Boundary Scan Verification

A Major Report

Submitted in partial fulfillment of the requirements
for the degree of

MASTER OF TECHNOLOGY

IN

VLSI DESIGN

By

Krishna Patel
18MECV12

Electronics & Communication Engineering Department
Institute of Technology
Nirma University
Ahmedabad - 382 481
May 2020

Boundary Scan Verification

A Major Project

Submitted in partial fulfillment of the requirements
for the degree of

MASTER OF TECHNOLOGY

IN

(VLSI DESIGN)

By

Krishna Patel

18MECV12

Internal Guide:	**External Guide:**
Dr. Vaishali Dhare	Mrs. Indira Gohad
Assistant Professor,	Engineering Manager,
Institute of Technology	Intel Technology India Pvt Ltd.
Nirma University	

Electronics & Communication Engineering Department

Institute of Technology

Nirma University

Ahmedabad - 382 481

May 2020

Declaration

This is to certify that

1. The thesis comprises my original work towards the degree of Master of Technology in VLSI Design at Nirma University and has not been submitted elsewhere for a degree.

2. Due acknowledgment has been made in the text to all other material used.

Krishna Patel
18MECV12

NIRMA UNIVERSITY

INSTITUTE OF TECHNOLOGY

NAAC ACCREDITED 'A' GRADE

Certificate

This is to certify that the project entitled **"Boundary Scan Verification"** submitted by **Krishna Patel (18MECV12)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in VLSI Design, Nirma University, Ahmedabad is the record of work carried out by her under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination.

Dr. Vaishali Dhare
Internal Guide,
Institute of Technology,
Nirma University, Ahmedabad

Dr. Usha Mehta
PG Coordinator - VLSI Design,
Institute of Technology,
Nirma University, Ahmedabad

Dr. Dhaval Pujara
Professor and Head,
Institute of Technology,
Nirma University, Ahmedabad

Dr. Alka Mahajan
Director,
Institute of Technology,
Nirma University, Ahmedabad

Date :

Place : Ahmedabad

Certificate

This is to certify that the project entitled **"Boundary Scan Verification"** submitted by **Krishna Patel (18MECV12)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in VLSI Design, Nirma University, Ahmedabad is the record of work carried out by her under our supervision and guidance at **Intel Technology India Pvt. Ltd**. In my opinion, the submitted work has reached a level required for being accepted for examination.

External Guide
Indira Gohad
Engineering Manager
Intel Technology India
Bangalore
Date : Place : Bangalore

Acknowledgment

The journey of my learning new things and the incremental success of my project required a lot of guidance and motivation from many people. Whatever I have done until now is the result of their supervision, and I would like to extend my words of gratitude to all these people.
I respect and thank Shivaprashant Bulusu (Hiring Manager) for giving me an internship in his team. I am thankful to him to give me this amazing opportunity and work in a pure DFT team.
I would like to thanks Indira Gohad (Manager) for providing me her valuable time, providing me with all necessary inputs, feedback for my work, and timely support during my project and in my thesis writing.
Besides my team, I would like to thank the rest of my thesis committee: First of all with all due respect I am deeply grateful to Dr. N.M. Devashrayee for his good advice, encouragement and constant support during my master's journey, not only for all learning he gave but also for its application in real life. I would extend my thanks to Dr. Usha Mehta for her guidance, inspiring words and constructive feedback during the journey. I would like to thank Dr. Vaishali Dhare for her suggestions, guidance, and motivation during my project. Lastly, I would like to thank all faculty members of Nirma University who were being so helpful during my master's study.
Finally, I give my forever grateful to my parents for their unconditional love and support. I would like to thank them for their encouragement in my higher studies, believing me, and never lose hope on me.

Krishna Patel
18MECV12

Abstract

This project aims for the verification of boundary scan architecture in any design for detecting bugs at an earlier stage while maximizing product quality. As the technology node reduces, the device becomes faster and the complexity of the chip increases. The reduction in feature size increases the probability of manufacturing defects in an Integrated Circuit. Thus, it is significant to test the IC once they are taped out. To address the testing challenges for the physical defects, DFT (Design for test) architecture has become popular in the industry. Boundary-scan is one of the DFT embedded in the chip to address the testing challenge between interconnects of the chip.

Functional verification of the implemented boundary scan in the design is essential to make sure that the design works the way it has been intended to perform. Many tools aid in the design verification including simulation tools and test content generation automation tools dedicated to boundary-scan. The test stimuli and testbench are generated using the Mentor Graphics Tessent tool. The test patterns are completely derived from the BSDL (Boundary-scan description language) file which provides a complete description of the boundary scan implemented in the device, no other design data is used in doing so. The test patterns are validated by integrating into the OVM (Open Verification Methodology) testbench environment dedicated to the validation of the TAP (Test Access Port) architecture. The test case covers all possible test scenarios needed to test the different instructions of boundary-scan. This validates the boundary-scan architecture and BSDL file at the IP (Intellectual Property) level.

The methodology is used to make all BSDL and boundary-scan validated individually executed at the IP level so that it provides consistent verification of boundary-scan implemented at SOC (System on Chip) level. The testbench using OVM increased the reusability and can be used at any level of the hierarchy.

Table of Contents

List of Tables

List of Figures

List of Abbreviation

ATPG	- Automatic Test Pattern Generation
BSDL	- Boundary Scan Description Language
CPU	- Central Processing Unit
DFT	- Design for Test
DR	- Data Register
DSP	- Digital Signal processor
DUT	- Design Under Test
FPGA	- Field Programmable Gate Array
IC	- Integrated Circuit
IEEE	- Institute of Electrical and Electronics Engineers
IP	- Intellectual Property
IR	- Instruction Register
JEDEC	- Joint Electron Device Engineering Council
JETAG	- Joint European Test Action Group
JTAG	- Joint Test Action Group
PCB	- Printed Circuit Board
SMT	- Surface Mounted Technology
PLL	- Phase Locked Loop
SVF	- Serial Vector Format
SOC	- System On Chip
STIL	- Standard Test Interface Language
TAP	- Test Access Port
TCK	- Test Clock
TDI	- Test Data Input
TDO	- Test Data Output
TMS	- Test Mode Select
TRST	- Test Reset
WGL	- Waveform Generation Language

Chapter 1

Introduction

The observation made by Gordon Moore, co-founder of Intel in 1965 known as Moore's Law was the number of transistors in a dense integrated circuit will be doubled every year since the integrated circuit was invented. Moore predicted that this trend would remain for the foreseeable future. And it is a proven law to date. A simple example of this law is progressing from SSI to VLSI.

Advancements in Semiconductor technology have lead Integrated Circuits towards higher speeds, smaller physical dimensions, and increasingly complex design functions integrated into a single device. The advancement in technology is highly motivated by the constant need in real time applications, performance enhancements, and making this world a better place to live. Nowadays, the demand for the internet of things, autonomous vehicles, machine learning, artificial intelligence, and internet traffic is growing exponentially, which acts as a driving force for scaling down transistors below the existing 7nm node for higher performance. There are several challenges with the reduction in technology node. Along with this comes the challenge to test this dense Integrated Circuits.

The reduction in feature size increases the probability of the manufacturing defect in an IC which will result in a faulty chip. A very small defect can result in a faulty transistor or interconnect. The presence of a small defect in a transistor or interconnect can result in a failure of a complete IC. These defects during the manufacturing process are unavoidable, so plenty of IC is suspected to be faulty. So, testing is needed to guarantee fault free products. It is also essential to test a component at various stages like chip level, board level, system level. A general rule of 10 states that the cost of detecting faulty IC increases as we move from every stage i.e from chip level to board level to system level.

Testing involves IC testing, PCB testing, and system testing and during operation of the system also. Testing includes finding faulty devices as well as increasing manufacturing yield. VLSI testing is important to designers, product engineers, test engineers, manufacturers, and end-users. Testing consists of applying the test stimulus to the inputs of the DUT (Design Under Test) and analyzing the output responses. Boundary Scan inserted in logic provides a scan path through all Input/Output pins of IC to assist in testing the interconnections of PCB. Figure 1.0.1 shows the basic approach of testing.

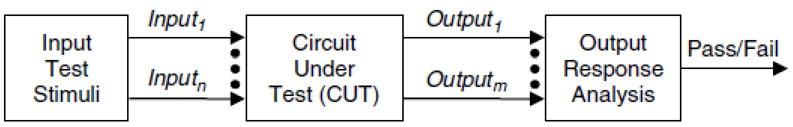


Figure 1.0.1: Basic Approach of testing

1.1 Motivation

- Finer packaging geometries and switch to new surface mounting technology made bed-of-nails test fixtures incapable of accessing nodes for testing. Boundary Scan Architecture solves the testing problem of interconnections between various device at chip level, board level or system level.
- Boundary Scan is found in almost every design and used to test the interconnects on the Printed Circuit Board. This addition of architecture in the design needs to be verified as the cost of finding bug increases after every stage of chip development.
- The motivation of this project is to understand thoroughly IEEE Standard 1149.1 Test Access Port and Boundary-Scan Architecture, IEEE Standard 1149.6 Boundary-Scan Testing of Advanced Digital Networks.

1.2 Objective

- The core objective of this project is to verify architecture of boundary scan implemented in any chip at the IP level.
- The project will involve verification of Boundary Scan Architecture using OVM based test bench environment with DUT specific initialization sequences.
- Learning of tool features used for the verification of Boundary Scan.
- This also includes the generation of test patterns used to test the Boundary Scan Architecture integrated at the IP level.

1.3 Overview of Thesis

The thesis comprises of literature survey for the boundary scan. Chapter 3 contains the description of the boundary scan architecture, various instructions, additional instructions supporting boundary scan for advance networks. Chapter 4 describes the implementation of the methodology used for the purpose of boundary scan verification. Chapter 5 contains results and discussion from the RTL simulations for various test. Lastly, Chapter 6 contains conclusion drawn from the project.

The thesis comprises of an in-depth description of Boundary Scan Architecture and explaining TAP Controller and its state machine. It also contains the explanation of individual tests which includes all instructions described by the 1149.1[1] and 1149.6 standard[2].

Chapter 2

Literature Survey

2.1 Digital Test before Boundary Scan

In-circuit testing was used to access internal nodes of an IC (Integrated Circuit) to check for joints, shorts, open to check whether the circuit is fabricated correctly. In-circuit testing is an example of white box testing. It is performed with bed of nails type test fixture. It was showing a reduction in effectiveness due to the loss of test points on products. In classical in-circuit test stimulus and responses are received from the board and the test through probes on a fixture connecting the tester to the board, test in this way the in-circuit test can measure discrete devices and test IC's. Challenges faced by In-Circuit testing are:
The IC packaging technology was dominated by dual-inline, through hole mounted packages. So every board signal was visible on the bottom of the board. It was easy to arrange In-Circuit nails to target the pins of the IC themselves. By the advent of SMT (Surface Mounted Technology) devices and finer packaging technologies, access of all nodes to the In-Circuit nails became difficult. An alternative was needed for the testing of devices. Boundary Scan contributes to solving this problem.

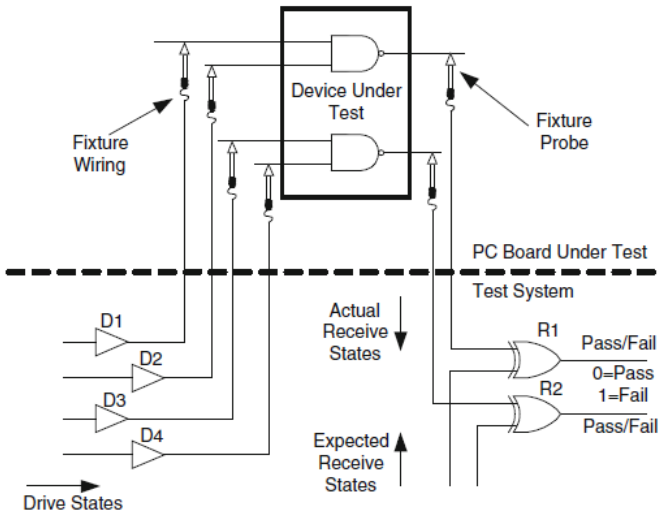


Figure 2.1.1: In Circuit test with access of full nodes

2.2 Introduction to Boundary Scan

Testing a standalone device is comparatively easy because all Input and Output pins are controllable and observable with the test equipment. Once the device is assembled on the Printed circuit board, the complexity of testing increases. The conventional in-circuit testing using the "bed-of-nails" testing method in which testing relies on probing test pins onboard and vias has already encountered problems in dealing with multiple layer boards. With the advent of surface mounting packages and multiple chip modules, this method becomes infeasible as no through hole pins are available for probing.

In the mid-1980s, a group of test engineers from various European electronics system companies worked together to seek possible solutions to the problem. This group, known as the JETAG (Joint European Test Action Group), attained a solution to approach this problem. They concluded to chain all the boundary input and output pins of the device into a shift register and used a concept similar to scan design to make input-output controllable and observable. Later in 1988, the group was accompanied by other more representatives from North American companies they had also been working on this problem and came to a similar conclusion. The merged group was renamed as JTAG (Joint Test Action Group). Through the works of the group, the concept of "boundary scan" was formally converted into a test architecture and a set of associated design rules were approved by the IEEE as a test standard (Std. 1149.1) in 1990. Since then, the standard has been applied by most electronics companies designing large Integrated Circuit. Today, almost all general-purpose CPU, DSP, FPGA, and various ap-

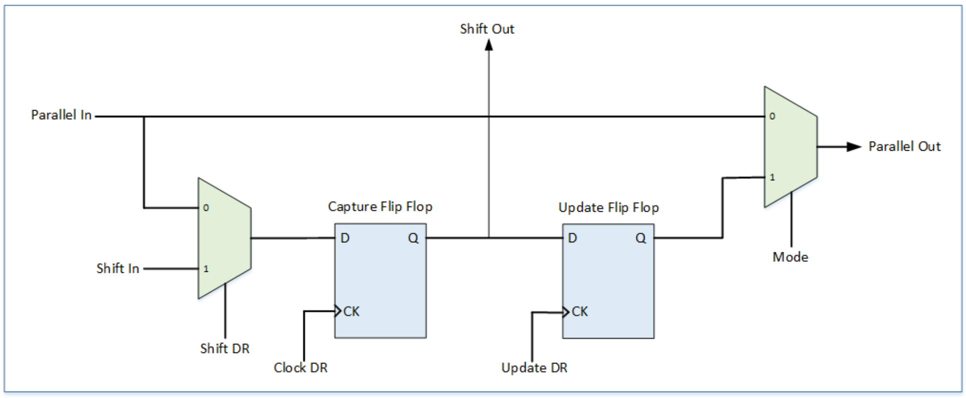


Figure 2.2.1: Basic boundary scan register cell

plication specific designs comply with the 1149.1 standard and the family of 1149. Because boundary scan provides a simple and efficient protocol for data communication, this standard has also been employed in many other applications, including power management, embedded instrumentation control, clock/PLL control, debugging/diagnosis, verification, and chip reconfiguration.

The Boundary Scan technique involves a scan based design forming a shift register like structure for every Input Output pads present in a chip. Each component pin present at component boundaries are stitched in serial register format is controllable and observable using scan testing fundamentals.

Figure 2.2.1 shows an example of a boundary-scan cell that can be applied for input/output pins of the IC. Depending on the control signal input applied to the multiplexers, data can be either driven from the register through the Parallel out port of the cell (into the core of the component design) or loaded into the scan register from the Parallel in port.(e.g., the input pin).

This boundary scan cells are connected at each input output pin of the IC and form a shift register structure around the boundary of the design. This path is given with appropriate control signals and a clock. Within a board containing several IC, the boundary scan registers from the single IC's can be connected to form a serial single path with other IC's. Figure 2.2.2 shows the complete path of Boundary Scan implemented at the board level.

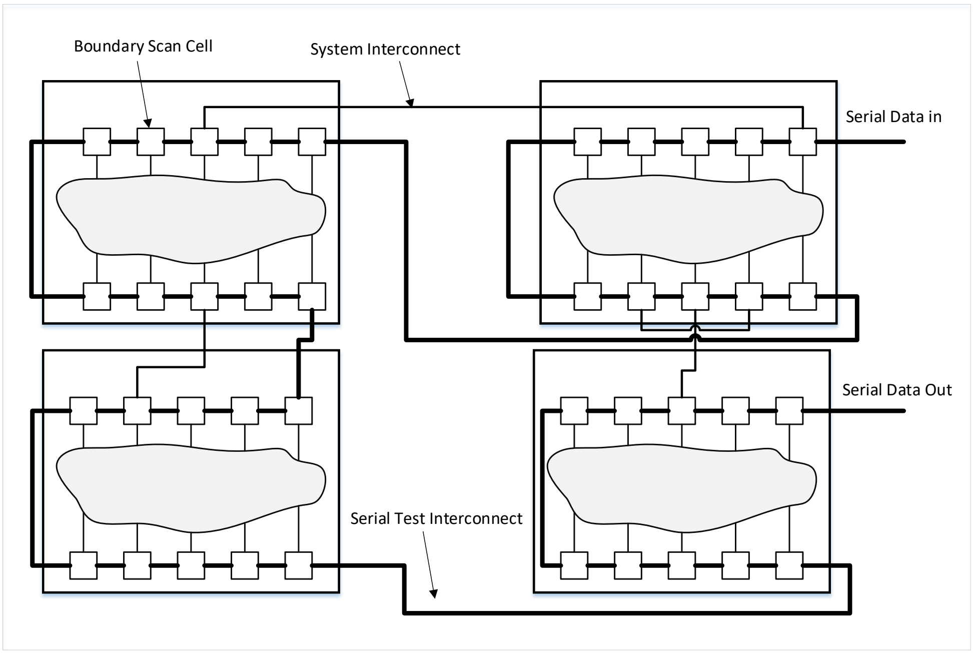


Figure 2.2.2: Boundary scannable board Design

2.3 Evolution of Boundary Scan

Standard 1149.1[1] explains a general-purpose boundary-scan implementation for digital chips. Several different boundary scan standards are serving different objectives. Boundary scan comprises of a family of test methodologies providing a comprehensive range of test problems. This includes chip level to system level, from system logic core to interconnections, from the digital circuitry to analog circuitry to mixed-signal circuitry, and of ordinary digital designs to designs compatible with very high speeds capability. Table 2.1 shows the family of standards identified as IEEE 1149.x standards supporting different applications. Standard 1149.1[1]is normally referred as boundary scan standard was approved by IEEE in 1990. A complete set of a language known as BSDL (Boundary Scan Description Language) was motivated for standardization and increasing demand for the boundary scan. Thus, BSDL also became a part of the standard after the first revision of the boundary scan standard (1149.1a). The revision of 1149.1[1] came in 2001 merging both standards.

Number	Objectives	Status
1149.1	Testing of digital chips and interconnects between chips	Std. 1149.1-2001
1149.2	Extended digital serial interface	Discontinued
1149.3	Direct access testability interface	Discontinued
1149.4	Mixed-signal test bus	Std. 1149.4-1999
1149.5	Standard module test and maintenance (MTM) bus	Std. 1149.5-1995
1149.6	High-speed network interface protocol	Std. 1149.6-2003

Table 2.1: 1149 family of Boundary Scan

The 1149.2 extended digital serial subset discontinued did primarily aimed at ASICs and with the high speed test capability due to the argument that some of the features could be covered in scan design. The 1149.3 Direct access testability interface was aimed at direct access of interfaces of the chip, as it overlapped with the 1149.5, it was also discontinued.
Standard 1149.4 aims boundary scan testing for the chip containing Analog I/O. It defines architecture compatible with analog input and output is introduced to as an analog boundary scan. Standard 1149.6[2] is an extension of 1149.1[1] for standardization of boundary Scan supporting high speed I/O chips. This includes networks that are single-ended, differential, AC coupled. These both standard are gaining more popularity than any other. The thesis is based on these two standards only.

2.4 Test Considerations of Core Based Design

In the era of SOC, standard gate level or cell-based design are not sufficient. The primary building blocks of the system are formed by core based design methodology. As in Figure 2.4.1 shown a typical SOC design which contains DSP, CPU, ALU, memory modules like RAM and ROM, some glue logic, ASIC cores, IPs, interconnections modules. Testing problems associated with this complex design increases as the complexity of the design increases.
A SOC integrator has to consider how to develop a test for cores which can or cannot be provided by the same vendor, can be in different form i.e hard or soft cores. Many verification problems are found in such a complex system that is not caught at the IP or sub-block level. Automation tools can be the greatest help in generating tests and circuitry required to test the system. Individual IPs can be verified and their corresponding BSDL file can be made error-

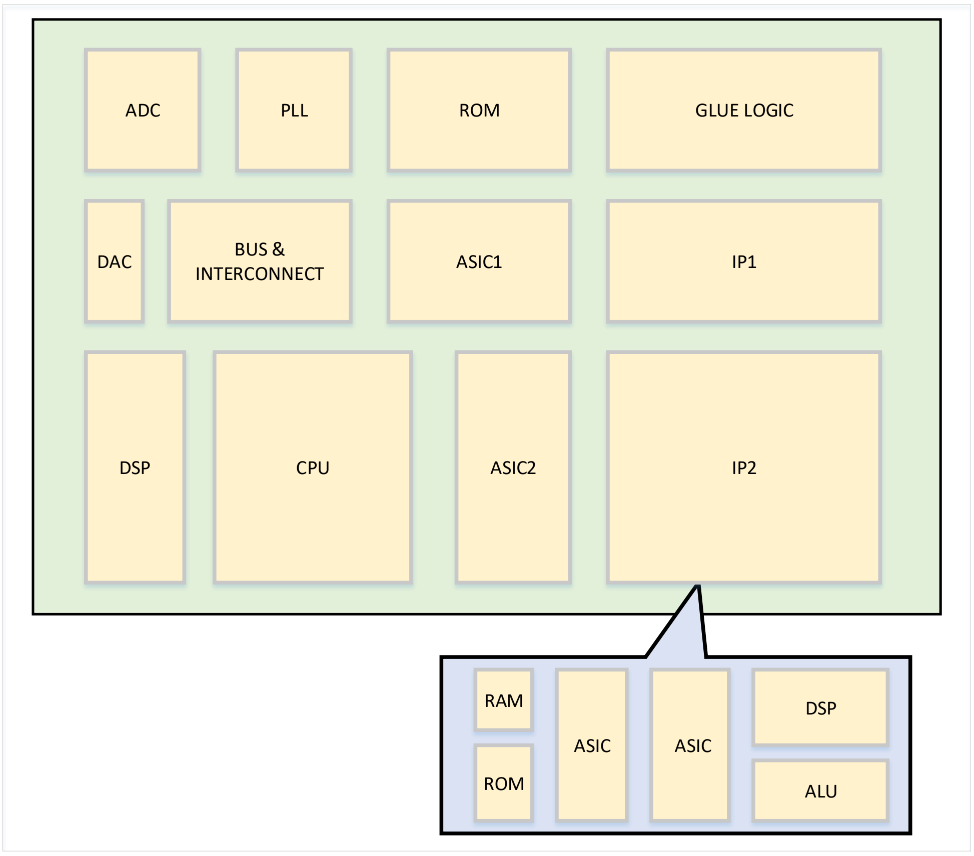


Figure 2.4.1: Core based SOC Design

free. So at the integration level at SOC, the integrator will not have to face unnecessary debug in verifying bugs found in IP. Test content generated from the automation tool can aid in verifying IP and their BSDL. So the top-level BSDL created from the individual IPs will be error-free.

Chapter 3

Boundary Scan

3.1 What is Boundary Scan

Boundary Scan is a testing standard. It is a standard set of the design rules which are implemented primarily at the IC. It also impacts on various product life cycle development as:

- At the Integrated Circuit level, it supports (BIST) Built-in Self Test.
- At the Board level, it supports the testing of interconnections of different IC integrated at PCB.
- At the system level, it supports testing of higher-level integration from modules to systems.

3.2 Boundary Scan Architecture

The basic architecture implemented at Integrated Circuit level is shown in the figure 3.2.1. The Boundary Scan is divided into four main hardware elements:

- A **TAP** (Test Access Port), consists of four mandatory pins: **TDI** (Test Data In), **TDO** Test Data Out, **TMS** (Test Mode Select), and **TCK** (Test Clock) and one optional pin, **TRST*** (Test Reset).
- A **TAP Controller**
- An **IR** (Instruction Register) and Instruction Decoder

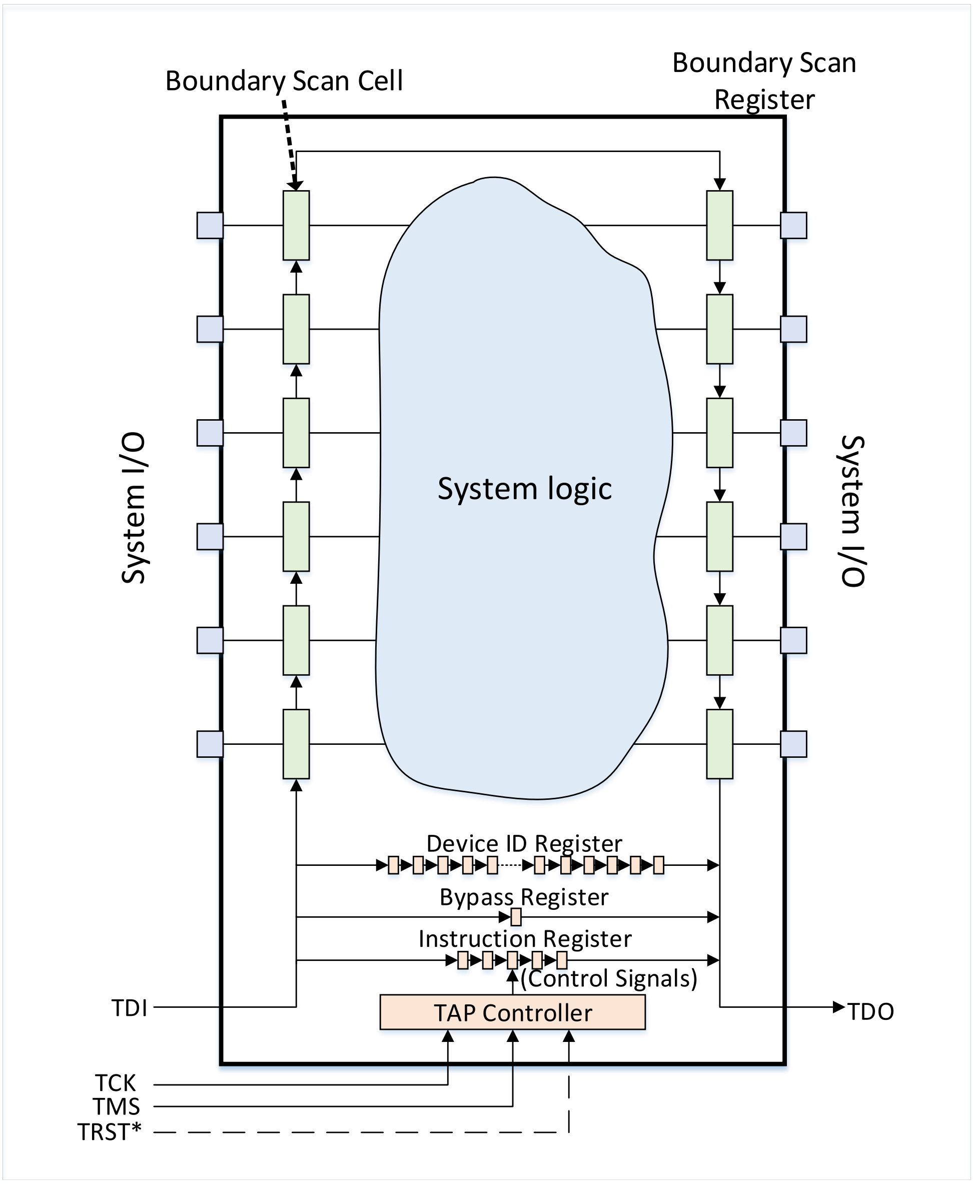


Figure 3.2.1: Simplified architecture of boundary scan

- Other TDR (Test Data Registers). Mandatory are boundary scan register, bypass register, other optional can be device ID register, user defined registers.

The TAP is a general purpose port available that provide access to boundary scan functions into a component. Dedicated pins are used for the boundary scan which is not used by any functional logic. The TAP controller is a 16-state FSM upon which boundary scan transactions rely on. Each instruction to be carried out by the boundary scan architecture must be serially loaded

into the IR through the TDI (Test Data Input) pin. The instruction decoder decodes based on instruction loaded in the Instruction register. The test data registers store system related operation or some test data.
A set of test instructions is provided like BYPASS, SAMPLE, PRELOAD, and EXTEST which are mandatory ones and optional ones include HIGHZ, CLAMP, INTEST, IDCODE, USERCODE, and RUNBIST.

3.2.1 Test Access Port

The boundary scan register and other test data registers are accessed through a standard interface, JTAG TAP interface. The standard implies presence of mandatory four test pins: TDI, TDO, TMS, TCK and one optional pin TRST* each available through a dedicated device pin which maynot be shared with any other function.

3.2.1.1 Test clock input

The TCK(Test Clock) is the clock input for all operations carried by boundary scan. The dedicated TCK is independent of the functional clock so that shifting and capturing of test data from one chip to another onboard can be carried out properly. Test signals respond on both the rising edge or falling edge of the TCK.

3.2.1.2 Test data input

Test data input (TDI) allows test data and instruction bits to be loaded serially in the various test data registers. Values on the TDI are shifted on the rising edge of TCK. When undriven TDI should be connected to a logic one.

3.2.1.3 Test mode select

Test mode select (TMS) is the control input of the TAP Controller. All boundary scan test operations will be controlled by TMS. Capturing, shifting and updating operations will occur only based on TMS. TMS value is sampled at a rising edge of TCK. When undriven TMS should be connected to a logic one.

3.2.1.4 TDO

Test data output (TDO) is then shifted out data from the test data register or instruction register. Changes in the value of TDO occurs only at the falling edge of TCK. TDO value should be in its inactive state other than the shifting operations.

3.2.1.5 TRST*

Test Reset (TRST*) is the optional pin of the hardware. It provides a hard reset to the boundary scan circuitry. TAP controller can asynchronous reset to the test logic reset state when a zero is asserted. This pin is independent of a functional reset.

3.2.2 TAP Controller

TAP controller FSM state diagram containing 16 states is shown in figure 3.2.2. The TAP Controller is a 16 state finite state machine that uses state diagram as shown in figure 3.2.2. TAP controller can change its state only on the positive edge of TCK. The TAP controller changes its state based on the value of TMS. TMS should be set to 0 or 1 before the rising edge of TCK. TMS is sampled only at the rising edge of TCK. TAP controller will be asynchronously reset to the test logic reset state on the assertion of TRST* known as a hard reset. TAP controller also can synchronously reset to the test logic reset state on the application of five consecutive 1's on TMS. The TAP controller will be in the reset state regardless of its current state in the state diagram. This reset is also known as a soft reset.[3]
The 16 states are divided into three parts. The first part contains the test logic reset and the RunTest/Idle state. The other two parts are two vertical columns in the state diagram containing 7 states each. They both columns are identical except for the state targeting the registers. The vertical column in the left targets states for the data register and the vertical column in the right targets states for instruction register. The behavior of both is the same.
The main function of the TAP controller includes some of the function listed below:

- Reset boundary scan test architecture.
- Provide control signals to load instruction in the IR.

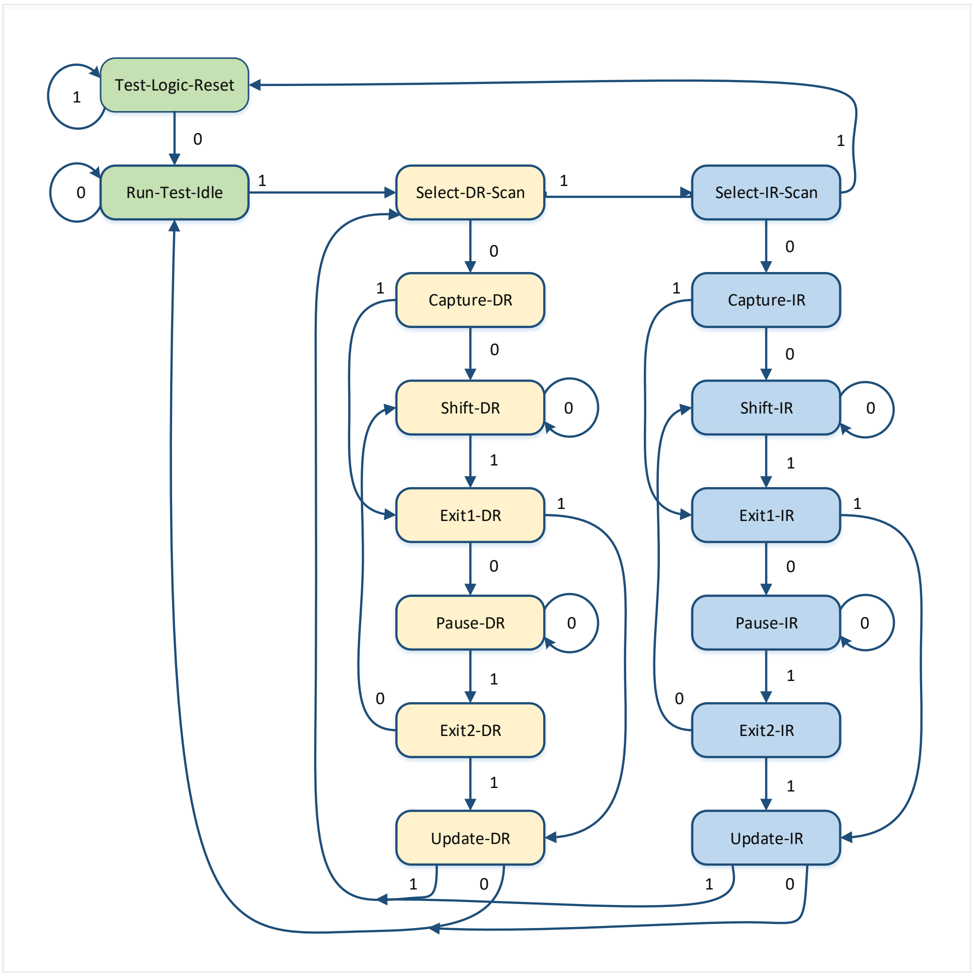


Figure 3.2.2: State diagram of TAP Controller

- Provide control signals to perform test operations of capturing and updating test data.
- Provide control signals to perform shift operations between TDI and TDO.

3.2.2.1 Test Logic Reset

In this TAP controller state test logic is in its reset state and the normal operation of IC can happen without any interrupt. The instruction register is decoded with the IDCODE instruction if the design has a device identification register available or the BYPASS instruction is decoded as the default instruction. When a low value is applied to the TRST*, the TAP controller resets and remains in this state. TAP controller can synchronously reset by applying TMS to high value for consecutive 5 TCK cycles.

3.2.2.2 RunTest/Idle

The controller enters this state when TMS is kept as high it remains in this state until TMS is held low. Activities in specifies logic occurs for certain instructions are present. While executing RUNBIST which causes self test, the controller remains in this state. For other instructions it is in the idle mode, test data registers retain their previous value.

3.2.2.3 Select DR Scan

A temporary controller state in which TAP remains for one TCK. Here controller decides to enter the data register column or instruction register column based on the TMS value given. If TMS value is changed to low the controller transits to Capture-DR state and if it is held high controller transits to Select-IR-Scan state.

3.2.2.4 Select IR Scan

A temporary controller state in which TAP remains for one TCK. If TMS is kept low the controller moves to Capture-IR state and if it is held high controller moves to the test logic reset state.

3.2.2.5 Capture IR

This is a state where parallel loading of values occurs in the Instruction Register on the rising edge of TCK. The two LSBs are hard coded to value “01”. Any higher order bits value may receive design specific values. The controller enters Exit1-IR if TMS is changed to high and Shift-IR if TMS is changed to low.

3.2.2.6 Shift IR

The Instruction Register is connected between TDI and TDO in this controller state. The instruction bits are shifted in the Instruction register from TDI on the rising edge of TCK. The controller remains in this state until all instruction bits are shifted so TMS is held low during shifting. The controller transits into Exit1-IR state when TMS is held high.

3.2.2.7 Exit1 IR

The controller state is Pause IR or Update IR based on the value of TMS. If TMS value is kept high controller moves to Update IR state if it is changed to low controller, transits to Pause IR state.

3.2.2.8 Pause IR

This state allows the boundary scan to pause its function and wait for any external operations to complete. For long data, sequence tester might have load and reload data. This state allows waiting for more data to shift in.

3.2.2.9 Exit2 IR

The controller is in Shift IR or Update IR state based on the value of TMS. If TMS is changed to high value, controller moves to Update IR state if it is held low controller moves to Shift IR state.

3.2.2.10 Update IR

The instruction shifted in the previous instruction is latched parallelly in the parallel hold register, on the falling edge of TCK. The new instruction latched becomes the current instruction, overriding the previous instruction loaded in the Instruction Register. If TMS is held high controller transits to Select-DR-Scan state if it is kept low controller transits to RunTest/Idle state.

3.2.2.11 Capture DR

This states loads the parallel data into the data register on the rising edge of TCK. If TMS is held to high controller transits to Exit-1-DR state and if it is held low controller moves to Shift-DR state.

3.2.2.12 Shift DR

This controller state places the decoded data register between TDI and TDO. It shifts one-bit data on each rising edge of TCK toward TDO. Simultaneously it shifts data from TDI. If TMS is high, the controller will enter Exit-1DR state and remains in Shift-DR state only when TMS is low. The controller will remain in this state until the shifting of all data is done.

3.2.2.13 Exit1 DR

This is a temporary controller state. The controller is in Pause-DR or Update-DR state based on the value of TMS. If TMS is held high controller moves to Update-DR state if it is held low controller moves to Pause-DR state.

3.2.2.14 Pause DR

This state allows boundary scan to pause its function and wait for some external operations to complete. For long data, sequence tester might have load and reload data. This state allows waiting for more data to shift in.

3.2.2.15 Exit2 DR

This is a temporary controller state. The controller is in Shift-DR or Update-DR state based on the value of TMS. If TMS is held high controller transits to Update-DR state and if it is held low controller transits to the Shift-DR state.

3.2.2.16 Update DR

In this controller state, data hold in the parallel flop of boundary scan register is loaded on the falling edge of TCK. The data hold in the parallel output changes its value on this state only. If TMS is high controller moves to Select-DR-Scan state if it is low controller moves to Run-test/Idle state.

3.3 Data Registers

All boundary scan instructions places the decoded data register between TDI and TDO. There is only one register selected at a time which is placed between TDI and TDO. Some of the registers like Boundary Scan Register, Bypass Register are mandatory to be present in the boundary scan architecture. Other data registers like Identification Register, User defined Register, and other test data registers are optional in the architecture.

3.3.1 Bypass Register

Bypass Register is a mandatory register with only one cell. It is a 1-bit register and placed between TDI and TDO when BYPASS instruction is decoded. Bypass register shortens the scan path between TDI and TDO. When boundary scan containing IPs are tested on board, bypass register is helpful to bypass some of the IPs and thus shortening the chain on the board. Also, when TAP is in Capture-DR state of TAP FSM then it captures a fix logic “0” and shifted out in the Shift-DR state.

3.3.2 Identification Register

Device Identification Register is an optional data register in the boundary scan architecture. When provided it is usually 32-bit register that stores complete identification information of the component in which boundary scan is implemented. When TAP FSM is in Capture-DR state it captures 32 bit value of the device Identification Register and the identification bits are shifted out in the subsequent Shift-DR state. This way the register is used to identify the manufacturers, part number and revision of the chip during the testing.

3.3.3 Boundary Register

The boundary register is formed by chain of boundary scan cells at each primary input and each primary output pins. This mandatory register can be used to monitor different activities on device input and output pins. Boundary scan cell can be of different types based on its observability and controllability feature on I/Os. Figure 3.3.1 shows typical symbol for all kind of boundary scan cell. The cell can be used as an input cell or output cell. Boundary Scan Register is formed by serial chain of such types of boundary scan cells. The “Parallel In” and “Parallel Out” are connected to system logic or device pins based on whether the cell is input or output cell. If the boundary scan cell is connected to input pin, “Parallel In” is the primary input of the device and connected to the input pin. “Parallel Out” is connected to the system logic. If the boundary scan cell is connected to the output pin, “Parallel In” is connected to the system logic and “Parallel Out” is the primary output of the device and connected to the output pin of the device.

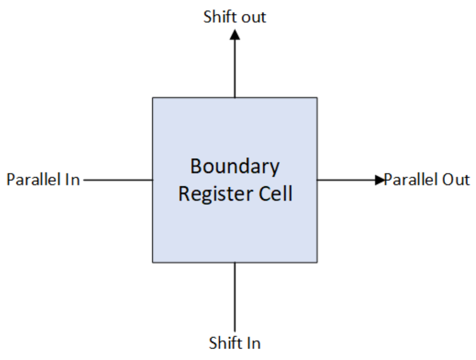


Figure 3.3.1: Typical boundary scan cell symbol

“Parallel In” and “Parallel Out” pins load/ unload data in parallel from input pin to system logic and vice versa. The data can be serially loaded in the boundary scan cell from “Shift In” and “Shift Out”. “Shift In” of first boundary scan cell in the chain is connected to TDI and “Shift Out” of that cell is connected to the next boundary scan cell “Shift In” Input pin. The “Shift Out” is connected to the “Shift In” of next cell. Thus forming a chain of boundary scan cells and eventually a boundary scan shift register. The “Shift Out” of last boundary scan cell in the chain will be connected to the TDO pin.
Figure 2.2.1 shows typical boundary scan cell. During normal mode of operation Mode=0, and data is passed from Parallel In to Parallel Out and the boundary scan cell is transparent to system logic. There are three logical operations performed by boundary scan cell: Capture, Shift ad Update. The data is stored in the capture flop on the pulse of Clock DR, when Shift DR is zero. So the data can be captured from Primary In alongwith logical operation of the chip. When Shift DR is one, the captured data can be shifted out to the next boundary scan cell. By applying a clock pulse to Update DR, data is made available at the output of the update flop. When Mode=1, system logic is made independent of the outside world and update flop output goes in the system logic.

3.4 Instruction Register and Instructions

An Instruction register shifts instruction into the design. Instruction register selects test to be performed or access data registers. A new instruction is being loaded into the register through TDI. At the completion of the shifting process the instruction shifted is being latched at the parallel output and the new instruction gets decoded. The minimum size of instruction register required is two. The two least shift register loads a fix binary value "01" when TAP transits through Capture IR state. Table 3.1 shows the operation of IR in each TAP FSM state.

Controller State	**Shift Register State**	**Parallel Output**
Test Logic Reset	Undefined	Decoded instruction IDCODE(or BYPASS)
Capture IR	Load "01" in LSBs and design data into MSBs	Hold last state
Shift IR	Shift to TDO	Hold last State
Exit1 IR	Hold last state	Hold last state
Exit2 IR		
Pause IR		
Update IR	Hold last state	Load bits from shift register
All other state	Undefined	Hold last state

Table 3.1: Instruction Register operations

3.4.1 BYPASS

The BYPASS instruction will place 1 bit bypass register between TDI and TDO. This instruction and bypass register are mandatory according to the IEEE 1149.1 standard. BYPASS instruction produces short path between scan path when many devices are present on board. Instruction register decodes to BYPASS instruction with opcode of all one's. When BYPASS instruction is decoded bypass register will be parallely loaded to 0 immediately upon TAP FSM transits through Capture-DR state. When TAP is in reset state, the default instruction decoded is BYPASS instruction when idcode register is not present in the device.

3.4.2 IDCODE

This instruction puts 32 bit idcode register between TDI and TDO. IDCODE is noncompulsory instruction. When TAP FSM is in reset state, the default instruction decoded is IDCODE instruction. The Idcode register of the device is parallely loaded with the idcode value of the device immediately upon TAP FSM transists through Capture-DR state. The LSB of Idcode is 1. So the first bit shifted out of TDO will be 1. Four MSB bits are the version number of the chip. Next 16 bits are the manufacturer part number. And next 11 bits are manufacturer identity number which are determined from JEDEC (Joint Electron Device Engineering Council). A component can be identified on a board through TAP pins with the Idcode instruction.

3.4.3 SAMPLE

Figure 3.4.1 shows test data flow when SAMPLE instruction is executed. The SAMPLE instruction places boundary scan register between TDI and TDO. It is the mandatory instruction and its opcode is user defined. SAMPLE instruction doesnot disconnects IC pins from the system core logic. Mode of boundary scan cell is 0. The capture flop of all boundary scan cell captures the data of the IC input in case of input pin and captures data of the system logic in case of the output pin. The Boundary Register thus takes value present at its “Parallel In” pin, taking snapshot of the pin activity being carried out. The operation of test data from the system pins and system logic will be loaded in the capture flop at the rising edge of TCK. The captured data can be shifted out when TAP FSM transits through Shift-DR state and select of the multiplexer becomes 1. Parallel output registers in the boundary scan cell loads the data in system logic or system output pin in the Update DR controller state on the negative edge of TCK.

3.4.4 PRELOAD

Figure 3.4.2 shows test data flow when PREELOAD instruction is executed. The PRELOAD instruction places boundary scan register between TDI and TDO. It is the mandatory instruction and its opcode is user defined. PRELOAD instruction allows shifting of test data between shift registers without affecting the normal operation of data flow. The shifting takes place at the positive edge of TCK and Shift DR = 1 of the multiplexer. The shifted data is parallely loaded

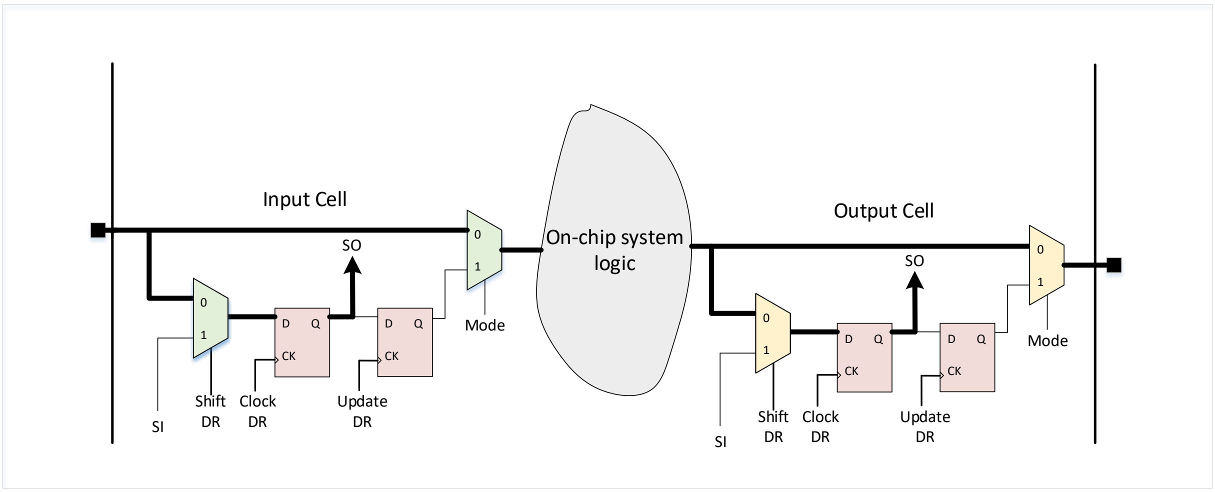


Figure 3.4.1: Data flow while Sample is selected

into the latched registers in the Update DR controller state at the negative edge of TCK. Since the Mode = 1, the data will be blocked and not affect the system logic operation. This updated data allows initial data to be loaded into the system logic before the selection of other instruction like EXTEST instruction so known data is present at the updated flop. Always PRELOAD instruction is executed before EXTEST like instruction to have proper data setup. The opcode of SAMPLE and PRELOAD instruction can be same.

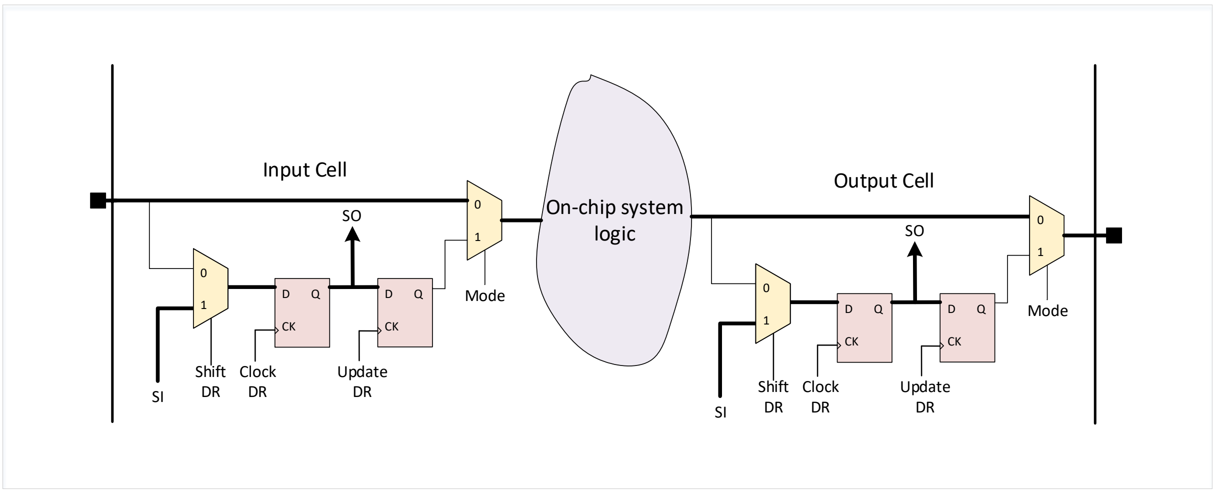


Figure 3.4.2: Data flow while Preload is selected

3.4.5 EXTEST

Figure 3.4.3 shows data flow when EXTEST instruction is executed. The EXTEST instruction places boundary scan register between TDI and TDO. It is the mandatory instruction and its opcode is user defined. The EXTEST instruction is vital instruction of the boundary scan and used in testing of interconnections between various devices of board. The Mode signal is 1 for EXTEST instruction. So the data present at the input of the multiplexer gets updated in the system logic or it is made available at the output pin. The output pin is in control of the boundary scan register. The first stimulus test data to be applied during the EXTEST instruction comes from the PRELOAD instruction. Thus when updated with the EXTEST instruction known data can be driven immediately instead of garbage value.

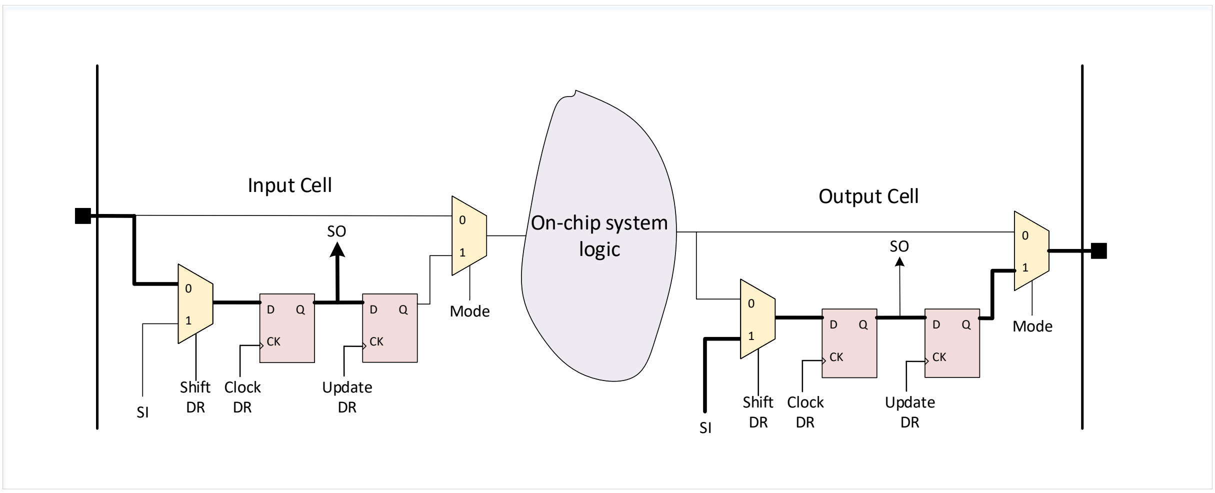


Figure 3.4.3: Data flow while Extest is selected

3.4.6 HIGHZ

The HIGHZ instruction is an optional instruction and places bypass register between TDI and TDO. When HIGHZ instruction is effective, all output drivers are placed in its inactive state. This instruction is purposive when an in-circuit tester wants to drive some desired states to test other chips so that the drivers can be protected by the overdrive damage. HIGHZ instruction causes output pins and bidirectional pins to go in highz state. The highz state of pins occur immediately upon TAP controller transits through Update IR state.

3.4.7 CLAMP

The CLAMP instruction is optional and placed bypass register between TDI and TDO. When CLAMP instruction is effective, all output drivers are under the control of the boundary register and are updated with those values present on it. The boundary register is preloaded with the values with the PRELOAD instruction. The output and bidirectional pins change its state with those preloaded sequences immediately upon the TAP controller transits through Update IR state. CLAMP thus provides digital guarding to the driver pins. When the board is tested, some of the nodes need to be forced with “0” or “1” to set some testable conditions. If these nodes get the source from boundary scan, then these nodes can be protected from overdrive damage.

3.5 Boundary Scan for Advanced Networks

All IC’s input output structures are updated with the advancing in the IC technology and to be compatible with the Moore’s Law. Now the IC pad design have in itself become complex and can contain number of transistors. The increase in complexity allows IC’s to work at higher frequencies of gigahertz range. The technology has advance from single ended transmission to differential signal transmission between ICs. In differential signal transmission of data, data is transmitted on two parallel wires as compared to single ended transmission. Single ended transmission are more prone to noise as they are referenced to fixed voltage levels. It can cause extra transition due to presence of noise. Differential signals are referenced to each other. The other data line is compliment of the original data line. So if noise is added to both lines, it gets cancelled due to compliment nature at the receiver side. So differential signal transmissions are preferred nowadays over single ended transmission due to increase in noise immunity.

3.5.1 Testing Advance I/O

The use of series capacitance either integrated in the chip itself or present on PCB, blocks DC value from passing through the signal path between the driver and receiver. Only the AC component of the signal can pass with the signal being high pass filtered. This is AC coupling between driver and receiver. The use of simple wires between driver and receiver is DC coupling. When

DC coupled, there may be offset errors when transmitting data between two incompatible devices with different set of logic levels, thus degrading their performance. In contrast, when AC coupling is used performance may be increased between two incompatible devices by the bias voltage provided by the bias generation circuit within the receiver or on the board by using AC coupling.
Advance I/O bring up new testing problems. The 1149.1 standard waveform is show in figure 3.5.1 where the data from driver gets updated in the Update DR state and is present at the receiver in the Capture DR state where it is captured. In DC coupled signal propagation is indefinite and is captured properly by the receiver.

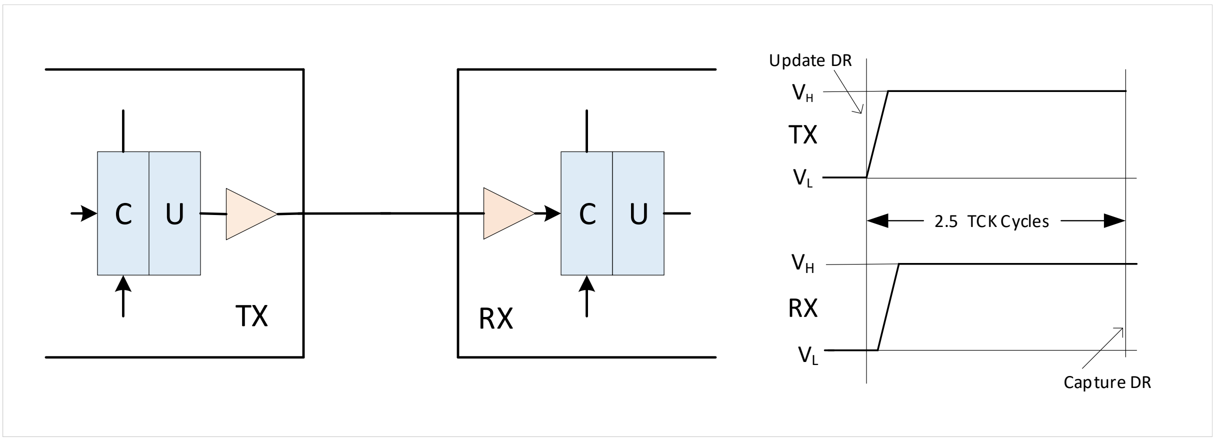


Figure 3.5.1: DC coupled signal propagation

In case of AC coupled signals, the signals get decay away before the Capture DR state of the receiver as shown in figure 3.5.2. IEEE standard 1149.6 [2] provides additional test resources to solve this problem. Advance input-output pins require a time varying signal i.e AC signal for testing of the interconnections. This may include single ended signal pins and differential pins that support AC coupling. It is also expected that a chip with pins supporting AC coupling may possess pins that are DC coupled. There are two types of operational mode: mission mode and test mode. In mission mode device performs its normal function. Test mode is categorized into two modes:

- **DC test mode**: It allows boundary scan testing between DC I/O pins which are DC coupled.

- **AC test mode**: It allows boundary scan testing between AC I/O pins which are AC or DC coupled.

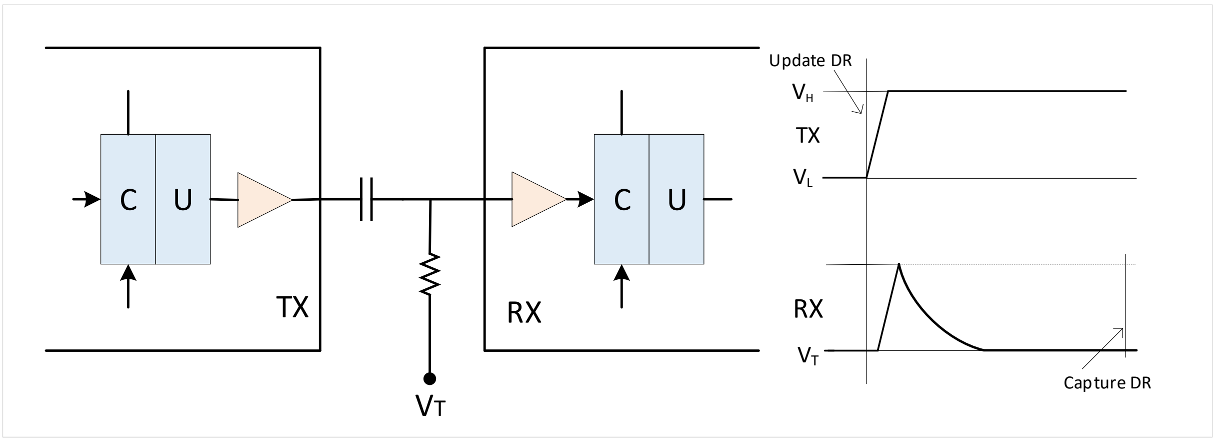


Figure 3.5.2: AC coupled signal decay

The two new variations of EXTEST instruction for advance I/O networks called the AC EXTEST instruction set are EXTEST_TRAIN and EXTEST_PULSE. These instructions will affect the behavior of the AC pins. Any DC pins in the device will behave as specified by EXTEST when either of these instructions for AC pins is selected.

3.5.2 Output drivers

AC pin driver causes transitions in EXTEST_TRAIN or EXTEST_PULSE instructions. The 1149.6 [2] standard functions on producing transitions on driver pins. The 1149.1 [1] standard functions on producing levels. If boundary scan cell behind the driver is loaded with logic one, the driver will be updated with high value until it is again updated with a logic zero. If the boundary scan cell has a logic one after Update DR state, the driver will retain its state. In AC EXTEST instructions it causes atleast two transitions each time TAP transits through Update DR state. The modification in the boundary scan cell as compared to the standard boundary scan cell is shown in figure 3.5.3.

In the test data path, there is capture flop and update flop as in a 1149.1 data cell. A multiplexer is included that selects between the output of the updated flop and the modulated signal with the Ex-Or gate along with AC test signal. For the EXTEST instruction output of the

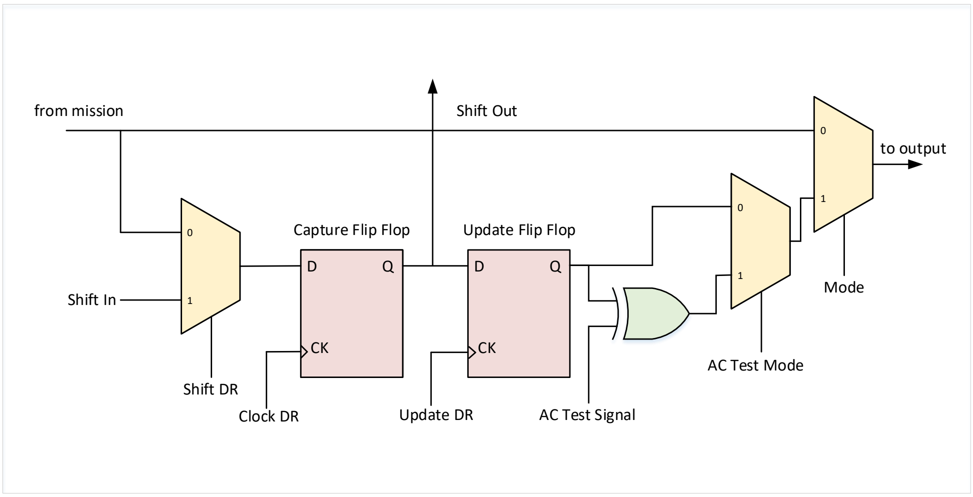


Figure 3.5.3: Boundary scan cell supporting AC instructions

updated flop is selected. For the AC EXTEST instructions, the modulated output is selected. An AC output pin driver expects two extra control signals: the AC test signal and AC test mode. AC test mode is control signal generated from the TAP controller that selects EXTEST or EXTEST_PULSE/EXTEST_TRAIN behavior. EXTEST_PULSE generates a pulse of inverted data on the driver that will remain as it is as long as TAP remains in the RunTest/Idle state. In EXTEST_TRAIN, a train of pulses is seen at half TCK clock cycle. The data in both will become the noninverted in the Select DR scan. The number of TCK clock cycles in the RunTest/Idle state tells the number of transitions between data and inverted data for EXTEST_TRAIN. The AC test signal remains deasserted when the TAP doesnot pass through the RunTest/Idle state ans the effect of AC instructions EXTEST_PULSE and EXTEST_TRAIN is same as the DC EXTEST instruction. EXTEST_PUSLE and EXTEST_TRAIN instructions places boundary scan register between TDI and TDO.

3.5.3 Input Test Receiver

All AC pins that receive data into an IC are implemented with the test receivers. Single ended AC pins have one test receiver and the differential AC pins have one test receiver per leg.

When either EXTEST_TRAIN or EXTEST_PULSE is in effect, the test receiver responds to only valid signal transitions at the receiver pin. The purpose of the test receiver is to reconstruct

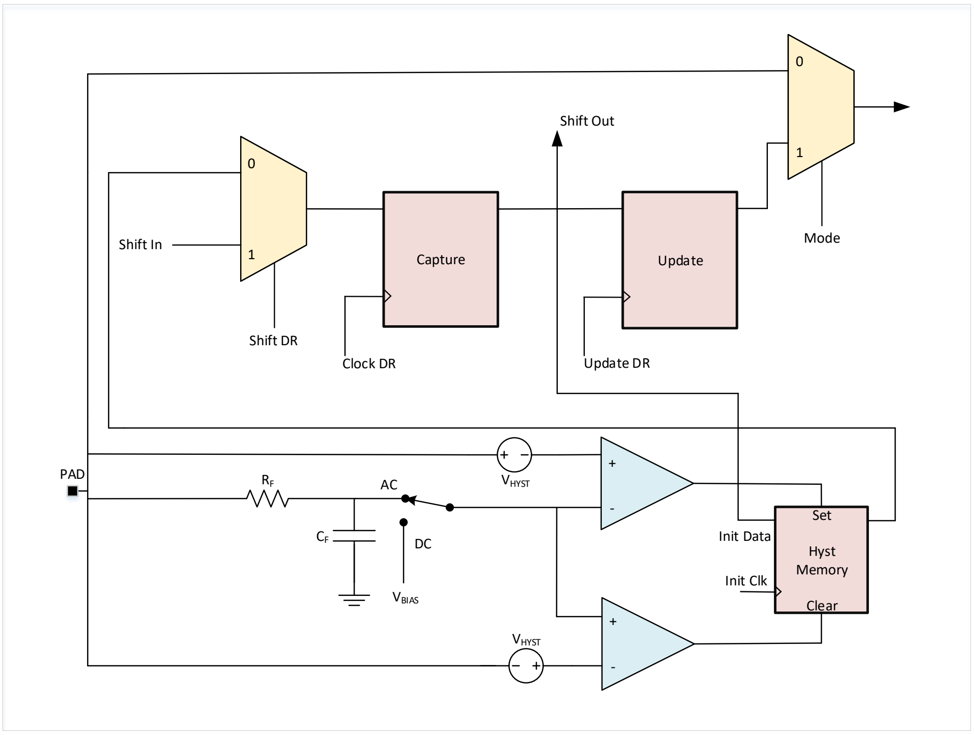


Figure 3.5.4: Test receiver circuit

the test signal which is driven by the upstream driver of that receiver which is either AC or DC coupled. For AC EXTEST instructions, it does this by detecting edges and doesn't respond to the levels of the input waveform. When EXTEST instruction is decoded, the test receiver behaves as a level detector.

The test receiver responds to the voltage levels as seen at the input pin. As shown in figure 3.5.4 an analog switch selects between DC and AC waveforms. For DC the analog switch is connected to the fix bias voltage. The test receiver consists of memory element with two comparators. The output of both the comparators are set and reset pin of the hysteresis memory element of the receiver. Each comparator has one pin from the input connected through the hysteresis voltage and other pin is connected to the common reference bias voltage. When the voltage present on the input pin is greater than $V_{BIAS} + V_{HYST}$, then the upper comparator will set the flipflop. When the voltage on the input pin is lower than the $V_{BIAS} - V_{HYST}$, then the lower comparator will reset the flipflop. The flipflop retains its state when any voltages other then this is present on the input pin. If there is any open circuit on an input pin, there would be no valid voltage level seen at the input pin. The initial data of the test receiver is captured by the boundary scan cell. The data input of the flipflop is the shifted out data from the previous shifting.

When the EXTEST_TRAIN or EXTEST_PULSE is selected, the test receiver responds to the valid signal transitions at the receiver pin. For the AC analog switch is connected to the low pass filtered signal of the input. Here the signals are self compared in the sense they are compared with its delayed version. This circuit forms an "edge detecting" circuit and it responds to only valid transitions. The positive input of the upper comparator detects rising edge of the input and the negative leg detects the delayed version of the input. The flipflop will be set for such rising transitions. The negative input of the lower comparator sees falling edge of the input and the positive leg sees the delayed version of the input. The flipflop will be reset for such falling transitions. The output of the flipflop is generation of the original driven waveform. The initial state are set with data input and clock of the flipflop. For any invalid transitions init data is captured by the boundary scan register cell. The initialisation of the hysteric memory occurs at the negative edge of TCK clock cycle in Exit1 DR or Exit2 DR state.
This chapter described the detailed implementation of boundary scan architecture in the chip. It described function of the TAP FSM, data registers that reside in the TAP, boundary scan instructions supported. It also described additional boundary scan instructions supported for the advance networks.

Chapter 4

Implementation

4.1 Verification Introduction

Verification is part of the VLSI design flow. A VLSI design can be described at different levels of abstraction. The flow consists of converting a higher level description of a design to a lower level description. From the architectural level the design is described at the Register transfer level, which contains combinational and sequential functions to be performed in data and control paths. Each design step of the VLSI design flow needs verification. Verification can be at RTL level, gate level or at the transistor level. The RTL level description must be verified for the functionality of the behavioral description before synthesis to the logical level. The logical level implementation must be verified to guarantee the correct functionality of the final design. In the final step of front end design, the logical level description is transformed into a physical level description to obtain the physical design and interconnection of the transistors in the device before fabrication. The physical level description is verified for the timing and operating frequency specifications. Design added to increase testability also needs to be verified to check the proper implementation of the design. So the insertion of extra hardware in any design for the testing purpose also needs to be verified to check its functionality and proper implementation at the RTL level.

4.2 Flow to generate tests

Boundary scan is verified from the test patterns which are generated from BSDL (Boundary Scan Description Language) file. A flow used to generate boundary scan patterns is provided by the Mentor Graphics Tessent tool[7] is used for verification. Boundary-scan patterns are generated using the Tessent shell. No design data other than a valid BSDL file is required for the pattern generation flow. Pattern signoff file contains a specification of the tests for which tests are to be generated. The test can be an instruction register test, bypass register, boundary scan register, sample, clamp, input, output test, highz. The tests cover all instructions supported by 1149.1[1] standard. It will generate steps in the vector form to verify the instructions which are supported by the design. The output of the flow can be a Verilog test bench or test patterns

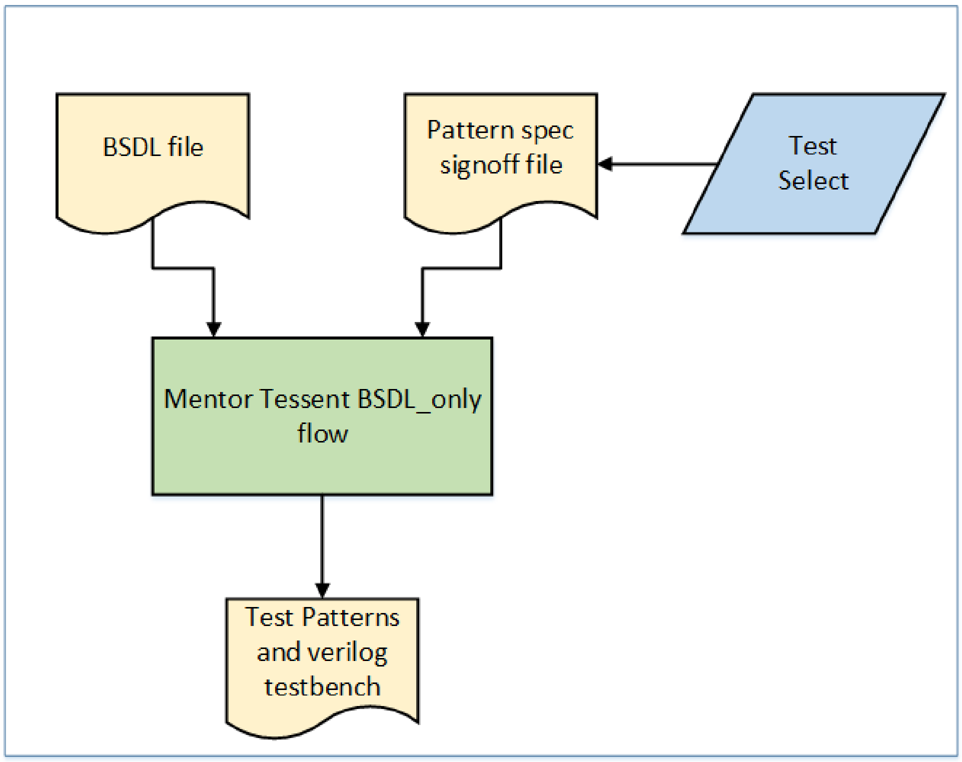

BSDL file
Pattern spec signoff file
Test Select
Mentor Tessent BSDL_only flow
Test Patterns and verilog testbench

Figure 4.2.1: Flow to generate tests

in the form of SVF, WGL, STIL formats.

4.2.1 BSDL file

The BSDL(boundary scan description language) gives a common platform by which IC manufacturers can describe testability characteristics in a chip. BSDL is a part of IEEE Std. 1149.1-2001[1]. BSDL is based on VHSIC hardware description language (VHDL) i.e. BSDL is written in subset of VHDL. It provides information about how a boundary scan IC is implemented in the device. This file is used by ATPG software tools to develop the test patterns for the chip.

The designer will describe the design specific attributes of boundary scan architecture such as length of boundary scan register, length of instruction register, various boundary scan instructions alongwith user defined instructions, opcode of instructions, Input Output pins assignment. It also describes which pin is connected to TDI and which pin is connected to TDO. It specifies the complete boundary scan chain connection of all pins from TDI to TDO, package of the chip, function of the primary pins of the device, scan port description, all data registers description. The CAD tools have also the capability of generating boundary scan circuitry and create test patterns from it based on the above information. The BSDL file includes the following data from which the tool will pick the data to create test patterns for boundary scan testing:

4.2.1.1 Entity Descriptions

The Entity Declaration is used to identify the name of the device which is described by the BSDL file. All the BSDL statements are written within the entity description.

Example:

```
entity xyz_ip is
--BSDL statements to describe the entity
end xyz_ip;
```

4.2.1.2 Generic Parameter

The Generic Parameter is the part that specifies which package is specified for the chip. The generic parameter allows the external application to select IC package from the BSDL file.

Example:

```
generic(PHYSICAL_PIN_MAP : string := "LGA");
```

4.2.1.3 Logical Port Description

Logical port description provides logical name to the Input/Output pins. It defines the attributes of all pins describing it as an input (in bit;), bi-directional (inout bit;), output (out bit;) or if it is not for boundary-scan (linkage bit;).

Example:

```
port (TCK, TMS, TDI: in bit;
TDO: out bit;
RXP_0:  in bit;
RXN_0:  in bit;
TXP_0:  buffer bit;
TXN_0:  buffer bit;
VCC, GND: linkage bit);
```

4.2.1.4 Standard USE statement

This statement is used to set the description of the BSDL. To include all features of 1149.1 standard “use STD_1149_1_2001.all” is used.

Example:

```
use STD_1149_1_2001.all;
use STD_1149_6_2003.all;
```

4.2.1.5 Device Package Pin Mappings

The Package Pin Mapping is used to determine the internal connections within an integrated circuit. It describes the mapping of package pins to logical names from port description.

Example:

```
attribute PIN_MAP of xyz_ip :  entity is PHYSICAL_PIN_MAP;
constant LGA : PIN_MAP_STRING :=
"RXP_0:1, RXN_0:2, TXP_0:3, TXN_0:4," &
```

```
"TDI:5, TMS:6, TCK:7, TDO:8";
```

4.2.1.6 Grouped Port Identification

Port grouping describes the pairing of pins which are differential to each other. If two pins carry data, differential signaling is used. Differential pairs in an IC are used to improve noise immunity.

Example:

```
attribute PORT_GROUPING of xyz_ip :  entity is
"Differential_Voltage ((TXP_0, TXN_0))";
```

4.2.1.7 Scan Port Identification

It identifies the dedicated pins that are used for the boundary scan implementation. These include TDI, TMS, TCK, TDO, and TRST (if used).

Example:

```
attribute TAP_SCAN_IN of TDI : signal is true;
attribute TAP_SCAN_MODE of TMS : signal is true;
attribute TAP_SCAN_OUT of TDO : signal is true;
attribute TAP_SCAN_CLOCK of TCK : signal is (40.0e6, BOTH);
attribute TAP_SCAN_RESET of TRST : signal is true;
```

4.2.1.8 Instruction Register Description

Instruction Register Description describes the optional, mandatory and user defined instructions. It contains instruction register length, opcodes of each instruction and and instruction register capture pattern.

Example:

```
attribute INSTRUCTION_LENGTH of xyz_ip :  entity is 8;
attribute INSTRUCTION_OPCODE of xyz_ip :  entity is
"extest            (00001001)," &
"sample            (00000001)," &
```

```
"idcode            (00001100)," &
"clamp             (00000100)," &
"highz             (00001000)," &
"extest_pulse      (00001110)," &
"extest_train      (00001111)," &
"bypass            (11111111) " ;
attribute INSTRUCTION_CAPTURE of xyz_ip:  entity is "00000001";
```

4.2.1.9 Register Access Description

Register Access Description describes which data registers will be placed between TDI and TDO when the instructions are decoded after Update IR.

Example:

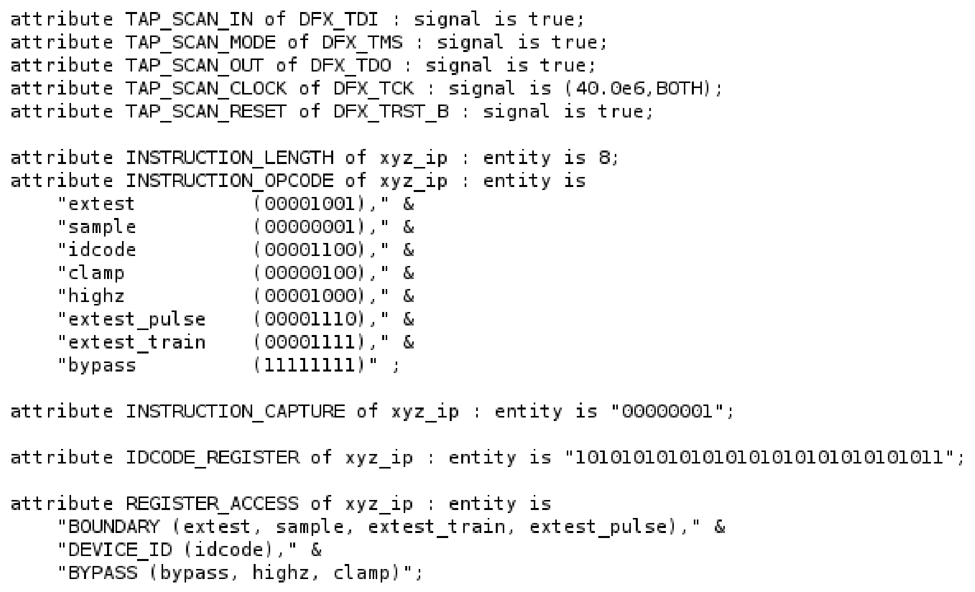
```
attribute REGISTER_ACCESS of xyz_ip :  entity is
"BOUNDARY  (extest, sample, extest_train, extest_pulse)," &
"DEVICE_ID (idcode)," &
"BYPASS    (bypass, highz, clamp)";
```

4.2.1.10 Boundary Register Description

Boundary Register Description provides the structure of the boundary-scan cells on the device. The boundary scan cell can be of different types for eg. bc_1, bc_2, etc. based on its structure and types of instruction it supports.

Example:

```
attribute BOUNDARY_LENGTH of xyz_ip :  entity is 4;
attribute BOUNDARY_REGISTER of xyz_ip :  entity is
-- num cell   port    function safe [ccell disval rslt]
"   3  (BC_1, TXP_0 , output2,  X)," &
"   2  (BC_0, *     , INTERNAL, X)," &
"   1  (BC_1, RXP_0 , input,    X)," &
"   0  (BC_1, RXN_0 , input,    X) " ;
```



```
attribute TAP_SCAN_IN of DFX_TDI : signal is true;
attribute TAP_SCAN_MODE of DFX_TMS : signal is true;
attribute TAP_SCAN_OUT of DFX_TDO : signal is true;
attribute TAP_SCAN_CLOCK of DFX_TCK : signal is (40.0e6,BOTH);
attribute TAP_SCAN_RESET of DFX_TRST_B : signal is true;

attribute INSTRUCTION_LENGTH of xyz_ip : entity is 8;
attribute INSTRUCTION_OPCODE of xyz_ip : entity is
    "extest         (00001001)," &
    "sample         (00000001)," &
    "idcode         (00001100)," &
    "clamp          (00000100)," &
    "highz          (00001000)," &
    "extest_pulse   (00001110)," &
    "extest_train   (00001111)," &
    "bypass         (11111111)" ;

attribute INSTRUCTION_CAPTURE of xyz_ip : entity is "00000001";

attribute IDCODE_REGISTER of xyz_ip : entity is "10101010101010101010101010101011";

attribute REGISTER_ACCESS of xyz_ip : entity is
    "BOUNDARY (extest, sample, extest_train, extest_pulse)," &
    "DEVICE_ID (idcode)," &
    "BYPASS (bypass, highz, clamp)";
```

Figure 4.2.2: Example of BSDL

4.2.2 SVF file

Serial vector format (SVF) provides the platform for exchanging descriptions of high level IEEE 1149.1 operations. The operations consist of scan operations and transition between different stable states of the TAP controller state diagram. SVF doesn't explicitly describe the state at every TCK. SVF is intended to reuse serial vectors during the life cycle of the product, from its implementation in the design stage to its use in the field. The SVF file is an ASCII file that consists of a set of different SVF statements. Each SVF commands are terminated by a semicolon. Scan data is expressed as hexadecimal within a statement.
The following set of SVF commands are supported:

- **ENDDR**: Indicates default end state for data registers scan operations. The stable states for ending scan operations of data registers are IRPAUSE, DRPAUSE, RESET, and IDLE.
- **ENDIR**: Indicates default end state for instruction register scan operations. The stable states for ending scan operations of instruction registers are IRPAUSE, DRPAUSE, RESET, and IDLE.
- **FREQUENCY**: Indicates maximum test clock frequency for different TAP operations.
- **PIO**: Specifies parallel test vectors to be loaded in all capture flop of data registers. The

characters of the PIO are H to drive logic 1, L to drive logic 0, U to detect logic 1, D to detect logic 0, Z drives high impedance, X detects unknown. The number of vectors in the PIO command equals the number of the logical names specified in the PIOMAP statement.

- **PIOMAP**: It maps PIO positions to the logical pin. It defines the input-output direction and logical name for each column mentioned in PIO. The first logical name in the PIOMAP maps to the first character of the string in PIO statements.
- **RUNTEST**: Forces the IEEE 1149.1 operations to the runtest/idle state for the specified period or the number of TCK clock cycles. This is used to control the operations of the RUNBIST instruction of the device.
- **SDR**: (Scan Data Register) Performs an IEEE 1149.1 Data Register scan operation between TDI and TDO. It specifies the value to be scanned into the target device and the values to be compared against the expected values scanned out of the target. When comparing TDO values, the mask option is provided to specify the number of bits to be compared. All values are coded as a HEX string.
- **SIR**: (Scan Instruction Register) Performs an Instruction Register shift operation between TDI and TDO. It specifies the value to be scanned into the instruction register of the target device and the values to be compared against the expected values scanned out of the instruction register of the target. When comparing TDO values, the mask option is provided to specify the number of bits to be compared. All values are coded as a hexadecimal string.
- **STATE**: Forces the TAP state to a specified stable state. It provides a list of states through which one can describe explicit states to travel through the TAP state machine. If the path state list is not provided default path is assumed based on the current state and the final stable state. IRPAUSE, DRPAUSE, RESET, and IDLE are the valid stable states.
- **TRST**: (Test Reset) Controls the optional Test Reset line to reset the TAP controller. It must be placed at the beginning of any statements to provide a reset sequence of boundary scan in the beginning.

4.3 Test Algorithm

Boundary scan is verified by the test patterns generated by the above mentioned Mentor Tessent tool using just the description provided in the BSDL file. The test patterns can be generated for various tests supporting multiple instructions of boundary scan. Below is described how the patterns are mapped from the BSDL file. Tessent tool receives all necessary information of the boundary scan to generate tests for the particular instructions and maps it to the SVF commands with the scanning sequence. The test algorithm with the scanning sequence for test logic reset, bypass reg, instruction register, sample, extest, extest_pulse, extest_train tests which are to be verified are described in the following sections.

4.3.1 Test Logic Reset Test

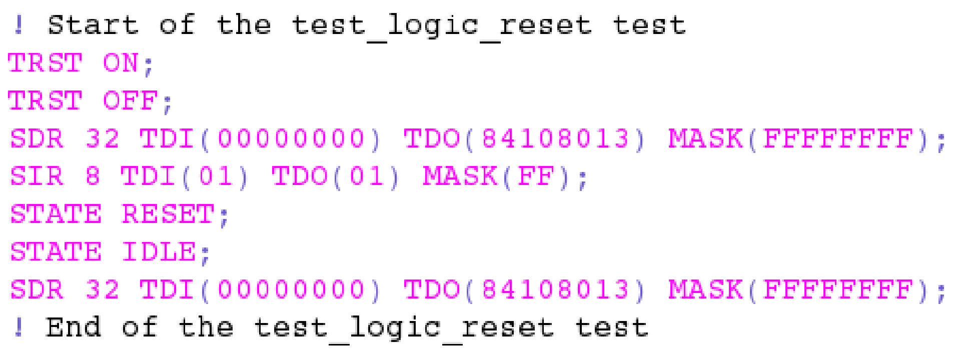
The following steps describes steps from the SVF file generated for the verification of the hard reset and soft reset of the boundary scan.

- **Step 1**: Initialise the TAP controller by pulsing TRST to 1-0-1 stream.
- **Step 2**: Check if RESET of TAP controller accomplished by reading the IDCODE value from TDO.
- **Step 3**: Preload Instruction register with SAMPLE instruction, which differs from the expected post reset Instruction register's instruction.
- **Step 4**: Reset the TAP controller by setting TMS=1 for 5 TCK cycles.
- **Step 5**: Check if RESET of TAP controller accomplished by reading the IDCODE value from TDO.

4.3.2 Instruction Register Test

The following steps describes steps from the SVF file generated for the verification of the instruction register of the TAP architecture.

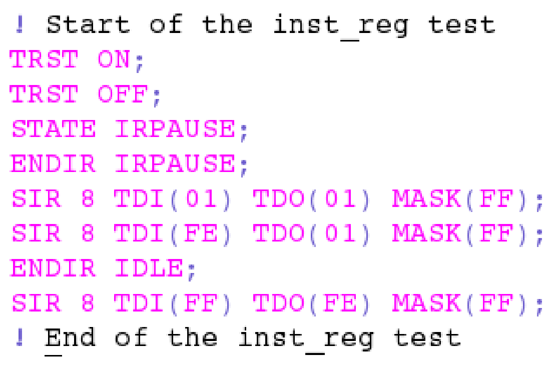
- **Step 1**: Verify the TAP FSM states sequence by skipping Shift-IR state via IDLE$\rightarrow$SELECT-DR$\rightarrow$ CAPTURE-IR$\rightarrow$EXIT1-IR$\rightarrow$PAUSE-IR.



```
! Start of the test_logic_reset test
TRST ON;
TRST OFF;
SDR 32 TDI(00000000) TDO(84108013) MASK(FFFFFFFF);
SIR 8 TDI(01) TDO(01) MASK(FF);
STATE RESET;
STATE IDLE;
SDR 32 TDI(00000000) TDO(84108013) MASK(FFFFFFFF);
! End of the test_logic_reset test
```

Figure 4.3.1: SVF file of reset test

- **Step 2**: Load the TDI via instruction register with walking one pattern ..0001.
- **Step 3**: Shift the first capture value of ..xx01 through TDO to verify proper values got captured in the instruction register.
- **Step 4**: End shifting in the Pause-IR state.
- **Step 5**: Again, load the TDI via instruction register with walking zero pattern ..1110.
- **Step 6**: Shift the value of previous walking zero pattern from the shift register to the TDO, to verify proper values shifted out from instruction register.
- **Step 7**: Again, load the TDI via instruction register with all one's pattern ..111 and shift the value of previous walking one pattern from the instruction register to the TDO.



```
! Start of the inst_reg test
TRST ON;
TRST OFF;
STATE IRPAUSE;
ENDIR IRPAUSE;
SIR 8 TDI(01) TDO(01) MASK(FF);
SIR 8 TDI(FE) TDO(01) MASK(FF);
ENDIR IDLE;
SIR 8 TDI(FF) TDO(FE) MASK(FF);
! End of the inst_reg test
```

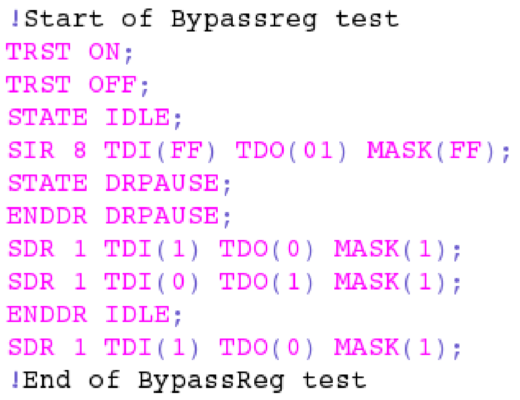
Figure 4.3.2: SVF file of instruction register test

Figure 4.3.2 shows the SVF file for the above mentioned steps. SIR commands attains value 8 from the Instruction register length specified in the BSDL file. It shifts “01” value in the TDI and scans out walking zero pattern i.e “FD” coded as hex value from the instruction register. Then again with the SIR command it scans in “FE” value in the TDI through the instruction register and scans out previously scanned in value in the TDI. Lastly the scan in pattern in TDI is “FF” and previously scanned in pattern is shifted out.

4.3.3 Bypass Register test

The following steps describes steps from the SVF file generated for the verification of the bypass register of the TAP architecture.

- **Step 1**: Verify the TAP FSM state sequences by skipping Shift-DR via IDLE→SELECT-DR→ CAPTURE-DR→EXIT1-DR→PAUSE-DR.
- **Step 2**: Load IR with the BYPASS instruction by shifting 11111111 in TDI. This places bypass register between TDI and TDO after Update-IR state.
- **Step 3**: Shifting first stimulus pattern logical one in the bypass register.
- **Step 4**: Shift the first capture value of logical zero through TDO to verify standard imposed value got captured in the bypass register.
- **Step 5**: End shifting in the Pause-DR state.
- **Step 6**: Shift second stimulus pattern of logical zero and compare the previous shifted pattern at TDO with logical one.
- **Step 7**: End shifting in the Pause-DR state.
- **Step 8**: Shift the last stimulus pattern with logical one and compare the previous shifted pattern at TDO with logical zero.



```
!Start of Bypassreg test
TRST ON;
TRST OFF;
STATE IDLE;
SIR 8 TDI(FF) TDO(01) MASK(FF);
STATE DRPAUSE;
ENDDR DRPAUSE;
SDR 1 TDI(1) TDO(0) MASK(1);
SDR 1 TDI(0) TDO(1) MASK(1);
ENDDR IDLE;
SDR 1 TDI(1) TDO(0) MASK(1);
!End of BypassReg test
```

Figure 4.3.3: SVF file of bypass register test

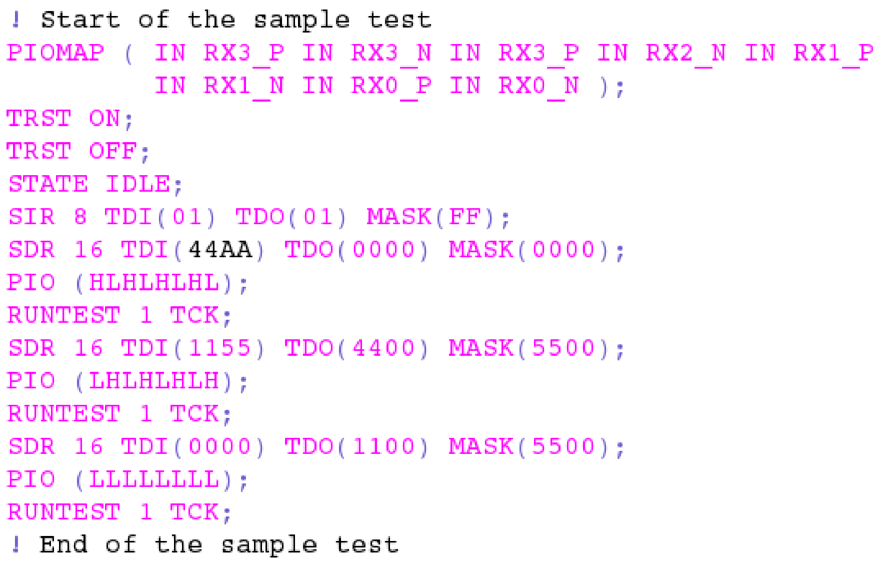
Figure 4.3.3 shows the SVF file for the above mentioned steps. SIR commands attains value 8 from the Instruction register length specified in the BSDL file. It shifts the opcode of the bypass instruction "11" value in the TDI and scans out default captured value of "0". With the SDR command of 1 bit, the scanning sequence is such that it consequently applies "0" and "1" pattern in the bypass register. The shifted sequence are captured during the subsequent shifting data register operation and shifted out through TDO.

4.3.4 Sample test

The following steps describes steps from the SVF file generated for the verification of the sample instruction of the boundary scan.

- **Step 1**: Load the instruction register with the SAMPLE instruction by shifting 00000001 in TDI. This places boundary scan register between TDI and TDO after Update-IR state.
- **Step 2**: Shift first stimulus even pattern in the boundary scan register.
- **Step 3**: Load parallel vectors in the input pins with odd pattern in the RunTest/Idle state before Capture-DR.
- **Step 4**: Shift next stimulus odd pattern in the boundary scan register and check the boundary scan captured values at TDO.
- **Step 3**: Load parallel vectors in the input pins with even pattern in the RunTest/Idle state before Capture-DR.
- **Step 4**: Shift next stimulus odd pattern in the boundary scan register and check the boundary scan captured values at TDO.
- **Step 5**: Load parallel vectors in the input pins with the safe pattern.
- **Step 6**: Shift the safe value in the boundary scan register and check the boundary scan captured values at TDO.

Figure 4.3.4 shows the SVF file for the above mentioned steps. Instruction register is loaded with sample instruction. Input pins are driven with the even and odd patterns using PIO command. "H" and "L" drives to logical one and logical zero respectively. The captured values are shifted through TDO. Thus sample instruction gets verified.



```
! Start of the sample test
PIOMAP ( IN RX3_P IN RX3_N IN RX3_P IN RX2_N IN RX1_P
         IN RX1_N IN RX0_P IN RX0_N );
TRST ON;
TRST OFF;
STATE IDLE;
SIR 8 TDI(01) TDO(01) MASK(FF);
SDR 16 TDI(44AA) TDO(0000) MASK(0000);
PIO (HLHLHLHL);
RUNTEST 1 TCK;
SDR 16 TDI(1155) TDO(4400) MASK(5500);
PIO (LHLHLHLH);
RUNTEST 1 TCK;
SDR 16 TDI(0000) TDO(1100) MASK(5500);
PIO (LLLLLLLL);
RUNTEST 1 TCK;
! End of the sample test
```

Figure 4.3.4: SVF file of sample test

4.3.5 Extest test

The following steps describes steps from the SVF file generated for the verification of the extest instruction of the boundary scan.

- **Step 1**: Set the input pins with an even pattern.
- **Step 2**: Load IR with the PRELOAD instruction by shifting 00000001 in TDI. This places boundary scan register between TDI and TDO after Update-IR state.
- **Step 3**: Shifting the first stimulus even pattern in the boundary scan register. This is the preload data for the extest instruction.
- **Step 4**: Load IR with the EXTEST instruction by shifting 00001001 in TDI. This will apply the first parallel test vector and the output pins get updated with the even pattern after Update IR state.
- **Step 5**: Set the input pins with an odd pattern and detect the updated patterns in the RunTest/Idle state.
- **Step 5**: Shift next stimulus odd pattern in the boundary scan register. The output pins get updated with the odd pattern after Update DR state.
- **Step 6**: Compare the captured TDO data for an odd pattern and load the boundary scan chain with the safe pattern.

```
! Start of the extest test
PIOMAP ( IN RX3_P IN RX3_N IN RX3_P IN RX2_N IN RX1_P IN RX1_N IN
         RX0_P IN RX0_N OUT TX0_P OUT TX0_N OUT TX1_P OUT TX1_N
         OUT TX2_P OUT TX2_N OUT TX3_P OUT TX3_N );
TRST ON;
TRST OFF;
PIO (LHLHLHLHXXXXXXXX);
SIR 8 TDI(01)TDO(01) MASK(FF);
SDR 16 TDI(44AA) TDO(0000) MASK(0000);
SIR 8 TDI(09) TDO(01) MASK(FF);
RUNTEST 1 TCK;
PIO (LHLHLHLHDUUDDUUD);
RUNTEST 1 TCK;
SDR 16 TDI(1155) TDO(4400) MASK(5500);
RUNTEST IDLE 1 TCK ENDSTATE IDLE;
PIO (HLHLHLHLUDDUUDDU);
ENDDR IDLE;
SDR 16 TDI(0000) TDO(1100) MASK(5500);
! End of the extest test
```

Figure 4.3.5: SVF file of extest test

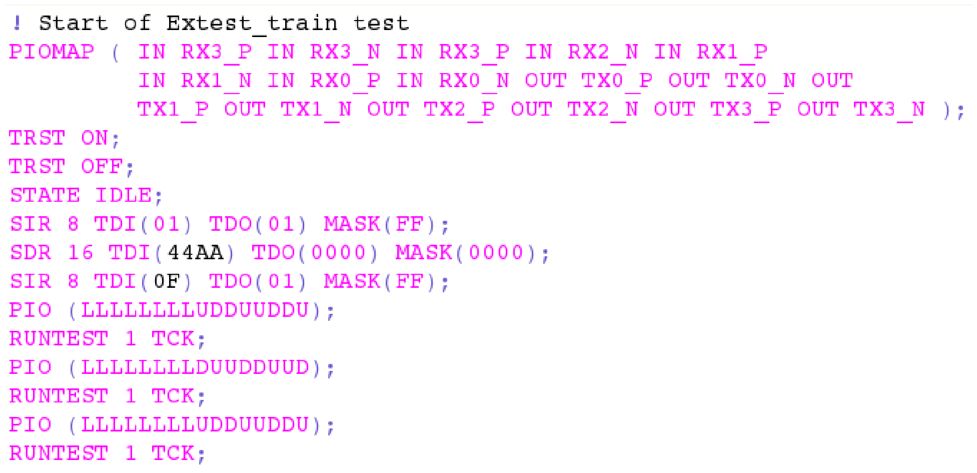
Figure 4.3.5 shows the SVF file for the above mentioned steps. Instruction register is decoded with preload instruction. Output pins are updated with the even and odd patterns from the preloaded data. “U” and “D” detects logical one and logical zero respectively. Thus we can compare if the data on the output pins gets updated with the preloaded data correctly. This data becomes the driver for other pins and is used to test the interconnection between chips. Thus the extest instruction will be verified with this sequence.

4.3.6 Extest_train test

The following steps describes steps from the SVF file generated for the verification of the Extest_train instruction of the boundary scan.

- **Step 1**: Load IR with the PRELOAD instruction by shifting 00000001 in TDI. This places boundary scan register between TDI and TDO after Update-IR state.
- **Step 2**: Shifting the first stimulus even pattern in the boundary scan register. This is the preload data for the extest_train instruction.
- **Step 3**: Load IR with the EXTEST_TRAIN instruction by shifting 00001001 in TDI. This will apply the first parallel test vector and the output pins get updated with the even pattern after Update IR state.

- **Step 4**: Check the updated output pins with even pattern over 3 TCK cycles to check the toggling of output pins in RunTest/Idle state.



```
! Start of Extest_train test
PIOMAP ( IN RX3_P IN RX3_N IN RX3_P IN RX2_N IN RX1_P
         IN RX1_N IN RX0_P IN RX0_N OUT TX0_P OUT TX0_N OUT
         TX1_P OUT TX1_N OUT TX2_P OUT TX2_N OUT TX3_P OUT TX3_N );
TRST ON;
TRST OFF;
STATE IDLE;
SIR 8 TDI(01) TDO(01) MASK(FF);
SDR 16 TDI(44AA) TDO(0000) MASK(0000);
SIR 8 TDI(0F) TDO(01) MASK(FF);
PIO (LLLLLLLLUDDUUDDU);
RUNTEST 1 TCK;
PIO (LLLLLLLLDUUDDUUD);
RUNTEST 1 TCK;
PIO (LLLLLLLLUDDUUDDU);
RUNTEST 1 TCK;
```

Figure 4.3.6: SVF file of extest_train test

Figure 4.3.6 shows the SVF file for the above mentioned steps. Instruction register is loaded with PRELOAD instruction to update the output cell to the known value in the EXTEST_TRAIN instruction. This updated value gets inverted in RunTest/Idle test at every negative edge of TCK. The driven data and the inverted data are compared with the “U” and “D” in the PIO commands which detects logical one and logical zero respectively.

4.4 Integration in Environment

The SVF file generated as mentioned in the previous section needs to be verified. It is integrated in the OVM testbench environment. The results are compared with actual versus expected value. The major components of the testbench environment are top, testbench, JTAG agent. The Testbench top instantiates DUT and connects the DUT signals to the interface. Test selects the SVF file which is driven to the DUT. Here different tests can be provided. Rest of the environment remains same. So Verification re-usability is achieved and any boundary scan test can be verified by just integrating one test pattern file for a particular BSDL file generated test content. Test contains environment in which scoreboard and JTAG agent are instantiated. JTAG driver, JTAG monitor and JTAG sequencer are instantiated in the JTAG agent. SVF commands are converted into sequences which is read by JTAG sequencer. The sequencer values are the transition value of FSM, TDI value, Instruction Register bits and TDO value for comparison in scoreboard. The JTAG driver receives the packet from the sequencer and drive JTAG signals to the JTAG interface. JTAG driver drives TDI values from the SVF commands which are converted into sequences. Signals at JTAG interface are driven at pin level and applied to DUT. JTAG Monitor captures signal activity from the DUT and send to other components. The captured signals are compared in the scoreboard where checkers are present which compares between expected and actual TDO value and output pin values as generated by the test.

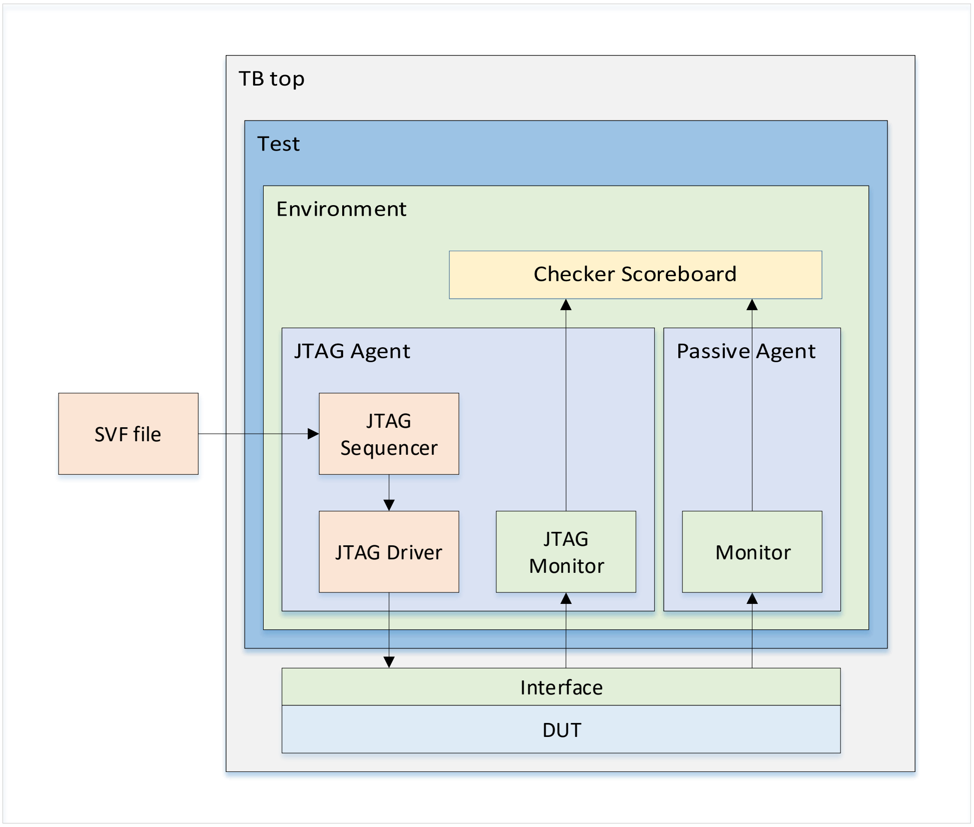


Figure 4.4.1: OVM testbench environment

4.4.1 Components in the OVM testbench

An OVM testbench is composed of reusable verification environment called OVM verification components. OVM testbench is used because of its configurability, reusable environment. OVM components are integrated with the DUT to verify the implementation of the design.

4.4.1.1 JTAG Sequencer

JTAG sequencer generates stimulus which are provided to the driver upon request. It reads SVF file and coverts the SVF commands into JTAG sequences. It captures the data into sequences present into SVF file. It drives the data to the driver with packet level transactions. The various fields that will pass from sequencer to the driver are TMS input stream, TDI input stream, expected TDO bits, masking of TDO bits for comparison, reset status of TAP controller, IR load bits, DR load bits, Expected DR bits and Actual DR bits.

4.4.1.2 JTAG Driver

JTAG driver consists of the tasks that drives the DUT through the pin interface. It receives packet level transactions from the sequencer and converts into pin level to drive the DUT. Data is driven through the JTAG interface of the DUT. Other data is driven to the input pins of the DUT.

4.4.1.3 JTAG Monitor

JTAG monitor converts the data on the output pins being monitored to the transactions. It sends those transaction level packet to the scoreboard component for comparison with the golden responses. Monitor samples the DUT signals but doesnot drive them. It updates the scoreboard at each clock. JTAG monitor sends packet to the scoreboard, TAP FSM state to sample data in that state, Shifting in of TDI data, number of shifted TDI data in SHDR state, parallel data driven through the input pins before CADR state.

4.4.1.4 JTAG Agent

JTAG agent encapsulates JTAG driver, JTAG monitor and JTAG sequencer. The agent will initiate transactions to the DUT. Agent is a container for driver, monitor and sequencer. Agents are configurable to be active or passive agent. Active agent is capable of driving the generated stimulus to the DUT. interface.

4.4.1.5 Passive Agent

Passive agent is capable of only monitoring the DUT signals. Monitor in passive agent is used for observing the pin signals of the DUT. The signals from the DUT are converted into packet level transactions. The packet will be send to the scoreboard for comparison with the expected output data. The packet will consists of DUT signals that don't need driving like Output pins of the DUT, TAP FSM state to sample data in that state, Shifted out TDO data, number of shifted TDO data in SHDR state, parallel data out through the output pins in UPDR state.

4.4.1.6 Scoreboard

Scoreboard compares the output transactions from the DUT with the expected value. The values compared will be shifted out TDO data versus the expected TDO data, parallel data of output pins with its expected value.

4.4.1.7 Environment

The Environment is the top level component instantiating JTAG Agent and Passive agent, scoreboard. The environment consists of the configurable properties with which one can configure it from different data registers length, instruction register length for different design under test.

4.4.1.8 Test

Test instantiates the Environment. Test selects the svf file which is given to the sequencer to read different svf commands. SVF file can contain tests for test logic reset, instruction register, bypass register, boundary scan register, sample, extest. So different tests can be provided. Rest of the environment will remain same and is reusable.

4.4.1.9 Top

Top instantiates test and the design under test and interface.
Thus various test patterns for the verification of different boundary scan instructions can be generated and verified by the above testbench environment.

Chapter 5

Results and Discussions

This chapter discuss about the results obtain from the previous methodology used for the verification of boundary scan. The SVF file generated for the verification of boundary scan are validated using the OVM testbench environment described in previous chapter. The simulation is done in Synopsys VCS tool to view the waveforms and debug the RTL schematic in case of miscompares in the result. The below sections shows the result passing waveform of bypass, test logic reset, extest and extest_train test. One of the failing scenario for idcode register test is also described.

5.1 Bypass test

The BYPASS instruction is verified by placing 1-bit bypass register between TDI and TDO. Firstly instruction register is shifted with the bypass opcode. After update IR, instruction register is decoded with the BYPASS instruction having opcode of “FF”. Bypass register is loaded with a logic one via TDI and captured value is compared with the logic zero. Then loading Bypass register with a zero and TDO is compared with previously scanned in value of one. Bypass register is again loaded with a logical one and comparing captured value with logic zero. Figure 5.1.1 shows the simulated response of the bypass test covering all the above mentioned scenario. Capture, shift and update operations of bypass register after decoding BYPASS instruction is verified by this test.

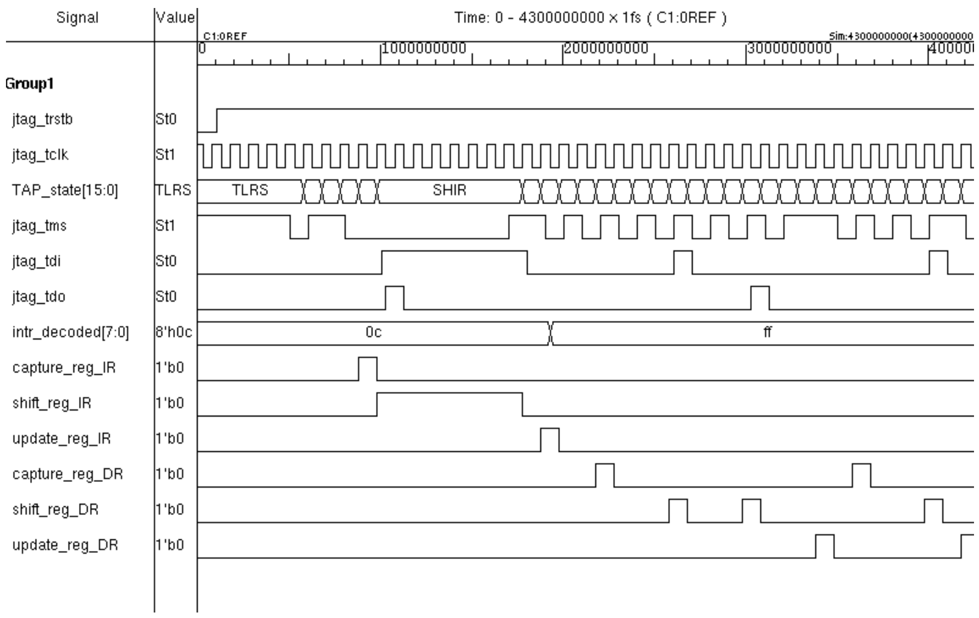


Figure 5.1.1: Waveform of bypass test

5.2 Test logic reset test

Test logic reset instruction is verified by placing TAP controller in reset state by both hard reset and soft reset. TAP FSM transverse through different states as per the SVF commands. Figure 5.2.1 shows the simulation result of the test. The default decoded instruction is IDCODE instruction with opcode value of “0C”. The test verifies if the instruction decoder selects idcode instruction when the TAP is resetted by hard and soft reset. When TAP FSM enters capture state, Idcode register (32-bit) value is captured in the reset state. In shifting state the captured idcode value is being shifted out through TDO. If the correct value is shifted out then idcode register is verified alongwith reset of TAP controller. So it is checked that default instruction decoded is Idcode when reset occurs. After that instruction register is loaded with other instruction then Idcode. Same scenario is verified for soft reset by applying consecutive five 1’s to the TMS. By applying five consecutive one to TMS, TAP controller state comes to Test logic reset state of the FSM. IDCODE instruction decoded. Capture, shift and update operations of idcode register is verified by this test.

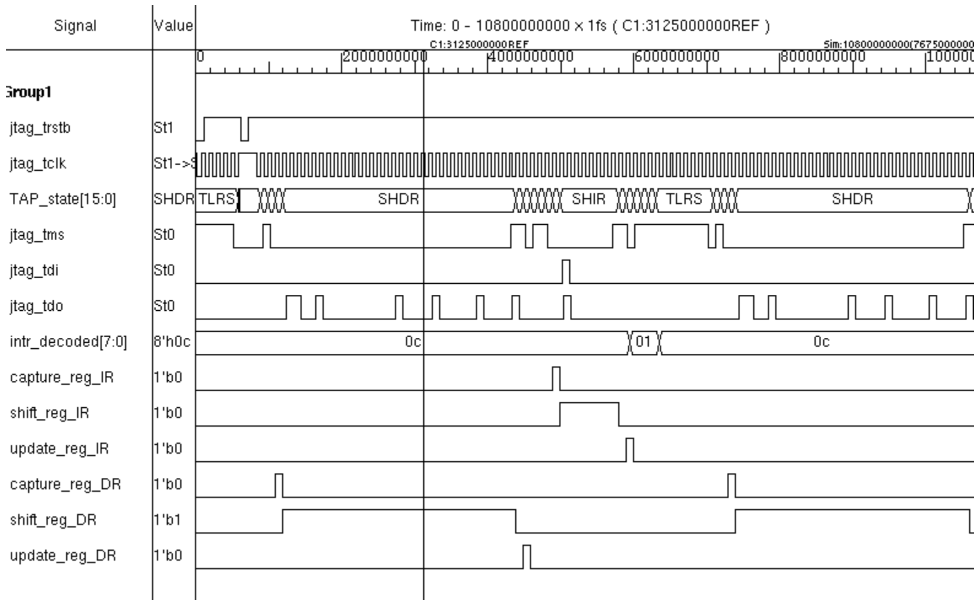


Figure 5.2.1: Waveform of test logic reset

5.3 Extest test

Extest test is verified by placing the boundary scan register between TDI and TDO. Instruction register is loaded with PRELOAD instruction of opcode “01”. After the boundary scan chain is loaded with even pattern (even pins set to 1) by shifting bits through TDI. Here length of boundary scan chain is 16, so the stimulus is clocked 16 times and shifted through TDI. Now the instruction register is loaded with EXTEST instruction of opcode “09” which allows output pins to be in control with the boundary scan cell. After Update-IR output pins gets updated with the predefined values i.e. even pattern and they are compared with those patterns.
Boundary scan chain is loaded with an odd pattern (odd pins set to 1) through TDI and boundary scan captured values from the previous even pattern are shifted through TDO and compared with the expected value. Figure 5.3.1 shows the simulated response of the extest test covering all the above mentioned scenario. The output pins are updated with opposite pattern after Update-IR and consecutive Update-DR state. Capture, shift and update operation of boundary scan register placed between TDI and TDO in Extest instruction gets verified by this test.

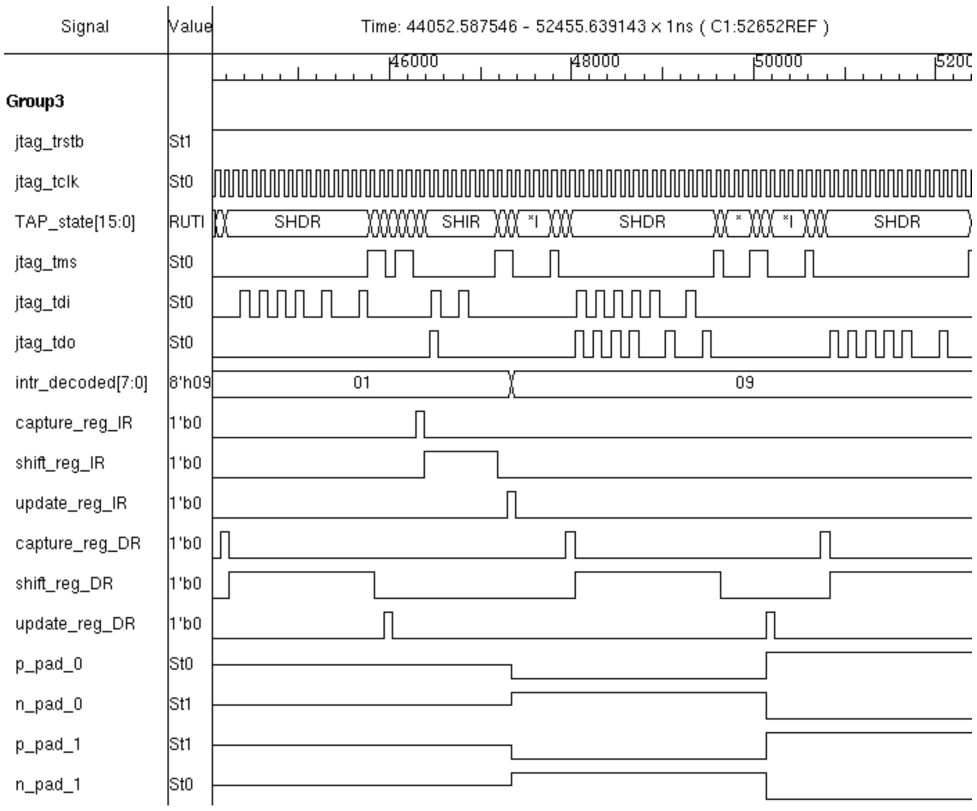


Figure 5.3.1: Waveform of extest test

5.4 Extest train test

Extest_train test is verified by placing the boundary scan register between TDI and TDO. Instruction register is loaded with PRELOAD instruction of opcode “01”. After the boundary scan chain is loaded with even pattern by shifting bits through TDI. Now the instruction register is loaded with EXTEST_TRAIN instruction bits of opcode “0F” which allows output pins to be in control with the boundary scan cell. After Update-IR output pins are driven with the loaded cell data from the predefined values and the output pins are compared with the expected values. AC EXTEST instruction is selected when AC test mode signal goes 1. In the RunTest/Idle state when AC test signal is toggled at negative edge of TCK, the driver output will toggle and driven with inverted data. Boundary scan chain is loaded with an odd pattern (odd pins set to 1) through TDI and boundary scan captured values from the previous even pattern are shifted through TDO and compared with the expected value. Figure 5.4.1 shows the simulated response of the test covering all the above mentioned scenario.

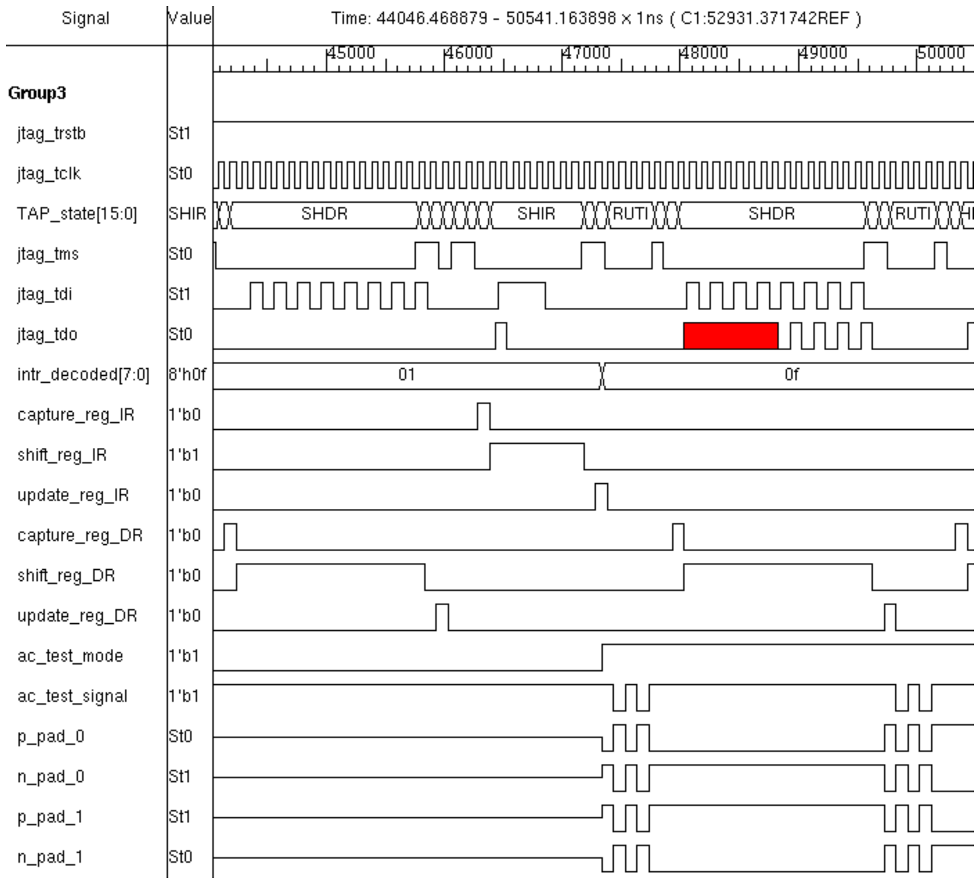


Figure 5.4.1: Waveform of extest_train test

5.5 Failing scenario

Figure 5.5.1 shows one of the failing scenario for the idcode test. Here firstly TAP is in reset state so the default decoded instruction is Idcode. Instruction register is shifted with the idcode instruction bits. In the Capture-DR state, idcode register gets captured with the 32 bit Idcode register value described in the BSDL file as per the design. This captured value gets shifted in the Shift-DR state. The values shifted in the failing case as shown in figure 5.5.1 is all ones which is incorrect. The decoded instruction was not correct so the value of idcode register was not captured and hence resulted into miscompares.

Good Data Stream: 1000 0100 0001 0000 1000 0000 0001 0011

Bad Data Stream: 1111 1111 1111 1111 1111 1111 1111 1111

The idcode instruction was mentioned “02” in the BSDL file, so when SVF file got generated

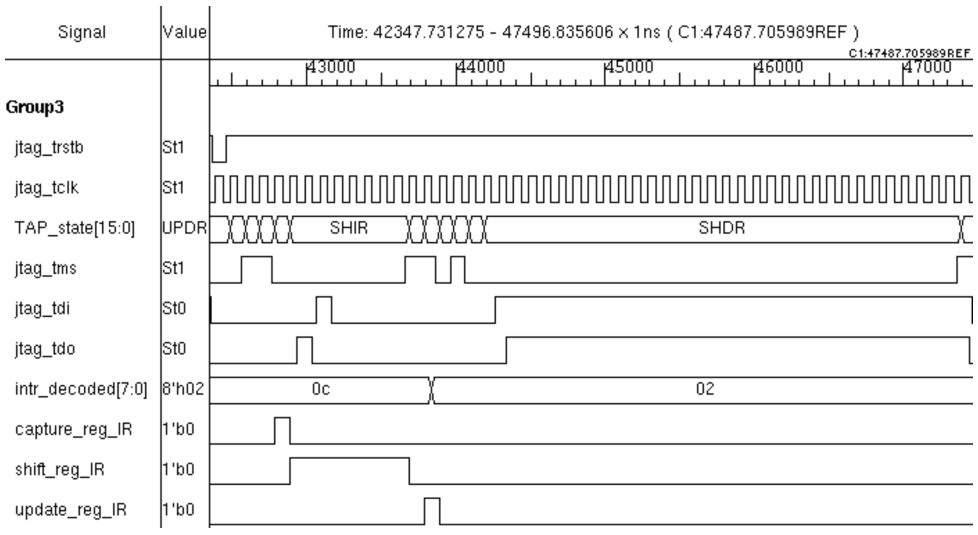


Figure 5.5.1: Waveform of idcode test

with this SIR value. Due to error in the BSDL file wrong instruction got decoded and failed the test. So the BSDL file of the IP should be perfectly coded otherwise it could cause an error in later stages also.

Thus, various simulation results for different tests like bypass register test, test logic reset test, extest test, extest_train test were discussed.

Chapter 6

Conclusion

Boundary scan solves the problem of testing interconnects between IC's and this DFT is found in almost all IP and SOC being manufactured nowadays. This project presents the methodology for verification of boundary scan. With minimal input required for the test pattern generation the flow provides faster test content generation. No functional data is being used for the verification of boundary scan. BSDL file generates test patterns of the required test selected based on the information provided in the BSDL file. The generated test patterns are validated by integrating in reusable verification environment. The reusability feature of OVM testbench environment allows to use different test based on the user provided input. Faster time to market can be achieved by using this flow of boundary scan verification. Boundary scan architecture and thus BSDL verified at IP level allows seamless verification of boundary scan at SOC level.

References

[1] “IEEE Standard Test Access Port and Boundary Scan Architecture,” in IEEE Std 1149.1-2001 , vol., no., pp.1-212, 23 July 2001 doi: 10.1109/IEEESTD.2001.92950

[2] B. Eklow , K. P. Parker and C. F. Barnhart, “IEEE 1149.6: a boundary scan standard for advanced digital networks,” in IEEE Design Test of Computers, vol. 20, no. 5, pp. 76 83, Sept. Oct. 2003. doi : 10.1109/MDT.2003.1232259

[3] Kenneth P. Parker. The Boundary Scan Handbook(4th Edition). Switzerland: Springer International Publishing

[4] C. Elakkiya, N. S. Murty, C. Babu and G. Jalan, “Functional Coverage - Driven UVM Based JTAG Verification,” 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Coimbatore, 2017, pp. 1-7. doi: 10.1109/ICCIC.2017.8524556

[5] Y. Zhang, Y. Wang and F. Huang, “A Platform with JTAG Debugging of SoC Based on System Level Verification,” 2019 IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC), Xi’an, China, 2019, pp. 1-2. doi: 10.1109/EDSSC.2019.8754419

[6] N. Arya and A. P. Singh, “IEEE 1149.1 test acess port (JTAG) verification using verilog simulation,” 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, 2016, pp. 106-111. doi: 10.1109/ICEEOT.2016.7754838

[7] Tessent® BoundaryScan Users Manual

[8] Mentor Support Website: `https://support.mentor.com/`