Boundary Scan Validation For SoC

Major Project Report

Submitted on Partial fulfillment of the requirements

for the degree of

Master of Technology

in

Electronics & Communication Engineering

(Embedded Systems)

By

Anshu Shah

(18MECE02)

Electronics & Communication Engineering Department

Institute of Technology

Nirma University

Ahmedabad-382 481

May-2020

Boundary Scan Validation For SoC

Major Project Report

Submitted on partial fulfillment of the requirements

for the degree of

Master of Technology

in

Electronics & Communication Engineering

By

Anshu Shah
(18MECE02)

Under the guidance of

External Project Guide:

Mr. Umapathi Nadendla

DFT Logic Designer

INTEL Technologies India Pvt Ltd,

Ahmedabad.

Internal Project Guide:

Dr. Akash Mecwan

Assistant Professor, EC Department,

Institute of Technology,

Nirma University, Ahmedabad.

Electronics & Communication Engineering Department

Institute of Technology-Nirma University

Ahmedabad-382 481

Declaration

This is to certify that

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

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

- Anshu Shah

18MECE02

Disclaimer

"The content of this thesis does not represent the technology,opinions,beliefs, or positions of INTEL Technologies India Pvt Ltd ,its employees,vendors, customers, or associates."

Certificate

This is to certify that the Major Project entitled **"Boundary Scan Validation For SOC"** submitted by **Anshu Shah (18MECE02)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in Embedded Systems, 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.The results embodied in this major project, to the best of our knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Date: Place: Ahmedabad

Dr. Akash Mecwan
Internal Guide

Dr. N.P.Gajjar
Program Coordinator

Dr. Dhaval Pujara
Head of the Department, EC

Director
Institute of Technology

Certificate

This is to certify that the Major Project entitled **"Boundary Scan Validation For SOC"** submitted by **Anshu Shah (18MECE02)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in Embedded Systems, 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.

Mr. Umapathi Nadendla
DFT Logic Designer
INTEL Technologies India Pvt Ltd
Bangalore

Acknowledgements

I would like to express my gratitude and sincere thanks to **Dr. N.P.Gajjar**, PG Coordinator of M.Tech Embedded Systems and **Dr. Dhaval Pujra** for guidelines during the review process.

I take this opporhutunity to express my profound gratitude and deep regards to **Dr. Akash Mecwan**, guide of my internship project for his exemplary guidance, monitoring and constant encouragement.

I would also like to thank **Mr. Umapathi Nadendla and Mr. Pradeep V**, external guide of my internship project from **INTEL Technologies India Pvt Ltd**, for guidance, monitoring and encouragement regarding the project.

- Anshu Shah

18MECE02

Company Profile

Intel is one of the worlds largest and highest valued semiconductor chip makers based on revenue, and is the inventor of the x86 series of microprocessors, the processors found in most personal computers (PCs). Intel supplies processors for computer system manufacturers such as Apple, Lenovo, HP, and Dell. Intel also manufactures motherboard chipsets, network interface controllers and integrated circuits, flash memory, graphics chips, embedded processors and other devices related to communications and computing.

Contents

List of Figures

Abstract

As printed circuit boards (PCBs) has become more complex and denser, the need for exhaustive testing becomes extensively important. The IEEE 1149.1 Boundary Scan Architecture provides access to internal module network required for interconnect testing in situations where physical access is sparse or not possible. As the advantage of boundary scan was becoming recognized in many commercial applications, it became apparent that a universal definition was needed to achieve the economies of using as industry standard. This led to the 1990 approval of IEEE Standard 1149.1. It gives standardized approach to testing interconnection between integrated circuit, testing integrated circuit itself and observing and modifying circuit activity during component's normal operation. Boundary Scan architecture tests pin connectivity without using physical test probes and captures functional data while a device is functioning normally. A set of test cases is defined, such that the component can respond to a set of instructions designed to support testing.

Typical Boundary Scan testing involves manually generating bsdl file and test cases and then validating the model.This report describes the limitation of this method and suggests an alternate, automated ,efficient and faster approach to Boundary Scan testing validation.

Chapter 1

Introduction

1.1 Motivation

In ancient times, when we used to have few pins on chip,testing the chip in terms of pin connection was very easy. We just used multi-meter to check connectivity of the any two pins. Testing connectivity for different IPs coming from different vendors also came handy.But now the pin count on the board is increasing rapidly day by day. So traditional methods used before for testing the chip cannot be used any longer.

In order to make this testing process less tedious some alternate method is needed. As the logic is becoming denser on the chip ,there is need for universal standard in order to have an established approach for testing this boards. It will help in reducing the testing time and increase accuracy of the testing. This will also give a common platform to different vendors to validate their boards using same standard. This will finally help in reducing time to market for company's product.

1.2 Objectives

Objectives of the project are as follows -

- To verify the input and output ports without actually affecting and interrupting the functionality of the board.
- To reduce the testing time by implementing a standard across the industry.
- To detect faults early in chip development production cycle.
- To make post-production testing of physical device easier.

1.3 Tools and Methodology Adopted

The tools and methodology used for development and implementation of this project is stated as below:

- Understanding of Boundary Scan architecture.
- Familiarizing with the Boundary Scan validation flow.
- Basic knowledge of OVM(Open Verification Methodology) and UVM(Universal Verification Methodology).
- Acquaintance of Verilog or System Verilog.
- Hands-On in Perl Scripting
- VCS Compiler for running the tests.
- Verdi tool for debugging the tests.

1.4 Scope of Work

The main objective is to analyze and implement the boundary scan on chip.Also, find the effective alternate way to implement the same and reduce the testing time, human labour and try to generate a more streamlined flow for this feature.

1.5 Thesis Organization

The first part of this chapter covered motivation, objective and Tools and Methodology adopted. Chapters 2 explains the Boundary Scan Architecture, its basic instructions and BSDL language used. Chapter 3 describes how the Boundary scan flow works and test environment for it. Chapter 4 discusses limitations of the flows in Chapter 3 and alternate method for it. Chapter 5 compares result of both the flows and challenges faced.Chapter 6 contains conclusion and future scope.

1.6 Time Line

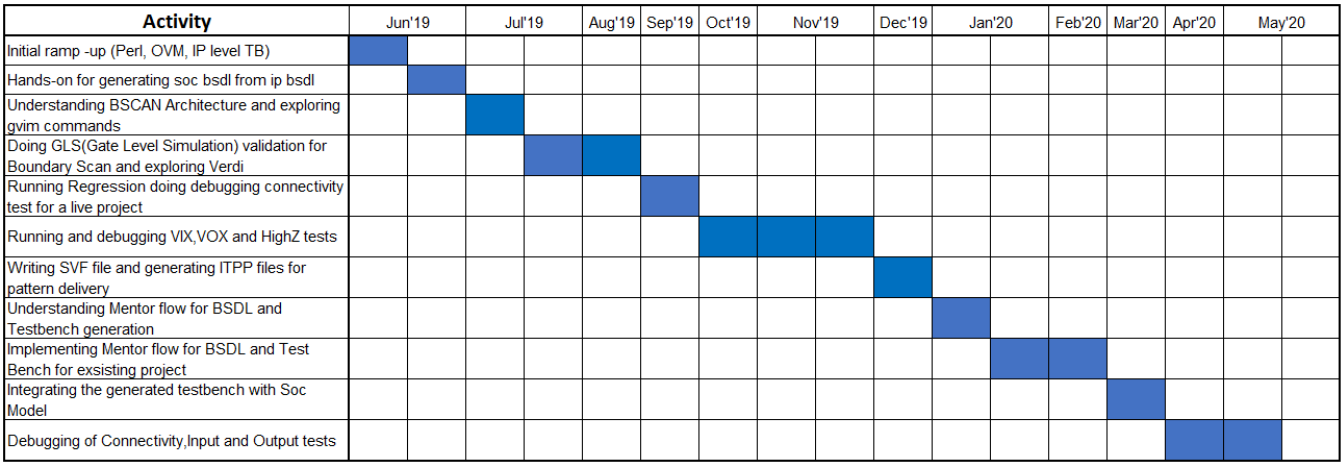

Figure 1.1: Time Line

Chapter 2

Literature Survey

2.1 Introduction

As traditional testing ways like in-circuit testers and bed of nail fixtures were inept due to increasing complexity and density of the pins on the board and so boundary scan was introduced. Boundary Scan(Bscan) commonly referred as JTAG is defined by IEEE standard 1149.1. It is the method for testing interconnects on printed circuit boards implemented at chip level. It can give the information of the board in terms of pin values, where it is not possible to have access to the nodes. Bscan also supports observing or modifying circuit activity without interrupting the component's normal function.

It has become popular for board level manufacturing test applications. As it is difficult to check all the pins individually for dense and complex PCBs, this standards will align all the pins in one chain by testing.It enables most of the boards to be tested just by use of minimal access. Thus, dramatically reducing development and production costs, speeding test development through automation and improving product quality.

This Chapter will further discuss the Bscan architecture[1] and its basic JTAG pins. It will than describe the individual entities in the architecture like TAP controller,

instruction register and boundary register. Moving further few mandatory boundary scan instructions will be discussed followed by Boundary Scan Description Language.

2.2 BSCAN Architecture

The basic architecture of 1149.1 Boundary-Scan is incorporated at the Integrated Circuit level. It consists of four basic pins, 3 input pins and one output and an optional pin, which are dedicated to Boundary-Scan. These pins form the Test Access Port (TAP) and are not shared for any other function. These pins are used with a simple protocol to communicate with on-chip Boundary-Scan logic. For device to work in testing mode the Test Mode Select pin should be high and for chip to work in functional mode it should be low.

Pin Name	Description
TDI(Test Data Input)	Serial test instructions and data are received at TDI
TDO(Test Data Output)	Serial output for test instructions and data from the test logic
TCK(Test Clock)	Clock for the test logic defined by this standard
TMS(Test Mode Select)	Signal received here is decoded by the TAP to control test operations
TRST(Test Reset-optional)	Provides asynchronous initialization of the TAP

Figure 2.1: Basic JTAG Pins

Figure 2.2 is the simplified architecture of the 1149.1 compliant IC. It consists of TAP controller, Instruction Register (IR), Bypass register, Device ID Register and four basic pins. The protocol is driven by TMS and TCK. TDI and TDO are used for serially shifting the data. Instruction Register and decode logic defines the mode in which data registers will function. Bypass Register is used to bypass the module. Bypass Register and the Boundary Register are mandatory rest can be optional. Device ID Register is used for identification of the device and make sure that the correct device is selected.

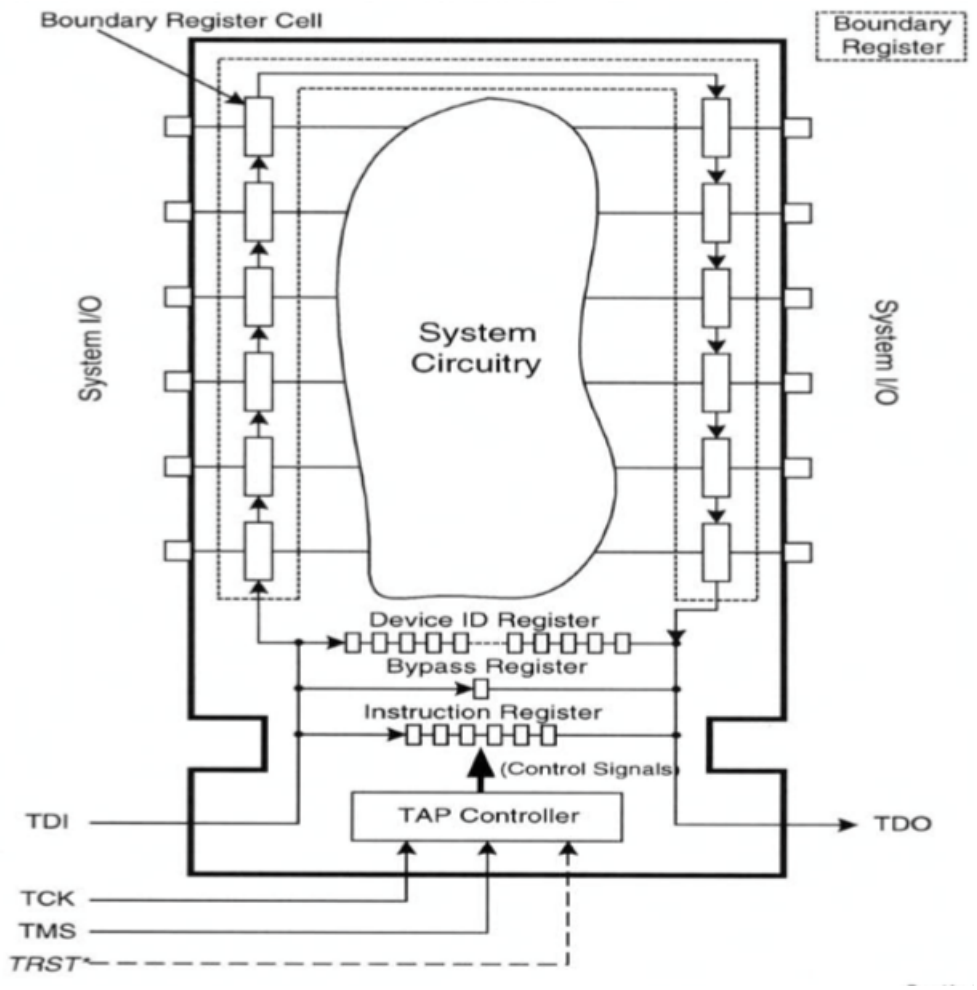

Figure 2.2: BSCAN architecture[1]

2.2.1 TAP Controller

TAP controller will control the test instruction and associated data that will be fed and result that will be read out. The TAP controller[2] is a finite state machine with a state containing sixteen states. A transition between states will occur on a rising edge of the TCK that is test clock. It can also occur asynchronously when test reset is made high,if at all test reset is present.

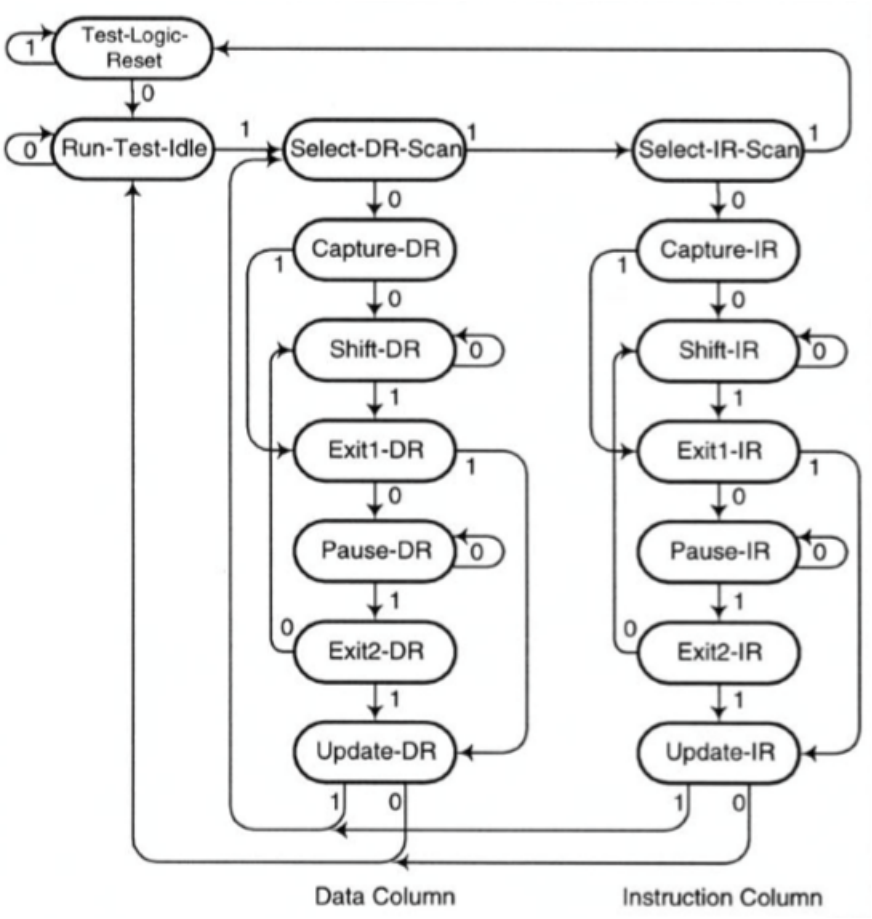

Figure 2.3: TAP Controller FSM [1]

Each state in FSM is described further along with its significance. TEST-LOGIC-RESET state is the initial state to start from. Irrespective of which ever its present state is, holding TMS high for continuous five cycle of TCK, will bring the state of TAP to this state. Now it is ready to take instruction to the register. RUN-TEST/IDLE will wait for the input from TMS and will go to Instruction Cycle or Data Cycle according to the given TMS value. The state SELECT-DR-SCAN shows the start of Data cycle. Next state SELECT-IR-SCAN shows the start of Instruction cycle. CAPTURE-DR instruction than captures the parallel data to data register. CAPTURE-IR state captures the instruction to instruction register. Another state SHIFT-DR basically shifts the data from data register serially. SHIFT-IR state will

shift the instruction from instruction register. UPDATE-DR updates the value of the data register in one cycle. UPDATE-IR then update the instruction register with new value. The basic test algorithm describes how the data from TDI is controlled through the TAP Controller and hoe the data actually flows between TDI and TDO -

Step 1: Initialize the TAP to TEST-LOGIC-RESET.

Step 2: Load the Instruction Register with the PRELOAD instruction. This will set the connection between the Boundary Register between TDI and TDO, but still will not give pin permissions.

Step 3: Shift the first data pattern into the Boundary Register. This is called the "preload" stage of the algorithm.

Step 4: Load the Instruction Register with EXTEST. This will set the Boundary Register between TDI and TDO and also give pin-permission after giving UPDATE-IR. This writes the first data pattern.

Step 5: Capture (read) the response pattern into the shift portion of the Boundary Register.

Step 6: Shift the captured response pattern out while shifting in the next stimulus pattern.

Step 7: Update (write) the next stimulus pattern.

Step 8: Is last stimulus pattern written ? If so, go to Step 9; otherwise go to step 5.

Step 9: Capture (read) the last response pattern.

Step 10: Shift in a "safe" stimulus pattern4 while shifting out the last captured response pattern.

Step 11: Update (write) the safe pattern.

Step 12: Go to TEST-LOGIC-RESET and halt the test.

2.2.2 Instruction Register

Any instruction through TAP goes to instruction register.It defines the mode in which the boundary scan cells will operate. So it also selects which test will be performed. Once the instruction is selected here than the data for the instruction is sent through TDI. Each instruction has a corresponding specific value which is given to the register.

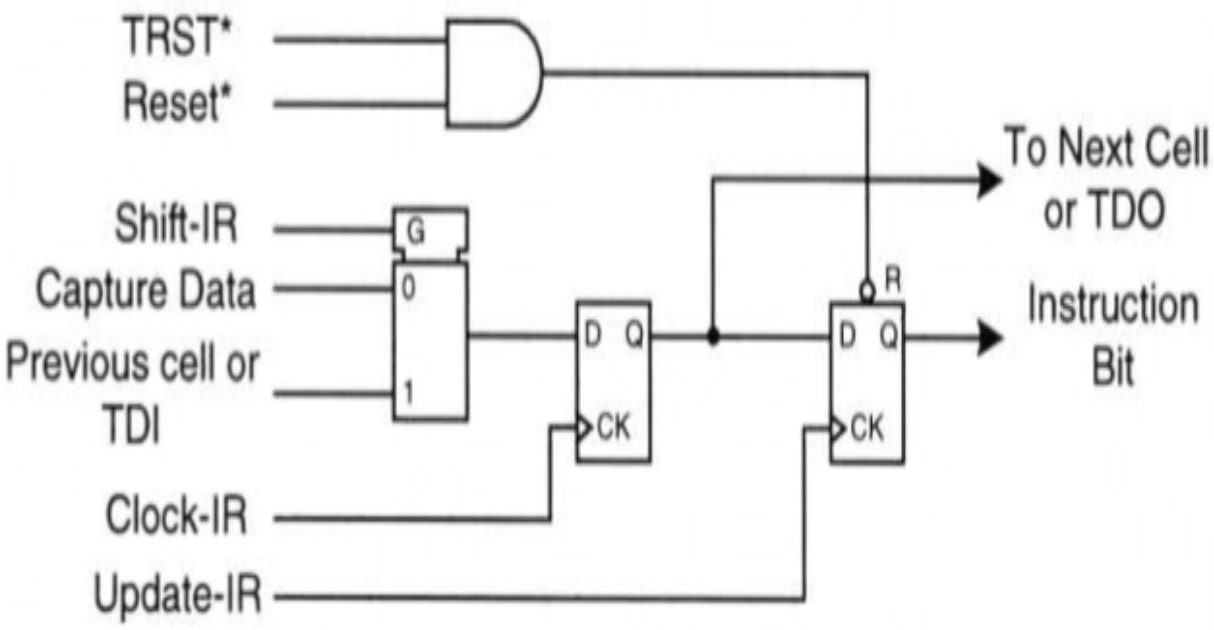

Figure 2.4: Instruction Register cell design [1]

2.2.3 Boundary Register

The register consists of capture (CAP) and update (UPD) flip-flops. The cell design shown in figure 2.5 is flexible enough to permit the cell to be used as an input or output cell. It has Shift-In and Shift-Out paths which will shift the data serially from one boundary scan cell to next cell. It also has one Parellel-In and Parellel-Out path which will transfer the data from external ports or circuit pins to internal logic. For example if the cell behaves as input then the data will be transferred from external pins to system circuitry.During shift operation it will just connect one boundary cell to another.

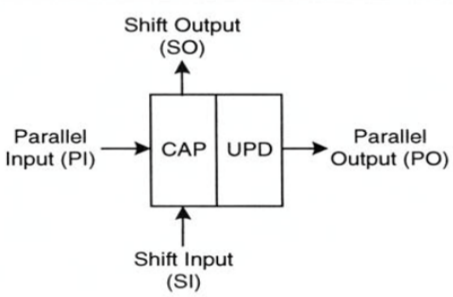

Figure 2.5: Typical Boundary Scan Cell [1]

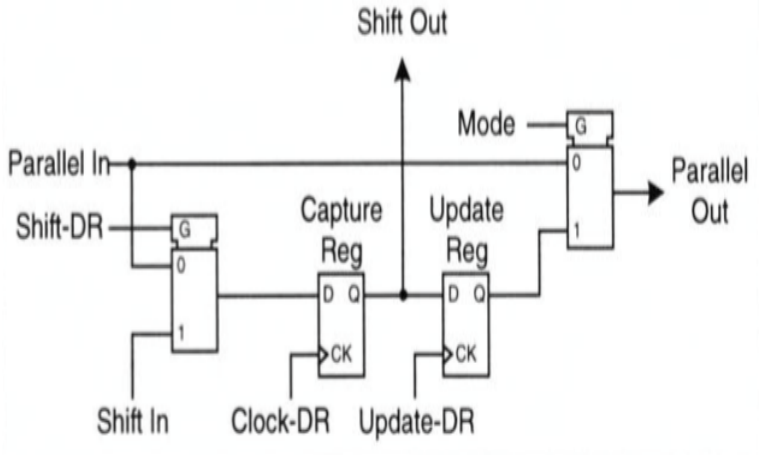

Figure 2.6: Detailed Boundary Scan Register [1]

2.3 Boundary Scan Instructions

There are certain boundary scan or IEEE1149.1 instruction standard used for testing. This instruction will help in setting the Instruction register to required value for performing the testing operation. This instructions are given to IP through TDI which further goes to TAP. Some of this instructions are mandatory for testing like- Sample/Preload, Extest, Bypass. There are some optional Instructions like- HighZ, IDCODE, UserCode, Clamp and Intest. All basic instruction are discussed individually in the following sections.

2.3.1 SAMPLE/PRELOAD

SAMPLE will basically used for observability of I/O pins that is it will select the Boundary scan cells and capture the values. This instruction can even function when chip is not in test mode. PRELOAD- controls system I/O pins meaning it will also select the Boundary scan cells but load some known values into the cells. This has 2 as tap instruction[3].

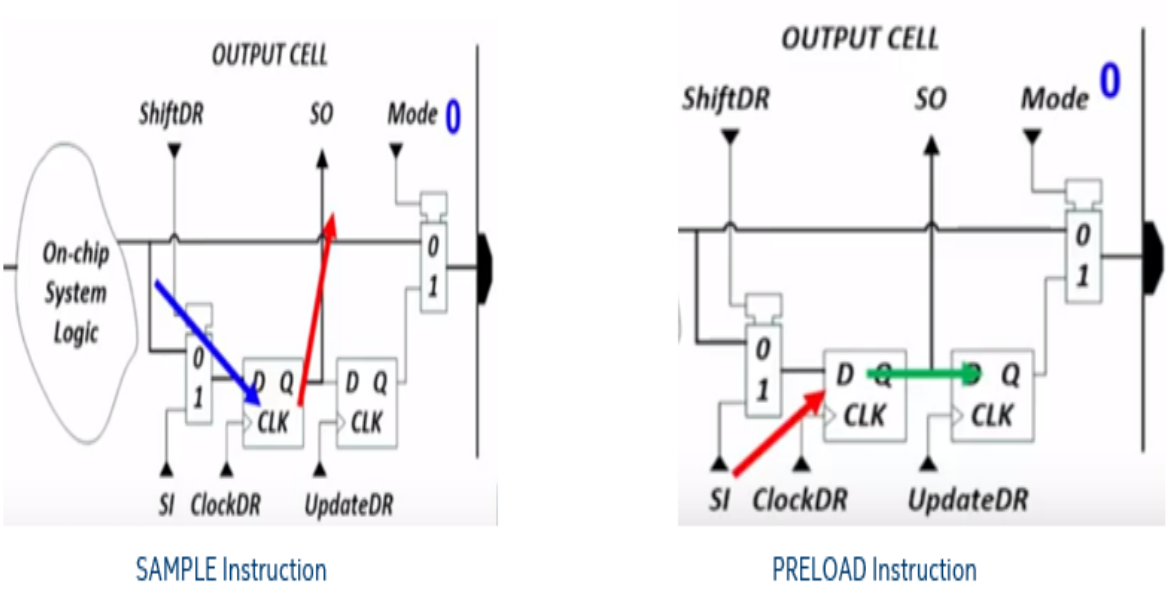

Figure 2.7: Sample/Preload Instruction [6]

2.3.2 EXTEST

This will place device in external boundary test mode. It will checks board level interconnects. This instruction will set the output pins to the value pushed by boundary scan cell. It is one of the mandatory instruction. Tap Instruction-8

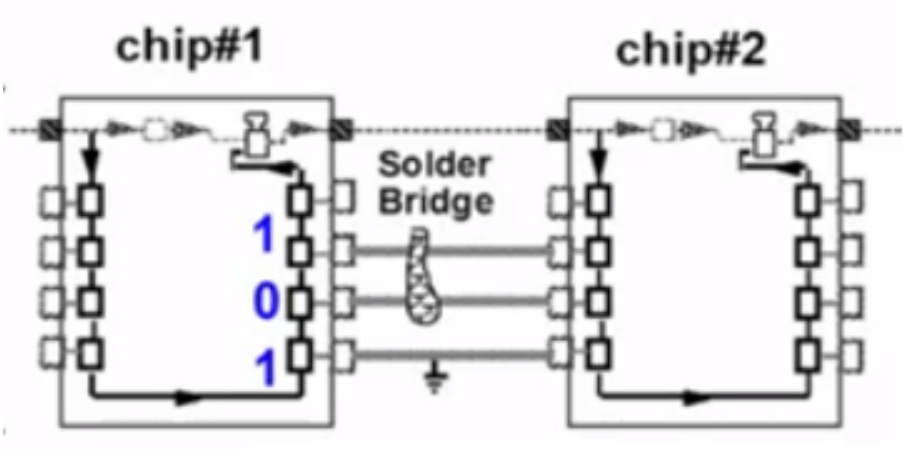

Figure 2.8: Extest example [6]

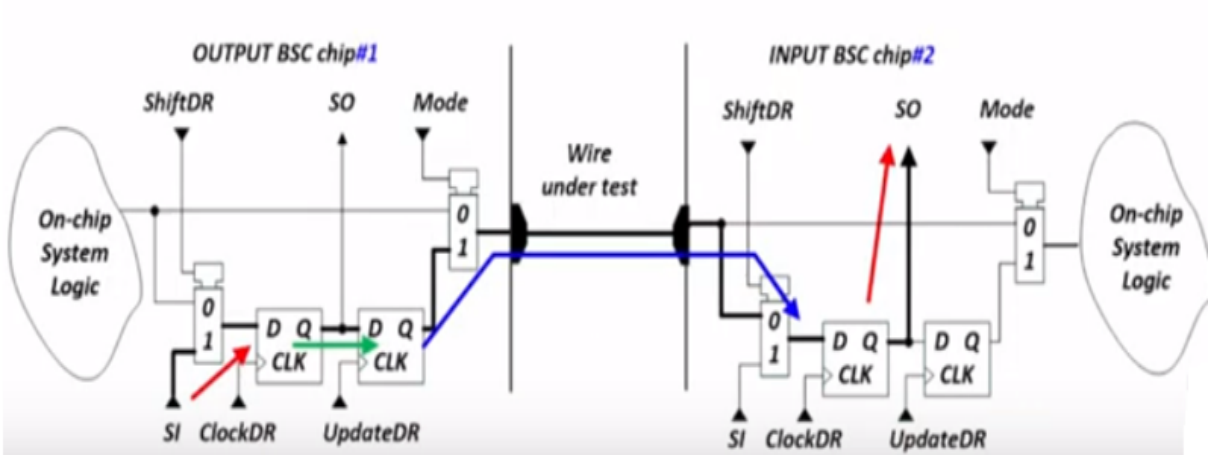

Figure 2.9: Extest Instruction [6]

2.3.3 BYPASS

In BYPASS instruction, the Bypass Register shortens the shift path within an IC to a single cell. It places single-bit BYPASS data register between TDI and TDO. This is useful for reducing shift time when testing other boundary-scan components on a board. It will save test time in following manner-if Bypass instruction is not present it has to go through chip IC56 =12 Cycles and while Bypassing chip IC56=9 Cycles. Tap Instruction-all 1's

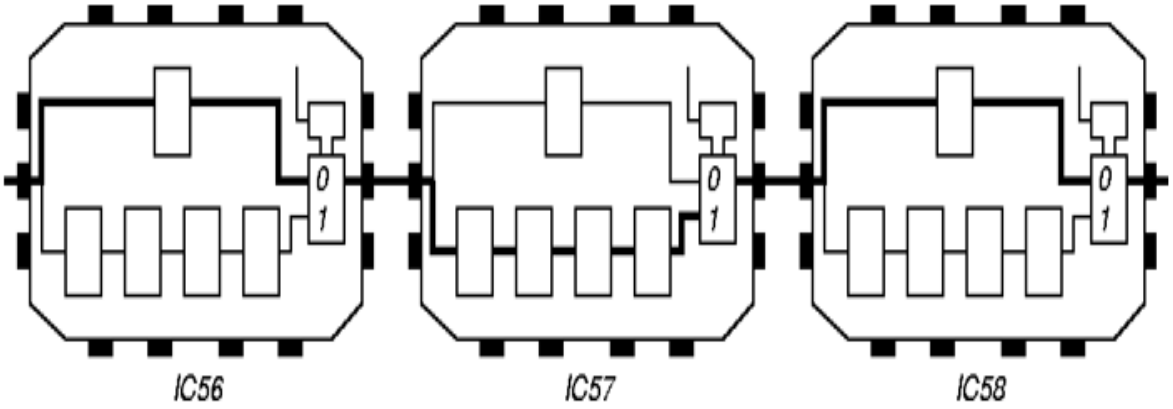

Figure 2.10: Bypass Instruction [2]

2.4 Boundary Scan Description Language(BSDL)

It is machine-readable language that allows complete description of testability features in components that comply with IEEE Std 1149.1.It acts as a bridge between vendors and companies that needs to exchange information about the design of test logic.BSDL file provides the complete information needed for about the top level ports, there description,function, safe value, disable value and also the boundary scan cell associated with it[4].

SOC level BSDL can be synthesized by IP level BSDLs. The figure 3.4 the example of how a bsdl file looks and what is contains. It is subset of VHDL and contains

entity component which defines all the ports external ports present along with there function. It consists of different attribute which describes the way in which the boundary scan cell will be applied. Pin map attribute will map the soc level pin to physical level pin. Attribute Instruction opcode contains opcode for all the instruction to be performed. Attribute Boundary register contains details of the inserted boundary scan netlist of for each cell.

In summary this chapter familiarised with Bscan architecture and steps for how TAP controls the flow of data from TDI to TDO through bscan instruction. It also described functionality and working of each instruction. Boundary Scan Description Language was covered further, telling how BSDL is inout file for generating the inserted netlist. Next chapter will cover the validation flow for performing Bscan test. It will also show how to validate the inserted bscan cells through chain connectivity test for a DUT. It than describes the validation environment used for testing the model. Finally it will discuss how this environment is used to validate the bscan cells through trational approach.

Chapter 3

Implementation and Traditional Approach

This chapter discusses the implementation part of the Boundary Scan and how it is tested. Bscan validation is performed using two approaches .This chapter will describe the first approach. But, before that it firsts discusses the bscan validation flow later describing the basic connectivity test performed for validating the bscan cells. It will further describe the OVM environment which is the key for Traditional approach. It will than further describe the Traditional approach.

3.1 BSCAN Validation Flow

Primarily, while performing Boundary Scan testing, custom flow is as follows

- Identify the model for which the Boundary Scan testing needs to be performed.
- Write the IP level BSDL(Boundary Scan Description Language) for each IPs
- Generate the soc level BSDL by combining all IP level BSDLs.
- Make the model scan-able by inserting boundary scan cells across each port. This model is now 1149.1 design circuit[5].

- Do testing for verifying the functionality of the boundary scan cells. Meaning verifying that the inserted netlist is correct or not.
- To verify bscan cells , test bench is generated first.
- The test environment is chosen to run test, in this case OVM environment is used.
- Than the basic test- chain connectivity test in ran and results are obtained.
- If the design is bscan insertion is proper than the design is ready for fabrication or else stitching of bscan cells needs to be modified.

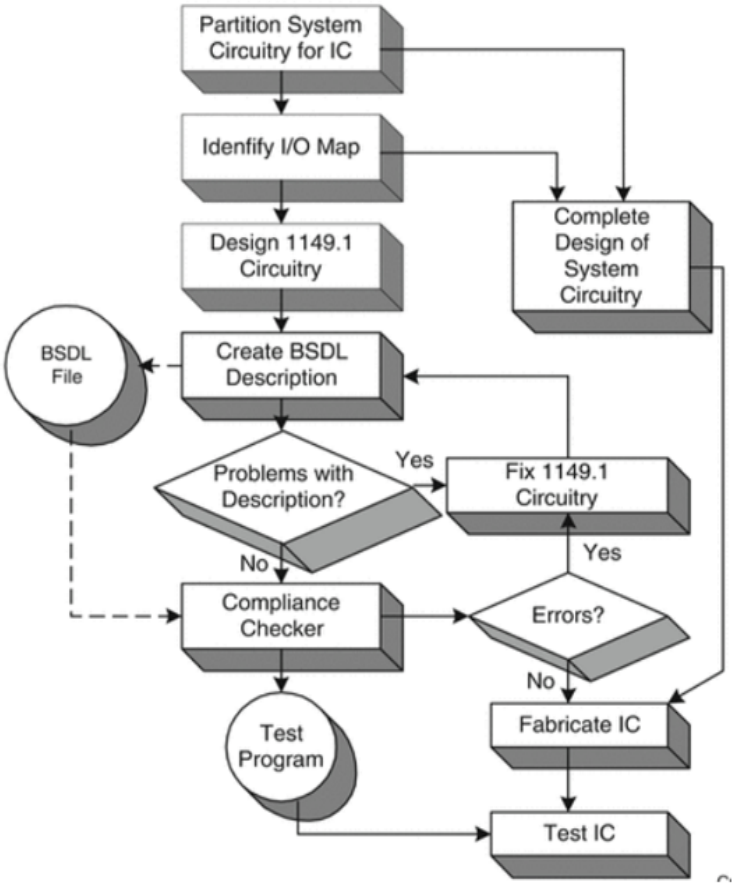

Figure 3.1: BSCAN validation Flow [1]

3.2 Chain Connectivity Test

The first and basic requirement after boundary scan insertion is to make sure that they are inserted properly and are functioning as required. The very basic test for this is called chain connectivity test.This is done by using the four basic pins of IJTAG standard.It ensures that the inserted boundary scan cells are connected properly to TDI, TDO and also to respective boundary cells[6].

This is done by serially shifting the input data from the TDI and reading the same data from TDO after certain cycles(which is equal to chain length).If the input and output data matched after the given clock cycle then the test is said to be passed or else chain connectivity is said to be broken.In this case usually one of the stitched boundary scan cells shift is having issue. TDO needs to be checked after the total number of chain length or boundary cells inserted cycles have been passed, as our given TDI must have reached the TDO after that cycle.

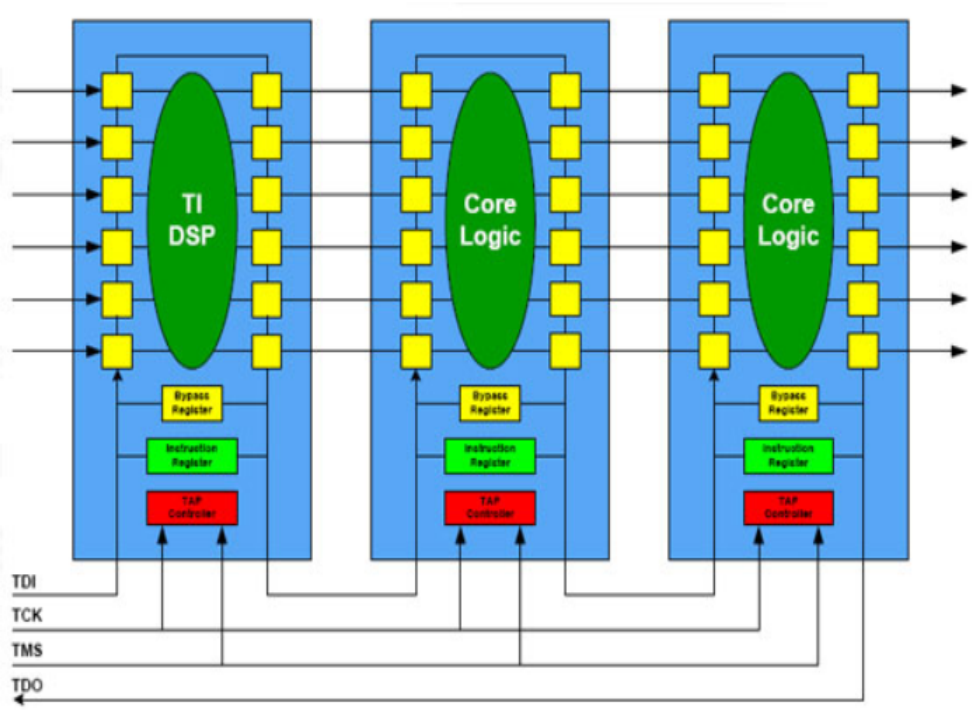

Figure 3.2: Chain connectivity Test [5]

3.2.1 Open Verification Methodology(OVM)

This chain connectivity test is manually written using system verilog(.sv) format. The validation of this test case with the model is done using OVM environment. The following figure shows the structure of verification environment for the chip. Test along with the design under test(dut) is called the top module. Test and dut are connected using an interface. Test consists of the validation environment. An environment can have multiple agents.
The connectivity test format is basically a sequence.Sequencer sends the transaction to driver and driver in turn will send the data to model(dut).The response from the dut is then monitored by monitor and also given to scoreboard that will compare the expected data with the actual data and give the results.

This verification methodology was chosen because it meets the verification requirements like it is reusable across abstraction layers and design (parameterization, generalization, minimize dependencies, well defined semantics) .It has Modular structure (localization of functionality, localization of data, communication through well defined interfaces).It uses standard interfaces (provide external view of object, hide implementation details, define interface semantics). Also supports stepwise refinement and abstraction (keep things at the highest level possible)
Can be implemented in SystemVerilog[7].

- Testbench Organization
  - The first part of defining a testbench architecture is to identify the key functions it must perform.
  - Then you can design and build components that fulfill each function.
  - Typically, testbenches do the following things: –
    - generate stimulus, either random or directed

        * convert the stimulus into pin activity on the DUT
        * watch output interfaces and intercept interesting responses
        * compare responses with some reference
        * count various kinds of responses, i.e. collect coverage information
        * control stimulus generation based on coverage (Optional)
        * Shuts off the environment when test intent is completed

- Category of Test-bench Components
    - The primary goal of the is to modularize the creation of the components
        * Transactors helps to move out of the signal level domain and into the transaction level domain.
        * This enables to design the rest of the testbench at the transaction level.
        * Makes it quicker to design, easier to maintain, and more reusable.
    - The components in the testbench can be divided into four categories[8]
        * Signal Level:- The DUT itself
        * Transactors:- Driver, Monitor
        * Transaction Level:- Agent, Sequencer, Scoreboard, Coverage
        * Intent Level:- Env, Test

- Testbench components
    - DUT
        * The Design Under Test.
        * This is a pin level component whose function is being verified by the testbench
    - Agent

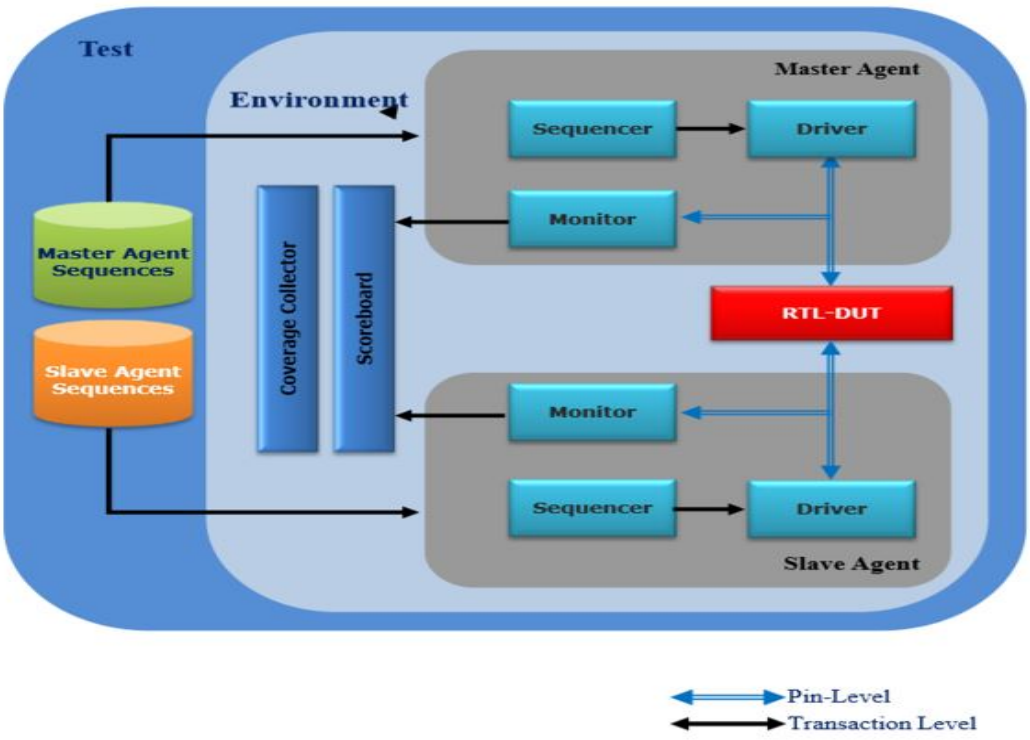

Figure 3.3: OVM testbench Architecture [7]

* Sequencer
  - The sequencer routes a sequence to the driver.
  - A sequence is a series of transactions.
* Driver
  - The driver converts the transaction level stimulus (one sequence item at a time) into pin activations on the DUT.
* Monitor
  - The monitor does the reverse function of the driver.
  - It watches pin activity and converts it to transactions. The monitor is passive, it does not drive any pins.

– Scoreboard

* A scoreboard tracks transaction level activity that is going in and

out of the DUT to ensure that DUT is functioning properly. For example, it may track packets in vs. packets out to see if all the packets sent into a communication device made it out intact.

- Coverage collector
  - A coverage collector has counters organized in bins.
  - It can tap into various observable points in the test-bench
  - It simply counts the transactions that are sent to it and puts the counts in the appropriate bins.
- Env
  - Instantiate all the agents, scoreboards, coverage collectors and connects them
  - Also contain the default configuration of the test-bench, standard end of test routine, initialization routine etc.
- Test
  - The test instantiate the Env
  - It can re-configure the Env based on test intent
  - It controls running of sequences on the available agents, can overwrite transaction constraints etc.

3.3 Traditional Approach- OVM environment

There were two approaches used for testing the model- Traditional Approach and Automated Approach, model in this case was a simple adder. Traditional approach is the one which is being used from long time for most of the projects at Intel.
Now the two most necessary thing required to validate the boundary scan is soc BSDL and test bench. OVM approach consists of manually writing the bsdl file for the model as in figure 3.4. Also the test bench for validating has to be written for the model along with creating the OVM validation environment. The testing environment methodology used here is OVM. Here for all the tests different verilog files are written. The build time for model is also more as the test environment becomes heavy due to ovm packages. After build the testing time for test is also long[9].

To conclude in this chapter execution part of the Boundary scan was covered. Also one of the two methods was discussed along with its validation environment and limitations. Next chapter will cover the alternate method to overcome the issues of OVM approach.

```
entity test_ip is
generic (PHYSICAL_PIN_MAP : string := "test_ip");
port(
  SOC_PAR1_D1       : in bit;
  SOC_PAR1_D2       : in bit;
  SOC_PAR1_Q3       : out bit;
  SOC_PAR1_CLK      : in bit;
  SOC_PAR1_RESET    : in bit;
  SOC_PAR1_SET      : in bit;
  JTAG_TDI          : in bit;
  JTAG_TMS          : in bit;
  JTAG_TDO          : out bit;
  JTAG_TCK          : in bit;
  JTAG_TRST         : in bit
   );

attribute PIN_MAP of test_ip : entity is PHYSICAL_PIN_MAP;

constant test_ip : PIN_MAP_STRING :=
" SOC_PAR1_D1      :  A0, " &
" SOC_PAR1_D2      :  A1, " &
" SOC_PAR1_Q3      :  A2, " &
" SOC_PAR1_CLK     :  A3, " &
" SOC_PAR1_RESET   :  A4, " &
" JTAG_TDI         :  T0, " &
" JTAG_TMS         :  T1, " &
" JTAG_TDO         :  T2, " &
" JTAG_TCK         :  T3, " &
" JTAG_TRST        :  T4";

attribute INSTRUCTION_OPCODE of test_ip : entity is
   "extest         (00000000)," &
   "sample         (00000010)," &
   "preload        (00000010)," &
   "idcode         (00001101)," &
   "clamp          (00001110)," &
   "highz          (00000110)," &
   "bypass         (11111111)" ;

attribute BOUNDARY_REGISTER of test_ip : entity is
-- num cell     port                  function        safe_value
"   0( bc_1   , SOC_PAR1_D1  ,        input          , X )," &
"   1( bc_2   , SOC1_PAR1_D2 ,        input          , X )," &
"   2( bc_3   , SOC_PAR1_Q3  ,        output         , X )";
```

Figure 3.4: BSDL file

Chapter 4

Automated Approach- Mentors' Tessent flow

This chapter will discuss the limitations of the OVM approach and also suggests an alternate method for it. It than discusses the Mentor Tessent Tool and how the Tool solves the challenge for former method. It finallay compares the two approaches.

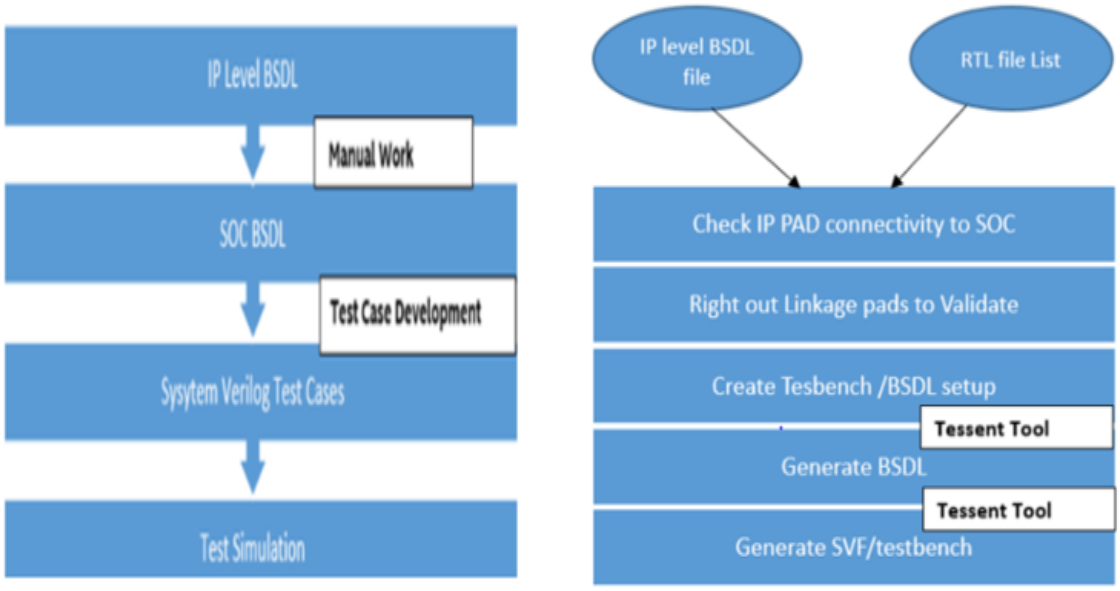

Figure 4.1: Traditional Approach Vs Automated Approach Flow

4.0.1 Tessent Tool

In this automated approach bsdl in generated through Mentors' Tessent flow and using the generated bsdl file test bench is also generated with the same flow. Here SoC pins and testing pins are directly connected to the test bench in turn reducing the test time and buils time significantly.

Tessent BoundaryScan automatically generates and integrates the RTL code for the TAP controller and boundary scan cells into the design RTL. It also generates the scripts required for logic synthesis, a BSDL description of the boundary scan functionality, simulation testbenches to verify the boundary scan implementation, and test patterns for manufacturing test. Tessent BoundaryScan provides a completely automated solution for adding standard boundary scan support to ICs of any size or complexity, reducing IC engineering development effort and improving time to-market.

The following image 4.2 shows the automatic generated bsdl and test bench using tool instead of writing manually. The tool requires IP level bsdl and pin map file as input file to generate soc level bsdl and testbench. BSDL generated using Tessent Tool for full adder is show in the following figure.

In this port section all the ports of the design are declared along with the TAP ports.That is the TAP ports which are included at the time of stitching boundary scan cells.

```
-- ********************************************************************************

-- BSDL file for design TOP
-- Generated by boundaryScanGenerate 2019.2
-- BSDL Version 2001

--   Designer:
--   Company:

--   Date: Mon Nov 25 21:07:58 2019

-- ********************************************************************************


 entity TOP is

-- This section identifies the default device package selected.

   generic (PHYSICAL_PIN_MAP: string:= "my_pack");

-- This section declares all the ports in the design.

   port (
         A          : in        bit;
         B          : in        bit;
         CIN        : in        bit;
         TCK        : in        bit;
         TDI        : in        bit;
         TEST_MODE  : in        bit;
         TMS        : in        bit;
         TRSTN      : in        bit;
         TDO        : out       bit;
         COUT       : buffer    bit;
         SUM        : buffer    bit
   );

   use STD_1149_1_2001.all;

   attribute COMPONENT_CONFORMANCE of TOP: entity is "STD_1149_1_2001";

   attribute PIN_MAP of TOP: entity is PHYSICAL_PIN_MAP;

-- This section specifies the pin map for each port. This information is
-- extracted from the port-to-pin map file that was read in using the
```

Figure 4.2: Declaring all the ports in the design

PIN MAP STRING section specifies the pin map for each port. Along with that,it also specifies the basic TAP ports. The next attribute defines boundary-scan instructions implemented in the design and their opcodes.This opcodes are universally accepted for Boundary Scan operation.

```
-- "read_pin_map" command.

     constant my_pack: PIN_MAP_STRING :=
        "A          : P1," &
        "B          : P2," &
        "CIN        : P3," &
        "TCK        : P7," &
        "TDI        : P4," &
        "TEST_MODE  : P11," &
        "TMS        : P8," &
        "TRSTN      : P5," &
        "TDO        : P6," &
        "COUT       : P10," &
        "SUM        : P9";

-- This section specifies the TAP ports. For the TAP TCK port, the parameters in
-- the brackets are:
--         First Field : Maximum  TCK frequency.
--         Second Field: Allowable states TCK may be stopped in.

   attribute TAP_SCAN_CLOCK of TCK  : signal is (10.0e6, BOTH);
   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_RESET of TRSTN: signal is true;

-- Specifies the number of bits in the instruction register.

   attribute INSTRUCTION_LENGTH of TOP: entity is 4;

-- Specifies the boundary-scan instructions implemented in the design and their
-- opcodes.

   attribute INSTRUCTION_OPCODE of TOP: entity is
     "BYPASS  (1111)," &
     "EXTEST  (0001)," &
     "SAMPLE  (0100)," &
     "PRELOAD (0100)," &
     "CLAMP   (0010)";

-- Specifies the bit pattern that is loaded into the instruction register when
-- the TAP controller passes through the Capture-IR state. The standard mandates
-- that the two LSBs must be "01". The remaining bits are design specific.
```

Figure 4.3: Specifying the pin map for each port

Section BOUDARY REGISTER section specifies characteristics of each cell in the boundary scan register from TDI to TDO. Each boundary scan register is mapped to the port and describes its function,safe value,disable value,control value and resulting state.

```
   attribute INSTRUCTION_CAPTURE of TOP: entity is "0001";

-- This section specifies the test data register placed between TDI and TDO for
-- each implemented instruction.

   attribute REGISTER_ACCESS of TOP: entity is
        "BYPASS    (BYPASS, CLAMP)," &
        "BOUNDARY (EXTEST, SAMPLE, PRELOAD)";

 -- Specifies the length of the boundary scan register.

   attribute BOUNDARY_LENGTH of TOP: entity is 6;

-- The following list specifies the characteristics of each cell in the boundary
-- scan register from TDI to TDO. The following is a description of the label
-- fields:
--       num      : Is the cell number.
--       cell     : Is the cell type as defined by the standard.
--       port     : Is the design port name. Control cells do not have a port
--                  name.
--       function: Is the function of the cell as defined by the standard. Is one
--                  of input, output2, output3, bidir, control or controlr.
--       safe     : Specifies the value that the BSR cell should be loaded with
--                  for safe operation when the software might otherwise choose a
--                  random value.
--       ccell    : The control cell number. Specifies the control cell that
--                  drives the output enable for this port.
--       disval   : Specifies the value that is loaded into the control cell to
--                  disable the output enable for the corresponding port.
--       rslt     : Resulting state. Shows the state of the driver when it is
--                  disabled.

   attribute BOUNDARY_REGISTER of TOP: entity is

--     num    cell    port        function      safe   [ccell  disval  rslt]
--
      "5     (BC_2,   A,          input,        X),                          " &
      "4     (BC_2,   B,          input,        X),                          " &
      "3     (BC_2,   CIN,        input,        X),                          " &
      "2     (BC_1,   SUM,        output2,      X),                          " &
      "1     (BC_1,   COUT,       output2,      X),                          " &
      "0     (BC_2,   TEST_MODE, input,         X)                           ";

 end TOP;
```

Figure 4.4: Specifying characteristics of each cell in the boundary scan register

4.1 Traditional Approach Vs Automated Approach

First thing is manual BSDL writing, all the IP level and soc level bsdls are written manually which is time-consuming and susceptible to human errors.Than is manual test cases writing where each test case is written manually which can often be inaccurate and less efficient leading to multiple iterations.For Non BSCAN insertion/Validation Flow there is no proper and generic bscan validation flow for insertion of boundary scan cells.

To overcome the above mentioned challenges an alternate method is proposed which aims at overcoming them and developing a method to ease the testing process.Tool will automatically generate soc level BSDL file and will create BSCAN test bench independent of project.It also creates a lighter test bench environment as it directly connects with the top level pins.The figure summarizes how the Tessent flow mitigates the challenges caused due to Traditional approach.

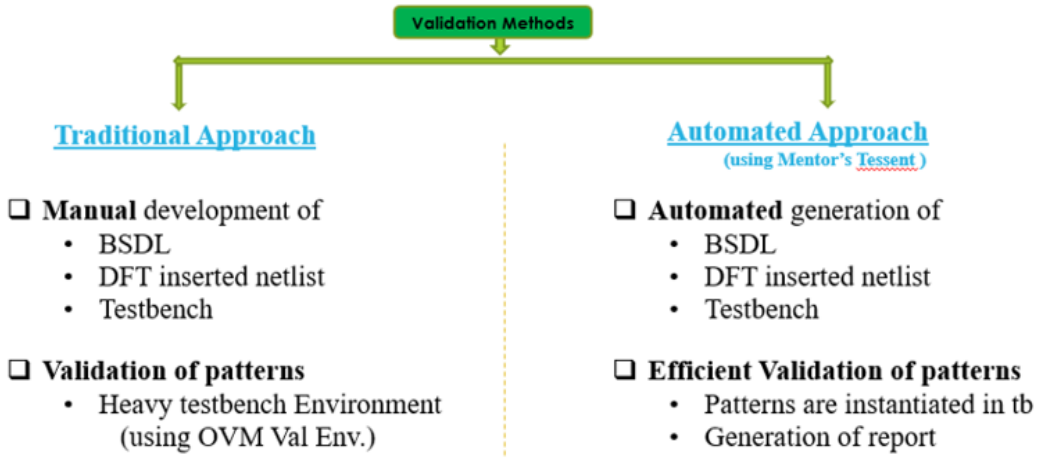

Figure 4.5: Challenges and solutions to mitigate Traditional Approach

To summarize this chapter covered drawbacks of ovm approach and discussed how it was resolved in the alternate approach. It also compares the two approach and establishes a significance difference between both.New chapter will quantify the results and compare them. It will also discuss the challenges faced while doing the validation flow in the project.

Chapter 5

Result

Same model was implemented using both the methods and few observations were made for Traditional and Automated approach. Objective was to reduce the test time and testing process easier. So two techniques were tried and the results obtained were quantified as shown in figure 5.1. Tessent tool has thus reduced the manual effort and decreased time to market in following way-

Feature	OVM Approach	Tessent Approach
Model Build Time	16 hours	4 hours
Test Run Time	8 hours	30 mins
overall test validation time	2 months	1 month

Figure 5.1: Results

Feature like Model build time ,test run time and the overall validation time were compared for both techniques.

- For Model build time was reduced by one-fourth for Tessent Approach, thus

significantly giving more window for validating and debugging. This is because the heavier OVM testbench is replaced with lighter testbench. Previously, dut was connected through agent and the data was sent. But here soc pins were directly connected to dut thus reducing the build time.

- For Test run time also there is steep time drop, which gives window for doing more experiments so to validate the model thoroughly.
- Tessent approach has thus reduced the overall time for validation to nearly half ,thus reducing time to market and meeting the objective.

5.1 Challenges

There were several issues while validating the model. Few are listed below -

- The generation of testbench with flow is greatest challenge as Mentor has different nomenclature for common name tests.
- Connectivity test was failing due to improper pin order in BSDL file.
- Some IPs like USB come from vendor with BSCAN cells already stitched and worked on opposite edge as compared to rest in-house stitched IPs. This caused TDO mismatch. To fix this one extra retiming flop was added which added half cycle delay.
- For INPUT test TDO mismatch was found. The issue was for the Data cell of failing pin was the mux was not tied to fromPad wire. Other challenge was one switch was added my functional team which caused power signal to be zero ,due to which TDO =x
- In OUTPUT test few bidirectional pins were not driven to correct value as enable for those pins was not set.
- Integrating the testbench with the model was also challenging.

- The test bench generates same test in group of 100 pins.For example total length is 300 and groups is of 100 pins than same test will run for 3 time.This issue was resolved integrating all the pins in one group and then generating the test.This reduced the test time significantly.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion and Future Scope

In this project, Boundary Scan testing flow was studied and implemented. From generation of bsdl to validation of the test case was implemented and analysed thoroughly.The two different methods were understood and implemented on live project rigorously.

Tessent flow resolves the issues of unforced human error and cumbersome test bench by more lighter test bench environment where SoC pin is directly connected to test bench ports and can be directly handled. The issue of manually genrating bsdl and test case was overcomed using Tessent tool.This approach enhanced the accuracy of generated bsdls.It is more seamless approach for validation.Fully plug and play compatibility is achieved.

This diminishes the need to regenerate the testcases and makes it reusable across the projects. The generated BSDL file is at par with the requirement thus reducing the manual effort. This approach is highly time efficient, labour-saving and minimizes unforced human errors, ultimately reducing time to market.

6.2 Future Scope

- Automated test bench has all the tests embedded in one, so a method to run individual tests is required. This will give the flexibility to choose the test depending on the requirement.
- Generating bypass bsdl for each individual IP validation. This will help in later stage for initial level debug.

References

[1] K. P. Parker, *Boundary Scan Handbook,Second Edition.* Kluwer Academic Publishers,eBook ISBN-0-306-47656-8, 2002.

[2] "IEEE standard test access port and boundary-scan architecture (IEEE Std 1149.1-2001)."

[3] "Jtag instruction." `https://www.youtube.com/watch?v=XEN01h9qkC4&list=PLvd8d-SyI7hjk_Ci0zpTqImAtpEjdK5JF&index=54`. [Online; accessed 25 November 2019].

[4] "Bscan cell description." `http://www.europractice.stfc.ac.uk/vendors/mg_boundaryscan-ds.pdf`. [Online; accessed 20 November 2019.].

[5] X. D. Xie, P. Li, and A. W. Ruan, "Design and implementation of boundary-scan circuit for FPGA," in *Testing and Diagnosis, ICTD, IEEE Circuits and Systems International Conference.*

[6] "Validation flow." `https://rd.springer.com/chapter/10.1007`. [Online; accessed 25 November 2019].

[7] "Basics of System Verilog and OVM." `https://www.doulos.com/knowhow/sysverilog/ovm/`.

[8] "OVM testbench." `http://www.testbench.in/OT_00_INDEX.html`. [Online; accessed 9 September 2019.].

[9] "Verification academy video lecture on OVM basics." `https://verificationacademy.com/courses/basic-ovm`.