Pre-Silicon Verification of Design For Debug Logic in SoC

Major Project Report

Submitted in partial fulfillment of the requirements
For the degree of

Master of Technology
In
Electronics & Communication Engineering
(VLSI Design)
By

Vanisha Shukla
(16MECV28)

Electronics & Communication Engineering Department
Institute of Technology
Nirma University
Ahmedabad - 382 481
May, 2018

Pre-Silicon Verification of Design For Debug Logic in SoC

Major Project Report

Submitted in partial fulfillment of the requirements
For the degree of

Master of Technology
In
Electronics & Communication Engineering
(VLSI Design)
By

Vanisha Shukla
(16MECV28)

External Project Guide:
Mr. Shabbir Topiwala
Enigineering Manager
Intel Technology India
Bangalore

Internal Project Guide:
Prof. Akasha Mecwan
Institute of Technology
Nirma University
Ahmedabad

Electronics & Communication Engineering Department
Institute of Technology
Nirma University
Ahmedabad - 382 481
May, 2018

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.

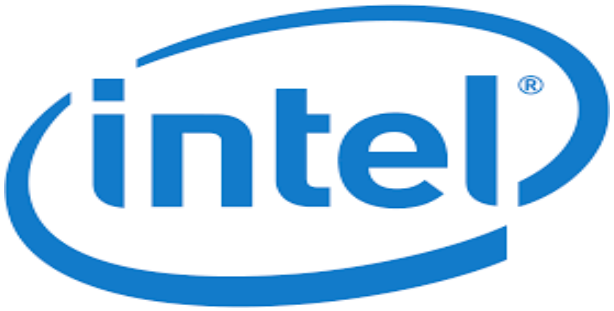
2. Due acknowledgment has been made in the text to all other material used.

Vanisha Shukla

Certificate

This is to certify that the Major Project entitled **"Pre-Silicon Verification of Design For Debug Logic in SoCs"** submitted by **Vanisha Shukla (16MECV28)**, 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.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.

Prof. Akash Mecwan	Prof. Dr N. M. Devashrayee
Internal Guide	PG Coordinator (VLSI Design)
Dr D. K. Kothari	Dr Alka Mahajan
Head, EC Dept.	Director, IT - NU
Date :	Place : Ahmedabad



Certificate

This is to certify that the Project entitled "**Pre-Silicon Verification of Design For Debug Logic in SoC**" submitted by **Vanisha Shukla (16MECV28)**, towards the submission of the Project for 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.

(External Guide)
Mr.Shabbir Topiwala
Engineering Manager
Intel Technology India
Bangalore

(Mentor)
Mr. Pradeep Hanji
Pre-Si Verification Engineer
Intel Technology India
Bangalore

Company Seal
Intel Technology India Pvt. Ltd.(Bangalore)

Date :

Place : Bangalore

Acknowledgment

Foremost, I would like to express my sincere gratitude to Mr. Shabbir Topiwala (Project Guide and Manager) and Mr.Pradeep Hanji (Mentor) for inspiration and professional guidance in this project.I would also like to thank Dr. N. M. Devashrayee (PG Coordinator, VLSI Design, ITNU) for providing the opportunity to carry out final year project as Internship at Intel Technology India Pvt. Ltd.

Next, I would like to thank my internal guide, Prof. Akash Mecwan for his valuable advice and continuous support throughout the project work.His guidance immensely helped me to carry out the work in a systematic and appropriate manner.

I would also like to express my gratitude to all faculty members of Nirma University for providing encouragement, exchanging knowledge during my post-graduate program.Special mention to Dr. Usha Mehta (Professor, ITNU) who always inspired us and guided us with her experience in the field of Testing and Verification.

- Vanisha Shukla (16MECV28)

Abstract

Pre-silicon verification techniques in use, cannot assure that all the bugs in software as well as hardware are spotted and removed before actual implementation of the design on silicon. In an analysis in the year 2007, it was observed that industry spends half of the total project duration in post-silicon validation and debugging.Design for debug methodology, helps in the speeding up post silicon debug by improving the internal signal observability of various system layers.

The scope of this work is the coverage of such design for debug logic implemented to SoC for tracing internal signals and for accessing registers of SoC partitions during post silicon validation and debugging.The verification part related to implementation of these logic are emphasized more in this documentation.

Adding to the above verification strategy and SV written for the same, some scripts(in Perl) are also included to aid faster debug of RTL simulation issues.

Table of Contents

List of Figures

List of Abbreviation

ASIC	Application Specific Integrated Circuit
DFD	Design For Debug
DMA	Direct Memory Access
MPSOC	Multi-Processor SoC
RAL	Register Abstraction Layer
RTL	Register Transfer Level
SV	System Verilog
TAP2SF	Test Access Port to SystemFabric
TTM	Time to Market
UPF	Unified Power Format
BFM	Bus Function Model

Chapter 1

Introduction

With the ever increasing demand of adding more and more functionality into a system, either SoC or SiP; it is becoming difficult to guarantee first silicon success of a given die. The existing techniques that is practised are functional verification, static timing analysis, emulation, formal verification and simulation. These techniques finds there limitations in guaranteeing that the first silicon will be error free. This calls for the need of post silicon debug process.Traditionally, it is observed that post silicon debug process is very time consuming. Moreover, post silicon validation requires good **observability** and **controllability** of internal nodes.This can be achieved by taking care of the observability nodes using extra logic, termed as the concept of **Design For Debug**.

Most of the verification process that are practised in industry, works on the model of the SoC and not the actual silicon. In such scenario, the chances that the error go through undetected, is increased. Moreover, pre-silicon verification coverage are mostly based on anticipated failures. Hence, there is a scope of unpredictable behaviour in the post the implementation of the system on silicon. Therefore, a structured debug architecture has become a very desirable aspect in SoCs.

1.1 Motivation

The ITRS states that the time to locate the root cause of a problem grows exponentially with the advancement in process technology that produce larger, denser and more complex designs.

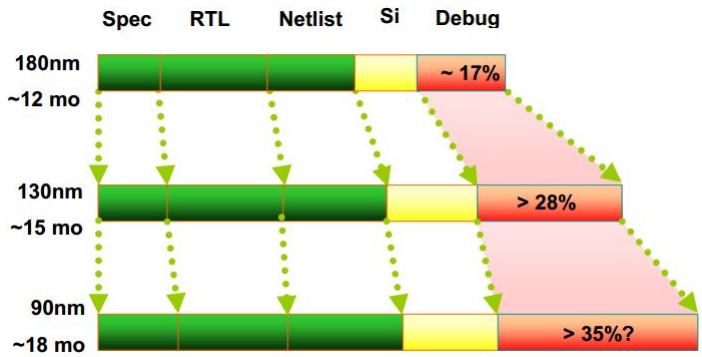


Figure 1.1.1: Technology vs Time To Market

As shown in figure, it has become the most time consuming part on average development cycle of a new chip.Despite its growing importance, silicon debug currently lacks the standard methodology and infrastructure. To ease the post-silicon debug and to shrink down the ever increasing time to market, design for debug has become an essential part of the development process.

The aim of **Design for Debug** is to provide following capabilities for post-silicon validation:

- Increased observability to internal signals of the SoC :This is achieved by extracting the signals, data and traffic information of the SoC through the debug IPs/logic blocks added to the design.
- Increased controlability: Using the design for debug implementation, post silicon validation team can choose the partition/IP/Logic Block whose signals needs to observed at a given instance. The other IPs/Logic blocks can be driven into sleep mode during this instance.
- Device behaviour under different power states can also be traced through monitor pins that are physically available at SoC package.
- Control the triggering, tracing and scanning of the SoC.

Above features makes DFD a critical pillar to aid faster post silicon debug and reduced time to market. Moreover, it forms a bridge between two independent processes, viz. post-silicon and pre-silicon debugs.

1.2 Objective

The primary objective of this work is to verify that the implementation of DFD logic in the RTL is meeting the specification and chip requirement. This is covered under the umbrella of following three work objectives:

- To verify that the debug logic added in RTL is implemented such that it is accessible for post-silicon debug.
- To assure that the added debug logic in the design does not affect the mainstream functionality of the system.
- To verify the connectivity and transactions of the logic network in SoC is as per the features defined for it.

The DFD logic that is verified as a part of this project is explained in a detailed manner. The logic block aids verification and debugging of writes and reads to the partition level TAP register. The second logic is used to trace out critical signals to the physical package pins.

1.3 Flow of Thesis

The thesis work starts with the detailed overview of **SoC, Pre-Silicon Verification and Post-Silicon Validation.**. Next, in the literature survey, **Role of DFD in design flow** is discussed by referencing and summarizing the papers published on DFD.The DFD logic proves to be acting as a bridge between Pre and Post Silicon debugging.This feature is discussed in detail.

Chapter Third consist of detailed description on **TAP2SF DFD IP** and the verification test

plan for the same. Similarly, Chapter Four consist of discussion on **Monitor Pins DFD IP** and its verification plan.

Chapter Fifth describes the verification tests flow and implementation. It covers the errore, fixes and results obtained on the same.Lastly, conclusion is drawn for the over all work with discussion on future work that should be done to enhance the overall debugging time and results.

Chapter 2

Literature Survey

As discussed in previous chapter, the time to market highly depends upon the time spent in post silicon debug. As the complexity of design increases, the post silicon debug time requirement is also increased. In this chapter, the SoC complexity and the process of pre and post silicon debug is discussed in detail. The idea is to understand the problem statement and the need of Design for Debug logic. The DfD logic is also discussed in detail, putting up more emphasis on validation of DfD logic.

2.1 System on Chip - An Overview

System on Chip (a.k.a SoC) is basically and Integrated Circuit wherein multiple component of a system is integrated on single chip. The term ”system” illustrates the interconnected network of components such as, memory unit, power unit, processors, peripherals, etc. The figure 2.1.1 below shows the blocks of SoC

The processors in an SoC could be a microprocessor or specialized processor for the desired purpose, like media processor for modem or audio/video applications.Moreover, SoC may consist of multiple processor. These processors are connected to each other either by memory sharing or by hardware components like mailboxes, channels,etc.

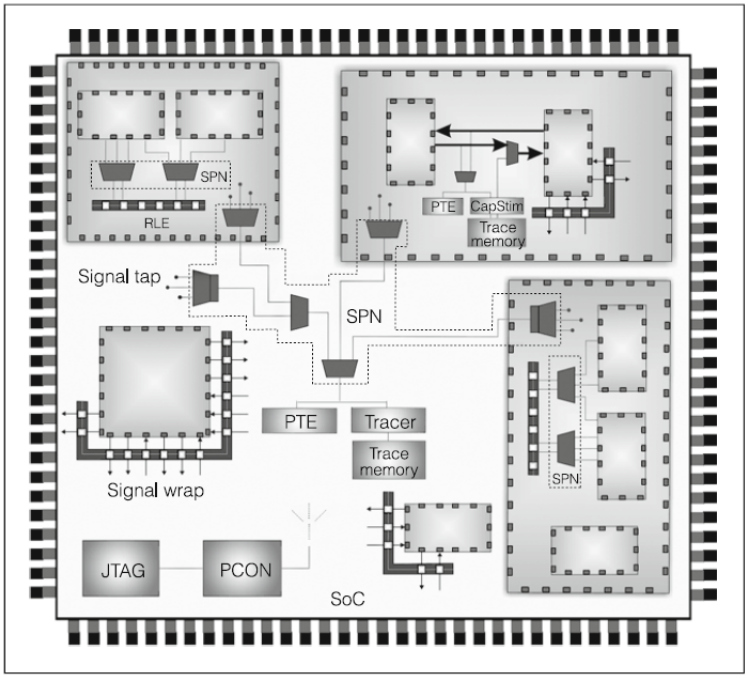


Figure 2.1.1: System on Chip

2.1.1 Components of the SoC

Earlier, SoCs were designed such that it could only be used for some dedicated applications, for instance, video processing. However, with the advancement in technology, current SoC are not specific to one role. Instead, the same SoC are reused for different devices.The figure 2.1.2 shown below, illustrates the basic building blocks of an SoC. A brief description of each of the above SoC components is as below [1]:

- **Processors :** The core of any System on Chip design is the Processor unit. Processors are generally accompanied with units like DMA controllers and accelerators. As per the ITRS, it is observed that from early 2005 to 2010, most of the ASICs consisted of a single Processor. Recently, Multiprocessor SoCs are getting the appearance and are more area efficient to keep improving the performance [1].

- **Buses :** Buses are the interconnects that are shared between major components of the SoC. Soc may contain number of buses such as, dedicated bus to and from power- man-

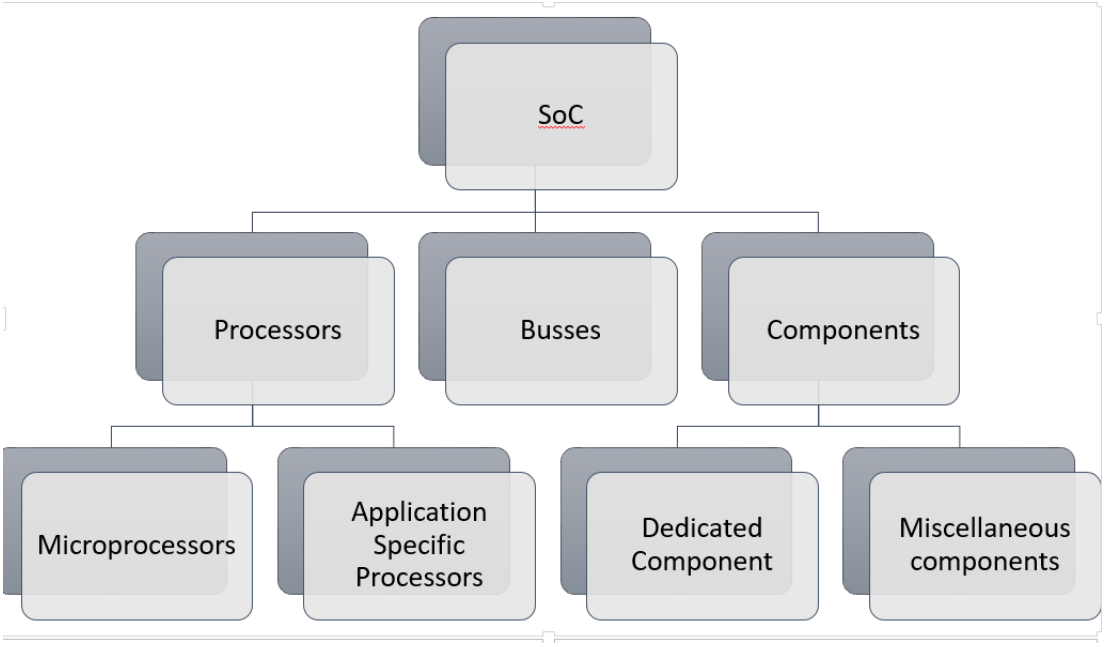


Figure 2.1.2: System on Chip -Components

agement unit and separate bus for other peripherals. The major drawback of system buses is the difference in protocols of the bus, which causes incompatibility amongst them.This barrier can be avoided by using Network on Chip as the next generation SoC interconnect.

- **Components :**

  The SoC components are broadly classified under following two categories:

  1. **Dedicated Components:**The standard circuits such as UART, USB, GPS, etc. can be the part of SoC. At times, more advanced IPs such as turbo decoders might also be the part of the SoC. The benefit of using a dedicated component is that it abides by the specifications and meets the constraints. However, the consequence of this is that the over all price of the chip is increased. Since, the IPs are usually black boxes, the complexity is increased and flexibility is reduced.

  2. **Miscellaneous Components:** This includes highly specific, functionality oriented components such s bio-chips, digital to analog converters, base-band to radio interfaces, etc.

2.1.2 SoC Design Flow:

The figure below 2.1.3 illustrates the three parallel process that are carried out in order to design and manufacture a SoC :

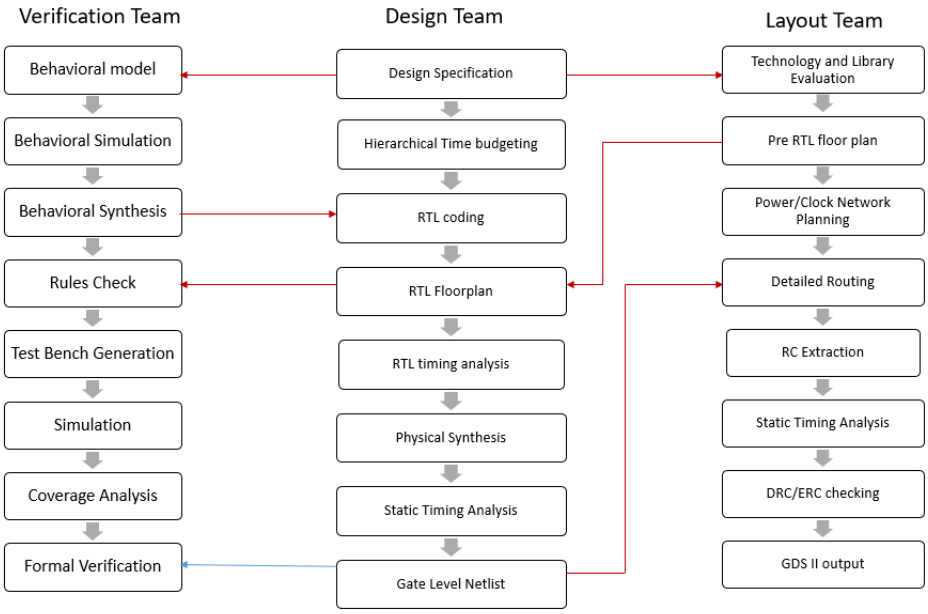


Figure 2.1.3: Typical SoC design Flow

The Design Team starts the SoC design cycle by defining the specifications. These specifications are the passed on to Verification (*Front end*) and Layout (*Back end*) teams.The behavioral modelling and synthesis by the verification team is used to build RTL codes and the floor plan. The floor plan is matched with the pre RTL floorplan by the backend team. Eventually, the frontend team verifies that the floorplan and RTL codes are feasible and meets the specification defined. The design process sign-off happens after the Gate Level Netlist is passed on to formal verification and detailed routing, which leads to GDS II and tapeout.

2.2 Pre-Silicon Verification

Like every other verification, the objective of pre-silicon verification is to verify the design is correct and fulfills the criteria in specification. The primary motive of pre-silicon verification is

to simulate/emulate and observe the model(mostly RTL) by providing real-world like stimuli. To achieve pre-silicon validation, modelling of the design is needed along with behavioural and bus functional models.

Following chart shows the standard front-end pre-silicon verification flow:

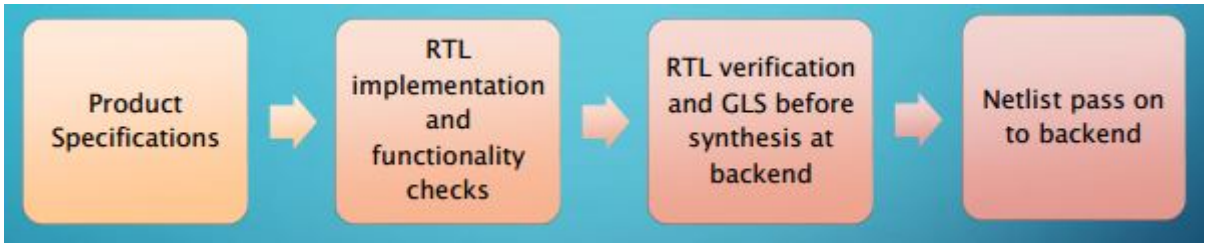


Figure 2.2.1: Front End Verification Flow Chart

As soon as the specifications are defined, the design flow moves to RTL implementation of the specification.The implementation and validation runs synchronously in the front end flow. Post this, the GLS is done and the netlist is passed on to back-end for physical design and implementation.

2.2.1 Key Aspects of Pre-silicon verification

In an SoC, there are various independent scenario and features which occurs simultaneously in real-world environment. The following considerations are hence to be made while performing pre-silicon verification of an SoC.

- **Concurrency** SoC has multiple port and interfaces that acts asynchronously, independently and concurrently. The pre-silicon verification environment should be capable of establishing concurrent real time like scenario.

- **Debug** Ideally, the problems can be resolved with ease if the root cause or the nodes closer to root cause are observable. Verification environment should provide this feature of tracing the root cause to facilitate debugging.

- **Configurability** The SoC verification environment should easily be configurable as they are required to test multiple platforms and features.

- **Result Checking** In SoC level pre-silicon environment, the result checking of multi-port system should be comparatively less cuber some.

- **Level of Abstraction** The selection of correct level of abstraction is the key to SoC level verification. While bit-level signals and interface are difficult to deal with, the transaction level or higher is widely preferred.

2.3 Post Silicon Validation of SoC

As discussed earlier, Post silicon validation is the most time consuming step in VLSI design flow. The post silicon validation is the process of validating the actual silicon prototype for the functionality and specification.The fig 2.3.1 shown below illustrates the typical post silicon debug flow [11].

Since the Post silicon validation is done on a packaged chip, it is not feasible to validate each and every design aspect, as it is done in Pre Silicon Validation.Hence, the post silicon debugging is a very time consuming and relatively less accurate task. Following are the key aspects and features of post silicon debug:

1. Determination of operational regions

2. Physical Properties of the Device

3. Electrical defects can also be diagnosed as functional errors

4. Device behaviour under actual stimulus

5. Device characteristics and behaviour when power rails are connected.

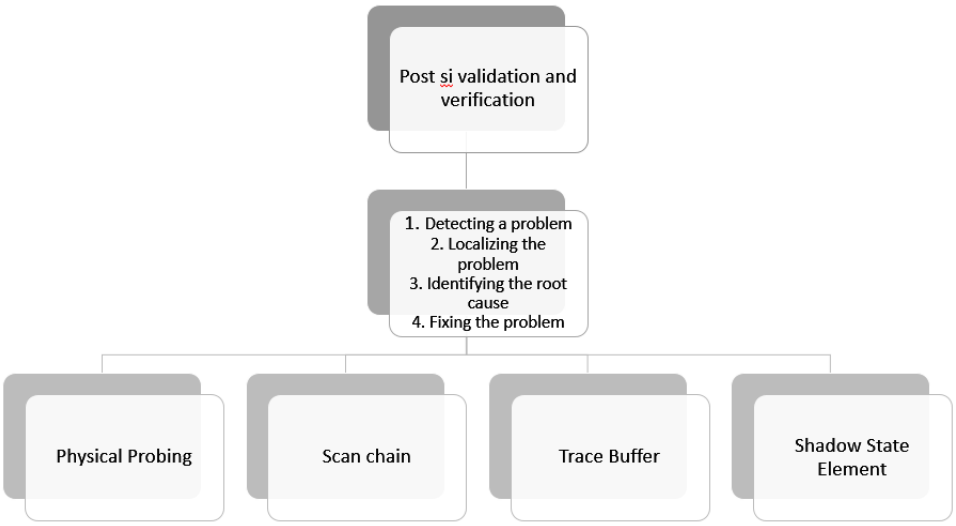


Figure 2.3.1: Post silicon debug flow

There are various methods by which the engineers can locate the actual area of issue. Most of these methods, however, relies on adjusting the operating regions and characteristics of the device. This eventually increases the time required to diagnose and fix a bug.

2.4 Concept of Design For Debug

The concept of DfD comes into picture in the very start of design flow. The RTL includes the debug logic along with the functional circuit in the code. This is done such that the functionality is unaffected by the debug blocks added.

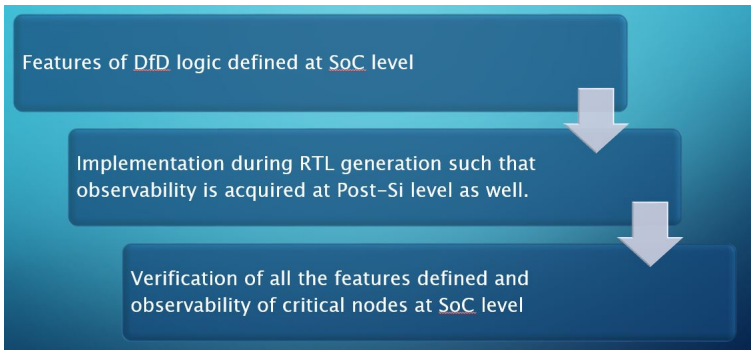


Figure 2.4.1: DfD in design flow

Next, Parallel to RTL synthesis, the verification environment is setup to ensure that the added DfD logic suffices the purpose of it's addition into RTL model.

This continues to pass down to physical design level wherein these blocks are devised as per the golden reference and feature definition.

2.4.1 Implementation of DFD in SoC Design Flow

The purpose of DfD logic is primarily to facilitate the observation and control of critical nodes of the design after the first silicon is taped out. Hence, it is necessary to list out critical nodes during RTL process. This is the primary step of DfD concept.

Refer fig2.4.2 which shows that the definition of DfD features takes place in the RTL level of the SoC design flow. Next, comes the implementation of these logic in the RTL such that

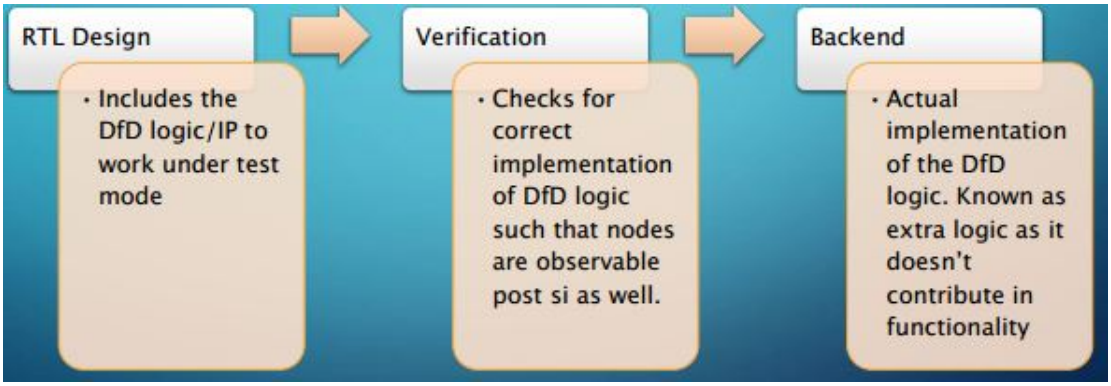


Figure 2.4.2: DFD Concept Flow Diagram

the required observability is achieved in the functional model of the SoC. These nodes should designed such that they are accessible at post-si level as well.

The most critical part of DfD inclusion lies in DfD verification.It is essential to verify not only the functionality of the DfD logic added but also that the base functionality of connected blocks are not affected due to this inclusion. This needs to be done by providing real-world like stimuli.

2.4.2 DFD as bridge between Pre and Post Si Verification

As seen above, the verification of silicon is divided into two mutually independent phases [5],viz. pre -silicon verification and post silicon validation of hardware. As the industry grows, the demand for and integrated methodology to meet the pre and post silicon validation in one platform, is increasing rapidly.

While there are various methods that are formulated for bridging the gap, the *Design for Debug* facilitates the debug process immensely by adding the controlability and observability in the SoC. DFD identifies the critical nodes, such as the PLL outputs and provides a network to trace them out to SoC pins. This helps identifying the bugs in the post silicon validation phase. The idea of DFD is to *add little extra to save lot more*. Adding the DFD circuit not only reduces the TTM but also helps providing the accurate fix for the post silicon bugs. There are few such DFD IPs that were implemented in the SoC and the design was verified by performing time based and input based simulations. In the upcoming chapters, such IPs are discussed in detail.

Chapter 3

TAP-to-System Fabric IP

TAP-to-System Fabric(TAP2SF) is a Design for Debug soft IP that plays a critical role in post silicon debug of system buses and partition level error tracing. The IP is a combination of TAP, JTAG/BFM and on-die system fabric.

Test Access port(TAP) and Joint Test Action Group(JTAG) plays crucial role in most of the DfD subsystem/IP/Design. TAP is basically a popular serial access port for testing and debugging. JTAG is IEEE 1149.1 standard for testing interconnects and functionality in the IPs and PCB.

Along with the above two, this IP incorporates an on-die fabric for credit based communications for exchanging messages between the IPs.These 3 ingredients, the JTAG, the TAP and the on-die fabric forms the building block of the IP.

3.1 Test Access Port

The Test Access Port or TAP is a IEEE standard 1149.1 provides the standard protocol to perform basic test operations.

As shown in Fig.3.1.1 this standard has 3 major building blocks:**TAP controller,Instruction Register and Data Register.**

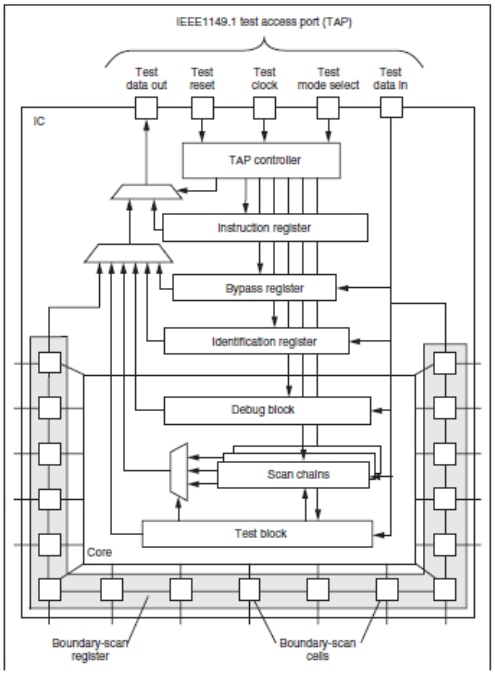


Figure 3.1.1: Test Access Port IEEE 1149.1

These components along with the TAP are capable of performing basic debug operations in the silicon.Brief description of these three components are as below:

- The Instruction Register : This serial shift register is configured to select the test to be performed.
- The Data Register : Consist of the stimulus data for the test to be performed and selected by the instruction register.
- TAP controller : Generate the control signals and clocks to execute the control sequence provided by TAP.

The TAP consist of essentially four pins and optional 5 pins. These are described below :

1. **TCK or Test Clock Input :** This pin provided the test clock which is independent of all the system clocks to facilitate synchronization between different SoC partitions/blocks. The rising edge of clock is used for loading the test input data while the falling edge of clock checking the test output data.

2. **TMS or Test Mode Select Input :** The test operations to be performed are decided by sequence of the serial combinations of 1's and 0's at each falling edge of TCk. These information is then sampled by the TAP controller to generate selection and control logic accordingly. It is pulled up to logic 1 by default.

3. **TDI or Test Data Input :** The data input through TDI is either populated to instruction register or the data register, on the basis of TMS in the previous clock. The input is taken in the falling edge while the register shift is performed in the rising edge. Like TMS, TDI is also designed to be such that it is set to logic 1 by default.

4. **TDO or Test Data Output :** The serial output at this pin either comes from the IR or DR based on the sequence in TMS. Once TDI is available, TDO is obtained after number of TCK cycles equivalent to register length. By default, it is high impedance state.

5. **TRST or Test Reset Input (optional) :**This can set the circuit to known starting state, which is very crucial in testing.

With this overview of the key features of a Test Access Port, the TAP2SF protocol is illustrated vividly in the sections ahead.

3.2 Overview of TAP2SF IP

As described in the introduction, the IP consist of JTAG/BFM, TAP registers and On -Die system Fabric.A controller in connection with the SoC or internal event within the SoC, configures the TAP network to get a software connection to the IP.The intended transaction is setup by writing on to number of **Test Data Registers (TDRs)**, as soon as the TAP becomes active.On writing to the TDR, the fabric endpoint gets ungated(if it was gated) and a request is made to it. If credits are available, then the transaction is sent. The host controller can then poll the response TDR/opcode to determine whether the transaction is complete.

3.2.1 IP Description

The IP is used for out of band access to configure and check status of the internal registers of the SoC fabric and functional Logic block for debug and testing purposes.This will help sending and receiving messages/data to and from the fuctional IP that would, in turn, help in the post silicon test and debug purpose.

The destination IP comes with a TAP and opcodes to ensure the interface between the TAP protocol and IP message fabric protocol.There is a set of test data registers at destination IPs that are required to be configured to access the registers and data internal to IP.

The data sent through the JTAG port consist of these configuration values along with the op-codes to configure the intermediate routers.The fig.3.2.1 below illustrates the IP block diagram:

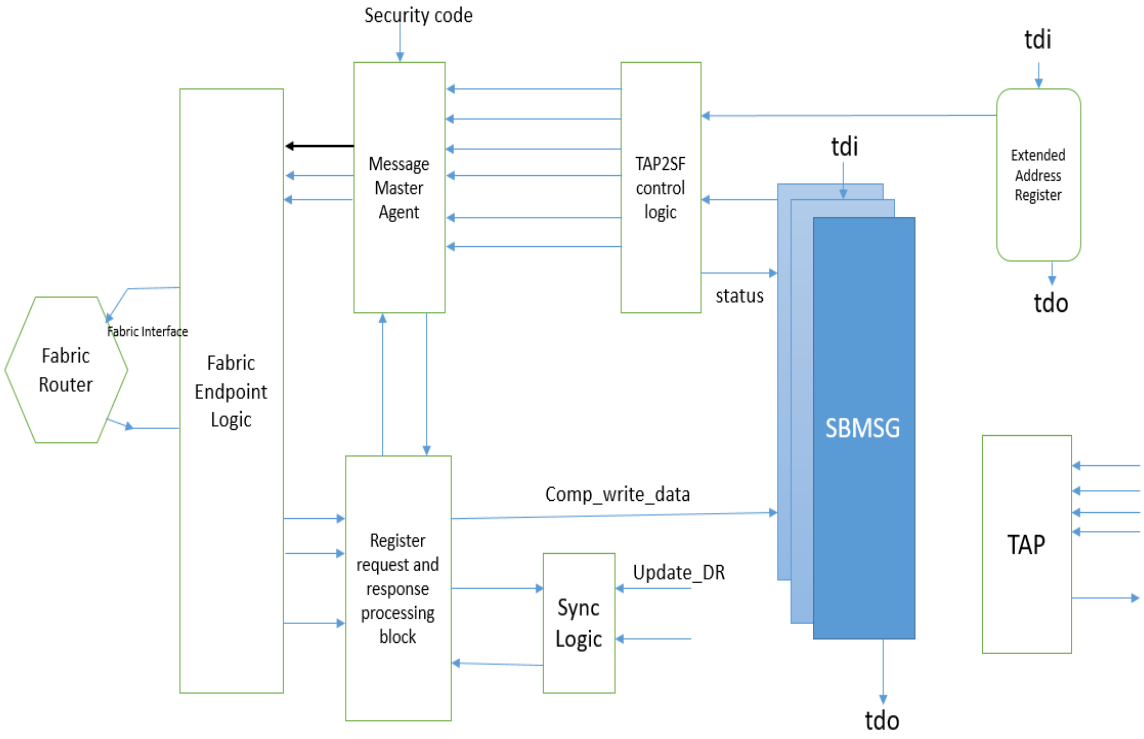


Figure 3.2.1: TAP2SF IP Block Diagram

3.2.2 Features of the IP used in the SoC

Depending upon the SoC specifications, the features or the expected functions of the IP is varied. However, few basic feature of the IP under consideration are as below :

1. Facilitates access to data transfer through a stream that is independent of main in-band data transfer, to configure the registers internal to SoC via TAP.
2. The IP can be used for message reads and writes.
3. The supported size of data for transaction varies from 0 to 64 bits.
4. Follows security features and protocols defined for the SoC to prevent unauthorized access of the internal registers of the SoC.

Apart from above mentioned features, there are other features as well, which are highly specific to the project. Those features are conceptual derivatives of above specified feature.

3.3 Verification Strategy for the IP

IP verification covers the test plan, test development and test execution. Upon the test results, the constraints are checked and the failures/bugs are fixed. This is done under the RTL simulations.

3.3.1 Security Features :

The major conflict in modern SoCs lies between security and debugging.The former prevents exposure of signals and data internal to SoC whereas the latter is a technique to trace SoC's internal signals and observation of critical nodes.The possible ways to balance this includes:

- Disable the debugging feature before the release of the SoC.
- Enable the authentication methods to observe SoC's critical nodes for debugging.

The first method is not recommended as debugging features are required even after SoC release

to facilitate the maintenance. Hence, most of the debugging logic has the second feature enabled. The IP into consideration, uses the available security feature on the basis of write, read and control register configuration.Various security methods that are brought into use on this IP are:

- Enabling only selective IPs for sending and receiving debug signals and preventing other IPs to be accessed through this protocol.
- Authentication of predefined security codes at the destination IP before tracing it's data/signals.
- Provision of overriding the security code in destination IP, to debug the issue of incorrect initial code being set in the destination IP.

Above are the basic security features taken into consideration while debugging using the IP under emphasis.

3.3.2 RTL Collateral

RTL collateral refers to the RTL files, descriptions, hierarchy or any such information, that will be needed to create a test-bench and provide the stimuli. For TAP2SF IP, it is required to test the register accessibility via **backdoor access** to the RTL register. The *backdoor access* is the method of accessing an RTL path/Register at zero access time. Unlike, typical method to access,i.e. *front door access*, this method of accessing the paths at zero simulation time, is a faster method. In UVM, backdoor access to registers are performed through **RAL** i.e., *Register Abstraction Layer*.The RAL utilities contains information that are needed for driving a bus [2]. The file contains the complete description of registers at system level. This aids the front and backdoor accesses to Registers. The information and RAL generation are provided by the RTL design team.The figure 3.3.1 illustrates the RAL concept

The next collateral that is needed for TAP2SF verification is the **port-ids** and **security codes** of the IP whose register needs to be accessed. The *port ids* are the opcodes to identify the IP for

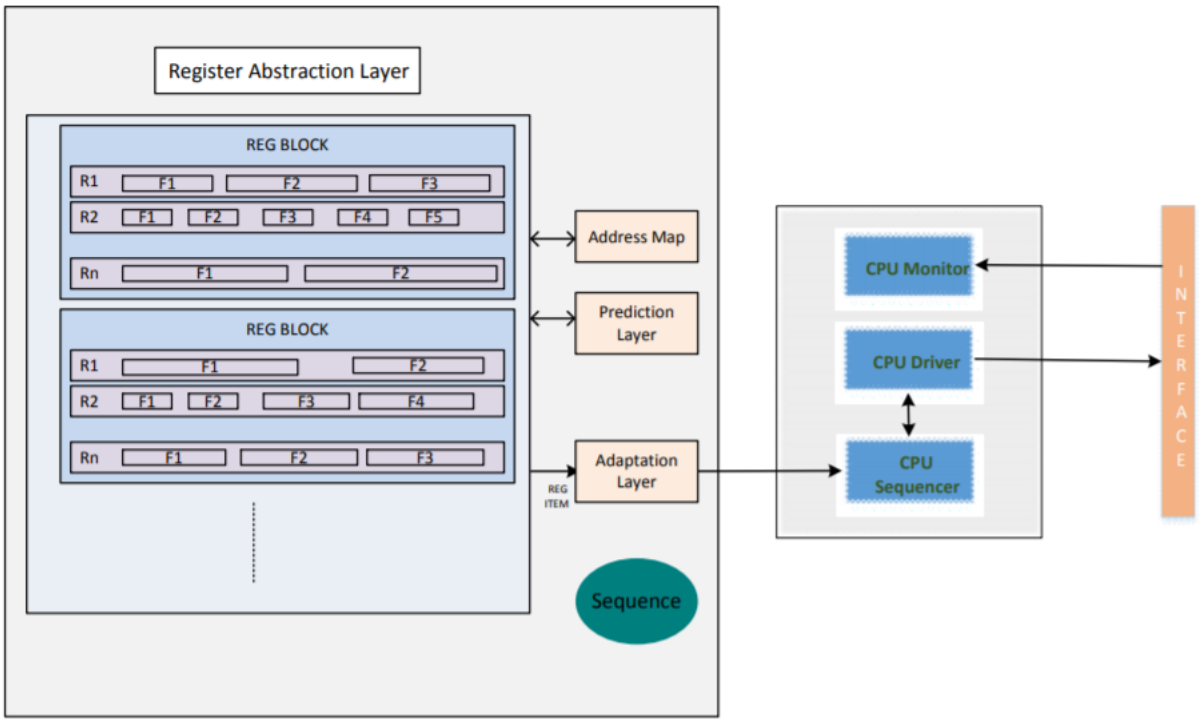


Figure 3.3.1: Concept of RAL in UVM

read and write transactions. The *Security Codes* may or may not be applicable for all the IPs. They are the unlock codes to ensure authorized access to the IPs.

3.3.3 Environment Build

The verification environment is build using OVM/ Intel Specific methodology and the packages/modules to mimic DUT. This is essential for developing test-bench on top of DUT and check for the scenarios. The Intel specific verification methodology exploits the concept of both UVM and OVM.The UVM test phases are as shown below [3] fig.3.3.2:

The TAP2SF is verified to run on for following two phases:

1. Before the Resets: To ensure access of critical IPs/Registers just after the silicon boot up.

2. After the Main phase: The main phase is again divided into sub phases, wherein the transactions are driven after the security unlock and resets are complete. This is also called User data phase.

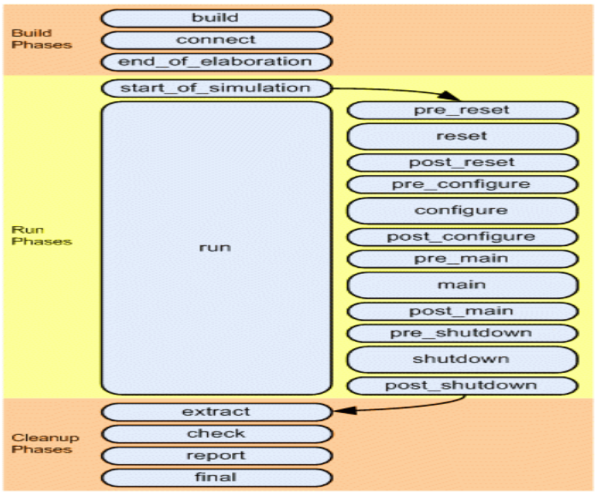


Figure 3.3.2: UVM PHASES

3. With Unified power format: This ensures the IP functionality matches the specification under power-on conditions.

Based on the above aspects, the test scenarios are developed.

3.3.4 Test Scenario and Regression List

Defining test scenario forms the most critical aspect of pre-silicon verification. The appropriate set of tests only can assure the functional as well as the code coverage of the given design under test.

For the IP under consideration, the test mainly covers all the features defined for the IP during the start of the project. Following are the key scenario for which a set of tests were generated :

1. Configuration of test data registers for write, read and control protocols for debugging through **external interface**
2. Ensuring the security features are enabled and does not affect the IP functionality.
3. Verify that the write/read happens after the **internal event/trigger** is initiated.
4. Intermediate router setting and configuration test to ensure that the connectivity is established properly.

5. Ensure that the features works properly data and destination address of possible widths and not just a particular size.

Based on the above scenarios, The code is generated on OVM methodology in SystemVerilog language. The test sequences are generated such that they cover all the IP Features. These sequences are then listed under one umbrella, that is called **Regression List**. The *Regression List* is the process of enlisting the test sequence of IP features into one file and re running them for all the designs where the IP is used. The failures in regression run represents the *bugs or mismatch of functionality of a feature*. The failures are brought into notice, analyzed and debugged to ensure the correctness of the functionality.

3.3.5 Base sequences

Base sequences are the standard test sequences/block of code to perform certain operations. They can be inherited and used in any of the test-bench directly in SystemVerilog.

The TAP2F tests uses the base sequence to access the RAL file and perform the read/write and check over the registers. It also includes the test sequence that waits for *internal trigger/event* to occur, so as to initialize transactions without external interface.

3.3.6 Test-bench flow

The test-becnch flow is developed for following three scenarios. Each of the flow is described using a flow chart:

1. Test Using RAL Access to the IP registers (fig.3.3.3).
2. Test Using Internal Event (fig.3.3.4).
3. Test Using Port id of target (fig.3.3.5).

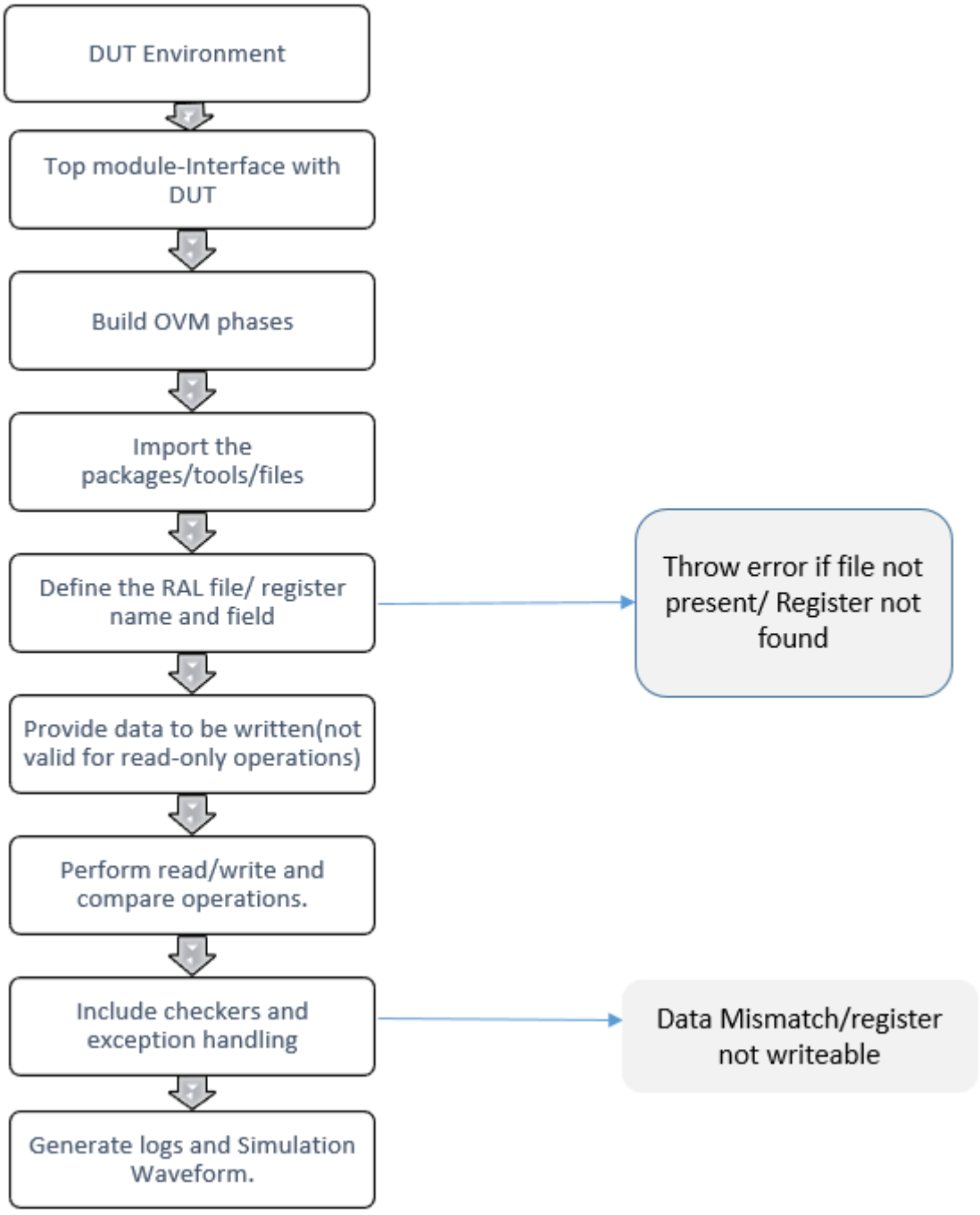


Figure 3.3.3: Test flow Using RAL

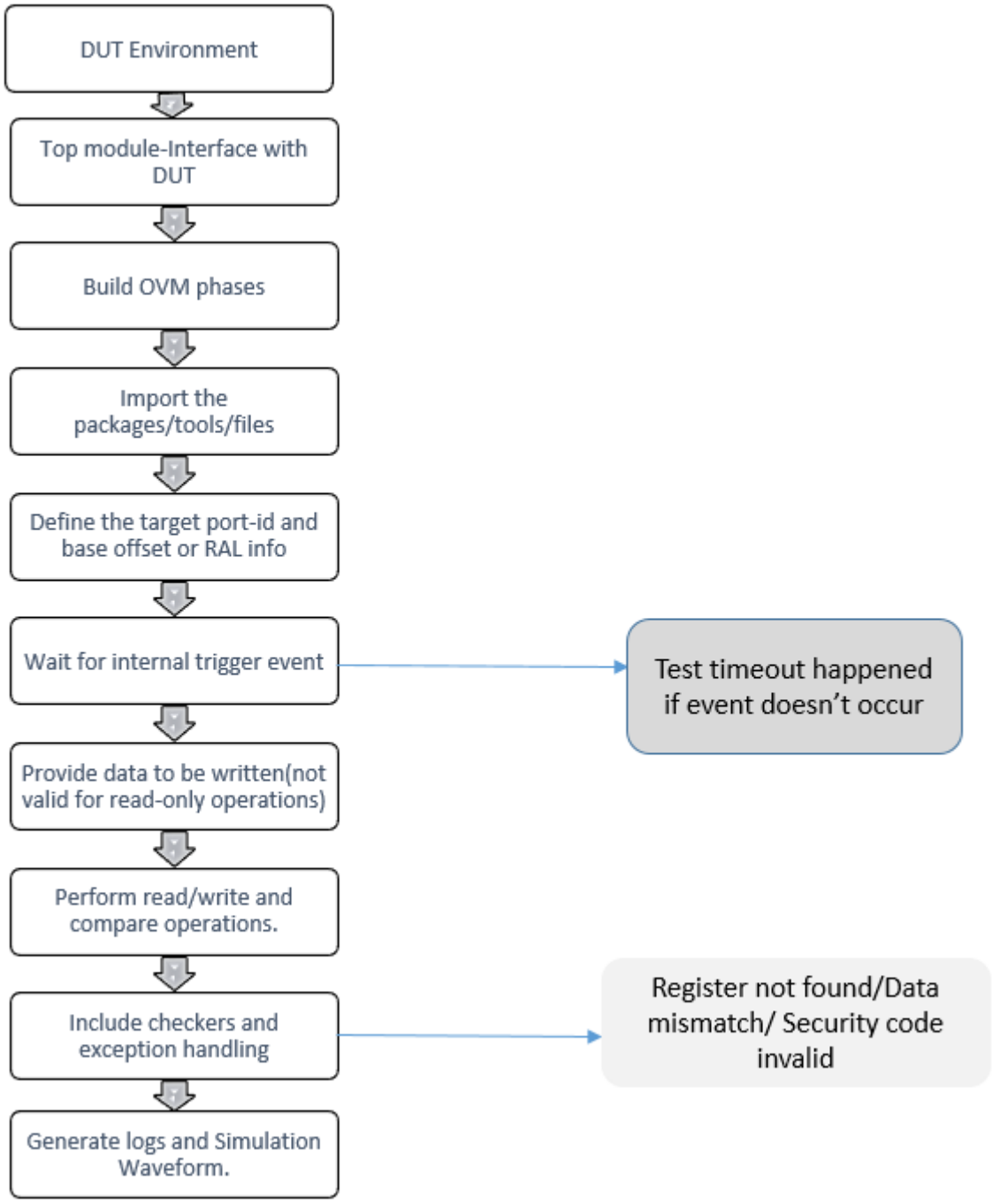


Figure 3.3.4: Test Flow based on internal event

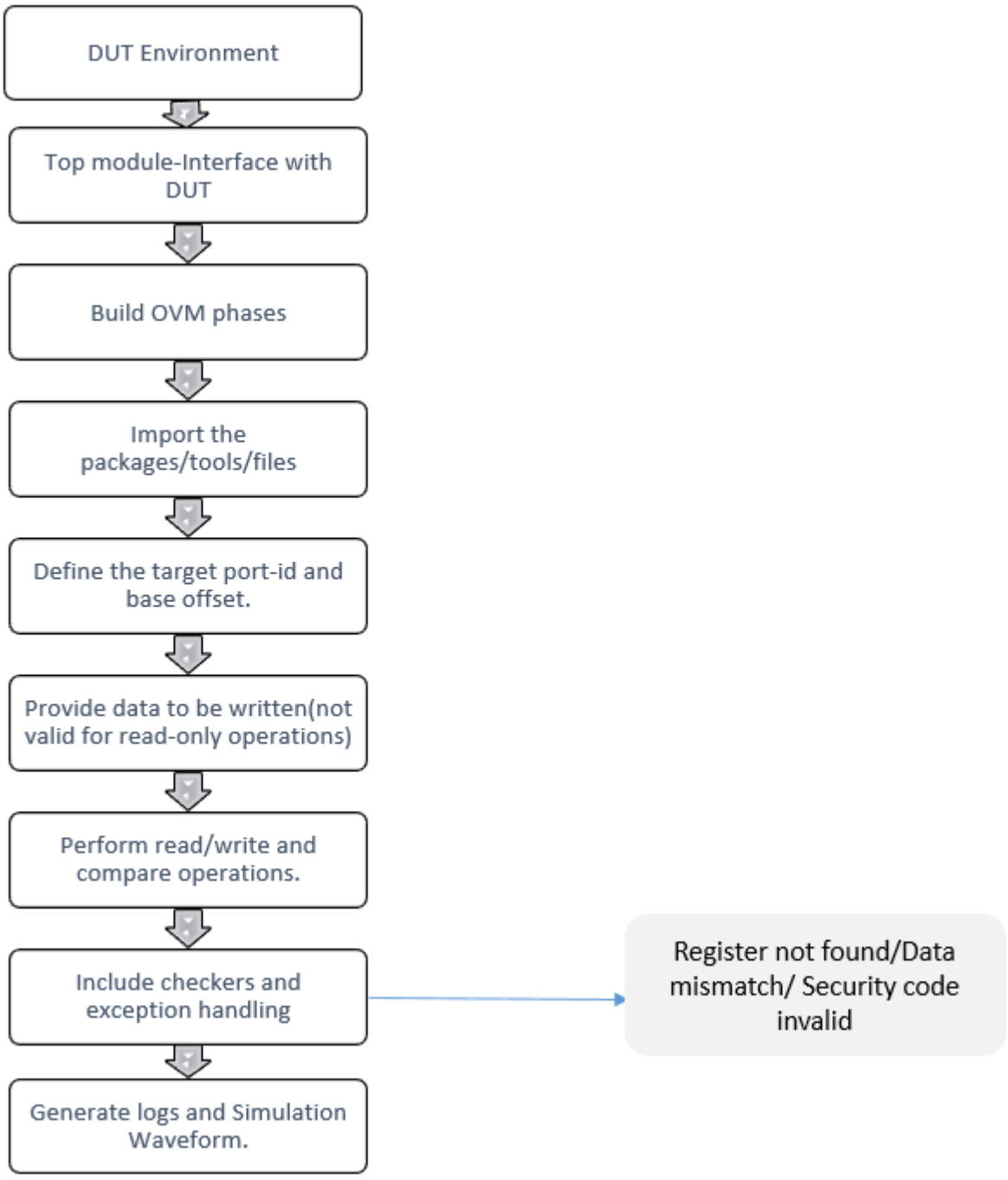


Figure 3.3.5: Test flow using Port id of target

Chapter 4

Monitor-Pin IP

The Monitor Pins DFD architecture provides a unified solution to have high speed analog/digital observe along the high speed input/output, PLL, Core, Analog IPs' etc. This is achieved by having only 2/4 SoC level pins dedicated for post silicon debug purposes.

4.1 IP Overview

The IP consist of network of logic blocks(such as multiplexers or pass-gates) that helps fetching out the critical signals from within the SoC partition or the SoC subsystem. The signals are configured, selected and traced out to SoC pins using this IP. The following fig 4.1.1 illustrates the block diagrammatic representation of the IP functionality.

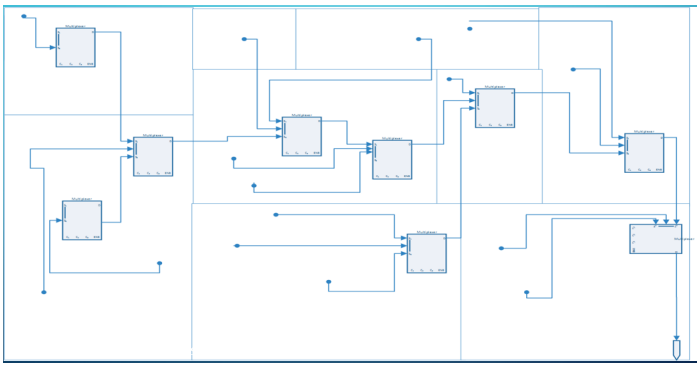


Figure 4.1.1: Monitor Pins network in an SoC

Each of the block represents a partition/subsystem of an SoC. As soon as the debug mode is enabled, the signals from each of the partitions enters the monitor pin network.Based on the selection and configurations of the monitor pins the signals can be observed on the oscilloscope connected to SoC package pins.

4.1.1 IP Description

The implementation method of the Monitor Pins highly depend upon the operating conditions and the Silicon architecture.The selection of the type of HIP to be used is completely dependent on the subsystem features and the behaviour of the output that has to be traced.Based on the type of signals to be traced and the die features, the two kinds of monitor pins were implemented in the SoC. These two types are illustrated as below:

- **Monitor Pins Type - I**

  This is the simplest monitor pins used in SoC. It consist of combination of *Multiplexers, TAP registers and I/O probes*. This is optimum for tracing out signals from passive components of the SoC. The digital signals are mostly traced out using this configuration.

  The TAP registers are configurable through BFM/JTAG during the post silicon process. On the basis of values fed in the TAP registers, the MUX selection is made and the corresponding signal is obtained in the SoC pins.

  This HIP is repeated multiple times in the design, such the it covers all the critical nodes. The final level HIP muxes all partition level the HIP outputs into two GPIO probes at SoC level.

- **Monitor Pins Type - II**

  This is an advanced monitor pin HIP used typically for tracing out active, analog signals. The voltages and frequencies can be traced out with minimum error through this network. The HIP consist of network of *pass gates and Remote Test Data Register (RTDR)*.

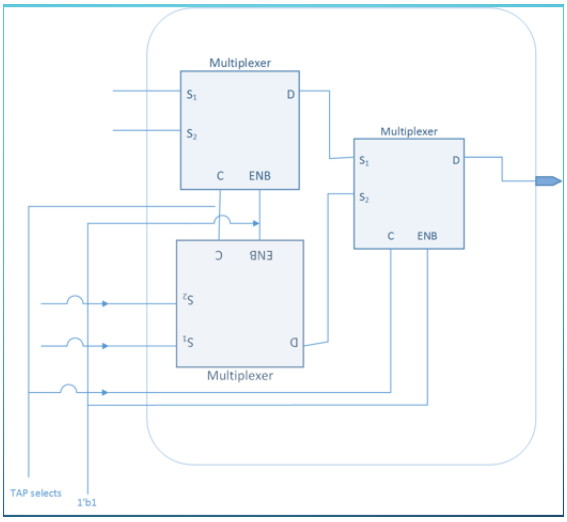


Figure 4.1.2: Monitor Pins HIP Type - I

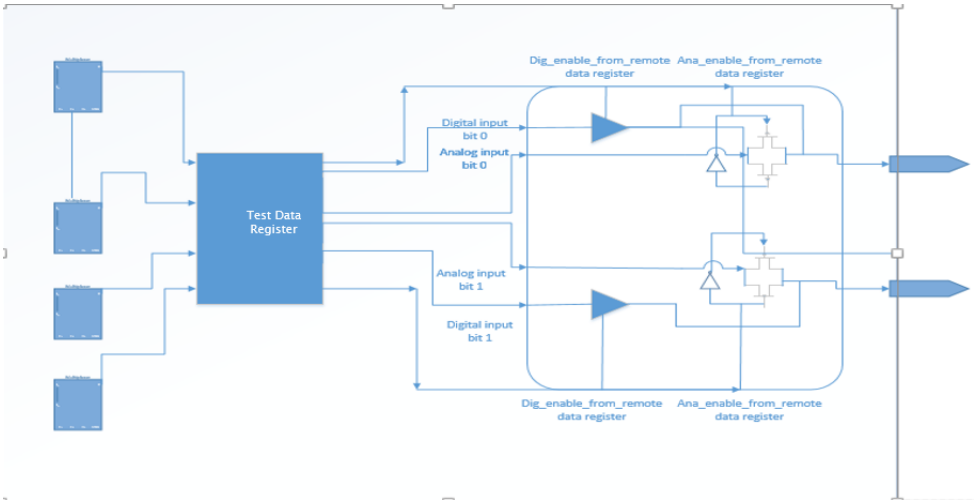


Figure 4.1.3: Monitor Pins HIP Type - II

As shown in fig.4.1.3, both analog and digital signals enters the monitor pin HIP. The RTDR is configured so as to enable either the digital or the analog signal to be viewed at the GPIO. The two outputs of the GPIO represents the bit 0 and bit 1 of the similar/mutually exclusive signals.

Apart from tracing the critical signals out, the monitor pins also provides the feature of *Providing input signal* through one of the SoC pins. The purpose of this is to aid the **Low Yield Analysis** during the post silicon validation and prototyping.For this feature, one of the monitor pins is implemented as **Inout pin** to provide dual functionality.

4.1.2 Key Benefits of Using this IP

This DFD IP offers many key benefits such as:

- Consistent and converged debug feature for design to implement and post silicon to use.
- The monitor pin features High Speed Digital observe (up to 2+ GHz), Analog observe.
- Simultaneous view capability for signals within an IP or between PLL and clock digital outputs.
- Pins to provide external voltage reference to IPs within the SOC and for parametric test for interconnects.

These features aids faster spotting of erroneous partitions/features of the SoC in post silicon hardware validation on the prototype. As the IP has the capability of tracing the clocks within the SoC, this IP is configured as the first few IPs after the silicon power-on and boot-up.

4.2 Verification Strategy for the IP

The monitor pins plays a very critical role in post silicon validation process. Hence, the pre silicon validation for the same is essential. The verification environment for the IP is such that it configures the DUT signal at each of the HIP input, one at a time and trace it at the output. The detailed plan is as discussed ahead.

4.2.1 Verification Requirements

The RTL information that are needed for the monitor pins verification is as below:

- RTL hierarchy of the signals to be viewed from each partition.
- The corresponding monitor pin allotted to each of the view signal.
- The Register Definition files for each of the TAP configurable register of the monitor pins.
- Signal mapping details , i.e. the partition signal to GPIO pin mapping.

These details are provided by the integrator/implementer of the design. The following table shows dummy values and description that are provided by the design team.

4.2.2 Verification Use Cases

As soon as the debug mode is enabled, the monitor pins starts rendering some default outputs. Using the TAP protocol, the TAP registers at each of the HIP is fed with the select values for the desired signals. At the final partitions, before the GPIO, there are a series of buffers to be enabled so as to choose between the digital and analog signals. Moreover, one of the SoC monitor pins is implemented as inout, i.e. it can receive input signals on the basis the buffer configuration.

Anticipating from the above descriptions, following are the various use cases that are identified and covered as the part of monitor pins pre-silicon verification.

Connectivity

Foremost use case is to verify that the monitor pin network are connected as per the specifications provided by the implementer. There may be scenarios where the pins are wrongly connected or interchanged while integrating the design. This is verified by configuring the HIP to trace out each of the signals are SoC pins, considering one at a time.

Response to input transitions

The IP should work for both **0 to 1 and 1 to 0** transitions. This is desirable because when the silicon will be powered on, the critical signals can be of any form. Since it is expected that the signals should be fetched out as they are, it is necessary to verify the IP for all feasible transitions.

Simultaneous Operations

There are two pins dedicated at SoC level. Both the input pins should be capable of rendering the outputs simultaneous. This is one of the salient feature of the monitor pins HIP and hence it is essential to verify.

TAP register access

The HIP has TAP registers to which the signal select values are fed.Based on these values, the signal is selected to be brought out at SoC Pins. Hence, it is critical to validate that the TAP registers are configurable, accessible and writeable. It is also necessary to check that the selected signal matches with the specification defined for that select value.

4.2.3 Test bench Approach

Based on the above use cases, the approach for the test bench development is as below fig4.2.1.The above flow covers the verification of all the signals that are expected to be viewed out at the SoC level Monitor pins.

The next use case, which need to be approached separately is the behaviour of the **inout configurable monitor pin**. This was out of scope of the project, yet can be verified by simply configuring the buffers. The buffers enables the pins to act as input or output on the basis of the configuration.

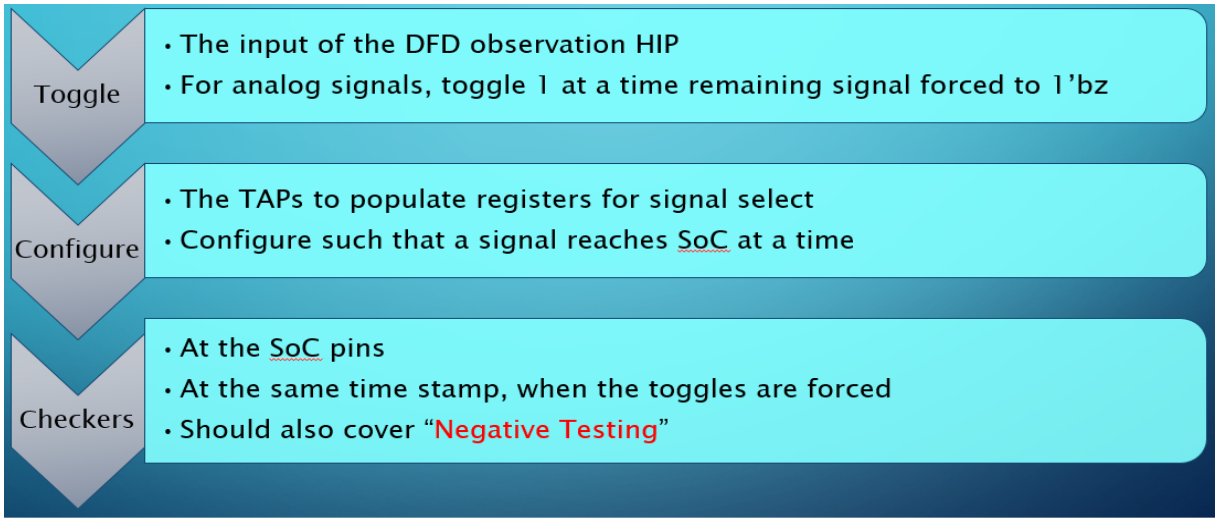


Figure 4.2.1: Monitor Pins Verification Flow

4.2.4 Test bench flow

The flow chart in the fig4.2.2 illustrates the testbench flow that was implemented to verify the connectivity of the design.

The TAP configuration can be automated by parsing the select values using the *Perl/TCL xml parser script*. This saves time and efforts and also reduces the human error. The sample xml format, that can be used for populating the TAP register to select one signal at a time is as shown above.

Along with the TAPs for HIP input selection, the buffers are also configured by TAP registers to select between the type of output to be viewed, i.e. digital/analog. This is also covered in the test bench before applying checkers to verify the outputs at SoC pins.

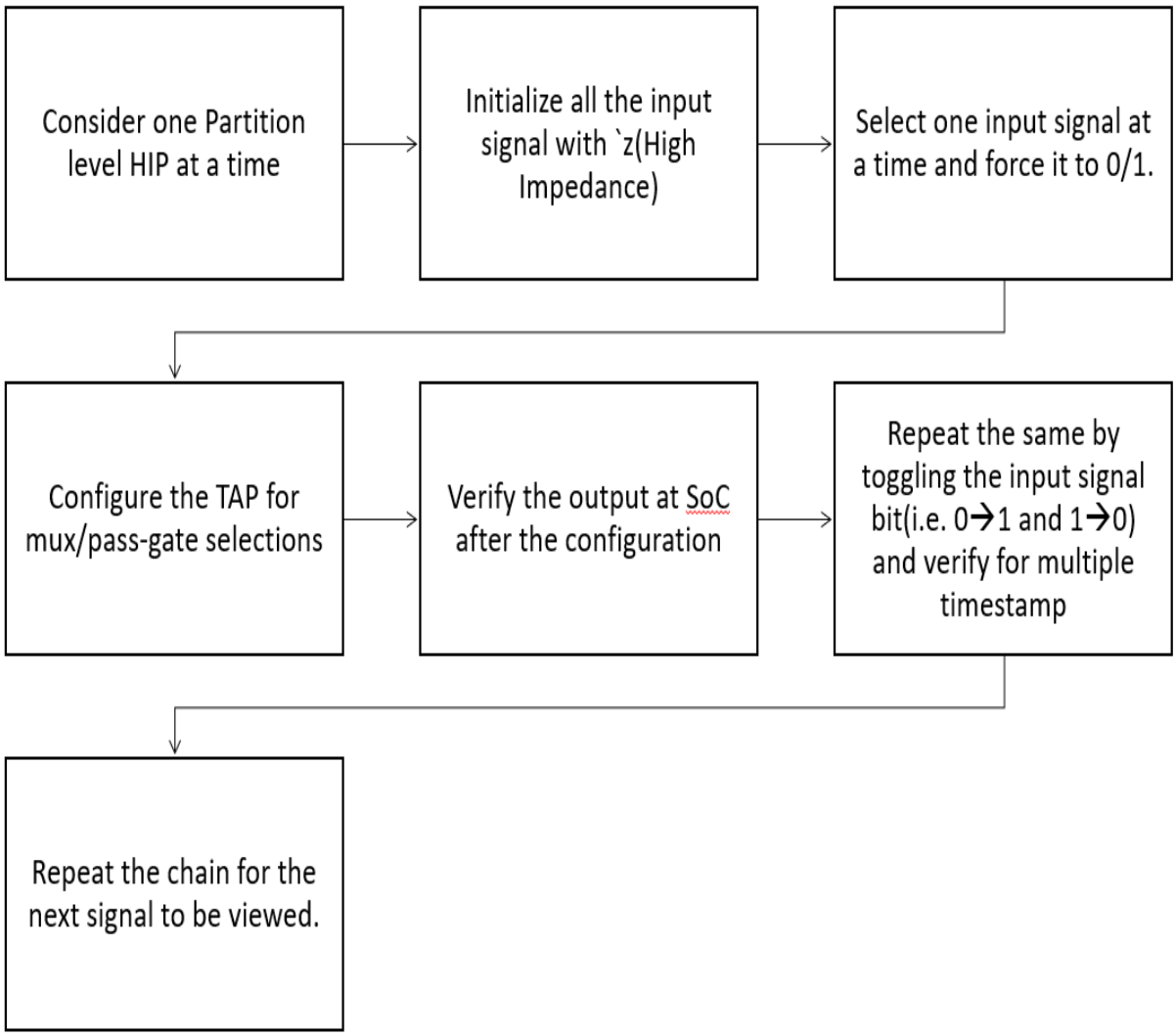


Figure 4.2.2: Test bench Flow diagram

Chapter 5

Verification Results

The analysis of the results/outputs obtained through testbench is as important as developing a testbench. Once the simulations are complete, the behaviour of the DUT can be studied through the reports and the wave-forms. The testbenches developed in this work are **Self-Checking**, meaning that the checkers are written in the testbench to verify the results with the specifications. The typical self-checking testbench flow is as shown in fig.5.0.1:

Transaction generator sends the stimulus information to scoreboard. In the similar fashion, monitor also sends data to scoreboard. This data is verified with the expected response.Adding to this, there are two kinds of verification methods, that are applied in the test bench. These are listed as below:

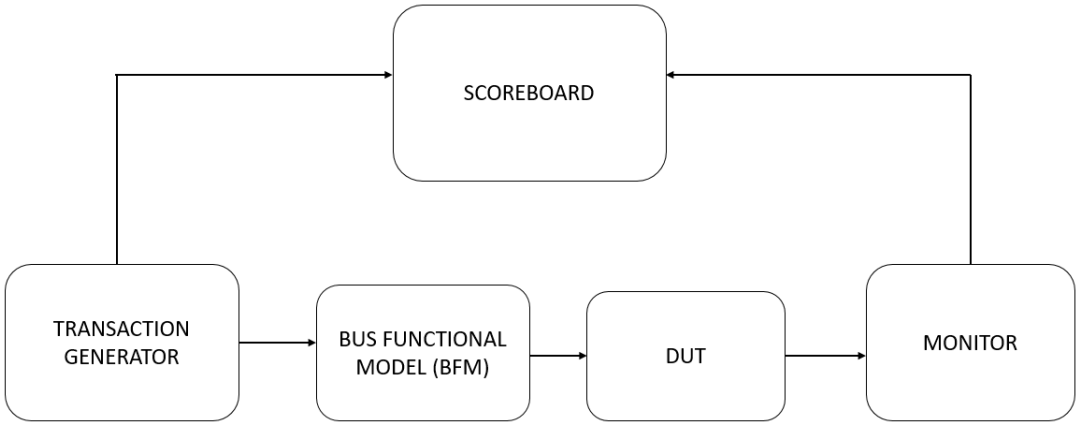


Figure 5.0.1: Self Checking Testbench

1. **Directed testbench:**This is suitable for the testbench wherein all the scenarios are well known.In this method, the stimulus is provided explicitly for all the given scenario. The coverage achieved is 100 percent but the test time is too large to be feasible. Hence, this method is considered only for DUT wherein the scenarios are known.The validation of **monitor-pins** IP is done using this method.

2. **Constrained Random Testbench:** Directed testbench misses out unpredictable scenarios.Moreover, the time required is too large. Hence, the concept of *random testbench* came into picture. The random testbench has the limitations like generating the invalid scenarios or generating the same scenario multiple times. This led to the concept of **constrained random testbench**, wherein the corner case are covered using selective random testing. The **TAP2SF IP** is validated through this method.The random value is generated using *seed value* provided in test run command.

5.1 TAP2SF - Verification and Results

The verification of TAP2SF IP follows a generic flow. The data write/read and the response is noted in the scoreboard. The generation of data to be written is based on the seed value provided in the test command. Following are the scenarios that are considered for this IP:

- Whether the behavioural aspects of the RTL are verified through the test.
- Whether the test bench covers all possible scenario for a given feature.
- Whether the test is able to simulate the functionality of the design and check the critical nodes.
- Whether test meets the required code and functional coverage criteria.

Based on above scenario, following are the steps/process that needs to be carried out for verifying TAP2SF.

```
/*************************************************************/
class my_test extends soc_dfx_boot_base_test;
  /*************************************************************/
  `ovm_component_utils(my_test)
  function new (string name="my_test", ovm_component parent=null);
    super.new (name, parent);
  endfunction : new

  function void build();
    super.build();
    set_config_int("*.dfx_tap_virt_seqr", "count", 0);
  endfunction : build
  function void connect();
    super.connect();
    env.set_test_phase_type("env", "DATA_PHASE", "tap_seq");
  endfunction //void

endclass // my_test


/*************************************************************/
module soc_dfx_tap2sf_wrrdrsp_test();
  /*************************************************************/
  initial begin
    run_test("my_test");
  end
endmodule // soc_dfx_tap2iosfsbgp_posted_wrrdrsp_test
```

Figure 5.1.1: OVM phases to build a testbench

5.1.1 Building Test Environment

Firstly, it is required to build a real-like simulation environment. This helps verifying the DUT for actual stimulus. This is achieved by using OVM methodology to setup the testbench phases and initialize the stimulus.This fig.5.1.1 shows the OVM phases that are initialized in the test-bench:

5.1.2 Calling Sequences

Once the OVM phases are defined, the test sequences are included as per the stimulus and type of configurations needed.For TAP2SF IP, following two base sequences were added:

- RAL Access Sequence: to access a register for write/read using the RAL information.
- Accessing register through destination port id and base address.

These sequences are included in the testbench to be used after the configuration on the TAP2SF IP register fields.

```
/*************************************************************/
task dfx_tap2sf_wr(bit [7:0] dest_portID,bit [15:0] addr, bit [31:0] data);
  /*************************************************************/
  tap_inst_t  taps;
  logic tdo_l[];
  taps = tap_inst_t'(ovm_tap_pkg::TAP2SF);
  tap_env = ovm_tap_env::get_ptr();
  set_access_path("tap_agent");
  tap_env.set_ir_all_check(0);
  tap_env.set_dr_check(0);

  `sla_msg (OVM_LOW, get_type_name(), ("dfx_tap2iosf_wr > Send SBMSGGO write transaction"));
  tap_set_cmd(taps,TAP2SF_MSG_FIELDS);
  tap_set_field("WRITE/READTransc"    ,0);
  tap_set_field("FIELD0" ,dest_portID); //set dest id

  tap_set_field("FIELD1" ,8'h7);  // set opcode
  tap_set_field("FIELD2" ,8'h00); //set addrlen ADDRLEN=1 48 bit address
  tap_set_field("FIELD3" ,8'hf);  //set SBE FBE
  tap_set_field("FIELD4" ,8'h0);  //set endpoint FID
  tap_set_field("FIELD5" ,addr[7:0]);   // Reg0
  tap_set_field("FIELD6" ,addr[15:8]);  //Reg0
  tap_set_field("FIELD7" ,data[7:0]);   //set write 1st_DataByt
  tap_set_field("FIELD8" ,data[15:8]);  //set write 1st_DataByt
  tap_set_field("FIELD9" ,data[23:16]); //set write 1st_DataByt
  tap_set_field("FIELD10",data[31:24]); //set write 1st_DataByt

  tap_env.set_dr_check(0);
  issue_tap_cmd();
  #5ns;
endtask // dfx_tap2sf_wr
```

Figure 5.1.2: Write transaction Configuration

5.1.3 Configuring Register Fields

The TAP2SF IP has a 96 bit register whose filed has to be configured for write/read transaction. Based on these values, the TAP2SF starts the protocol. Example of configuring a write transaction is as shown in fig.5.1.2

For initiating transaction based on internal event, the stimulus can be provided by writing the value to the triggering block register. This register would then initialize the sequence of internal triggering.

5.1.4 Errors and Fixes

The most common errors is encountered while validating the IP are stated below, along with there description :

1. **Time Out Error :** This error occurs when the test run time exceeds the predefined time allotted for the test to complete. This can be due to a hung condition in which the test is stuck at a point, awaiting certain response.

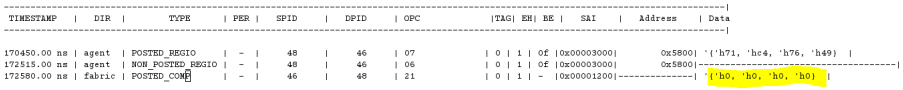


Figure 5.1.3: Tracker Showing data mismatch

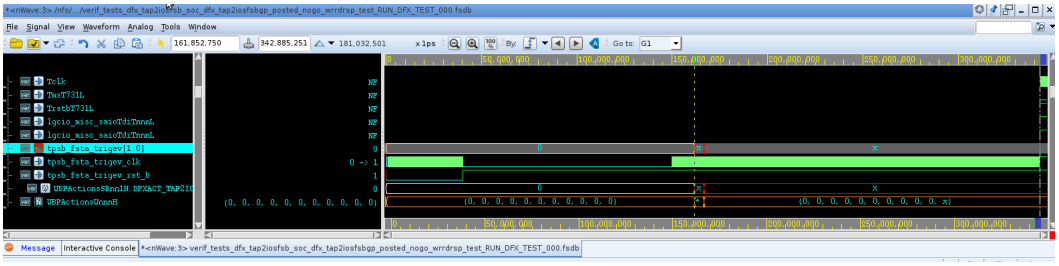


Figure 5.1.4: No trigger received

2. **Required Text Unavailable :** This occurs when the file pointer or the data handler provided in the test is incorrect or unavailable in the design environment.

3. **Cross Module Reference :** If the given RTL hierarchy doesn't match with the actual DUT, this error is encountered.

4. **Mismatch between data :** While checking the register for write/read operation, if the read data does not match with write data, then this error is encountered. The fig. 5.1.3 shows the empty tracker, wherein the write data is sent to port id 16'h46 but the data cannot be read back from the same. This causes data mismatch error

   Fix: The reason for this error is any of the following:

   1.Incorrect destination portid/base address initialized in the testbench.

   2.The destination register might be a Read Only register.

   3.Incorrect source/destination portid assigned in the RTL.

5. **Internal event triggering is not occurring** This is the case when the transaction waits for internal event. Consider the following case where all the field value is "X" as the trigger signal *trigev* is "X". Refer fig.5.1.4

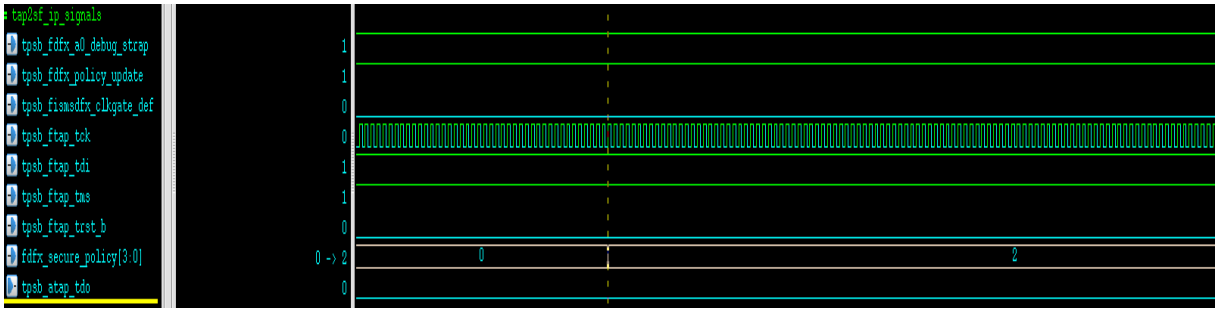


Figure 5.1.5: Source IP signals

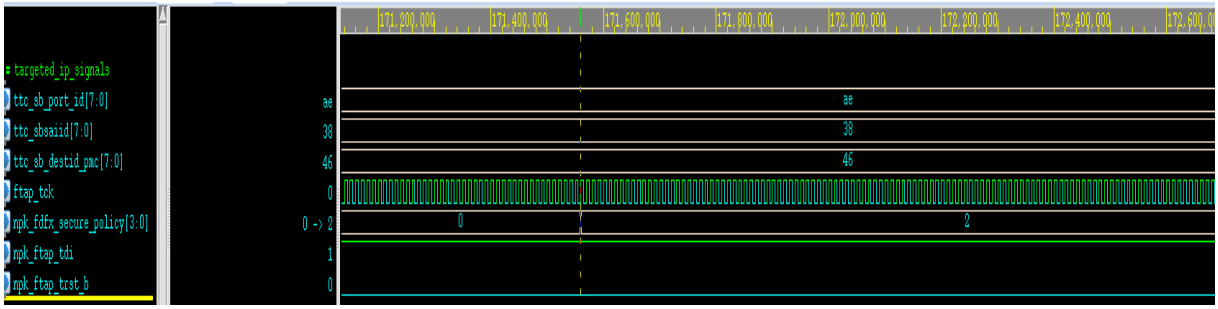


Figure 5.1.6: Destination IP signals

5.1.5 Successful/Passing transaction results

In the case when the transaction is initialized successfully, the source IP i.e. TAP2SF IP will have following signal values reached. Refer fig.5.1.5

In case of successful reception of data in the destination, the following signals are set. Refer fig.5.1.6

When the transaction is complete at destination, the values of register field can be observed as below.Refer fig.5.1.7

At the same instance, the source signals will have the following signals. Refer fig.5.1.8

Toggle Coverage Report

The IP level toggle coverage obtained is 48.9 percent. The chart is as in fig.5.1.9

The SoC level coverage report is as shown in fig.5.1.10:

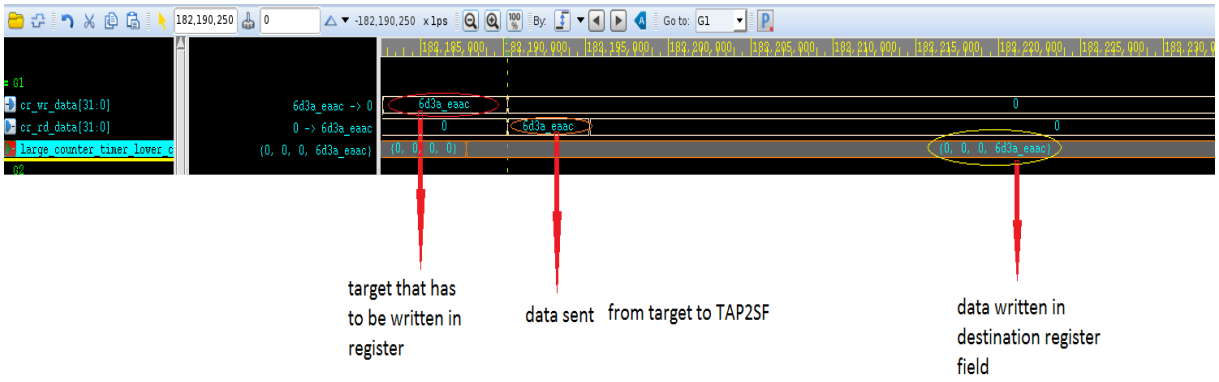


Figure 5.1.7: Destination Register fields after successful write/read

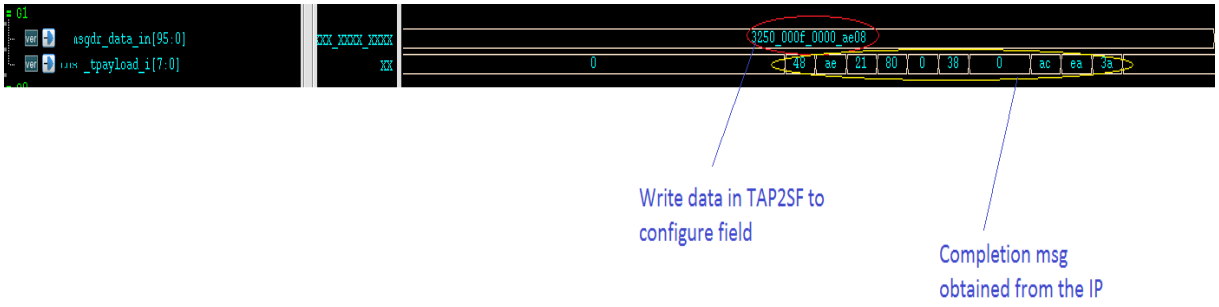


Figure 5.1.8: TAP2SF completion payloads

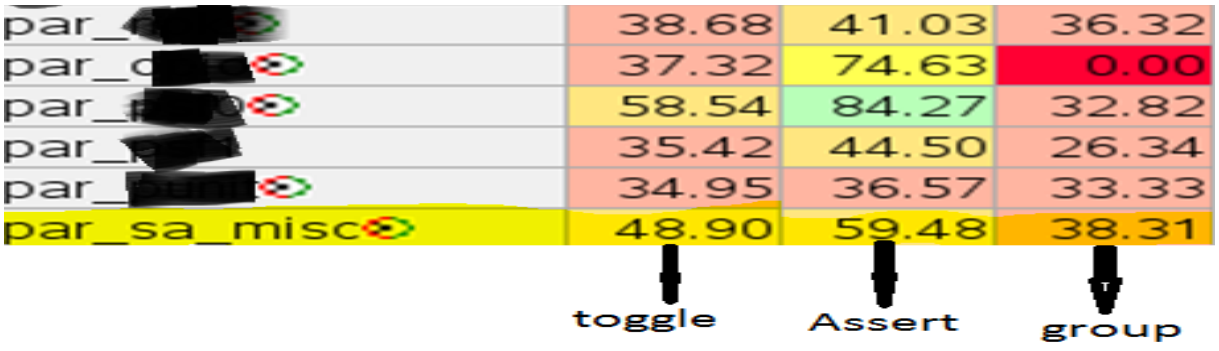


Figure 5.1.9: IP Coverage Report

Date: Mon Oct 30 22:11:58 2017
User: amuthur1
Version: L-2016.06-SP2-6
Command line: urg -dir passed_socrtl_ww43.5.vdb passed_memss_ww43.5.vdb passed_socrtlgt_ww43.5.vdb
/exclusion_files.el
Number of tests: 3

Total Coverage Summary

SCORE	TOGGLE	ASSERT	GROUP
44.45	71.40	27.84	34.10

Hierarchical coverage data for top-level instances

SCORE	TOGGLE	ASSERT	NAME
49.62	71.40	27.84	soc_tb

Total Module Definition Coverage Summary

SCORE	TOGGLE	ASSERT
35.99	46.32	25.66

Total Groups Coverage Summary

SCORE	INST SCORE	WEIGHT
34.10	6.49	1

Figure 5.1.10: SOC Toggle Coverage

src num	ip name/viewpins	par_ccf	par_east_fabric_top	r_east_fabric_top_c	par_disp_fabric	par_west_fabric	par_lgcio_misc	par_gt_hip	par_gt
2	spare	don't care	don't care	1	don't care	don't care	2	don't care	don't care
3	pdlvrastl	don't care	don't care	2	don't care	don't care	2	don't care	don't care
4	pdlvrastret0	don't care	don't care	3	don't care	don't care	2	don't care	don't care

Figure 5.2.1: TAP select value of various HIPs for a given signal

5.2 Monitor Pins-Verification and Results

The verification method for monitor pins uses the concept of directed testbench. The environment build up is same as that for TAP2SF IP. However, stimulus is provided by toggling the HIP input pins to 0 transit to 1 and 1 transit to 0.

5.2.1 Preparing the signal selection sheet

To drive a signal to SoC pins, it is required to simultaneous select that signal in all the HIPs in the network.For this, the RTL designers provides the connectivity details. Based on that, the TAP selection sheet can be made as in fig5.2.1

5.2.2 Defining Macros

The RTL signal that has to be toggled, is defined under a *def.sv* file. This file holds all the macros that can be used in the testbench through their handlers.This provides **Reuse-ability,**

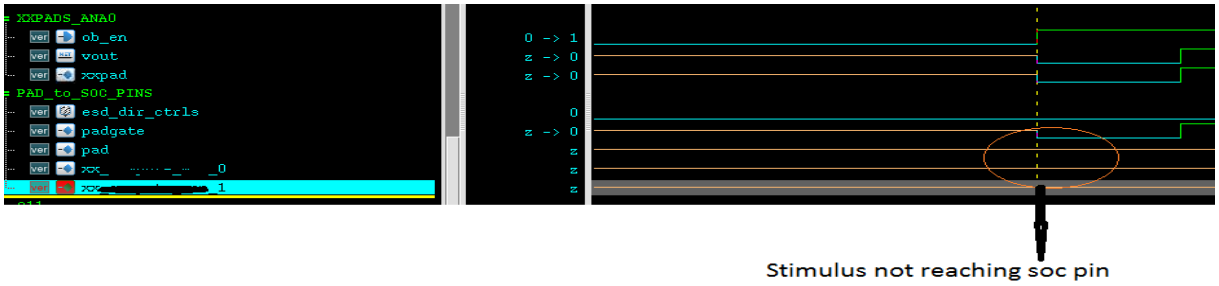


Figure 5.2.2: Signals not reaching SoC probes- Checker failure

modularity and flexibility to the code.

5.2.3 Configuring Network

Based on the select values as in fig5.2.1 the network is configured and the signal is driven to the SoC pin.This is done multiple times in a simulation to observe the results for multiple timestamps.

5.2.4 Checkers

The checkers are initialized each time when the signal is toggled and the network is configured. The checkers validates that the stimulus provided, matches with the SoC package pin values at any given instance.

5.2.5 Errors and Results

The error occurs every time the checkers fail, i.e. the stimulus doesnot reach the SoC pin. Refer fig.5.2.2 The reason of failure can be any of the following:

- Wrong connectivity of network in the RTL.
- Incorrect signal select value fed to the Selection registers.
- For monitor pins HIP Type-II, the passgates might not get enabled. This is majorly due to Unified Power Format build failure. Due to UPF failure, the power rails doesnot get connected to passgates of the monitor pin HIP. Refer fig.5.2.3 showing upf build failure.

```
================================================================================
[Check Modes]
  Mode:ncc_rtl_test -> Satisfied
================================================================================
[Test Summary]
  Test ID               : verif_tests_dfx_tap2iosfsb_soc_dfx_tap2iosfsbgp_posted_wrrdrsp_fuse_test_before_warm_reset_RUN_DFX_TEST
  Test Name             : verif_tests_dfx_tap2iosfsb_soc_dfx_tap2iosfsbgp_posted_wrrdrsp_fuse_test_before_warm_reset
  Test Type             :
  Status                : FAIL
  Cause                 : Logfile Errors
  Logfile Error Count   : 176
  Logfile Warning Count : 19278
  Host Machine          : scc300193
  CPUTIME               : 15070.58
  Runtime               : 15577
  Sim_Time              : 283002000
  SimRate               : 0
================================================================================
[Error Summary]
# (All, Sorted by Severity, Other)
  1 - ALL : 176
    1462.75 ns: soc_tb.soc.par_east_fabric.sbrssa_wrap.sbrssa.sbrssa_inst.sbcport2.both_inmsg_puts_asserted: ERROR: Sideband tpcput and tnpput are both asserted
================================================================================
CLOSE DATE    : 01/26/18 08:17
================================================================================
```

Figure 5.2.3: UPF build failure report

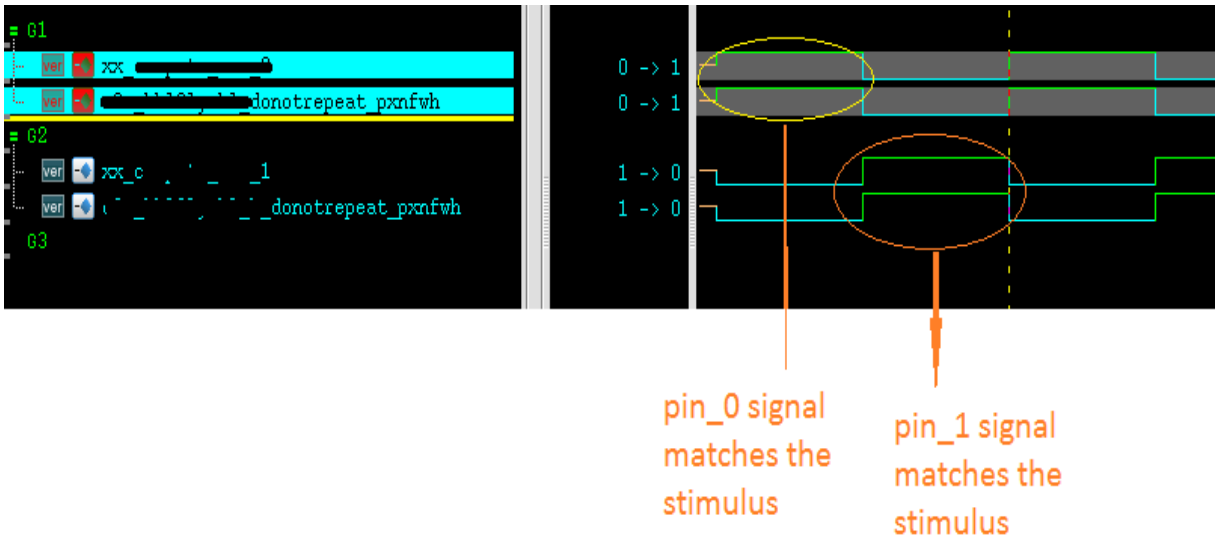


Figure 5.2.4: Passing Waveform

Refer fig5.2.4, that shows the passing testbench waveform.

Chapter 6

Conclusion and Future Work

Design for debug acts as a bridge between post silicon and pre silicon debug processes. Although highly similar, these two processes are mutually exclusive and done separately. This increases the time required to make the silicon production ready.

The DFD IPs implemented and verified in the project can help the post silicon team to spot out the exact area of bug and the eventual fix for the same. This enhances the efficiency of the design without affecting the privacy and security. A little compromise on area can largely provide benefit by reducing the time to market.Considering the ever increasing complexity of the SoC, DFD logics comes as a boon to the industry.Since the IP is implemented only to assist post silicon debug, it is **disabled before bulk production of the chip**. This ensures the security of the design.

As a part of project, the OVM methodology is implemented using SystemVerilog. This turns out to be the advanced and efficient method of verification and debugging of IPs.The directed and constrained random testbenches are implemented, that assures the verification of the corner cases.

6.1 Future Work

The future work includes:

- Enhancing monitor pins testbench: Currently, the monitor pins are validated partition-wise. This means, there is a separate test for each partition. This leads to increase in regression run-time as the environment needs to be built up again and again every time a new partition is tested.The time can be reduced by merging the test benches into single testbench.

- XML parser to feed TAP register feed value: Instead of manually providing, the xml-parser script in PERL/Python can be included in the testbench, that can populate the register write value based on the signals to be configured in Monitor Pins network.

- Improvement of TAP2SF toggle coverage: This can be done by including more number of tests in the regression for write/read in various partition level registers.

References

[1] Risset T. (2011) SoC (System on Chip). In: Padua D. (eds) Encyclopedia of Parallel Computing. Springer, Boston, MA

[2] Akkem Sailaja,"UVM RAL: Register on Demand",*DVCON Conference and Exhibition, 2015*

[3] Rosenberg Sharon, UVM Concepts and Architecture,*48th Design and Automation Conference*, San Diego,CA, June, 2011.

[4] J. Backer, D. Hly and R. Karri, "Secure design-for-debug for Systems-on-Chip,"2015 IEEE International Test Conference (ITC), Anaheim, CA, 2015, pp. 1-8.

[5] A. DeOrio, J. Li and V. Bertacco, "Bridging pre- and post-silicon debugging with BiPeD," 2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, 2012, pp. 95-100.

[6] N. Nicolici and H. F. Ko, "Design-for-debug for post-silicon validation: Can high-level descriptions help?,"2009 IEEE International High Level Design Validation and Test Workshop, San Francisco, CA, 2009, pp. 172-175.

[7] P. Nagvajara and B. Taskin, "Design-for-Debug: A Vital Aspect in Education,"2007 IEEE International Conference on Microelectronic Systems Education (MSE'07), San Diego, CA, 2007, pp. 65-66.

[8] B. Vermeulen and S. K. Goel, "Design for debug: catching design errors in digital chips," in IEEE Design Test of Computers, vol. 19, no. 3, pp. 35-43, May-June 2002.

[9] A. Adiret al., ”A unified methodology for pre-silicon verification and post-silicon validation,”2011 Design, Automation Test in Europe, Grenoble, 2011, pp. 1-6.

[10] M. Melaniet al., ”An Integrated Flow from pre-Silicon Simulation to post-Silicon Verification,”2006 Ph.D. Research in Microelectronics and Electronics, Otranto, 2006, pp. 205-208.

[11] R. Agalya and S. Saravanan, ”Recent trends on Post-Silicon validation and debug: An overview,”2017 International Conference on Networks Advances in Computational Technologies (NetACT), Thiruvanthapuram, 2017, pp. 56-63.