FUNCTIONAL SCAN DESIGN AT RTL

FUNCTIONAL SCAN DESIGN AT RTL

BY

HO FAI KO, B. ENG. & MGT. (COMPUTER)

JUNE 2004

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

AND THE COMMITTEE ON GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

MASTER OF APPLIED SCIENCE (2004) McMaster University
(Electrical and Computer Engineering) Hamilton, Ontario

TITLE:	Functional Scan Design at RTL
AUTHOR:	Ho Fai Ko, B. Eng. & Mgt. (Computer)
SUPERVISOR:	Dr. Nicola Nicolici
NUMBER OF PAGES:	ix, 94

Abstract

Scan chain design is an essential step in the manufacturing test flow of digital integrated circuits. Its main objective is to generate a set of shift register-like structures (i.e., scan chains), which, in the test mode of operation, will provide controllability and observability of all the internal flip-flops. The number of scan chains, the partitioning of flip-flops to different scan chains and the order of flip-flops within each scan chain are three important factors that may impact the test cost and test quality parameters, such as performance degradation, area overhead, test application time, volume of test data and delay fault coverage.

In this thesis we investigate a novel approach to design scan chains at the register transfer level (RTL) of design abstraction. By embedding the test data transfers in the RTL description, we constrain the logic synthesis tool to share the functional and test logic. As a result, the functional scan chains may incur lower performance degradation and area overhead when compared to the dedicated scan chains generated at lower levels of design abstraction. In addition, by exploiting the information available in the control/data flow graph (CDFG) extracted from the RTL description, we consciously partition and order flip-flops in each scan chain to address test application time, volume of test data and delay fault coverage. Furthermore, because the above tasks operate on CDFGs, which contain substantially less data than the logic networks or physical layouts, the test development time for generating functional scan chains has an insignificant impact on the design cycle. A new RTL test scan insertion tool has been developed and interfaced to third party electronic design tools to generate experimental results that demonstrate the benefits of the proposed approach.

Acknowledgments

I give a sincere gratitude to the people who do engineering with pure, unselfish and honest passion as they are the people who made me grow and appreciate the world in the way I see.

I am deeply indebted to my supervisor, Nicola Nicolici, from whom I learned a lot as he is always looking after me and brings me laughters and tears during the project. My colleagues in the Computer-Aided Design and Test (CADT) Research Group, Qiang Xu, Baihong Fang, David Lemstra and Adam Kinsman have always assisted me in so many ways. I would like to thank them for their great company when disasters strike. I also would like to express my gratitude to the graduate students, faculty, administrative and technical members in Department of Electrical and Computer Engineering at McMaster University for their assistance during my study and research. Moreover, I wish to acknowledge Micronet for their financial support, and Drs. Capson and Szymanski for their suggestions during my defence.

Members of my family and many more than I can include here, have loved me far beyond what I can ever return. I will start by thanking my parents for their love and continuous support. I would like to thank my girlfriend for her continuous trust and confidence in me. I also would like to thank my brother and sister, who bring sunshine whenever I feel blue. Finally, I would not forget every single one of my friends, who have always believed in me. I sincerely appreciate all the love, support and trust, which make my work possible.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Escalating size and complexity of very large scale integrated (VLSI) circuits make verification and test a bottleneck in the design flow [7]. Digital VLSI circuits use scan chains as a method to control and observe the internal state elements [7]. The aim of the work described in this thesis is to give a new perspective on scan chain design at the register-transfer level of design abstraction. To better illustrate how the proposed method can be beneficial in the development of VLSI circuits, it is essential to understand the design flow and the state-of-the-art practice for hardware specification and implementation. The design flow will be outlined in Section 1.1 and the test flow will be introduced in Section 1.2. To quantify the effectiveness of different test methodologies, the concepts of test cost and test quality will be discussed in Sections 1.3 and 1.4. Finally, the organization of this thesis is provided in Section 1.5.

1.1 Circuit Models And Design Flow

The design flow of VLSI circuits and systems starts with a system specification as shown in Figure 1.1. The specification is then translated to algorithmic descriptions, which are used for architectural planning. At this stage, the design flow will be divided into two directions: *software development* and *hardware development*. In this thesis, we will focus only on the hardware development process.

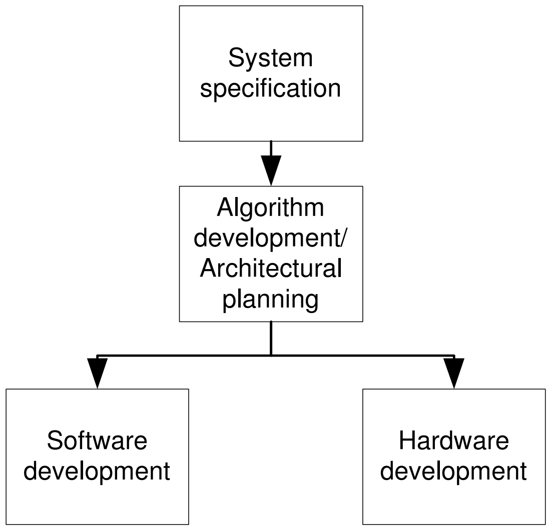


Figure 1.1: System specification refinement and hardware/software partitioning

Circuit models are developed to reduce size and complexity of the design specification for hardware development. A model of a circuit gives the relevant features with varying amounts of details at different levels of design abstraction. A less detailed model at a higher level of design abstraction is *synthesized* into circuit model at a lower level of design abstraction with refined details [25]. Models can be classified in terms of *views* and *levels of abstraction*. The different views are categorized as: *behavioral*, *structural* and *physical*. They are represented by the three axes in the Y-chart in Figure 1.2. This Y-chart representation was first proposed by Gajski [15].

- *Behavioral view*: The behavioral view represents the design as a black box and describes its outputs in terms of its inputs and time. The structure and physical information of the black box is unknown in this view.

- *Structural view*: The structural information of the black boxes from the behavioral view are given by the structural view, where each black box is represented as a set of components and the interconnects between them.

- *Physical view*: The physical view adds dimensionality to the structure. In this view, the size and position of each black box, as well as the port and connection in the final layout are specified.

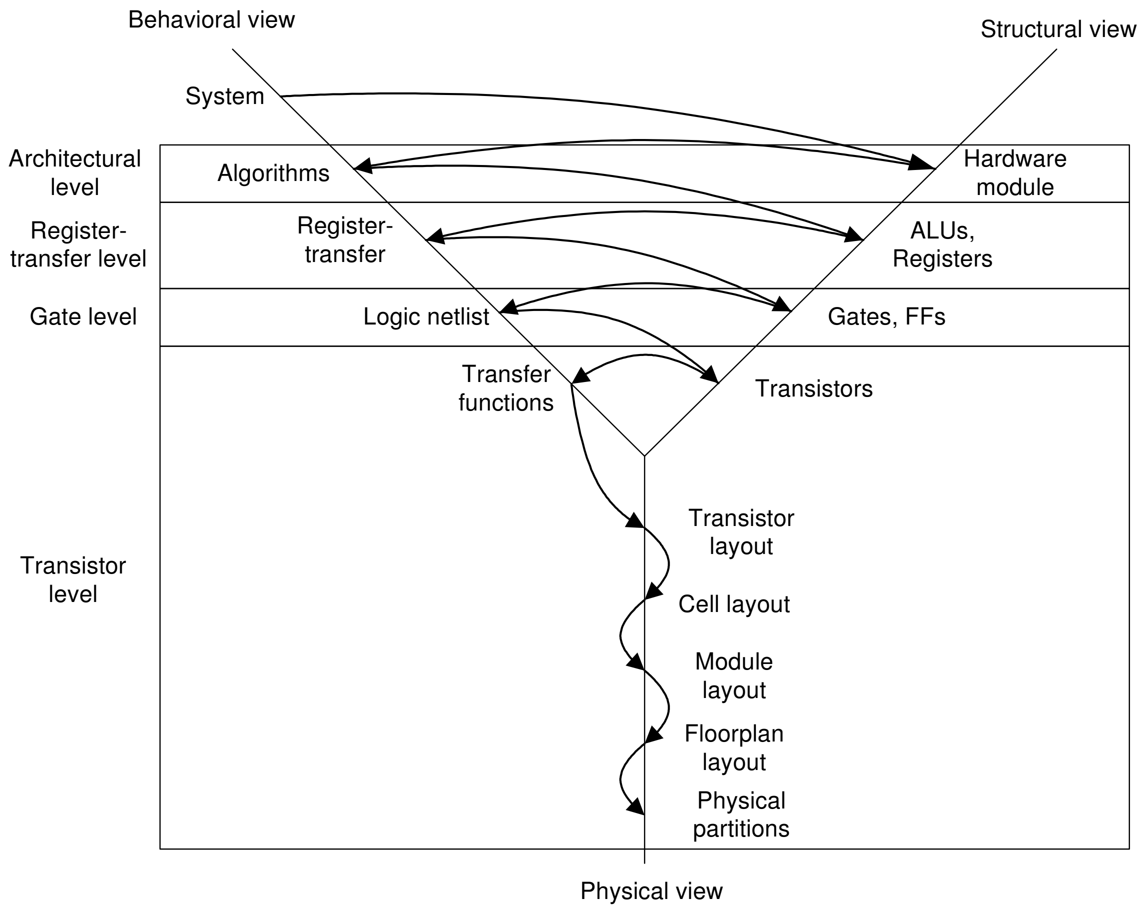


Figure 1.2: Y-chart representation of circuit model [15]

The different views of a circuit model describe different types of information of each component in a design. For example, when modeling a Fast Fourier Transform (FFT) circuit, the behavioral view specifies the mathematical relationship between the input and the expected output signals of the design. The structural view gives the choice of components (e.g., butterfly blocks) and their interconnects for the implementation of the circuit. Lastly, the physical view details the physical information (e.g., dimension, location) of every component that is used in the structural view. By synthesizing the circuit specifications from a higher level to a lower level of abstraction, these components and their interconnects are better specified. The four main levels of design abstraction are: *architectural*, *register-transfer*, *gate* and *transistor*.

- *Architectural level*: The specification described at the architectural design abstraction level contains the main components that are used for the design. These components usually include processors, memories and buses. However, they are treated as black boxes and only their behaviors are specified. At this level of abstraction, the actual implementation of the components is unknown. In other words, the architectural design abstraction level only describes a circuit as a set of operations, such as data computations or transfers.

- *Register-transfer level*: At the register-transfer level (RTL), a digital circuit evaluates a set of transfer functions, which are described by registers and functional units, such as arithmetic-logic units (ALUs). The RTL description thus gives better understanding on the actual hardware implementation of the components described at the architectural design abstraction level.

- *Gate level*: At the gate level of design abstraction, the transfer functions of a circuit at the RTL description are transformed into logic equations. Theses equations are evaluated by a set of primitives that are obtained from a targeted technology library. These primitives typically include various logic gates (e.g., AND, OR, XOR gates), and different types of flip-flops (FFs).

- *Transistor level*: The transistor design abstraction level describes circuits using transistors, which resemble the exact implementation of the primitives from the gate level description on the silicon die. Hence, this abstraction level contains a huge amount of information as size and complexity of designs increase.

Using the FFT circuit as an example, information about the behavior of different components (e.g., various types of butterfly blocks) and their interconnects can be found in the architectural description. At the RTL, data transfers within, as well as among the butterfly blocks are specified by a set of transfer functions. These functions will then be synthesized into the gate level netlist which gives the implementation of each function in terms of logic gates. Finally, the transistor design abstraction level indicates how the gates are translated into electronic devices.

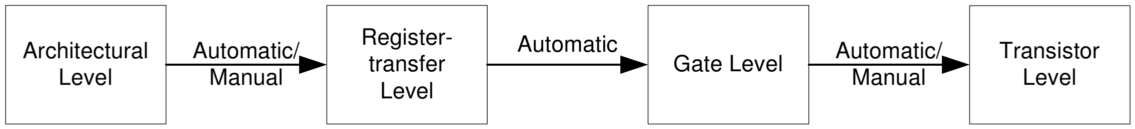


Figure 1.3: VLSI design flow

The process of transforming the circuit model from a less detailed abstraction level (e.g., architectural level) to a more detailed abstraction level (e.g., register-transfer level) is called *synthesis*. The subsequent steps to synthesize a circuit description from a higher level to a lower level of design abstraction is the *Design Flow*, which is illustrated in Figure 1.3. As shown in the figure, the architectural level is the highest level of design abstraction. It is used primarily for translating the architectural specification so that the design can be simulated. Although a number of automatic algorithms have been proposed in the recent years, automatic architectural synthesis, which synthesizes a circuit model from the architectural abstraction level to the RTL, has yet to reach its maturity. *State-of-the-art practice starts the design flow from the RTL description.* At the RTL, a circuit is described as a set of registers and transfer functions resembling the flow of data between the registers using *hardware description languages* (HDLs) like VHDL and Verilog HDL. The registers are implemented directly as flip-flops (FFs), while the transfer functions are implemented as blocks of combinational logic. This direct one-to-one relationship of registers and transfer functions in the design helps reduce the complexity of transforming the RTL design into gate level structural netlist during the automated synthesis process [4]. Finally, the structural netlist at the gate level of design abstraction is transformed into physical layout at the transistor level automatically, by incorporating manually generated standard cells.

In addition to using automated tools for circuit synthesis and optimization, another consideration in VLSI design is to improve the manufacturing yield, by accounting for circuit testability during the design process. This is accomplished by inserting various design-for-test (DFT) structures into the circuit while maintaining the original functionality of the design. These DFT structures and the test flow will be discussed in the following section.

1.2 Test Flow

Microelectronic circuits are tested after manufacturing to screen fabrication errors. Thus *manufacturing test* is the verification of circuit fabrication [25].

1.2.1 Manufacturing Test And Fault Models

Fabrication anomalies of integrated circuits in the manufacturing process may cause some circuits to behave erroneously. *Manufacturing test* helps to detect the physical defects that lead to faulty behaviors of the fabricated circuits. These defects can be detected by *parametric tests for chip pins* and *tests for functional blocks* [7]. Parametric tests include *DC tests* and *AC tests*. DC parametric tests are used for detecting shorts, opens, maximum current, leakage, output drive current and threshold levels. AC parametric tests are for testing, setup and hold times, functional speed, access time, refresh and pause time, and rise and fall time. These tests are usually technology-dependent and can be performed without any regard to the chip functionality. On the other hand, the tests for functional blocks check for the proper operation of a manufactured circuit by testing the internal chip nodes using input vectors. The corresponding circuit responses are compared to the expected responses for pass/fail analysis. These technology independent tests for functional blocks can be further divided into *functional tests* and *structural tests*.

- *Functional tests*

  Functional tests verify the functionality of each component in the circuit. To completely exercise the circuit functions, a complete set of test patterns is needed. For a circuit with n inputs, the number of input vectors will be 2^n. For instance, a 64-bit ripple-carry adder will have 2^{129} input vectors. To apply the complete test set to the *circuit-under-test* (CUT) using an *automatic test equipment* (ATE), it would take 2.158×10^{22} years, assuming that the tester and circuit can operate at 1 GHz [7]. Due to the exhaustive nature of complete functional tests, testing time is prohibitively large for logic blocks, which makes them infeasible for testing complex digital integrated circuits.

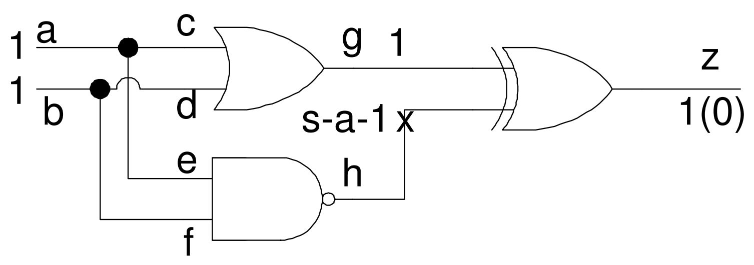


Figure 1.4: Example of single stuck-at fault

- *Structural tests*

  On the other hand, structural tests depend on the netlist structure of a design. Depending on the logic and timing behavior of electrical defects, different *fault models* are introduced to allow automatic algorithms to be developed for test generation, test application and test evaluation. The most commonly used fault model is the *single stuck-at fault model.* It assumes a single line of the logic network to be stuck at a logic 0 (s-a-0) or logic 1 (s-a-1). An illustration of the stuck-at model is shown in Figure 1.4. In this example, the targeted fault is (s-a-1) at node h, which can be sensitized by the input vector $\{1,\ 1\}$ from inputs $\{a,\ b\}$. The correct response for this circuit at output z is 1. The faulty response is therefore 0. When using the *single stuck-at fault model* for the 64-bit ripple-carry adder, only 1728 stuck-at faults would need to be excited with 1728 test patterns in the worst case scenario [7]. Another fault model that is gaining attention is the *delay fault model*, which will be detailed in Section 1.4.

1.2.2 Test Pattern Generation

After a fault model has been selected for structural test, the next step is to generate a set of test patterns. The outcome of test generation is a set of input vectors, which are applied to the circuit inputs to sensitize targeted faults, and a set of expected output responses, which are used for comparison with the actual circuit responses. Test generation is done by *automatic test pattern generation* (ATPG). There are two types of ATPG algorithms. They are *combinational ATPG* and *sequential ATPG.*

- *Combinational ATPG*

  Combinational ATPG is one of the most important steps in the test flow. It is proven to be NP-Complete [7], which makes it prohibitively expensive in terms of CPU run time and volume of test data when applied to complex VLSI circuits [11]. Because of its complexity, many heuristics have been investigated [7] and all of them are based on four main operations: *excitation*, *sensitization*, *justification*, and *implication*. To generate a pattern for a stuck-at fault on a line (or wire), the fault is first excited, the response would then be sensitized to an observation point (e.g. primary output), and the logic values required on the input lines are justified. At the same time, the implications of logic values on other gates will be determined. Figure 1.4 can be used to illustrate the four operations. To excite the stuck-at-1 fault at node h, the value of that wire has to be set to 0. The effect of the fault is sensitized to the primary output z. In order to excite the fault at node h with a value of 0, the values of the primary inputs $\{a,\ b\}$ are justified to be $\{1,\ 1\}$. This input combination implies the value of node g to be 1. By iteratively applying the four operations to all the faults in the circuit, a complete set of test patterns can be generated.

- *Sequential ATPG*

  If the internal state elements are not controllable, *sequential ATPG* is needed. There are several reasons why test pattern generation for sequential circuits is more difficult than for combinational circuits. First of all, the output responses of the circuit depend not only on the input patterns, but also on the internal states of the circuit. These internal states may be synchronous or asynchronous. Also, sensitizing a fault to a primary output requires the circuit to be driven to a known state. This sensitization process alone might require more than one pattern, and the order in which the test patterns are applied is critical. Furthermore, propagating the effect of the fault to an observable output may take several clock cycles. In addition, multiple clock domains further complicate test pattern generation for sequential circuits, because the relations between clock domains must be followed to avoid any unpredictable behavior [26].

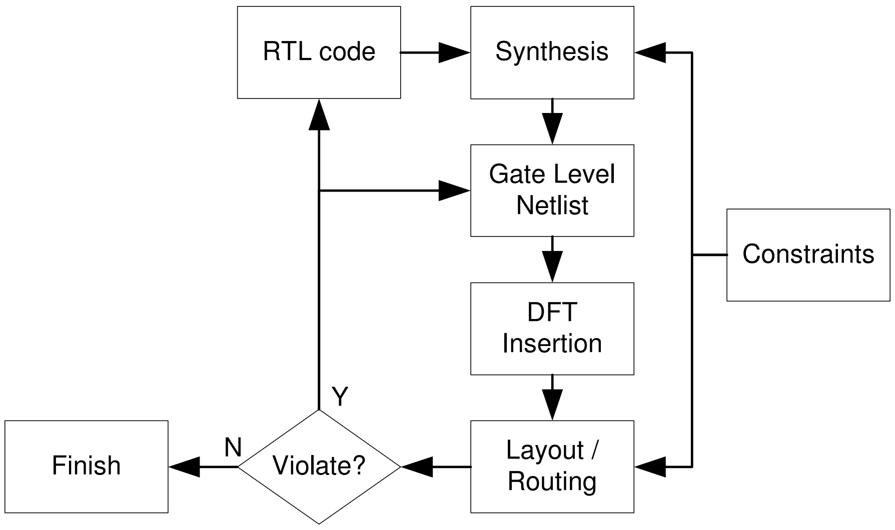


Figure 1.5: Design flow with DFT

The difficulty in controlling and observing internal states makes sequential ATPG inapplicable to large circuits. The enhancement of circuit testability will allow ATPG to generate test patterns for complex VLSI designs in a more efficient way (i.e., tractable given the resources at hand). Thus, techniques to improve the controllability and observability of a design are needed.

Traditionally, circuit testability was considered as an after thought. Efforts were only done at the end of the design cycle. However, this approach often led to low fault coverage or rising production costs due to the unforseen increase in cycle time as size and complexity of VLSI circuits grow. As a result, *design-for-test* (DFT), was introduced to account for testability within the design cycle [4]. Figure 1.5 shows the modified design flow when considering testability within the design cycle. In this scenario, DFT structures are inserted after the structural netlist at the gate design abstraction level is obtained from logic synthesis tools. Although considering testability within the design cycle may increase the development cost, it can be offset by the decrease in production cost and improved manufacturing yield [46]. The common DFT structures that are used to enhance testability of VLSI circuits will be detailed in the following section.

1.2.3 Types Of Design-For-Test Structures

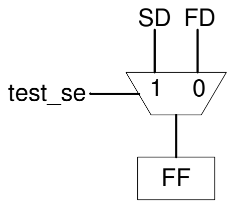
The DFT structures that are considered here are *scan design* and *built-in self-test*.

Scan Design

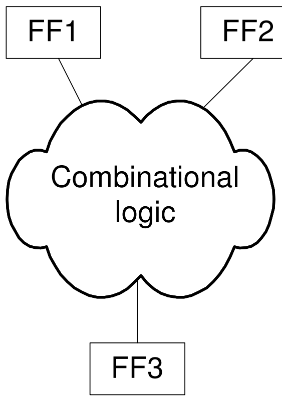
It is common for VLSI designs today to have internal state signals which cannot be easily controlled from primary inputs or observed at primary outputs. This prohibits sequential ATPG to be tractable to complex VLSI designs, which may contain thousands (or even millions) of state elements. In order to enhance controllability and observability of large sequential circuits, the *scan method* is used to transform sequential circuits into combinational circuits from the test generation standpoint. Hence, the more tractable combinational ATPG algorithms can be used [26].

The scan method attempts to control and observe the sequential elements (i.e., FFs) inside a circuit by inserting a test mode such that, when the circuit is in this mode, all the FFs are connected together to form one or multiple shift registers. These shift registers, also known as *scan chains* (SCs), are connected to primary inputs and primary outputs, which are called *scan inputs* (SIs) and *scan outputs* (SOs) respectively. By serially shifting arbitrary values into the SCs from SIs (called scan in), all the internal FFs can be set to desired states. Similarly, the internal FFs can be observed by scanning out their values in the SCs through SOs. As a result, the circuit becomes fully controllable and observable [7]. The complete controllability and observability of a scan design eliminates the need for sequential ATPG. Instead, the scan-flip-flops in the circuit are treated as *pseudo-primary-inputs* and *pseudo-primary-outputs*. Thus, from the ATPG standpoint, the sequential circuit is transformed into a combinational circuit.

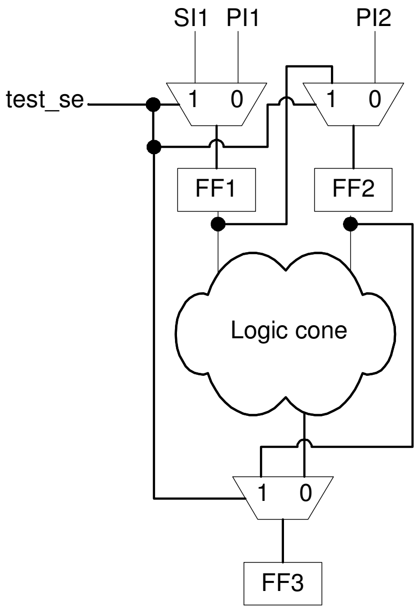
In order to construct the SCs, original FFs in the design will have to be replaced with special *scan-flip-flops* (SFFs). A SFF has an additional 2-input multiplexor (MUX) that is connected to the input of the FF. The hardware structure of an SFF is illustrated in Figure 1.6(a). In the normal functional mode, the SFF reads the value from the functional data input of the MUX, thus retaining the original functionally of the design. Conversely, in the test mode, the SFF takes its value from the scan data



(a) SFF



(b) Original circuit



(c) Scan circuit

Figure 1.6: Scan design

(SD) input of the MUX, which is connected to another FF in the SC. Figure 1.6(b) shows a circuit without scan. In this circuit, the input FFs $\{FF1, FF2\}$ connect to a combinational logic block, which feeds the output FF $\{FF3\}$. By replacing the three FFs in Figure 1.6(b) with SFFs, the scan design is shown in Figure 1.6(c). In this scan design, the signal *test_se* indicates whether the circuit is operating in the normal mode or in the test mode. In the normal mode, the scan circuit has the same functionality as the original circuit. In the test mode, The FFs are connected to form an SC with the following order: $\{FF1, FF2, FF3\}$. This SC can be used to shift in test vectors through the scan input $SI1$.

Built-in Self-test

After test patterns are generated from ATPG with the aid of the scan method, the next step is test application. The constrained test access to the I/O pins limits the

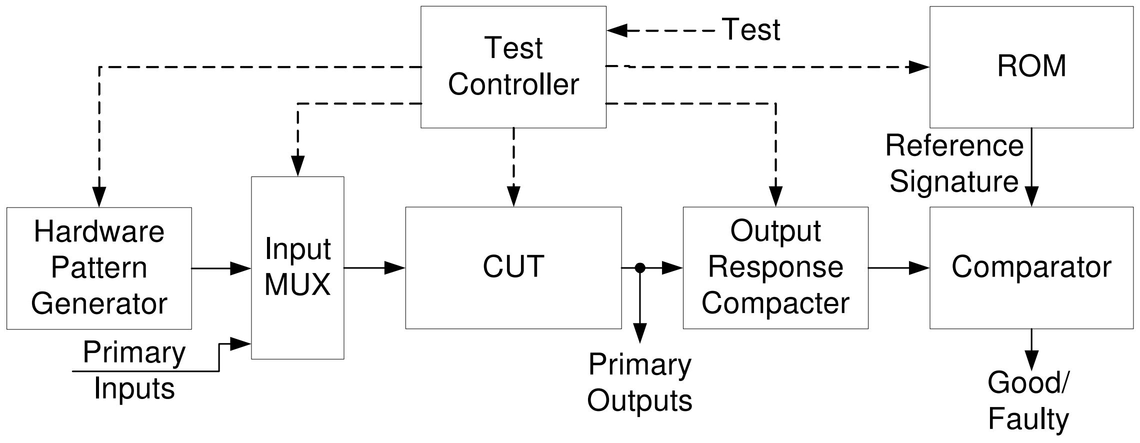
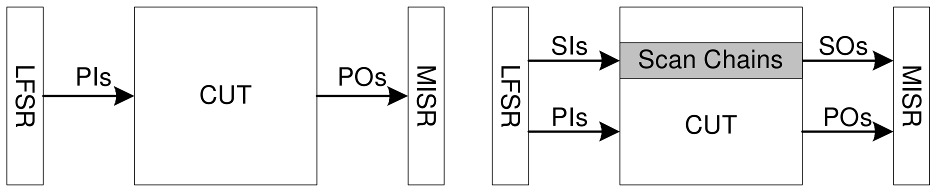


Figure 1.7: General BIST architecture [7]

number of internal scan chains that can be directly driven by the tester. Because the longest internal scan chain determines the test application time, for circuits with many flip-flops and a low number of test pins, the time the circuit spends on the tester may be prohibitively large. Further, the huge volume of test data introduces another problem when external testers are employed. This is because storing the test data will require either reloading of buffers or the use of expensive testers with gigantic buffers. Hence, a new approach for test application called *built-in self-test* (BIST) has emerged [33]. Instead of feeding test patterns and observing circuit responses of the CUT from an external tester, on-chip test pattern generators and response analyzers controlled by a test controller are used in BIST. Figure 1.7 illustrates a general BIST architecture. There are two types of BIST schemes for applying test patterns to the CUT. They are *test-per-clock* and *test-per-scan*.

- *Test-per-clock*

  The architecture for the test-per-clock BIST system is shown in Figure 1.8(a). In this architecture, the PIs of the CUT are driven by the *linear feedback shift register* (LFSR), and the POs are connected to the *multiple input signature register* (MISR) to generate a response signature for the circuit. By generating and applying a new test pattern to the CUT continuously from the LFSR, a new set of faults are tested every clock period [7]. However, the BIST controller for



(a) Test-per-clock scheme (b) Test-per-scan scheme

Figure 1.8: BIST test application schemes

the test-per-clock BIST system can be quite complex, which may result in high area overhead. Moreover, the use of large test registers (e.g., LFSRs, MISRs) can significantly impact both area and performance of the original design.

- *Test-per-scan*

  The test-per-scan BIST system is shown in Figure 1.8(b). In this system, the concept of test-per-clock is combined with the scan method. As a result, a two-step process is required for each new set of faults. In addition to the single clock period for conducting the test, a series of shifts for the SC are needed to initialize the circuit and read out all the test results. Therefore, the test-per-scan system will take several clock cycles per pattern. However, the test control and test hardware is non-intrusive since it reuses the available scan structure for test application. This leads to lower area and performance overhead when compared with the test-per-clock system. In addition, it fits easily into any designs which already have scan structures in place.

In addition to the above, details of other DFT structures can be found in [7, 26]. In this thesis, we restrict the discussion to the scan method due to its applicability to large circuits and different fault models, its suitability for both BIST and ATE-based test application and its ease of integration in the VLSI design flow. In order to evaluate the effectiveness of various scan architectures, different parameters are introduced to quantify the benefits or drawbacks of each architecture. These parameters will be discussed in the next section.

1.3 Test Cost

Scan structures can significantly improve the testability of complex VLSI designs. However, the enhancement does not come for free. In order to quantify the added *test cost* of different scan structures, a number of parameters are used. The parameters that are considered in this thesis are: *performance penalty*, *area overhead*, *test application time*, *volume of test data* and *test development time*.

- *Performance penalty*

  Performance penalty is mainly caused by the extra hardware from the inserted scan structures. The replacement of every FF with SFF shown in Figure 1.6(a) brings an additional MUX to the circuit. Each additional MUX located on the critical path of the design adds performance penalty equivalent to two gate-delays. It is obvious that as the number of FFs in the critical path increases, the performance penalty grows proportionally. Moreover, the extra wires used for the creation of scan paths raise the capacitive loading on the FF outputs, which may also increase the propagation delays. In general, the propagation delays in scan design increase around 5% [7].

- *Area overhead*

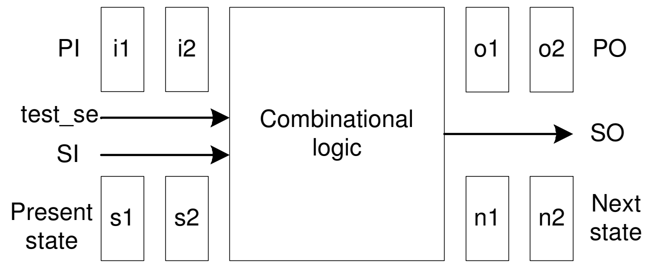
  It is obvious that the insertion of scan structures for testability improvement introduces area overhead to the design [7]. In the case of the scan method, the increase in silicon area is due to the complexity of the scan device, FF, or latch. For instance, the gate overhead for the SFF in Figure 1.6(a) will be the input multiplexor, which is equivalent to four logic gates. In addition to the gate overhead, the scan method may require a significant amount of routing, which can impact the chip area. In a scan design, the test enable (test_se) signal is routed to all FFs, and the output of each FF is routed to the SD input of the subsequent FFs in an SC. To reduce area occupied by the interconnect wires from the scan design, one can re-order the sequential elements in the chain. However, this effort to diminish routing overhead can only be performed at the layout generation or routing step in the design flow.

- *Test application time*

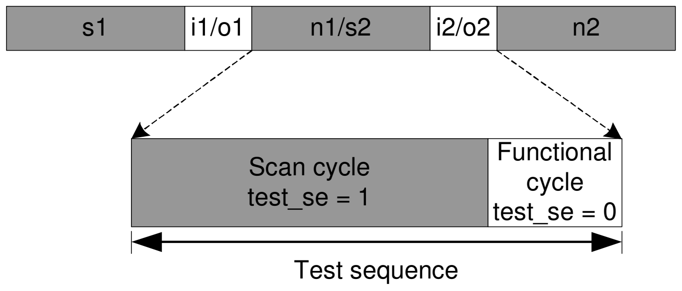
  Testing of scan circuits targeting faults in the combinational logic is a multi-step process. It involves shifting the test patterns generated by combinational ATPG into the SFFs, applying the shifted test patterns to the circuit, and shifting test responses out of the chip. The time it takes to complete this three-step process is the *test application time*. Figure 1.9 demonstrates the entire test application procedure for a circuit with one SC. The test patterns for the combinational logic are shown in Figure 1.9(a). There are two sets of test patterns in this example. $\{i1, i2\}$ are the parts of the test vectors applied at the primary inputs. $\{s1, s2\}$ are the parts of test vectors applied through internal FFs. $\{o1, o2\}$ and $\{n1, n2\}$ are the circuit responses available at the primary outputs and from the internal FFs respectively. Figure 1.9(b) illustrates the test sequences. For each sequence, the following steps are performed. Firstly, an input test vector for internal FFs is shifted into the chip by setting the test control signal (*test_se*) to 1 in the scan cycle. After that, an input test vector for primary inputs will be applied when *test_se* is set to 0 in the functional cycle. This allows the internal states to be updated and the circuit responses to be propagated to the primary outputs. Finally, the updated states are shifted out of the chip in the following sequence. They will then be used together with the expected responses for pass/fail analysis. As we can see from the example, the test application time is dominated by the scan time of internal FFs. To reduce test application time, one can divide FFs into multiple SCs which are driven simultaneously. Figure 1.10(a) shows the structure of a single SC. For this structure, the scan time of each test pattern for a test sequence will be n clock cycles. On the other hand, Figure 1.10(b) divides the SC into k segments. Each of these SCs has its own dedicated scan input and scan output. The scan time for each pattern is then reduced to $\lceil \frac{n}{k} \rceil$ clock cycles. However, additional pins or a more complex pin-multiplexing scheme will be required for this structure.

- *Volume of test data*

  The volume of test data (VTD) represents the amount of data a circuit needs to



(a) Combinational test vectors



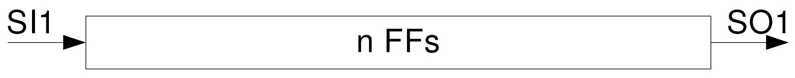
(b) Scan test sequences

Figure 1.9: Scan test sequences for single-clock design

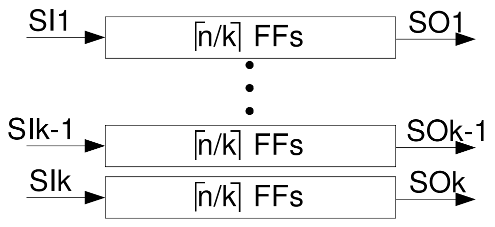
achieve a desirable fault coverage for a targeted fault model. For a scan design, the VTD can be calculated using Equation 1.1.

$$VTD = Num_{patterns} \times (2 \times Num_{SFF} + Num_{PIs} + Num_{POs}) \tag{1.1}$$

In this equation, $Num_{patterns}$ represents the number of test patterns for the design to achieve a desirable fault coverage. Num_{SFF} indicates the number of scan-flip-flops in the design. Num_{PIs} and Num_{POs} denote the number of primary inputs and primary outputs respectively. For example, if a circuit with 5,000 SFFs, 128 primary inputs and 128 primary outputs requires 2,000 test patterns to achieve a single stuck-at fault coverage of over 99%, the VTD will be around 20 Mbits. As size and complexity of VLSI designs increase, the VTD grows rapidly. To supply this massive VTD during test application, the size



(a) Scan design with one scan chain



(b) Scan design with k scan chains

Figure 1.10: Scan designs with different number of scan chains

of ATE buffers will have to be large enough to store all the data in one test session. Otherwise, reloading of buffers will be required. In both cases, the cost of test will be increased [8].

- *Test development time*

  The time it takes to transform an original design into a testable design that meets all the environmental and/or timing constraints is called *test development time*. The test development process includes the insertion and optimization of scan structures. For example, the scan insertion step will allocate each FF in one of the multiple scan chains and the optimization step may be required for ordering the FFs in each scan chain to reduce the routing overhead. If the optimization step cannot reduce the routing overhead sufficiently, some FFs may need to be reassigned to different SCs and the test development process proceeds iteratively. If, after a predefined number of iterations, the use of scan still violates the constraints, the original design may have to be changed to compensate for the added penalty of the scan structures. As a result, the prolonged test development time can directly affect the cost of the design.

1.4 Test Quality

Although it was proven that the single stuck-at fault model can cover a large spectrum of physical defects, new issues arise when circuits are implemented in nanometer technologies [12]. For example, to compensate for the decreasing effectiveness of quiescent current-based (I_{DDQ}) testing for circuits manufactured in smaller process geometries, the *delay fault model* is essential to screen the process variations that may affect only the circuit timing. The objective of testing for delay faults is to detect defects that adversely affect the timing behavior of a circuit without changing its logical operation under static conditions. A delay fault is detected when the amount of time a desired transition takes to propagate through the circuit from an initialization point to an observation point exceeds the period allowed for it [21]. To achieve delay fault detection, at-speed test application of two consecutive patterns is required, which imposes added constraints on scan development, as discussed next.

To detect a delay fault in a scan circuit, the primary inputs and internal flip-flops are used as initialization points, while the observation points include both primary outputs and flip-flops [38]. This scan-based delay fault test consists of two test patterns, V_1 and V_2. The first pattern V_1 is called the *initialization pattern* and it must first be applied to the circuit to initialize the logic into a known state. The second pattern V_2, called the *excitation pattern*, is then applied in the successive clock pulse to trigger the desired transition. This process is called the *two-pattern test methodology.* One way to apply two-pattern tests is the *broadside test application strategy.* In this strategy, the initialization pattern is first scanned into the SC and applied to the circuit to drive all the memory elements to known states. The excitation pattern is then derived as the combinational circuit's response to the initialization pattern. One major disadvantage with this strategy is that it complicates the test pattern generation problem [21]. *Skewed-load* (or last-shift launch) test application strategy for two-pattern delay fault test methodology eliminates the need for sequential ATPG. It can also reuse the available DFT infrastructure provided for stuck-at fault testing. In this strategy, the pattern V_1 is loaded into the SCs prior to test application, with V_2 as the shifted version of V_1. This correlation between the pattern pair imposes some

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
V_1	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
V_2	γ	α_1	α_2	α_3	α_4	α_5	α_6	α_7

Table 1.1: Example of a two-pattern test

restrictions on the possible patterns that can be applied due to the SC order [38]. To demonstrate this restriction, Table 1.1 shows the application of the pattern pair $\{V_1, V_2\}$ in the FF set $\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$. We assume that the SC is connected such that a single shift of the SC causes each bit to move one position to the right. The value α is the bit data in the initialization pattern V_1. The value γ is the new value that is shifted into the SC while V_1 is shifted once to obtain the excitation pattern V_2. Because scan chain order will determine the correlation between the test patterns and hence it will influence the detectability of delay faults, it is essential to investigate new ways to insert scan structures that account for skewed-load delay fault testing.

1.5 Thesis Organization

This thesis presents a new way to introduce scan structures into a circuit at the RTL. The remainder of the thesis is organized as follows. Chapter 2 gives details about how different parameters in SC construction can affect the cost of testing and summarizes the relevant prior approaches. The strengths and limitations of these approaches help introduce the motivation and objectives for the research presented in this thesis.

Chapter 3 introduces a new method to analyze circuits described at the RTL for SC construction. Detailed algorithms on how to analyze the RTL description of a design in order to build SCs are also included in this chapter. These algorithms generate generic scan structures for any type of designs (e.g., control-intensive designs, data-intensive designs). Moreover, by satisfying a set of constraints while building the SCs, *in addition to reducing area and performance penalty, the inserted scan structure can be tuned to enhance delay fault test quality* of a design using the skewed-load two-pattern test application strategy.

To address the escalating volume of test data and its influence on test cost, Chapter 4 will introduce two new reconfigurable Illinois Scan Architectures [17]. When combined with the scan structures that are generated by the functional scan synthesis process at RTL with additional constraints, *the volume of test data for scan designs can be effectively reduced*. Finally, the conclusion and suggestions for further work are given in Chapter 5.

Chapter 2

Scan Chain Design

The scan method is the most commonly used DFT technique to enhance testability of complex VLSI designs. The simplicity of the scan method allows automatic algorithms to be developed for scan insertion. This reduction in human effort allows the scan method to be applicable to large digital integrated circuits. In a scan design, testability is enhanced by replacing original FFs with SFFs as described in Section 1.2.3. Complete controllability and observability of the combinational circuit can be obtained by shifting data in and out of the chip through internal SCs. As a result, test pattern generation for complex digital integrated circuits can be done by combinational ATPG. The generated test patterns can then be applied to the CUT using external ATE or BIST as discussed in Section 1.2.3. Nonetheless, the many benefits from the scan method do not come for free. Various parameters are used to quantify the cost of scan. The discussion of different cost parameters has been provided in Sections 1.3 and 1.4. In Section 2.1, the contributing factors that influence different cost parameters of the scan method will be discussed. Section 2.2 gives the justification for constructing scan chains at RTL. A summary of the relevant DFT approaches for test cost reduction will be provided in Section 2.3. Finally, the motivations and objectives of this research will be outlined in Section 2.4.

2.1 Scan Chain Architecture

In this section, the different factors that will affect various cost parameters when inserting scan to a circuit will be detailed. The contributing factors that are considered are: *scan path construction techniques*, *scan cell partitioning into multiple scan chains* and *scan cell order*.

2.1.1 Scan Path Construction Techniques

The scan method enhances testability by providing a mechanism to transport test vectors into the internal state elements of a design. This path for transporting test data between FFs is called the *scan path* (SP). There are different ways to construct SPs between FFs and they influence the area and performance overhead of the scan design. In this thesis, two types of SPs are considered: *dedicated scan paths* and *functional scan paths*.

Dedicated Scan Path

The simplest way to connect SFFs is to use a *dedicated scan path*. In this approach, all the FFs are replaced with the SFFs shown in Figure 1.6(a). The multiplexing logic at the input of a SFF allows the insertion of dedicated SP to disregard the functional logic of the original design. This is because when using dedicated SPs to connect SFFs, test data is transported through a path that is completely independent of the functional path (FP), on which functional data is transported in the normal mode. Moreover, to create an SP, an extra wire is inserted to connect the output of a source SFF to the SD input of a destination SFF. This wire can then be used to transport test data when the circuit operates in the test mode. Due to this independency between dedicated SPs and FPs, simple automatic algorithms can be developed to insert scan structures for complex VLSI designs. In state-of-the-art practice, synthesis tools allow designers to perform scan insertion automatically at the gate level of design abstraction, after the structural netlist of the design is obtained [41].

In this dedicated SP architecture, the amount of additional hardware required to construct SPs increases rapidly with respect to the number of internal state elements in a design. These extra MUXes of the SFFs and routing of additional wires for the creation of SPs introduce undesirably large area overhead. Furthermore, for every added MUX located on the critical path of the design, a performance penalty of two-gate delays will be injected. Hence, a better way to construct SPs such that area and performance overhead can be reduced is necessary. This leads to the proposal of the *functional scan path* architecture, which is discussed next.

Functional Scan Path

Instead of inserting additional hardware to construct SPs, the *functional scan path* architecture tries to organize the sequential elements of the design into SPs by utilizing existing FPs as much as possible. The original FPs, which are used to transport functional data in the normal mode, are reused as SPs to transport test data in the test mode. The fact that a design will never operate in the normal mode and the test mode at the same time makes this path sharing mechanism possible. This is because functional data and test data will travel on the same path only at different time frames. The sharing of FPs and SPs diminishes the need to insert extra wires for SP construction. Moreover, although complex multiplexing logic is required to distinguish SPs from FPs in the dedicated SP architecture, it is not necessary when constructing functional SPs. Consequently, by utilizing existing FPs, the functional SP architecture may be able to reduce the amount of extra hardware needed to construct SPs. This reduction in test logic in turn decreases the area and performance overhead associated with the scan method.

An example of how an FP is converted into an SP is shown in Figure 2.1. In this example, instead of introducing an additional MUX, an OR gate is inserted to connect FFA and FFC together as an SP in the test mode when the value of the test control signal *test_se* is 1. By identifying cost saving FPs and converting them into SPs, the need for additional hardware for SP creation is diminished. Moreover, utilizing these FPs can reduce the routing overhead. This is because additional wires will not be needed to connect the SFFs. After SPs between FFs are constructed, they are combined together to build SCs. The next section will detail on how SP selection for SC construction can also affect the cost of test.

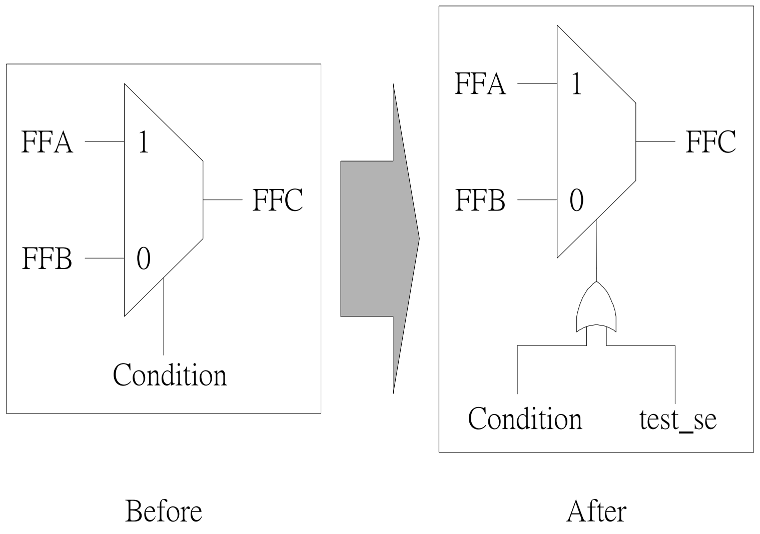
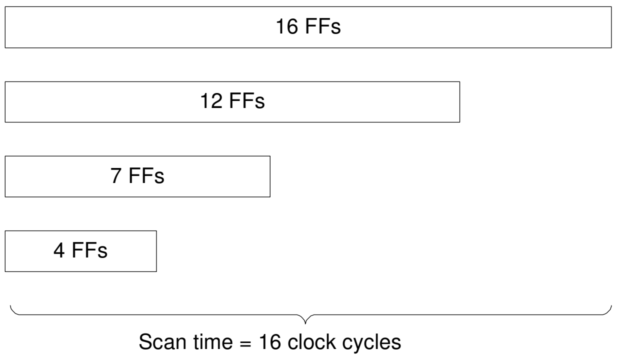


Figure 2.1: Modification of functional path

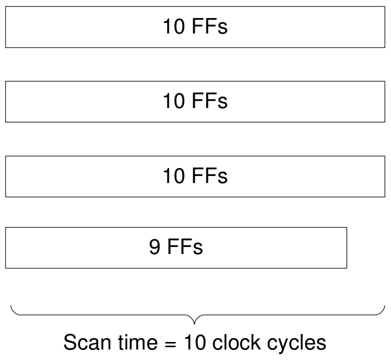
2.1.2 Scan Cell Partitioning Into Multiple Scan Chains

As mentioned in Section 1.3, the test application time for a design is dominated by the scan time of test vectors. As circuit size and complexity increase, the number of FFs in a design rises with it. The growth in FF number prohibits all the FFs to be chained into a single lengthy SC. Thus, it is common to divide FFs into multiple SCs to reduce the scan time. When the SCs are driven simultaneously, it is obvious that the scan time of the longest SC becomes the scan time of the design. This is because a circuit cannot be tested until all the FFs are set to known states. This phenomenon is illustrated in Figure 2.2(a), in which the scan time is 16 clock cycles. Hence, it is always beneficial to balance the length of SCs. Figure 2.2(b) shows four balanced SCs, where each SC has 10 FFs. Although this example has the same number of FFs as in Figure 2.2(a), the scan time is reduced to only 10 clock cycles.

In addition to test application time, SP partitioning into multiple SCs may affect the test quality for delay fault tests, which will be discussed in Section 3.5. Moreover, Chapter 4 will show that the volume of test data of a scan design can also be better compressed by exploring this contributing factor.



(a) Unbalanced scan chains



(b) Balanced scan chains

Figure 2.2: Scan time of different scan chain lengths

2.1.3 Scan Cell Order

It was already mentioned in Section 2.1 that an SC is constructed by chaining FFs together in a design, and the path between any two FFs in an SC is an SP. It is then obvious to conclude that an SP only exists between two FFs if they are chained adjacent to each other in an SC. This order of SFFs in an SC thus impacts the wire lengths of the SP in both scan construction techniques described in Section 2.1.1. For instance, it is assumed that two FFs are chained adjacent to each other. If they are located physically far away from each other in the design, it will require a long wire to establish the SP. Routing of this long SP thus becomes more troublesome. Furthermore, the scan cell order also affects the utilization of FPs as SPs, which directly impacts the area and performance overhead of the scan design when constructing functional SCs. For example, it is assumed that to reduce the cost of test in the case of functional scan, the FP between a source FF (FFA) and a destination FF (FFB) will need to be utilized as SP. However, if FFA and FFB are not placed adjacent to each other in an SC, the FP will not be reused. This is because there is no need to establish a path between FFA and FFB to transport test data any more. Thus, the savings from functional scan have vanished.

In addition to affecting the area and performance overhead, Section 3.5 will show that scan cell ordering also affects the delay fault coverage of a scan design when using the skewed-load test application strategy for two-pattern tests. Also, the effectiveness in compressing test data of a scan design can be enhanced through consciously determining the scan cell order by incorporating additional constraints during the scan synthesis process. This will be discussed in detail in Chapter 4.

2.2 Gate vs. Register-Transfer Abstraction Level

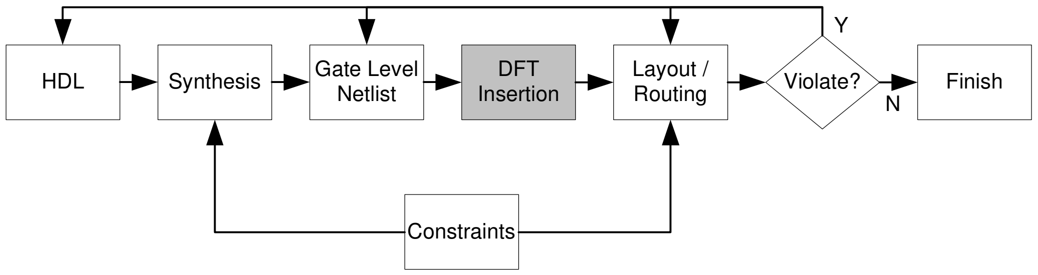
The scan chain architecture was already shown to directly affect the cost of test for scan designs. Nonetheless, constructing SCs at different levels of design abstraction can also impact the cost parameters. For instance, test development time can be influenced by introducing DFT structures at the RTL. This is because considering DFT early in the design cycle can produce a different design flow that allows synthesis tools to better meet design constraints by optimizing test logic and functional logic concurrently, which is not possible when DFT structures are inserted at the gate level [14].

- Gate level DFT insertion

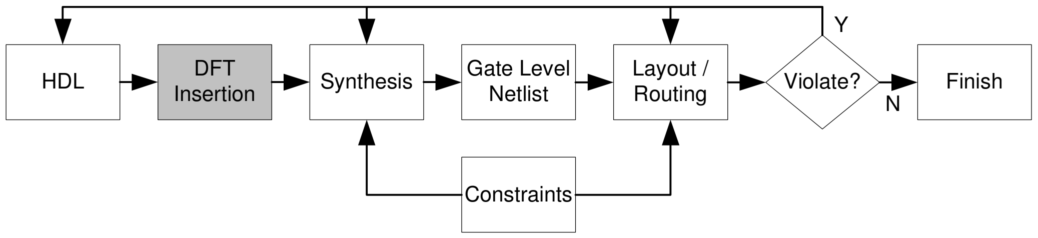
  Figure 2.3(a) illustrates the state-of-the-art design flow, where DFT structures are inserted after the RTL circuit description has been synthesized into the gate level structural netlist. In this case, the impact of DFT structures in terms of test cost will not be realized until late in the design flow. If design constraints are violated due to the addition of DFT structures, either the DFT structures or the original design will have to be modified to compensate the cost of test. This iterative process of circuit re-optimization may translate into lengthy development time, which in turn increases the cost of the design.

- RTL DFT insertion

  On the other hand, Figure 2.3(b) shows a different design flow, where DFT structures are inserted at the RTL. By introducing DFT at the RTL, synthesis



(a) Gate level DFT Insertion



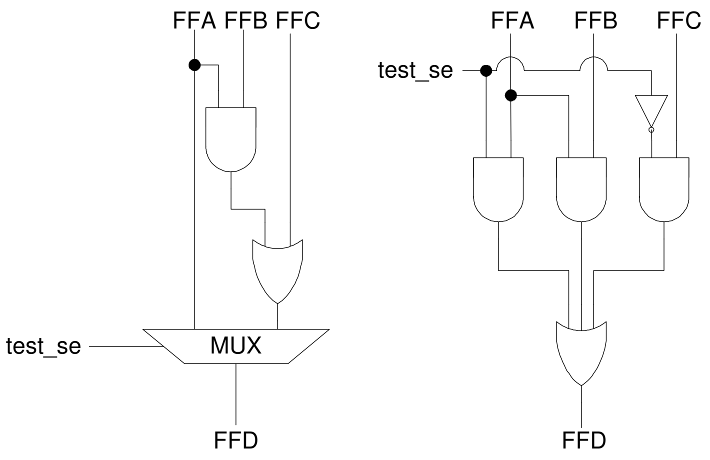
(b) RTL DFT Insertion

Figure 2.3: Design flow with gate level and RTL DFT insertion [19]

tools can have greater flexibility to optimize the testable circuit to meet the design constraints by considering the test logic and functional logic simultaneously during synthesis. This is illustrated in Figures 2.4(a) and 2.4(b). Figure 2.4(a) shows the implementation of gate level scan, where an extra MUX is inserted to create an SP. In this case, the performance penalty for the critical path from FFA to FFD is 4 gate-delays. Conversely, by inserting scan at the RTL, synthesis tools can optimize the scan logic together with the original circuit, producing the optimized design in Figure 2.4(b). In this case, the delay in the critical path is reduced to only 2 gate-delays. The problem of wire delay can also be addressed by embedding test logic into functional logic to eliminate the need for long wires in the construction of SPs. Furthermore, this flexibility of synthesis tools may be able to lessen the area overhead of scan designs. This is because the synthesis tool can now better optimize the testable circuit to reduce the logic per FF ratio. In addition, introducing scan at the

RTL can eliminate the need to re-order scan cells at later stages of the design flow (i.e., layout and routing stage) to reduce routing overhead from the scan structure based on two reasons. First of all, extra wires are not needed when SPs are created by reusing existing FPs in a design. Secondly, in the case when additional wires are needed for SP creation, the inserted wires will not affect the timing closure of the synthesized design. This is because during RTL/logic synthesis, the scan structure is already taken into consideration by the synthesis tools. Thus, the resulted netlist will not contain wires that may influence the timing closure of the design after the creation of SPs. Moreover, constructing scan at the RTL (prior to logic and physical synthesis) can facilitate the embedding of additional constraints, which will tune the generated scan structure for different objectives, without affecting the timing closure of the scan circuit. However, there is an disadvantage in constructing SCs at the RTL. If there are redundant FFs in the original design, synthesis tools will not be able to remove them during logic synthesis anymore as in the case when scan is inserted at the gate level. This is because these redundant FFs are now included in the SPs in the scan design. However, note, this problem can also be solved using a pre-processor that parses the initial RTL description for any redundant sequential elements (caused mainly by legacy or parameterized RTL code) and marks them as un-scannable.

Different levels of design abstraction also affect the choice of techniques to construct SPs. To identify the different types of FPs described in Section 2.1.1 available in a design, the original functional information in the design must be analyzed. Although one can construct functional SPs at the gate design abstraction level, the huge size and complexity of the circuit description at this level make the task of FP identification extremely difficult. As a result, *state-of-the-art synthesis tools only provide automatic algorithms to construct dedicated SPs at the gate level* [41]. As a result, it is beneficial to insert functional scan at the RTL because the size and complexity of a design that must be analyzed are significantly lower that at the gate level. Thus, tractable algorithms can be developed to identify FPs at RTL.



(a) Gate Level scan (b) RTL scan

Figure 2.4: Optimization of scan logic

2.3 Relevant Approaches On The Scan Method

As shown in Section 2.1 and 2.2, various factors can affect the test cost of the scan method. A number of methods had been proposed to explore these factors and attempt to reduce the cost of test for scan designs. In this section, the strengths and limitations of these proposals will be summarized. The summaries are divided into four categories relevant to the work presented in this thesis. The first category goes over the proposals that target to enhance testability using various DFT test structures at the RTL. Techniques to reduce the cost of scan are evaluated in the second category. The third category reviews methods that aim to improve delay fault coverage for complex digital integrated circuits using the scan method. The fourth category discusses the various techniques to improve test data compression for scan designs.

2.3.1 Enhancing Testability At RTL

Roy et al. proposed a method to insert test points at RTL in order to improve single stuck-at fault coverage of a design in the BIST mode [35]. By using RTL controllability/observability measures, test points are inserted in the RTL description prior to logic synthesis. Boubezari et al. introduced another method for RTL test point insertion [6]. Controllability and observability information is gathered by analyzing the internal signals of each functional module (FM) in the design. The FMs include simple adders/subtracters, comparators, multipliers and multiplexers. As a result, the method becomes dependent on how the FMs are synthesized into hardware. Lin et al. proposed another method to enhance testability of a design. By analyzing the controllability of the design, test points and extra logic are inserted to establish SPs in the combinational logic of the design [23, 24]. However, in addition to the circuit information from the RTL description, these methods also require gate level information for the computation of controllability and observability of the design. The identification and modification of functional logic at the gate level that can be shared for test purposes may lead to very high computational cost. Thus, they are not applicable for large industrial designs. To avoid the computation of controllability and observability, the scan method can be used to enhance testability of complex circuits. The simplicity of scan makes it the most popular DFT technique in the test community. In the next section, various techniques that were proposed to reduce the cost of scan will be discussed.

2.3.2 Techniques To Reduce The Cost Of Scan

Roy presented a method for chaining FFs at RTL [34]. By employing a bottom-up iterative algorithm to the VHDL source for different processes, memory elements identified in the design are chained randomly to establish SPs. By constructing SCs at RTL, synthesis tools can better optimize test logic together with functional logic. Although this method could reduce area overhead of the scan design, the structure of the design is not exploited. Chickermane and Zarrineh proposed a method to transform memory elements to level-sensitive-scan-design (LSSD) at the early stage of the design flow [10]. By analyzing the structural information of the design, additional logic is inserted to preserve functional behavior of the SFFs in the normal mode, while ensuring the scan functionality in the test mode. Although this approach allows synthesis tools to optimize test logic and functional logic together during synthesis, no information on how to divide the memory elements among different SCs in order to reduce the overall area overhead of the scan design is provided. Norwood and McCluskey presented an orthogonal scan methodology that utilizes existing FPs through functional units to establish SPs [27, 29]. Orthogonal scan paths are defined during the allocation and binding phases of architectural synthesis, based on the assumption that functional units can be treated as transparent objects in the SPs. To achieve this, the routing of control signals to the functional units incurs additional area overhead. Bhattacharya and Dey proposed H-SCAN [5], which was later improved by Asaka et al. to H-SCAN+ [3]. The H-SCAN approach uses only the FPs that include a multiplexer (MUX) in between two registers and, by modifying the logic that controls the MUXes, SPs can be established in the test mode. This is achieved by exploiting the structural information of RTL designs obtained from behavioral descriptions through high level synthesis. Although H-SCAN can reduce both the area and performance penalty associated with gate level scan, no consideration is given for the case when no MUXes are present in between FFs and no detail is provided on how the SFFs are chained to create multiple balanced SCs.

Recently Huang et al. proposed a new method for inserting scan structures at the RTL [19]. The costs to transform different types of FPs described in Section 2.1.1 to SPs are first calculated. A greedy algorithm is then used to search through FPs and select the ones that can be used to establish SPs with minimum cost. The FFs that are not selected by the greedy algorithm are then merged into the selected path by cutting and slicing, after the SCs have been generated. Due to the cost criteria used by the greedy search algorithm, FFs that belong to multiple paths are overlooked. Therefore, only a limited number of FPs are explored. In addition, the SI/SO selection is not related to scan path generation algorithm. Hence, the number of FPs that are explored is further reduced. All of the above issues will ultimately influence not only the added DFT area, but also the timing of the scan circuit.

In practice, the RTL description of a design is organized in different modules, thus producing a design hierarchy. As a result, it is essential to insert scan structures to hierarchical designs at RTL. In [19], different processes in the source code are treated as independent entities. Once SPs in each process are constructed, they are then connected to each other via the communication channels between different processes in the same module. Although the method from [19] is able to construct SPs for all the processes in a single module, no detailed information is provided on how it can be applied to hierarchical designs where FFs are located in different processes of various modules. Nevertheless, one may be able to extend the method from [19] to support hierarchical designs in a bottom-up approach by replacing the processes with module instantiations and afterward applying the same greedy algorithm. As a result, the algorithm will first try to organize FFs in the child modules before combining the different SPs in the parent modules. However, one problem with this extended approach is the generation of balanced SCs. To produce balanced SCs, slicing and stitching will have to be applied to the SCs that exceed the desired length. Therefore, if the slicing point is located inside a child module deep down in the design hierarchy, then placing the sliced FFs into the SCs from other modules may adversely influence both area and performance.

2.3.3 Improving Delay Fault Coverage

One benefit of the scan method is its applicability for the delay fault model. Section 1.4 already mentioned the importance of delay fault testing. To detect delay faults using the scan method, the skewed-load test application technique for two-pattern test strategy can be used. However, due to the correlation of patterns that can be applied using this technique, delay fault coverage for large designs may be insufficient. In this section, several relevant previous methods targeting the improvement of fault coverage in delay fault testing using various techniques will be reviewed.

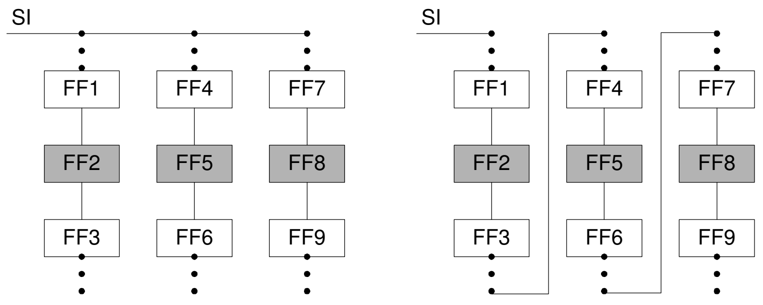
The straightforward way to eliminate the restrictions on the possible patterns that can be applied to the combinational circuit block is to use enhanced scan flip-flops (ESFFs) proposed in [13]. ESFFs are special flip-flops (FFs) that can hold two bits of state information. The extra storage allows any arbitrary pattern pairs $\{V_1, V_2\}$ to be stored before application. Although the use of ESFFs removes any correlation between the two consecutive patterns, the extra hardware for building the special FFs translates to high area overhead. Instead of transforming all the FFs into ESFFs, Cheng et al. introduced a technique to reduce the number of ESFFs needed in a design, which results into a partial ESFFs structure in [9]. By analyzing the pattern sets generated from a full ESFFs structure of the circuit, it was established that only a subset of FFs need to be made ESFFs in order to be able to achieve high delay fault coverage. However, even with partial ESFF, the area overhead for designs with high flip-flop to logic gate ratio can still be large.

Instead of using ESFFs, Pomeranz and Reddy showed in [32] that by dividing SCs into multiple small segments, each with its own scan input and scan output ports, the delay fault coverage can also be improved. Nonetheless, the exhaustive nature of their algorithm restricts the method to be only applicable to small designs. Moreover, having a very large number of SCs in a design may be undesirable since, to drive all the SCs concurrently, testers with high number of channels are also required. A DFT scheme that helps improve delay fault coverage was proposed in [22]. By adding *SCLATCH*es and additional MUXes to the design, scan paths can be reconfigured to allow arbitrary pattern pairs to be loaded in the scan cells. However, their approach is not only test set dependent, but the additional hardware may also result in significant area overhead. Scan mapping for applying two-pattern tests in a standard scan design environment was described in [44]. By inserting combinational mapping logic before the scan elements, the second pattern V_2 of a pattern pair can be generated in the next clock pulse. Although this approach can improve the delay fault coverage, the area and performance overhead associated with the mapping logic may exceed that of enhanced scan. Another approach that inserts additional hardware to the design to improve delay fault coverage was proposed by Altaf-Ul-Amin et al. in [2]. The circuit is analyzed in order to extract the hierarchically two-pattern testable (HTPT) data paths, which are then used to determine where and what kind of additional hardware is inserted to the design to transport test data. Although this approach slightly increases the area overhead compared with enhanced scan, it is only applicable to data intensive designs.

By analyzing the circuit at the gate level, Hurst and Kanopoulos suggested that FFs are divided into different modules according to the logic cones that they are driving [20]. The FFs in the SCs are then rearranged so that no two adjacent FFs in the same SC are from the same module. The test pattern pairs can now be stored in the adjacent FFs of an SC to facilitate arbitrary second pattern V_2 and to be loaded to the circuit during test. Instead of adding complex logics to the design, this approach only requires FFs to be moved around in different SCs. This results only in an insignificant amount of area overhead. However, the complexity of analyzing a circuit at the gate level prohibits the application of this approach to large circuits. In order to reduce the complexity of circuit analysis, orthogonal scan generated from behavioral descriptions was described in [28]. Information about the FFs in a circuit is gathered at the behavioral level, where complexity is low. The information is then utilized to construct scan paths during architectural synthesis so that no two scan cells driving the same functional unit are placed adjacent to each other in the scan path. This approach maintains all the benefits from [20], while allowing complex circuits to be analyzed at a higher level of design abstraction. However, the underlying assumption is that the circuit specification is given in the behavioral domain at the algorithmic level. Furthermore, by accounting for orthogonal scan during high level synthesis, only the data path of the circuit is guaranteed to have high delay fault coverage.

2.3.4 Improving Test Data Compression Ratio

Another concern associated with the scan method is the huge volume of test data required to test a scan design. This problem can be solved by employing the Illinois Scan Architecture. It was proposed by Hamzaoglu and Patel [17] and is aimed at reducing test data volume with minimum area overhead. Due to its ability to compress test data with very low area overhead, Illinois Scan is emerging as a promising low-cost test method [18, 47]. Illinois Scan operates in two modes: the *broadcast scan mode* and the *serial scan mode*. In the broadcast scan mode, a single scan input (SI) is connected to multiple SCs. This allows the same set of test patterns to be shifted into



(a) Broadcast mode (b) Serial mode

Figure 2.5: Different modes of the Illinois scan architecture

multiple chains simultaneously. For example, let's assume there is a circuit having 100 flip-flops (FFs) that would be split into four SCs with 25 FFs in each chain. When using the broadcast mode of Illinois Scan, FFs 1, 26, 51 and 76 would receive the same set of test data, while FFs 2, 27, 52 and 77 receive another set of test data, and so on. In other words, FFs in different SCs, but at the same scan depth, will receive identical set of test data from the common SI in the broadcast mode. As a result, the volume of test data and test application time can be reduced. However, due to the strong correlation of the test data that is fed in different SCs in the broadcast mode, the resulting fault coverage of the design may decrease [18]. For example, there are three SCs being driven by a single SI in Figure 2.5(a). If $FF2$, $FF5$ and $FF8$ drive the same logic cone and they are located in different SCs at the same scan depth, the fault coverage will be reduced due to the limited set of test data this logic cone can receive from the three FFs in the broadcast mode. To improve fault coverage, serial mode would then be used to reconnect the multiple SCs into a single long SC, as shown in Figure 2.5(b). This allows FFs that drive the same logic cone to receive any arbitrary set of test data. However, the large volume of serial test patterns will reduce the effectiveness of this technique to compress the overall test data set.

Hamzaoglu and Patel proposed a technique to rearrange scan cells in order to reduce the number of serial patterns for Illinois Scan [17]. By analyzing a complete partially specified test set for the full scan circuit without using any test set compaction algorithms, compatibility classes of the FFs are computed. A pair of FFs is classified as compatible if they have the same set of test patterns. Once the compatibility classes are computed, scan cells are rearranged such that FFs at the same scan depth but distributed in different SCs must be compatible. Although the greedy heuristic could improve the effectiveness of Illinois Scan for test data volume reduction, this approach is test set dependent.

When FFs are divided into multiple SCs in Illinois Scan, the SC length k is an important parameter that could affect both the scan time and the test data volume reduction effectiveness. When k is small, the number of SCs increases, which decreases the test application time by lowering the amount of test data loaded in the broadcast mode. However, the high number of SCs increases the number of faults that cannot be tested in the broadcast mode. As a result, large amounts of serial test patterns, which starts increasing test data volume and test application time, will be needed. An incremental test generation algorithm for finding the optimal SC length k for Illinois Scan was proposed by Pandey and Patel [30]. By first choosing a number n smaller than or equal to the total number of FFs in the design, where n has a large number of prime factors, ATPG is applied to generate the test data for each Illinois Scan for different values of n, which is updated according to Equation 2.1.

$$n_{new} = \frac{n_{old}}{smallest\ prime\ factor\ of\ n_{old}} \tag{2.1}$$

Although by incrementally reducing the fault list can reduce ATPG time in a single run for multiple Illinois Scan configurations, the recursive nature of the algorithm may become computational expensive for large designs.

In addition to the serial mode and broadcast mode with SC length k, Pandey and Patel introduced a reconfiguration technique for Illinois Scan such that an additional broadcast mode is inserted to the Illinois Scan Architecture [31]. In this additional mode, SCs are reconfigured so that the first SC (SC1) is rotated by one bit, while SC2 is rotated by two bits, until all subsequent SCs are rotated in the same manner. As a result, FFs that have the same scan depth in the normal broadcast mode will have different scan depths in this additional mode due to the SC rotation. This allows a new set of parallel test data to be applied for the previously undetected faults and thus it reduces the number of serial test patterns. Another technique for reconfiguring the Illinois Scan Architecture was proposed by Samaranayake et al. [36]. Instead of using a single SI to drive multiple SCs in the broadcast mode, multiple SIs are employed by this technique. These SIs are combined with mapping logic that consists of MUXes and inverters. The controls of mapping logic are determined by analyzing the fault lists. If a targeted fault is undetected in one configuration, the circuit will be reconfigured to allow different patterns to be fed into the SCs. The mapping logic allows any SI to drive any combination of SCs, with the ability to invert test data while they are being shifted onto the SCs if necessary. Thus, FFs in the same scan depth can receive any combination of test patterns in different broadcast modes. Consequently, the need of serial patterns can be eliminated, which in turn reduces test application time and test data volume. Although this technique can improve the effectiveness of test data compression, the analysis of compatability of SCs in different configuration by applying ATPG recursively may significantly exceed the processing time of regular ATPG. Another way to eliminate the need for serial patterns is to use the Reconfigurable Shared Scan-in architecture (RSSA) proposed in [39]. By inserting extra mapping logic to the Illinois Scan Architecture, the scan enable signal can be used to configure the architecture into different broadcast modes during test application. However, the analysis of potential conflicts for scan cell allocation into multiple SCs can be computationally expensive for large designs when it is performed at low levels of design abstraction (e.g., gate level).

Since the compression effectiveness in Illinois Scan is dependent on the presence of correlations between FFs in the same scan depth in multiple SCs, it is necessary to investigate a generic yet computationally-efficient functional SC design method at RTL that can reduce volume of test data, and at the same time, decrease area overhead and improve the speed of the synthesized scan circuit.

2.4 Motivation And Objectives

The simplicity of scan makes it the most commonly used DFT technique for testing complex VLSI designs. However, in addition to all its benefits, it also increases the cost of test, which was detailed in this chapter. In order to reduce test cost, Section 2.1 had explained that by considering various contributing factors while constructing SCs, different test cost parameters will be affected. Section 2.2 illustrated that by introducing scan at the RTL, the cost of test can be further decreased. Although the problem of inserting scan at the RTL has been analyzed in the past, the previous solutions have different limitations as mentioned in Section 2.3.2. Moreover, to the best of the author's knowledge, none of the previous solutions investigate the use of functional scan at the RTL to enhance delay fault coverage or to reduce the volume of test data. Thus, a new generic method for *constructing functional scan chains at the RTL* is needed.

The proposed method, which will be detailed in Chapter 3, uses the FP classification technique from [19] to identify FPs that can reduce the cost of scan when they are used as SPs. However, in order to better utilize these cost saving FPs when generating scan structure at the RTL, an alternative technique will be introduced to address the following problems during SC construction for digital integrated circuits:

1. How to order the memory elements in the SCs such that the maximum number of cost saving FPs are used?
2. Which existing PI/PO should be reused as SI/SO in designs with multiple SCs?
3. How to balance the length of the SCs in order to reduce test application time?

The alternative technique to the above problems is justified by the need to *facilitate the inclusion of custom constraints into the scan synthesis process*. As a result, the generated scan structure can be optimized for different objectives. For instance, as delay fault testing becomes growingly important, the skewed-load test application strategy was introduced to facilitate two-pattern delay fault test with scan designs. However, the delay fault coverage using this strategy strongly depends on the correlation of test patterns within an SC as discussed in Section 1.4. Consequently, *a* *set of new constraints will be introduced* in Section 3.5 for SCs construction such that correlation of test patterns within an SC can be eliminated. Moreover, as mentioned in Section 1.3, test application time and volume of test data become the bottlenecks of the scan method. Although the *Illinois Scan Architecture* can reduce the volume of test data with low area penalty, while reusing the existing scan structure of a testable design, the effectiveness of this technique in compressing test data is limited by scan cell partitioning in multiple SCs and scan cell order within each SC. As a result, Chapter 4 will present *two new reconfigurable Illinois Scan Architectures*, as well as *a set of new constraints to be incorporated in the scan synthesis process* to minimize the occurrence of data correlations between multiple SCs in the broadcast mode of these architectures.

Chapter 3

Functional Scan Synthesis At RTL

In this chapter, a new method to construct functional SCs at the RTL is introduced. By analyzing designs at the RTL, the complexity of identifying reusable FPs in a design is reduced when compared to the gate level counterpart. Because of this, the generated scan structure can be better optimized to reflect the area and performance savings from functional scan in opposed to the dedicated scan path structure. Moreover, RTL scan insertion also gives better flexibility to the synthesis tools to optimize the test and functional logic simultaneously, which will lead to improved circuits speeds with similar area when compared to the gate level dedicated scan insertion. In addition, by incorporating additional constraints in the scan synthesis process, the generated scan structure can be optimized to enhance delay fault coverage, or to reduce the volume of test data.

The process of constructing functional scan at RTL can be divided into multiple steps, which is illustrated in Figure 3.1. The implementation effort of each step of the program will be discussed in Appendix B. The program starts by analyzing a design that is described by Verilog HDL at the RTL. The analysis then produces a set of *control/data flow graphs* (CDFGs). This process of CDFG generation will be detailed in Section 3.1. After that, the CDFGs will be used to generate a *sequential graph* (S Graph), which will be explained in Section 3.2. The process of *SP identification* from the generated S Graph will then follow and will be discussed in Section 3.3. These SPs will then be connected to construct SCs as described in Section 3.4. After discussing

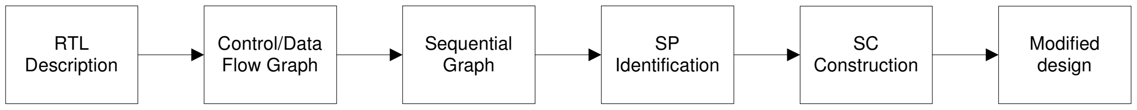


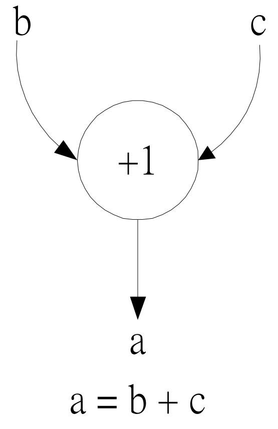
Figure 3.1: Functional scan synthesis at RTL

the algorithms to generate scan structures that are optimized to reduce area and performance penalty, Section 3.5 introduces the new constraints that can tune the scan design to enhance delay fault coverage when using the skewed-load two-pattern test application strategy. Experimental results will then be shown in Section 3.6.

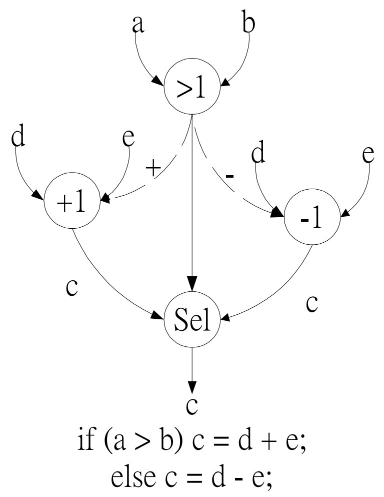
3.1 Generating Control/Data Flow Graph

The first step in the RTL scan synthesis process is to generate a set of CDFGs from the RTL description of a design. An CDFG contains control and data flow information between memory elements of a circuit. To represent the two types of information, an CDFG will contain control nodes and operation nodes, as well as control edges and data edges. In order to better explain the CDFG, two examples are illustrated in Figure 3.2. In Figure 3.2(a), a simple addition operation described by the transfer function $a = b + c$ is translated into an CDFG. The data node $\{+1\}$ shows that an addition is performed. This operation takes inputs from the two data edges $\{b, c\}$, and outputs the sum to the data edge $\{a\}$. In this case, no control information is presented. On the other hand, Figure 3.2(b) shows the CDFG of a simple IF statement. In this case, two operations that are denoted by the two data nodes $\{+1, -1\}$ are specified in the transfer function. The operation $\{< 1\}$ is used as the condition to determine which result should be acquired in the control node $\{Sel1\}$. The control edge $\{+\}$ indicates the result from operation $\{+1\}$ will be selected if the condition is evaluated to TRUE. Conversely, the control edge $\{-\}$ specifies that the result from the other operation will be chosen.

By analyzing all the transfer functions specified in Verilog HDL, a set of CDFGs will be generated for a design. These CDFGs can then be used to create a sequential graph, which will be discussed in the following section.



(a) Data operation



(b) Control operation

Figure 3.2: Examples of CDFGs

3.2 Generating Sequential Graph

After the set of CDFGs are created from the RTL description, the next step in the scan synthesis process is to generate a *sequential graph* (S Graph). An S Graph is used to show the control and data flow between all memory elements (i.e., flip-flops) in a design. It starts with a set of nodes that represent PIs, and ends with a set of nodes that represent POs of the design. Moreover, for every memory element that is found from the analysis of CDFGs, a node will be created. And whenever a path where data can be transfered between two elements is found from the CDFGs, a data edge will be inserted to connect the corresponding source and destination nodes in the S Graph. Similarly, whenever a path where control information can be transfered is identified from the CDFGs, a control edge will be inserted to the S Graph.

After the S Graph is successfully generated, a complete view of control and data flow of the design can be obtained. Since data edges in the S Graph represent data flow between memory elements, they can also be categorized as different types of FPs.

As a result, the next step in the scan synthesis process is to identify the candidate FPs that may lead to savings in area and performance when they are reused as SPs. The algorithms for doing that will be detailed in the following section.

3.3 Identifying Reusable Functional Paths

Reutilization of FPs as SPs in a design is the main reason why functional scan incurs less area and performance penalty compared with scan structures that are generated by the dedicated SP technique as described in Section 2.1.1. The three types of FPs that can be identified are [19]:

Type I: Paths that directly connect registers in the original circuit.

Type II: Paths that already have some multiplexer-like logic between registers.

Type III: Paths that have complex logic between registers.

It is obvious that the savings in terms of area and performance penalty will not be the same if different types of FPs are reused as SPs when constructing SCs. It is thus necessary to estimate the reutilization cost of the various types of FPs. They can be stated as follows:

Type I: These paths can be used as SPs without any additional cost.

Type II: These paths can be reused by inserting an AND/OR gate to control the MUX and establish an SP in the test mode as shown in Figure 2.1. As a result, the cost for using this type of FP is 1 logic gate.

Type III: To utilize these paths, additional MUXes must be added. In other words, the cost will be the size of a MUX.

Using the above cost structure, the cost to reuse each FP (i.e., data edge) in the S Graph can be calculated. By identifying FPs of Types I and II and converting them into SPs, the need for additional MUXes is eliminated. Moreover, utilizing these FPs can reduce the routing overhead. This is because additional wires will not be needed

Variable Name	Representation
S_Graph	$\{V, E \mid V \in \text{FF}, E \in \text{FP}\}$
V_i	Vertex i in S_Graph
V_{start}	Vertex in S_Graph without incoming edge
SP_i	SP i in design
SC_i	Scan chain i
SL_i	Length of scan chain i
SI_i	Scan input i
SO_i	Scan output i
L_i	List of data type i

Table 3.1: Terminology for scan synthesis

to connect the SFFs. Furthermore, despite the insertion of MUXes for the reuse of Type III FPs, synthesis tools can optimize these extra MUXes and the functional logic simultaneously if they are introduced before the gate design abstraction level. *Algorithms 1* and *2* illustrate an alternative technique to [19] for detecting these potential FPs. However, before introducing the algorithms, the terminology that will be used for all the algorithms throughout the thesis is listed in Table 3.1.

FPs are defined as paths between two FFs in a design where MUX insertion for chaining FFs together in an SC is not required, and *SPs represent a sequence of FPs that are reused in an SC*. Thus, the first step in *Algorithm 1* is to remove all the edges in the S Graph that represent Type III FPs, which require additional MUXes to be inserted for SP establishment. After that, *self loops of memory elements* and *feedback loops* in the S Graph are removed since we are not interested in building SCs with loops. There are two main types of feedback loop that can be found in the S Graph as illustrated in Figure 3.3. For the Type I feedback loop shown in Figure 3.3(a), the last edge in the loop is removed. This is because removing that edge can produce the longest possible sequence of cost saving FPs (i.e., SP) from the loop. On the other hand, in the Type II feedback loop, illustrated in Figure 3.3(b), since the length of SP will be the same no matter which edge is eliminated, the edge with the highest cost will be removed. After that, at step 3 of *Algorithm 1*, the edges that represent paths that violate any custom constraints if they are scan chained are removed. These constraints that can tune the generated scan structures

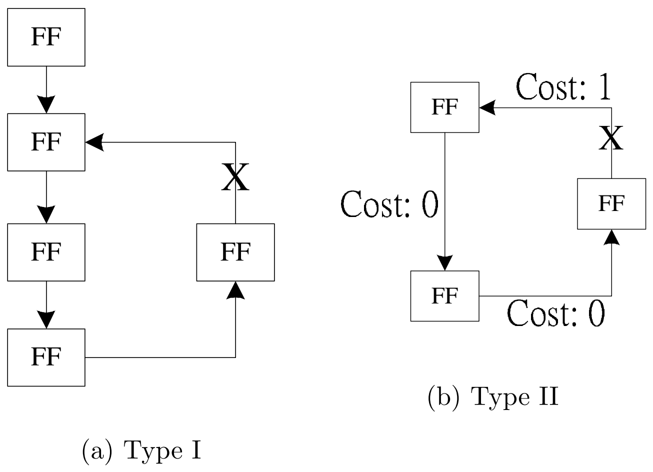
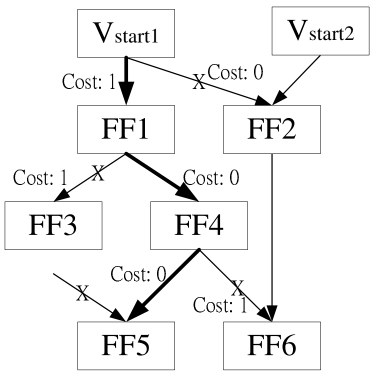


Figure 3.3: Different types of feedback loops

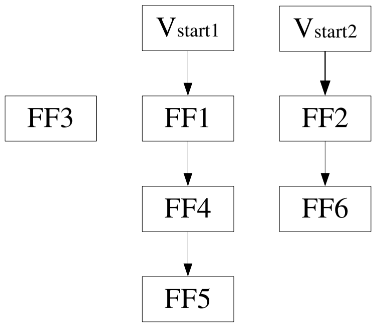
to enhance delay fault coverage or reduce the VTD will be introduced in Sections 3.5.1 and 4.3. This is different to prior works on RTL scan [3, 5, 19, 34] that focused only on reducing area and performance penalty.

At this stage of *Algorithm 1*, only FPs that can bring potential savings to the scan design are retained in the S Graph. Figure 3.4(a) shows a sample of S Graph at this stage. However, even though all of the remaining edges can potentially reduce the cost of scan, not all of them can be reused as SPs. This is because in order to prevent corruption of test data that is shifted into the SCs during test, each memory element can only be accessible through one SC. As a result, it is necessary to remove the redundant edges in this S Graph, so that all the remaining edges can be reused as SPs. This is done in *Algorithm 1* by the FOR loop at step 4, which will start traversing the S Graph from the *candidate nodes* (i.e., nodes that do not have any incoming edge) by iteratively applying the algorithm *TraverseSGraph* shown in *Algorithm 2*.

The goal of *Algorithm 2* is to remove all redundant edges so that each memory element is accessible through only one path. Moreover, in order to reduce the cost of scan, it should only remove a minimum number of redundant edges from the S Graph. In order to better explain this algorithm, a portion of an example S Graph,



(a) Starting S Graph



(b) Final S Graph

Figure 3.4: S graph at different stages

after the removal of loops, is shown in Figure 3.4(a). Starting from V_{start1} in the S Graph, $FF1$ is chosen and kept in this sequence of FPs (i.e., SP) even though the cost of using that path is higher than that of $FF2$. This is because $FF1$ is the only child node of V_{start1} with one incoming edge (steps 1-6). The reason *Algorithm 2* gives higher priority to child nodes with single rather than multiple incoming edges is that even if a child with multiple incoming edges does not get selected in this iteration, it may be chosen later by another SP since there are multiple paths going into that child. On the other hand, if a child with single incoming edge is not selected, there will not be any other paths available to include that node. Thus, *this algorithm aims to utilize as many FPs as possible* in the circuit. Continuing to traverse the S Graph from $FF1$, $FF3$ and $FF4$ are chosen as single-input children in step 1. However, according to step 3, only $FF4$ will be chosen and kept in the SP since it has a lower cost. The edge between $FF1$ and $FF3$ will be removed at step 5, since $FF3$ will no longer be considered to be included in this SP. Moving on to traverse the S Graph from $FF4$, $FF5$ and $FF6$ are selected as new candidates that could be included in the SP at step 7. However, according to step 9, only $FF5$ will be kept in this SP since it has a lower cost. The edge connecting $FF4$ and $FF6$ will thus be removed at step 11. Also, the incoming edge connecting to $FF5$ will be removed at step 12 since $FF5$ should not be considered in any other SP. The algorithm will stop here since $FF5$ does not have any more children. However, it will start again from another candidate node until all the SPs are identified and stored in L_{SP}. At this stage, the S Graph in Figure 3.4(a) will be simplified to the S Graph in Figure 3.4(b), and from which, the next step of determining the order of SPs for SC construction in the scan synthesis process will follow.

Algorithm 1: SPIdentification

```
  Input  : $S\_Graph$
  Output: $L_{SP}$
1 Remove $ALL$ edges in $S\_Graph$ representing MUX insertion;
2 Remove any $feedback\ loop$;
3 Remove $ALL$ edges that violate custom constraints;
4 foreach candidate node $V_{start}$ in $S\_Graph$ do
5     Set $V_{start}$ visited;
6     $SP_{current}$ = TraverseSGraph($S\_Graph$, $V_{start}$, $SP_{empty}$);
7     $L_{SP} = L_{SP} \cup SP_{current}$;
  end
8 Return $L_{SP}$;
```

3.4 Constructing Scan Chains

After all the redundant edges in the S Graph are removed using *Algorithm 2*, the next step in the scan synthesis process is to connect the cost saving SPs in the S Graph for SC construction. It should be noted that to connect two SPs together to form an SC, a MUX will be needed. For example, to connect the SP starting from V_{start1} and V_{start2} in Figure 3.4(b), a MUX will be needed to connect $FF5$ and V_{start2}. Since we want to produce balanced SCs while minimizing the area and performance penalty, only a minimum number of MUXes should be added at this stage. An alternative technique to connect SPs for SC construction is shown in *Algorithm 3*. The first step in this algorithm is to determine the desired length of the SCs. This is because by considering the SC length during scan construction, slicing and trimming of SCs in the later stage will not be necessary in order to build balanced SCs. The optimal

Algorithm 2: TraverseSGraph

```
Input  : $S\_Graph$, $V_{start}$, $SP_{current}$
Output: $SP_{updated}$
1  $L_{single}$ = list of child nodes of $V_{start}$ with single input;
2  if ($L_{single} \neq NULL$) then
3      $V_{cheap} = V_i$ from $L_{single}$ with lowest cost;
4      $SP_{current} = SP_{current} \cup V_{cheap}$;
5      Remove $ALL$ other outgoing edge from $V_{start}$;
6      $SP_{updated}$ = TraverseSGraph($S\_Graph$, $V_{cheap}$, $SP_{current}$);
   else
7      $L_{multiple}$ = list of child nodes of $V_{start}$ with multiple inputs;
8      if ($L_{multiple} \neq NULL$) then
9          $V_{cheap} = V_i$ from $L_{multiple}$ with lowest cost;
10         $SP_{current} = SP_{current} \cup V_{cheap}$;
11         Remove $ALL$ other outgoing edges from $V_{start}$;
12         Remove $ALL$ other incoming edges to $V_{cheap}$;
13         $SP_{updated}$ = TraverseSGraph($S\_Graph$, $V_{cheap}$, $SP_{current}$);
       end
   end
14 Return $SP_{updated}$
```

SC length for a design can be calculated using Equation 3.1. After the desired SC length is obtained, the next step in the algorithm is to decide which PI should be used as SI in the design. This is done by steps 4-10 in *Algorithm 3*. By giving higher priority to PIs that are already connecting to some existing FPs, routing and gate overhead could be further reduced when they are reused as SIs. Only when such PIs cannot be found, the algorithm will randomly select any available PI as SI. Once an SI is chosen, the order of scan cells for that SC is determined by iteratively executing *Algorithm 4* at step 12.

$$SL_{desired} = \lceil \frac{TotalFF_{design}}{Num_{SC}} \rceil \tag{3.1}$$

Algorithm 4 determines which SP should be chosen and appended to the current SC if its length has not reached the upper bound yet. Every time when the algorithm needs to find a new SP for a chain, it looks at two parameters in the existing chain to make the decision. The first parameter that is considered is *the length of SP that is needed*. The algorithm will always try to find an SP that has length less than or equal to the difference between the desired SC length ($SL_{desired}$) and the current SC length ($SL_{current}$) as shown in steps 4 and 10. This is because we want to avoid slicing of SP as much as possible while producing balanced SCs. *The possible physical location of the SPs* is the second parameter that needs to be considered when finding candidate SP. When two separate SPs need to be connected together, a MUX will be added between the end of $SP1$ and the beginning of $SP2$. Moreover, when two memory elements are connected to each other indirectly (i.e. there are some combinational logic in-between the registers), they are more likely to be placed closer to each other during layout and routing. As a result, in order to reduce routing overhead, we would like to choose an SP that connects to $SP1$ indirectly as our candidate $SP2$ so that the routing overhead for the additional MUX can be minimized. This is why from steps 4-9, we are giving high priority to the SPs that are connected indirectly as candidate SPs. However, if SPs that are indirectly connected cannot be found, steps 10-15 of the algorithm will randomly choose an SP that is still to be chained. After the candidate SP has been chosen, it will need to be checked if any custom constraint is violated at step 16. These constraints for enhancing delay fault coverage or reducing VTD of scan designs will be discussed in Sections 3.5.1 and 4.3 respectively. If the chosen SP meets all the constraints, it will then be appended to the current SC. Finally, when the SC is long enough, it randomly picks a scan output port from a list of available scan outputs (L_{SO}) and appends it to the SC (step 19).

In both *Algorithms 3* and *4*, the routine *AppendSC* is used to append the selected SP into the current SC. This routine will append the SC until all FFs in the SP are inserted, or when the number of FFs inserted reaches a desired number. This way, the length of SC will never exceed the desired SC length, and thus, avoiding slicing of SC in the later stage of the scan synthesis process. After the order of scan cells in each SC are determined using the above algorithms, the RTL description of the design will be modified to incorporate the scan structure accordingly. An example of how a description is modified for scan chain insertion will be shown in Appendix A.

Algorithm 3: StartChaining

```
   Input  : $L_{SI}$, $L_{SO}$, $Num_{SC}$, $L_{SP}$
   Output: $L_{SC}$
1  Find optimal SC length $SL_{desired}$ of the design;
2  $Num_{current\_chain} = 1$;
3  while ($Num_{current\_chain} \leq Num_{SC}$) do
4      $L_{SI\_candidate}$ = SI that connects to $SP_{any}$ in $L_{SP}$;
5      if ($L_{SI\_candidate} \neq NULL$) then
6          $SI_{selected}$ = SI that connects to $SP_{longest}$ in $L_{SP}$;
7          $SC_{current} = SC_{current} \cup SI_{selected}$;
8          $SC_{current}$ = AppendSC($SC_{current}$, $SP_{longest}$);
       else
9          $SI_{selected}$ = Any scan input in $L_{SI}$;
10         $SC_{current} = SC_{current} \cup SI_{selected}$;
       end
11     Remove $SI_{selected}$ from $L_{SI}$;
12     $SC_{updated}$ = ChainFF($SC_{current}$, $SL_{desired}$, $L_{SO}$, $L_{SP}$);
13     $L_{SC} = L_{SC} \cup SC_{updated}$;
14     $Num_{current\_chain} = Num_{current\_chain} + 1$;
   end
15 Return $L_{SC}$;
```

The generated RTL scan design can then be interfaced with the RTL/logic synthesis tools (or RTL-to-GDSII flow [37]) with the DFT infrastructure in place.

In addition to reducing area and performance penalty of the generated scan structures, custom constraints can be inserted into the algorithms to tune the created scan structure for various objectives. In the following section, the extra constraints for enhancing delay fault coverage will be introduced. Furthermore, the scan structure can also be optimized to reduce the volume of test data, as discussed in Section 4.3.

Algorithm 4: ChainFF

```
Input  : $SC_{current}$, $SL_{desired}$, $L_{SO}$, $L_{SP}$
Output: $SC_{updated}$
1  $SL_{current}$ = length of $SC_{current}$;
2  if ($SL_{current} < SL_{desired}$ && $L_{SP} \neq NULL$) then
3      $SL_{diff} = SL_{desired} - SL_{current}$;
4      $L_{SP\_indirect}$ = All $SP_{indirect}$ in $L_{SP}$ with length $\leq SL_{diff}$;
5      if ($L_{SP\_indirect} \neq NULL$) then
6          $SP_{longest}$ = longest $SP_{indirect}$ in $L_{SP\_indirect}$;
       else
7          $L_{SP\_indirect}$ = All $SP_{indirect}$ in $L_{SP}$;
8          if ($L_{SP\_indirect} \neq NULL$) then
9              $SP_{longest}$ = longest $SP_{indirect}$ in $L_{SP\_indirect}$;
           else
10             $L_{SP\_random}$ = All $SP_{random}$ in $L_{SP}$ with length $\leq SL_{diff}$;
11             if ($L_{SP\_random} \neq NULL$) then
12                 $SP_{longest}$ = longest $SP_{random}$ in $L_{SP\_indirect}$;
               else
13                 $L_{SP\_random}$ = All $SP_{random}$ in $L_{SP}$;
14                 if ($L_{SP\_random} \neq NULL$) then
15                     $SP_{longest}$ = longest $SP_{random}$ in $L_{SP\_indirect}$;
                   end
               end
           end
       end
16     if ($SP_{longest}$ does not violate any custom constrains) then
17         $SC_{current}$ = AppendSC($SC_{current}$, $SP_{longest}$, $SL_{diff}$);
18         $SC_{updated}$ = ChainFF($SC_{current}$, $SL_{desired}$, $L_{SO}$, $L_{SP}$);
       end
   else
19     $SC_{updated} = SC_{current} \cup$ ($SO_{select}$ from $L_{SO}$);
   end
20 Return $SC_{updated}$;
```

3.5 Improving Delay Fault Coverage Through Functional Scan

Delay fault testing is gaining importance in the test community to compensate for the decreasing effectiveness of quiescent current-based testing for circuits manufactured in smaller process geometries. To facilitate delay fault testing using existing scan structures available in a design, the *skewed-load two-pattern test application strategy* can be used. However, as described in Section 1.4, due to the presence of pattern correlations, sufficient test quality of scan designs using this strategy cannot be achieved. In Section 3.5.1, a technique to improve delay fault coverage through consciously constructing functional scan structure will be introduced.

3.5.1 Constraints To Improve Delay Fault Coverage

Skewed-load test application strategy for two-pattern testing of scan circuits has the restriction of data correlation between test pattern pairs. This was already discussed in Section 1.4. To remove the above limitation, which leads to loss in delay fault coverage, we account for three constraints during the RTL scan synthesis process in *Algorithms 1 and 4*. The three constraints: *Vertical Constraint 1*, *Vertical Constraint 2* and *Horizontal Constraint*, and a sample scan structure built with these constraints is shown in Figure 3.5:

1. *Vertical constraint 1* tries to place two FFs in adjacent positions in the same scan chain only if they drive different logic cones;

2. *Vertical constraint 2* gives higher priority to FFs with larger bit index (for data paths) during FF selection;

3. *Horizontal constraint* spans FFs driving the same logic cone into different scan chains, if possible;

By using the *Vertical Constraint 1*, two FFs driving the same logic cone will be interleaved with FFs driving another logic cones, resulting an interleaved SC structure. In the example of Figure 3.5, the right most SC consists of FFs $\{A_3$, B_3, A_1,

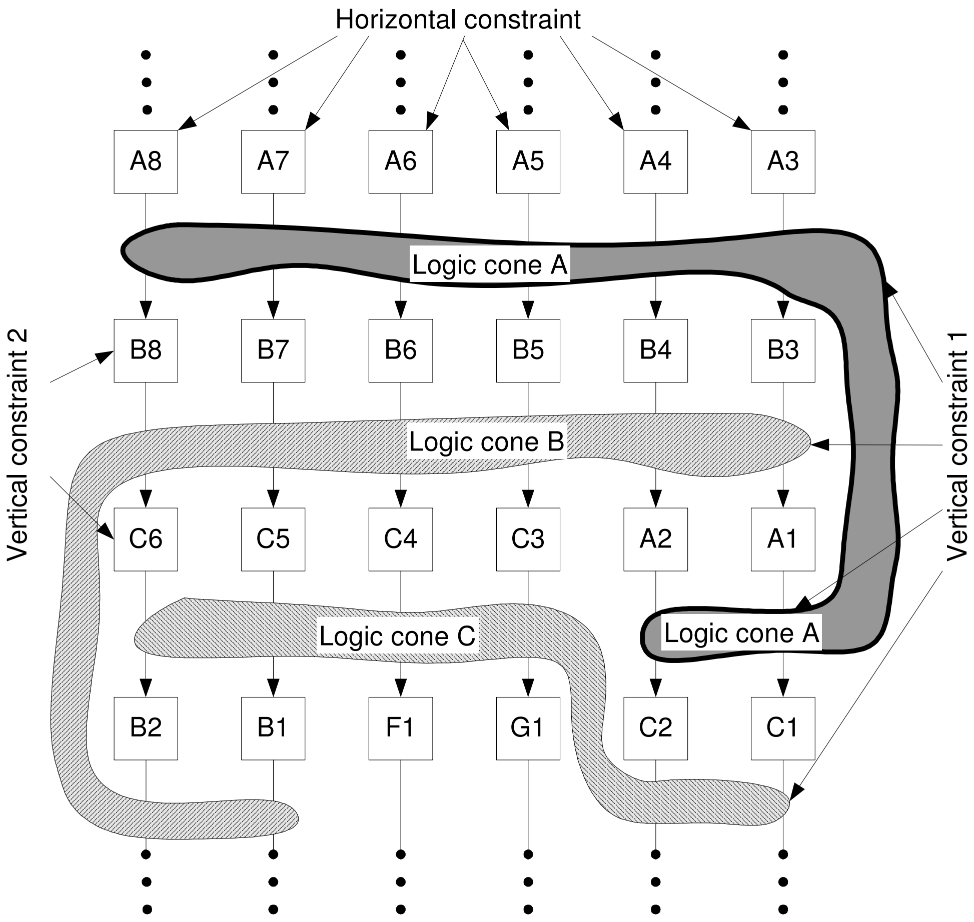


Figure 3.5: Considering delay fault testing constraints for scan chain designs at RTL

$C_1\}$. When targeting a transition fault in logic cone A, the pattern pair $\{V_1, V_2\}$ that will excite the fault can be loaded onto the SC by shifting the first pattern V_1 onto the FFs $\{A_3, A_1\}$, while using the other FFs $\{B_3, C_1\}$ as buffers to store the second pattern V_2. In this case, any arbitrary pattern pair can be loaded to the target logic cone A. Moreover, when a design has multiple SCs, the *Horizontal Constraint* can further enhance the delay fault coverage by spanning FFs driving the same logic cone onto multiple SCs, as illustrated in Figure 3.5. By applying this constraint in the SC construction process, FFs that drive the same logic cone are distributed onto different SCs, reducing the likelihood of these FFs to be put onto a single SC at later stage during SC generation. This allows the *Vertical Constraint 1* to be better satisfied throughout the complete scan synthesis process. The reason for having *Vertical Constraint 2* is also to aid the satisfaction of *Vertical Constraint 1*. In the example shown in Figure 3.5, after chaining FF set $\{A_8, \ldots, A_3\}$ in the first level of the SCs, without *Vertical Constraint 2*, the FF set $\{C_6, \ldots, C_1\}$ can be chosen to be placed onto the second level. In this case, *Vertical Constraint 1* would be violated since the FF set $\{B_8, \ldots, B_1\}$ would have to be placed in levels three and four. By better satisfying *Vertical Constraint 1* from the aid of *Vertical Constraint 2* and *Horizontal Constraint* in *Algorithms 1 and 4*, the restriction of skewed-load test application scheme due to pattern correlation for two-pattern delay fault testing can be eliminated, resulting higher delay fault coverage without the addition of any complex logic.

3.6 Experimental Results

We integrated the new constraints to the RTL scan synthesis process and performed experiments on a DSP core from the SCU benchmark set [1] and the ITC benchmark B14 [40] using commercial synthesis and ATPG tool flow [42]. The experimental results for area/performance analysis are shown in Tables 3.2 and 3.3. The testability results for both of the circuits can be found in Tables 3.4 and 3.5. In all the tables, the results are subjected to rounding error.

Tables 3.2 and 3.3 show the experimental results for area overhead and circuit performance of the DSP core and B14, when comparing the gate level full scan and the proposed RTL full scan. For both tables, Column 1 shows the timing constraints used for logic synthesis. Column 2 provides the total number of SCs and Columns 3, 5 and 7 indicate whether the timing constraints were met during synthesis for the non-scan circuit (i.e., no DFT included), the circuit with gate level scan and the RTL scan circuit. Columns 4 and 6 show the area overhead when compared to the non-scan circuit for gate level scan and RTL scan accordingly. Column 7 shows the difference in area overhead between gate level scan and RTL scan. For example, for the DSP core with a clock period of 10.7 ns, the original circuit, gate level scan and RTL scan circuits can all meet the timing constraint. One thing to note is that there are 639 FFs in the original circuit and gate level scan, while RTL scan has 664 FFs

Period	#	Original	Area OH				Δ
	of	(No DFT)	Gate level scan		RTL scan		
(ns)	SC	Timing Met?	%	Met?	%	Met?	%
10.70	8	Yes	6.01	Yes	6.18	Yes	-0.16
	16		6.01	Yes	6.10	Yes	-0.09
	24		6.01	Yes	6.53	Yes	-0.51
10.65	8	No	4.80	No	5.72	Yes	-0.92
	16		4.80	No	6.75	Yes	-1.95
	24		4.80	No	5.70	Yes	-0.90
10.60	8	Yes	5.94	No	6.78	Yes	-0.84
	16		5.94	No	7.90	Yes	-1.96
	24		5.94	No	8.26	Yes	-2.32
10.55	8	No	5.39	No	6.39	Yes	-1.00
	16		5.39	No	6.10	Yes	-0.70
	24		5.39	No	5.89	Yes	-0.50
10.50	8	No	6.41	No	7.57	Yes	-1.16
	16		6.41	No	7.36	Yes	-0.94
	24		6.41	No	7.23	Yes	-0.82
10.45	8	No	5.90	No	6.38	Yes	-0.49
	16		5.90	No	6.96	Yes	-1.06
	24		5.90	No	7.02	No	-1.12
10.40	8	No	5.61	No	5.77	Yes	-0.16
	16		5.61	No	5.87	Yes	-0.26
	24		5.61	No	6.42	Yes	-0.81
10.35	8	No	5.94	No	7.09	No	-1.15
	16		5.94	No	6.94	No	-1.00
	24		5.94	No	6.93	Yes	-0.99
10.30	8	No	6.16	No	8.13	No	-1.98
	16		6.16	No	7.18	No	-1.03
	24		6.16	No	8.87	Yes	-2.71

Table 3.2: Area/performance for the DSP core with delay constraints

after synthesis. This is because without RTL scan, the synthesis tool can optimize the circuit by removing redundant FFs. However, these redundant FFs will not be removed in RTL scan since these FFs are now included in the scan paths. Despite the presence of these redundant FFs, the average increase in area overhead for RTL scan when compared to that of gate level scan is only 1.02% for the DSP core. In the case of B14, the area overhead is, however, *reduced* on average by 3.55%. This is because, the scan synthesis process tries to reduce area overhead of the scan design by reusing existing FPs as SPs in the circuit. In addition, by constructing SCs at the RTL, the synthesis tool can optimize test logic and functional logic concurrently, thus

Period	Number of	Original (No DFT)	Area OH				Δ
			Gate Full		RTL Full		
(ns)	SC	Timing Met?	%	Met?	%	Met?	%
5.75	4	Yes	13.84	Yes	5.63	Yes	8.21
	8		13.85	Yes	10.28	No	3.57
	12		13.85	Yes	8.98	Yes	4.88
5.70	4	Yes	13.29	Yes	7.58	No	5.71
	8		13.29	Yes	6.33	No	6.96
	12		13.29	Yes	11.99	No	1.30
5.65	4	No	4.97	No	0.86	Yes	4.11
	8		4.97	No	6.78	No	-1.81
	12		4.97	No	4.37	No	0.60
5.60	4	Yes	5.19	Yes	0.32	Yes	4.87
	8		5.19	Yes	1.36	No	3.83
	12		5.19	Yes	1.29	Yes	3.90
5.55	4	Yes	10.29	Yes	7.91	No	2.39
	8		10.29	Yes	10.01	No	0.28
	12		10.29	Yes	8.08	No	2.21
5.50	4	No	4.66	No	-1.63	Yes	6.29
	8		4.66	No	2.33	No	2.32
	12		4.66	No	0.31	Yes	4.35

Table 3.3: Area/performance for B14 with delay constraints

improving the timing of the design. In the case of the DSP core, RTL scan with 24 SCs helps improve the timing to 10.3 ns, an improvement of 2.91% when compared to the original circuit, which has a timing of 10.6 ns. For B14, RTL scan with 4 and 12 SCs helps improve the timing by about 1% to 5.5 ns when compared to the original circuit, which has a timing of 5.55 ns.

Tables 3.4 and 3.5 show the testability results for the DSP core and B14 generated by a commercial ATPG tool [42]. The timing constraints and number of SCs are listed in columns 1 and 2 respectively. The column labeled AU represents ATPG untestable faults, which are faults that are untestable due to the limitation of scan cell arrangement. FC corresponds to fault coverage for transition faults, CTP denotes the number of compressed test patterns, and CPU represents test generation time in seconds. The readers should note that the total faults for the circuits with different timing constraints and number of SCs are different due to the differences in logic blocks that were synthesized from the RTL description. It can be seen from Table 3.4 that the proposed method consistently improves the delay fault coverage to over

Period	SC	Gate level scan				RTL scan				Δ			
(ns)	#	AU	FC	CTP	CPU	AU	FC	CTP	CPU	AU	FC	CTP	CPU
10.70	8	1563	95.82	315	191.62	178	98.08	294	161.41	1385	2.26	21	30.21
	16	2193	93.98	320	209.57	209	98.22	292	168.83	1984	4.24	28	40.74
	24	2488	92.91	375	275.94	655	97.53	287	167.68	1833	4.62	88	108.26
10.65	8	1564	95.58	298	270.89	177	98.32	300	164.93	1387	2.74	-2	105.96
	16	2188	93.70	307	327.53	203	98.25	304	165.15	1985	4.55	3	162.38
	24	2520	92.44	350	409.28	656	97.45	272	158.46	1864	5.01	78	250.82
10.60	8	1600	95.52	311	197.59	177	98.20	297	156.21	1423	2.68	14	41.38
	16	2275	93.80	312	210.11	202	98.40	310	214.28	2073	4.60	2	-4.17
	24	2563	92.46	379	298.91	653	97.51	295	178.15	1910	5.05	84	120.76
10.55	8	1558	95.41	301	171.57	180	98.41	324	222.67	1378	3.00	-23	-51.10
	16	2238	93.67	328	218.88	207	98.19	306	160.77	2031	4.52	22	58.11
	24	2499	92.62	356	266.71	662	97.51	264	138.44	1837	4.89	92	128.27
10.50	8	1583	95.93	301	151.54	172	98.28	310	133.02	1411	2.35	-9	18.52
	16	2183	94.16	320	175.87	208	98.18	305	186.21	1975	4.02	15	-10.34
	24	2492	93.24	361	223.95	662	97.56	271	142.91	1830	4.32	90	81.04
10.45	8	1550	95.58	297	182.77	185	98.23	321	194.92	1365	2.65	-24	-12.15
	16	2183	93.66	310	230.50	212	98.29	316	200.87	1971	4.63	-6	29.63
	24	2499	92.54	377	312.15	660	97.74	290	168.53	1839	5.20	87	143.62
10.40	8	1566	95.63	276	190.56	172	98.36	305	223.38	1394	2.73	-29	-32.82
	16	2230	93.75	300	200.62	202	98.17	291	177.17	2028	4.42	9	23.45
	24	2450	92.67	337	268.89	667	97.37	299	171.25	1783	4.70	38	97.64
10.35	8	1539	95.82	277	182.96	178	98.18	323	218.51	1361	2.36	-46	-35.55
	16	2192	94.01	313	222.35	208	98.13	290	168.34	1984	4.12	23	54.01
	24	2453	92.96	364	293.44	658	97.55	284	199.09	1795	4.59	80	94.35
10.30	8	1577	95.83	325	175.72	176	98.27	318	185.27	1401	2.44	7	-9.55
	16	2222	93.97	348	220.52	205	98.11	315	211.44	2017	4.14	33	9.08
	24	2502	92.91	390	262.89	658	97.49	277	154.46	1844	4.58	113	108.43

Table 3.4: Skewed-load delay fault coverage results for the DSP core

Period	SC	Gate level scan				RTL scan				Δ			
(ns)	#	AU	FC	CTP	CPU	AU	FC	CTP	CPU	AU	FC	CTP	CPU
5.75	4	295	98.06	696	87.03	1134	96.03	652	40.06	-839	-2.03	44	46.97
	8	344	97.97	702	87.24	1061	96.12	661	80.77	-717	-1.85	41	6.47
	12	1011	95.20	689	95.95	1194	95.64	627	18.27	-183	0.44	62	77.68
5.70	4	261	98.28	717	60.93	1081	95.97	645	60.92	-820	-2.31	72	0.01
	8	304	98.11	727	67.05	1039	96.27	667	62.71	-735	-1.84	60	4.34
	12	974	95.33	712	76.46	1208	95.73	666	76.16	-234	0.40	46	0.30
5.65	4	273	98.09	674	104.04	1067	96.33	671	71.75	-794	-1.76	3	32.29
	8	331	97.87	698	104.90	1181	95.78	664	113.88	-850	-2.09	34	-8.98
	12	1002	94.46	673	121.91	1135	95.73	657	44.34	-133	1.27	16	77.57
5.60	4	251	98.24	691	81.95	1059	95.94	659	103.64	-808	-2.3	32	-21.69
	8	304	98.07	692	79.78	1084	96.14	671	89.79	-780	-1.93	21	-10.01
	12	989	94.74	673	91.81	1043	96.06	678	92.47	-54	-1.32	-5	0.66
5.55	4	261	98.37	669	94.13	1085	96.11	660	90.05	-824	-2.26	9	4.08
	8	312	98.04	698	95.96	1057	96.44	659	95.96	-745	-1.60	39	0.00
	12	954	94.76	672	109.62	1223	95.60	638	82.59	-269	-0.84	34	27.03
5.50	4	329	97.97	695	112.33	1026	96.27	671	73.95	-697	-1.70	24	38.38
	8	970	94.84	684	125.41	961	96.35	640	68.10	9	1.51	44	57.31
	12	572	96.87	689	113.51	1109	95.79	659	62.77	-537	-1.08	30	50.74

Table 3.5: Skewed-load delay fault coverage results for B14

Period (ns)	SC #	TF	AU	Faults in test paths	FC with faults in test paths	FC without faults in test paths
10.70	8	64942	178	121	98.08	98.27
	16	65382	209	123	98.22	98.41
	24	65460	655	123	97.53	97.72
10.65	8	65466	177	119	98.32	98.50
	16	65524	203	118	98.25	98.43
	24	65238	656	127	97.45	97.64
10.60	8	64800	177	121	98.20	98.39
	16	67140	202	118	98.40	98.58
	24	66290	653	121	97.51	97.69
10.55	8	67360	180	121	98.41	98.59
	16	66610	207	120	98.19	98.37
	24	65470	662	126	97.51	97.70
10.50	8	65324	172	125	98.28	98.47
	16	66610	208	122	98.18	98.36
	24	65670	662	126	97.56	97.75
10.45	8	66284	185	125	98.23	98.42
	16	66604	212	124	98.29	98.48
	24	66724	660	123	97.74	97.92
10.40	8	67354	172	116	98.36	98.53
	16	66170	202	116	98.17	98.35
	24	65934	667	128	97.37	97.56
10.35	8	67804	178	121	98.18	98.36
	16	67340	208	121	98.13	98.31
	24	67178	658	121	97.55	97.73
10.30	8	67412	176	119	98.27	98.45
	16	68366	205	118	98.11	98.28
	24	66162	658	121	97.49	97.67

Table 3.6: Adjusted skewed-load delay fault coverage results for the DSP core with RTL scan when functionally redundant faults in the test paths are discarded

Period (ns)	SC #	TF	AU	Faults in test paths	FC with faults in test paths	FC without faults in test paths
5.75	4	34210	1134	1112	96.03	99.28
	8	35362	1061	1029	96.12	99.03
	12	33042	1194	1139	95.64	99.09
5.70	4	33412	1081	1055	95.97	99.13
	8	34414	1039	1035	96.27	99.28
	12	34608	1208	1153	95.73	99.06
5.65	4	34426	1067	1031	96.33	99.32
	8	33902	1181	1137	95.78	99.13
	12	34926	1135	1080	95.73	98.82
5.60	4	34996	1059	1025	95.94	98.87
	8	36382	1084	1051	96.14	99.03
	12	34414	1043	1008	96.06	99.06
5.55	4	34894	1085	1015	96.11	99.02
	8	36004	1057	1022	96.44	99.28
	12	34958	1223	1194	95.60	99.02
5.50	4	35202	1026	989	96.27	99.08
	8	33156	961	908	96.35	99.09
	12	34026	1109	1086	95.79	98.98

Table 3.7: Adjusted skewed-load delay fault coverage results for the B14 with RTL scan when functionally redundant faults in the test paths are discarded

97%, with an average improvement of 3.9% when compared with gate level scan. This is mainly due to the reduction of ATPG untestable faults because of the scan cells arrangement according to the constraints proposed in our approach. However, the above improvement in ATPG untestable faults does not account for the undetectable faults in the test paths, which are unique for functional scan and, if present, will not affect the functional mode of operation. For example, in the case of B14 (Table 3.5), the presence of these undetectable faults in the test paths will decrease (on average) the delay fault coverage of RTL scan by 0.94% when compared with gate level scan. Because these undetectable faults in the test paths cannot affect the functional mode of operation, they can be discarded from the fault coverage computation. Tables 3.6 and 3.7 show the delay fault coverage results for the DSP core and B14 after they are adjusted by discarding these AU faults. In these tables, the timing constraints and the number of SCs are listed in columns 1 and 2 respectively. The column labeled TF corresponds to the total number of delay faults in the synthesized circuit, and AU represents ATPG untestable faults, which includes faults that are in the test paths. Column 5 gives the number of faults that are in the test paths. Finally, the delay fault coverage before and after the adjustment are shown in Columns 6 and 7 respectively. As one can see from the results, after re-calculating the delay fault coverage for RTL scan by discarding the faults in the test paths, which, if present, will not affect the functional operation of the circuit, the average delay fault coverage is improved to 98.18% and 99.09% for the DSP core and B14 respectively. Furthermore, due to the interleaved SC structure that is generated, there could be multiple independent logic cones presented in a single scan chain, which gives ATPG more flexibility to merge patterns targeting faults in different logic cones into a single test pattern. As one can see from the experimental results for the DSP core and B14 in Tables 3.5 and 3.4, the average reduction in number of test patterns for RTL scan compared with gate level scan is about 29 and 34 respectively. In addition, our proposed solution has also improved the test generation time by an average of 57.44 and 21.29 seconds for the DSP core and B14 accordingly. It is also important to note that the computational time for scan insertion in the RTL code is in the range of tens of seconds on a Pentium-M at 1.3 GHz.

3.7 Summary

This chapter presented an alternative technique to [19] for constructing functional scan chains at the RTL. By extracting functional information from the RTL description using CDFGs and the S Graph, functional paths that can lead to potential savings in area and performance overhead, when they are reused as scan paths, are identified. Moreover, the additional constraints that are integrated into the scan synthesis process eliminate the restriction of data correlation between test pattern pairs. As a result, the generated scan structure can lead to enhanced delay fault coverage, low area overhead, as well as improved timing. It is important to note that, if the scan chains are constructed after the logic or physical synthesis steps in the design flow, imposing constraints on the scan chain order may conflict with timing closure or wiring minimization objectives. However, this is not the case for the proposed solution for enhancing delay fault coverage, where the constraints on the functional scan path construction are included in the RTL description *prior to* logic synthesis. As a consequence, both the test and functional logic and interconnect are subsequently optimized simultaneously during logic and physical synthesis.

Chapter 4

Test Data Compression Through Functional Scan

Due to the decrease in design complexity at the RTL, analysis of circuits for functional information becomes feasible. The extracted information can then be used to identify cost saving FPs that can be reused as SPs in a design. The previous chapter has already shown how by reusing these cost saving FPs as SPs during SC construction at the RTL, one could reduce area overhead, improve timing and enhance the delay fault coverage of a scan circuit.

The large volume of test data (VTD) is another problem associated with the scan method. It was already mentioned in Section 2.3.4 that the Illinois Scan Architecture (ISA) can reduce VTD while reusing the existing scan structure in a design. However, due to the correlations of patterns in multiple SCs in the broadcast mode, the effectiveness of test data compression from the ISA is limited. In Section 4.1, a technique to eliminate the correlations of patterns by inserting dummy cells into the scan design is discussed. Due to the excessive area overhead of this technique, Section 4.2 introduces two new reconfigurable ISAs. Section 4.3 then introduces a set of new constraints to be incorporated to the scan synthesis process that was detailed in Chapter 3. Experimental results shown in Section 4.4 indicate that the VTD for functional scan synthesized at RTL can be effectively reduced, regardless of the resulting logic network or the subsequently generated structural test sets.

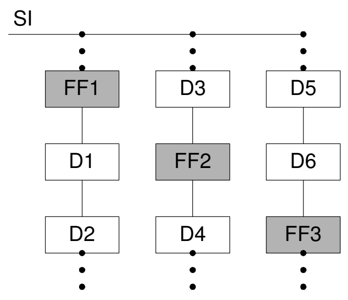


Figure 4.1: Illinois scan with dummy cells

4.1 Illinois Scan With Dummy Cells

Our first attempt to improve the stuck-at fault coverage for the ISA was to insert dummy cells into SCs to eliminate correlations between test patterns by preventing FFs that drive the same logic cone to be placed at the same scan depth in different SCs. As shown in Figure 4.1, FF1, FF2 and FF3 drive the same logic cone. By inserting dummy cells $\{D1, \ldots, D6\}$ into the SCs, the FFs are no longer at the same scan depth in multiple chains, thus allowing arbitrary patterns to be fed to the target logic cone in the broadcast mode. This eliminates the need for the serial test patterns which increases both the VTD and the test application time of scan designs. Since the inserted dummy cells do not serve any purpose and appear transparent to the circuit functionality in the normal operation mode, these redundant FFs serve as area overhead for the purpose of test only.

In order to minimize the number of dummy cells needed to improve the stuck-at fault coverage for the ISA, the algorithm will insert a dummy cell to an SC only if it is not able to find an un-chained FF that does not drive the same logic cone as the FFs placed at the same scan depth in all the other SCs. The algorithm was applied to the ITC benchmark circuit *B14* [40] with 244 FFs. Figure 4.2 shows the number of dummy cells inserted, while Figure 4.3 illustrates the actual test data compression ratio of B14 with different number of SCs driven by 1, 2 and 4 SIs with the ISA. The actual compression ratio is calculated by Equation 4.1. The reader should note that the targeted compression ratio for Illinois Scan with no serial test patterns is $\frac{Number\ of\ SCs}{Number\ of\ SIs}$.

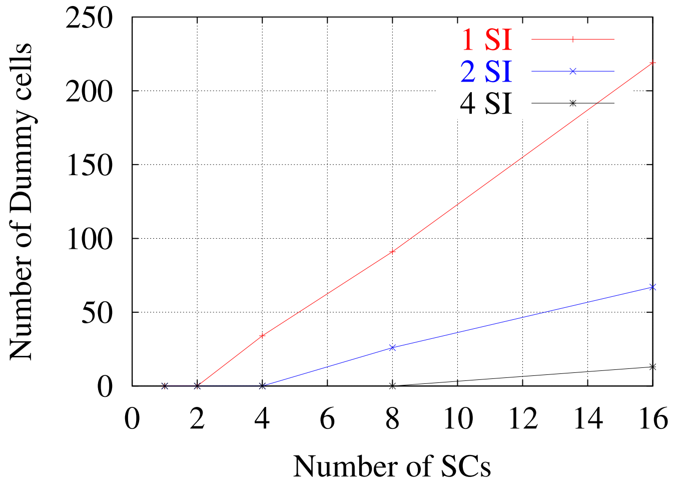


Figure 4.2: Area overhead for B14

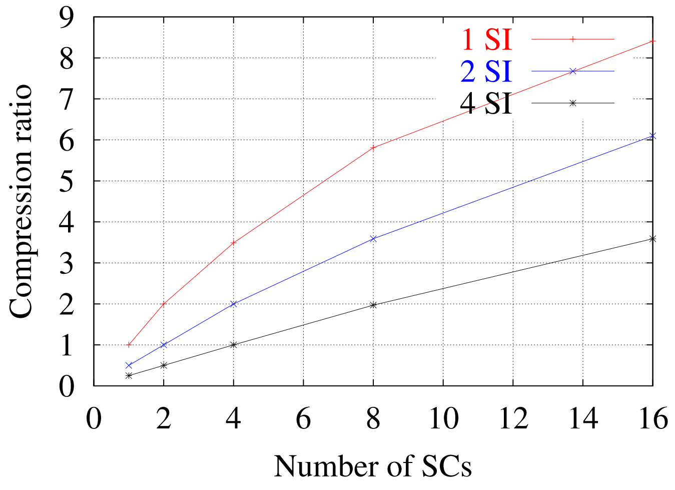


Figure 4.3: Compression ratio for B14

$$compression\ ratio = \frac{Total\ FFs\ excluding\ dummy\ cells}{(Length\ of\ longest\ SC) * (Number\ of\ SIs)} \tag{4.1}$$

Although a compression ratio of over 8X can be obtained for B14 when there are 16 SCs driven by a single SI using ISA (see Figure 4.3), there are over 200 dummy cells inserted to the design, which only has 244 FFs before scan insertion (see Figure 4.2). In order words, this method reduces test data volume and test application time while achieving the desired stuck-at fault coverage at the expense of having an undesirably large area overhead. On the other hand, by using multiple SIs, the number of dummy cells needed can be reduced. For instance, only 12 dummy cells are needed for B14 when using four SIs to drive 16 SCs (see Figure 4.2). However, the resulting compression ratio is only 3.6X in this case (see Figure 4.3). Having shown that dummy cell insertion is not capable of improving the fault coverage without a significant area penalty, we introduce two new Reconfigurable Illinois Scan Architectures (RISAs) that can effectively reduce the VTD for scan designs in the next section.

4.2 Reconfigurable Illinois Scan Architectures

The benefits of reducing the test data volume and the test application time of scan designs with the ISA are mainly contributed by shifting the same set of test data into multiple SCs from a single SI. However, when FFs that drive the same logic cone are placed at the same scan depth in different SCs, the possible set of test patterns this logic cone can receive will be limited, thus reducing the fault coverage of the design. Although the serial mode in the ISA can help improve the fault coverage, the serial test mechanism prolongs the test application time and increases the VTD. This problem can be solved by using the two RISAs with multiple broadcast modes and different SC lengths in each mode.

4.2.1 Reconfigurable Illinois Scan Architecture

The purpose of the RISA is to eliminate correlations between patterns in multiple SCs by employing multiple broadcast modes. Figure 4.4 [1] illustrates an example of the proposed architecture. In this example, there are 24 FFs equally distributed in 8 SCs. It is assumed that FF2 and FF5 drive the same logic cone, while FF14, FF17, FF20 and FF23 drive another logic cone. In mode 0 when the test select signals $\{TS2, TS1, TS0\}$ have the values of $\{0, 0, 0\}$, the architecture functions as the original ISA in the broadcast mode, where a single SI drives 8 SCs. However, when in mode 1, when the TS signals change to $\{0, 0, 1\}$, the SCs are reconfigured such that the adjacent SCs are combined into a single SC, resulting in 4 SCs with length of 6 to be driven by the same SI. In this mode, the conflict between FF2 and FF5 will be eliminated since they will be at different scan depths in the same SC. When the TS signals change to $\{0, 1, 1\}$ (mode 2), the SCs will be reconfigured again such that only 2 SCs with length of 12 will be driven by the same SI. In this mode, all the conflicts between FF14, FF17, FF20 and FF23 are eliminated. Finally, in mode 3 with $\{TS2, TS1, TS0\}$ equal to $\{1, 1, 1\}$ we will end up with a single scan chain where as long as a test pattern exist for a fault it will be generated. *Note, although the test data volume is increased every time the SCs are reconfigured, the rate of increase of the RISA is lower when compared to that of the serial broadcast mode in the original ISA.*

If there are 2^n SCs in a design, then the total number of broadcast modes in the RISA is $n + 1$ (including the serial scan mode). This is illustrated in Figure 4.5 where there are four SCs, which can be configured in three modes. Whenever the RISA is reconfigured, the adjacent SCs will be chained together as one long SC. There will be n TS signals, whose values can be stored in n state elements, which is negligible from the added area standpoint. The update of the TS signals can be done in between test sessions, with different broadcast modes, using a small on-chip controller programmable through the boundary scan interface. The decimal equivalents of the $n + 1$ values for the n TS signals are $2^i - 1$, with i from 0 to n.

[1]The left data input of the 2 to 1 MUXes is selected by the driving the control signal to 1

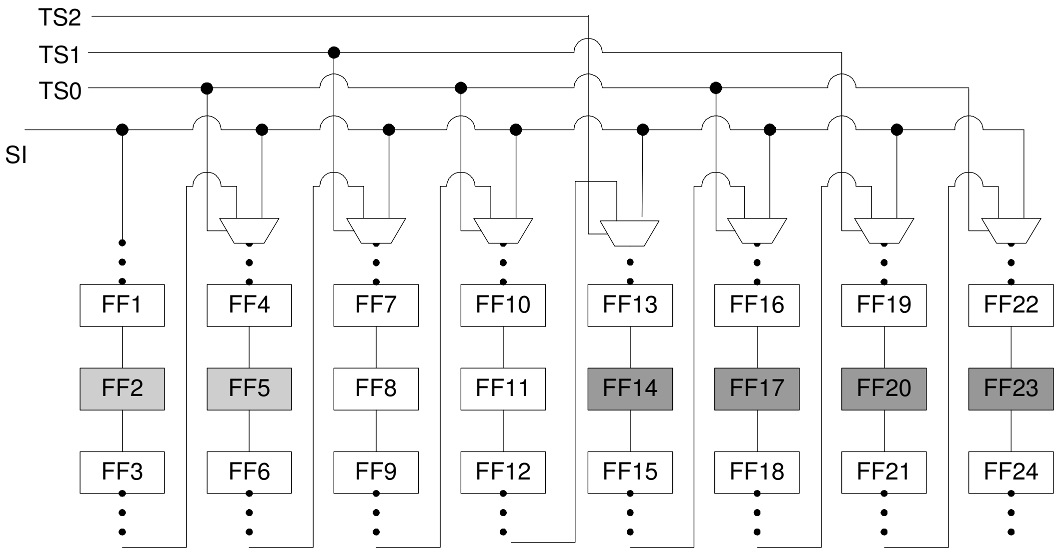


Figure 4.4: Reconfigurable Illinois scan

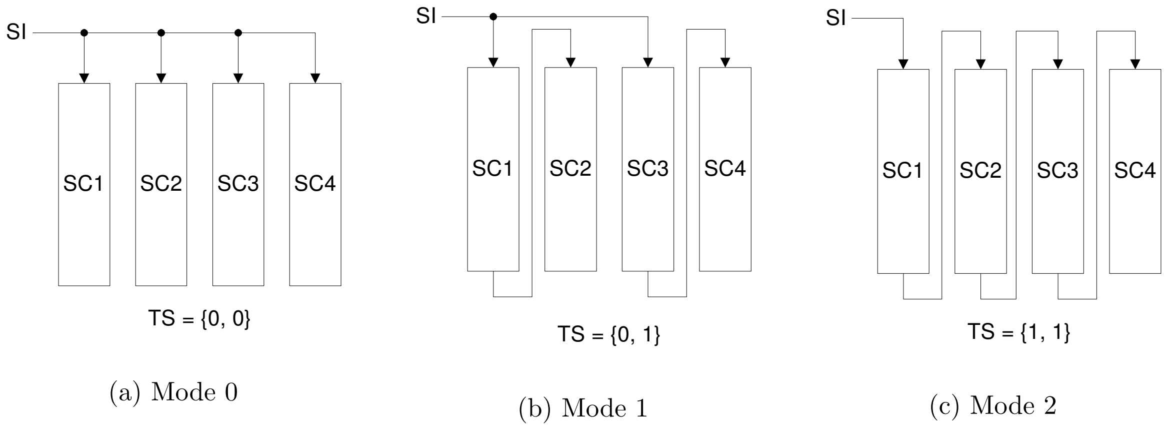


Figure 4.5: Different broadcast modes for the RISA

4.2.2 Reconfigurable Illinois Scan With Inversion

In this section, a modified version of RISA will be introduced. This new architecture can further improve the effectiveness of test data compression by inserting a negligible amount of extra logic to the original RISA. An example of the modified RISA is shown in Figure 4.6. In this architecture, a set of exclusive-OR (XOR) gates are inserted into the original RISA shown in Figure 4.4. These XOR gates are controlled by the input signal INV such that they function as inverters when the value of INV is set to a logic 1. Hence, the test patterns that are shifted into the selected SCs can be inverted. Recall from Section 4.2.1 that the RISA eliminates correlations between patterns in multiple SCs by employing various broadcast modes such that SCs are combined together in different modes. For example, in Figure 4.6, FFs {14, 17, 20, 23} are conflicting each other in SCs {5, 6, 7, 8}. If for detecting a fault the FFs {14, 17, 20, 23} must be assigned to {0, 1, 1, 0} then to satisfy the above values in the RISA proposed in the previous section (Figure 4.4) the architecture needs to be reconfigured until the four SCs are combined into a single SC. However, when using the modified RISA, the values can be satisfied when SCs {5, 6}, and SCs {7, 8} are combined as two SCs using the inverting property of the architecture. Thus, the VTD can be reduced by allowing multiple SCs to be driven by the same SI with the aid of the inverting capability of the modified RISA. For a design with n SCs, in addition to the $n + 1$ broadcast modes (including the serial scan mode) of operation, another n broadcast modes with inverting inputs to adjacent SCs can be configured in the modified RISA with inversion ability.

The XOR gates must be aligned such that for every pair of SCs *in any configuration*, there will be one XOR gate driving one of the SCs. For 2^n scan chains the questions is where to place the 2^{n-1} XOR gates, such that the scan cells at the same scan depth level in any two neighboring scan chains, in any broadcast mode, will be uncorrelated? An example of incorrect assignment of XOR gates is shown in Figure 4.7. In this example, the XOR gates are assigned to $SC2$ and $SC4$. Although this setting does provide inverting capability to adjacent SCs to eliminate correlations between patterns between the SC pairs {$SC1$, $SC2$} and {$SC3$, $SC4$} as shown in Figure 4.7(a), the inverting capability is lost when the RISA is reconfigured as shown

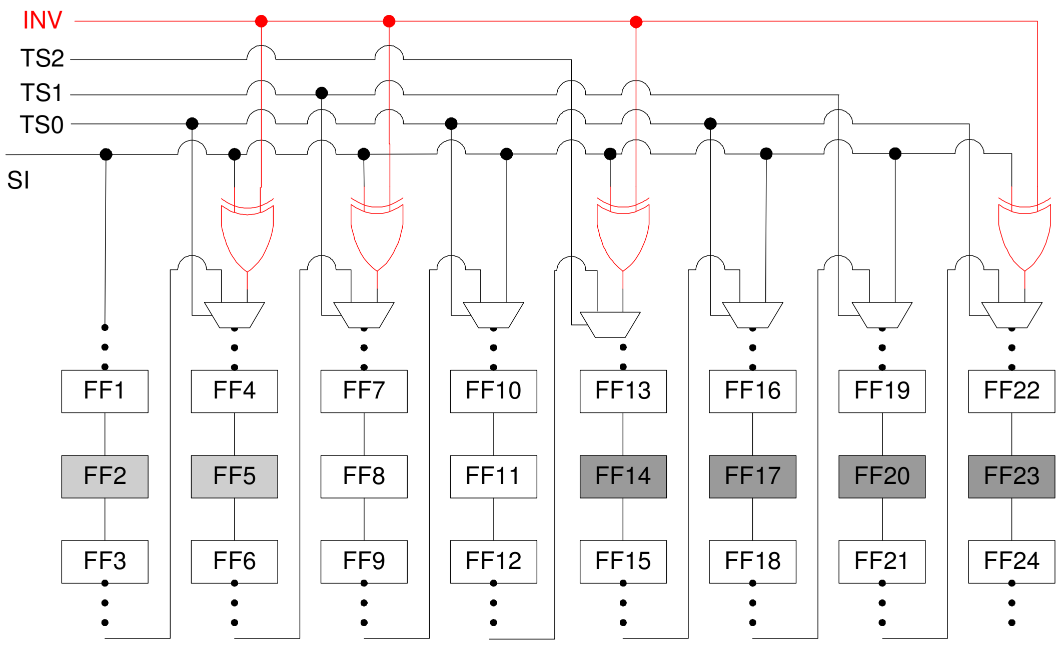
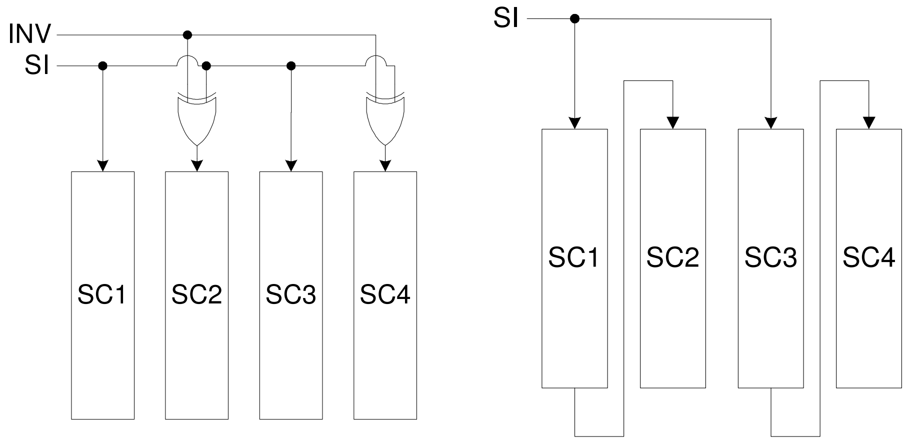


Figure 4.6: Reconfigurable Illinois scan with inversion

in Figure 4.7(b). The XOR gates are no longer present *at the beginning of a SC for pattern inversion from the SI*. For the correct assignment, the XOR gates are assigned to $SC2$ and $SC3$ in Figure 4.8(a). After reconfiguration in a new broadcast mode, the XOR gate of $SC3$ is kept at the beginning of the combined SC. Thus, the test patterns from the SI can be inverted for the combined SC in the new configuration as illustrated in Figure 4.8(b). Table 4.1 shows the assignment of XOR gates for a different number of SCs. For example, in the case of 4 SCs, the binary pattern 0110 indicates that XOR gates for inversion should be placed at the beginning of $SC2$ and $SC3$. For 2^n scan chains, the binary pattern for the assignment of XOR gates is given by the truth table of the n input XOR gate.

Number of SCs	Assignment of XOR gates
4	0110
8	01101001
16	0110100110010110

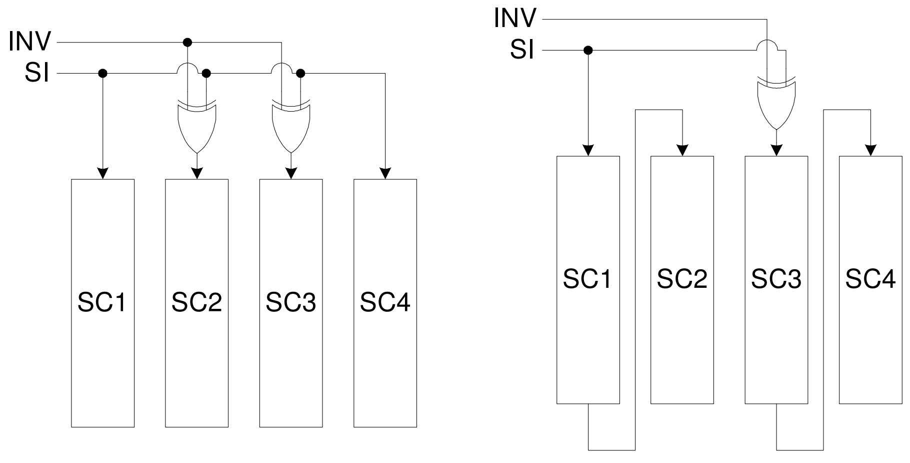
Table 4.1: Assignment of XOR gates for designs with the RISA with inversion



(a) Mode 0

(b) Mode 1

Figure 4.7: Incorrect assignment of XOR gates in the RISA with inversion



(a) Mode 0

(b) Mode 1

Figure 4.8: Correct assignment of XOR gates in the RISA with inversion

Both of the proposed RISAs improve the effectiveness in test data compression of scan designs by employing multiple broadcast modes in order to reduce correlations between patterns in different SCs. However, it should be noticed that if majority of correlations cannot be removed during the early broadcast modes of the reconfiguration process, the effectiveness in compressing test data using the proposed RISAs will be reduced. Hence, it is beneficial to arrange FFs in the SCs such that the appearance of correlation can be reduced in the first place. In the following section, two constraints that aid the FP selection process during scan construction will be introduced. By integrating the new constraints in the scan synthesis process described in Chapter 3, functional scan structures that are optimized for the RISAs can be obtained. As a result, not only the VTD can be decreased, the advantages of reduced area overhead and improved performance of functional scan designs can be retained.

4.3 Constraints For Functional Scan Construction at RTL To Reduce Volume of Test Data

As shown in the previous chapter, by introducing test infrastructure prior to RTL/logic synthesis and ATPG, it is ensured that test and functional logic are generated simultaneously, which may benefit the timing closure. To achieve this, the RISA must be inserted in the RTL description, which, consequently, asks for an investigation for extra constraints in the functional scan synthesis algorithms that account for the specific features of the RISA architectures proposed in the previous section.

It was mentioned in Section 2.3.4 that whenever FFs that drive the same logic logic are placed at the same scan depth in multiple scan chains, the fault coverage in the broadcast mode will be reduced. As a result, in order to improve the effectiveness in compressing test data of the generated RISA, the following constraints are needed in the scan synthesis process in *Algorithms 1 and 4*:

> Constraint 1: FFs that drive the same logic cone must not be placed at the same scan depth in multiple SCs if possible.

Constraint 2: FFs that drive the same logic cone must be placed in the same SC if possible.

Constraint 3: FFs that drive the same logic cone should be placed in adjacent SCs if both constraints 1 and 2 cannot be satisfied.

In order to better explain the constraints, an illustrative example is given in Figure 4.9. In this example, four sets of FFs {(1, 2, 3, 5, 6), (4), (7, 8, 9), (10, 11, 12)} are driving four different logic cones. FFs {1, 4, 7, 10} can be placed at the same scan depth in multiple SCs, as illustrated in Figure 4.9. In order to aid the satisfaction of constraint 1, constraint 2 is introduced since when FFs that drive the same logic cone are placed in the same SC, satisfaction of constraint 1 is facilitated. This is shown in Figure 4.9 where FFs {7, 8, 9} and FFs {10, 11, 12}, drive different logic cones. When both constraints 1 and 2 fail to be satisfied, constraint 3 will try to put the conflicting FFs in adjacent SCs. This is because if the conflicting SCs are placed adjacent to each other, when the RISA changes the broadcast mode (from 4 to 2 SCs driven by one SI in Figure 4.9) the adjacent SCs will be combined into a single SC. Thus, the conflicts between FFs, which occur in the broadcast mode to 2^{i+1} SCs will be removed when RISA is reconfigured to the broadcast mode to 2^{i} SCs. Furthermore, For the RISA with inversion from Figure 4.6, the change of the broadcast mode from 2^{i+1} to 2^{i} may not be necessary, because the removal of conflicts may occur by exploiting the inversion ability.

4.4 Experimental Results

To demonstrate the effectiveness of the proposed RISAs, we have adapted the scan synthesis algorithms from Chapter 3 and applied them to the DSP core from the SCU benchmark set [1] and the ITC99 benchmark *B14* [40]. There are a maximum of 32 SCs for the DSP core, and 16 SCs for the B14. For synthesis/ATPG, we have used a commercial tool flow [42]. It is important to note that the computational time for scan insertion is in the range of tens of seconds on a Pentium-M at 1.3 GHz, which justifies the advantage of analyzing the developing the scan infrastructure at the RTL.

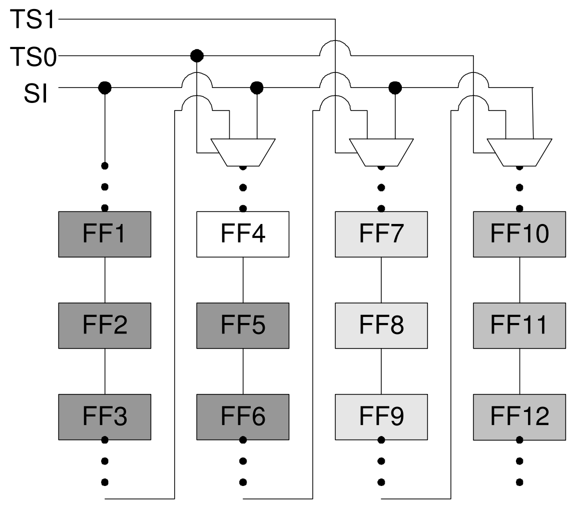


Figure 4.9: Illustrative example of Illinois scan construction

4.4.1 Reconfigurable Illinois Scan

Tables 4.2 and 4.3 show the experimental results for area overhead and circuit performance for the DSP core and B14 respectively, when comparing the non-scan circuit with the gate level full scan and the proposed RISA with scan inserted at RTL. In both tables, column 1 shows the timing constraints used for logic synthesis. Column 2 provides the total number of SCs and Columns 3, 5 and 7 indicate whether the timing constraints were met during synthesis for the non-scan circuit (i.e., no DFT included), the circuit with gate level scan and the RTL scan circuit. Columns 4 and 6 show the area overhead when compared to the non-scan circuit for gate level scan and RTL scan accordingly. Column 8 shows the difference in area overhead between gate level scan and RTL scan. One thing to note is that there are 639 FFs in the original circuit and gate level scan, while RTL scan has 664 FFs after synthesis. This is because without RTL scan, the synthesis tool can optimize the circuit by removing the redundant FFs. Despite the presence of these redundant FFs, the area overhead of RTL scan increases on average by only 0.05% when compared to the gate level scan, which is insignificant. By constructing SCs at the RTL, the synthesis tool can improve timing by optimizing the test logic and functional logic concurrently. For example, although for B14 the timing is not improved, RTL scan insertion for the DSP

Period	# of	Original (No DFT)	Area OH				Δ
			Gate Full		RTL Full		
(ns)	SC	Timing Met?	%	Met?	%	Met?	%
10.80	32	Yes	4.84	Yes	5.52	Yes	-0.68
10.75	32	Yes	4.89	Yes	5.86	Yes	-0.97
10.70	32	Yes	6.01	Yes	5.78	Yes	0.23
10.65	32	No	4.80	No	4.94	Yes	-0.15
10.60	32	Yes	5.94	No	5.52	Yes	0.41
10.55	32	No	5.39	No	5.76	No	-0.37
10.50	32	No	6.41	No	5.99	Yes	0.42
10.45	32	No	5.90	No	5.52	No	0.37
10.40	32	No	5.61	No	5.56	Yes	0.05
10.35	32	No	5.94	No	5.73	Yes	0.20

Table 4.2: Area/performance for the DSP core with constraints for RISA

Period	# of	Original (No DFT)	Area OH				Δ
			Gate Full		RTL Full		
(ns)	SC	Timing Met?	%	Met?	%	Met?	%
5.90	16	Yes	6.73	Yes	7.96	Yes	-1.23
5.85	16	Yes	9.32	Yes	11.73	Yes	-2.41
5.80	16	Yes	1.32	Yes	4.93	Yes	-3.61
5.75	16	Yes	13.85	Yes	8.70	Yes	5.15
5.70	16	Yes	13.29	Yes	10.52	No	2.77
5.65	16	No	4.97	No	9.14	No	-4.17
5.60	16	Yes	5.19	Yes	6.08	No	-0.90
5.55	16	Yes	10.29	Yes	8.65	Yes	1.64

Table 4.3: Area/performance for B14 with constraints for RISA

core helps improve the timing to 10.35 ns, an improvement of 2.42% when compared to the original circuit, which fails to meet timing constraints beyond 10.6 ns.

Table 4.4 shows the test data compression results generated by a commercial ATPG tool [42] for B14. The timing constraint is listed in column 1. Column 2 shows the number of SCs that are driven by a single scan input in the broadcast mode. The columns labeled FC give the fault coverage for stuck-at faults. CTP corresponds to the number of compressed test patterns generated by ATPG and VTD denotes the volume of test data for the particular scan configuration. The last two rows of the table show the cumulative VTD and the compression ratio for the gate level and the RTL scan design. By consciously constructing the scan structure for the RISA to reduce the number of conflicts between FFs at the same scan depth located in different SCs, while placing conflicting SCs adjacent to each other, most of the faults become detectable when a single SI drives multiple SCs. This leads to savings in both test data volume and test application time. For example, when the common SI drives

Period	SC	Gate Full			RTL Full			Δ		
(ns)	#	FC (%)	CTP	VTD	FC (%)	CTP	VTD	FC (%)	CTP	VTD
5.90	16	95.03	487	7792	98.02	521	8336	2.99	-34	-544
	8	97.52	44	1408	98.89	40	1280	1.37	4	128
	4	98.81	31	1984	99.51	44	2816	0.70	-13	-832
	2	99.26	33	4224	99.54	1	128	0.28	32	4096
	1	99.27	2	512	99.54	1	256	0.27	1	256
Cumulative				15920			12816			3104
Compression				8.39			10.43			1.24

Table 4.4: Detail testability results for B14 with RISA

Period	Gate Full			RTL Full			Δ		
(ns)	FC (%)	VTD	CR_Gate	FC (%)	VTD	CR_RTL	FC (%)	VTD	CR_Gate/CR_RTL
10.80	98.99	27468	5.04	99.66	25347	5.46	0.67	2121	1.08
10.75	99.16	36771	3.78	99.63	28161	4.94	0.48	8610	1.31
10.70	99.07	34209	4.26	99.50	31584	4.62	0.43	2625	1.08
10.65	99.02	31857	4.75	99.58	31185	4.85	0.56	672	1.02
10.60	98.82	37275	4.25	99.56	28560	5.55	0.74	8715	1.31
10.50	98.98	34251	4.47	99.56	29484	5.20	0.58	4767	1.16
10.40	98.98	36372	3.84	99.57	28392	4.92	0.59	7980	1.28
10.35	99.09	30534	4.93	99.62	29757	5.06	0.52	777	1.03
Average	99.01	33592	4.41	99.59	29059	5.08	0.57	4533	1.16

Table 4.5: Testability results for the DSP core with RISA

4 SCs, a fault coverage of 99.51% can already be obtained for RTL scan compared to only 98.81% for gate level scan.

Tables 4.5 and 4.6 show the simplified testability results for the DSP core and B14, only for the cases where the timing of RTL scan is satisfied. Column 1 gives the timing constraints used for logic synthesis. Columns labeled FC represent fault coverage for stuck-at faults. VTD denotes the cumulative volume of test data. CR corresponds to the compression ratio, which is calculated by Equation 4.2.

$$CR = \frac{VTD\ of\ circuit\ when\ FFs\ are\ connected\ in\ a\ single\ chain}{VTD\ of\ circuit\ with\ Illinois\ scan\ architecture} \tag{4.2}$$

Period	Gate Full			RTL Full			Δ		
(ns)	FC (%)	VTD	CR_Gate	FC (%)	VTD	CR_RTL	FC (%)	VTD	CR_Gate/CR_RTL
5.90	99.27	15920	8.39	99.54	12816	10.43	0.27	3104	1.24
5.85	99.33	17696	7.99	99.21	14944	9.46	-0.11	2752	1.18
5.80	99.21	16672	8.40	99.41	14976	9.35	0.19	1696	1.11
5.75	99.39	17872	7.79	99.47	14720	9.46	0.09	3152	1.21
5.55	99.33	17088	8.03	99.45	15056	9.11	0.12	2032	1.13
Average	99.31	17050	8.12	99.42	14502	9.56	0.11	2547	1.17

Table 4.6: Testability results for B14 with RISA

Period	Gate Full			RTL Full			Δ		
(ns)	FC (%)	VTD	CR_Gate	FC (%)	VTD	CR_RTL	FC (%)	VTD	CR_Gate/CR_RTL
10.80	99.03	14406	9.61	99.66	11130	12.44	0.63	3276	1.29
10.75	99.20	20475	6.99	99.58	18249	7.84	0.39	2226	1.12
10.70	99.08	21084	6.66	99.55	18942	7.41	0.47	2142	1.11
10.65	98.99	18753	7.53	99.61	18186	7.76	0.62	567	1.03
10.60	98.89	20370	7.55	99.61	18522	8.31	0.72	1848	1.10
10.55	99.12	20874	7.66	99.70	19362	8.26	0.58	1512	1.08
10.50	99.03	18921	7.78	99.62	19320	7.62	0.59	-399	0.98
10.45	99.05	21525	6.68	99.52	18144	7.93	0.47	3381	1.19
Average	99.05	19551	7.56	99.61	17731	8.45	0.56	1819	1.12

Table 4.7: Testability results for the DSP core with RISA with inversion

Period	Gate Full			RTL Full			Δ		
(ns)	FC (%)	VTD	CR_Gate	FC (%)	VTD	CR_RTL	FC (%)	VTD	CR_Gate/CR_RTL
5.90	99.34	6672	20.60	99.31	3632	37.85	-0.02	3040	1.84
5.85	99.38	7456	19.36	99.34	5328	27.10	-0.04	2128	1.40
5.80	99.34	7808	17.38	99.30	5008	27.09	-0.05	2800	1.56
5.75	99.29	7600	18.43	99.41	4992	28.05	0.12	2608	1.52
5.70	99.31	7664	18.44	99.47	5424	26.05	0.16	2240	1.41
5.65	99.35	7632	17.34	99.50	4848	27.30	0.15	2784	1.57
5.60	99.49	7872	16.55	99.49	5008	26.02	0.00	2864	1.57
5.55	99.20	8208	16.81	99.45	5328	25.90	0.25	2880	1.54
5.50	99.42	7520	18.35	99.49	5616	24.57	0.07	1904	1.34
Average	99.35	7603	18.14	99.42	5020	27.77	0.07	2583	1.53

Table 4.8: Testability results for B14 with RISA with inversion

Despite the different logic networks that were obtained after logic synthesis, as well as the different test sets generated, the experimental results show that the RISA reduces the VTD for both gate level scan and RTL scan. However, by utilizing the control and data flow information extracted from the RTL description to build the scan structure, the reduction in VTD is further enhanced when compared with gate level scan insertion. For B14, the average compression ratio for the RTL scan is 9.56X, an improvement of 1.17X when compared to the gate level scan, which only has an average compression ratio of 8.12X.

4.4.2 Reconfigurable Illinois Scan With Inversion

The same set of constraints are used for SC generation, using the algorithms discussed in Section 4.3, for both RISAs. Hence they have the same impact on circuit speed, while difference in area is insignificant because it is given only by the XOR gates used for inversion. In the following we discuss the results for test data compression.

Tables 4.7 and 4.8 show the testability results for the DSP core and B14. The labeling of columns of the two tables follow the same patterns as Tables 4.5 and 4.6 that were introduced in Section 4.4.1. As we can see from the results, despite the different logic networks that were obtained after logic synthesis, as well as the different test sets generated for RTL scan of the DSP core and B14, the experimental results show that the RISA with inversion further reduces the VTD for both gate level scan and RTL scan. For example, the average compression ratio for the B14 for RTL scan is 27.77X, compared with a compression ratio of 18.14X for the gate level scan. This once again concludes that by utilizing the control and data flow information extracted from the RTL description to build the scan structure, the generated scan structure can be tuned to eliminate correlations between patterns in multiple SCs sooner using the broadcast modes in the modified RISA. It should be noted that the compression ratio for the B14 is better than the maximum achievable reduction of 16X for the original ISA, where the same set of test patterns are shifted into 16 SCs. This is because with the presence of the XOR gates in the modified RISA, the ATPG tool is *constrained, by the inverting feature of the SCs, on how the don't care bits are filled in a test pattern*. Due to the heuristic nature of filling the don't care bits in the test cubes generated for the single scan chain and how faults are ordered in the netlist, when embedding the RISA with inversion constraints, the ATPG tool may be able to detect *more faults through fault simulation*, which ultimately leads to less test patterns. Thus, the number of test patterns for designs with RISA with inversion may be smaller than the number of test patterns for the single scan chain design.

When comparing RISA with inversion against RISA, regardless of the scan type (i.e., gate level or RTL scan), the compression ratio is significantly improved. For example, the average compression ratios for the DSP core and B14 when using RTL scan are 5.08X and 9.56X when using the original RISA, as shown in Tables 4.5 and 4.6. When compared to the compression ratios of 8.45X and 27.7X for the same circuits using RTL scan for RISA with inversion, we can conclude that the inversion capability (provided by the carefully placed XOR gates) can effectively eliminate the ATPG conflicts caused by sharing the scan inputs and thus detect faults *early in the broadcast modes*, which leads to the improvement in compression ratio.

4.5 Summary

In this chapter, we have investigated the effectiveness of creating scan chains at RTL for test data compression using a new reconfigurable Illinois scan architectures. When compared to the gate level scan, it was found that the scan infrastructure built using only the control and data flow information available at the RTL can lead to similar or even better improvements in test data compression (for a fault coverage target over 99%), regardless of the final implementation of the logic network or the manufacturing test set. It addition, it was demonstrated with experimental data that by consciously embedding an inversion feature in the reconfigurable Illinois scan architecture, test data compression can be substantially increased.

Chapter 5

Conclusion

Motivated, by the escalating design cycle times, caused in part by the increasing gap between the level of specification for design and test hardware, in this thesis we have investigated new ways to insert scan structures. By analyzing the RTL description for functional information, the existing functional paths that can be transformed into scan paths are identified and then used in the RTL scan synthesis process. The original designs can then be interfaced to the RTL/logic synthesis tools (or RTL-to-GDSII flow [37]) with the scan infrastructure already in place. During this process, synthesis tools can optimize the functional and test logic concurrently which may benefit the timing and area of the resulted logic networks. Furthermore, it was found that another advantage from building functional scan chains at the RTL is the ability to tune the generated scan structures to improve the delay fault coverage or volume of test data. This is achieved by accounting for special constraints, during RTL scan synthesis, that influence the place and order of each flip-flop in every scan chain.

There are several important topics for future work. It is common to have complex RTL designs that are specified hierarchically using various components described in different HDLs. Thus, it will be beneficial for the proposed scan synthesis process to be able to deal with hierarchical and mixed-language specifications. In addition, because the research findings presented in this thesis are based on designs that have a signal clock signal, it is important to extend the proposed algorithms to deal with multiple clock domains.

Appendix A

RTL Code Modification For Functional Scan

A program has been developed for the purpose of the research presented in this thesis. The flow of the program has already been discussed in Chapter 3. In order to demonstrate how the RTL description is modified for scan chain insertion, a simple divider written in Verilog HDL is shown as an example in *Sample Code 1.* There are 11 FFs $\{quotient[3:0], remainder[3:0], done, present\_state[1:0]\}$ driven by the clock signal $\{clk\}$ in this example. The *S Graph* that illustrates the control and data flow between the FFs is shown in Figure A.1. In the *S Graph*, the solid edges represent the data flow, while the broken edges denote the control flow between FFs.

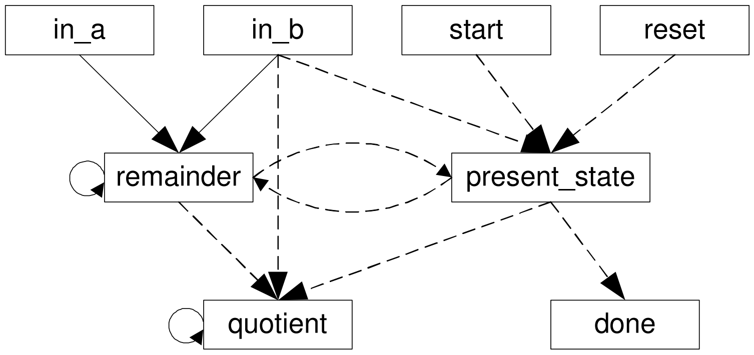


Figure A.1: S graph for the divider example in sample code 1

Sample Code 1: Input RTL description for a divider

```
module rtl_input (clk, reset, start, in_a, in_b, quotient, remainder, done);
    input clk, reset, start;
    input [3:0] in_a, in_b;

    output [3:0] quotient, remainder;
    reg [3:0] quotient, remainder;
    output done;
    reg done;

    parameter S_IDLE = 'd0;
    parameter S_INIT = 'd1;
    parameter S_FETCHINPUT = 'd2;
    parameter S_BUSY = 'd3;

    reg [1:0] present_state, next_state;

    // Update present state
    always @ (posedge clk) begin
        if (reset == 1'b1) begin
            present_state <= S_IDLE;
        end
        else begin
            present_state <= next_state;
        end
    end

    // Next state
    always @ (present_state or start or remainder or in_b) begin
        next_state = present_state;
        case (present_state)
            S_IDLE: begin
                if (start == 1'b1) begin
                    next_state = S_INIT;
                end
            end
            S_INIT: next_state = S_FETCHINPUT;
            S_FETCHINPUT: next_state = S_BUSY;
            default: begin
                if (remainder < in_b) begin
                    next_state = S_IDLE;
                end
            end

        endcase
    end
```

Sample Code 1: Input RTL description for a divider (continued)

```
    // Done signal
    always @ (posedge clk) begin
        if (present_state == S_IDLE) begin
            done <= 1'b1;
        end
        else begin
            done <= 1'b0;
        end
    end

    // Remainder calculation
    always @ (posedge clk) begin
        if (present_state == S_FETCHINPUT) begin
            remainder <= in_a;
        end
        else begin
            if (remainder >= in_b) begin
                remainder <= remainder - in_b;
            end
        end
    end

    // Quotient calculation
    always @ (posedge clk) begin
        if (present_state == S_INIT) begin
            quotient <= 0;
        end
        else begin
            if (present_state == S_BUSY) begin
                if (remainder > in_b) begin
                    quotient <= quotient + 1;
                end
            end
        end
    end
endmodule
```

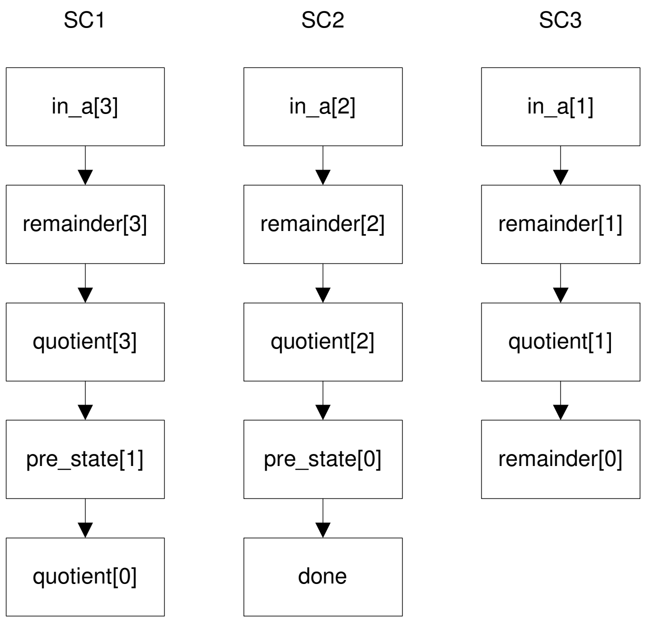


Figure A.2: Scan cell partitioning for sample code 2

Sample Code 2 presents the modified RTL code after the example from *Sample Code 1* has gone through the RTL scan synthesis process. In this case, the skewed-load delay fault testing constraints from Section 3.5.1 are applied to construct three SCs. The orders of FFs in the three SCs are shown in Figure A.2, and the corresponding RTL modifications are marked with vertical lines in *Sample Code 2*. In this example, the original PIs $\{in\_a[3], in\_a[2], in\_a[1]\}$ and POs $\{quotient[0], done, remainder[0]\}$ are reused as SIs and SOs respectively in the test mode. As a result, additional scan pins and routing of scan signals are not necessary. Moreover, since the delay fault testing constraints are applied in this case, the FFs in the SCs are organized such that all the constraints can be met to enhance delay fault coverage without incurring extra area overhead.

In addition to the delay constraints discussed in Section 3.5.1, the custom constraints mentioned in Section 4.3 for tuning the generated scan structure to improve test data compression when using the two proposed RISAs can also be applied. These constraints can be chosen by different command-line parameters for the program.

Sample Code 2: Output RTL description for a divider with functional scan

```
module rtl_output (clk, reset, start, in_a, in_b, quotient, remainder, done,
test_se);
    input clk, reset, start, test_se;
    input [3:0] in_a, in_b;
    output [3:0] quotient, remainder;
    reg [3:0] quotient, remainder;
    output done;
    reg done;
    parameter S_IDLE = 'd0;
    parameter S_INIT = 'd1;
    parameter S_FETCHINPUT = 'd2;
    parameter S_BUSY = 'd3;
    reg [1:0] present_state, next_state;
    // Update present state
    always @ (posedge clk) begin
        if (test_se == 1'b0) begin
            if (reset == 1'b1) begin
                present_state <= S_IDLE;
            end
            else begin
                present_state <= next_state;
            end
        end
        else begin
            present_state[1] <= quotient[3];
            present_state[0] <= quotient[2];
        end
    end
    // Next state
    always @ (present_state or start or remainder or in_b) begin
        next_state = present_state;
        case (present_state)
            S_IDLE: begin
                if (start == 1'b1) begin
                    next_state = S_INIT;
                end
            end
```

Sample Code 2: Output RTL description for a divider (continued)

```
        S_INIT: next_state = S_FETCHINPUT;
        S_FETCHINPUT: next_state = S_BUSY;
        default: begin
            if (remainder < in_b) begin
                next_state = S_IDLE;
            end
        end
    endcase
end
// Done signal
always @ (posedge clk) begin
    if (test_se == 1'b0) begin
        if (present_state == S_IDLE) begin
            done <= 1'b1;
        end
        else begin
            done <= 1'b0;
        end
    end
    else begin
        done <= present_state[0];
    end
end
// Remainder calculation
always @ (posedge clk) begin
    if (test_se == 1'b0) begin
        if (present_state == S_FETCHINPUT) begin
            remainder <= in_a;
        end
        else begin
            if (remainder >= in_b) begin
                remainder <= remainder - in_b;
            end
        end
    end
```

Sample Code 2: Output RTL description for a divider (continued)

```
        else begin
            remainder[3] <= in_a[3];
            remainder[2] <= in_a[2];
            remainder[1] <= in_a[1];
            remainder[0] <= quotient[1];
        end
    end
    // Quotient calculation
    always @ (posedge clk) begin
        if (test_se == 1'b0) begin
            if (present_state == S_INIT) begin
                quotient <= 0;
            end
            else begin
                if (present_state == S_BUSY) begin
                    if (remainder > in_b) begin
                        quotient <= quotient + 1;
                    end
                end
            end
        end
        else begin
            quotient[3] <= remainder[3];
            quotient[2] <= remainder[2];
            quotient[1] <= remainder[1];
            quotient[0] <= present_state[1];
        end
    end
endmodule
```

Appendix B

Implementation effort of RTL functional scan synthesis process

The program that was developed to facilitate the RTL functional scan synthesis process detailed in this thesis can be divided into five steps as discussed in Chapter 3. The first step of the RTL functional scan synthesis process is to parse RTL circuit descriptions written in Verilog HDL for *CDFG generation*. This part of the program is written in three different languages. They are *Lex*, *YACC* and *ANSI C*. The remaining steps, which include *S Graph generation*, *SP identification*, *SC construction* and *modification of RTL code*, are written only in *ANSI C*. Table B.1 shows the breakdown of different steps of the program in terms of number of lines of code. Moreover, in order to reduce the development efforts, the program was developed using the publicly available data structure from *libHLS* [43].

Step Number	Description	Number of Lines of Code
1	CDFG generation	6309
2	S Graph generation	5167
3	SP identification	5016
4	SC construction	2895
5	Circuit modification	3995
	Total:	23382

Table B.1: Breakdown of code size for the RTL functional scan synthesis process

After the program modifies the RTL description of the design to incorporate the new functional scan structure, a set of scripts will then be generated so that the scan design can be interfaced with two commercial test tools from Synopsys. In order to synthesize the scan design for area and performance analysis, the Design Compiler was used [41]. After a circuit has been synthesized, the TetraMax ATPG tool was used to generate test patterns and other test related experimental results such as the delay fault coverage with the gate level structural netlist [42].

Bibliography

[1] *SCU-RTL Benchmarks.* Santa Clara University, Santa Clara, CA, 1998.

[2] M. Altaf-Ul-Amin, S. Ohtake, and H. Fujiwara. Design for hierarchical two-pattern testability of data paths. In *Proc. of 10th Asian Test Symposium*, pages 11–16, 2001.

[3] T. Asaka, S. Bhattacharya, S. Dey, and M. Yoshida. H-SCAN+: A practical low-overhead RTL design-for-testability technique for industrial designs. In *Proc. International Test Conference*, pages 265–274, 1997.

[4] H. Bhatnagar. *Advanced ASIC Chip Synthesis.* Kluwer Academic Publishers, 2nd edition, 2002.

[5] S. Bhattacharya and S. Dey. H-SCAN: A high level alternative to full-scan testing with reduced area and test application overheads. In *Proc. 14th IEEE VLSI Test Symposium*, pages 74–80, April 1996.

[6] S. Boubezari, E. Cerny, B. Kaminska, and B. Nadeau-Dostie. Testability analysis and test-point insertion in RTL VHDL specifications for scan-based BIST. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(9):1327–1340, September 1999.

[7] M. L. Bushnell and V. D. Agrawal. *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits.* Kluwer Academic Publishers, Boston, 2000.

[8] K. Chakrabarty, V. Iyengar, and A. Chandra. *Test Resource Partitioning for System-on-a-Chip.* Kluwer Academic Publishers, 2002.

[9] K. T. Cheng, S. Devadas, and K. Keutzer. Delay-fault test generation and synthesis for testability under a standard scan design methodology. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(8):1217–1231, August 1993.

[10] V. Chickermane and K. Zarrineh. Addressing early design-for-test synthesis in a production environment. In *Proc. International Test Conference*, pages 246–255, 1997.

[11] S. Chiusano, F. Corno, and P. Prinetto. RT-level TPG Exploiting High-Level Synthesis Information. In *Proc. 17th IEEE VLSI Test Symposium*, pages 341 –346, Apr. 1999.

[12] S. Deniziak and K. Sapiecha. Developing a high-level fault simulation standard. *Computer*, 34:89–90, May 2001.

[13] B. Dervisoglu and G. Stong. Design for Testability: Using Scanpath Techniques for Path-Delay Test and Measurement. In *Proc. International Test Conference*, pages 365–374, 1991.

[14] S. Dey, A. Raghunathan, and R. K. Roy. Considering testability during high-level design. In *Design Automation Conference 1998. Proceedings of the ASP-DAC*, pages 205–210, Asia and South Pacific, Feb. 1998.

[15] D. Gajski. *Silicon Compilation.* Addison-Wesley Longman Incorporated, MA, 1988.

[16] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., San Fransico, 1979.

[17] I. Hamzaoglu and J. H. Patel. Reducing test application time for full scan embedded cores. In *Proc. IEEE VLSI Test Symposium*, pages 369–375, April 2000.

[18] F. F. Hsu, K. M. Butler, and J. H. Patel. A case study on the implementation of the illinois scan architecture. In *Proc. International Test Conference*, pages 538–547, 2001.

[19] Y. Huang, C. C. Tsai, N. Mukherjee, O. Samman, D. D. W. T. Cheng, and S. M. Reddy. On RTL scan design. In *Proc. International Test Conference*, pages 728–737, 2001.

[20] J. P. Hurst and N. Kanopoulos. Flip-flop sharing in standard scan path to enhance delay fault testing of sequential circuits. In *Proc. of 4th Asian Test Symposium*, pages 346–352, 1995.

[21] K. Kim, S. Mitra, and P. Ryan. Delay defect characteristics and testing strategies. *IEEE Design and Test of Computers*, 20(5):8–16, Sept 2003.

[22] J. Leenstra, M. Koch, and T. Schwederski. On scan path design for stuck-open and delay fault detection. In *Proc. of European Test Conference*, pages 201–210, 1993.

[23] C. C. Lin and K. T. Cheng. Test-point insertion: Scan paths through functional logic. *IEEE Trans. Comput.-Aided Design Integrated Circuits*, 17(9):838–851, Sept. 1998.

[24] C. C. Lin, M. Lee, M. M. Sadowska, and K. C. Chen. Cost-free scan: A low-overhead scan path design methodology. In *Proc. International Conference on Computer Aided Design*, pages 528–533, 1995.

[25] G. D. Micheli. *Synthesis and Optimization of Digital Circuits.* McGraw-Hill Inc., 1994.

[26] S. Mourad and Y. Zorian. *Principles of Testing Electronic Systems.* Wiley-Interscience Publication, 2000.

[27] R. B. Norwood and E. J. McCluskey. Synthesis-for-scan and scan chain ordering. In *Proc. 14th IEEE VLSI Test Symposium*, pages 87–92, 1996.

[28] R. B. Norwood and E. J. McCluskey. Delay testing of data paths with scan. In *Proc. International Test Synthesis Workshop*, 1997.

[29] R. B. Norwood and E. J. McCluskey. High level synthesis for orthogonal scan. In *Proc. 15th IEEE VLSI Test Symposium*, pages 370–375, 1997.

[30] A. R. Pandey and J. H. Patel. An incremental algorithm for test generation in illinois scan architecture based designs. In *Proc. Design Automation and Test in Europe*, pages 368–375, March 2002.

[31] A. R. Pandey and J. H. Patel. Reconfiguration technique for reducing test time and test data volume in illinois scan architecture based designs. In *Proc. 20th IEEE VLSI Test Symposium*, pages 9–15, April 2002.

[32] I. Pomeranz and S. M. Reddy. On the coverage of delay faults in scan designs with multiple scan chains. In *Proc. IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 206–209, 2002.

[33] J. Rajski and J. Tyszer. *Arithmetic Built-In Self-Test For Embedded Systems.* Prentice-Hall, Inc, 1998.

[34] S. Roy. RTL Based Scan BIST. In *Proc. VHDL International Users' Forum*, pages 117–121, October 1997.

[35] S. Roy, G. Guner, and K. T. Cheng. Efficient test mode selection and insertion for RTL-BIST. In *Proc. International Test Conference*, pages 263–272, October 2000.

[36] S. Samaranayake, E. Gizdarski, N. Sitchinava, F. Neuveux, R. Kapur, and T. W. Williams. A reconfigurable shared scan-in architecture. In *Proc. 21th IEEE VLSI Test Symposium*, pages 9–14, April 2003.

[37] M. Santarini. RTL-to-GDSII flow shows signs of maturity. In *EE Design*, June 3 2002.

[38] J. Savir and S. Patil. Scan-based transition test. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(8):1232–1241, August 1993.

[39] N. Sitchinava, S. Samaranayake, R. Kapur, E. Gizdarski, F. Neuveux, and T. W. Williams. Changing the scan enable during shift. In *Proc. 22th IEEE VLSI Test Symposium*, 2004.

[40] M. Sonza-Reorda, F. Corno, and G. Squillero. ITC'99 Test Benchmarks Web Site. http://www.cad.polito.it/tools/itc99.html, 1999.

[41] Synopsys Synthesis Tools. Design Compiler. http://www.synopsys.com/products/logic/design_compiler.html, 2003.

[42] Synopsys Test Tools. TetraMAX ATPG. http://www.synopsys.com/products/test/tetramax_dsA4.pdf, 2003.

[43] S. Tarafdar. libHLS v2.1.0. http://www.ece.neu.edu/groups/rpl/libHLS/, 2002.

[44] N. A. Touba and E. J. McCluskey. Applying two-pattern tests using scan-mapping. In *Proc. of 14th VLSI Test Symposium*, pages 393–397, 1996.

[45] B. Vinnakota and N. K. Jha. Synthesis of sequential circuits for parallel scan. In *Proc. European Design Automation Conference*, pages 366–370, 1992.

[46] J. F. Wakerly. *Digital Design Principles and Practiecs.* Prentice Hall, third edition, 2001.

[47] D. Wu, M. Lin, S. Mitra, K. Kim, A. Sabbavarapu, T. Jaber, P. Johnson, D. March, and G. Parrish. H-DFT: A hybrid DFT architecture for low-cost high quality sturctural testing. In *Proc. International Test Conference*, pages 1229–1238, October 2003.

Index