AT-SPEED SCAN INSERTION AND AUTOMATIC TEST PATTERN GENERATION OF INTEGRATED CIRCUITS WITH FAULT-GRADING AND SPEED-GRADING

by

Joseph Fang

B.A.Sc. University of British Columbia, 2000

PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF ENGINEERING

In the School
of
Engineering Science

SIMON FRASER UNIVERSITY

Summer 2005

APPROVAL

Name: **Joseph Weizhou Fang**

Degree: **Master of Engineering**

Title of Project: **At-Speed Scan Insertion And Automatic Test Pattern Generation Of Integrated Circuits With Fault-Grading And Speed-Grading**

Examining Committee:

Dr. Mirza Faisal Beg
Committee Chair
Assistant Professor, Engineering Science

Dr. Karim S. Karim
Senior Supervisor
Assistant Professor, Engineering Science

Dr. Karim Arabi
Supervisor
Manager, Design Automation, PMC-Sierra Inc.

Date Defended/Approved: February 4, 2005

SIMON FRASER UNIVERSITY

PARTIAL COPYRIGHT LICENCE

ABSTRACT

With the growing complexity of today's integrated circuit designs, engineers have abandoned the use of pure functional test vectors wherever possible, and adopted various DFT solutions to make their designs more test-friendly. The most common DFT approach for digital designs is scan insertion and automatic test pattern generation (ATPG). ATPG is performed based on fault models associated with the design or gates within the design. Traditionally, the most popular model is the stuck-at model. However, as transistor size continues to shrink, new defect mechanisms start to appear that affect the speed of the design, and so can no longer be properly modelled by this model. Consequently, a new fault model called transition-delay fault models is created to allow ATPG to detect at-speed defects. Another model called path-delay fault model is also created for speed-grading/binning and I/O timing characterization on scan-inserted designs.

As part of an ongoing DFT development for PMC-Sierra Inc., a suite of automation flows have been implemented to perform AC-Scan ATPG. This includes transition-delay ATPG with DC top-up ATPG for delay defect detection, path-delay ATPG for speed-grading/binning and I/O timing characterization, and AC-scan ATPG for RAM interfaces with multi-load algorithm.

DEDICATION

To my parents, my wife and my unborn child.

ACKNOWLEDGEMENTS

I would like to thank Dr. Karim S. Karim for his supervision during the progress of this project. His willingness to take on the supervisory duties and meaningful advices during the course of this project is most appreciated. I would also like to thank Dr. Karim Arabi for his constant supervision and mentoring during the progression of this project and the past four years. I would not have been able to become a qualified and successful DFT engineer without his guidance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ATE	Automated Test Equipment
ATPG	Automatic Test Pattern Generation
BIST	Built-in Self Test
DFT	Design for Testability
DLL	Delay Lock Loop
DUT	Device under Test
EDA	Electronic Design Automation
I/O	Input/Output
IC	Integrated Circuit
PI	Primary Input
PLL	Phase lock Loop
PO	Primary Output
RAM	Random Access Memory
SE	Scan-Enable
SoC	System-on-a-Chip
STA	Static Timing Analysis

CHAPTER 1 INTRODUCTION AND BACKGROUND

1.1 General Overview of Integrated Circuit Testing

A majority of today's digital electronic components consists of collections of Silicon-based Integrated Circuits (ICs). Generally speaking, each piece of IC is designed to perform certain digital or logical functions at a particular speed. However, manufacturing process variations and/or contaminations during IC fabrication may cause the actual silicon to deviate from the targeted performance or malfunction altogether. Hence, it is important that every fabricated piece of IC be thoroughly tested to guarantee its quality. This is the driving force behind a variety of IC test approaches.

Similar to testing of any system, the basic idea of digital IC testing is to apply stimuli to the device under test (DUT), and compare the actual output of the DUT with the expected response. The collection of input stimuli and associated expected output responses are called test vectors. More specifically, in a set of test vectors, different combinations and sequences of logic zeroes and ones are applied onto the primary input (PI) ports of the DUT. These sequences are meant to sensitize portions of the DUT's internal circuitry, causing it to change the logic states of related primary output (PO) ports. The vectors contain the expected response of a working DUT based on the input stimuli, and compares the actual output logic values with the expected values. A mismatch between the expected and actual circuit behaviour would indicate that a faulty DUT has been detected and that it is to be discarded.

To measure the effectiveness of a set of test vectors, the notion of test coverage needs to be introduced. Test coverage is a measurement of the percentage of the circuit's total gates that is properly tested with the test vectors. Ideally, this needs to be 100% such that every gate in the design is tested to guarantee that the DUT is validated in its entirety. However, this is not always a realistic goal. In general, 95% is considered acceptable for ICs implemented in older technologies, and 97%-99% is needed for ICs in more current technologies.

A piece of IC is tested by being placed on an automated test equipment (ATE), also called production tester.[1] The I/O ports of the IC are connected to tester channels of the tester. Furthermore, test vectors are stored in the tester, which applies the stimuli to the IC's input ports and compares the output response from the IC's output ports through the tester channels. There is only a finite amount of storage space in an ATE, meaning that the desired test coverage must be achieved by a finite number of test vectors. This number, defined as vector count, must be less than the physical memory limitation of the tester to be fully loaded into it. In a way, vector count is a benchmark of the vector efficiency. Older testers can store up to eight million test vectors while newer ones have memory upper bounds at 32 million vectors or higher. Aside from avoiding reaching the tester memory limit, it is always good practice to keep the vector count low due to the high cost of tester time.

Before the advent of any structured testing approaches, test vectors are often in the form of manually written functional testbenches. These testbenches are designed to sensitize as many gates in the design as possible through functional operations of the device. They are usually derived from simulation vectors written as part of design verification to confirm behavioural specification of the design. [1] This kind of test approach is often inefficient and ineffective due to the difficulties to cover all corner

cases of the design with limited availability of vector space and manpower, especially for today's multi-million gate designs. A large part of the inadequacies of these manually written vectors, also called functional test vectors, is due to the existence of sequential elements such as flip-flops and latches which create depth in the design: In an edge-triggered design, in order to test a gate embedded in the core of a block of digital circuit, the input stimuli must traverse through multiple layers of flip-flops to sensitize the gate, and similarly the output of the gate must travel through multiple layers of flip-flops before arriving at a primary output to be observed. As designs increase in complexity and levels of sequential elements grow, vector count increases exponentially due to the limited number of I/Os available to traverse the depth of the core logic. It is clear that IC testing will become unmanageable without a more structured as well as more automated approach.

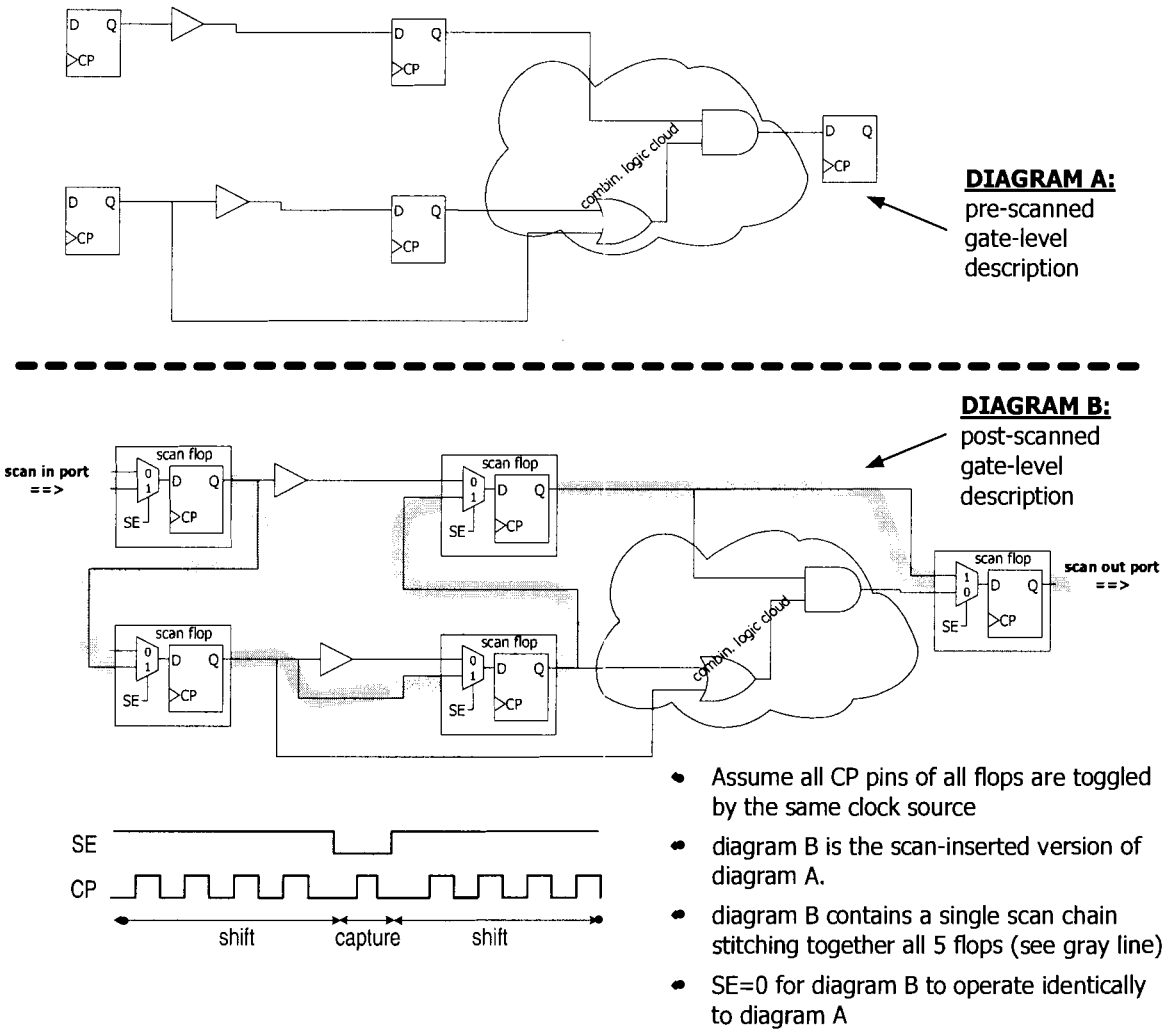
1.2 Automated Testing: Scan Insertion and Automatic Test Pattern Generation

As ICs become increasingly more difficult to test through conventional functional vectors, IC engineers start to design test structures into digital circuits to help ease the burden of testing. This concept of making circuits more testable is called Design-for-Testability (DFT). As mentioned, the major contributor to the complexity of the functional test vectors is the existence of flip-flops and the stages of pipelines they create. However, this issue can be resolved if the flip-flops can be made directly controllable and observable. This idea is realized through scan-insertion, a conceptually simple and highly automated DFT implementation method, and one of the most useful solutions offered by DFT to make digital IC testing manageable. In today's IC design industry, a large part of quality assurance of digital ICs is based on running scan test vectors through a scan-inserted (also called scan-stitched) design. [1] From the view of a

designer, this involves two main steps: scan insertion, and automatic test pattern generation (ATPG).

1.2.1 Scan Insertion

Figure 1: Scan insertion through replacement and stitching of flip-flops



The following is a description of scan insertion: After a design has been synthesized down to a gate-level description (also called a netlist), an electronic-design-automation (EDA) tool such as DFTAdvisor from Mentor Graphics or DFTCompiler from

Synopsys Inc. is used to link together all edge-triggered sequential elements to form long chains of shift registers called scan chains. [2] This process requires the introduction of a DFT-specific input pin called scan-enable (SE). [1] Usually, an SE-controlled Multiplexer is inserted onto the data input of every D-flop such that the functional input is routed into the flop when SE is assigned '0' and another scan-mode data input, normally called scan-in, is routed in when SE is assigned '1'. The scan-chain is constructed by connecting the output of one flop to the scan-in of the next flop, forming a long shift-register when SE equals '1'. The scan-in of the first flop in the chain is driven by a device-level scan-in port; and the output of the last flop in the chain drives a device-level scan-out port. SE is held at '0' while the device is operating in functional mode so that functional data paths are carrying data across. Above is a diagram depicting scan insertion.

1.2.2 Automatic Test Pattern Generation (ATPG)

With the scan chains constructed, the flops are transformed from deeply embedded internal nodes of the circuits into control and observe points for testing. This is accomplished by asserting SE to '1' and pumping data onto each flop in the scan chain(s), with continuous clock pulses, from the device-level scan-in port(s). The data shifted into the flops (also called scan cells) are the input stimuli that launch into the functional paths from the flops' outputs and arrive at the data-input of other scan cells, sensitizing combinational gates along the way. Then, SE is de-asserted to break the scan chains and connect the functional paths, and clock signals are pulsed to capture the data traversed through the functional paths onto the scan cells. Finally, the same shift operation is done to pump the captured data out through the scan-out port. The data shifted out make up the device's response to the input stimuli from the shift-in

operation and are compared for mismatches. Because of the structured nature of scan-testing, today's EDA tools, such as FastScan from Mentor Graphics or TetraMAX from Synopsys Inc., make use of this shift-and-capture sequence to automatically create test patterns/vectors to validate the circuits efficiently, usually achieving 95% test coverage or higher at reasonable vector/pattern count. Because of its consistent delivery of results, the approach of using EDA tools to automatically generate scan vectors is the mainstream method of creating test vectors today.

At this point, it is worthwhile to define some of the commonly used terms for ATPG: Each clock pulse while SE equals '1' forms a shift vector; Each clock pulse while SE equals '0' is called a capture vector or capture cycle; The entire sequence of continuous shift operations that fully load and/or unload the scan chains in the design is called a shift sequence or load/unload operation; A shift-sequence and the capture vector(s) immediately trailing it together form a test pattern or scan pattern.

In order to perform test pattern/vector generation, the ATPG tools use fault models to describe the behaviour of physical defects. Traditionally, the most commonly used model is called the stuck-at model, which translates all physical defects of digital circuits into particular nodes or pins of digital gates being unable to make transitions and so permanently "stuck" at a particular logic level. [3] Since the model assumes that faulty gates are permanently connected to VDD or VSS, scan-testing can be done at very low speed as the stuck-at values are always present without regards to clock frequency. This model has been one of the best tools for testing IC, but its usefulness is diminishing as IC technology makes its advances into 0.13um and 90nm.

1.3 Need for At-Speed Testing

As advancements are made to design, development and fabrication of ICs, chips are made to run faster and faster and are designed to contain more and more complex functionalities. [4] This drive for faster performance and System-On-a-Chip (SoC) design structure pushes the boundary of IC fabrication, reducing the transistor size. From a test perspective, the ever-shrinking transistor in deep-submicron technology has caused the emergence of new defect mechanisms such as resistive via. Faulty gates with this type of defect mechanism exhibit the behaviour of very slow data transitions. [5] In other words, the gate still correctly makes the transition, just not at a high enough speed. To detect this type of faults, a new DFT strategy is needed to test the device at its functional operating frequency. [6] One has been derived from the conventional scan/ATPG approach: at-speed scan/ATPG, also called AC-scan/ATPG. To distinguish itself from AC-Scan, the conventional scan/ATPG methodology with stuck-at fault model is also called DC-Scan/ATPG.

Clearly, this new type of defects cannot be modelled by "permanently stuck-at" logic levels, and so requires a set of new "temporarily stuck-at" fault models that accomplish scan-testing at or close to the device's functional frequency of operation. There are two fault models introduced: transition-delay model, and path-delay model. [7] The current industry trend is to use these models to augment stuck-at models. However, they soon will become the dominant fault models.

With Transition-delay fault model, a digital gate is modelled such that a transition (0-to-1 or 1-to-0) on its inputs must result in a corresponding transition on the output. [7] The ATPG tool running with this type of modelling creates test patterns to launch a transition (rather than a constant value as in the case of DC-scan) into the targeted input

of the gate, then captures the corresponding post-transitioned value of the gate's output onto a scan-cell, and finally shifts out the scan-cell value to verify the transition. Path-delay fault model is similar to transition-delay model. The difference is that, with path-delay model, the transition is defined on an entire path (i.e. from source flop to destination flop through all combinational gates in between). This fault model is used to characterize an entire path. It is ideal for speed-binning or speed-grading in which the critical paths are selected to be sensitized by the ATPG tool. The tool then tests if a transition on the source flop can result in a corresponding transition on the destination flop within the specified speed. A failure does not necessarily mean physical defect, but rather than the "faulty" chip may belong to a lower grade of ICs and sold as a less expensive part.

CHAPTER 2 PROJECT DESCRIPTION

2.1 Project and Report Overview

The project to be described by this report is on the implementation of an industry-standard AC-Scan and ATPG automation flow to enable at-speed test for PMC-Sierra Inc.. PMC-Sierra is a "fabless" semiconductor design house. Its core technology revolves around developing ICs that enables transmission, processing and storage of Internet data, as well as general-purpose microprocessors. The project enables at-speed testing of ICs using existing DFT structures contained within the ICs, including scan-flops and various clock gating architectures to provide clock controllability during testing. This project is further broken down into the following components each of which will be discussed in detail in a later chapter of this report:

2.1.1 Develop AC-Scan ATPG flow for physical defect detection

This component requires the implementation of a push-button automation flow to automatically create test patterns with the transition-delay fault model. It is described in Chapter 3. From the perspective of this report, issues related to design partitioning in the context of AC-scan is of more interest than the actual coding of scripts to run the ATPG tool. Therefore, the discussion will focus on the design intrusions, automation flow and pitfalls.

2.1.2 Develop AC-Scan ATPG flow to qualify operating speed of devices

This component is discussed in Chapter 4. There are two sub-components in the discussion both of which revolve around ATPG with path-delay fault models: One is the normal ATPG of internal (flop-to-flop) data paths for speed-grading/binning. The other is AC-Scan ATPG on paths connected from/to device-level primary I/Os to verify their timing specifications on silicon (also called I/O characterization). There are some design related issues here such as access of delay-locked loop (DLL) in AC-Scan mode that are also explored. Also, the choice of data-paths for speed-grading is discussed.

2.1.3 Develop AC-Scan ATPG flow for at-speed test of RAM interface

Neither of the two components above supports at-speeding testing of RAM interfaces due to the complexity of the ATPG algorithm when testing RAMs. Chapter 5 describes an ATPG flow based on AC-Scan that specifically targets data and address buses of RAMs. More specifically, the ATPG algorithm of the ATPG tool itself is described, and some design-related issues are also discussed.

2.2 Project significance

This project bares a high degree of significance to PMC-Sierra. In a cost-driven business environment, it is important to keep every working part to maintain good profit margin for the company, and discard every defective part to ensure customer confidence. This means there is to be no or at least next-to-none test-escapes and false-rejects. This need is further magnified by the lower yields due to immaturity of the newer technologies. Lower yield increases the likelihood of faulty parts and therefore the opportunity for test-escapes, hence placing a tougher constraint on test coverage which is directly proportional to vector count. Even though the financial benefits of

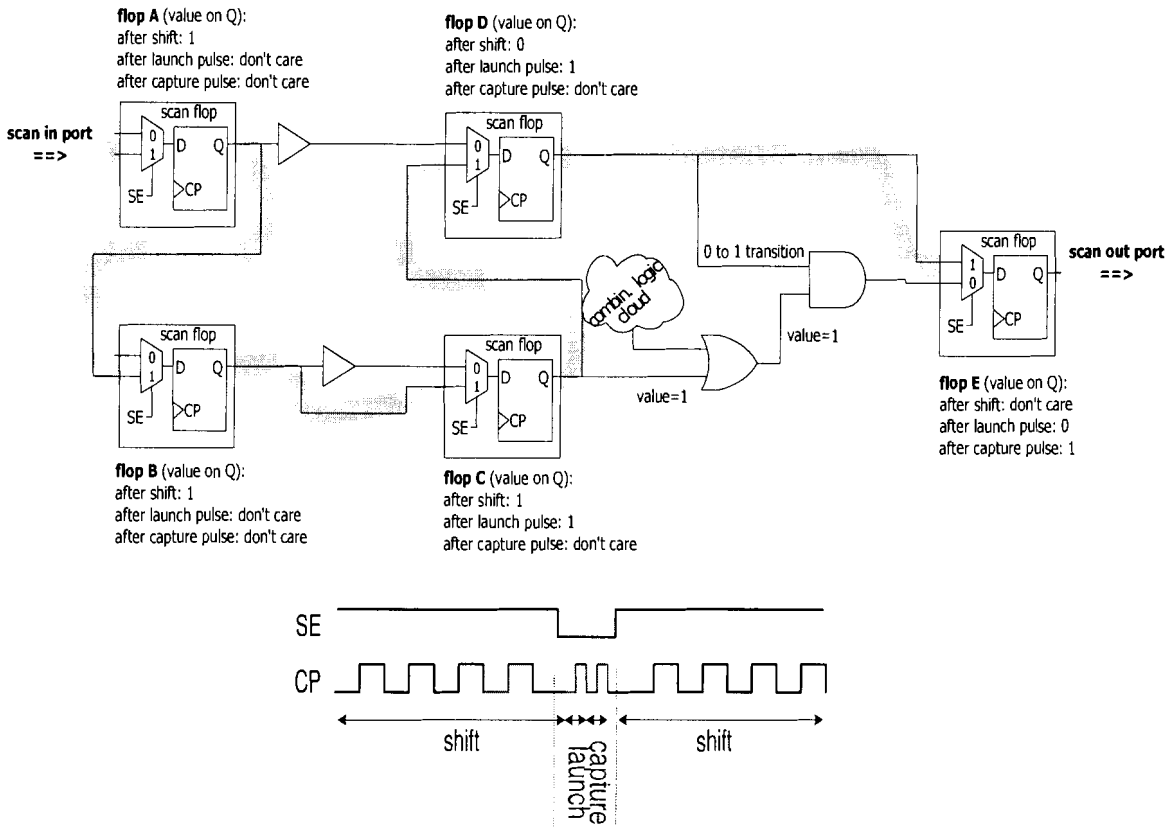
testable designs are not easy to quantify [8], the use of structured tests will shorten the design cycle and result in overall savings to the company. [9] The generation and use of test patterns is a major contributor to the overall cost of IC development. ATPG itself is a non-recurring cost that only applies during the design phase. [9] This includes cost of human engineering resource, license cost for the usage and maintenance of ATPG tools, computer resources. The more major cost here is the actual test time of each fabricated silicon as it is a recurring cost that applies to every piece of IC to be shipped to customer or discarded due to detected defects. [9] Therefore, it is vital that a straight-forward ATPG flow is put in place to ensure efficient vector count at acceptable test coverage.

CHAPTER 3 TRANSITION-DELAY ATPG

3.1 Transition-Delay ATPG Methodology

As mentioned, Transition-delay fault model is used to detect at-speed manufacturing defects on silicon. With this fault model, a digital gate is modelled such that a transition (0-to-1 or 1-to-0) on its inputs must result in a corresponding transition on the output. The ATPG tool is designed to understand the model, and creates two consecutive clock pulses at fast clock frequency to make up the capture cycles, a procedure called double-pulse. [10] Using shift-in operation, it first presets the input of the gates with the pre-transition value in preparation for the double-pulse, then launches the post-transition value into the gate from a scan cell with the first clock pulse to cause a transition on the gate's output, and tries to capture the final value on the gate's output with the second pulse onto a scan cell connected to the output of the gate. Finally, contents of the scan chains are shifted out for comparison. Please note that the model itself does not specify the propagation speed of the input transition to the output, but rather just specifies the values before and after the transition. As long as a transition on the gate of interest is properly launched and captured through the double-pulse of the clock, the fault is considered to be detected at-speed. This way, it is up to the designer to choose a clock frequency for the double-pulse and the tool to implement it. The following diagram provides an example of transition-delay ATPG and the method of detecting at-speed faults using double-pulse of the clock.

Figure 2: AC-Scan Transition-delay ATPG example



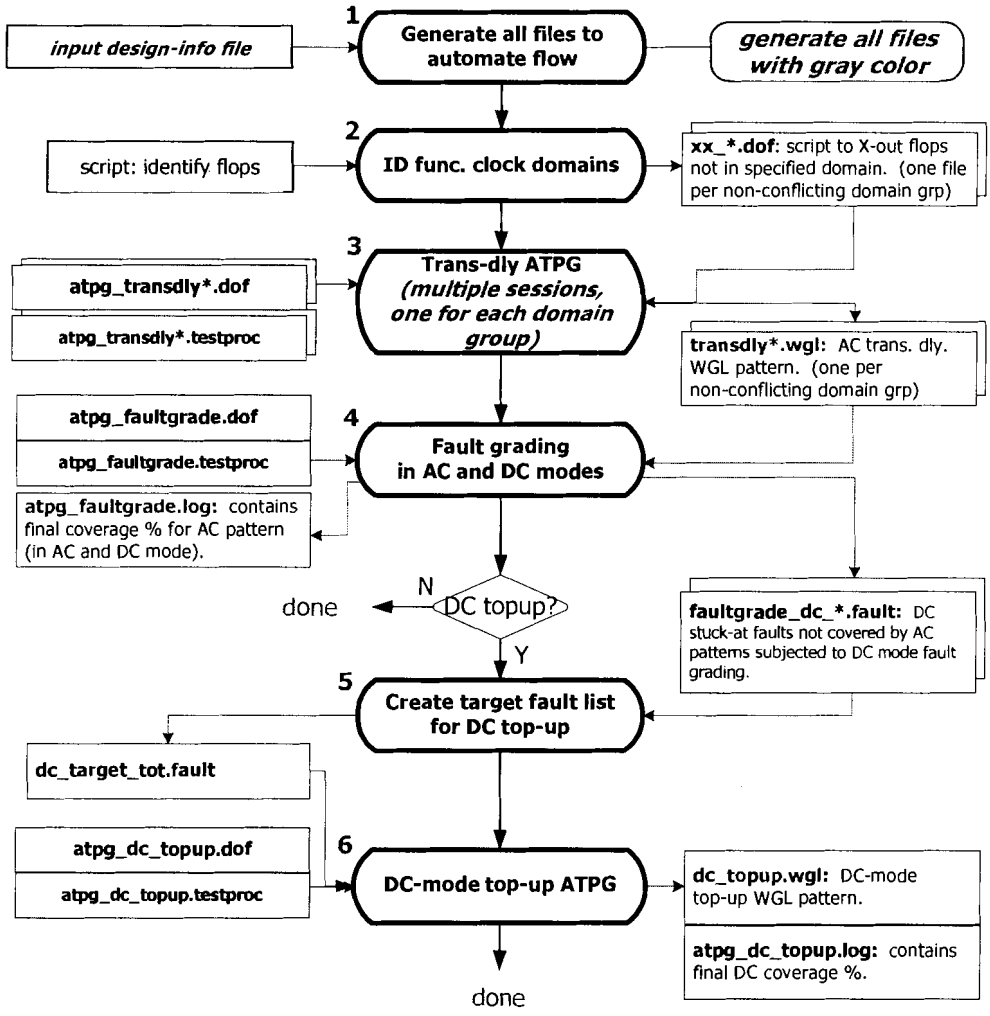
The diagram above shows five scan flops linked together into one single scan chain. SE pin of all flops are connected together, as with CP pins. The 0➔1 transition on the input of the AND gate connected to Q of flop D is targeted transition-delay ATPG. The basic idea here is that the pattern must launch a 0-to-1 transition on the targeted input of the AND gate; all other inputs to the AND gate must stay transparent to allow the targeted transition through the gate; finally the destination flop must capture the post-transitioned value so that it can be shifted out. Here is how the test is accomplished in detail: A '0' is shifted into flop D to drive the targeted input of the AND to '0'; A '1' is shifted into flop C such that the OR gate drives a '1' into the non-targeted input of the AND, hence allow the preset value '0' to be observed by flop E. Then SE (also called scan_en) drops to '0' to activate functional paths. The launch clock pulse launches the

'1' shifted into flop A across flop D to start the propagation. This pulse also launches the '1' shifted into flop B across flop C to maintain the '1' on the OR gate. The capture pulse, which follows immediately after the launch pulse to form the at-speed clock period, captures the '1' through the AND gate from flop D onto flop E. Then the shift-out operation will pipe the data on flop E out to the scan_out port for observation. In the presence of an at-speed defect on the targeted input of the AND gate, the transition launched from flop A would have been too slow to register the '1' on flop E, causing a mismatch.

3.2 Transition-Delay ATPG Flow Description

With the concept of AC-Scan transition-delay ATPG understood, an automation flow is created to generate test transition-delay patterns on designs requiring at-speed defect detection. Generally speaking, the flow specifies design-specific double-pulse sequences and relies on the ATPG tool to generate patterns accordingly, but there are more details involved. The diagram below is a flow chart of AC-Scan ATPG flow for Transition-Delay model. It is broken down into 6 stages each of which is described more closely in the paragraphs following:

Figure 3: AC-Scan Transition-delay ATPG flow chart



3.2.1 Generate all files to automate flow

This is stage one of the flow. Since the flow is automated and generic across all designs, user only needs to enter information needed by the flow in a design-info file. This file is fed into a script that generates all files necessary to accomplish downstream tasks. There are several advantages to this type of design methodology:

- Automated design flow shortens development time and increases efficiency, allowing designers to concentrate on tasks specific to the design.

- A single data file at the start of the flow creates a clear point of intrusion, making debugging easier.

- Running through ATPG requires expertise in the area of DFT. The flow is designed to provide a higher level of abstraction to allow designers without relevant knowledge and/or experience to still accomplish their jobs. In essence, automation flows are designed to shield designers from the intricate details of the tools contained within. This creates more clear-cut knowledge boundaries and better technical organization.

3.2.2 Identify and group functional clock domains

This is stage two of the flow. In this stage, static timing analysis tool is used to classify different branches of the scan-mode clock source into functional clock domains.

A piece of integrated circuit can be partitioned according to the clock source of a collection of sequential elements. A group of flops driven by the same clock source in normal functional/operational mode, along with the combinational logic surrounding the flops, forms a single partition, called a functional clock domain. Each functional clock domain toggles at its own specified speed according to design requirements. Therefore, to perform at-speed test, it is necessary to be able to exercise each domain at its own clock frequency. This presents a unique challenge in scan-mode as, due to pin limitation and characteristics of clocking strategy, several functional domains may be driven from the same scan-mode clock source/pin, making it hard to target only a particular portion that corresponds to a functional domain. To be more specific, if a scan-mode clock pin toggles two functional clock domains, one at 77MHz, the other at 311MHz, testing the 311MHz portion of the flop collection at-speed would result in massive false failures in the 77MHz portion because data paths formed by the flops in the 77MHz domain are not

designed to run at any higher speed. Alternatively, testing the entire collection of flops at 77MHz would mean that the 311MHz portion is not stressed enough to present realistic test coverage.

Therefore, this stage of the flow tries to partition and cleverly group the different functional domains within the scan-mode clock source(s), and then builds the ATPG invocation files to help target only a single group of non-conflicting functional domains for each ATPG run coming up later. Each invocation file targets a particular domain group by masking out all flops not in the targeted group.

The grouping of functional domains depends on pin equivalency of the scan-mode clock sources/pins. The default behaviour is that domains can only be grouped together within a single ATPG run if they are all driven from different scan-mode clock pins and none of the clock pins are pin-equivalent (i.e. none are forced to toggle at the same time during capture mode of scan operation). The following example illustrate this idea better:

A block of circuits has five functional clock domains (A, B, C, D and E) and three scan mode clock pins (clk1, clk2 and ecbi_wrb), and following configurations:

- Clk1 controls functional domain A during scan testing.
- Clk2 controls functional domains B, C during scan testing.
- Ecbi_wrb controls functional domain D, E during scan testing.
- Clk1 and clk2 are pin-equivalent (i.e. they are to toggle in identical fashion).

Therefore, according to the rules above, here are the clock groups:

- group 1: toggle clk1 and ecbi_wrb to target domains A and D respectively.
- group 2: toggle clk2 and ecbi_wrb to target domains B and E respectively.
- group 3: toggle clk2 to target domain C

Each group will become an ATPG run in the next stage.

3.2.3 Performing ATPG in AC transition-delay mode

This is stage three of the flow. In this stage, ATPG is run for each group of functional clock domains using the generated files from stages one and two. If run successfully, each ATPG will result in the generation of a pattern file containing AC-scan transition-delay patterns targeting a specify group of clock domains.

3.2.4 Fault grading in AC and DC modes

In this stage, the patterns generated from the previous stage is fed back into the ATPG tool to find the total coverage offered by the entire collection of patterns. This is a process known as fault-grading. Even though the intended targets of all ATPG runs are mutually exclusive, some tested faults, such as the faults in the scan chains, are still accounted for in more than one run. Therefore, fault-grading is not as simple as summing together the coverage number of all runs.

Fault-grading is performed with AC transition-delay fault model to produce the final AC coverage percentage, as well as with DC stuck-at fault model to produce the achieved DC coverage embedded in the AC pattern. [7] Also, a list of DC-mode non-testable faults is written out in preparation for DC (stuck-at) pattern top-up.

3.2.5 Perform ATPG in DC (stuck-at) mode for coverage top-up

Stage four of the flow created coverage percentage of the AC patterns in DC mode as a side benefit. At the same time, a list of non-testable faults is in DC mode generated. In stages five and six, this list is collected to form the target fault list for DC-mode pattern top-up. The purpose of this top-up is to save tester memory and test time by avoiding generating full DC patterns. In stage six, ATPG is invoked in DC mode using the generated invocation files from stage one and the fault list from stage five. The run should result in the generation of a DC pattern file targeting the fault list. The final total DC coverage is recorded in the log file. This completes the flow.

3.3 Benchmarks

Typical AC coverage for a block with 95% DC coverage is between 60% to 85%. AC pattern/vector count is usually between three to five times of the DC counterpart. Here are some of the reasons for the lower coverage in AC when compared to DC:

1. Reset lines are not tested. This is acceptable because reset typically only requires DC coverage.

2. Cross-clock paths are not tested. Since ATPG targets each functional clock domain, data paths traversing through the domains are not tested. This is usually an acceptable limiting factor in coverage target as well, because cross-clock paths are usually false-paths and not qualified at a particular clock frequency.

3. Block boundaries are hard to test due to the double-capture scheme. This characteristic will be explained further in detail in the next section.

4. Transition-delay model is a more complex model than the stuck-at model because it involves setting up a targeted node with two opposing logic values to test for any one fault, whereas stuck-at model only requires a single value to be propagated onto the node. This added complexity inherently results in negative impact on the performance of the ATPG tools and therefore pattern generation efficiency and effectiveness. [7]

5. This flow does not test RAM interfaces. RAM testing requires a more elaborate ATPG algorithm called multi-load ATPG. In multi-load ATPG, data is written into selected locations of the RAM in at-speed fashion in one pattern, then is retrieved in at-speed fashion from the same locations in a later pattern. This requires the ATPG tool to keep track of the content of RAM at all times. To ensure pattern efficiency on the logic away from RAM interfaces, RAMs will be tested in a completely separate AC-scan ATPG run and the coverage gained from there is additive to the results obtained in this flow.

3.4 Scan Strategy for AC-Mode ATPG

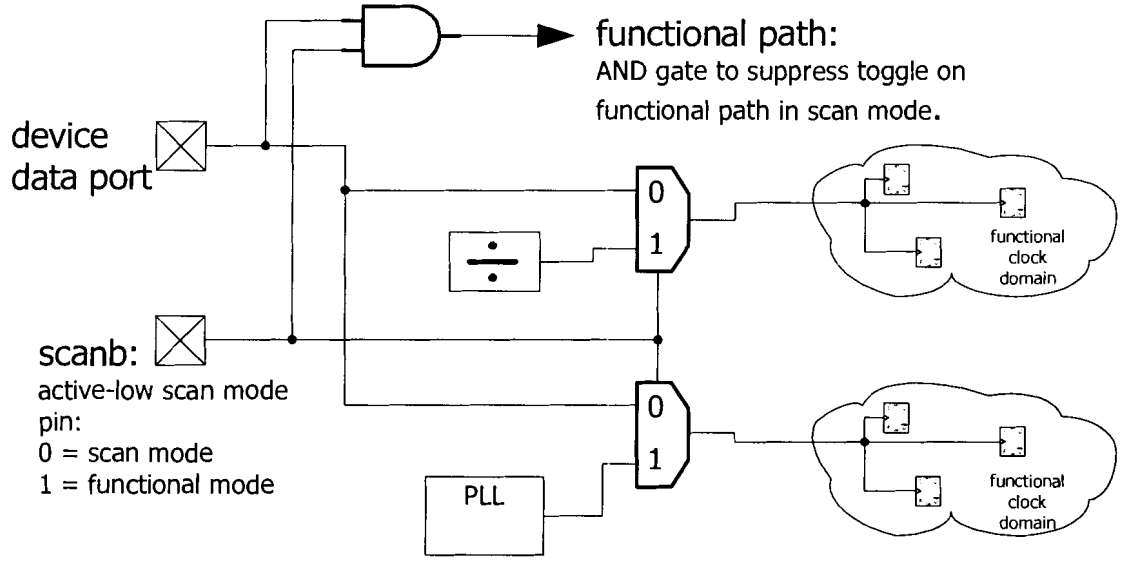
This section introduces some of the scan strategies to enable AC transition-delay ATPG and DFT in general. More specifically, the necessity of scan clock sharing is described as well as strategies for coverage increase such as block-level ATPG and launch-off-shift ATPG.

3.4.1 Pin sharing for scan clocks

As mentioned earlier, a device can be partitioned according to functional clock domains. Flops inside each functional clock domain are driven by the same clock source. There are two classes of clock sources, those coming directly from device-level

input ports, and those driven internally by phase-lock-loops (PLLs), clock dividers, or interrupts which are in the forms of clock lines driven by output of a flop. The lack of direct top-level pin controllability in the second class of clock sources poses a DFT problem since a flop without a clock port at device-level cannot be inserted into scan chains. Therefore, non-timing-critical data ports are selected to multiplex into the clock path in scan mode and serve as the clock source (see diagram below). This allows the flops in a clock domain with internal functional clock source to be scanned, hence increases testability of the device.

Figure 4: Clock sharing in scan mode



3.4.2 Testability with block-level ATPG

Netlists with gate count beyond 1.5 Million are usually too large a target for any of the currently available ATPG tools. Therefore, to make DFT possible for large devices, hierarchical DFT is used such that sub-blocks of the device are targeted individually for scan insertion and ATPG. ATPG creates test vectors which, in essence,

is a testbench containing stimuli to the DUT and expected response from it. In order to ensure that patterns generated at block level can still be applied at device-level, functional I/Os of the block need to be masked out such that the only useable I/Os are scan-related pins such as scan-in bus, scan-out bus, scan-enable (also referred to as SE), and scan mode clocks. This type of pin masking is necessary because the only pins that can be guaranteed by a generic DFT implementation methodology to map one-to-one from block-level to the device are scan-related pins. All connections of block-level functional I/Os are design-dependent and cannot be used during block-level ATPG because their top-level wiring is unknown at this stage.

Masking out functional I/O results in a testability hit because inputs will be launching 'X' into the boundary interface and output cannot be used to observe data coming out of them. Though, with DFT-friendly partition practices, design can be broken into block in which the block-level functional I/Os are immediately registered by flops on the boundary of the block. This will minimize the negative impact of testability drop due to uncontrollable/unobservable functional I/Os in DC scan mode since there is minimal logic on the edge of the block.

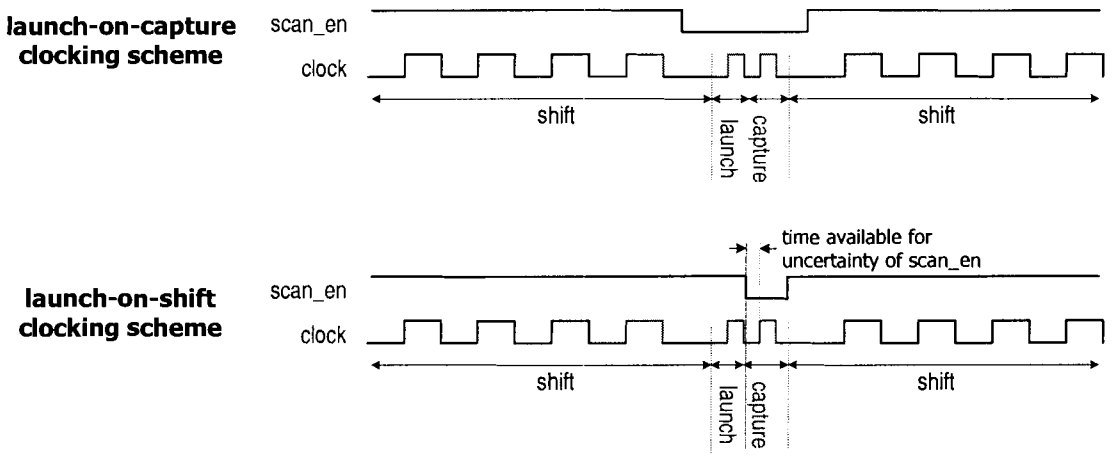
However, providing registers for functional I/Os alone is not sufficient for maintaining reasonable AC scan coverage. Because testing circuit at-speed requires two consecutive clock pulses, the X's on the inputs are launched across the boundary flops and contaminates logic between the boundary flops and core flops. The impact on test coverage by this behaviour depends on the amount of logic contaminated by the X's and can be quite significant. To resolve this issue, the following workaround is introduced: Scan input boundary flops of the block into a standalone scan chain with a dedicated scan-enable (SE). For block-level AC-scan ATPG, hold scan-enable of the input boundary scan chain constantly high such that chain structure is maintained

through the double-pulse of the scan clock(s). This way, value launched into the logic after an input boundary flop will be data shifted into the previous scan element in the chain, instead of the 'X' from functional input. Note that since the scan chain is maintained, scan paths in the chain will be targeted for at-speed test. Therefore it is important to either specifically exclude the scan paths in ATPG or synthesize the input boundary scan paths at functional speed.

3.4.3 Launch-on-shift ATPG

So far, the double-pulse clocking scheme discussed is termed as broadside or launch-on-capture. The name comes from the fact that scan-enable (SE) falls to '0' long before the start of the launch pulse and stays at '0' through both clock pulses. [7] This approach best mimics the functional behaviour of the DUT. However, it places strains on the ATPG tool to back-track through potentially large logic cones to place onto the D input of flops the correct logic value so that a properly transition can be launched through the double-pulse. This peculiarity, depending on the capability of the ATPG tool, can limit test coverage and result in long run time.

Figure 5: Launch-on-capture VS. launch-on-shift



The launch-on-shift scheme (as shown in the above diagram) offers a slightly different way of controlling scan-enable (SE). With this method, scan-enable is held high through the launch pulse and drops before the capture pulse. Its advantage is that the launch data becomes the output of the previous scan-element in the scan chain rather than the output of logical cones in the functional path. This makes ATPG much easier on the tool because the launch pulse becomes the final shift cycle and the tool can avoid traversing clouds of combinational logic to derive launch values. The problem with this approach is the strain put into building a tight clock tree for scan-enable (SE). Because the double-pulse is issued at-speed, most of the time there is only a few nanoseconds between the pulses. This is not a lot of time to have scan-enable fully propagated into the SE pin of every flop in the design. Balancing such a large clock-tree is quite challenging, and much of the time impossible given the area constraints and routing congestion. Therefore, launch-on-shift is not widely adopted in the industry.

CHAPTER 4 PATH-DELAY ATPG

4.1 Path-Delay ATPG Methodology for Speed-Grading/Binning

Before diving into the specifics of the speed-grading/binning with scan vectors, it is worthwhile to discuss the specific characteristics of test vectors used for speed-binning/grading. A synchronous digital design consists of sequential elements such as flip-flops and data-paths linking the sequential elements to each other and to device-level I/O ports. For the discussion of speed-grading/binning, the definition of a data-path can be simplified to be a passage from a source flop to the destination flop through a collection of combinational gates. Within a clock domain, upon a clock edge, all flops simultaneously sample data fed into them by the data-path attached to their inputs, and launch that same data to the paths connected to their outputs. This means that data sent out from the source flop on a clock edge must arrive/settle at the destination flop before the next clock edge, or a setup timing violation has occurred. Since data-paths vary in length, the longest path, called the critical path, determines the gap between two consecutive clock edges and therefore the operational frequency of the device. Typically, speed-grading/binning vectors are designed to exercise the critical paths in the design through consecutive clock pulses. By shrinking the duration between the clock pulses, the exercised paths will have less time to complete their data propagation from the source flop to the destination flop. The minimum gap between the clock pulses, or the minimum clock period, such that data can still be transferred across the targeted data-paths marks the speed or grade of the device. Therefore, for speed-grading, the same set of test vectors are run on the DUT multiple times, each time with an

incrementally shorter clock period until mismatches occur between the DUT's actual response and expected outcome. Then the DUT is said to be graded for the most recent passing clock period/frequency. A speed-binning operation is similar to speed-grading, except with only a few steps of clock period increments. During speed-binning, critical paths of a DUT is subjected to testing with the same vectors multiple times, each at a different prescribed clock speed. Each clock speed defines a class of the same device capable of operating at a specified operation frequency. Therefore, the DUT passing one speed specification or "bin" but failing the next is placed into the passing bin.

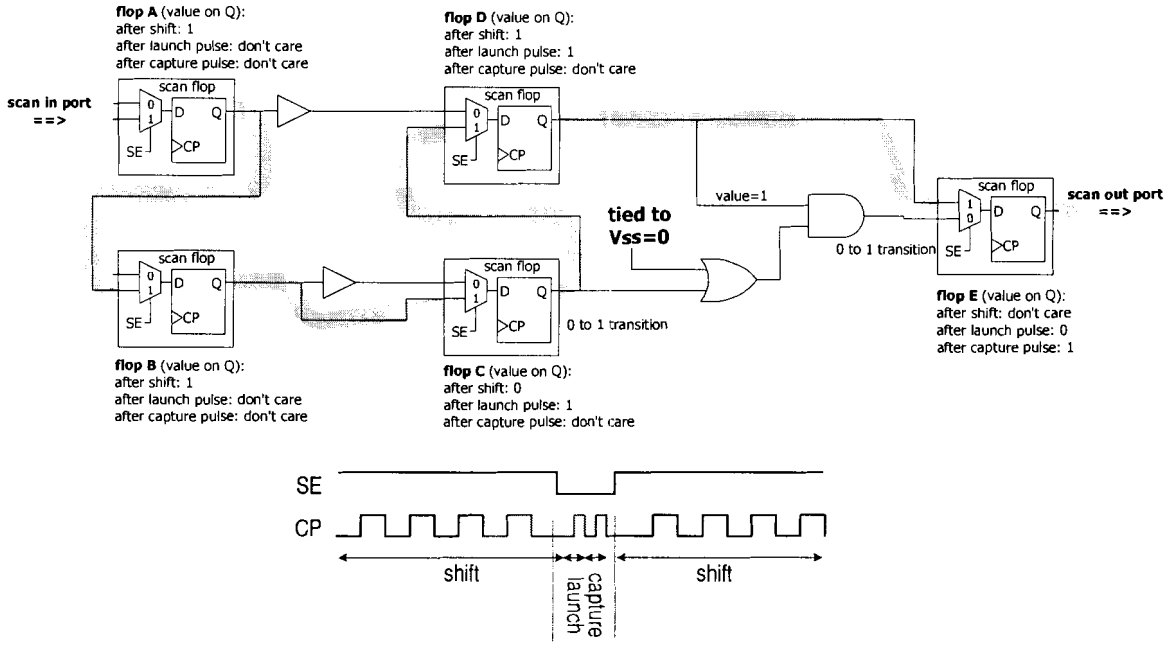
In order to create speed-grading/binning test vectors efficiently and automatically, scan chains must be utilized to convert all flops into control and observe points. This way, ATPG tool can be used to generate the test vectors automatically. In order to do that, a new fault model is required: Path-delay fault model.

Path-delay fault model models an entire data path with two faults: A slow-to-rise fault that states a 0-to-1 transition on the source flop's output pin must invoke a transition on the input of the destination flop within a clock period. A slow-to-fall fault that states a 1-to-0 transition on the source flop's output pin must invoke a transition on the input of the destination flop within a clock period. [11]

Today's ATPG tool is designed to support the path-delay model, and creates two consecutive clock pulses at fast clock frequency, a procedure called double-pulse. This process is similar to transition-delay ATPG. Using scan-shifting, it first loads/presets the source flop's output with the pre-transition value and its data-input with the value to be transitioned, then with SE de-asserted, launches the transition value with the first clock pulse of the double-pulse, and tries to capture the transition value with the second pulse on the destination flop. Then content of the destination flop (which is a scan cell inside a

scan chain) is shifted out for comparison. Please note that the model itself does not specify the clock speed, but rather just specifies the values before and after the transition. As long as a transition on the data-path of interest (most likely a critical path) is properly launched and captured through the double-pulse of the clock, the path is considered to be sensitised at-speed. This way, it is up to the designer to choose a clock frequency of the double-pulse and the tool to implement it. The following diagram provides an example of path-delay ATPG and the method of detecting at-speed faults using double-pulse of the clock.

Figure 6: AC-Scan path-delay ATPG example



The diagram above shows five scan flops linked together into one single scan chain. SE pin of all flops are connected together, as with CP pins. The targeted fault is the slow-to-rise fault on the data-path from flop C to flop E through the OR gate and the AND gate. To test for this fault, a 0➔1 transition on the Q of flop C needs to be created

and launched through both OR-AND gates through double-pulse of the clock, and the post-transition value ('1') needs to be captured into flop E after the end of the double-pulse. Therefore, here are the values on the flops after shift-in:

- A '0' is stored in flop C through scan shifting to serve as the pre-transition value.
- A '1' is stored in flop B also through scan shifting, and therefore appears at data-input of flop C, to serve as the post-transition value. When SE drops to '0', this '1' is launched across flop C during the first pulse of the double-pulse, then propagates through the OR and AND to be captured onto flop E on the second pulse of the double-pulse.
- A '1' is stored through scan shifting in flop D to hold the AND gate transparent.
- A '1' is stored in flop A through scan shifting, and therefore appears at data-input of flop D, to be launch across flop D during the double-pulse. This maintains the transparent state of the AND gate so that data can propagate through it from flop C.

4.2 Path-Delay ATPG Methodology for Input/Output Characterization

In order for a piece of IC to fit into a board-level system, it must respect the input and output timing of its neighbouring ICs to properly communicate with them. For example, data-out bus of a device is usually specified to send out new bits close to the rising edge of the clock-output signal. The gap between clock edge and the data-edge is usually part of the timing specification and is strictly enforced by the design of the device. With this timing behaviour clearly defined, the downstream module receiving data from the device can be designed to sample its corresponding inputs accordingly based on clock arrival. In order to verify that all I/O timing specifications have been

satisfied, test vectors need to be created to sensitize the data-paths on the boundary of the device.

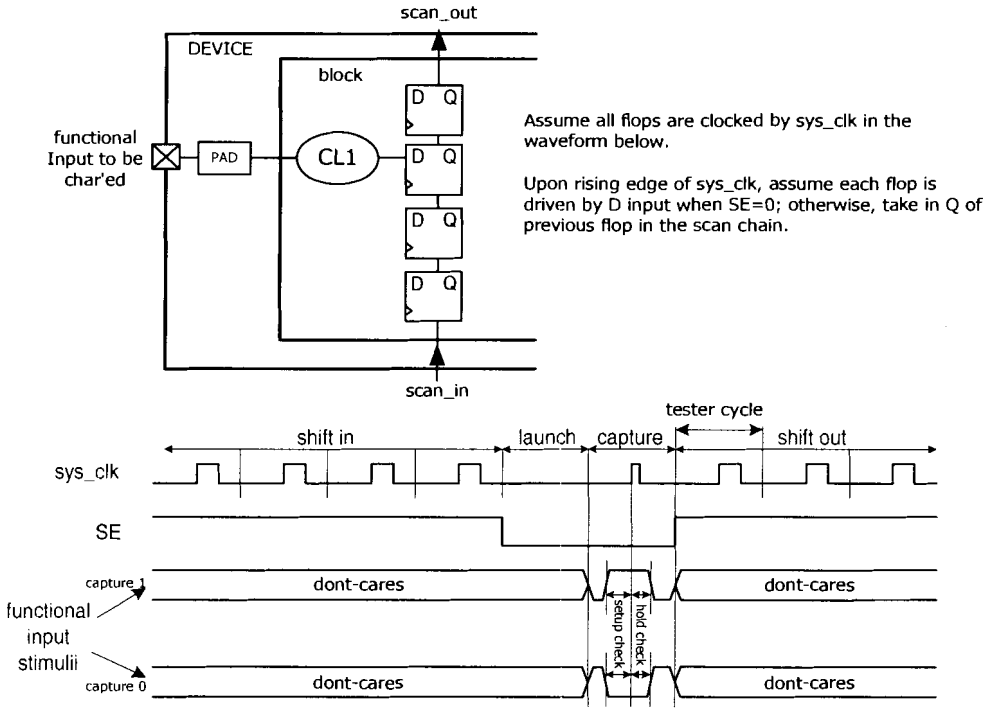
I/O characterization vector creation on a device with scan chains properly constructed can be accomplished by extending the capability of AC-Scan path-delay ATPG. With minor modifications to the ATPG tool's invocation and some post-processing of the pattern files generated, one can characterize device-level I/Os by feeding to the ATPG tool data-paths from inputs to flops and/or from flops to outputs. When patterns are created to properly sensitize these boundary paths, Test Engineer can move the data edges of the I/Os to properly characterize their timing.

The following shows the exact operation of the various types of I/O characterization. The actual pattern generation will follow the waveforms described here. In general, the patterns follow the sequence of multi-cycle shift-in operations, then a launch cycle followed by a capture cycle to drive/sample the I/Os and the associated flops, then finally a multi-cycle shift-out operation.

4.2.1 Input setup/hold characterization

Timing of data transmitted into a device's input ports is usually specified as time between data transition before the clock edge (setup time) and after the clock edge (hold time). Patterns need to be generated to verify that the device can properly sample incoming data on the input ports when they are sent in according to the setup and hold timing specifications.

Figure 7: AC-Scan input setup/hold characterization



The diagram above shows the operations of input characterization. During input characterization, the shift-in operation usually sets up the data-path from the functional input to the destination flop such that this path can be sensitized properly when SE drops to '0'. Then SE is set to '0' at the beginning of the (empty) launch cycle. Then in the capture cycle, with the data-path ready to be sensitized, a 0→1→0 or 1→0→1 data sequence is applied to the functional input such that the triggering clock edge is situated in the middle value of the sequence. The middle logic value is captured into the destination flop by the clock pulse. Finally, the captured value is shifted out for comparison. The test pattern is generated with the data edges far away from the clock edges to guarantee that the pattern passes. Then test engineer will rerun the test pattern by moving the data edges closer to the clock edge until a failure occurs. The

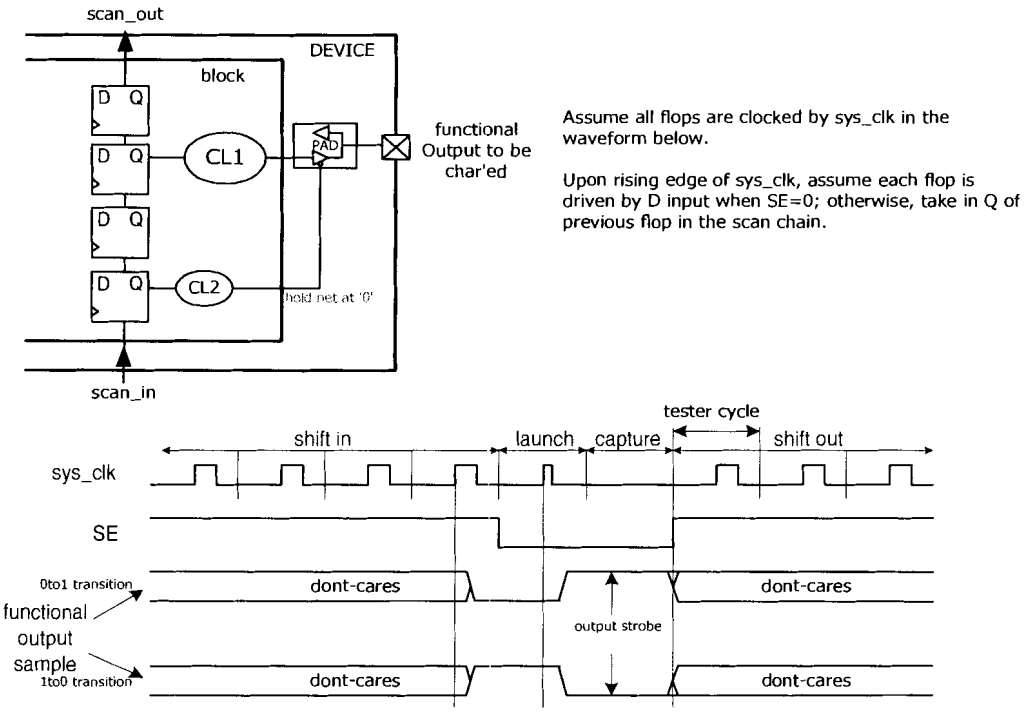
location at which the failure occurs provides the setup and hold time achieved by the device. This can then be compared with the timing specification intended by the design.

4.2.2 Output data propagation characterization

Timing of data transmitted out of a device's output ports is usually specified as the maximum allowed time delay of the data-out value compared to a clock signal (output propagation time). In cases where an entire data bus is transmitting, there is usually also a timing specification on the maximum skew among every index of the bus, i.e. all output transition must occur relatively close to each other within a time window. To verify either timing specification, patterns need to be generated to sample the output ports after a clock edge. Then, the longest delay subtracting the shortest delay provides the skew measurement.

The diagram below shows the operations of output data transition characterization, where the data path from the flop to the output port through CL1 is to be timed. During this characterization, the shift-in operation loads the data-out flop with the value before the transition (in case of 0-to-1 transition, '0'). Also, the values loaded into the scan chains (not shown in diagram) provide the post-transition value onto the D pin of the data-out flop. This way, during the launch cycle, when SE drops to '0' and the clock is pulsed, the value on the D input is launched across the flop to start the propagation. A strobe point is placed in the capture cycle to sample the post-transition value on the output port. A test engineer can move the strobe point forward in time to find the time of failure. The time of failure is the measured output timing on the port and can be compared with the intended timing specification.

Figure 8: AC-Scan output data-propagataion characterization

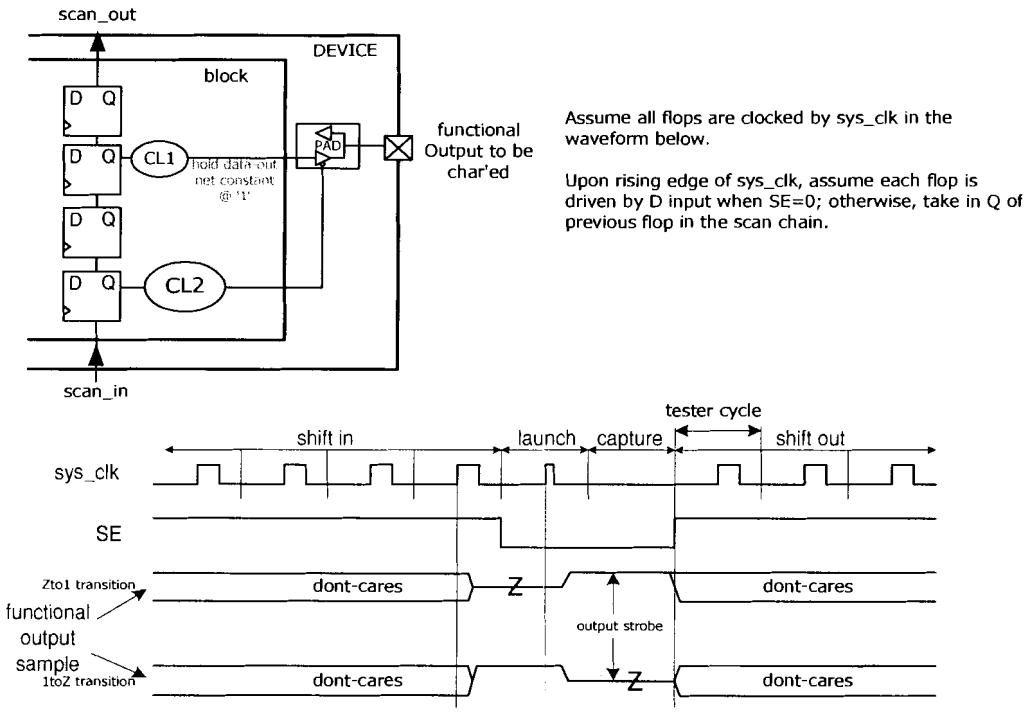


Throughout both the launch and capture cycles, the patterns generated by the ATPG tool need to hold the active-low output-enable at '0' to ensure that the values on data-out can propagate out without obstruction. The placement of this type of sensitization behaviour can be done in the ATPG tool through provision of ATPG constraints within the tool.

4.2.3 Output tristate characterization: 1←→Z

Often for tri-stateable output ports or bi-directional ports, there are timing specification on the time needed to enable/disable the output. In this characterization exercise, transition between tristate and '1' of the port can be timed.

Figure 9: Output tristate characterization (1 and Z)

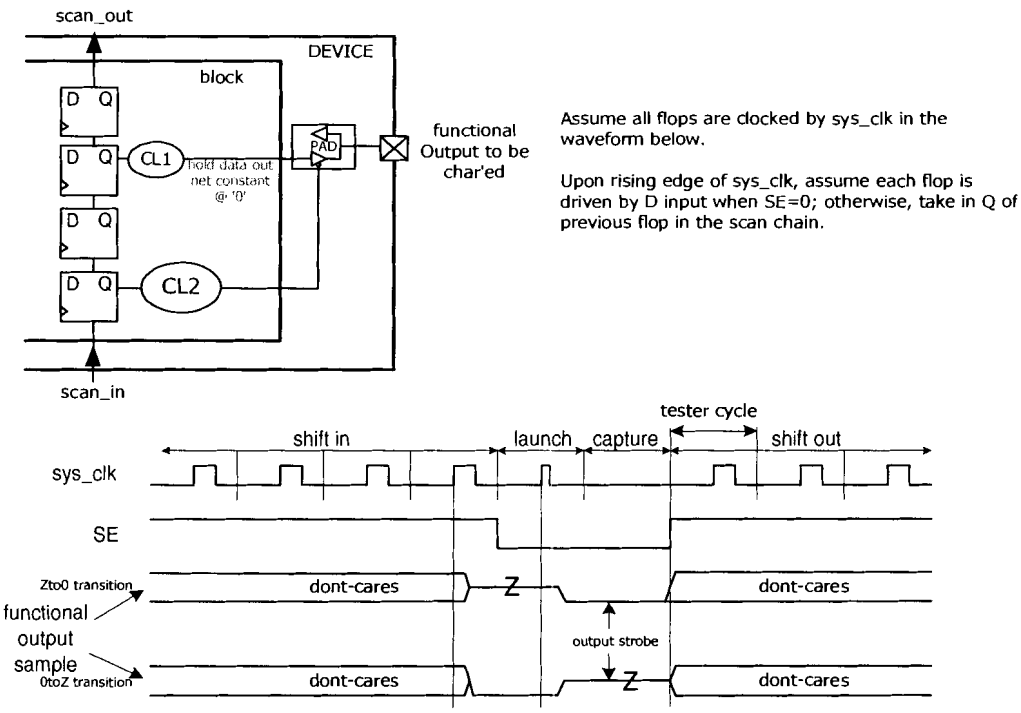


The above diagram shows the operations of characterization tristate-to-one and one-to-tristate timing on an output port, where the data path from the flop to the output port through CL2 is to be timed. During this characterization, the shift-in operation loads the output-enable flop with the value before the transition ('1' for case of Z-to-1 transition, '0' for case of 1-to-Z transition). Also, the values loaded into the scan chains (not shown in diagram) provide the post-transition value onto the D pin of the output-enable flop. This way, during the launch cycle, when SE drops to '0' and the clock is pulsed, the value on the D input is launched across the flop to start the propagation. A strobe point is placed in the capture cycle to sample the post-transition value on the output port. Test engineer can move the strobe point forward in time (possibly into the launch cycle) to find the time of failure. The time of failure is the characterized output timing on the port.

Throughout both the launch and capture cycles, the patterns generated by The ATPG tool need to hold the data-out port at '1' to ensure that the device-level port registers a '1' when output is enabled. The placement of this type of sensitization behaviour can be done in the ATPG tool through provision of ATPG constraints. In fact, due to the tool's inability to directly handle data-paths through output-enable of tristate buffers, pattern is generated at block-level with 0←→1 transitions on the block-level output-enable port. Then the pattern is post-processed to convert the 0←→1 transitions into 1←→Z transitions.

4.2.4 Output tristate characterization: 0←→Z

Figure 10: AC-Scan output tristate characterization (0 and Z)



Similar to the above section, this operation is to time transition between tristate and '0' on an output or bi-directional port. The above diagram shows the operations of characterization tristate-to-zero and zero-to-tristate timing on an output port, where the data path from the flop to the output port through CL2 is to be timed.

During this characterization, the shift-in operation loads the output-enable flop with the value before the transition ('1' for case of Z-to-0 transition, '0' for case of 0-to-Z transition). Also, the values loaded into the scan chains (not shown in diagram) provide the post-transition value onto the D pin of the output-enable flop ('0' for case of Z-to-0 transition, '1' for case of 0-to-Z transition). This way, during the launch cycle, when scan_en (SE) drops to '0' and the clock is pulsed, the value on the D input is launched across the flop to start the propagation. A strobe point is placed in the capture cycle to sample the post-transition value on the output port. A test engineer can move the strobe point forward in time (even into the launch cycle) to find the time of failure. The time of failure is the characterized output timing on the port.

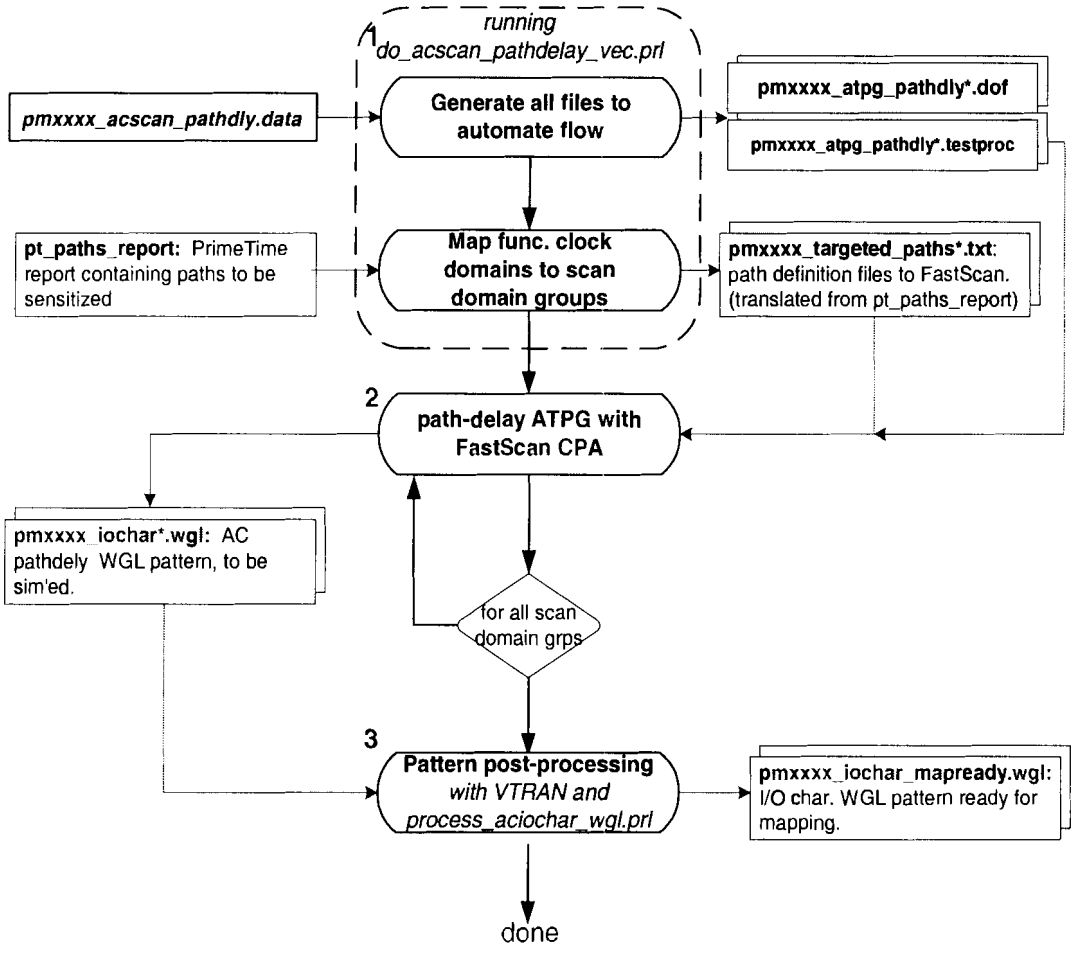
Throughout both the launch and capture cycles, the patterns generated by the ATPG tool need to hold the data-out port at '0' to ensure that the device-level port registers a '0' when output is enabled. In fact, due to the tool's inability to directly handle data-paths through output-enable of tristate buffers, pattern is generated at block-level with 0←→1 transitions on the block-level output-enable port. Then the pattern is post-processed to convert the 0←→1 transitions into 0←→Z transitions.

4.3 Path-Delay ATPG Flow Description

The diagram below is a flow chart of AC-Scan ATPG flow for Path-Delay model. It is broken down into 3 stages each of which is described more closely in the

paragraphs following. Please note that for I/O characterization, since there are four types of characterizations as described in the last section, each type of characterization requires a separate invocation to the flow.

Figure 11: AC-Scan path-delay ATPG flow-chart



4.3.1 Generate all files to automate flow

This is stage one of the flow. Since the flow is automated and generic across all designs, user only needs to enter information needed by the flow in a design-info file.

This file is fed into a script that generates all files necessary to accomplish downstream tasks. There are several advantages to this type of design methodology:

- Automated design flow shortens development time and increases efficiency, allowing designers to concentrate on tasks specific to the design.
- A single data file at the start of the flow creates a clear point of intrusion, making debugging easier.
- Running through ATPG requires expertise in the area of DFT. The flow is designed to provide a higher level of abstraction to allow designers without relevant knowledge and/or experience to still accomplish their jobs. In essence, automation flows are designed to shield designers from the intricate details of the tools contained within. This creates more clear-cut knowledge boundaries and better technical organization.

Included as part of the file generation is a list of path files each containing the critical paths within a clock domain. There are as many of these files as the number of clock domains.

4.3.2 Performing ATPG in AC path-delay mode

This is stage two of the flow. In this stage, ATPG is run for each group of critical paths. If run successfully, each ATPG will result in the generation of a WGL pattern/vector file containing AC-scan path-delay patterns targeting a collection of the paths for a specific clock domain. These WGL patterns are to be translated into simulation testbenches and simulated for verification. For speed-grading/binning ATPG, the flow ends here. For I/O characterization, since the vectors are generated at block-level, the next step prepares for vector mapping to device-level.

4.3.3 Pattern post-processing

As mentioned, all characterization patterns are generated at block-level to avoid tool limitation in handling tristate logic. Therefore patterns need to be mapped to device-level to be useable on production testers. For example, for tristate-to-one check on a particular pin, the actual tristate logic is outside the block-level circuit. Therefore The ATPG tool is fed a path from the output flop to the block-level output pin leading to the output-enable of the tristate logic/pad. Consequently, the block-level pattern generated by The ATPG tool only records '1' and '0' on the output-enable pin. In order to properly map the pattern to device-level, the pattern post-processing step creates a new pin that translates '1' and '0' on the output-enable pin into 'Z' and '1' respectively.

Other than pattern manipulation for tristate checks, other types of characterization may also require post-processing. This usually is related to scan-mode pin sharing. All scan-in/out pins of the device are shared with functional I/Os at the boundary of the device, i.e. outside the block. This means that for a device I/O reused for scan-in/out, the logic to realize the sharing of functionality (between scan and normal modes) is not present at block-level. i.e. the block would contain two pins which will be merged outside. Therefore in this situation, when the ATPG tool generates patterns on the block, it will place patterns onto both pins. Care has been taken to ensure that the patterns in both pins (stimuli or strobes) do not conflict with each other. The post-processing step would merge values on the two pins into a new pin.

4.4 AC Path-delay ATPG Issues

This section discusses some of the issues/peculiarities related to AC path-delay ATPG. More specifically, emphasis will be placed in deciding on acceptable test coverage, choosing a good clock source for I/O characterization, and path-selection.

4.4.1 Acceptable test coverage

For speed-grading/binning, while having the mindset of trying to achieve as high a test coverage as possible on the critical paths, there is no absolute requirement for the test coverage. If a number must be given, experience has shown that anywhere between 40-60% should be reasonably achievable. Since this test is not meant to detect manufacturing defects on the die, it does not require extremely high coverage to find gross performance outliers. Instead, it is trying to detect the effect of fabrication process variations on the DUT. Process variations, such as small shifts of doping level from wafer to wafer, provide gradual changes in circuit performance between neighbouring dies on a wafer or between dies on neighbouring wafers. [12] They affect entire dies, contributing to a slight overall performance enhancement or degradation of the dies. Therefore, testing a few representative paths may be adequate to gauge the overall performance of the device. The important point here is to ensure that the critical paths are targeted for ATPG. This may be somewhat tricky as, due to process variation, the critical paths reported from static-timing analysis of the design during chip development may become faster paths in silicon, making the other presumably shorter/faster paths the speed-determining paths. Therefore, it is necessary to choose at minimum a collection of paths that represents the top 10% of the total paths in terms of length. Path selection is discussed further in a later section.

On the other hand, 100% test coverage is required for I/O characterization. In these types of tests, each path chosen to be targeted for ATPG represents a part of a timing specification. Because every timing specification is required to be verified, every path must be properly sensitized, meaning 100% test coverage is a hardened requirement. If this is not achievable, functional vectors need to augment the AC-Scan I/O characterization vectors.

4.4.2 Clock source for I/O characterization

It is common for an input or output timing specification to be based on input clock edges. This characteristic provides an interesting challenge for scan-mode testing. More specifically, a typical functional-mode device-level clock pin toggles all flops under its control through a delay-lock-loop circuit (DLL). A DLL compensates for the delay introduced by the clock tree by adding the correct amount of extra delay to the clock line such that the clock signal at the device-level input clock and the signal at the clock pins of the associated flops are exactly 360-degrees out of phase. [13] In other words, the clock signal at the source and leaves of the clock tree are synchronized, but off by exactly one full clock cycle.

During scan, the DLL stops working because its flops operate as elements in the scan chain. However, for I/O characterization, the DLL needs to be operational and locked at the right functional clock frequency. This causes a logical conflict as AC-Scan path-delay ATPG for I/O characterization requires that the scan chain structure be maintained, and that the DLL be in functional state. The workaround to this issue is to scan the DLL separately from the rest of the digital logic such that the device contains two scan modes, one for the DLL and one for everything else. This way, path-delay I/O characterization ATPG would have access to the necessary scan chains and the functionalities of the DLL.

4.4.3 Path Selection Issues Related to Speed-Grading/Binning

Data paths selected for path-delay ATPG is usually chosen for their propagation time. The propagation time is reported from static timing analysis (STA) of the design by using commercial STA software such as PrimeTime from Synopsys Inc. Since STA results are obtained from timing models of gates and calculated timing delays of wires,

actual timing of the selected path in silicon may be different from the STA report. This presents a challenge for speed-grading/binning as a device that passes path-delay test at a particular clock frequency may in fact be faster or slower than the tested speed. Furthermore, actual critical paths in silicon may not even be reported as critical paths by STA prior to chip tape-out. Consequently, a way to make better use of path-delay ATPG is to find correlation between STA data, tested device speed, and actual device speed. Also, some techniques are needed to select the right collection of paths.

4.4.3.1 Some path-selection techniques

Since speed-grading/binning essentially checks for the effect of process variation on the IC's performance, it is important to choose a set of data paths that are evenly distributed across the die of the chip. [14] Also on the same note of testing process variation, Dr. Karim Arabi introduced the following concept during a verbal discussion of this issue. It is described as follow: All data-paths within a device can be classified according to their timing, forming a timing distribution. One can choose a group of timing paths within each section of the timing curve and test each group according to their maximum allowable clock frequency. The relative numbers of paths within all groups form a statistically representative sample of the full timing distribution. All targeted paths must pass path-delay test at their respective speed to qualify the device. This approach is better than only selecting the critical paths because it qualitatively verifies the correlation between tested speed and STA data.

4.4.3.2 Correlating path-delay results with actual device speed

A paper written by Bruce D. Cory, Rohit Kapur and Bill Underwood titled "Speed Binning with Path Delay Test in 150-nm Technology" introduced an interesting idea for

correlating path-delay results with actual device. It may be possible to apply its concept here to increase the usefulness of path-delay ATPG in the context of this project. Here is a brief description of the correlation process.

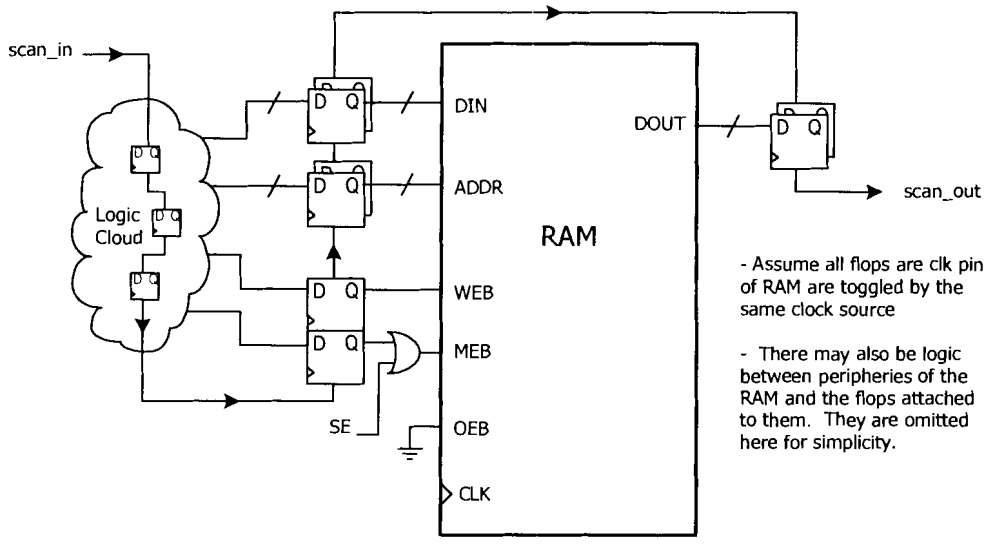
After tape-out of any design, the product will progress through a prototyping phase before production release. Prior to production release, the fabricated IC will be thoroughly tested to verify its behavioural functionality. This is not done by running scan vector on ATEs, but by plugging the device onto a system board that is designed in parallel to the design of the device itself. The board is meant to validate all functionalities of the device at the required clock speed to flush out all functional bugs. (Hopefully, there are no critical bugs discovered at this stage that warrants revisions of the design). This type of functional tests alone is actually sufficient to speed-grade/bin the chip. However, the test is usually too time-consuming to be economically performed on every device. This is the reason for path-delay ATPG. Since the functional prototyping tests can qualify the device at a maximum clock frequency (referred to as Fmax), and path-delay test can be performed on the same device, the failed clock frequency obtained from the path-delay test can be correlated with that of the functional test. [14] From this correlation, future devices produced after production release can simply be tested through path-delay scan vectors and qualified at a particular bin according the correlation relationship. The test-case shown in the aforementioned paper provided very clear and linear correlation between path-delay results and functional test results.

CHAPTER 5 AT-SPEED ATPG ON RAM INTERFACE

5.1 Overview of RAM Interface

Since RAM is a complex sequential module, testing its interfaces requires extra ATPG effort than the normal AC-Scan transition-delay ATPG. More specifically, it invokes the multi-load ATPG algorithm where a particular fault is tested through multiple shift loads of scan data. [15] For reasons of tool/pattern efficiency, this algorithm should be invoked separately on logic requiring its presence only (namely RAMs). Consequently, a flow for testing the RAM interfaces is developed as a standalone component, separated from the normal (or single-load) AC-scan ATPG flow. Since RAM testing with scan chains is relatively new to PMC-Sierra, much of the effort spend here involves understanding behaviour of the ATPG tool during the operation. The following sections will describe, among other topics, the author's understanding of how the ATPG tool targets and tests the different RAM interfaces with its built-in ATPG algorithms. This understanding was obtained through generating scan patterns on the interfaces of a test-case, simulating the patterns to ensure its integrity, and painstakingly tracing logic states of the RAM interfaces to interpret the purpose of all state changes on relevant nodes of the design asserted by the ATPG tool. This was a necessary and important step in gaining confidence in the tool's ability in order to develop an automation flow around it and release the flow to the design community of PMC-Sierra.

Figure 12: Simplified diagram of RAM interface



The above figure shows a simplified view of the RAM interface. Logic excluded from the diagram is mainly Built-In Self-Test (BIST) bypass logic between the peripheries of the RAM and the attached flops. The scan chain, shown as the arrowed line from scan_in to scan_out through all flops, does not dictate the order of the flops in the chain, but merely depicts that the sequential elements are scan-inserted and serve as proper control and observe points during ATPG. At this time, it is worthwhile to briefly describe the following pins of the RAM:

5.1.1 MEB: Active-Low Module Enable

This is the active-low module-select pin. Its functional logic needs to be gated with SE such that the RAM module is disabled during scan-shift operation. This is because of the multi-load nature of the RAM patterns. Much of the testing is done through writing data into certain RAM locations from data shifted in on one scan load,

and reading from those locations at the end of the next scan load. In order to preserve the content of the write operations through scan shifting, RAM needs to be disabled.

5.1.2 OEB: Active-low Output Enable

This is the active-low output-enable pin, and is present in some RAM modules. As part of the PMC standard, OEB is tied to '0' such that output is always enabled.

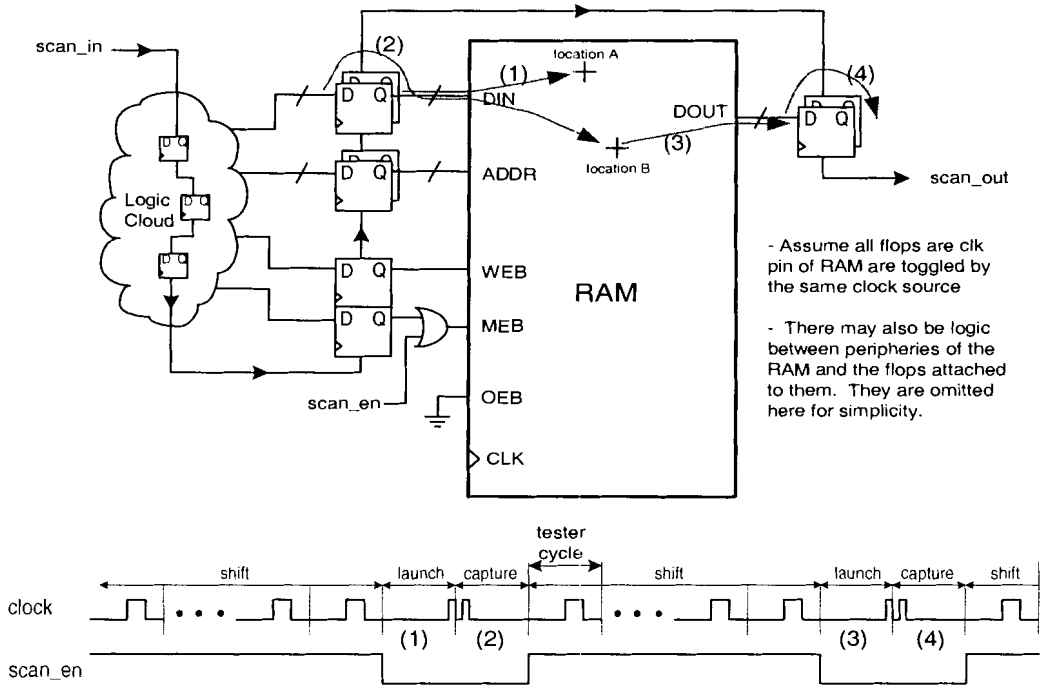
5.2 At-Speed ATPG Methodology for RAM Interface

There are three main interfaces to be tested: DIN, ADDR, and DOUT. Testing each interface requires a different ATPG approach, as implied by the ATPG tool. Please note that the ATPG methodology described here uses only transition-delay fault model because path-delay ATPG on RAM interfaces is not yet available at the time of this project. However, the impact of this deficiency is minimal to the higher-speed RAMs as the RAM interfaces are usually directly registered with minimal combinational gates in between. This makes the path-delay coverage implied if the interfaces are well tested in transition-delay mode.

5.2.1 Testing DIN of RAM

The diagram below depicts the strategy for testing DIN of RAM by the ATPG tool. This test involves two scan loads, one to write complementary bits to two different locations (marked as locationA and locationB in the above figure) in the RAM, and the other to read the content from those locations. Here is the description in detail:

Figure 13: Testing DIN of RAM



1. First apply a scan load (shift) that provides the following setup: Q pins of the ADDR flops would select locationA once latched in; D pins of the ADDR flops would select locationB once it is clocked to the Q pins and latched in; Q pins of the DIN flops would drive data (call it dataA) into locationA upon a clock pulse; D pins of the DIN flop would contain bits that are complementary to dataA (call the D pin values dataB) and should be driven into locationB after two consecutive clock pulses; Active-low write-enable (WEB) is setup to be '0' for the next two clock pulses to allow for two consecutive write operations.

2. As marked in the diagram as (1), scan_en drops to '0' and a clock pulse is issued to write dataA into locationA.

3. As marked in the diagram as (2), a second clock pulse is issued to write dataB into locationB in at-speed fashion. At this stage, the DIN input bus of the RAM has been sensitized at-speed due to the double pulse from (1) and (2). If there is an at-speed defect on a bit of the bus, that bit in locationB would have falsely received the corresponding bit in locationA. The rest of the sequence reads out the content of locationB to verify the write operaion.

4. Apply another scan load (shift) that provides the following setup: Q pins of the ADDR flops would select locationB; WEB is setup to be '1' to allow for a read operation.

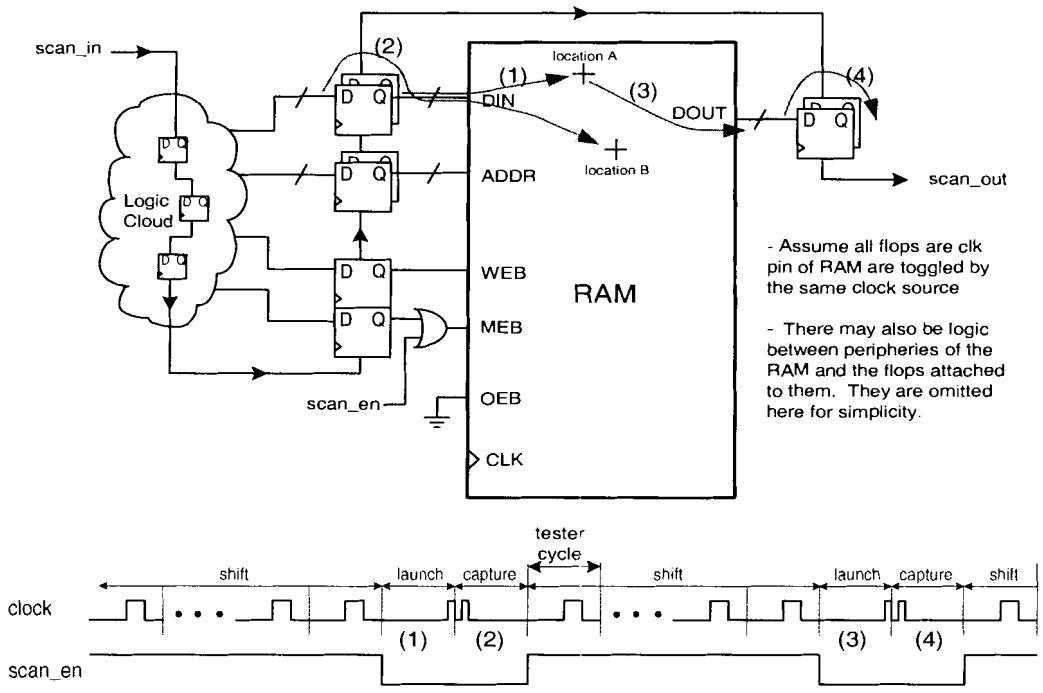
5. As marked in the diagram as (3), scan_en drops to '0' and a clock pulse is issued to read the content of locationB (dataB) to the DOUT bus of the RAM.

6. As marked in the diagram as (4), another clock pulse is issued to capture the values on DOUT onto the DOUT flops. Then the shift-out operation pipes the content of the DOUT flop to the scan_out port for sampling. This verifies whether the at-speed write operation has been done fault-free, and completes the test.

5.2.2 Testing ADDR of RAM

The diagram below depicts the strategy for testing ADDR of RAM by the ATPG tool. This test involves two scan loads, one to write complementary bits to two different locations (marked as locationA and locationB in the above figure) in the RAM, and the other to read the content from those locations. Here is the description in detail:

Figure 14: Testing ADDR of RAM



1. First apply a scan load (shift) that provides the following setup: Q pins of the ADDR flops would select locationA; D pins of the ADDR flops would select locationB once it is clocked to the Q pins; Q pins of the DIN flops would drive data (call it dataA) into locationA upon a clock pulse; D pins of the DIN flop would contain bits that are complementary to dataA (call the D pin values dataB) and should be driven into locationB after two consecutive clock pulses; WEB is setup to be '0' for the next two clock pulses to allow for two consecutive write operations. Please note that the actual addresses signifying locationA and locationB are only different by a single bit. The bit that is different is the target of the at-speed test.

2. As marked in the diagram as (1), scan_en drops to '0' and a clock pulse is issued to write dataA into locationA.

3. As marked in the diagram as (2), a second clock pulse is issued to write dataB into locationB in at-speed fashion. At this stage, the changing bit in the ADDR bus of the RAM has been sensitized at-speed from the double-pulse generated by (1) and (2). If there is an at-speed defect on that bit, dataB would be incorrectly written into locationA instead of locationB. The rest of the sequence reads out the content of locationA to verify that its content is dataA, not dataB.

4. Apply another scan load (shift) that provides the following setup: Q pins of the ADDR flops would select locationA; WEB is setup to be '1' to allow for a read operation.

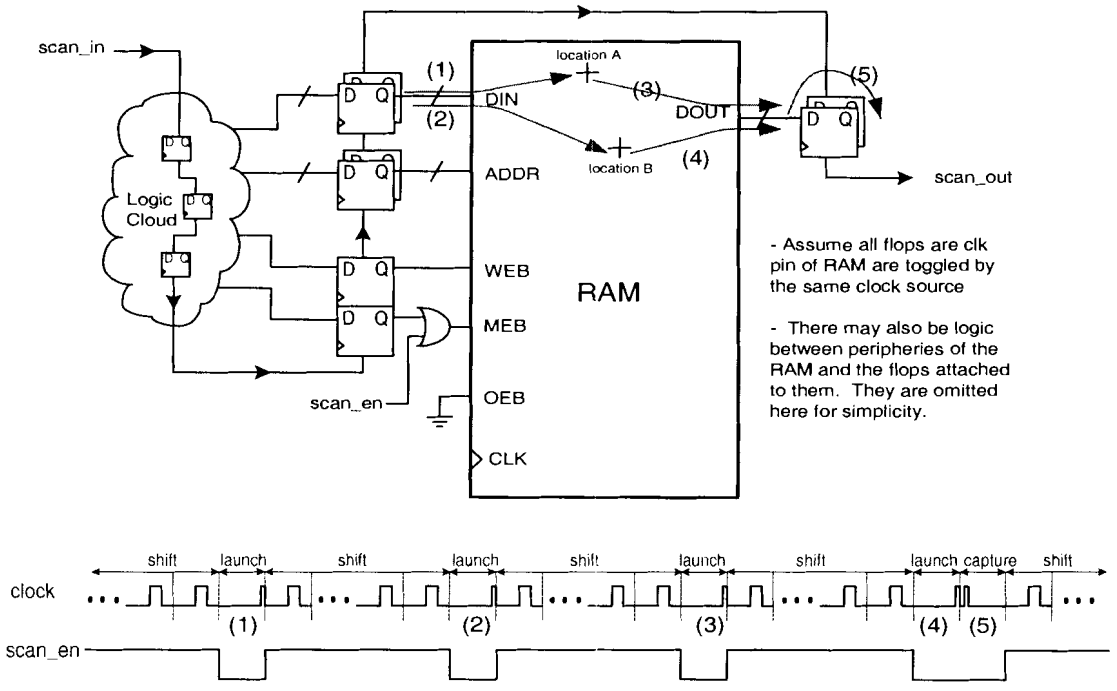
5. As marked in the diagram as (3), scan_en drops to '0' and a clock pulse is issued to read the content of locationA to the DOUT bus of the RAM.

6. As marked in the diagram as (4), another clock pulse is issued to capture the values on DOUT onto the DOUT flops. Then the shift-out operation pipes the content of the DOUT flop to the scan_out port for sampling. This completes the test.

5.2.3 Testing DOUT of RAM

The above diagram depicts the strategy for testing DOUT of RAM by the ATPG tool. This test involves four scan loads, all of which involve read and write access to two locations different RAM locations (marked as locationA and locationB in the figure above). Here is the description in detail:

Figure 15: Testing DOUT of RAM



1. First apply a scan load (shift) that provides the following setup: Q pins of the ADDR flops would select locationA; Q pins of the DIN flops would drive data (call it dataA) into locationA upon a clock pulse; WEB is setup to be '0' to allow for a write operation.

2. As marked in the diagram as (1), scan_en drops to '0' and a clock pulse is issued to write dataA into locationA.

3. Apply another scan load (shift) that provides the following setup: Q pins of the ADDR flops would select locationB; Q pins of the DIN flops would drive data (call it dataB) into locationB upon a clock pulse; WEB is setup to be '0' to allow for a write operation.

4. As marked in the diagram as (2), scan_en drops to '0' and a clock pulse is issued to write dataB into locationB. Most, if not all, of the bits in dataB is complementary to dataA.

5. Apply another scan load (shift) that provides the following setup: Q pins of the ADDR flops would select locationA; WEB is setup to be '1' to allow for a read operation.

6. As marked in the diagram as (3), scan_en drops to '0' and a clock pulse is issued to read content of locationA (dataA) to the DOUT ports of the RAM. This operation presets the value of the DOUT. Since OEB is always enabled and MEB is disabled during shift, the DOUT value will be maintained through the next shift load

7. Apply another scan load (shift) that provides the following setup: Q pins of the ADDR flops would select locationB; WEB is setup to be '1' to allow for a read operation.

8. As marked in the diagram as (4), scan_en drops to '0' and a clock pulse is issued to read the content of locationB (dataB) to the DOUT bus of the RAM.

9. As marked in the diagram as (5), another clock pulse is issued to capture the values on DOUT onto the DOUT flops in at-speed fashion. If there are no delay defects on the DOUT pins, this clock pulse would have captured the content of dataB onto the DOUT flops. Because many, if not all, of the bits in dataA and dataB complement each other, the pulses in (4) and (5) together have launched a set of transitions on the DOUT pins and captured the post-transition value onto the associated flops. Then the shift-out operation pipes the content of the DOUT flop to the scan_out port for sampling. This completes the test.

5.3 RAM ATPG Issue(s)

Currently, due to the ATPG tool's inability to enforce double-pulses for sensitizing all interfaces of RAM—namely, the DOUT interface, the generated patterns may be problematic if a PLL is used to toggle the RAM clock(s) during capture. PLL is currently designed to issue two at-speed pulses in scan mode upon a falling edge of scan_en. This conflicts with the setup requirement of the DOUT test which issues single-pulse waveforms. Therefore, three workarounds are proposed here:

1. If the I/O rate of the scan-mode clock port is fast enough, disable the PLL and use the top-level clock source directly to generate the pulses in capture mode.

2. Since PMC's PLL setup can be done solely through JTAG operations, pattern file can be post-processed to insert JTAG sequences to bypass the PLL for the single-pulse captures. In this case, it is important that none of the JTAG pins (tck, tdi, tdo, tms, trstb) are shared for scan purposes. Sharing JTAG pins for other means is not only a problem for this work-around, but also a direct violation of the JTAG standards.

3. Reuse a top-level functional pin to control the PLL bypass mux in scan mode. This way, the mux can bypass the PLL for the single pulses with a simple pin constraint on the mux-control port.

CHAPTER 6 TEST-CASE RESULTS

As part of the verification for the ATPG automation flows as well as the ongoing IC development within the company, various aspects of the flow have been subjected to thorough testing. This chapter presents some of the results from the transition-delay ATPG flow and the path-delay ATPG flow. Unfortunately, at this time, meaningful data have not been collected on ATPG of RAM interface due to delay in delivery of a generic RAM wrapper that, among other features, contains the gating logic to enable RAM ATPG. Consequently, the flow was only subjected to a simple test-case as a proof of concept. ATPG was successful on that test-case.

Table 1: Test-case results for AC-scan Transition-delay ATPG

Testcases	block 1	block 2	block 3	block 4	block 5
gate count	567908	434716	409087	40201	143000
clock domain count	1	2	2	1	1
No. of scan chains	40	40	32	32	40
longest chain	817	2029	415	141	290
AC pattern count	2011	2037	2556	977	4029
final AC coverage	66.42%	79.68%	80.86%	68.75%	79.74%
DC fault-grade coverage	86.13%	90.66%	94.97%	85.82%	96.18%
DC top-up pattern count	3805	963	644	1228	121
final DC coverage	96.85%	98.08%	98.56%	94.75%	96.58%
total pattern count	5816	3000	3200	2205	4150
total vector count	4751672	6087000	1328000	310905	1203500

Table 1 records pattern and coverage results for five design blocks subjected to the transition-delay ATPG flow. AC pattern count and final AC coverage describe the effectiveness and efficiency of the initial ATPG run in AC mode. The coverage ranges between 66% and 80%, and is inline with the expected coverage number of a testable design. Please also note that, in some cases, patterns were truncated to save tester memory. The DC fault-grade coverage describes the DC-mode coverage of the patterns generated for AC. The DC faults not covered by the AC patterns were then targeted for the DC-mode top-up ATPG to obtain the final DC coverage and the total pattern count. The last row of the table records the vector count. Vector count is approximated by multiplying the pattern count with the length (or numbers of flops) of the longest chain in the design and, as mentioned earlier, directly impacts the availability of ATE memory space. From this relationship, one can understand that keeping the scan chains short is another effective way of lowering vector count. As an example, block 2 in the above table has only half of the pattern count of block 1. However, its vector count exceeds that of block 1 because its chain length is close to three times of the block 1's chain length.

Figure 16 records the incremental coverage increases as AC-scan transition-delay patterns are generated for four of the five blocks mentioned in Table 1. A notable observation here is that some of the curves in the graph show discontinuity during the incremental increases. These discontinuous points marks the limit of achievable coverage for one clock domain and the targeting of another clock domain as the ATPG tool gives up on the first one. For example, on the curve for block 3, the ATPG tool was able to achieve as much as 42% AC coverage by targeting clock domain one. As the number of testable faults depletes in that domain, the tool moved on to target the second domain. Block 2 exhibits the same behaviour at approximately 73%.

Figure 16: AC Transition-delay coverage VS. pattern count

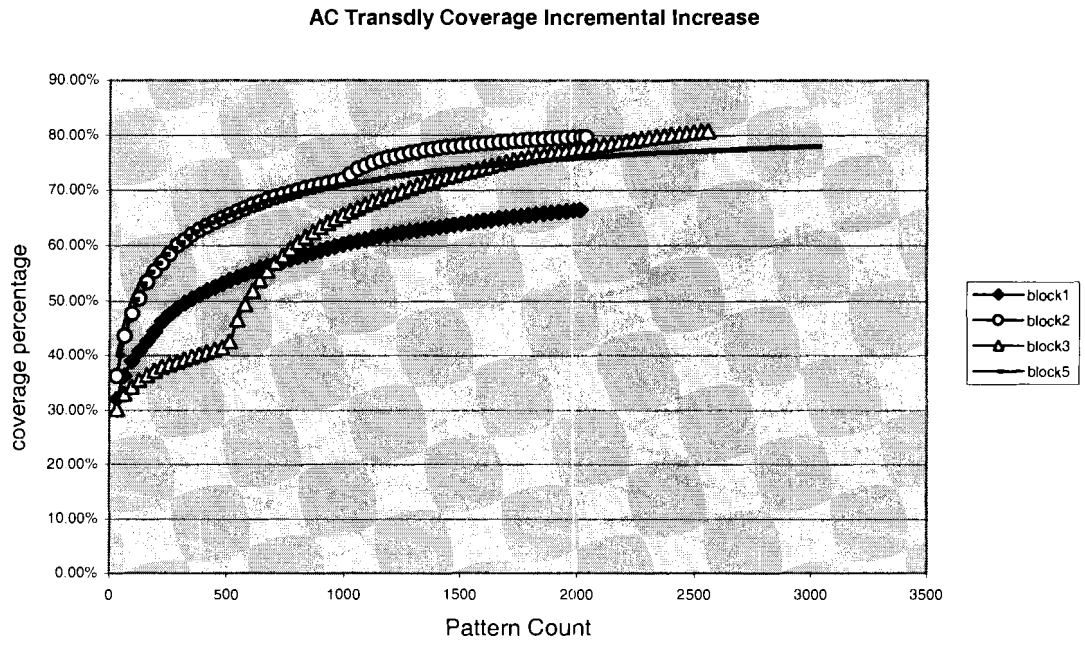


Table 2 shows the test-case results for path-delay ATPG for speed-grading. Coverage is usually between 50% and 65%. Block 3 is considered as an outlier due to the small number of paths targeted for ATPG. Path-delay ATPG for I/O timing characterization has also been done on one testcase and 100% coverage was obtained from that.

Table 2: Test-case results for AC-scan Path-delay ATPG

Testcases	block 1	block 2	block 3
No. of targeted paths	5937	13147	8
Path-delay pattern count	1450	1799	16
Path-delay coverage	64.41%	50.32%	75.00%
Tested paths	3824	6616	6

CHAPTER 7 CONCLUSION

With the growing complexity of today's integrated circuit designs, engineers have abandoned the use of pure functional test vectors wherever possible, and adopted various DFT solutions to make their designs more test-friendly. The most common DFT approach for digital designs is scan insertion, because of its relative simplicity in conceptual understanding and implementation automation. During scan insertion, flip-flops in the design are converted into scan flops and linked into chains of shift registers called scan chains. This way, data input stimuli are shifted into every flop through their respective scan chains. Then, after the stimuli are given the time to propagate through their functional data paths, the flops are triggered to capture the result of the propagation. Finally, the results are marched out of the chains for strobing. The structured nature of scan-testing paves the way to various automated test-pattern generation tools. These tools are capable of automatically generating test vectors on a scan-inserted design of significant gate-count based on a stuck-at fault model, usually achieving 95% or higher test coverage within hours of processing. This, compared to the multiples of man-weeks/months of manual vector creation, creates great savings in engineering and corporate resource.

As transistor size continues to shrink, new defect mechanisms start to appear that can no longer be properly modelled by the stuck-at fault model. These new types of defects adversely affect the speed of the IC's operation. More specifically, a defect may no longer cause a node be permanently "stuck" at a particular logic level when a transition is required, but cause the desired transition to take place at a much slower

speed than needed. This class of new defect mechanisms prompted the creation of the transition-delay fault model. ATPG tools supporting this model create patterns that launch two at-speed clock pulses in between the shift-in/out operations. The two pulses are accompanied with test vectors that launch and capture data transitions on the targeted node, hence testing the node in at-speed fashion. As part of this project, an automated ATPG flow is created to generate test vectors for at-speed defect detection. The flow involves finding the functional clock domains and their respective clock speed, identifying the scan-mode clock sources for each functional domain, performing ATPG on each domain in AC-scan mode to obtain transition-delay test coverage, fault-grading the AC vectors in DC mode, and finally creating DC-mode top-up vectors. This test approach provides good AC and DC test coverage with the least amount of vectors possible.

Along with transition-delay fault model, another model called path-delay fault model is also created. This fault model models an entire data path with two faults: A slow-to-rise fault that states a 0-to-1 transition on the source flop's output pin must invoke a transition on the input of the destination flop within a clock period; A slow-to-fall fault that states a 1-to-0 transition on the source flop's output pin must invoke a transition on the input of the destination flop within a clock period. This model is ideal for speed-grading/binning and can also be applied quite effortlessly to I/O characterization on a scan-inserted design. This is because the model focuses on the transition of data-paths rather than a particular node of a particular gate within a data-path. The basic idea of speed-grading/binning is to sensitize the critical paths of the design at the highest speed possible; similarly, I/O characterization is done through sensitisation of timing-critical data-paths connected to the design's primary I/O ports. Therefore, both speed-grading/binning and I/O characterization can be done by feeding the ATPG tool with the

paths of interest. An automated path-delay ATPG flow is created based on this approach.

For designs containing large RAMs, data paths responsible for RAM access sometimes becomes the most difficult to meet timing. Therefore, it is important to guarantee that boundaries of RAMs are defect-clean so that the data-paths interfacing to their pins can satisfy timing. Due to the relatively complex nature of RAM access compared to operating flops, ATPG tools make use of a different algorithm called multi-load ATPG to test RAM interfaces. A flow has been developed to test RAM interfaces at-speed based on the multi-load algorithm. This flow is designed to target DIN, DOUT, and ADDR pins of RAMs.

With the AC-scan pattern generation flow fully implemented, scan-inserted devices can be tested in at-speed fashion. It is recommended that chips be subjected to the transition-delay patterns, along with the DC top-up patterns, first to filter out all defective parts before running the path-delay patterns.

BIBLIOGRAPHY

[1] Kenneth D. Wagner. Robust Scan-based logic test in VDSM technologies. *Computer*, Volume 32, Issue 11, Nov. 1999, pp. 66 – 74

[2] A. Kobyashi, S. Matsue, H. Shiba. Flip-Flop Circuits with Fault Location Test Capability. *Proc. IECEJ national Convetional*, 1968, p. 962

[3] Jan M. Rabaey. Digital Intergrated Circuits. 1996, Section 11.6, p. 685

[4] Alfred L. Crouch, John C. Potter, Jason Doege. AC Scan Path Selection for Physical Debugging. *Design & Test of Computers*, IEEE, Volume 20, Issue 5, Sept.-Oct. 2003, pp. 34 – 40

[5] G. Aldrich and B. Cory. Improving Test Quality and Reducing Escapes. *Proc. Fabless Forum, Fabless Semiconductor Assoc.*, 2003, pp. 34-35.

[6] Mukunk Sivaraman, Andrzej J. Strojwas. Path Delay Fault Diagnosis and Coverage—A Metric and an Estimation Technique. *Computer-Aided Design of Integrated Circuits and Systems*, IEEE Transactions on, Volume 20, Issue 3, March 2001, pp. 440 – 457

[7] Xijiang Lin, Ron Press, Janusz Rajski, Paul Reuter, Thomas Rinderknecht, Bruce Swanson, Nagesh Tamarapalli. High-Frequency, At-Speed Scan Testing. *Design & Test of Computers*, IEEE, Volume 20, Issue 5, Sept.-Oct. 2003, Pp. 17 – 25

[8] A. Carbine, D. Feltham. Pentium Pro Processor Design for Test and Debug. *Proc. IEEE International Test Conference, IEEE Computer Society Press*, Los Alamitos, Calif., 1997, pp. 294-303.

[9] Kenneth B. Butler. Estimating Economic Benefits of DFT. *Design & Test of Computers*, IEEE, Volume 16, Issue 1, Jan.-March 1999, pp. 71 – 79

[10] S. Pateras. Achieving At-Speed Structural Test. *Design & Test of Computers*, IEEE, Volume 20, Issue 5, Sept.-Oct. 2003, pp. 26 – 33

[11] R. Tekumalla, P. Menon. On Redundant Path Delay Faults in Synchronous Sequential Circuits. *Computers*, IEEE Transactions on, Volume 49, Issue 3, March 2000, pp. 277 – 282

[12] A. Keshavarzi, J. Tschanz, S. Narendra, V. De, W. Daasch, K. Roy, M. Sachdev, C. Hawkins. Leakage and process variation effects in current testing on future CMOS circuits. *Design & Test of Computers*, IEEE, Volume 19, Issue 5, Sept.-Oct. 2002, Pp. 36 – 43

[13] B. Garlepp, K. Donnelly, Jun Kim, P. Chau, J. Zerbe, C. Huang, C. Tran, C. Portmann, D. Stark, Yiu-Fai Chan, T. Lee, M. Horowitz. A Portable High-Speed DLL for CMOS Interface Circuits. *Solid-State Circuits*, IEEE, Volume 34, Issue 5, May 1999, Pp. 632 – 644

[14] Bruce D. Cory, Rohit Kapur, Bill Underwoods. Speed Binning with Path Delay Test in 150-nm Technology. *Design & Test of Computers*, IEEE, Volume 20, Issue 5, Sept.-Oct. 2003, pp. 41 – 45

[15] Mentor Graphics Inc. *Design-for-Test Datasheet: ATPG with Embedded Compression*. Retrieved December 22, 2004, from http://www.mentor.com/products/dft/atpg_compression/testkompress/loader.cfm?url=/commonspot/security/getfile.cfm&pageid=15343