500 32.48%
4	13025	8018 38.44%	7070 45.72%	6144 52.83%	11093 14.83%	10320 20.77%	9200 29.37%
5	10520	6518 38.04%	5320 49.43%	5320 49.43%	8987 14.57%	8320 20.91%	7000 33.46%
6	9017	5506 38.94%	4420 50.98%	4420 50.98%	7307 18.96%	6960 22.81%	6000 33.46%
7	7514	4786 36.31%	4020 46.50%	4009 46.64%	6467 13.93%	6080 19.08%	5000 33.46%
8	7013	4259 39.27%	3580 48.95%	3574 49.04%	5738 18.18%	5420 22.71%	4900 30.13%
9	6011	3839 36.13%	3200 46.76%	3200 46.76%	5138 14.52%	4900 18.48%	3740 37.78%
10	5510	3539 35.77%	2880 47.73%	2873 47.86%	4598 16.55%	4360 20.87%	3700 32.85%

Table 7.1: Comparing multiple scan chain methodologies - Cases 1 and 2

methodologies. However the capability to exploit temporal parallelism by running the chains independently provides additional savings in test time.

• In Cases 1 and 3, the two kernels of test length 200 and 500 do not share any flip-flops. The "asynchronous" methodologies exploit this by assigning each kernel's flip-flops to separate chain groups and testing the two kernels independently. In Case 2, since flip-flops are shared by these two kernels, both kernels are tested by the same chain group. Hence the two-phase scheme provides little improvement over the single mode case. The enhanced scheme however achieves greater reductions in test time by testing these kernels at a faster rate even when synchronizing between the chain groups.

No. of chains	Test time (eq. lgth.)	Case 3		
		TT (1 mode)	TT (2-ph)	TT (enh.)
1	49598	-	-	-
2	25049	23762 5.14%	20060 19.92%	18500 26.14%
3	17033	16934 0.58%	12600 26.03%	12600 26.03%
4	13025	12092 7.16%	10280 21.07%	9500 27.06%
5	10520	10052 4.45%	7640 27.38%	6736 35.97%
6	9017	8282 8.15%	6380 29.24%	6367 29.39%
7	7514	7196 4.23%	5700 24.14%	5484 27.02%
8	7013	6296 10.22%	5260 25.00%	5100 27.27%
9	6011	5693 5.29%	4400 26.80%	4400 26.80%
10	5510	5234 5.01%	4060 26.32%	3980 27.77%

Table 7.2: Comparing multiple scan chain methodologies - Case 3

• The results illustrate the tradeoffs available with respect to test time and pin count. For example in Case 3, the test time with 7 asynchronous chains (16 pins) is less than the test times with 9 optimal synchronous chains (19 pins) and 10 equal length chains (21 pins).

In Table 7.3 we compare the test times with the four multiple chain methodologies for a full scan design of the *USC Data Path-I*. The format of the table is similar to that used in Table 7.2. The design corresponds to the modified version of the circuit in Fig. 4.6 in which the ALU is replaced with a more complex module requiring 500 test patterns.

The test application schemes with asynchronous chains are more complex as compared to the test schemes for optimal/equal length chains with a single mode signal. This might impact the test control hardware/software requirements. With regard to the routing overheads in configuring the chains, similar approaches as discussed in Section 5.7 can be used to tradeoff the test time with routing area. Note that the use of a flush policy only constrains the assignment of flip-flops to chains. The heuristic approach to partitioning the flip-flops among the chain groups ensures that all flip-flops of a register are controlled by the same mode signal. Within a chain group, we can similarly ensure that a register wholly lies in a chain by redistributing flip-flops among the chains.

7.8 Increasing the Number of Mode Signals

Throughout the chapter we have assumed the use of two mode signals to control the scan chains. A natural extension to the methodology is to consider three or more mode signals. The two-phase overlapped scheme as well as the techniques to configure the chains can be suitably modified to account for more than two mode signals. However the advantages in using more mode signals to reduce the test time must be weighed against the increased test control complexity. More importantly, the marginal returns in terms of test time savings tend to diminish with increasing number of mode signals. To illustrate this, let us consider the circuit shown in Fig. 7.1(b). The advantages of asynchronous chains can be fully exploited in a circuit when (a) there is a large disparity in both the test lengths of the kernels and

No. of chains	Test time (eq. lgth.)	TT (1 mode)	TT (2-ph)	TT (enh.)
1	52604	-	-	-
2	26552	23804 10.35%	20932 21.17%	20500 22.79%
3	18035	16504 8.49%	11292 37.39%	10500 41.78%
4	13526	11368 15.95%	8775 35.12%	8775 35.12%
5	11021	8944 18.85%	7695 30.18%	7695 30.18%
6	9518	7507 21.13%	5896 38.05%	5500 42.21%
7	8015	6314 21.22%	4932 38.47%	4509 43.74%
8	7013	5584 20.38%	4455 36.48%	4455 36.48%
9	6512	5102 21.65%	4058 37.68%	3972 39.00%
10	6011	4713 21.59%	3663 39.06%	3663 39.06%

Table 7.3: Results for a full scan design of the *USC Data Path-I*

the number of flip-flops involved in testing them, and (b) the number of flip-flops shared between the kernels is minimal. In this respect, the circuit of Fig. 7.1(b) represents a "best" case scenario to illustrate the use of asynchronous chains.

For this circuit, let us initially evaluate the percentage savings in test time in using two asynchronous chains with two mode signals over two optimal chains with a single mode signal. Based on the calculations in Section 7.2.1, this is given by the following expression.

$$\%\ Savings = \begin{cases} \frac{\alpha-1}{\alpha+1} \times 100 & \text{if } \alpha \leq 2 \\ \frac{\alpha-1}{2\alpha-1} \times 100 & \text{if } \alpha > 2 \end{cases}$$

For $\alpha = 2$, the test time savings is equal to 33%. However as $\alpha \rightarrow \infty$, the test time savings asymptotically converge to 50%. For the circuit of Fig. 7.1(b), the use of three mode signals can provide no additional savings in test time over two mode signals. This is due to the fact that there are only two kernels in the circuit. Consider the addition of a third kernel to the circuit similar to the first two kernels. Let the test length of the kernel be equal to W/β and let the number of flip-flops involved in testing the kernel be equal to βL. In addition, let us assume $\alpha \geq 2$, and $\beta \geq 2\alpha$. This is to avoid considering the two cases when configuring optimal chains with a single mode signal (see Section 7.2.1). Let us compare the optimal test times with three scan chains for one, two and three mode signals, respectively. In all three cases, an optimal configuration will consist of each kernel's flip-flops being assigned to separate chains. With two mode signals, the chain containing flip-flops of the kernel with test length W will have a separate mode signal. The remaining two chains are controlled by a single mode signal. The percentage savings in test time in using two mode signals over a single mode signal is given by

$$\mathcal{PS}_1 = \frac{1 - \frac{1}{\alpha}}{3 - \frac{\alpha}{\beta} - \frac{1}{\alpha}} \times 100.$$

Similarly the percentage savings in using three mode signals over a single mode signal is given by

$$\mathcal{PS}_2 = \frac{2 - \frac{\alpha}{\beta} - \frac{1}{\alpha}}{3 - \frac{\alpha}{\beta} - \frac{1}{\alpha}} \times 100.$$

Let us assume $\beta = 2\alpha$. As $\alpha \to \infty$, $\mathcal{PS}_1 \to 40\%$ and $\mathcal{PS}_2 \to 60\%$. The use of a third mode signal can only provide in the best case an additional savings of 20% over two mode signals. Note however that these savings are dependent on the value of β. If $\beta = 3\alpha$, the corresponding asymptotic values for $\mathcal{PS}_1$ and $\mathcal{PS}_2$ will be 37.5% and 62.5%, respectively. With three mode signals, to achieve savings twice as large as with two mode signals, β should tend to α^2. In other words the number of patterns and number of flip-flops should scale geometrically to achieve proportionate savings with three or more mode signals.

The savings in test time by using asynchronous chains greatly depend on the circuit topology. The use of two mode signals can be exploited if a given "realistic" circuit approximately conforms to the topology of Fig. 7.1(b). This translates to being able to divide the kernels in the circuit into two groups such that (1) the test lengths and number of flip-flops associated with each group scale linearly and in inverse proportion, and (2) there is minimal sharing of flip-flops between the groups. The ability to partially satisfy these constraints led to the savings in test time for the case study in Section 7.7. However to fully benefit from the use of three or more mode signals, greater restrictions are imposed on the circuit topology. For example with three mode signals, there is low probability that a circuit will exhibit a quadratic scaling factor as well minimal mutual sharing between three groups of kernels. Rather than increasing the number of mode signals, more savings could potentially be obtained by increasing the number of scan chains.

7.9 Summary

The main contribution in this chapter has been to introduce the idea of asynchronous multiple scan chains. We have presented efficient test application schemes and optimization techniques to configure the scan chains. The large savings in both test application time and pin count highlight the effectiveness of the approach. Asynchronous chains will provide maximum benefits in designs consisting of multiple kernels with minimal sharing of scan flip-flops between kernels. A designer however should consider the test control and routing overheads prior to incorporating the methodology in a design.

Chapter 8

Test Scheduling for Scan Designs

8.1 Introduction

Throughout our discussion up to this point, we have modeled a scan design as a set of disjoint kernels and associated with each kernel a set of scan registers and primary I/O. This model was sufficient to capture the basic elements in a partitioned scan design. However more refined techniques can be used to further divide each of the disjoint kernels. Current trends in top-down hierarchical design make use of well characterized standard modules from a design library. Knowledge of these standard modules as well as high-level structural and functional information can be exploited to partition the circuit into smaller kernels. Since test generation is done on smaller pieces of logic, the costs of test generation are reduced. More importantly, as we will show in this chapter, the test application costs can also be reduced with a combined approach to scan chaining and test scheduling.

To illustrate the effects of more refined partitioning, consider the scan design shown in Fig. 8.1(a) which includes seven scan registers configured into a single chain. Every scan register contains 4 flip-flops, except for R_1 and R_7 which each contain 8 flip-flops. In a conventional approach such as LSSD [2], the scan registers are replaced by pseudo-inputs/outputs and the circuit is processed as a single block of logic by an ATPG system. Let us consider the internal structure of the design as shown in Fig. 8.1(b). The design consists of four combinational modules K_1, K_2, K_3, and K_4 and a multiplexer. At a "coarse" level of partitioning, maximally connected

combinational logic can be clustered to generate two disjoint kernels. The modules K_1, K_2, K_3, and the multiplexer would form a single kernel, and K_4 is treated as a separate kernel. However a more refined view of the circuit can be generated by subdividing these disjoint kernels. By isolating the multiplexer as a separate entity and exploiting its function in transporting data unchanged from input to output, K_3 can be separated from the rest of the logic. Similarly identifying the fanout from register R_6 will permit K_1 and K_2 to be treated as individual kernels. A number of different approaches to partition a circuit based on structural and functional information can be found in [35, 40, 41].

For the example of Fig. 8.1(b), we can treat the scan design as being made up of four independent kernels. We associate a set of test patterns individually with each kernel. These patterns could be obtained either by running test generation individually on the kernel, or as part of the module description from a design library. However, unlike our earlier circuit model, every subset of kernels cannot be concurrently tested. The output of scan register R_6 fans out to the kernels K_1 and K_2. The *resource conflict* embodied in the use of register R_6 precludes the *simultaneous* application of tests to both kernels. Similarly consider the kernels K_2 and K_3 feeding the scan register R_7 via the multiplexer. Since both K_2 and K_3 require R_7 to latch results, they cannot be tested simultaneously. The presence of resource conflicts will require an extension to our circuit model. The details in modeling these resource conflicts will be explained shortly.

A more significant effect in extending the circuit model is the increased complexity of minimizing the test application time. The presence of resource conflicts makes it necessary to consider test scheduling in the optimization process. Testing each kernel individually will lead to large test application times. We need to exploit potential parallelism in testing kernels concurrently to reduce the overall test time. The test scheduling problem cast in this fashion has been studied in a number of previous works, mainly applied to BIST designs. However in scheduling tests for scan designs, an additional degree of freedom is provided by the scan chain organization. In a scan design, kernels are tested by serially shifting in patterns and shifting out results. As a result the time to test a kernel is a function of both the organization

of the scan chain and the shift policy. This adds a completely new perspective to the test scheduling problem.

In this chapter we develop a range of scheduling techniques to minimize the test time for partitioned scan designs with resource conflicts. A single scan chain is employed to configure the scan registers. The techniques take into account (a) parallelism in testing kernels concurrently, (b) the shift policy, and (c) the organization of the scan registers. Consequently different tradeoffs are provided in the design costs of test application time, routing overhead and control complexity. When employing the minimum shift policy in applying tests, we show that there exists a close interaction between the order of registers in the chain and the subsequent test scheduling phase. Based on this observation we develop an iterative approach between the two phases to minimize the overall test time.

8.2 A Generalized Circuit Model

As in our earlier case, we associate the sets $KSEQ$ and $WSEQ$ with a partitioned scan design. Resource conflicts between two kernels preclude the simultaneous application of tests to both kernels. As mentioned earlier, these resource conflicts can manifest in the simultaneous use of scan registers or primary inputs/outputs, either in applying patterns or receiving results. These resource conflicts are modeled in the form of a *test compatibility graph* (TCG). Each node of the TCG represents a kernel, and edges represent compatibility relationships between the kernels. In other words, if an edge exists between two nodes, there are no resource conflicts between the corresponding kernels and the tests for the two kernels can be applied concurrently. For the design in Fig. 8.1(b), the TCG is shown in Fig. 8.2(a).

Extending the circuit model also relaxes the constraints on the number of kernels tested by a scan register. A scan register can be involved in applying patterns to, and/or receiving results from, more than one kernel. Two additional parameters are used to characterize each scan register R_j ($1 \leq j \leq M$).

1. K_j^D - subset of kernels to which patterns are applied by R_j.

2. K_j^R - subset of kernels from which results are latched by R_j.

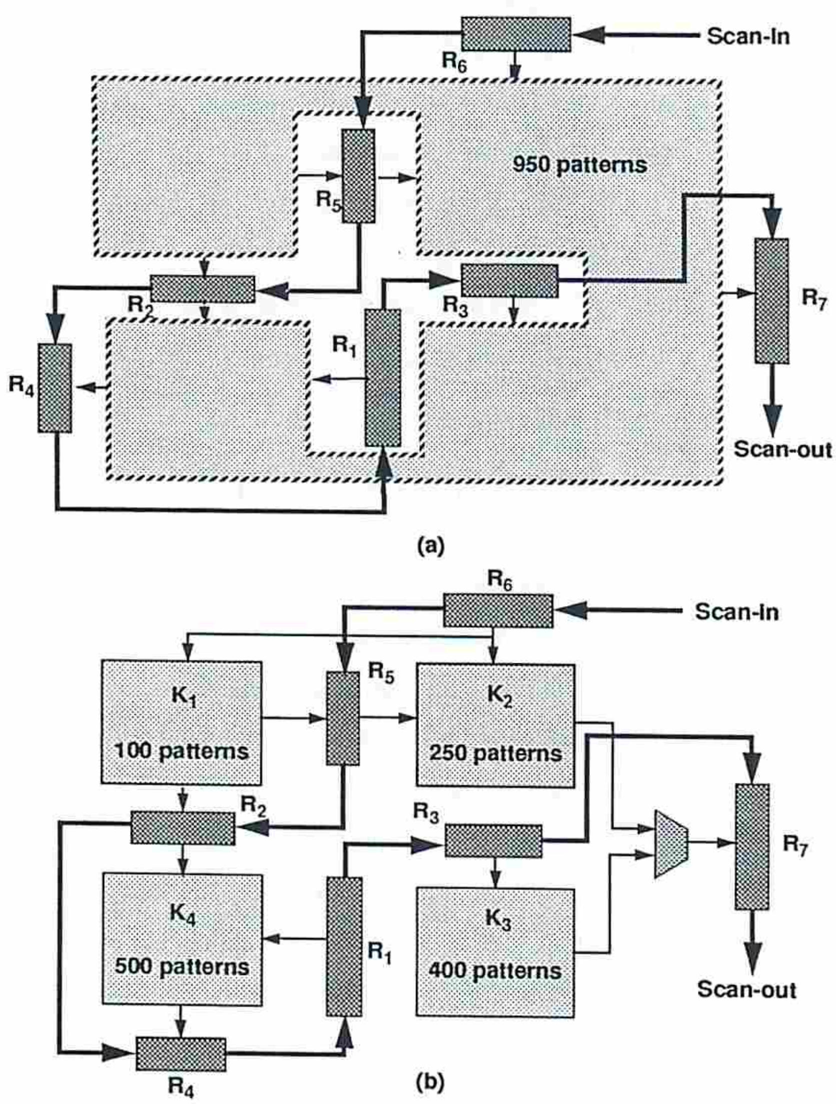
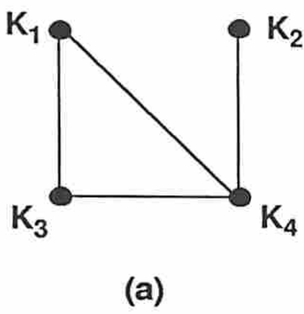


Figure 8.1: (a) Example scan design. (b) Internal structure of design.



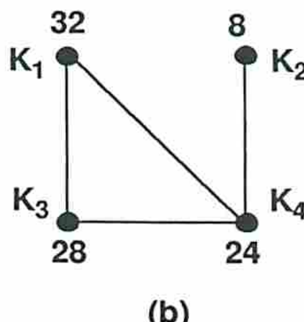


Figure 8.2: (a) TCG of example scan design. (b) Modified TCG for minimum shift policy.

The corresponding parameters for register R_7 in Fig 8.1(b) are given by $K_7^D = \emptyset$ and $K_7^R = \{K_2, K_3\}$. The driver (receiver) weight d_j (r_j) of a register R_j is given by the sum of the test lengths of kernels in K_j^D (K_j^R).

Definition A **complete** test session TS_i is characterized by means of three parameters: (1) the subset of kernels tested in the session, (2) the chain cycle CC_i of the session, and (3) the number of patterns N_i applied in the session.

Recall that a test session as used in the overlapped scheme specified the set of kernels tested in the session and the number of patterns applied to these kernels. To complete the definition of the session, the scan chain configuration was used to determine the chain cycle of the session. We use the term *test session* to refer to a complete session that has one of its three parameters not fully specified. In the scan chaining phase this parameter is the chain cycle of the session. As we will see shortly, the parameter to be specified in the test scheduling phase is the number of patterns. In either case, the goal of the respective phase is to complete the specification of each test session to generate a *test schedule* that minimizes the associated test time.

Definition A **test schedule** TS consists of a set $\{TS_1, TS_2, \ldots, TS_\gamma\}$ of complete test sessions.

Before considering the test scheduling techniques, we briefly discuss the differences in scheduling tests for scan designs as compared to BIST designs.

8.3 Comparing BIST and scan scheduling

The need to shift both patterns and results in testing scan designs gives rise to two important differences as compared to BIST designs. These differences highlight both the additional flexibility available in scheduling tests for scan designs and the importance of the scan chain organization.

1) *Computation of test time* - In BIST designs the test time for a kernel is mainly determined by the number of patterns to be applied, and the number of clock cycles to transport these patterns from the test pattern generator to the kernel and from the kernel to a signature analyzer. In contrast the test application time in scan designs is dominated by the time spent in shifting test data. Consider kernel K_2 in our example design. With the flush policy, the total time to test the kernel is equal to $250(36 + 1) = 9250$ clock cycles. If two or more kernels are being concurrently tested in a session, their test patterns are merged and shifted simultaneously into the scan chain. Since the scan chain is shifted for its entire length in every session, *the time to test a kernel is independent of other kernels being tested concurrently.* This observation is implicitly assumed in the scheduling approaches developed in [14, 15, 33] for BIST designs. Under the *flush policy*, the test scheduling problem is thus similar to that encountered in a BIST design.

The difference arises in using the minimum shift policy. With this policy the total time to individually test K_2 is given by $250(8 + 1) = 2250$ clock cycles. Under the minimum shift policy, the time to test a kernel depends on the other kernels being tested concurrently with it, and the order of scan registers in the chain. For example, if kernel K_4 is concurrently tested with K_2, the merged pattern will need to be shifted for 24 clock cycles before being applied to the kernels. As a result the time to test K_2 will increase to $250(24 + 1) = 6250$ clock cycles. Modifying the order of registers in the chain will similarly affect the time to test K_2 as shown in Chapter 4.

2) *Contiguity in kernel testing* - In [14], three test scheduling disciplines are introduced. These disciplines deal with different constraints in partitioning the tests of a kernel across multiple test sessions. For example with the *nonpartitioned* discipline,

the testing of all kernels in a session must be completed before another subset of kernels is tested. On the other hand, *partitioned* testing permits the testing of a kernel to be arbitrarily interrupted and completed over multiple sessions. In BIST designs, the use of partitioned testing incurs the overhead of time spent in saving and restoring the partial states of the test pattern generator and signature analyzer. This in turn might affect the complexity of the test control hardware/software. However in scan designs, since each pattern is shifted in through the chain, the tests for a kernel can be interrupted any number of times with minimal impact on the test control complexity. We therefore restrict our attention to partitioned testing in scheduling tests for scan designs.

To illustrate the effect of the above two differences, Table 8.1 compares the test times of five optimal test schedules for the design in Fig. 8.1(b). The details in deriving these schedules and the associated test times will be explained shortly. The test schedules are classified based on the shift policy and the scheduling discipline. For example with a flush policy and partitioned testing, an optimal schedule consists of two test sessions. In the first session, 400 patterns are concurrently applied to kernels K_1, K_3 and K_4. At the end of the session, K_1 and K_3 are fully tested and 100 more patterns need to be applied to K_4. In the second session TS_2, 250 patterns are applied to K_2 and K_4 until both are fully tested. The total test time is equal to $400(36 + 1) + 250(36 + 1) = 24050$ clock cycles. As can be seen, there exists a significant difference in the test times for the five schedules. In particular note the tremendous savings achieved with a minimum shift policy. The first four schedules assume the ordering of scan registers as shown in Fig 8.1(a). The fifth schedule represents an optimal schedule for a different order of the registers in the chain. Note the difference both in the test schedule and the total test time as compared to the fourth schedule. The last column in the table represents the percentage savings over the time required to test the design in Fig. 8.1(a), i.e., $950(36 + 1) = 35150$ clock cycles. The derivation of the test length of this unpartitioned design is discussed in the section on experimental results.

	Schedule disc.	Shift policy	No. of sessions	Kernels tested	No. of patts.	Test Time (clk. cycs)	Perc. savings
1	Non partition	Flush	2	K_1, K_3, K_4 K_2	500 250	27750	21.0
2	Non partition	Minimum shift	2	K_1, K_3, K_4 K_2	500 250	18750	46.6
3	Partition	Flush	2	K_1, K_3, K_4 K_2, K_4	400 250	24050	31.5
4	Partition	Minimum shift	4	K_1, K_3, K_4 K_3, K_4 K_2, K_4 K_2	100 300 100 150	15850	54.9
5	Partition	Minimum shift	4	K_1, K_3, K_4 K_2, K_4 K_3, K_4 K_3	100 250 150 150	12650	64.0

Table 8.1: Test schedules classified by scheduling discipline and shift policy

8.4 A Framework for Test Scheduling

In this section we outline a general framework for the test scheduling problem and formulate an integer linear program to deal with the different scheduling techniques. As mentioned earlier, a partitioned discipline is used in the test application process. We formally state the optimization problem as applied to scan designs.

Optimization Problem - Given a set of kernels $KSEQ$, their test lengths $WSEQ$ and the associated TCG, determine an appropriate ordering of scan registers in the chain and a test schedule for applying the required tests to the kernels so as to minimize the total test application time.

A two-phase approach dealing with both scan chaining and the scheduling of tests represents the most general case of the optimization problem. As we will show, such an approach needs to be adopted only when employing the minimum shift policy. As a first step we therefore assume that the scan registers have already been configured into a single chain. In testing the kernels in a design, two cases are considered based on the degree of parallelism in the test process; (1) *sequential* testing in which every

kernel is tested independently in a separate test session, and (2) *parallel* testing in which no restriction is placed on the number of kernels concurrently tested in a session. Sequential testing permits us to distinguish the test time savings obtained in testing kernels concurrently, from the savings in reducing shifting time via the chain organization and shift policy.

Recall that a test session in the test scheduling phase only specifies the subset of kernels tested and the chain cycle used in the session. In parallel testing every complete subgraph of the TCG can be mapped to a test session. The nodes of the subgraph correspond to the kernels tested in the session, and the chain cycle depends on the shift policy that is employed. In particular it is useful to define the concept of a *clique* or a maximal complete subgraph of the TCG. We represent a clique by the set of kernels corresponding to the nodes in the clique. The cliques of the TCG will be used in determining optimal test schedules.

For a given degree of parallelism and a given shift policy, the goal of the test scheduling phase is to determine an optimal test schedule. In other words the test scheduling algorithm determines the set of test sessions in the optimal test schedule and specifies the number of patterns to be applied in each session. Since every complete subgraph of the TCG can generate a test session, it is necessary to prune the search space.

Definition A test session is said to be **feasible** if it can be an element of an optimal test schedule. It is sufficient to only consider feasible test sessions in determining an optimal test schedule.

The definition of a feasible test session depends on the shift policy and degree of parallelism employed in the test process. For example a feasible test session with the minimum shift policy may not be feasible with the flush policy. Details of the feasible test sessions for each of the different cases are discussed in the next section.

The problem of determining an optimal test schedule is formulated as an integer linear program. The inputs to the program are the feasible test sessions, and the test lengths of the kernels in the design. Let the set of feasible test sessions be denoted by $\{TS_{i_1}, TS_{i_2}, \ldots, TS_{i_p}\}$. For each feasible test session, the chain cycle is determined by the shift policy, the scan registers used in the session, and the scan chain configuration. The variables are the number of patterns N_{i_α} $(1 \leq \alpha \leq p)$ to

be applied in each session. For each kernel $K_j (1 \leq j \leq n)$ and each feasible test session TS_{i_α}, the function $F_{i_\alpha}(K_j)$ is defined in the following manner.

$$F_{i_\alpha}(K_j) = \begin{cases} N_{i_\alpha} & \text{if } K_j \text{ is tested in session } TS_{i_\alpha} \\ 0 & \text{otherwise} \end{cases}$$

The following integer program is a generalized formulation of the test scheduling problem.

$$\textit{minimize} \quad \sum_{\alpha=1}^{p} N_{i_\alpha}(CC_{i_\alpha} + 1)$$

subject to

$$\sum_{\alpha=1}^{k} F_{i_\alpha}(K_j) \geq W_j \quad \text{for kernel } K_j \ (1 \leq j \leq n)$$

$$N_{i_\alpha} \geq 0 \quad \text{for session } TS_{i_\alpha} \ (1 \leq \alpha \leq p)$$

The objective function represents the total time required to shift in and apply N_{i_α} patterns in each of the sessions TS_{i_α} $(1 \leq \alpha \leq p)$. A constraint equation is introduced for each kernel K_j $(1 \leq j \leq n)$. The sum of the patterns applied in all sessions in which the kernel is tested should at least be equal to the test length of the kernel. By solving the integer program, an optimal schedule is derived from those feasible sessions in which the number of patterns applied is greater than 0. Note that the order of sessions in the schedule does not affect the test application time. In the next section we individually analyze each of the test scheduling techniques. For each technique the set of feasible sessions to be used in the above formulation are identified.

8.5 Classification of Scheduling Techniques

We classify the test scheduling techniques based on two criteria; (1) the shift policy, and (2) the degree of parallelism. In addition we consider the case when the test

lengths of the kernels can either be of *equal* or *unequal* lengths. For the case of equal length kernels, let N denote the test length of each kernel. The equal length case simplifies the solution of the integer linear program. This simplification is a result of the following lemma.

Lemma 18 To derive an optimal schedule for a design with equal length kernels, it is sufficient to consider test sessions in which N patterns are applied in each session. In other words, every kernel is fully tested in a single test session.

Proof The lemma is trivial in the case of sequential testing. With parallel testing, let us consider the flush policy. A similar proof can be derived for the minimum shift policy. Consider any optimal test schedule for a given design. Remove all sessions from the schedule in which N patterns are applied in the session. Let $\{TS_1, TS_2, \ldots, TS_\beta\}$ denote the set of remaining sessions, in each of which less than N patterns are applied. Consider the subgraph of the TCG induced by the nodes (kernels) tested in these sessions. Let σ denote the cardinality of a minimum size clique cover of this subgraph, i.e., there exists σ disjoint subsets of kernels, no two of which can be tested concurrently. A lower bound on testing all kernels in the subgraph is thus equal to σN. Hence $N_1 + N_2 + \ldots + N_\beta = \sigma N$. We replace the β test sessions with σ test sessions, each corresponding to a clique in the minimum size cover. As a result the total test application time cannot increase. □

For each scheduling technique, the example design of Fig. 8.1(b) is used to illustrate the derivation of the optimal test schedule.

a) *Flush policy with sequential testing*

Both the set of feasible sessions and an optimal schedule consists of n test sessions $\{TS_1, TS_2, \ldots, TS_n\}$. In session TS_i $(1 \leq i \leq n)$, W_i patterns are applied to kernel K_i with a chain cycle equal to the length of the scan chain. For our example design, the total test time with this scheduling technique is equal to $(100 + 250 + 400 + 500)(36 + 1) = 46250$ clock cycles.

b) *Flush policy with parallel testing*

For the case when the flush policy is used with parallel testing, we make use of the following result.

Lemma 19 For both equal and unequal length kernels, the feasible test sessions correspond to the cliques of the TCG.

Proof Consider an optimal test schedule and assume there exists a test session TS_i that does not correspond to a clique of the TCG. The kernels tested in TS_i must form a complete subgraph in the TCG. From the definition of a clique, we can expand this subgraph by adding nodes and edges until it corresponds to a clique in the TCG. With the flush policy, since the entire scan chain is shifted for each pattern, the time to test a subset of kernels is independent of other kernels tested concurrently. Hence the session TS_i can be replaced with a test session corresponding to the clique with N_i patterns applied to the kernels in the session. □

For our example design, the TCG consists of two cliques $\{K_1, K_3, K_4\}$ and $\{K_2, K_4\}$. Since both cliques are *essential*, i.e., include at least one kernel that is not an element of any other clique, an optimal test schedule will also consist of two test sessions. In the first session, 400 patterns are applied to kernels K_1, K_3 and K_4 at the end of which K_1 and K_3 are fully tested. In the second session, 250 patterns are applied to kernels K_2 and K_4. The total test time is equal to 400(36 + 1) + 250(36 + 1) = 24050 clock cycles, a 48% reduction over the sequential case. These savings arise solely as a result of exploiting parallelism in the test application process.

With the flush policy, the objective function in the integer linear program can be simplified since the chain cycle is constant over all sessions. Hence the objective function is given by $\sum_{\alpha=1}^{p} N_{i_\alpha}$, where $\{TS_{i_1}, TS_{i_2}, \ldots, TS_{i_p}\}$ represent the feasible test sessions. With equal length kernels, Lemma 18 permits us to further restrict each N_{i_α} to be a $0-1$ variable. The solution of the integer linear program will generate a test schedule in which a kernel might be tested in multiple sessions. In particular a kernel that is fully tested in a session might appear in the set of kernels tested in a subsequent test session. Even within a session, once a kernel is fully tested, we would like to ensure it is not tested by the remaining patterns applied in the session. Although overtesting a kernel in this way can enhance the fault coverage, we reorganize the schedule to delete such multiple occurrences. This introduces more x's in the test patterns providing greater flexibility to a test compaction technique. More importantly the reorganized schedule identifies the kernels that *need* to be

Test session	Kernels tested	Number of patterns	Chain cycle	
			Flush policy	Minimum shift policy
TS_1	K_1, K_3, K_4	100	36	32
TS_2	K_3, K_4	300	36	28
TS_3	K_2, K_4	100	36	24
TS_4	K_2	150	36	8

Table 8.2: Optimal schedule with parallel testing for both shift policies

tested in each session. This information is necessary to configure the scan chain with the minimum shift policy.

Definition Given an ordering of complete test sessions in a schedule, the test schedule can be modified by (1) subdividing every session in which one or more kernels are fully tested prior to the end of the session, and (2) removing every instance of a kernel fully tested in a session from every subsequent test session. The resulting set of complete test sessions is referred to as a **refined** test schedule.

For the optimal schedule previously determined, the refined test schedule is as shown in Table 8.2. The refined test schedule is sensitive to the order of the test sessions in the schedule. In our optimal schedule let us assume the order of sessions consists of $\{K_2, K_4\}$ followed by $\{K_1, K_3, K_4\}$. The resulting refined schedule is shown in Table 8.3. The process of refining a schedule however does not affect the total test time. Use of the flush policy in the scheduling phase provides for increased flexibility in ordering the registers in the chain, and a simple test control strategy.

c) *Minimum shift policy with sequential testing*

Under a minimum shift policy, the chain cycle of a session in which patterns are solely applied to kernel K_i is denoted by $CC(K_i)$. We refer to $CC(K_i)$ as the chain cycle of kernel K_i. For example the chain cycle of K_1 is equal to 32. The chain cycles of each kernel are evaluated and attached to the corresponding node in the TCG. For the design in Fig 8.1(b), the modified TCG is shown in Fig. 8.2(b).

As with sequential testing under the flush policy, an optimal test schedule with this technique consists of n sessions $\{TS_1, TS_2, \ldots, TS_n\}$. In session TS_i, W_i test patterns are applied to kernel K_i with a chain cycle of $CC(K_i)$. The total test time

for the example design is equal to 100(32 + 1) + 250(8 + 1) + 400(28 + 1) + 500(24 + 1) = 29650 clock cycles.

d) *Minimum shift policy with parallel testing*

Consider a test session in which kernels K_1 and K_4 are concurrently tested. The chain cycle required to test both kernels is given by $max(CC(K_1), CC(K_4))$, i.e., 32 clock cycles. In general the chain cycle of a session in which two or more kernels are concurrently tested is given by the maximum of the individual chain cycles of the kernels.

To determine the feasible test sessions for this technique, some additional definitions are required. Given a clique $\mathcal{C}$ of the TCG, let K_β denote the kernel with maximum chain cycle among all kernels in the clique. The chain cycle of a session in which the kernels in $\mathcal{C}$ are concurrently tested is equal to $CC(K_\beta)$. Consider any subset of $\mathcal{C}$ and let K_α denote the kernel with maximum chain cycle in the subset. All subsets of $\mathcal{C}$ such that $CC(K_\alpha) < CC(K_\beta)$ are denoted as *subcliques* of $\mathcal{C}$. For the TCG in Fig. 8.2(b), the clique represented by $\{K_1, K_3, K_4\}$ has three subcliques corresponding to $\{K_3, K_4\}, \{K_3\}$ and $\{K_4\}$, respectively.

For a given TCG the subcliques of every clique can be enumerated. Let $\mathcal{CS}_a$ denote any clique or subclique with K_a representing the kernel with maximum chain cycle in $\mathcal{CS}_a$. Consider a specific subclique $\mathcal{S}_b$ and let K_b denote the kernel with maximum chain cycle in $\mathcal{S}_b$. The subclique $\mathcal{S}_b$ is said to be *dominated* if there exists either a clique or another subclique $\mathcal{CS}_a$ such that, (1) $CC(K_a) = CC(K_b)$, and (2) $\mathcal{S}_b \subset \mathcal{CS}_a$. In our example, among the subcliques of $\{K_1, K_3, K_4\}$, $\{K_3\}$ is dominated by the subclique $\{K_3, K_4\}$, and $\{K_4\}$ is dominated by the clique $\{K_2, K_4\}$.

Lemma 20 With parallel testing and a minimum shift policy, the set of feasible sessions corresponds to the union of the set of all cliques and the set of non-dominated subcliques.

Proof Assume that in an optimal schedule, there exists a test session TS_i that does not correspond to a clique or a non-dominated subclique. Let $\mathcal{SG}$ denote the subgraph induced by the set of kernels tested in this session. The kernels in $\mathcal{SG}$ must be a subset of at least one clique $\mathcal{C}$ in the TCG. If $\mathcal{SG}$ is not a subclique of $\mathcal{C}$, TS_i can be replaced with a test session corresponding to $\mathcal{C}$. Else if $\mathcal{SG}$ is a subclique,

we replace TS_i with a session corresponding to a clique or subclique that dominates $\mathcal{SG}$. In both cases the chain cycle required in applying N_i patterns to the kernels tested in the session does not increase. □

For the design in Fig. 8.1(b), the feasible test sessions correspond to the following sets of kernels: $\{K_1, K_3, K_4\}, \{K_3, K_4\}, \{K_2, K_4\}$ and $\{K_2\}$ with chain cycles equal to 32, 28, 24 and 8, respectively. By solving the integer program, we obtain an optimal schedule consisting of four sessions as summarized in Table 8.2. The chain cycles used in each session are shown under the minimum shift policy column. The total test time is equal to $100(32 + 1) + 300(24 + 1) + 100(24 + 1) + 150(8 + 1) =$ 15850 clock cycles, a 46.6% savings over sequential testing with the minimum shift policy. As with the flush policy, in determining an optimal schedule for designs with equal length kernels, the variables in the integer program need only assume $0 - 1$ values.

Among the four scheduling techniques, the greatest savings in test time are obtained with parallel testing under a minimum shift policy. However with this technique, both the optimal test schedule and the associated test time depend on the chain configuration. This technique is discused in more detail in the next section. In particular we study ways to effectively order the registers in the chain so as to minimize the overall test time.

8.6 Minimum Shift Policy

8.6.1 Importance of the scan chain configuration

To illustrate the effect of the scan chain configuration on the total test time, 1000 random permutations of the scan registers were generated for the example design. For each ordering, the scheduling algorithm for parallel testing under the minimum shift policy was used to determine the optimal test time. In Fig. 8.3, the different test times are plotted against their frequency of occurrence in the 1000 sample cases. The test times are normalized with respect to the optimal test time for parallel testing under the flush policy, i.e., 24050 clock cycles. For a worst case ordering the test

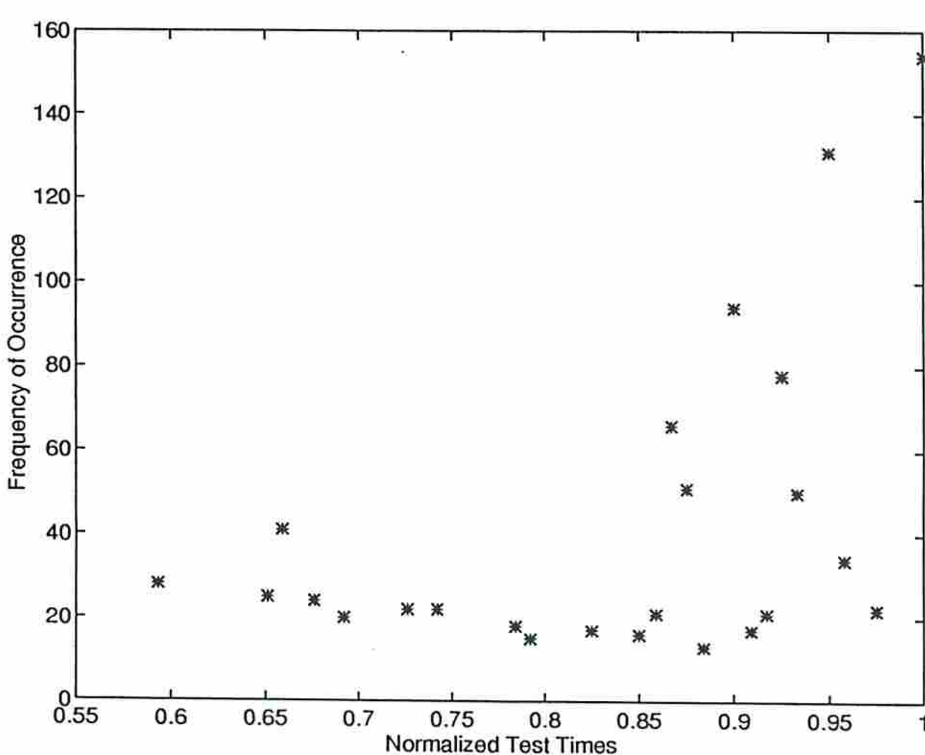


Figure 8.3: Optimal test times versus frequency of occurrence for 1000 orderings.

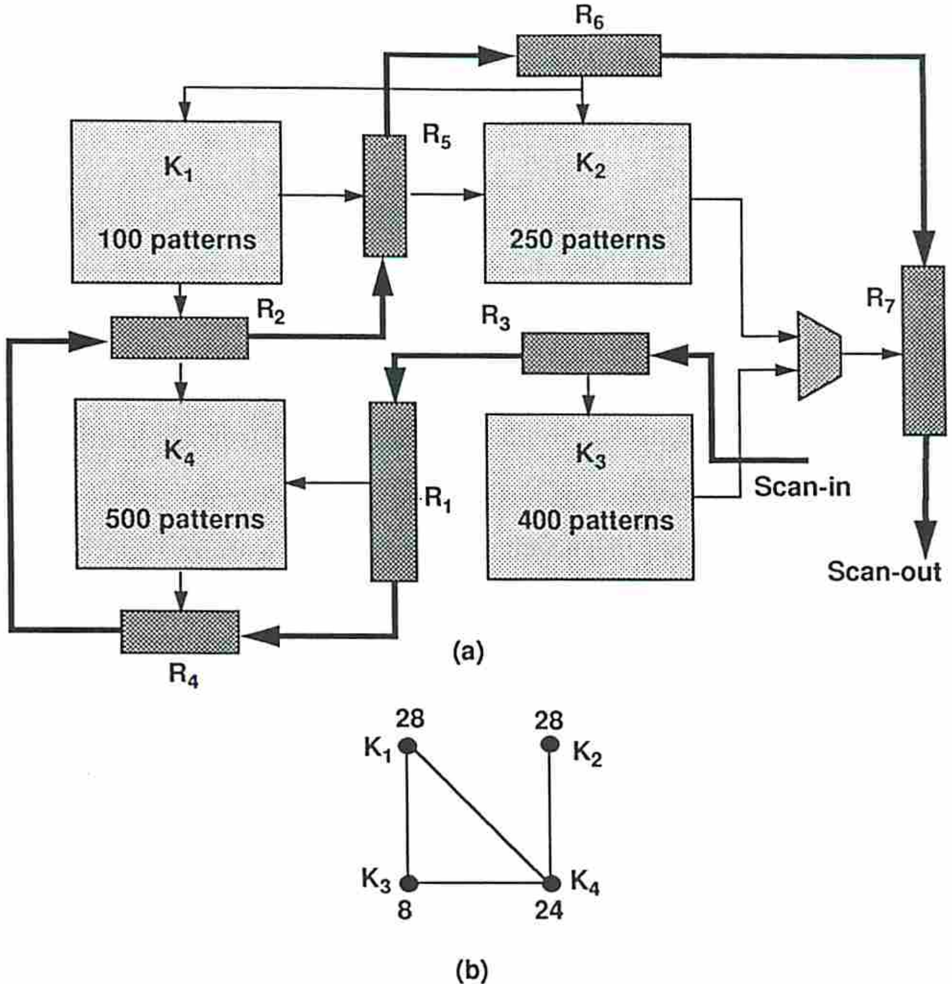


Figure 8.4: (a) Modified scan chain for design. (b) TCG of design.

time reduces to that under the flush policy. The range of normalized test times varying from 0.58 to 1 highlights the importance of the scan chain for the minimum shift policy.

To more clearly understand the influence of the scan chain on the scheduling of tests, consider our example design with a different order of scan registers as shown in Fig. 8.4(a). As a result the chain cycles of the kernels assume different values and the labeled TCG is shown in Fig. 8.4(b). The new set of feasible test sessions consists of the following subsets of kernels: $\{K_1, K_3, K_4\}, \{K_2, K_4\}, \{K_3, K_4\}$ and $\{K_3\}$ with chain cycles of 28, 28, 24 and 8, respectively. An optimal test schedule consists of the four test sessions shown in Table 8.3. Note the difference as compared

Test session	Kernels tested	Number of patterns	Chain cycle	
			Flush policy	Minimum shift policy
TS_1	K_1, K_3, K_4	100	36	28
TS_2	K_2, K_4	250	36	28
TS_3	K_3, K_4	150	36	24
TS_4	K_3	150	36	8

Table 8.3: Summary of test schedule for different order of scan registers

to the optimal schedule in Table 8.2. The associated test application time is equal to 15250 clock cycles, a 4% reduction over the previous scan chain.

The order of registers in the chain is required to determine both the feasible test sessions and their respective chain cycles. The test scheduling phase determines the subset of feasible sessions to be used in an optimal schedule and the number of patterns applied in these sessions. This close interaction between the two subtasks makes the problem of minimizing test time extremely complicated. With the minimum shift policy, a globally optimal solution is a combination of an optimal scan chain π_{opt} and its associated optimal schedule $\mathcal{TS}_{opt}$. A brute-force approach to generate a solution is to enumerate every possible scan chain and for each chain optimally solve the scheduling problem. Due to the exponential size of the search space, such an approach would be computationally infeasible. We therefore adopt a solution approach that divides the complete problem into a scan chaining phase and a test scheduling phase. In the next subsection, we show how the two phases are combined to minimize the total test time.

In the scan chaining phase, an objective function is required to evaluate the relative costs of two orderings without optimally solving the scheduling problem. With no knowledge of the schedule used in applying tests, it is difficult to formulate such an objective function. Recall that in the test scheduling phase, we assumed a given order of registers in the scan chain. Similarly in formulating the scan chaining problem let us assume a given *partial* schedule.

Definition Given a test schedule $\mathcal{TS}$ of complete test sessions, the corresponding **partial** schedule is identical to $\mathcal{TS}$ except that in each test session, the values of

the chain cycles are not specified. In other words, each test session of the partial schedule is characterized only by the subset of kernels tested and the number of patterns applied to these kernels.

The partial schedule derived from the test schedule of Table 8.2 is shown in Table 8.5. Note that the overlapped scheme is also an example of a partial schedule since it only specifies the kernels tested, and the number of patterns applied in each session. A partial schedule provides an objective function for the scan chaining phase. The order of registers will determine the chain cycles of each session and hence the total test time. Based on the concept of partial schedules, a constrained version of the scan chaining problem is formulated.

Constrained Scan Chaining Problem: Given a partial test schedule, determine an ordering of scan registers to minimize the test time for the given schedule.

Note that the above formulation generalizes the register sequencing problem presented in Chapter 4. The objective in Chapter 4 was to minimize the test time for a specific partial schedule, i.e., the overlapped scheme. Given a scan design we need to generate an appropriate partial schedule to be used in the scan chaining phase. Consider the following sequence of operations: (1) generate a random ordering of scan registers, (2) for this ordering, determine the optimal test schedule by use of the scheduling algorithm, and (3) derive the partial schedule of this optimal test schedule. Any one of the partial schedules obtained in this fashion can be used in the scan chaining phase. For example in configuring the chain for our example design, we could use either of the partial schedules derived from the schedules in Tables 8.2 and 8.3, respectively.

Although the above sequence of operations will generate a partial schedule, it is difficult to determine *apriori* the "best" partial schedule to use in the chaining phase. Recall that a globally optimal solution with the minimum shift policy is a combination of an optimal chain π_{opt} and an associated optimal test schedule $\mathcal{TS}_{opt}$. Ideally if the partial schedule corresponding to $\mathcal{TS}_{opt}$ was used in the chaining phase, the solution would correspond to a global optimum. The partial schedule derived from $\mathcal{TS}_{opt}$ is referred to as a *partially optimal* schedule.

The question arises as to whether a partially optimal schedule can be determined prior to the chaining phase. In determining a partial schedule, instead of a single

random ordering, let the sequence of operations be performed on every permutation of the registers in the chain, i.e., $M!$ orderings. If every ordering were to generate the same partial schedule, this would also have to correspond to a partially optimal schedule. This observation is valid for two classes of scan designs. The key point to note in these designs is that the scan chaining phase alone is sufficient to generate a globally optimal solution. These two classes correspond to the two extremes in the compatibility relationships among kernels in a design, i.e., n compatible kernels and n incompatible kernels.

Consider designs consisting of n compatible kernels, i.e., the TCG is represented by a complete graph on n nodes. This is equivalent to our previous circuit model in which every subset of kernels can be concurrently tested. In Chapter 3 we stated that the overlapped scheme represents a partially optimal schedule. A formal proof is given in the following lemma.

Lemma 21 For a design with n compatible kernels, the overlapped scheme constitutes a partially optimal schedule.

Proof Given any ordering of registers in the chain, we determine the optimal test schedule and show that the corresponding partial schedule is the overlapped scheme. Consider any two feasible test sessions TS_i and TS_j. The set of kernels tested in TS_i must be a proper subset of the kernels tested in TS_j, or vice versa. Else a feasible test session TS_α is created by taking the union of the kernels tested in the two sessions. The chain cycle of TS_α will be equal to $\max(CC_i, CC_j)$. The clique or subclique corresponding to this session will dominate at least one of the subcliques corresponding to TS_i or TS_j, contradicting the assumption that both are feasible test sessions. Hence the test sessions in an optimal schedule can be ordered $\{TS_1, TS_2, \ldots, TS_\gamma\}$ with $\gamma \leq n$ such that the kernels tested in TS_{i+1} are a proper subset of the kernels tested in TS_i $(1 \leq i < \gamma)$. This schedule can be refined to a test schedule of n test sessions, with kernel K_i fully tested at the end of session TS_i. The corresponding partial schedule is equivalent to the overlapped scheme. □

At the other end of the spectrum, for a design with n incompatible kernels, a partially optimal schedule corresponds to sequential testing, i.e., the schedule consists of n sessions $\{TS_1, TS_2, \ldots, TS_n\}$. In session TS_i, W_i patterns are applied

to kernel K_i ($1 \leq i \leq n$). The scan chaining phase is provided with an objective function given by

$$TT = \sum_{i=1}^{n} W_i(CC_i + 1)$$

For more general designs, it is difficult to determine *apriori* the partially optimal schedule. Let us therefore assume a given partial schedule $\{TS_1, TS_2, \ldots, TS_\gamma\}$ with N_i patterns applied to the kernels tested in TS_i ($1 \leq i \leq \gamma$). The problem of determining an appropriate partial schedule is considered in the next subsection. The goal of the chaining phase is to minimize the chain cycle of each session and hence the total test time for the given schedule. Thus

$$TT = \sum_{i=1}^{\gamma} N_i(CC_i + 1)$$

8.6.2 Configuring the Scan Chain

The decision variables in configuring the chain are the positions of the registers in the chain. The objective function to be minimized corresponds to the weighted sum of the chain cycles. Since the constrained chaining problem is similar to the problem addressed in Chapter 4, most of the techniques can be directly adapted to this more generalized version. In particular the notation used in Section 4.3 is used to model the optimization problem. Similarly the NP-completeness of the problem follows from the proof furnished in Section 4.4.1. The difference arises in using the adjacency interchange property to restrict the size of the search space. The interchange property outlined in Chapter 4 is only valid on the assumption that all kernels are compatible and the overlapped scheme is used in applying tests. Both the enhanced circuit model and the use of a partial schedule different from the overlapped scheme will require us to appropriately modify the property. In this regard, the parameters of the scan registers will play a crucial role. The parameters for all seven registers of the design in Fig. 8.1(b) are shown in Table 8.4.

The adjacency interchange property is generalized in two ways. The first deals with its use independent of any partial schedule. Recall that a partial schedule was introduced in the chaining problem to permit two different orderings to be

Register R_j	K_j^D	K_j^R	d_j	r_j	l_j	$T(R_j)$
R_1	K_4	$\emptyset$	500	0	8	TS_1, TS_2, TS_3
R_2	K_4	K_1	500	100	4	TS_1, TS_2, TS_3
R_3	K_3	$\emptyset$	400	0	4	TS_1, TS_3, TS_4
R_4	$\emptyset$	K_4	0	500	4	TS_1, TS_2, TS_3
R_5	K_2	K_1	250	100	4	TS_1, TS_2
R_6	K_1, K_2	$\emptyset$	350	0	4	TS_1, TS_2
R_7	$\emptyset$	K_2, K_3	0	650	8	TS_1, TS_2, TS_3, TS_4

Table 8.4: Parameters for scan registers in example design

compared without optimally solving the scheduling problem. It is however possible to determine the relative positions of certain registers in the chain independent of the partial schedule. In other words these registers must satisfy this relative order for every partial schedule and hence in the optimal scan chain π_{opt}.

Lemma 22 - *Schedule-independent sequencing rule*
Consider any two registers R_α and R_β where $K_\alpha^D \subseteq K_\beta^D$ and $K_\beta^R \subseteq K_\alpha^R$. For a given scan chain, if either (1) $p_\alpha < p_\beta$ and $l_\alpha = l_\beta$, or (2) $p_\alpha = p_\beta - 1$, interchanging the positions of the two registers will not increase the test time with any partial schedule.

Proof The interchange does not affect the driver and receiver distances of the other registers in the chain. As a result the chain cycle of a test session in which both registers are not used will remain the same. For a test session in which one or both registers are used, one of three cases is possible: (a) both are used as drivers and/or receivers, (b) R_α is used as a receiver, or (c) R_β is used as a driver. Since the interchange decreases both the driver distance of R_β and the receiver distance of R_α, the chain cycles of these sessions cannot increase. □

For the registers R_5 and R_6 in our example, since $K_5^D \subset K_6^D$ and $K_6^R \subset K_5^R$, R_6 must always be assigned to a lower-numbered slot as compared to R_5. The idea of a precedence relation used in Chapter 4 can similarly be invoked on every pair of equal length registers that satisfy the conditions of Lemma 22. The resulting *precedence* graph is shown in Fig. 8.5(a).

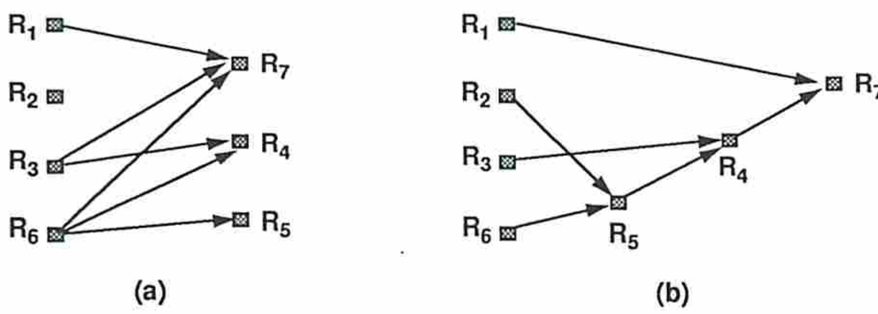


Figure 8.5: (a) Precedence graph independent of partial schedule. (b) Modified precedence graph based on partial schedule.

Test session	Kernels tested	Number of patterns
TS_1	K_1, K_3, K_4	100
TS_2	K_2, K_4	250
TS_3	K_3, K_4	150
TS_4	K_3	150

Table 8.5: Partial test schedule used to configure scan chain

The use of Lemma 22 however provides limited reduction in the search space. The need for a partial schedule is therefore essential to further constrain the search space as well as evaluate the relative costs of different orderings. For our example design, let us adopt the partial schedule shown in Table 8.5. For a given partial schedule, let $T(R_j)$ denote the subset of test sessions in which R_j $(1 \leq j \leq M)$ is used either as a driver and/or receiver. The last column in Table 8.4 shows the corresponding sessions for each register. Using this information the adjacency interchange property can also be employed in the case when a partial schedule is known prior to configuring the chain. The property as presented in Chapter 4 restricts the partial schedule to the overlapped scheme. The following lemma generalizes it to any given partial schedule.

Lemma 23 - *Schedule-dependent sequencing rule*
Given any partial schedule and a scan chain, the position of two registers R_α and R_β in the chain can be interchanged as in Lemma 1 if both the following conditions are satisfied: (1) $T(R_\alpha) \subseteq T(R_\beta)$ or vice versa, and (2) $d_\alpha \leq d_\beta$ and $r_\alpha \geq r_\beta$.

Proof Similar to proof of Lemma 1. □

Corollary - If there exists k pure drivers (receivers) used in all sessions, these registers can be placed in the first (last) k slots in the scan chain. □

In the case of the overlapped scheme, only the second condition needs to be verified since any two registers automatically satisfy the first condition. The corollary generalizes the dominance property presented in Chapter 4. In the partial schedule of Table 8.5, since R_7 is used as a pure receiver in all sessions, it can be assigned to the last slot in the scan chain. The more generalized adjacency property is used to add more edges to the precedence graph of Fig. 8.5(a). The resulting precedence graph is shown in Fig. 8.5(b).

To optimally configure the chain for any given partial schedule, the implicit enumeration technique of Section 4 can be directly employed. The different aspects of the technique can be adapted to the constrained chaining problem. These aspects include (1) lower bounds on the chain cycles of each session, (2) evaluation of look-ahead bounds for intermediate configurations, (3) generation of initial solutions, (4) pruning mechanisms based on the look-ahead bounds, and (5) pruning mechanisms

based on the precedence graph and Lemma 23. Routing constraints can be similarly modeled in the technique by restricting the set of candidates for the next available slot in the chain.

8.6.3 Iterating between scheduling and chaining

With the minimum shift policy, the solution approach is divided into two phases: (1) a *test scheduling phase* - given a scan chain configuration, determine an optimal schedule to minimize the test time, and (2) a *scan chaining phase* - given a partial test schedule, determine a scan chain configuration to minimize the test time for the schedule. The following pseudo-code outlines the steps used in the test scheduling phase. The *size* of a clique or subclique is equal to the number of nodes in it.

Test-Generation-Phase (Inputs: $\pi, KSEQ, TCG$)
begin

Evaluate chain cycles of each K_i in $KSEQ$ $(1 \leq i \leq n)$.
Attach chain cycles to corresponding nodes in TCG.
Enumerate all cliques of TCG and store in array *Cliques*.
Let $\mathcal{C}_{max}$ denote maximum size among all cliques in *Cliques*.
For $\alpha = \mathcal{C}_{max} - 1$ down to 1
    For each clique, generate all subcliques of size α.
    Check if subclique is dominated by any element of *Cliques*.
    If not dominated, insert subclique in *Cliques*.
For each clique or subclique $\mathcal{SC}$ in *Cliques*.
    Evaluate maximum chain cycle among all kernels in $\mathcal{SC}$.
    Introduce variable with corresponding weight in objective function.
For each kernel K_i in $KSEQ$
    Evaluate all feasible sessions in which K_i is tested.
    Introduce constraint equation in integer linear program.
Solve integer linear program.
Generate refined optimal test schedule.
Return *Optimal-Test-Time*.

end

The procedure **Generate-Optimal-Chain()** in Section 4.5 provides an outline of the technique used in the scan chaining phase. The partial schedule is added as an input to the procedure. In addition the extended definitions of the precedence graph and adjacency interchange property are used in pruning the search space. The two phases are combined in an iterative fashion to generate a "global" solution that minimizes the test time. The pseudo-code of the iterative procedure is given below.

Iterative_procedure()
begin
Initialize $Chain_time = 0$.
Initialize $Schedule_time = -1$.
Let π = a topological ordering of schedule-independent precedence graph.
Initialize $chain\text{-}flag = sched\text{-}flag = 1$.
While ($Chain_time$ is not equal $Schedule_time$)
If ($sched\text{-}flag$ equals $chain\text{-}flag$)
$Schedule_time$ = **Test-Generation-Phase**($\pi, KSEQ, TCG$).
Let $\mathcal{TS}$ = optimal refined test schedule from scheduling phase.
Initialize $\mathcal{PTS}$ = partial test schedule derived from $\mathcal{TS}$.
Increment $sched\text{-}flag$ by 1.
Else
$Chain_time$ = **Generate-Optimal-Chain**($\mathcal{PTS}, S, WSEQ$).
Let π = optimal scan chain derived from chaining phase.
Increment $chain\text{-}flag$ by 1.
Output $\mathcal{TS}$, π and associated test time.
end

If (1) the TCG is a complete graph on n nodes, or (2) the TCG consists of n disjoint nodes, or (3) sequential testing is used, a single iteration of the scan chaining phase is sufficient to generate an optimal solution. In these three cases the partially optimal schedule is known apriori. For every other case, we start with a topological ordering of the schedule-independent precedence graph. This ordering is input to the test scheduling phase and the associated optimal test schedule is obtained. The corresponding partial schedule is then input to the scan chaining phase. In this way we iterate over the two phases until their corresponding test times converge to the

same value. The resultant scan chain and its associated test schedule represent a *locally* optimal solution with respect to the initial ordering. This solution is not guaranteed to be globally optimal. To increase the probability of converging to a global optimal, the procedure can be repeated with different topological orderings of the precedence graph. The best solution among the resultant set of locally optimal solutions can be chosen as the final solution. Note that if every topological ordering of the precedence graph is either used as an initial ordering or encountered in the course of the iterations, the final solution is guaranteed to be globally optimal.

In the iterative procedure, an alternate technique is to start with a given partial schedule. For the TCG in our example design, it can be shown that there exists only two possible partial schedules. These two partial schedules can be derived from the schedules in Tables 8.2 and 8.3, respectively. Hence a globally optimal solution can be obtained by individually inputting these schedules to the scan chaining phase and choosing the best solution. However for more complex TCGs, the number of partial schedules could be exponential in the number of kernels in the design. Note that the feasible sessions with the minimum shift policy correspond to both the set of cliques and non-dominated subcliques of the TCG. With no knowledge of the scan chain, it is impossible to identify the set of dominated subcliques. In such a case, every subset of nodes for a given clique could potentially generate a session in the partial schedule.

8.7 Experimental Results

8.7.1 A Case Study

For the design in Fig. 8.1, optimal test times for the different scheduling techniques are compared in Table 8.6. The test times are given in terms of the number of clock cycles. In a conventional scan design approach, if the circuit is handed as a single block of logic to a combinational ATPG system, it is difficult to predict the number of patterns that will be generated. This would depend on both the structure of the logic and the characteristics of the ATPG system, e.g., the number of faults targeted per pattern and the strategy used to specify don't care bits. In our design a lower

bound on the number of patterns is equal to 650, i.e., no pattern can simultaneously test K_2 and K_3 due to the multiplexer. Under a worst case scenario the number of patterns would be equal to 1250, i.e., every pattern tests only one of the kernels in the design. In Table 8.6, for the conventional approach the number of patterns generated is given by the average of these two numbers. The last column in Table 8.6 represents the percentage savings over the test time with the second technique, i.e., the conventional approach based on the combined scheme. With the minimum shift policy, the optimal scan chains for both sequential and parallel testing are identical for the example design, although in general they can be different. The order of registers in the chain is given by $R_3, R_1, R_2, R_6, R_5, R_4, R_7$.

The last entry in the table is based on dividing the circuit into two disjoint kernels consisting of K_4 and the "composite" kernel consisting of K_1, K_2, K_3, and the multiplexer. As outlined above, the test length of the composite kernel is equal to 700, i.e., the average of the best and worst case scenarios. The test time shown for this entry corresponds to the use of an optimal scan chain under the overlapped scheme and a minimum shift policy. The ordering of registers in the chain is given by $R_3, R_6, R_1, R_5, R_2, R_4, R_7$.

8.7.2 Discussion

The results with our example circuit illustrate the advantages of a combined approach in identifying smaller kernels, and using efficient chaining and scheduling techniques to reduce the overall test time. Partitioning a circuit into *disjoint* kernels will always reduce the total test time as compared to the unpartitioned circuit. However the test time savings in further partitioning each of these kernels depends on a number of factors. These include the logic complexity of the subkernels, their degree of overlap, the efficiency of the test generator, and the manner in which test patterns are generated. To more clearly illustrate these aspects, consider the two designs shown in Fig. 8.6. The registers in both designs are of equal length, i.e., 4 flip-flops.

Consider the design shown in Fig. 8.6(a) and let $\alpha \geq 1$. Two possible options can be adopted in using an ATPG system to generate test patterns. The design can either

	Scheduling Technique	No. of sessions	Kernels tested	No. of patterns	Chain cycle	Test Time	% savings
1	Flush policy Sequential testing	4	K_1	100	36	46250	-
			K_2	250	36		
			K_3	400	36		
			K_4	500	36		
2	Conventional approach (average case)	1	K_1, K_2, K_3, K_4	950	36	35150	0
3	Flush policy Parallel testing	4	K_1, K_3, K_4	100	36	24050	31.6
			K_3, K_4	300	36		
			K_2, K_4	100	36		
			K_2	150	36		
4	Minimum shift policy Sequential testing	4	K_1	100	24	20850	40.7
			K_2	250	24		
			K_3	400	8		
			K_4	500	16		
5	Minimum shift policy Parallel testing	4	K_1, K_3, K_4	100	24	12650	64.0
			K_2, K_4	250	24		
			K_3, K_4	150	16		
			K_3	150	8		
6	Minimum shift policy Overlapped scheme	2	K_{comp}, K_4	500	24	16700	52.5
			K_{comp}	200	16		

Table 8.6: Optimal test times for different scheduling techniques

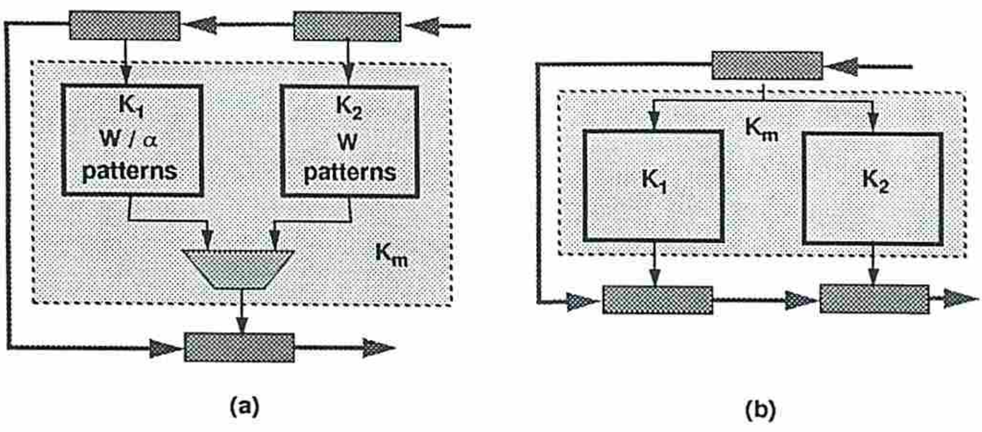


Figure 8.6: (a) Example design with multiplexer. (b) Example design with a fanout point.

be processed as a single block of logic K_m, or each of the subkernels K_1 and K_2 can be processed individually. Due to the multiplexer, the minimum number of patterns to test K_m cannot be less than the sum of the minimum number of patterns required for K_1 and K_2, respectively. With the second option the registers in the chain can be ordered to exploit the disparity in the test lengths of the two subkernels. Assume the ordering of registers as shown in the figure and let a minimum shift policy be employed. If $W_m = W_1 + W_2$ and $\alpha = 1$, the test time savings in using the second option over the first is equal to 25%. As $\alpha \rightarrow \infty$ the total savings can increase to 50%. Individually processing each subkernel is thus a better design option in this case. Note that additional patterns might need to be used to detect all faults in the multiplexer.

Let us turn our attention to the design shown in Fig 8.6(b). The subkernels K_1 and K_2 might represent standard library modules, each associated with a precomputed set of test patterns. In this case, since patterns need to be applied individually to each subkernel, the design is modeled as two incompatible kernels. As in the previous case the chaining phase can exploit any disparity in the test lengths to reduce the test application time. However if test patterns are to be generated by an ATPG system, processing K_1 and K_2 individually might not be the best option. For example, if K_1 and K_2 are identical circuit modules, testing them individually will increase the test time by 50% as compared to the time in testing the design as a

single kernel K_m. However by modifying the manner in which tests are generated for the design, we can still exploit the internal structure to reduce the test application time. Two different options can be adopted in this regard.

- Test patterns are first generated for the subkernel K_2. The resultant test set can be used as random patterns to catch faults in K_1 as well. Test generation for K_1 can then be restricted to the remaining undetected faults. Note however that the order in which the subkernels are processed will affect the size of the final test set.

- Test patterns are generated assuming the design consists of a single kernel K_m. The circuit is then fault simulated with each pattern in the test set. Once a fault is detected, it is dropped from the fault list. Each pattern can be tagged with the subkernels in which it detects one or more faults. In this way, the test set can be divided into three smaller sets based on whether a pattern detects faults in both subkernels, or solely in K_1 or K_2.

With the first option, it is difficult to predict the size of the test set relative to the number of patterns in processing the design as a single kernel. However for both options, the design can be modeled as three incompatible kernels consisting of K_m, K_1 and K_2, respectively. For each kernel, we can identify the number of patterns that need to be applied to it and the registers involved in testing the kernel. By employing the technique based on sequential testing under the minimum shift policy, the total test time can be minimized. The second option will clearly lead to test time savings over treating the design as a single kernel. However the test time savings with the first option will depend on the relative sizes of the test sets.

The partitioning of a scan design can be performed either manually by designers based on their knowledge of the functionality and structure of modules, or by means of automated partitioning software. In the latter case, the computational costs of the partitioning techniques should not outweigh the savings achieved in reducing test generation costs. Even with a given partitioned design, it is difficult to determine the "best" option to use in generating tests. The logic complexity of the subkernels, their degree of overlap and the sophistication of the test generator should be considered in determining an appropriate strategy. As a result the final test application

time greatly depends on both the partitioning technique and the strategy used in generating tests. In any case, the techniques presented in this chapter provide the means to minimize the test time for a given design.

8.8 Summary

The use of more refined techniques to partition a scan design gives rise to resource conflicts in testing the different kernels. It thus becomes essential to consider the test scheduling problem in minimizing the total test time. Previous approaches to the scheduling problem only exploit parallelism in testing kernels concurrently. In scan designs an additional facet is provided by reducing the shifting time in the test process. As a direct consequence the scan chain configuration and the shift policy greatly affect the total test time. In this chapter we have presented a range of scheduling techniques classified by the degree of parallelism and shift policy used in the test process. In particular the use of parallel testing with the minimum shift policy can provide significant savings in test time. For this technique, there exists a close interaction between the scan chaining and test scheduling phases. We have developed an efficient algorithm that iterates between the two phases to minimize the overall test time. In the next chapter, we show how these scheduling techniques and the different chain methodologies can greatly enhance the scan design process.

Chapter 9

Integrating to a Scan Design System

9.1 Introduction

Each of the chaining and scheduling methodologies developed in the previous chapters can individually be used for a given design. In doing so, a designer can efficiently minimize the test application time for the chosen methodology. However in incorporating scan testability, a designer is often faced with the dual requirement of having to reduce the test overheads while at the same time satisfying given design constraints. In this context it is essential to provide a designer with a range of strategies to effectively tradeoff different design costs.

In this chapter we discuss the integration of the chaining and scheduling techniques to a scan design system. Although the techniques can enhance the capabilities of any system, we deal specifically with the design framework of the `SIESTA` scan system developed at USC. `SIESTA` consists of four main subsystems dealing with the issues of scan register selection, circuit partitioning, scan chaining and test scheduling. Its main objective is to efficiently explore the design search space to generate optimized testability solutions. By systematically addressing each facet of the scan design process, it attempts to find solutions that satisfy both a designer's goals and constraints. A brief overview of the system architecture is provided in the next section.

By integrating the chaining and scheduling techniques to `SIESTA`, a designer can analyze the different tradeoffs available with regard to incorporating scan testability.

In particular it is useful to determine the relative design costs in employing each technique for a given scan design. An efficient way to achieve this is by means of design estimators that predict the effects of a particular technique on the different design costs. In Section 9.3 we discuss the utility of such design estimators in the scan design process. In addition we indicate some approaches to develop such estimators within the SIESTA design framework. The wide range of techniques embodied in the chaining and scheduling subsystem greatly enhance the scan design process. We conclude this chapter with some experimental results on a datapath circuit. The design costs of the various chain methodologies are compared to highlight the range of options available to a designer. More importantly we show the effectiveness of the different techniques in generating optimized scan design solutions.

9.2 System Overview

We briefly discuss the general architecture of the SIESTA scan system and put in perspective the role of the chaining and scheduling subsystem in the design process. Fig. 9.1 shows the design flow through the different components of the system. Each of the modules in the system is briefly described.

- **Pre-processing** - Circuits are stored and manipulated within Cbase which is an object-oriented design framework that provides a repository of VLSI design data and operations [48]. As shown in Fig. 9.1, designs can be input into Cbase in two different ways. In particular EDIF provides a gateway to accept designs from the ADAM high-level synthesis system [47]. Given any design it is useful to represent the design as blocks of combinational logic separated by registers. CRETE is a pre-processor that reorganizes a given hierarchical or flat description of a circuit into a new hierarchical description where all combinational logic and flip-flops are identified and partitioned into disjoint *clouds* and registers, respectively [37]. A cloud is defined as a maximal group of directly interconnected combinational blocks. This view of the circuit enables efficient application of concepts such as partitioning, resource allocation and test scheduling as used in various DFT and BIST methodologies. CLARION is a similar circuit pre-processor that provides a more refined view by restricting

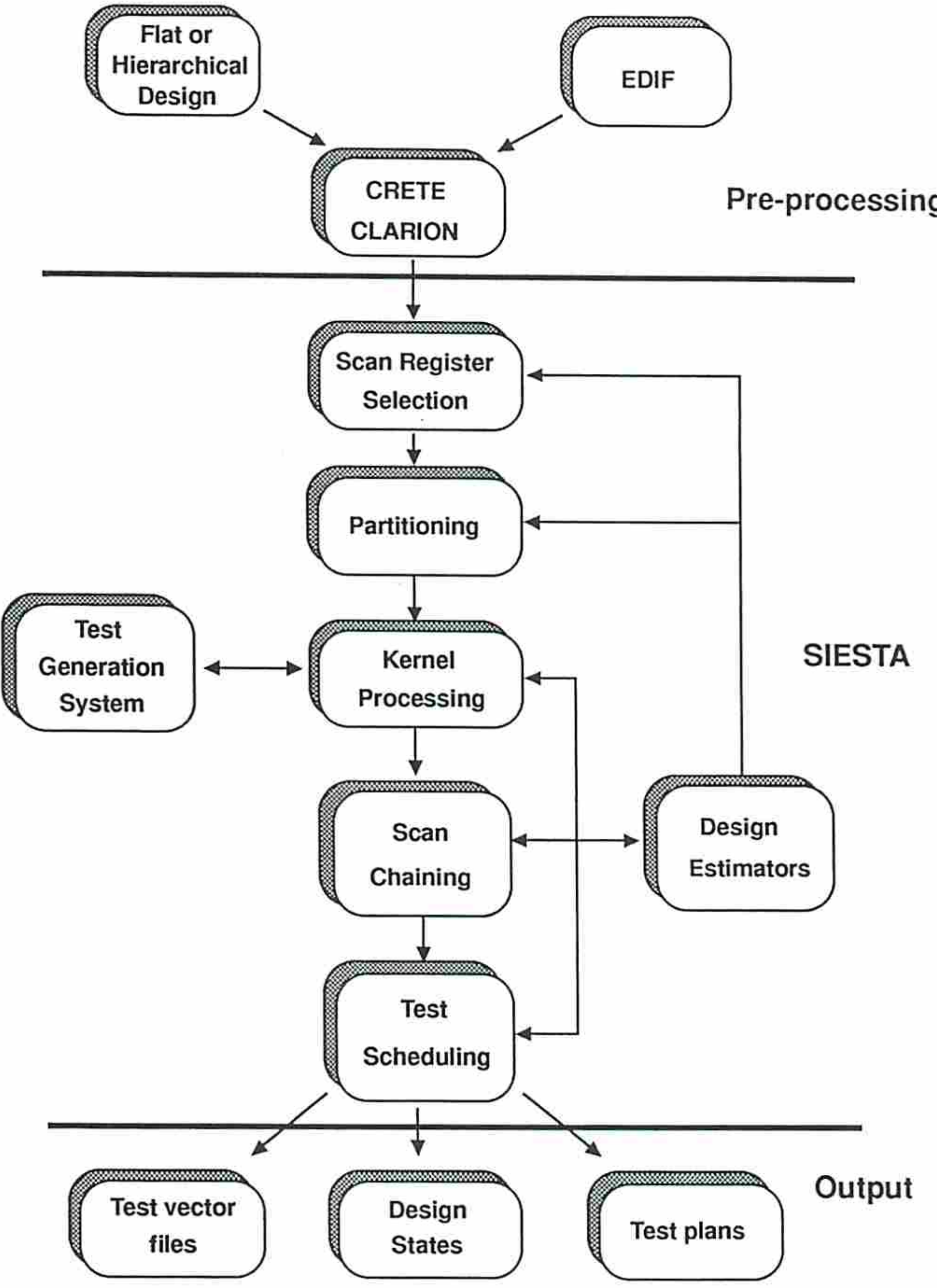


Figure 9.1: Flowchart of the SIESTA scan design system.

the manner in which connected logic is clustered into clouds [41]. Consequently certain fanout points in the design are isolated as independent entities.

- **Scan register selection** - The top level of the hierarchical circuit from CRETE/CLARION consists of clouds, fanout points and registers. This represents the formal input to SIESTA. The first stage deals with selecting the set of storage elements to be made scannable. Both full scan and a range of partial scan methodologies are employed in the system. The partial scan methodologies include:

  1. *BALLAST* - a minimal set of registers is selected such that the resultant kernel is both acyclic and balanced [6]. Every fault in a balanced kernel can be detected by a single test pattern held at the inputs of the kernel for a number of clock cycles equal to the sequential depth of the kernel. Combinational test pattern generation is sufficient to obtain a complete test set. BALLAST takes advantage of registers that already possess shift capabilities and avoids selecting registers that lie in critical paths [35].
  2. *ACYST* - a minimum number of registers is selected to break all cycles in the circuit. The resulting acyclic kernel requires a sequence of one or more test patterns applied over consecutive clock cycles to detect a single fault. The test sequence length is bounded by the sequential depth of the kernel. To generate test patterns for the kernel, either sequential TPG or combinational TPG with a multiple fault model can be used [35].
  3. *CYCLIST* - this methodology incurs the least area overhead in terms of scan flip-flops by selecting a minimum number of flip-flops to break all cycles except self-loops [5]. The resulting kernel is cyclic and the number of test patterns required to fully test the kernel might be exponential in the number of unscanned flip-flops. Sequential TPG is needed to generate the required test patterns.

  All three of the above methodologies in conjunction with full scan provide tradeoffs between the area and performance overheads based on the choice and number of scan flip-flops, and the resultant complexity of the test generation process.

- **Partitioning** - The kernels resulting from the scan selection analysis can often be broken down into smaller kernels. Two different partitioning schemes can be employed. The first termed as *output-based* partitioning can be used in one of two modes. The two modes differ based on the degree of overlap between the resulting partitions. In the first mode the kernel is broken down into smaller disjoint subkernels. For example with full scan designs, the use of output-based partitioning in this mode divides the circuit into disjoint clouds. The second mode exploits the fanout points in the design to further reduce the size of each subkernel. As a result two subkernels might share one or or more fanout points. The second partitioning scheme termed as *size-based* partitioning is applicable to partial scan designs. Size-based partitioning divides the entire kernel into smaller disjoint subkernels by adding more scan registers. The designer can set an upper bound on the number of gates in each of the resulting partitions.

- **Kernel Processing** - This module is involved in appropriately modifying each kernel in the partitioned scan design before invoking the appropriate TPG system to generate test patterns. In each case, the scan flip-flops are replaced by pseudo-inputs/outputs and in the case of balanced kernels, non-scan registers are replaced by delayless wires. With full scan and the BALLAST methodology, it is sufficient to invoke the combinational TPG system. On the other hand, the sequential TPG system needs to be used with the ACYST and CYCLIST methodologies. A file of test patterns is created for each kernel in the design.

- **Scan chaining** - The primary function of this module is to configure the scan registers into a chain organization based on the user's choice. The chain methodologies incorporated in this module include a single scan chain and reconfigurable scan chain under both shift policies, and multiple scan chains under the flush policy. Note that the chaining phase implicitly assumes knowledge of the schedule used in applying tests. When the design consists of disjoint kernels the overlapped scheme is employed. In the presence of incompatibilities the chaining phase iterates with the scheduling phase to determine both the chain configuration and the associated test schedule. In every case the main objective is to minimize the test application time. A designer is also provided

with an option to vary the number of multiplexers and number of chains in the case of a reconfigurable chain and multiple chains, respectively.

- **Test Scheduling** - This module is closely linked with the strategy used in the scan chaining phase. Note that certain kernels in a design might be sequential while other kernels might be combinational. The test scheduling phase will determine the appropriate set of actions needed to test the individual kernels. For each test session in the optimal test schedule, the following information is generated: (1) the kernels tested in the session, (2) the number of patterns applied, (3) the chain cycle of the session, (4) the chain(s) used in the session, and (5) the number of clock cycles for which each pattern needs to be held in the scan chain. This information is encapsulated in a *test plan* file that will constitute one of the outputs of the system.

- **Design Estimators** - At each step of the design process, it is difficult to predict the effect of a decision on the quality of the final design. In particular it is important to know if a particular technique will violate any testability goals. In addition the large number of alternative methodologies in each module of the system leads to a huge design space. The design estimator module will encapsulate a set of bounding functions that can be used to guide the designer in determining a feasible solution. The details of these estimators will be discussed in the next section.

- **System outputs** - For every feasible design generated by the system, information about the selected scan registers, the routing of the scan chains, additional hardware for the chain methodology, the number of pins, the fault coverage, and test application time is written into a file. This *design state* file is concatenated with the test plan file and handed to the test control subsystem [49]. The test control subsystem will synthesize the required test controller and appropriately modify the circuit database. The test patterns for the individual kernels in a design also have to be *edited* to conform to the scan chain organization and the test schedule. This will facilitate their application by automatic test equipment.

The integration of the chaining and scheduling subsystems to SIESTA will complete the scan design framework as depicted in Fig. 9.1. For a given circuit, this will enable us to explore the design search space and analyze the various costs and tradeoffs in incorporating scan testability. The capabilities of the system in this regard will be illustrated in Section 9.4.

9.3 Design Estimators

9.3.1 Motivation

In general a designer may specify a set of testability goals and design constraints that need to be satisfied in making a design scan testable. For example the goals might be specified as lower bounds on fault coverage and/or upper bounds on area overhead and test time. Similarly design constraints might be expressed in the form of critical paths that restrict the choice of scan registers. In employing the scan design system, it is important to guide the designer in selecting a set of designs that satisfy his/her requirements. The following two issues highlight the motivation in using design estimators within the scan design environment.

- As outlined in the previous section, SIESTA incorporates testability as a set of sequential tasks dealing with scan register selection, partitioning, kernel processing, scan chaining and test scheduling. The range of alternative methodologies for each of these tasks leads to a huge design space. Fig 9.2 represents a design scenario highlighting the options at each stage of the design cycle. The prohibitive computational costs in examining every design makes it necessary to prune certain portions of the search tree in generating a feasible design.

- Many of the tasks involved in the scan design process such as test generation, configuring of scan chains and test scheduling have been shown to be NP-hard. Before deciding to generate optimal solutions to any of these tasks, a *heuristic* estimate of the effects of a chosen methodology on the different design costs can be used to efficiently direct the search process.

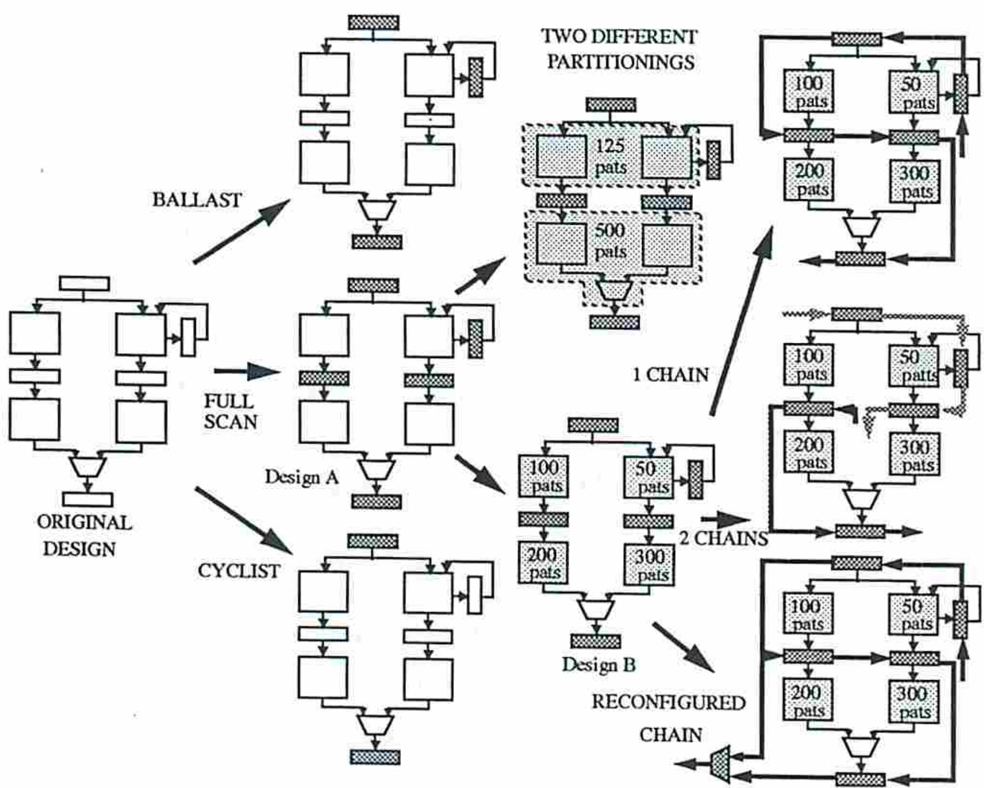


Figure 9.2: Search tree model of design space.

Each of the design tasks in SIESTA is performed independently in its own problem space. However the solutions to each of the tasks are interdependent and information in the form of partial design states must be passed between the tasks to generate a complete solution. The design estimators will enable earlier tasks to determine the effect of their decisions on the solutions to subsequent tasks and hence on the global solution. In this way estimators can lead to two potential benefits. They can be used to (1) choose those branches in the search tree that most likely lead to acceptable solutions, and (2) prune certain design states and their descendants from further consideration.

9.3.2 Characteristics of the Design Estimators

The search tree shown in Fig 9.2 will be used to describe the salient features in developing design estimators. Note that the amount of information in the partial design states increases with each level of the search tree. At every level a partial design state will permit certain overheads to be accurately evaluated; however determining other design costs will require further information provided by subsequent tasks in the design process. Consider the partial design A shown in Fig. 9.2. All registers are made scannable and the resulting kernel is partitioned. To predict the design costs of test application time, one can make use of information about the logic complexity of each kernel and the registers involved in testing the different kernels. On the other hand, for design B in the next level of the tree, test generation has been carried out on the different kernels. This additional piece of information can be used to more accurately estimate the final test application time. The design estimators should thus take into account the partial state of the design.

Given a partial design state, let us assume we want to predict the test time for a particular chaining and test scheduling methodology. Measures such as the lower and upper bounds on test time can be determined to guide the search process. The *admissibility* of these measures relate to their ability in correctly bounding the final test time either from above or below. Note that the computations necessary in evaluating these measures should not offset the gains in reducing the number of states that are examined.

Our main emphasis in this discussion is to estimate the test application time for the different methodologies at a specific level of the search tree. This corresponds to the partial design state after the tasks of scan register selection, partitioning, and test generation. The design at this stage consist of kernels with known test lengths prior to both chaining and scheduling. Consider for example the partial design state B shown in Fig. 9.2. At this point, lower and upper bounds on the test application time can be generated for each of the different chaining and scheduling methodologies. Based on these estimates, the designer can either refine the design state using a particular methodology or *backtrack* to a previous state in the search tree.

9.3.3 Estimating the Test Application Time

In this section we present some approaches in developing admissible lower and upper bounds on the optimal test time for the chaining and scheduling methodologies. Consider the circuit shown in Fig. 9.3(a) consisting of two kernels K_1 and K_2 with test lengths of 100 and 400, respectively. The five scan registers in the design each contain 4 flip-flops. We will use this example to illustrate the derivation of lower and upper bounds on the test time. For designs in which all kernels are compatible, the overlapped scheme represents an optimal test scheme. The estimators can use this information in determining the bounds on test time with each chain methodology. Note that for each kernel in the design we can determine the scan registers (flip-flops) used in testing it. Let us consider each of the three chain methodologies for such designs. In developing estimators for each methodology we adopt the notation used in the respective chapter describing the methodology. As outlined in Chapter 3, a scan design consists of n kernels and is associated with the sets $KSEQ$ and $WSEQ$.

Single Scan Chain - For the overlapped scheme with a minimum shift policy, an upper bound on test time is equivalent to the use of the combined scheme, i.e., the entire chain is shifted for each pattern and the number of patterns applied is equal to W_n. For our example design in Fig. 9.3(a), this corresponds to $400(20 + 1) = 8400$ clock cycles. A tighter upper bound can be derived by using a topological ordering of the precedence graph described in Chapter 4. For this given ordering, the test

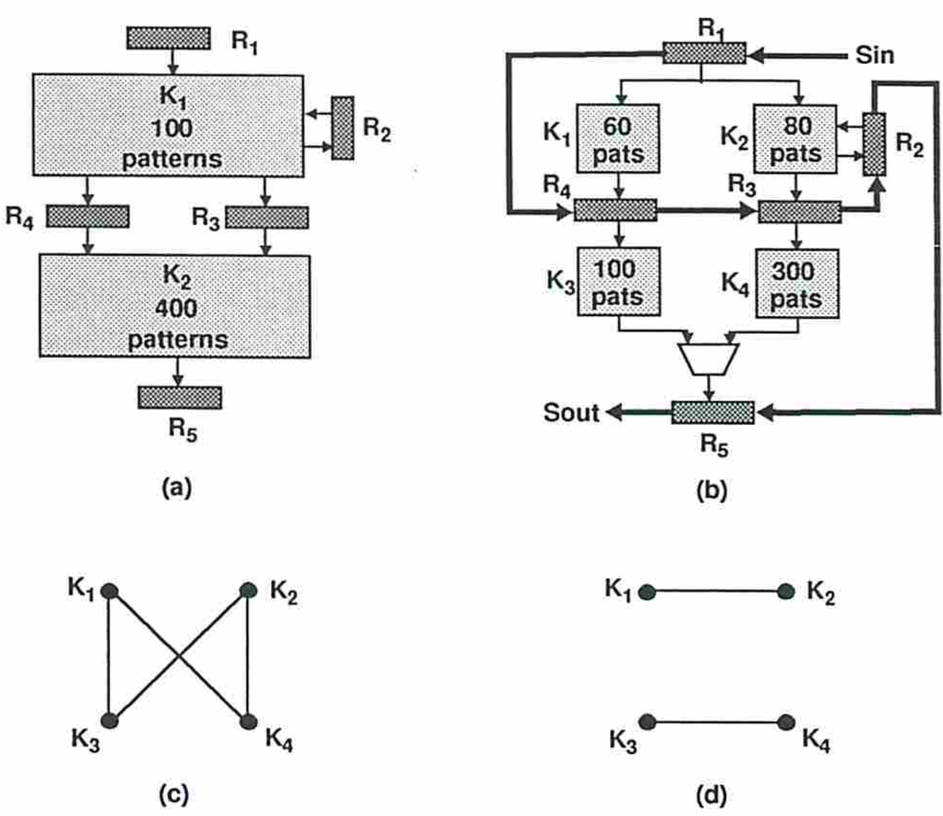


Figure 9.3: (a) Partial design of two compatible kernels. (b) Partial design of four incompatible kernels. (c) TCG of design. (d) TIG of design.

time is obtained by evaluating the chain cycles for each session of the overlapped scheme under a minimum shift policy. Note that evaluating this upper bound is of time complexity polynomial in the number of registers and sessions. To derive a lower bound on the test time, the expressions derived in Section 4.5.1 can be used to determine lower bounds on the chain cycles in each session. Equation 3.1 can then be used with these minimum chain cycles to obtain a lower bound on the test time. The admissibility of this bound follows from the manner in which the chain cycles are obtained. For our example circuit, the lower bound is equal to 100(16 + 1) + 300(8 + 1) = 4400 clock cycles.

Upper Bound

- Construct the precedence graph for the M scan registers based on the driver and receiver weights of each register.
- Generate a topological ordering of registers to obtain a feasible scan chain π.
- Evaluate chain cycles for each session and use the following equation (similar to equation 3.1) to obtain an upper bound.

$$\mathcal{T} = \sum_{i=1}^{n}(W_i - W_{i-1})(CC_i + 1) \tag{9.1}$$

Lower Bound

- For each session, evaluate L_d = sum of lengths of drivers: L_r = sum of lengths of receivers; and L_{dr} = sum of lengths of driver-receivers. Let L represent the sum of the lengths of the M scan registers.
- Use the following equation to evaluate minimum chain cycles for each session of the overlapped scheme.

$$MCC = \begin{cases} max(L_d, L_r) & \text{if } L_{dr} = 0 \\ max(L_d, L_r, \lceil \frac{L - L_{dr}}{2} \rceil) + L_{dr} & \text{if } L_{dr} > 0 \end{cases} \tag{9.2}$$

- Evaluate lower bound by using the minimum chain cycles in equation 9.1.

Reconfigurable Scan Chain - With the flush policy an upper bound on test time is given by use of the combined scheme. As with a single scan chain, a topological ordering of the precedence graph can be used to generate an upper bound with the minimum shift policy. Lower bounds for both policies can be obtained as outlined in Section 6.3.2. Note that the upper bounds assume no reconfiguration, while the lower bounds are independent of the number of multiplexers used to reconfigure the chain. Ideally it would be advantageous to determine upper and lower bounds assuming j multiplexers are used in reconfiguring the chain. Deriving such refined bounds however might not be as useful since the results in Chapter 6 clearly show that in most cases 1-3 multiplexers are sufficient to achieve test times within 5% of the lower bound. Recall that with the flush policy, the algorithm to generate an optimal solution is of time complexity polynomial in the number of kernels and multiplexers. The efficiency of the algorithm makes it more advantageous to generate optimal test times rather than rely on the use of bounds to guide the designer. For our example design, the lower bound with the flush policy is given by $100(20 + 1) + 300(12 + 1) = 6000$ clock cycles. Similarly with the minimum shift policy, it is equal to $100(16 + 1) + 300(8 + 1) = 4400$ clock cycles.

Lower Bound

- For each session, let L_d, L_r and L_{dr} denote the same values as with a single scan chain.
- For the flush shift policy, the minimum chain cycle of a session is given by $L_d + L_r + L_{dr}$.
- For the minimum shift policy, the minimum chain cycle of a session is equal to $\max(L_d, L_r) + L_{dr}$.
- Use equation 9.1 to generate lower bounds on test time for both shift policies.

Multiple Scan Chains - Given that k scan chains are to be used, an upper bound is given by the test time in configuring equal length chains. For the design in Fig. 9.3(a), an upper bound in configuring four chains is given by $400(5 + 1) = 2400$ clock cycles. To determine a lower bound, let us evaluate the least possible

chain cycle that can be employed in a session. For each session we can identify the flip-flops that are not used in the session. Let α_i denote the number of flip-flops that are not used in a particular session TS_i. To determine a lower bound on the chain cycle of the session, two possible scenarios can be considered. In the first case, the α_i flip-flops can be isolated in a single chain and the remaining $N - \alpha_i$ flip-flops equally divided among the remaining $k - 1$ chains. In the second case all N flip-flops are equally divided among the k chains. In other words the minimum chain cycle that can be employed in session TS_i is given by $min\,(\lceil \frac{N}{k} \rceil,\ \lceil \frac{N-\alpha_i}{k-1} \rceil)$. For the example design in Fig. 9.3(a), let us assume four chains are to be configured. Lower bounds on the chain cycles of the two sessions are equal to 5 and 4, respectively. Thus a lower bound on the test time is equal to $100(5 + 1) + 300(4 + 1) = 2100$ clock cycles. The admissibility of this bound follows from the optimistic assumption used in evaluating the chain cycles. Similar to a reconfigurable chain under the flush policy, using the dynamic programming algorithm to generate optimal test times can provide more effective feedback to a designer.

Upper Bound

- Assuming equal length chains, the chain cycle of every session is equal to $\lceil \frac{N}{k} \rceil$.
- Using equation 9.1 an upper bound on the total test time can be determined.

Lower Bound

- Let α_i denote the number of flip-flops that are not used in session TS_i $(1 < i \leq n)$.
- If $\alpha_i = 0$, the minimum chain cycle of the session is equal to $\lceil \frac{N}{k} \rceil$.
- Else the minimum chain cycle of the session is given by $min\,(\lceil \frac{N}{k} \rceil,\ \lceil \frac{N-\alpha_i}{k-1} \rceil)$.
- The minimum chain cycles for each session are used in equation 9.1 to derive a lower bound.

For designs in which all kernels are not mutually compatible and in the absence of a chain configuration, deriving bounds on the test time is more difficult. The only information available in evaluating bounds is the test lengths of the kernels, the

TCG of the design, and the scan registers used in testing each kernel. For sequential testing under the flush policy, the optimal test time can be derived in time complexity polynomial in the number of kernels. Let us consider the remaining three scheduling techniques for the case of a single scan chain. We use the design shown in Fig. 9.3(b) to illustrate the evaluation of these bounds. This design represents a more refined view of the design in Fig. 9.3(a). The TCG of the design is shown in Fig. 9.3(c).

Parallel testing with flush policy - Note that information about the ordering of the registers in the chain is not required with a flush policy. Both upper and lower bounds on the test time can be determined by employing the complementary graph of the TCG referred to as the *test incompatibility* graph (TIG). For our example design the TIG is shown in Fig. 9.3(d). A graph coloring algorithm tries to color each node in a graph such that no two adjacent nodes, i.e. nodes with a common incident edge, have the same color [43]. Based on this idea, a *greedy* coloring approach can be used to generate maximal sets of compatible kernels. These sets of kernels can in turn be used to generate an upper bound on the test time. For example in the TIG of Fig. 9.3(d), the node K_1 is initially tagged with a specific color. Since only one of either K_3 and K_4 can be colored with the same color, one of them is chosen in creating the maximal set $\{K_1, K_3\}$. These two nodes along with any incident edges are then removed from the TIG and the same procedure is repeated for the remaining nodes. In this way two compatible sets of kernels $\{K_1, K_3\}$ and $\{K_2, K_4\}$ are generated. Each set can be mapped to a test session. Since every kernel can only be an element of a single maximal set, it is fully tested in a single test session. The upper bound on the test time is thus given by $100(20 + 1) + 300(20 + 1) = 8400$ clock cycles. The time complexity in deriving this upper bound is of order $O(n^2)$ where n represents the number of kernels.

Given any two cliques, one clique is said to be *larger* than the another if the sum of the test lengths of the kernels in the first clique is greater than the corresponding value for the second. To determine a lower bound on the test time, we enumerate the cliques of the TIG to determine the largest clique. Since no two kernels in this clique can be concurrently tested, the time to sequentially test the kernels in the clique represents a lower bound on the test time. For our example design, a lower

bound is given by the clique $\{K_3, K_4\}$ in the TIG of Fig. 9.3(d). The associated test time is equal to $100(20+1)+300(20+1)=8400$ clock cycles. In determining the lower bound, note that the cliques of the TIG need to be enumerated. In deriving an optimal test schedule, note that each of these cliques is mapped to a variable in the integer linear program.

Upper Bound

- Given the TCG of the design, generate the complementary graph, i.e., the TIG.
- Initialize *Total-test-time* $= 0$ and let L denote the length of the scan chain.
- Initialize *Current-maximal* $= \emptyset$.
- Color a node in the graph with a specific color and add it to the set *Current-maximal*. Determine a node that is not adjacent to any node in *Current-maximal* and add it to the set. Continue in this greedy fashion until no more nodes can be added to *Current-maximal*.
- Delete all nodes in *Current-maximal* along with any adjacent edges from the TIG. Let W_{max} denote the test length of the node with maximum test length in *Current-maximal*.
- The nodes in *Current-maximal* are mapped to a test session with test time given by $W_{max}(L+1)$. Add this value to *Total-test-time*.
- Repeat previous four steps until there exist no more nodes in the TIG.
- The value of *Total-test-time* provides an upper bound on the test time.

Lower Bound

- Generate the TIG from the TCG of the design.
- Enumerate all cliques of the TIG and for each clique C, let $W(C)$ = sum of test lengths of kernels in the clique.
- Initialize *Total-test-time* $= 0$ and let L denote the length of the scan chain.

- Choose the clique with maximum weight and for each kernel K_α in the clique, evaluate a cost function given by $W_\alpha(L+1)$.

- Let *Total-test-time* be equal to the sum of the cost functions of all kernels in the maximum weight clique. This corresponds to a lower bound on the total test time.

Sequential testing with minimum shift policy - To determine an upper bound, a topological ordering of the precedence graph as outlined in Section 8.6.2 can be used as a "hypothetical" scan chain. Based on the ordering of registers, the chain cycle of each kernel can be determined to evaluate the total test time. Consider the order of registers as shown in the scan chain of Fig. 9.3(b). The chain cycles of the four kernels K_1, K_2, K_3 and K_4 are equal to 16, 16, 8 and 12, respectively. The corresponding test time is thus equal to 8480 clock cycles. To evaluate a lower bound on the test time, we can determine lower bounds on the chain cycles to individually test each kernel. For each kernel the chain can be configured to solely minimize its respective chain cycle. For example in testing K_2, if the registers are ordered as R_1, R_4, R_2, R_3, R_5 from scan-in to scan-out, the chain cycle of K_2 is minimized, i.e., 12 clock cycles. Hence a lower bound on test time for the design is equal to $60(4+1) + 80(12+1) + 100(4+1) + 300(4+1) = 3340$ clock cycles.

Upper Bound

- Construct the precedence graph for the M scan registers based on the driver and receiver weights of each register.

- Generate a topological ordering of registers to obtain a feasible scan chain π.

- For each kernel in $KSEQ$, determine the drive, receive and chain cycle to solely test the kernel based on the ordering of registers in π.

- The test time to solely test kernel K_i $(1 \leq i \leq n)$ is given by $W_i(CC(K_i)+1)$, where $CC(K_i)$ represents the chain cycle to test kernel K_i.

- An upper bound is given by adding up the test times to individually test each kernel.

Lower Bound

- For each kernel K_i in $KSEQ$, let L_d, L_r and L_{dr} represent the sum of the lengths of registers used as drivers, receivers, and driver-receivers, respectively, in testing K_i.
- Use equation 9.2 to evaluate the minimum chain cycles to test each kernel.
- A lower bound is given by the sum of the test times to individually test each kernel.

Parallel testing with minimum shift policy - As in the sequential case, the precedence graph can be used to obtain an ordering of registers in the chain. For this given ordering, the TIG can be modified by attaching the chain cycle of each kernel to the respective node. An upper bound on test time can then be obtained by use of the greedy coloring approach on the TIG as outlined for the flush policy. For example let us assume the coloring algorithm identifies the two maximal sets given by $\{K_1, K_3\}$ and $\{K_2, K_4\}$. Note that in mapping these sets to test sessions, we take into account the kernels being tested and their respective chain cycles. For the ordering of registers as shown in Fig. 9.3(b), the chain cycles of K_1 and K_3 are equal to 16 and 8, respectively. The set of kernels $\{K_1, K_3\}$ is thus mapped to two sessions. In the first session, 60 patterns are applied to both kernels with a chain cycle of 16. In the second session 40 patterns are applied to K_3 with a chain cycle of 8. A similar situation occurs with the second set of kernels $\{K_2, K_4\}$. Hence an upper bound on the test time is given by $60(16 + 1) + 40(8 + 1) + 80(16 + 1) + 220(12 + 1) = 5600$ clock cycles.

To derive a lower bound, the minimum chain cycle to individually test each kernel can be obtained as in the sequential testing technique. For each node in the TIG we attach a value given by the product of the kernel's test length and its minimum chain cycle. The cliques of the TIG can be enumerated and the clique with a maximum sum of the individual node values is used to derive a lower bound. The lower bound on test time is given by the time to sequentially test the kernels in this clique. For our example design, the clique with maximum sum in the TIG is given by $\{K_3, K_4\}$ with an associated test time of $300(4 + 1) + 100(4 + 1) = 2000$ clock cycles.

Upper Bound

- Construct precedence graph for M scan registers based on driver and receiver weights of each register.
- Generate a topological ordering of registers to obtain a feasible scan chain π.
- Generate maximal subsets of compatible kernels as outlined for parallel testing with the flush policy.
- Let $MS = \{K_{m_1}, K_{m_2}, \ldots, K_{m_\gamma}\}$ denote a maximal subset such that $m_1 < m_2 < \ldots m_\gamma$.
- The time to test all kernels in the maximal subset MS is given by the following expression. Assume $W_{m_0} = 0$.

$$\sum_{i=1}^{\gamma}(W_{m_i} - W_{m_{i-1}})(CC(K_{m_i}, \ldots, K_{m_\gamma}) + 1)$$

  where $CC(K_{m_i})$ denotes the chain cycle to individually test K_{m_i}, and $CC(K_{m_i}, \ldots, K_{m_\gamma})$ is given by $max(CC(K_{m_i}), \ldots, CC(K_{m_\gamma}))$.

- An upper bound is given by the sum of the test times for each of the maximal compatible subsets of kernels.

Lower Bound

- For each kernel in the TIG, attach a cost function given by the product of the test length and minimum chain cycle of the kernel to the corresponding node in the TIG.
- Enumerate all cliques of the TIG
- Choose the maximum weight clique where the weight of a clique is equal to the sum of the cost functions of the kernels in the clique.
- For each kernel K_i in the maximum weight clique, the time to solely test the kernel is equal to $W_i(MCC(K_i) + 1)$ where MCC_i represents the minimum chain cycle to test K_i, as derived in the sequential case.

Test Scheme	Shift policy	Chain configuration	No. of chain	Upper bound	Lower bound
Combined	Flush	Single chain	-	8400	8400
Overlapped	Min. shift	Single chain	-	5600	4400
Overlapped	Flush	Reconfigurable	-	8400	6000
Overlapped	Min. shift	Reconfigurable	-	5600	4400
Overlapped	Flush	Multiple	2	4400	4400
Overlapped	Flush	Multiple	3	3200	2900
Overlapped	Flush	Multiple	4	2400	2100

Table 9.1: Upper and lower bounds on test time for chain methodologies

- The sum of the times to test the kernels in the maximum weight clique represents a lower bound on the test time.

Two points are worth noting about the lower and upper bounds derived for the different techniques. The admissibility of the bounds follows from the manner in which the bounds are derived. In addition the computations necessary to evaluate the bounds have less complexity as compared to deriving optimal solutions. In traversing the design space, one possible way to guide a designer is to provide a table of the upper and lower bounds for each chaining and scheduling methodology. This should be done prior to invoking any of the algorithms for a specific methodology. This in turn will provide a designer with an approximate idea of the "best" case and "worst" case test times for each methodology. For example with the partial design state shown in Fig. 9.3(a), Table 9.1 could aid a designer in choosing an appropriate chain methodology.

The estimation of test time can be done one step earlier in the design cycle by considering partial design states prior to test generation. However this would require us to estimate the test lengths of the different kernels in the design. The feasibility of design estimators at this level of the search tree is a subject for future study. With the increasing efficiency of current combinational test generation systems, the reduced costs of test generation might not warrant complex heuristic functions. On the other hand, design estimators might prove useful for designs containing sequential kernels. One possible approach to tackling this problem is to use the results of running

test generation on designs considered early in the design process for designs that are subsequently analyzed. For example it is useful to generate a full scan design since knowledge of the test lengths for the different kernels in the circuit can be subsequently used to estimate the test lengths of larger kernels composed of these smaller kernels. Another possible approach is to employ some form of structural gate-level analysis to determine probabilistic bounds on the test length of a kernel.

The design estimators discussed in this section will provide the means to tightly integrate the chaining and scheduling subsystem with the other modules in the scan design system. However their effectiveness in both traversing and pruning the design search space needs to be further investigated.

9.4 Experimental Results

Given a circuit, `SIESTA` permits a designer to explore a range of options in making the circuit testable. In particular structured circuits such as those used in data path and signal processing applications are ideal candidates to study the different design costs and tradeoffs. The application of `CRETE/CLARION` to such circuits generates a number of clouds of varying complexity. Conversely circuits such as finite state machines and control-dominated designs do not prove to be as interesting. In such circuits the combinational logic, made accessible by scan techniques, can in general be clustered only into a single cloud.

The `SIESTA` design system was used to run a set of experiments on the *USC Data Path-I* shown in Figure 4.6. Recall that this circuit consists of 7 combinational modules, 7 registers of width 8 bits, and 3 registers of width 16 bits. By exercising different options in selecting scan registers, an entire range of scan designs can be generated. For example, the BALLAST technique requires a minimum of four registers, e.g., R_2, R_4, R_5, R_9, to be made scannable. By adding and removing constraints on this selection, different designs using the same methodology are produced. Having selected the scan registers, the resulting kernels in each design are partitioned in various ways by modifying the user specified size limit in size-based partitioning. Test patterns for each kernel are obtained using the ATPG system. During test generation each control input to a kernel is treated as a primary input.

DI	SSM	AO	CR	SSR	LSB	NKP	NTP	MSD	T1	T2
1	FS	104	-	All regs.	-	1	221	0	23309	23309
2	FS	104	-	All regs.	-	7	179	0	18899	7803
3	BPS	32	-	R2, R4, R5, R9	2500	1	814	5	30964	30964
4	BPS	32	R5	R2, R3, R4, R10	2500	1	834	3	30056	30056
5	BPS	40	R4, R9	R2, R3, R5, R6	2500	1	1005	3	44260	44260
6	BPS	48	-	R2, R4, R5, R8, R10	1400	2	576	3	29424	21406
7	BPS	64	-	R2, R3, R4, R5, R6, R9, R10	1000	4	179	1	11878	8144
8	BPS	72	-	R2, R4, R5 R6, R8, R9, R10	500	4	179	2	13497	7557
9	APS	16	-	R2, R4	2500	1	2772	5	53434	53434

Table 9.2: Nine scan designs of the *USC Data Path-I*

For each design we evaluate optimal test times for a single scan chain under both the combined scheme, and the overlapped scheme with a minimum shift policy.

Table 9.2 shows nine representative designs that reflect the range of options and tradeoffs embodied in the system. In the table the following abbreviations are used; DI – design index; SSM – scan selection methodology (FS - full scan, BPS - BALLAST partial scan and APS - ACYST partial scan); AO – area overhead in terms of the number of scan flip-flops; CR – critical registers, i.e., registers that cannot be scanned; SSR – selected scan registers (from both the scan methodology and partitioning process); LSB – limit used in size-based partitioning based on the number of gates; NKP – number of kernels after partitioning; NTP – number of test patterns for largest kernel; MSD – maximum sequential depth among all kernels; T1 - test time under the combined scheme for a single scan chain; and T2 – test time for an optimal scan chain under the overlapped scheme with a minimum shift policy. The test times are in terms of the number of clock cycles.

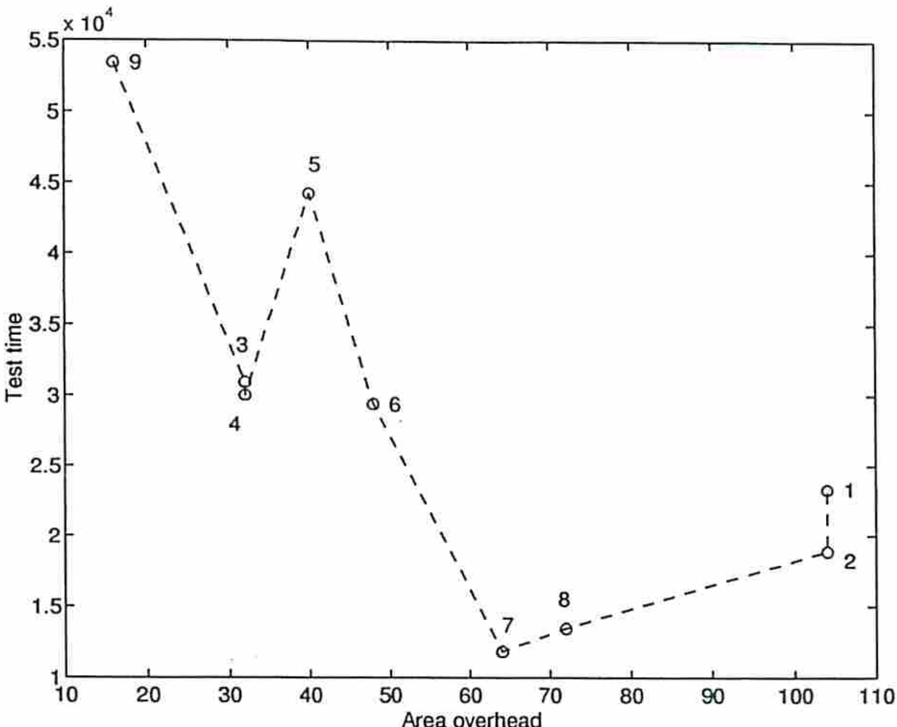


Figure 9.4: Area-test time tradeoffs with the combined scheme.

Designs 1 and 2 represent full scan designs. The single kernel in Design 1 is partitioned into seven disjoint clouds in Design 2. Designs 3, 4 and 5 denote different implementations of the BALLAST methodology. Since only four registers are made scannable, both the area overhead and the time required to shift in a single test pattern are reduced. However the additional number of test patterns required to achieve complete coverage of all faults increases the test time as compared to full scan. Designs 6, 7 and 8 highlight the advantage of combining a partial scan methodology with partitioning techniques. These designs require a reduced number of scan registers as compared to full scan, and use less patterns to test each kernel as compared to the unpartitioned BALLAST designs. In addition testing the kernels concurrently reduces the overall test application time. Design 9 shows the results of applying the ACYST methodology; the disproportionate increase in the number of test patterns offsets the advantage of a reduced number of scan flip-flops, leading to large test times.

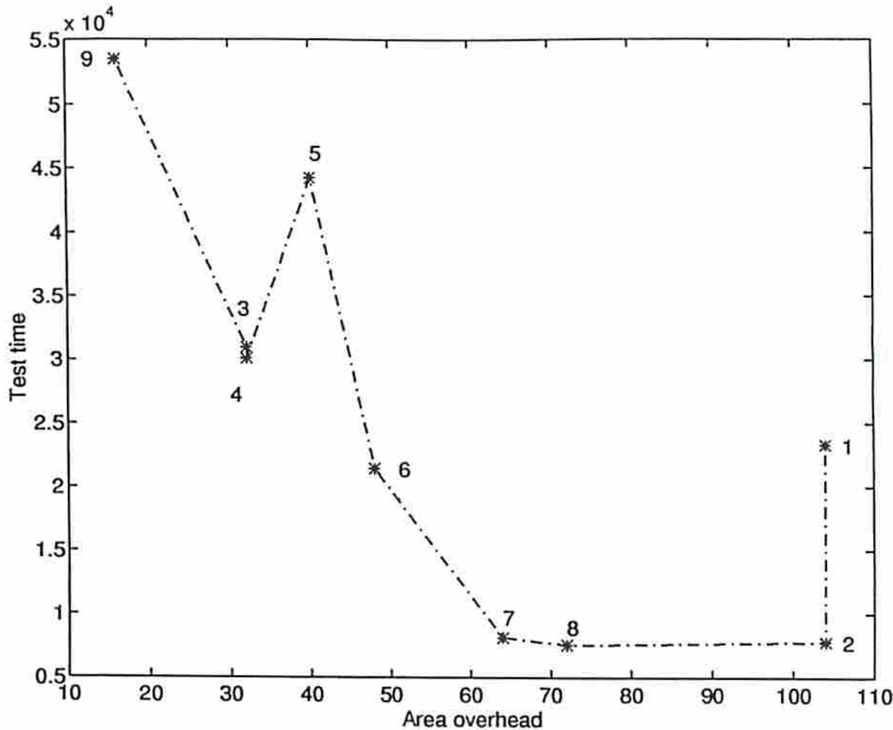


Figure 9.5: Area-test time tradeoffs with an optimal single scan chain.

Fig. 9.4 illustrates the area overhead-test time tradeoffs for the *USC Data Path-I* using the nine scan designs in Table 9.2. The graph plots the area overhead as given by AO against the test time as given by T1 in the table. The design indices shown in Table 9.2 are used to label the points in the graph. Similarly Fig. 9.5 plots the area overhead against the test time as given by T2 in Table 9.2. Contrary to the classical area-time tradeoff curve, in which area is inversely proportional to test time, SIESTA generates designs such as Designs 7 and 8 with lower area overheads and test times as compared to a full scan design. This feature of the system highlights the benefits in adopting a range of strategies to explore the design space. More importantly note the effect of configuring an optimal scan chain on the area-time plot. The test times of Designs 1, 3, 4, 5 and 9 are not influenced by using the overlapped scheme since these designs only consist of a single kernel. On the other hand, test time savings of 60% and 28% are obtained for Designs 2 and 6, respectively. Note that by configuring an optimal chain, the test time for Design 8 is reduced by a greater percentage as compared to Design 7. Generating such optimized scan solutions affords a designer great flexibility in choosing a design that satisfies his/her requirements.

To fully appreciate the range of available options, let us consider a specific scan design of the *USC Data Path-I*. In Design 2 we replace the *ALU* which has a test length of 179 with a more complex module that requires 500 test patterns. This increases the disparity in the frequency of usage of the scan flip-flops in the design. This modified design is used to compare the design costs and tradeoffs for the various scan chain methodologies as shown in Table 9.3. The methodologies are ordered in the table based on decreasing test application times. The first three columns denote the test application scheme, the shift policy, and the chain configuration, respectively. The test times are given in terms of clock cycles and the percentage savings are with respect to the first methodology. The logic overhead column lists any additional hardware or control signals required over the first methodology. The last column reflects the constraints with regard to flexibility in routing the scan chain(s). For example with two equal length chains no constraints are imposed on both the assignment and order of scan flip-flops in the chains. Conversely with a reconfigured chain under the minimum shift policy, both the assignment and order of registers in the serial chains are constrained.

9.5 Summary

The integration of the scan chaining and test scheduling subsystem completes the SIESTA design framework as shown in Fig. 9.1. We have discussed the motivation in using design estimators within the scan design system. In addition some approaches to developing such estimators for the chaining and scheduling methodologies have been indicated. However their use in guiding the design process and pruning portions of the search tree needs to be further studied. A more significant aspect in integrating the subsystem is the increased flexibility it brings to the scan design process. A designer is provided with a range of options to both optimize the test application overheads and trade off different design costs. A distinctive feature of SIESTA is its ability to provide a spectrum of optimized solutions that address both the testability costs and design constraints. As shown in the flowchart of Fig. 9.1, the user has the capability to backtrack and iterate over each of the modules in the system to find a design solution that satisfies his/her requirements.

	Test scheme	Shift policy	Chain configuration	Test time	Pin count	Logic overhead	Routing constraint
1	Combined	flush	single chain	52604	3	-	no constraint
2	Overlapped	flush	reconfigured chain	29244 44.4%	3	1 mux	assignment
3	Combined	flush	2 equal length chains	26552 49.5%	5	-	no constraint
4	Overlapped	flush	reconfigured chain	24532 53.4%	3	3 muxes	assignment
5	Overlapped	flush	2 optimal chains	23804 54.7%	5	-	assignment
6	Two-phase overlapped	flush	2 asynchronous chains	20932 60.2%	5	1 mode signal	assignment
7	Overlapped	min. shift	single chain	15828 69.9%	3		ordering
8	Overlapped	min. shift	reconfigured chain	15684 70.2%	3	2 muxes	assignment & ordering
9	Combined	flush	4 equal length chains	13526 74.3%	9	-	no constraint
10	Overlapped	flush	4 optimal chains	11368 78.4%	9	-	assignment
11	Two-phase overlapped	flush	4 asynchronous chains	8775 83.3%	10	1 mode signal	assignment

Table 9.3: Design costs for range of chain methodologies

Chapter 10

Conclusion and Future Work

With the increasing demand for quality in both design and fabrication, the costs of testing occupy a significant portion of the design cycle. In particular with increasing device complexities, the costs of test application grow in proportion to both the number of scan elements and the number of test patterns. In this thesis we have studied and analyzed different ways to minimize the overheads of test application for scan designs. An entire range of techniques has been developed to provide a flexible design methodology that minimizes the test application time while trading off different design costs. The main contributions of the research deal with the two main issues of scan chaining and test scheduling. In addition we have shown how integrating these techniques to a scan design system can greatly enhance the overall design process. In this chapter we present a summary of the contributions made in this thesis, and discuss some open problems that could possibly stimulate further research. As per the organization of the dissertation, we individually consider each of the chaining methodologies and scheduling techniques, and their subsequent integration to the SIESTA scan design system.

10.1 Scan Chain Methodologies

The different aspects of scan chaining and test scheduling are closely linked to the manner in which a scan design is partitioned. In general there exists a close interaction between the problem of configuring the registers into an organization, and the

associated problem of applying tests to the different kernels in a design. However for circuits in which every subset of kernels can be concurrently tested, we formally proved that the overlapped scheme represents an optimal test application scheme. In particular it exploits information about the design and is more efficient as compared to the conventional combined scheme. In conjunction with the overlapped scheme we introduced the concept of a shift policy to specify the amount of shifting during the test process. The optimization problem is thus reduced to configuring the scan registers so as to minimize the test time under the overlapped scheme. The key point to note is that the organization of the scan registers has a significant impact on the total test time. Based on the three criteria of the test scheme, the shift policy and the chain configuration, a classification of the different chain methodologies was developed. This provided an ideal starting point to study the different chain organizations and correlate their respective design costs. An underlying theme to our work on scan chaining is the observation that certain scan registers are accessed more frequently than other registers in the test process. This observation is exploited in the different chain methodologies

10.1.1 Configuring a Single Scan Chain

In using the overlapped scheme with the minimum shift policy, the ordering of registers in a single chain influences the total test time. We show how the optimal ordering of registers is sensitive both to the test lengths of the kernels, and the roles played by the scan registers. The basic idea in the methodology is to reduce the shifting time by placing the frequently accessed registers close to the ends of the chain. The problem of optimally ordering the registers was formally shown to be NP-hard. Based on a detailed analysis of the problem, an implicit enumeration technique was developed to generate an optimal ordering of registers. In addition we developed lower and upper bounds on the optimal test time. An important point to note is that the single chain methodology adds no extra hardware to the design and can lead to significant savings in test time over the combined scheme. The potential savings in test time are in fact theoretically unbounded, since they greatly depend on the characteristics of the kernels and registers in a design. Any routing overhead in configuring the chain only involves the routing of a single-bit wire between registers

or flip-flops. In addition the implicit enumeration technique can be suitably modified to incorporate any routing constraints. Two possible extensions to the methodology can be considered.

- To characterize the complexity of optimally ordering the registers under the assumption that all registers are of equal length.
- To consider a more constrained version of the problem that takes into account the positions of the scan registers in the physical layout. The objective function might be to minimize the test time under the constraint that either (1) the total wire length, or (2) the wire length between any two adjacent registers, be less than a user-specified upper bound. Note that our implicit enumeration technique can be modified to take into account the second case. However a more efficient algorithm might be required for designs with a large number of registers.

10.1.2 Multiple scan chains

The most significant aspect of our work on multiple scan chains is to dispel the popular notion that equal length chains guarantee minimum test time. We clearly show how unequal length chains can lead to lower test times. The basic idea in our methodology is to assign the frequently used scan elements to shorter scan chains. Given a design with N scan flip-flops and a user's requirement k on the number of chains, a polynomial-time algorithm was developed to optimally configure the chains. The algorithm is based on dynamic programming and has a worst-case time complexity of order $O(kN^2)$. Since the flush policy is used in applying tests, the optimal chain configuration only constrains the assignment of flip-flops to the k chains. The designer is thus provided full flexibility to specify the order of flip-flops within each chain. More importantly the efficiency of the algorithm permits a designer to evaluate optimal test times for any number of chains, and make appropriate tradeoffs between test time and pin count. The advantages of configuring optimal chains can be fully appreciated in circuits consisting of multiple kernels, with large disparities both in the test lengths and number of flip-flops involved in testing each kernel. Some additional issues that could be tackled include the following.

- The design of an efficient algorithm to configure multiple chains for the minimum shift policy. As pointed out in Section 5.7.3, this would constrain both the assignment and order of scan elements within each chain. The designer must therefore weigh the savings in test time against the possible effects on the routing area overhead.

- The routing constraints in configuring the scan chain can be more explicitly modeled in the problem by use of physical layout information. In this regard an approach based on genetic algorithms is currently being investigated [50].

- In our technique we assume that all scan flip-flops are controlled by a single clock. The methodology could be modified to allow for different clocking schemes that might restrict certain sets of flip-flops to lie in one or more scan chains.

10.1.3 A Reconfigurable Scan Chain

Reconfiguring a single scan chain provides a mechanism to alter the chain configuration for different sessions of the overlapped scheme. This led to the development of a novel methodology based on inserting multiplexers at specified positions in the scan chain. The savings in test time arise from bypassing those registers that are not frequently accessed in the test process. Both the flush and minimum shift policies provide a designer with different reconfiguration strategies to satisfy any constraints on test time and area overhead. Detailed analysis and optimization algorithms have been developed for both shift policies. The algorithms provide effective ways to tradeoff test time with the logic overheads of the bypass multiplexers and any routing constraints. The potential advantages of reconfiguring the scan chain were illustrated for a general class of circuits. Our experimental results clearly suggest that 1-3 multiplexers are sufficient in most cases to achieve the desired reductions in test time. Considering the significant savings in test time and the minimal area overhead of any inserted multiplexers, reconfiguration is clearly both a practical and effective design methodology. A number of possible avenues can be pursued in extending the basic ideas and concepts.

- The main emphasis of our work dealt with the reconfiguring of a single scan chain. A natural extension is to migrate the idea to configurations consisting of multiple scan chains.

- Exploit the existence of registers with hold modes in the chaining process. Their capability to retain data unchanged over multiple clock cycles can be exploited in reducing the shifting time. For example, if registers with hold modes are placed at either ends of a scan chain, data can be stored in these registers. This reduces the time to shift data from the scan-in pin and shift results to the scan-out pin. Note that such registers can also be exploited in the single and multiple chain methodologies.

- The primary goal in reconfiguring the scan chain was to reduce the shifting time in applying tests. An additional motivation for reconfiguring the chain is to reduce the power dissipation during the test process. If the registers bypassed by a multiplexer are placed in hold mode, the bit values in these registers will remain constant. As a result the switching activity during the test process can be reduced. In reconfiguring a scan chain, objective functions that take into account both the test time and switching activity should be considered. This is discussed in greater detail in the section on test scheduling.

10.2 Scan Chaining and Test Scheduling

For the previous three chain methodologies the overlapped scheme represented an optimal test scheme. As a result a division could be made between the scan chaining and test scheduling problems without compromising on the optimality of the final solutions. In general however the role played by the physical organization of the scan registers greatly influences the subsequent test scheduling phase. In this regard we studied two different concepts that highlight this close interaction between the two problems.

10.2.1 Asynchronous Multiple Scan Chains

In employing multiple scan chains, the use of a single mode signal constrains the operation of the chains in the test application process. All chains are forced to shift data simultaneously and apply this data to the circuit in the same clock cycle(s). By providing an additional mode signal, the bandwidth of test data to the different kernels can be increased. This forms the basis of the methodology described in Chapter 7 in which the idea of asynchronous multiple chains is introduced. The spatial parallelism in using multiple chains can be combined with the temporal parallelism in running the chains independently to provide savings in the overall test time. However to minimize the test time, we showed that a combined approach to scan chaining and test scheduling is required. An efficient test scheme referred to as the *two-phase* overlapped scheme was used to exploit the asynchronous operation of the chains. Based on this scheme, techniques to configure the chains were developed to minimize the test time. In a post-processing step, the test scheme was enhanced to further reduce the overall test time. Experimental results clearly illustrate the advantages of this new methodology. In particular savings are obtained both in the test time and pin count at the expense of an additional mode signal. Asynchronous chains will provide maximum benefits in designs consisting of multiple kernels with minimal sharing of scan flip-flops between the kernels. Further work can be considered in the following areas.

- Extending the approach to more than two mode signals and studying the advantages as compared to increasing the number of chains. In addition considering the case when the minimum shift policy is used in applying tests.

- Adopting a more unified approach that does not break up the overall problem of minimizing test time into two separate problems, i.e., fixing the test application scheme and then configuring the scan chains to minimize test time for this given scheme.

10.2.2 Test Scheduling for Scan Designs

The use of more refined techniques to partition a scan design gives rise to test resource conflicts. The importance of examining the test scheduling problem in the context of scan designs was discussed in Chapter 8. All previous approaches to the scheduling problem only exploit parallelism in testing kernels concurrently. However for scan designs an additional facet is provided by reducing the shifting time during the test process. As a result both the shift policy and the scan chain organization play an important role in minimizing the test time. The key differences in scheduling tests for BIST and scan designs were outlined in Section 8.3. The different scheduling techniques were classified based on the degree of parallelism and the shift policy. For these techniques a generalized formulation of the optimization problem in the form of an integer linear program was developed. The technique based on parallel testing with the minimum shift policy can provide significant savings in test time. For this technique, we illustrated the close interaction between the order of registers in the chain and the subsequent test scheduling phase. A two-phase algorithm that iterates between the chaining and scheduling phases was designed to minimize the overall test time. A number of issues can be addressed to further enhance the work.

- For designs with a large number of kernels and registers the computational costs of the two-phase algorithm might be excessive. More efficient algorithms need to be developed in such cases. The concepts must also be extended to deal with other chain configurations such as multiple chains and a reconfigurable scan chain.

- As pointed out in Section 8.7.2, the savings in test time through efficient chaining and scheduling are closely linked to the partitioning techniques and the manner in which tests are generated for the design. In particular the approach based on partitioning the test set exploits the structural topology of a kernel to reduce the test time. Further studies need to be done to establish its relative merits. It would be useful to identify characteristics of partitioned designs that will definitely lead to reduced test application times.

- An additional need for test scheduling arises from the power dissipated during the test process. Even if modules have been synthesized for low power dissipation, in the course of testing they are subject to input transitions that typically do not occur in normal functional mode. This problem is even more significant in scan testing since test patterns are serially shifted in through one or more chains. Extraneous switching activity at inputs to blocks currently not being tested will increase the total power dissipation. This in turn might lead to "hot spots" and result in total circuit failure. Test scheduling and chaining techniques need to be developed to reduce the total switching activity in the test process.

10.3 Integrating to a Scan Design System

Scan design has traditionally been viewed as a rigid and costly approach to incorporate testability. This in turn has led to a reluctance on the part of designers to fully integrate it within the design process. The `SIESTA` scan system provides a versatile design platform to produce optimized scan solutions that address both a designer's testability goals and constraints. The integration of the chaining and scheduling techniques complete the design framework as outlined in Section 9.2. We discussed the motivation for design estimators in guiding the design process and pruning the search tree. Lower and upper bounds on test time were developed for the different techniques. These bounds can be used by the designer to efficiently explore the design search space. The chaining and scheduling techniques greatly enhance the capabilities of the system. They provide a designer with wide range of options to minimize the test time while trading off design costs associated with routing and logic overhead, pin count and test control complexity. The capabilities of the `SIESTA` design system can be enhanced in a number of ways.

- The effectiveness of the design estimators in traversing and pruning the search space needs to be further investigated. Estimators also need to be developed for partial design states prior to the test generation phase.

- Integrating a physical design tool to layout scan designs and studying the effects of the different scan selection and chaining methodologies on the routing and area overheads.

- At present partitioning and test generation are done independently of the scan chaining and test scheduling tasks. Approaches to more closely link these different issues need to be studied. For example, test generation can be performed assuming a known scan chain configuration and possibly the schedule used in applying tests. This in turn might provide ways to further reduce the overall design costs.

- The underlying framework of `SIESTA` can be extended to deal with other issues. For example testing of the clock lines and clock logic, and asynchronous portions of the circuit. With the increasing complexity of logic synthesis systems, DFT schemes targeted only for stuck-at-faults may be insufficient to guarantee acceptable quality levels. `SIESTA` can be extended to consider more comprehensive fault models such as transistor stuck-open and delay faults.

Appendix

I - Dynamic Programming Algorithm for Flush Policy

Let the number of different weights for the N flip-flops be equal to n, the number of kernels in the design. Let S_i represent the set of all flip-flops with weight W_i and in accordance with Lemma 7, the flip-flops are placed in the order $S_n, S_{n-1}, \ldots, S_2, S_1$ from scan-in to scan-out. We use the notation $L(a,b)$ $(a \geq b)$ to denote the total number of flip-flops in the sets $S_a, \ldots, S_b$. The problem to be solved involves optimally inserting j bypass points between the boundaries of the sets of flip-flops $S_n, \ldots, S_1$, and is denoted by $\mathcal{P}(n,1,j)$. The test time of an optimal configuration is denoted by $\mathcal{T}(n,1,j)$. Similarly we use the notation $\mathcal{P}(a,b,\gamma)$ $(a \geq b)$ to denote the sub-problem of inserting γ bypass points between the sets $S_a, \ldots, S_b$.

Assume that in an optimal configuration, the first bypass point closest to the scan-in pin lies between flip-flops of weight W_k and W_{k-1}. The test time for the configuration is thus equal to

$$\mathcal{T}(n,1,j) = (W_n - W_{k-1})(L(n,k)+1) + \mathcal{T}(k-1,1,j-1)$$

The key point to note is that the sub-problem $\mathcal{P}(k-1,1,j-1)$ must be optimally solved. Similar to the multiple chain methodology, an optimal solution to the complete problem contains within it optimal solutions to sub-problems.

To recursively determine an optimal solution to $\mathcal{P}(n,1,j)$ in terms of optimal solutions to sub-problems, consider any sub-problem $\mathcal{P}(a,b,\gamma)$ where $n \geq a \geq b \geq 1$ and $\gamma \leq (a-b)$. The optimal value of $\mathcal{T}(a,b,\gamma)$ is recursively defined as follows. If $\gamma = 0$, $\mathcal{T}(a,b,0)$ is equal to $(W_a - W_{b-1})(L(a,b)+1)$. For the case of $\gamma > 0$, let us assume the first bypass point from the left lies between flip-flops in the set S_k and S_{k-1} $(a \geq k > b)$. The value of $\mathcal{T}(a,b,\gamma)$ will be equal to $\mathcal{T}(a,k,0)+\mathcal{T}(k-1,b,\gamma-1)$. However this recursive equation assumes a known value of k. In general there are $a-(b+\gamma-1)$ different feasible values for k, namely $a, a-1, \ldots, b+\gamma$. Since these are the only feasible values for placing the first bypass point, the optimal value

of $\mathcal{T}(a,b,\gamma)$ is obtained by choosing the minimum for these values of k. Hence in general,

$$\mathcal{T}(a,b,\gamma) = \begin{cases} (W_a - W_{b-1})(L(a,b)+1) & \text{if } \gamma = 0 \\ \\ min_{a \geq k \geq b+\gamma}\{\mathcal{T}(a,k,0) + \mathcal{T}(k-1,b,\gamma-1)\} & \text{if } \gamma > 0 \end{cases}$$

The final step in the algorithm is to employ a bottom-up approach to compute the optimal solutions for each sub-problem. The following procedure assumes that the weights of the flip-flops are stored in an array $W[0 \ldots n]$ where $W[0] = 0$. The cumulative lengths of the flip-flops are stored in an array $L[0 \ldots n]$ where $L[i]$ is equal to $L(i,1)$ and $L[0] = 0$. Hence the value of $L(a,b)$ is given by $L[a] - L[b-1]$. The optimal value of every sub-problem is stored in a three-dimensional array $T[1 \ldots n, 1 \ldots n, 0 \ldots j]$.

Generate-Optimal-Test-Time (Inputs: j, n)
begin
1. For $a = n$ down to 1
2. For $b = a$ down to 1
3. $T[a,b,0] = (W[a] - W[b-1])(L[a] - L[b-1] + 1)$
4. For $\gamma = 1$ to j
5. For $a = n$ down to $(1+\gamma)$
6. For $b = (a-\gamma)$ down to 1
7. $T[a,b,\gamma] = \infty$
8. For $k = a$ down to $b+\gamma$
9. *temp-var* $= T[a,k,0] + T[k-1,b,\gamma-1]$
10. if *temp-var* $< T[a,b,\gamma]$
11. $T[a,b,\gamma] =$ *temp-var*
12. Return $T[n,1,j]$

end

The above procedure fills up the array T in increasing order of the number of multiplexers from 0 to j. Hence for every sub-problem $\mathcal{P}(a,b,\gamma)$, determining its optimal value involves adding two previously evaluated terms for each value of k (line 9) and storing the minimum in the entry $T[a,b,\gamma]$ (line 11). The above

procedure only returns the optimal value of $\mathcal{T}(n,1,j)$; the optimal configuration can be obtained if the corresponding value of k is also stored in line 11. Finally an inspection of the nested loops in the procedure reveals that the worst-case time complexity of the algorithm is of order $O(jn^3)$.

II - Proof for Theorem 2

It is easy to see that $SCC \in NP$ since a nondeterministic algorithm need only guess at an ordering of the scan registers and check in polynomial time whether the minimum chain cycles are achieved for each session. We transform the known NP-complete problem *Subset Sum* to our problem [43].

Subset Sum
Given a finite set A with cardinality p, a "size" $s(a) \in Z^+$ for each $a \in A$, and a positive integer B, is there a subset $\hat{A} \subseteq A$ such that $\sum_{a \in \hat{A}} s(a) = B$.

The basic units of the Subset Sum instance are the individual elements $a \in A$ and let $\mathcal{L} = \sum_{a \in A} s(a)$. Let W_1 and W_2 be two positive integers such that $W_2 > W_1$. In transforming Subset Sum to SCC, let the local replacement for each $a \in A$ be a single register R_k with $l_k = s(a), d_k = W_1$ and $r_k = 0$. We augment the set S with three additional registers, (1) R_{p+1} with $l_{p+1} = 1, d_{p+1} = W_2$ and $r_{p+1} = W_1$, (2) R_{p+2} with $l_{p+2} = 1, d_{p+2} = W_1$ and $r_{p+2} = W_2$, and (3) R_{p+3} with $l_{p+3} = B, d_{p+3} = 0$ and $r_{p+3} = W_2$. Hence in the corresponding instance of SCC, the set S is equal to $\{R_1, R_2, \ldots, R_{p+1}, R_{p+2}, R_{p+3}\}$. The test scheme will consist of two sessions TS_1 and TS_2 with W_1 patterns applied in TS_1 and $(W_2 - W_1)$ patterns applied in TS_2. This instance can be constructed in polynomial time from the Subset Sum instance.

Let us evaluate the minimum chain cycle of each session for the above instance of SCC. The sum of the lengths of all drivers and receivers in TS_1 is equal to $\mathcal{L}+2$ and $B+2$, respectively. Hence the minimum chain cycle of the session is $\mathcal{L}+2$. Similarly in TS_2, since only registers R_{p+1}, R_{p+2} and R_{p+3} will be used, the minimum chain cycle of the session is equal to the maximum of l_{p+1} and $(l_{p+2} + l_{p+3})$, i.e., $B+1$. To achieve this minimum chain cycle in TS_2 for any ordering of the registers, R_{p+2} and R_{p+3} must occupy in order the last two slots in the chain.

Consider register R_{p+1}. Since it is used as a driver-receiver in TS_1 and as a driver in TS_2, its position in the chain must be such that (1) the number of clock cycles to shift a bit into it from the scan-in pin is less than or equal to $B+1$, and (2) the number of clock cycles to shift a bit from it to the scan-out pin is less than or equal to $\mathcal{L}+2$. These constraints on R_{p+1} translate to dividing the remaining p registers into two subsets such that the sum of the lengths of registers in the sets are equal to B and $\mathcal{L}-B$, respectively. To achieve the minimum chain cycles in both sessions, the $(p+3)$ registers need to be placed in the following order starting from the scan-in pin; all registers in the subset of length B, R_{p+1}, all registers in the subset of length $\mathcal{L}-B, R_{p+2}, R_{p+3}$.

Thus the desired subset $\hat{A}$ exists for the instance of Subset Sum if and only if a feasible ordering of registers exists for the corresponding instance of SCC. □

Reference List

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital Systems Testing and Testable Design.* Computer Science Press, New York, N.Y., 1990.

[2] E.B. Eichelberger and T.W. Williams. A logic design structure for testability. In *Proc., 14th Design Automation Conf.*, pages 462–468, June 1977.

[3] S. Funatsu, N. Wakatsuki, and A. Yamada. Designing digital circuits with easily testable considerations. In *Proc., IEEE Semiconductor Test Conf.*, pages 98–102, 1978.

[4] B. Konemann, J. Mucha, and G. Zwiehoff. Built-in logic block observation techniques. In *Proc., IEEE Int'l Test Conf.*, pages 37–41, October 1979.

[5] K.-T. Cheng and V.D. Agrawal. A partial scan method for sequential circuits with feedback. *IEEE Trans. on Computers*, 39(4):544–548, April 1990.

[6] R. Gupta, R. Gupta, and M.A. Breuer. The BALLAST methodology for structured partial scan design. *IEEE Trans. on Computers*, 39(4):538–543, April 1990.

[7] M. Abramovici, J. Kulikowski, and R.K. Roy. The best flip-flops to scan. In *Proc., Int'l. Test Conf.*, pages 166–173, October 1991.

[8] V. Chickermane and J.H. Patel. An optimization based approach to the partial scan design problem. In *Proc., Int'l. Test Conf.*, pages 377–386, September 1990.

[9] H. Ando. Testing VLSI with random access scan. In *Proc., COMPCON*, pages 50–52, 1980.

[10] B. Vinnakota and N.K. Jha. Synthesis of sequential circuits for parallel scan. In *Proc., European Conf. on Design Automation*, pages 366–370, March 1992.

[11] K.-T. Cheng, S. Devadas, and K. Keutzer. A partial enhanced scan approach to robust delay-fault test generation. In *Proc., Int'l. Test Conf.*, pages 403–410, October 1991.

[12] B.I. Dervisoglu and G.E. Stong. Design for testability: using scanpath techniques for path-delay test and measurement. In *Proc., Int'l. Test Conf.*, pages 365–374, October 1991.

[13] B. Konemann. LFSR-coded test patterns for scan designs. In *Proc., European Test Conf.*, pages 237–242, 1991.

[14] G.L. Craig, C.R. Kime, and K.K. Saluja. Test scheduling and control for VLSI built-in self-test. *IEEE Trans. on Computers*, 37(9):1099–1109, September 1988.

[15] C. H. Chen. Graph partitioning for concurrent test scheduling in VLSI circuit. In *Proc., 28th Design Automation Conf.*, pages 287–290, June 1991.

[16] P. Goel and B.C. Rosales. Test generation and dynamic compaction of tests. In *Proc., Int'l. Test Conf.*, pages 189–192, October 1979.

[17] S.B. Akers, C. Joseph, and B. Krishnamurthy. On the role of independent fault sets in the generation of minimal test sets. In *Proc., Int'l. Test Conf.*, pages 1100–1107, September 1987.

[18] I. Pomeranz, L.N. Reddy, and S.M. Reddy. Compactest: A method to generate compact test sets for combinational circuits. In *Proc., Int'l. Test Conf.*, pages 194–203, October 1991.

[19] S. Chakravarty and S.S. Ravi. Computing optimal test sequences from complete test sets for stuck-open faults in CMOS circuits. *IEEE Trans. on Computer-Aided Design*, 9(3):329–331, March 1990.

[20] T.M. Niermann, R.K. Roy, J.H. Patel, and J.A. Abraham. Test compaction for sequential circuits. *IEEE Trans. on Computer-Aided Design*, 11(2):260–267, February 1992.

[21] D.K. Pradhan and J. Saxena. A design for testability scheme to reduce test application time in full scan. In *Proc., VLSI Test Symp.*, pages 55–60, April 1992.

[22] S.Y. Lee and K.K. Saluja. An algorithm to reduce test application time in full scan designs. In *Proc., Int'l Conf. on Computer-Aided Design*, pages 17–20, November 1992.

[23] H. Fujiwara and A. Yamamoto. Parity-scan design to reduce the cost of test application. In *Proc., Int'l Test Conf.*, pages 283–292, September 1992.

[24] P. Chen, B. Liu, and J. Wang. Overall consideration of scan design and test generation. In *Proc., Int'l Conf. on Computer-Aided Design*, pages 9–12, November 1992.

[25] W. Lai, C. Kung, and C. Lin. Test time reduction in scan designed circuits. In *Proc., Euro. Design Automation Conf.*, pages 489–493, October 1993.

[26] S. Bhawmik et al. Threading of multiple scan paths in a VLSI circuit. In *Proc., Int'l. Test Conf.*, pages 735–743, September 1988.

[27] P.P. Fasang, J.P. Shen, M.A. Schuette, and W.A. Gwaltney. Automated design for testability of semicustom integrated circuits. In *Proc., Int'l Test Conf.*, pages 558–564, November 1985.

[28] S. Lee and K.G. Shin. Design for test using partial parallel scan. *IEEE Trans. on Computer-Aided Design*, 9(2):203–211, February 1990.

[29] R. Gupta and M.A. Breuer. Ordering storage elements in a single scan chain. In *Proc., Int'l Conf. on Computer-Aided Design*, pages 408–411, November 1991.

[30] IEEE Standard 1149.1-1990. *IEEE Standard Test Access Port and Boundary Scan Architecture.* IEEE Standards Board, New York, N.Y., 1990.

[31] Y. Choi and T. Jung. Configuration of a boundary scan chain for optimal testing of clusters of non boundary scan devices. In *Proc., Int'l Conf. on Computer-Aided Design*, pages 13–16, November 1992.

[32] S.P. Morley and R.A. Marlett. Selectable length partial scan: a method to reduce vector length. In *Proc., Int'l Test Conf.*, pages 385–392, November 1991.

[33] W.B. Jone, C.A. Papachristou, and M. Pereira. A scheme for overlaying concurrent testing of VLSI circuits. In *Proc., 26th Design Automation Conf.*, pages 531–536, June 1989.

[34] M.S. Abadir and M.A. Breuer. Test schedules for VLSI circuits having built-in test hardware. *IEEE Trans. on Computers*, 35:361–367, April 1986.

[35] R. Gupta. *Advanced Serial Scan Design for Testability.* Ph.D. thesis, Univ. of Southern California, Dept. of Electrical Engr. – Systems, 1991. CEng. Technical Report 91-10.

[36] S. Oostdijk, F. Beenker, and L. Thijssen. A model for test-time reduction of scan testable circuits. In *Proc., European Test Conf.*, pages 243–252, 1991.

[37] R. Gupta, R. Srinivasan, and M.A. Breuer. CRETE: Hierarchical reorganization of circuits for DFT and BIST. *IEEE Design & Test*, pages 49–57, September 1991.

[38] F. Beenker et al. Implementing macro test in silicon compiler design. *IEEE Design & Test*, pages 41–51, April 1990.

[39] E.J. McCluskey and S. Bozorgui-Nesbat. Design for autonomous test. *IEEE Trans. on Computers*, C-30(11):866–875, November 1981.

[40] P.S. Bottoroff et al. Test generation for large logic networks. In *Proc., 14th Ann. Design Automation Conf.*, pages 479–485, September 1977.

[41] I. Parulkar, C.A. Njinda, and M.A. Breuer. Extraction of a high-level structural representation from circuit descriptions with applications to DFT/BIST. In *Proc., 31st Design Automation Conf.*, pages 345–350, June 1994.

[42] M. Abramovici, K.B. Rajan, and D.T. Miller. Freeze!: A new approach for testing sequential circuits. In *Proc., 29th Design Automation Conf.*, pages 22–25, June 1992.

[43] M.R. Garey and D.S. Johnson. *Computers and Intractability.* W. H. Freeman and Co., New York, NY, 1979.

[44] K.R. Baker. *Introduction to Sequencing and Scheduling.* J.Wiley & Sons, Inc., 1974.

[45] F. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT Press, Cambridge, Massachusetts, 1990.

[46] E. L. Lawler. Optimal sequencing of a single machine subject to precedence constraints. *Management Science*, 19(5):544–546, January 1973.

[47] R. Jain, K. Kucukcakar, M. Mlinar, and A. Parker. Experience with the ADAM synthesis system. In *Proc., 26th Design Automation Conf.*, pages 56–61, July 1989.

[48] R. Gupta, W.H. Cheng, R. Gupta, I. Hardonag, and M.A. Breuer. An object-oriented VLSI CAD framework: A case study in rapid prototyping. *IEEE Computer*, pages 28–37, May 1989.

[49] D. Mukherjee. *An Integrated Test Controller Synthesis System.* Ph.D. thesis, Univ. of Southern California, Dept. of Electrical Engr. – Systems, August 1994.

[50] R. K. Chennagiri. Private Communication. March 1994.