Automation of I/O ring design and logic realization

Major Project Report

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

In

Electronics & Communication Engineering

(VLSI Design)

By

BHAUMIK MATHOLIA

(10MECV11)

Department of Electronics & Communication Engineering

Institute of Technology

Nirma University

Ahmedabad-382 481

May 2012

Automation of I/O ring design and logic realization

Major Project Report

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

In

Electronics & Communication Engineering

(VLSI Design)

By

BHAUMIK MATHOLIA

(10MECV11)

Under the Internal guidance of

Dr. Amisha Naik

and

External guidance of

Mr. Vivek Sinha

Department of Electronics & Communication Engineering

Institute of Technology

Nirma University

Ahmedabad-382 481

Declaration

This is to certify that

1. The thesis comprises of 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 acknowledgement has been made in the text to all other material used.

BHAUMIK MATHOLIA

Certificate

This is to certify that the Major Project entitled **"Automation of I/O ring design and logic realization"** submitted by **BHAUMIK MATHOLIA**,towards the partial fulfilment of the requirements for the degree of Master of Technology in VLSI Design of Nirma University, Ahmedabad is the record of work carried out by him 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.

Internal Guide

Dr. Amisha Naik
(Associate Professor, VLSI Design)
(Nirma University)

External Guide

Mr. Vivek Sinha
(Senior Technical Manager)
(STMicroelectronics)

HOD

Prof. A. S. Ranade
(Professor, EC)
(Nirma University)

Director

Dr. K Kotecha
(Director, Institue of Technology)
(Nirma University)

Date: ---

Place: Ahmedabad

Acknowledgement

It gives me a great pleasure to take this opportunity to thank **STMicroelectronics Pvt. Ltd.** and **Joshipura Jwalant** for giving me such a great opportunity to do project in their esteemed organization. I deem it my privilege to have carried out this dissertation work under this well-known quality conscious organization.

I would like to express my sincere thanks to my mentor **Mr. Rakesh Gulati**, for guiding me through this project. He has been a source of inspiration and has constantly inspired me to give my best and exert my capabilities to the fullest. Without his arduous task of reviewing my work at every step , the project could not have been completed in the present form.

I would also like to express special thanks to my Technical Manager **Mr.Vivek Sinha** & all my team members who were always there whenever needed. They always supported me and were always ready to help whenever I had any doubt.

I would also like to thank **Prof. N. M. Devashrayee**, Course Coordinator (VLSI Design), Nirma University of Science And Technology, for providing me an opportunity to take up this training and for their constant support and encouragement.

- BHAUMIK MATHOLIA

10MECV11

Abstract

Manual design can occur with small number of transistors. As number of transistors increase through VLSI design, the amount of evaluation and decision making has become overwhelming. Computer-aided Design (CAD) tool automates this whole process, hence it reduces TTM (Time To Market). CAD tools allow large and complex problems to be solved. CreateChipIOLogic (CCIOL) is a CAD tool, which is used to generate padring, insert BSR (Boundary Scan Cell) at RTL level and generate IO Muxing logic. Output file of this tool is a Verilog file. As per the required specifications of the different projects at STMicroelectronics, new functionalities are added in the tool. Hence, Regression Testing plays important role to check whether the previously tested tool code runs correctly or not and the added functionality has not negatively impacted any functionality that it offered previously. Regression testing steps **(I)** Make test plans and create test cases for CreateChipIOLogic tool **(II)** Check generated output logic corresponding different inputs. **(III)** Check syntax of generated Verilog file using ncsim (Cadence) tool. **(IV)** Check equality of different hierarchy structures by using Formality (Synopsys) tool. **(V)** Check generated code is synthesizable or not by using Design compiler (Synopsys) and Spyglass tool.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

- Application-specific integrated circuit (ASIC) design is based on a design flow that uses hardware description language (HDL). Most electronic design automation (EDA) tools used for ASIC flow are compatible with both Verilog and very high speed integrated circuit hardware description language (VHDL). In this flow, the design and implementation of a logic circuit are coded in either Verilog or VHDL. Simulation is performed to check its functionality. This is followed by synthesis. Synthesis is a process of converting HDL to logic gates. After synthesis, the next step is APR (auto-place-route).

- Manual (Human)design can occur with small no of transistors. As no of transistor increase through SSI and VLSI, the amount of evaluation and decision making would become overwhelming (Trade-offs). So Computer-aided Design (CAD) Tool automate this whole process. CAD tools allow large problem to be solved.

- List of CAD tools made internally by ST
  - Sqirrel
  - CreateChipIOLogic
  - Tortoise

  - SDC Promotion
  - Spirit2regbank
  - UGN BIST assembler
  - FEKit

- CAD Tool provide several advantages:
  - Ability to evaluate complex conditions in which solving one problem creates other problems.
  - Use analytical method to access the cost of decision.
  - Use synthesis method to help provide a solution.
  - Allow the process of proposing and analyzing a solution to occur at same time.
  - It reduces TTM.

- CreateChipIOLogic (CCIOL) tool is useful to generate padring, insert BSR (Boundary Scan Cell), and generate IO Muxing logic. The Output of CreateChipIOLogic tool is verilog file. This tool can reduce human error by increasing automation and also reduce the TTM.

- Any SOC, there are only two real physical constraints that dictate how much functionality can be incorporated: Package pin out and a fixed amount of die area. Bigger packages with more IO cost, more money, consume more board area, and can require costly socket. So there is an always motivation to pack as much functionality as possible into the small package. Thus providing great deal of power and flexibility to the customers. So IO Multiplex Pads are required.

- Any CAD tool code change can cause existing functionality to break. Changes to a tool code component could impact dependent components.

- It is commonly observed that a tool code fix could cause other bugs.

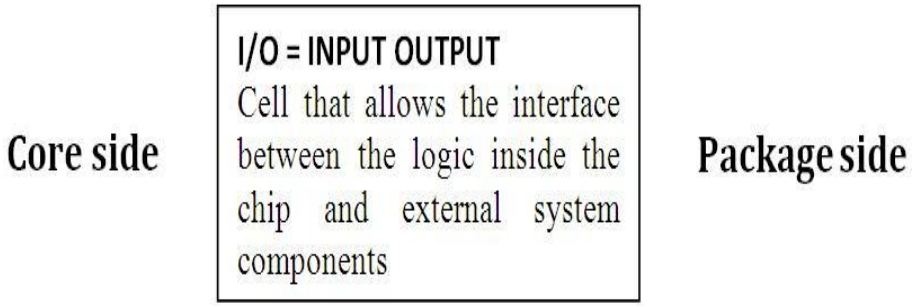
All this affects the quality and reliability of the system. Hence Regression Testing, since it aims to verify all this, is very important.

Chapter 2

IO Cells

2.1 What are I/O's?

- I/O is the specially designed element which interfaces to core signal to off chip environment.



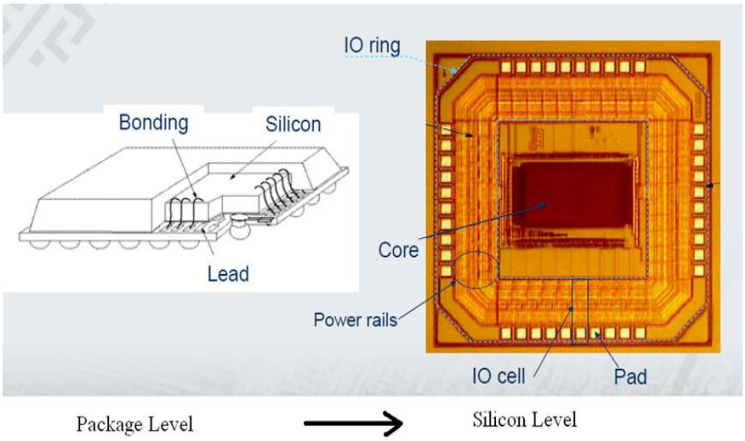


Figure 2.1: Silicon View at Chip Level

2.2 Why do I/O's need special attention? Why are they important?

- Any input signal which comes from off chip device has to be checked by the I/O for any discrepancy in its behavior other than the defined for the core and if it finds any characteristic of the signal which can damage the core, it either modifies the signal or simply rejects it. It also checks the signal going from core to the outside world. So I/Os are responsible for proper functioning of the entire chip. Thus however efficient the core design may be, it is the I/Os which determine the efficiency of the chip.

- It is necessary for the designer to analyze the designed I/O under the practical conditions to verify the deriving strength of the chip, delay in signal, power etc as they all are heavily dependent on the I/O irrespective of whether the core is compliant with the specifications or not. Because even a minor difference in the performance of the I/O than the desired one can damage the whole circuit or even can cause problems to the off chip circuit it? It is the responsibility of the I/O to limit the outgoing signaling in all respect like amplitude, frequency,

delay etc, under the specified one, for proper functioning.

2.3 A Typical I/O Buffer Contains...

- A typical bidirectional I/O consist of predrivers, slew rate controller, output stage, Receiver section, ESD protection circuit and I/O pad .All these blocks are very critical for proper functioning of the chip and off chip connected components and so they require special attention while designing.

2.4 Electrostatic Discharge

- Electrostatic Discharge, in short ESD, protection is very important to save the chip from unwanted voltages which gets developed at the pin due to some source coming in contact with the pin. These large accumulated charges can destroy the transistor, so a mechanism is needed which can effectively and quickly discharge this unwanted accumulated charge. And here again the importance of metal rings come into picture as they discharge accumulated charges and protects the chip. Care should be taken that there are no acute angles in the metal layers as they could be a potential areas of "hotspots".

2.5 Power Supplies

- Power supply is required to bias the circuit and, as the circuit consists of both the I/O and the core, therefore we have different power supply for these two. The primary goal of having different rails is to minimize the effect of the noisy output buffers. The power supply for core should be clean i.e.it should not contain any noise. I/O section is connected to the outside world where the signal is more prone to get diluted with noise; therefore the separation for the two power supply is must in those chips where even a slight amount of noise

can degrade the performance of core to a greater extent. In some chips, where the noise has little or no effect on the performance of the core, both I/Os and core can have the same power supply.

2.6 Place & Route Cells (Fillers and Corner Cells)

- In a packaged chip, all the I/Os are abutted to one another. Th metal power ring continuity is automatically ensured by abutment. But if there are free spaces between I/Os, filler cells must be used to fill up these empty areas. There are filler cells with different width which can be used to fill up the different sizes. It is advised to start the filler cell placement by the widest cell possible and then continue with decreasing sizes until the gaps are completely filled. Filler cells only contain the geometrical information of the metal rings at each level of metal.

- As the I/Os are not placed at the corner edges of the cell and in spite of having filled the gaps between I/Os, complete continuity of metal rings is not achieved unless and until the corner edges are filled by corner cells. They also have only the geometrical information of the metal rings. Both filler and corner cell serves the same purpose but they are placed at different positions in the packaged chip.

- So in short the purpose of filler cell and corner cell is to have:

  - Continuity of metal power rings which is responsible for Uniform Distribution of Power.
  - Electrostatic Discharge protection.
  - Nwell closing to prevent latch up.

2.7 PADS

- Pads are basically a sandwich of various metal layers used in the design. The pads consist of pins and metal connection on all sides to provide the power connection to both the Core and the I/O elements. Multiple power pads are often used to reduce the noise. The internal elements of the I/O circuit being connected to one power pad while the external elements , the circuit part which will have interface with the off chip elements being connected to a different power pad . All the power pads are finally shorted by a metal layer at the bonding pad. The noisier power pads, the one connected to the output transistor, are separated from the substrate to prevent the noise coupling through the substrate.

- Basic power pads used inside a chip are:
  - Input Power Pad
  - Output Power Pad
  - Tristate and Bidirectional pad

2.8 Types of I/Os & Specific Functions

- 3 different types of an I/O
  - Input
  - Output
  - Bi-directional

- Specific functions:
  - Pull-up,Pull-down
  - Hysteresis

- Analog
- Supplies

Chapter 3

IEEE 1149.1/JTAG Boundary Scan Standard

- In 1985, a group of European companies formed Joint European Test Action Group (JETAG).
- In 1990, the Institute of Electrical and Electronic Engineers (IEEE) refined the concept and created the 1149.1 standard, known as IEEE Standard Test Access Port and Boundary Scan Architecture

3.1 History

- 1985
  - Joint European Test Action Group (JETAG, Philips)
- 1986
  - VHSIC Element-Test & Maintenance (ETM) bus standard (IBM et al.)
  - VHSIC Test & Maintenance (TM) Bus structure (IBM et al.)
- 1988

  - Joint Test Action Group (JTAG) proposed Boundary Scan Standard

- 1990
  - Boundary Scan approved as IEEE Std. 1149.1-1990
  - Boundary Scan Description Language (BSDL) proposed by HP

- 1993
  - 1149.1a-1993 approved to replace 1149.1-1990

- 1994
  - 1149.1b BSDL approved

- 1995
  - 1149.5 approved

3.2 Overview of 1149 Family

Number	Title	Status
1149.1	Testing of digital chips and Interconnection between chips	Std. 1149.1-1990 Std. 1149.1a-1993 Std. 1149.1b-1994(BSDL)
1149.2	Extended digital Serial Interface	Near Completion
1149.3	Direct Access Testability	Discontinue
1149.4	Mixed Signal Test Bus	Started Nov. 1991
1149.5	Standard Module Test and Maintenance (MTM) Bus Protocol	Std. 1149.5-1995

Table 3.1: Overview of 1149 Family

3.3 Purpose of Standard

- This standard defines test logic that can be included in an integrated circuit to provide standardized approaches to
  - testing the interconnections between integrated circuits once they have been assembled onto a printed circuit board or other substrate
  - testing the integrated circuit itself and
  - Observing or modifying circuit activity during the component's normal operation.
- The test logic consists of a boundary-scan register and other building blocks and is accessed through a Test Access Port (TAP).

3.4 Basic Chip Architecture for 1149.1

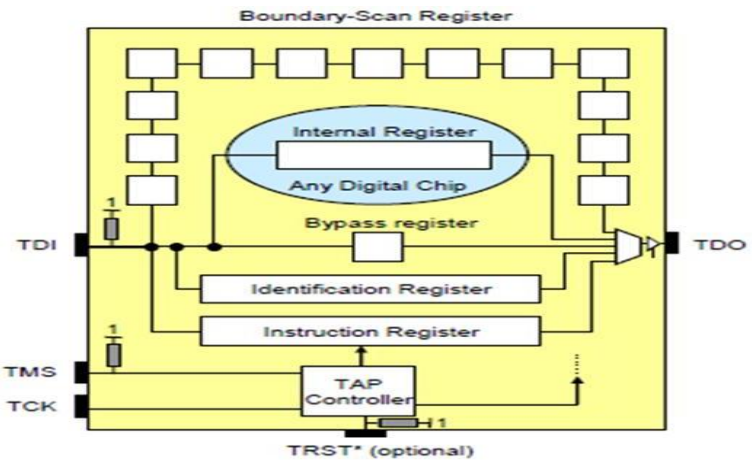


Figure 3.1: Basic Chip Architecture for 1149.1

- The top level schematic of the test logic defined by IEEE Std 1149.1 includes three key blocks:

- **The TAP Controller:**
  - This responds to the control sequences supplied through the test access port (TAP) and generates the clock and control signals required for correct operation of the other circuit blocks.
- **The Instruction Register:**
  - This shift register-based circuit is serially loaded with the instruction that selects an operation to be performed.
- **The Data Registers:**
  - These are a bank of shift register based circuits. The stimuli required by an operation are serially loaded into the data registers selected by the current instruction. Following execution of the operation, results can be shifted out for examination. There are three data registers.

    I. The Device ID Register (DIR) reads-out an identification number which is hardwired into the chip.

    II. The Bypass Register (BR) is a 1-cell pass-through register which connects the TDI to the TDO with a 1-clock delay to give test equipment easy access to another device in the test chain on the same board.

    III. The Boundary Scan Register (BSR), intercepts all the signals between the core-logic and the pins.

3.5 Boundary Scan Cell

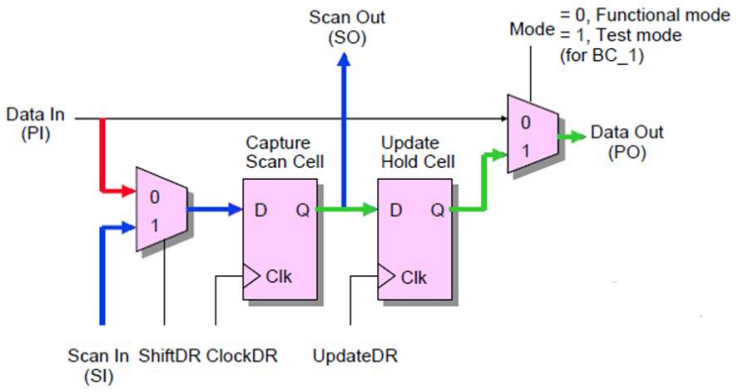


Figure 3.2: Boundary Scan Cell

3.6 Boundary Scan Cell: Operation Modes

1. **Normal:** Mode Control=0
   - PI -> PO
2. **Scan:** ShiftDR=1, ClockDR
   - TDI -> ... -> SI -> SO -> ... -> TDO
3. **Capture:** ShiftDR=0, ClcokDR
   - PI -> QA, OUT driven by PI or QB
4. **Update:** Mode Control=1, UpdateDR
   - QA -> PO

3.7 TAP Controller

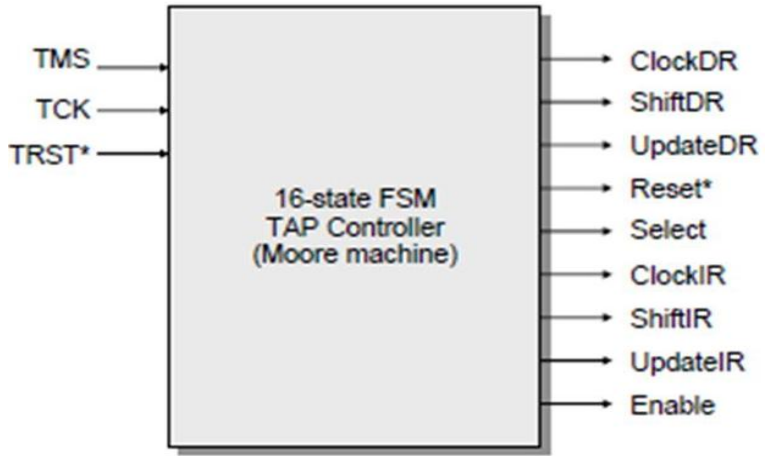


Figure 3.3: TAP Controller

3.8 TAP Controller: Signals

- **Test Clock Input (TCK):** Clock for test logic
- **Test Mode Select (TMS):** Switches system from functional to test mode.
- **Test Data Input (TDI):** Accepts serial test data and instructions used to shift in vectors or one of many test instructions.
- **Test Data Output (TDO):** Serially shifts out test results captured in boundary scan chain.
- **Test Reset (TRST*):** Optional asynchronous TAP controller reset.
- The other signals, Reset, Select and Enable are distributed as follows:
  - Reset is distributed to the Instruction register and to the target Data Register.
  - Select is distributed to the output multiplexer.
  - Enable is distributed to the output driver amplifier.

3.9 TAP Controller: Distribution of Signals

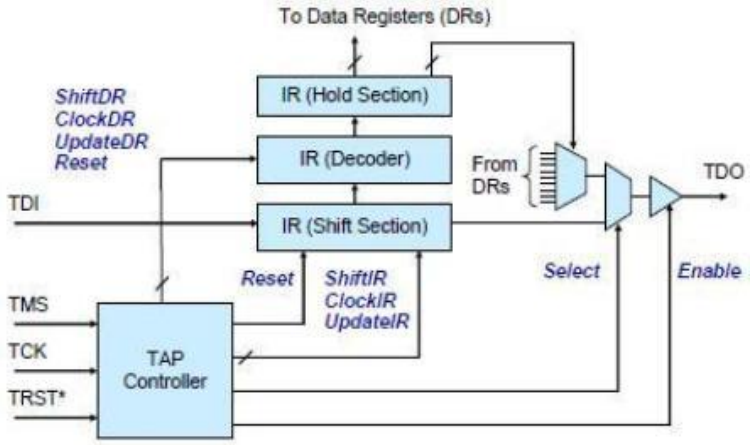


Figure 3.4: Distribution of Signals

3.10 TAP Controller: State Diagram

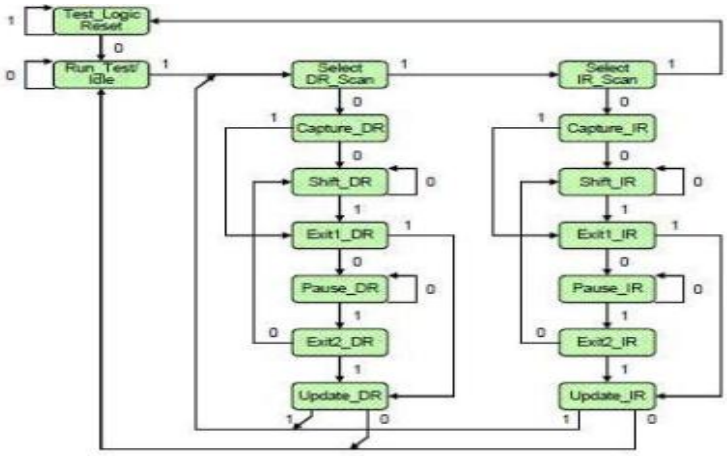


Figure 3.5: State Diagram of TAP Controller

3.11 TAP Controller: State Description

- **Test-Logic-Reset:** normal mode
- **Run-Test/Idle:** wait for internal test such as BIST
- **Select-DR-Scan:** initiate a data-scan sequence
- **Capture-DR:** load test data in parallel
- **Shift-DR:** load test data in series
- **Exit1-DR:** finish phase-1 shifting of data
- **Pause-DR:** temporarily hold the scan operation (allow the bus master to reload data)
- **Exit2-DR:** finish phase-2 shifting of data
- **Update-DR:** parallel load from associated shift registers

3.12 IEEE 1149.1 Standards: Instruction Set

- **EXTEST:**
  - Test interconnection between chips of board
- **SAMPLE/PRELOAD:**
  - Sample and shift out data or shift in data only
- **BYPASS:**
  - Bypass data through a chip
- **Optional:**
  - INTEST, RUNBIST, CLAMP, IDCODE, USERCODE, HIGH-Z, etc.

3.13 Boundary Scan Description Language (IEEE 1149.1b)

- **Purpose:**
  - This provides a standard language for Boundary Scan device.
  - To simplify a design work for Boundary Scan
    - Automatic synthesis is possible
  - To promote consistency throughout ASIC designers, device manufacturers, foundries, test developers and ATE manufacturers
  - For easy incorporation into software tools for test generation, analysis, and failure diagnosis
  - To reduce possibility of human error when employing boundary scan in a design.
- **Features of BSDL:**
  - Describes the testability features of boundary scan devices which are compatible with 1149.1
  - It's a subset of VHDL
  - Elements of a design which are absolutely mandatory for the 1149.1 and system-logic are included in the language
    - Example -BYPASS register, TAP controller, etc.
  - BSDL may be used in a full or a partial VHDL environment

Chapter 4

CreateChipIOLogic Tool

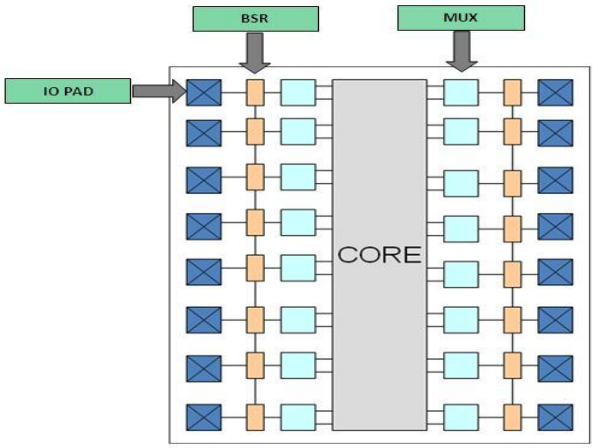


Figure 4.1: Generic Architecture of Chip

4.1 Purpose and Scope

- Generation of padring netlist
- Generation of Muxing & Demuxing logic on the output and input path respectively

- Insert of Boundary Scan Register
- Generation of BSDL view
- Glue insertion
- PSL based assertion to test the Muxing logic
- IPXACT view generation

4.2 CreateChipIOLogic: Flow

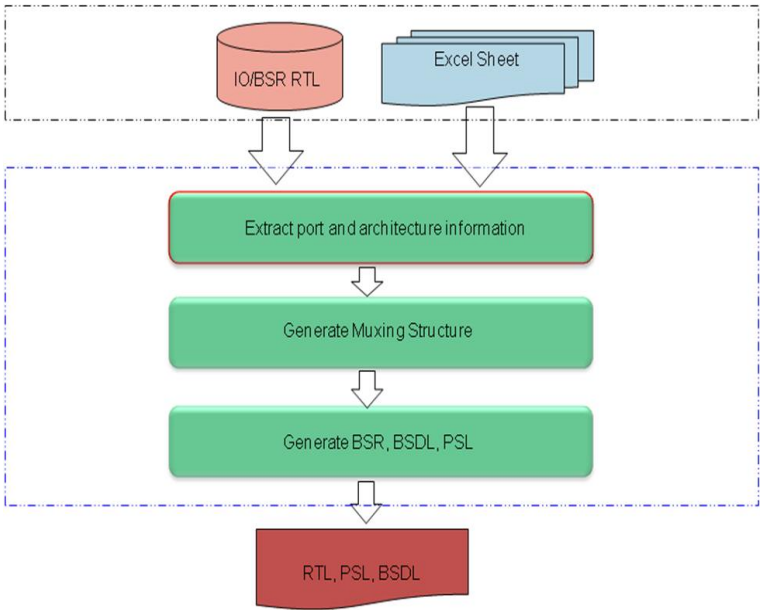


Figure 4.2: CreateChipIOLogic Flow

4.3 CreateChipIOLogic: Input Files

4.3.1 I/O and BSR Verilog File

- Can be obtained from libraries or manually created.

4.3.2 Excel Workbook

- Macro enabled environment provided with the tool to assist the designer in creation of input excel file.

I have Added new functionality for finding duplicate ports in Demuxing sheet of Input Excel Workbook by using VBA language.

4.4 CreateChipIOLogic: Generation Mechanism

- Generation mechanism can be sub-divided into
  - Tool configuration
  - Pad Ring generation
  - Output Muxing
  - Input Muxing
  - BSR insertion
  - Controlling Pad and BSR control signals
  - BSDL (Boundary Scan Description Language) generation
  - PSL generation
  - Hierarchy structure

4.5 CreateChipIOLogic: Output Files

4.5.1 Verilog File

- CreateChipIOLogic generate Verilog files for each hierarchy defined in sheet. CreateChipIOLogic also generate separate Verilog file for BSR and Top module to instantiate all other hierarchy and BSR module.

4.5.2 PSL Views

- Property Specification Language (PSL) is a language developed by Accellera for specifying properties or assertions about hardware designs. The properties can then be simulated or formally verified. Since September 2004 the standardization on the language has been done in IEEE 1850 working group. In September 2005, the IEEE 1850 Standard for Property Specification Language (PSL) was announced.

- Property Specification Language aims to be used with multiple electronic system design languages such as:

  - VHDL (IEEE 1076),
  - Verilog (IEEE 1364),
  - System Verilog (IEEE 1800), and
  - SystemC (IEEE 1666) by OSCI.

- PSL is a language for the formal specification of hardware. It is used to describe properties that are required to hold in the design under verification. PSL provides a means to write specifications that are both easy to read and mathematically precise. It is intended to be used for functional specification on the one hand and as input to functional verification tools on the other. Thus, a PSL specification is an executable documentation of a hardware design.

4.5.3 BSDL (IEEE 1149.1b) Views

- The BSDL (Boundary Scan Description Language) language allows description of the testability features in IEEE Std. 1149.1-1990 compliant devices. This language can be used by tools that make use of those testability features. Such tools include testability analysis, test generation and failure diagnosis. Note that BSDL itself is not a general purpose hardware description language.

With a BSDL description of a device and knowledge of the standard, it is possible for tools to completely understand the data transport characteristics of the device. With additional capabilities provided by VHDL, it is possible to perform simulation, verification, compliance analysis, and synthesis functions. Support for these functions is beyond the scope of BSDL alone.

4.5.4 Log File

- CreateChipIoLogic generates a log file. Log file contains information about errors and warnings.

4.6 Advantages of CreateChipIoLogic vs. BSR insertion by DC

- **Ease of use**
  - Insertion of customized JTAG is easier as compared to usage of default JTAG.
- **Limitations are easy to control at RTL level**
  - For example some timing problems are easy to handle at RTL level while inserting BSR.
- **Architecture Control**
  - Full control on type of cell, structure and hierarchy.
- **Reduced Effort**
  - No need to specify the IO specification again to DC as single sheet handles Muxing, padring and BSR insertion.

– No need to wait till synthesis of core for BSR insertion as Muxing, padring and BSR insertion can be handled in parallel even if core is not complete.

4.7 Configurable Muxing Architecture

- CreateChipIOLogic implements simple NxN priority Muxing for modes.
- There is three type of Muxing structure is supported as shown below.

4.7.1 Simple Muxing Structure

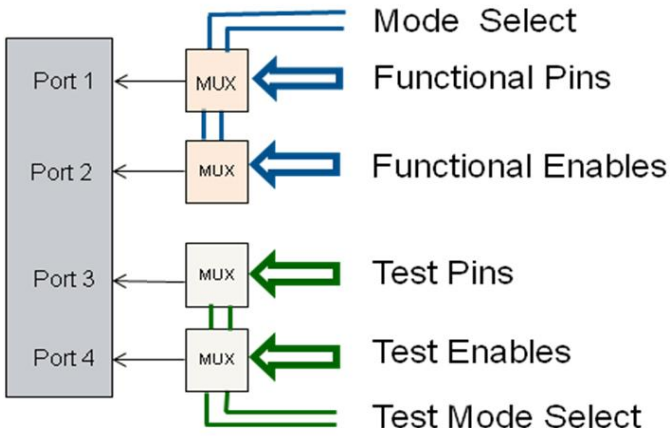


Figure 4.3: Simple Muxing Structure

- There is only one Mux between any port of pad and Core in simple Muxing structure as shown in Fig. 4.3

4.7.2 Muxing Chain

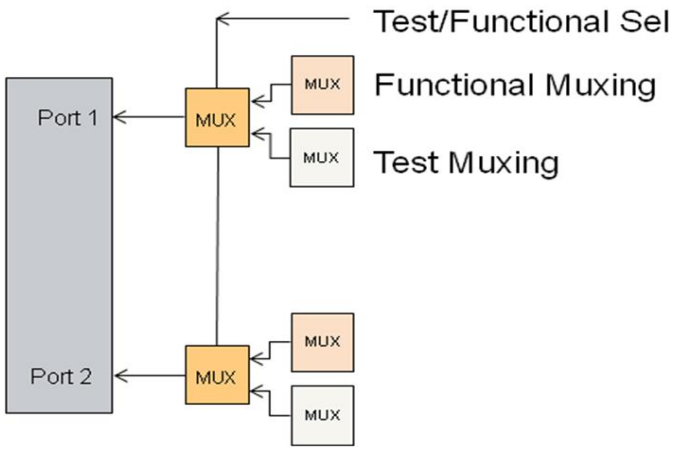


Figure 4.4: Muxing Chain

- There are more than one Muxes between any port of pad and Core in Muxing Chain structure as shown in Fig. 4.4 and also Multiple Mux can be added to Select lines of Mux.

4.8 Supported Features

4.8.1 BSR insertion at RTL level

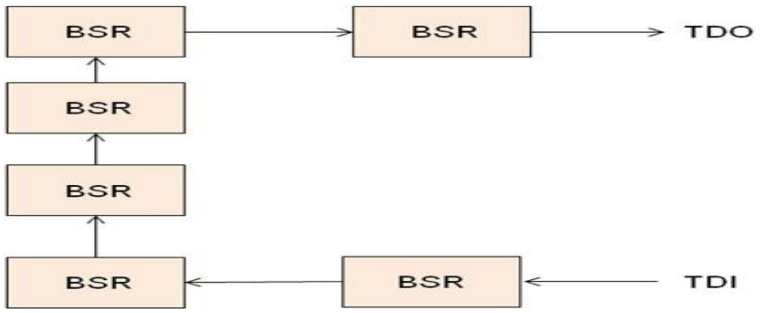


Figure 4.5: BSR Chain

- User defined BSR cells in Excel Customizable chain order.

- BSR cell is inserted in between PAD cell and Mux/Demux/Core where pad inputs and output ports getting connected with Mux/Demux/Core are connected on the boundary of BSR register.

4.8.2 Dedicated excel based control for pad special signals

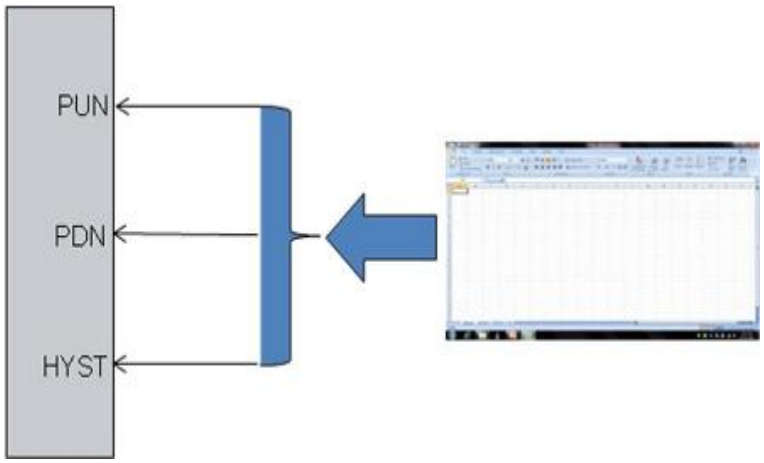


Figure 4.6: Pad Special Signal

- Enables better pin to pin control for pad signals.
- Pad ports can also be tied globally, as well as in groups.

4.8.3 Glue insertion and IPXACT component view generation

- Supported for insertion of combinational glues directly from excel sheet.
- Act a good container for last minutes changes.

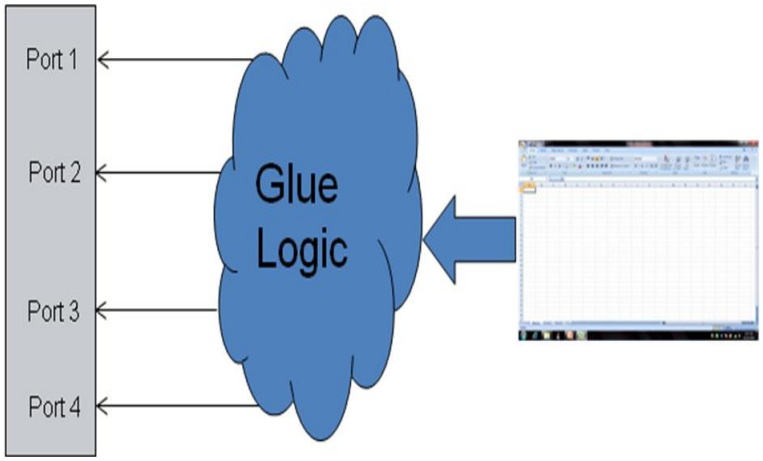


Figure 4.7: Glue Insertion

4.8.4 Multiple Logical Hierarchy

- CreateChipIOLogic support three type of hierarchy structures.

4.8.4.1 Hierarchy Type I

- CreateChipIOLogic tool generate only single Verilog file for IO ring and IO Muxing logic in Hierarchy Type I. If BSR is inserted, it generate separate BSR Verilog file.

4.8.4.2 Hierarchy Type II

- CreateChipIOLogic tool generate one Verilog file for IO ring and it generate Verilog files for each Mux Mode in Hierarchy Type II. If BSR is inserted, it generate separate BSR Verilog file.

4.8.4.3 Hierarchy Type III

- CreateChipIOLogic tool generate one Verilog file for IO ring and it generate Verilog files for each Mux Mode and that all Mux Modes module is define in

one Verilog file in Hierarchy Type III. If BSR is inserted, it generate separate BSR Verilog file.

4.8.5 Support to STM Projects

- I provided support to different division for different projects in IO ring generation activity.

Chapter 5

Regression Testing

5.1 What is Regression Testing?

- If a piece of CAD tool code is modified for any reason testing needs to be done to ensure that it works as specified and that it has not negatively impacted any functionality that it offered previously. This is known as Regression Testing.
- Regression Testing attempts to verify:
  - That the application works as specified even after the changes/additions /modification was made to it.
  - The original functionality continues to work as specified even after changes /additions/modification to the CAD tool application.
  - The changes/additions/modification to the CAD tool application has not introduced any new bugs.

5.2 When is Regression Testing necessary?

- Regression Testing plays an important role in any Scenario where a change has been made to a previously tested tool code. Regression Testing is hence an

important aspect in various Software Methodologies where tool code changes enhancements occur frequently.

- Any CAD tool Development Project is invariably faced with requests for changing Design, code, features or all of them.
- Some Development Methodologies embrace change.
- For example 'Extreme Programming' Methodology advocates applying small incremental changes to the system based on the end user feedback.
- Each change implies more Regression Testing needs to be done to ensure that the System meets the Project Goals.

5.3 Why is Regression Testing important?

- Any tool code change can cause existing functionality to break. Changes to a tool component could impact dependent Components.
- It is commonly observed that a tool code fix could cause other bugs.
- All this affects the quality and reliability of the system. Hence Regression Testing, since it aims to verify all this, is very important

5.4 Regression Testing - What to Test?

- Since Regression Testing tends to verify the tool application after a change has been made everything that may be impacted by the change should be tested during Regression Testing. Generally the following areas are covered during Regression Testing:
  - Any functionality that was addressed by the change
  - Original Functionality of the tool

– Performance of the tool after the change was introduced

5.5 Regression Testing - How to Test?

- Like any other Testing Regression Testing Needs proper planning. For an Effective Regression Testing to be done the following ingredients are necessary.

5.5.1 Create a Regression Test Plan

– Test Plan identified Focus Areas, Strategy, Test Entry and Exit Criteria. It can also outline Testing Prerequisites, Responsibilities, etc.

5.5.2 Create Test Cases

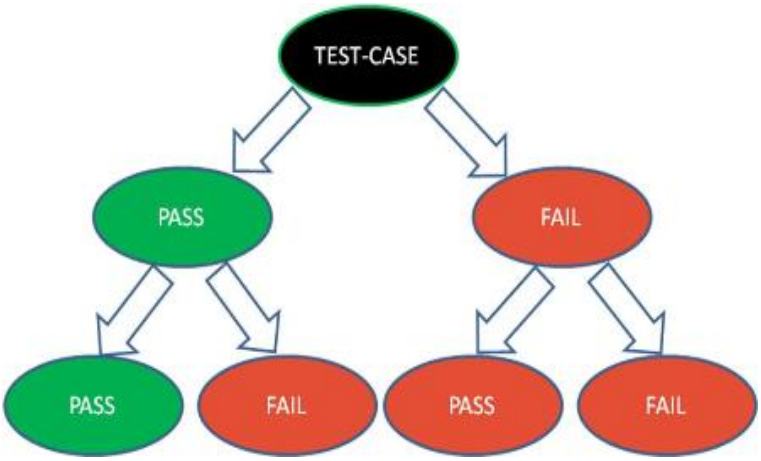


Figure 5.1: Test Case Flow

– Test Cases that cover all the necessary areas are important. They describe what to Test, Steps needed to test, Inputs and Expected Outputs. Test Cases used for Regression Testing should specifically cover the functionality addressed by the change and all components affected by the

change. The Regression Test case may also include the testing of the performance of the components and the application after the change(s) were done.

5.5.3 Error Tracking

– As in all other Testing Levels and Types it is important Errors are tracked systematically, otherwise it undermines the Testing Effort. Error tracking is done by ncsim, Formality and Design Compiler tool.

5.5.3.1 Ncsim (Cadence):

* Check syntax of verilog file (generated by CreateChipIOLogic) by using ncsim (Cadence tool).

5.5.3.2 Formality (Synopsys):

* Formality is an application that uses formal techniques to prove or disprove the functional equivalence of two designs or two technology libraries. For example, you can use Formality to compare a gate-level netlist to its register transfer level (RTL) source or to a modified version of that gate-level netlist. After the comparison, Formality reports whether the two designs or technology libraries are functionally equivalent. The Formality tool can significantly reduce your design cycle by providing an alternative to simulation for regression testing.
* The techniques Formality uses are static and do not require simulation vectors. Consequently, for design verification you only need to provide a functionally correct, or "golden," design (called the reference design) and a modified version of the design (called the implementation design). By comparing the implementation design against the reference design, you can determine whether they are functionally

equivalent to each other. Technology library verification is similar except that each cell in the implementation library is compared against each cell in the reference library one cell at a time.

* Todays design methodology requires regression testing at several points in the design process. Currently, traditional simulation tools, such as event-driven and cycle-based simulators, handle this regression testing. However, as designs become larger and more complex and require more simulation vectors, regression testing with traditional simulation tools becomes a bottleneck in the design flow. The bottleneck is caused by these factors:
  - Large numbers of simulation vectors are needed to provide confidence that the design meets the required specifications.
  - Logic simulators must process more events for each stimulus vector because of increased design size and complexity.
  - More vectors and larger design sizes cause increased memory swapping, slowing down performance.
* Formality fits well within established electronic design automation (EDA) methodologies used to create large-scale designs, because it can replace the traditional simulation tools used for regression testing. This replacement, combined with the continued use of static timing analysis tools, gives you two distinct advantages: significantly reduced verification times and complete verification.
* Reduced verification times occur because Formality requires no input vectors. Reducing gate-level simulation time means you can spend more time verifying the initial golden RTL design to get better functional coverage. Formality maintains that coverage through all subsequent regressions.
* Complete verification, the second advantage, means 100 percent equiv-

alence over the entire vector space. Complete verification is significant because you no longer have to compromise the subset of vectors for gate-level simulation.

* The following list summarizes the Formality features:
  - Proves two designs or technology libraries are functionally equivalent, faster than verification using event-driven simulators
  - Provides complete verification (not vector-dependent).
  - Performs RTL to RTL, RTL to gate, and gate to gate design verifications.
  - Performs automatic hierarchical verification.
  - Verifies a wide range of design transforms or modifications, including pipeline retiming and reencoded finite state machines.
  - Offers schematic views and isolated cone of logic views that support location of design discrepancies.

5.5.3.3 Design Compiler (Synopsys):

* The Design Compiler tool is the core of the Synopsys synthesis products. Design Compiler optimizes designs to provide the smallest and fastest logical representation of a given function. It comprises tools that synthesize your HDL designs into optimized technology-dependent, gate-level designs. It supports a wide range of flat and hierarchical design styles and can optimize both combinational and sequential designs for speed, area, and power.The resulting gate-level netlist is a completely structural description with only standard cells at the leaves of the design. Internally, a synthesis tool performs many steps including high-level RTL optimizations, RTL to unoptimized boolean logic, technology independent optimizations, and finally technology mapping to the available standard cells.

* Synopsys provides a library called Design Ware which includes highly optimized RTL for arithmetic building blocks. For example, the Design Ware libraries contain adders, multipliers, comparators, and shifters. DC can automatically determine when to use Design Ware components and it can then efficiently synthesize these components into gate-level implementations.
* Design Compiler is used for logic synthesis, which is the process of converting a design description written in a hardware description language such as Verilog or VHDL into an optimized gate-level netlist mapped to a specific technology library.

Chapter 6

Conclusion

Manual (Human) design can occur with small no of transistors. As no of transistor increase through SSI and VLSI, the amount of evaluation and decision making would become overwhelming. So, Computer-aided Design (CAD) Tool automates this whole process and it reduce TTM. CAD tools allow large problem to be solved. After modifying any CAD tool code, it can cause existing functionality to break. Changes to a tool component could impact dependent components. It is commonly observed that a tool code fix could cause bugs, so regression testing is necessary. Regression testing of CreateChipIOLogic tool needs to be done to ensure that **(i)** there are no any syntax errors in generated Verilog file. **(ii)** Generated output logic is correct or not according to their inputs. **(iii)** Generated RTL code is synthesizable or not.

References

[1] Ken P. Parker, "The boundary-scan handbook: analog and digital", *Kluwer Academic Publishers*,2003.

[2] ST internal documents

[3] http://www.ee.ic.ac.uk/pcheung/teaching/ee3_DSD/ti_jtag_seminar.pdf

[4] www.wikipedia.com