Development of CAD Checks for Validation of Library Views

Submitted By

Ankitkumar Narsinhbhai Chaudhari

14MCEC04

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2016

Development of CAD Checks for Validation of Library Views

Major Project

Submitted in fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering

Submitted By

Ankitkumar Narsinhbhai Chaudhari

(14MCEC04)

Guided By

Prof. Rupal A Kapdi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2016

Certificate

This is to certify that the major project entitled **"Development of CAD Checks for Validation of Library Views"** submitted by **Ankitkumar Narsinhbhai Chaudhari (Roll No: 14MCEC04)**, towards the fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering of Institute of Technology, Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Dr. Sanjay Garg
Professor and Head,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Prof.P. N. Tekwani
Director,
Institute of Technology,
Nirma University, Ahmedabad

Dr.Priyanka Sharma
Coordinator M.Tech-CSE,
Computer Science Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Prof. Rupal Kapdi
Assistant Professor,
Computer Science Engineering,
Institute of Technology,
Nirma University, Ahmedabad

CERTIFICATE

This is to Certify that the Major Project Report **Development of CAD Checks for Validation of Library Views** submitted by **Chaudhari Ankitkumar (14MCEC04)** as the fulfillment of the requirements for the degree of Master of Technology in Computer Science and Engineering from Institute of Technology, Nirma University is the record of work carried out by him under my supervision. The work submitted in our opinion has reached a level required for being accepted for the examination and it is pure research based as this project is related to the enhancement and development.

Date:

Project Manager

Mrs. Jyoti Kumar

TR&D Department

STMicroelectronics, India

Statement of Originality

I, **Ankitkumar Narsinhbhai Chaudhari**, Roll. No. **14MCEC04**, give undertaking that the Major Project entitled "**Development of CAD Checks for Validation of Library Views**" submitted by me, towards the fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Signature of Student

Date:

Place:

Endorsed by

Prof. Rupal A Kapdi

(Signature of Guide)

Acknowledgements

It gives me tremendous joy in communicating thanks and significant appreciation to my guide **Prof. Rupal kapdi**, Associate Professor, Computer Science Department, Institute of Technology, Nirma University, Ahmedabad for her significant direction and nonstop support all through this work. The gratefulness and nonstop bolster she has granted has been an extraordinary inspiration to me in coming to a higher objective. Her direction has activated and sustained my scholarly development that I will profit by, for quite a while to come.

It gives me an immense pleasure to thank **Dr. Sanjay Garg**, Hon'ble Head of Computer Science and Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Prof.P. N. Tekwani**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation he has extended throughout course of this work.

It gives me an immense pleasure to thank **Mrs Jyoti Kumar** Manager, ST Microelectroniocs, Greater Noida. I exploiting her vast information and great future vision and owe her heaps of appreciation for essentially affecting this report.

I would like to thank my Mentor, **Mr. Rishabh Bansal** , ST Microelectroniocs, Greater Noida for her valuable guidance. Throughout the training, he has given me valuable advice on project work. he has given me all kind of support to handle complex situation.

I would like to thank you to my Institution, all faculty members of Computer Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

- Ankitkumar Narsinhbhai Chaudhari

14MCEC04

Abstract

Electronic gadgets that we are utilizing as a part of our everyday life are made out of several chips, which is fixed inside these gadgets for their appropriate working. This chip is a combination of different programmable programmable logic gates, memories , registers and latches.Library is an accumulation of cells and Logic gates that are fabricated onto the chip.

The project is related to automation of validation of libraries which are the Intellectual Property. These libraries represent design data of cells, transistor level design and timing information, actual mask level design that will be integrated and fabricated on a chip. The automation requires plugins to validate different views of a library under test. These plugin check different views of library. The plugins are in TCL, C Shell.

Abbreviations

CDL	Circuit Design Language.
GDS	Graphical Design Stream.
LVS	Layout Vs Schematic.
DRC	Design Rule Check
LEF	Library Exchange Format
TCL	Tool Command Language
VLSI	Very Large Scale Integration
CAD	Computer-aided Design
V	Verilog

Contents

List of Figures

Chapter 1

Introduction

1.1 General

Design team developes IP.System on chip designers combine all the Ips on the single chip.One of the demanding jobs for IP providers is related to IP models or representation needed in the design are targeted for the System on Chip(SOC) development.[1] The main issue for IP developers are[**?**]

- Coherency between different Library Views.
- Accuracy between IP models and IP available in market or developed by design team.

This project aims at providing a validation solution which reports mismatches or modelling errors for Libraries and IP that can seriously delay an IC design project[]

This project has two phases previously the libraries were validated using software called Ipscreen . New Enhancement phase is now going on in which instead of validating whole library one can validate individual views of the library.

1.2 Objective of Study

The increasing number of components from different suppliers and the amount of process corners that need to be covered lead to an explosion in both data-volume and data-variety.

This project aims at providing a validation solution which reports mismatches or modelling errors for Libraries and IP that can seriously delay an IC design project.

This validation solution can help in achieving a working design within a predictable flow by ensuring that every design step is validated and every imported IP-component is qualified.Also helps CAD teams and IC designers achieving a high quality of design data in a short time.

1.3 Motivation

These plugins bundle collection of checks. Prior to automation, there were two ways in which the validation was done

- Some checks required the user to manually check all the relationship between various different views.
- By giving input to EDA tool through command line or GUI, the results had to be collected, and analyzed for each cell. This would take weeks and the results of validation may still be prone to human errors.

Chapter 2

Literature Survey

2.1 Introduction of Library IP

Library is characterized as an arrangement of all outline information accessible for an IP.[2] The configuration information

comprises of transistor level configuration of the IP, timing data that will be fabricated on chip. We can likewise say that library is an accumulation of cells, involving of different perspectives which are helpful for outlining a chip. Cell is a segment performing an essential capacity and a perspective is a specific representation of a cell. Because of expanded unpredictability of circuit and shorter time to market SoC planners can't generally focus on outline of essential building pieces. This database can be reused in outlining different Framework on Chip(SOC).[3]

2.2 Types of Library

The distinctive sorts of Library are as per the following:

2.2.1 Core Library

It comprises of a gathering of cells called Standard Cells. They execute essential rationale capacities like Inverter, hooks and ip operations and so on. Advanced outlines are assemble from fundamental parts like gate, register, counters, viper, sub-tractors, RAM, ROM. Center Library is accumulation of such building squares. Physical/logical/timing models are made for these cells.[4]

2.2.2 Input/Output Library

It comprises of a group of cells called I/O buffers. I/O buffers are intended to interface of chip design to inside chip environment and vice-versa . I/Os are placed on the periphery of the chip. Any signal which originates from chip environment (outer voltages are at a run of voltage of 2.5V, 3.3V or 5V) into the chip, must be checked by I/O for any error in its conduct other than characterized by the center for that specific sign. In the event that I/O finds any signal challenging the normal conduct from it, it alters the signal so as to guarantee appropriate working of chip.

2.2.3 Memory Library

These library contain memory of diverse structural planning. Samples are SRAM, DRAM, ROM. As there can be number of memory sizes, we execute fundamental building square of memory and design the era of diverse memory sizes.

2.2.4 Analog and Mixed Cell Library

These library are executed in a full custom or semi custom way utilizing the CORE library. Illustration of Analog and Mixed Signal Library are Digital to Analog Convertor, U.S.B, Phase Locked Loop(PLL).

2.3 What are Views

Specific representation of a cell might have a layout view, schematic view, symbolic view , Abstract view.It can be classified as

- Front End Views
- Backend Views

  These views are required at different stages in RTL to GDSII flow. The information these views have is been converted into the respective ASCII or Binary format which is the more abstract and precise information that is used by the CAD tools.

As examined beforehand, Library is a collection of cells and every cell has views which are the representation of the cell. Every one of the cells have Layout view, Abstract perspective, Schematic perspective, Symbolic perspective, Timing perspective and so forth. A cell is conveyed as an arrangement of perspective and every perspective is

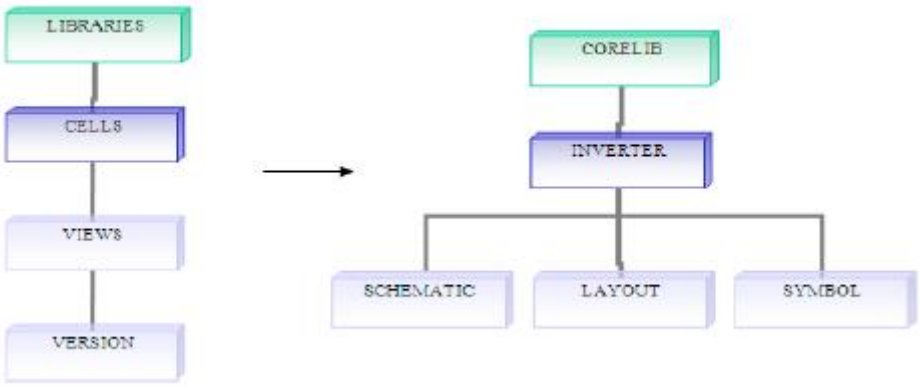

Figure 2.1: Library View Block Diagram

utilized by distinctive instrument as a part of a given electronic outline etc.The essential Library perspectives are as:

2.3.1 Symbolic View

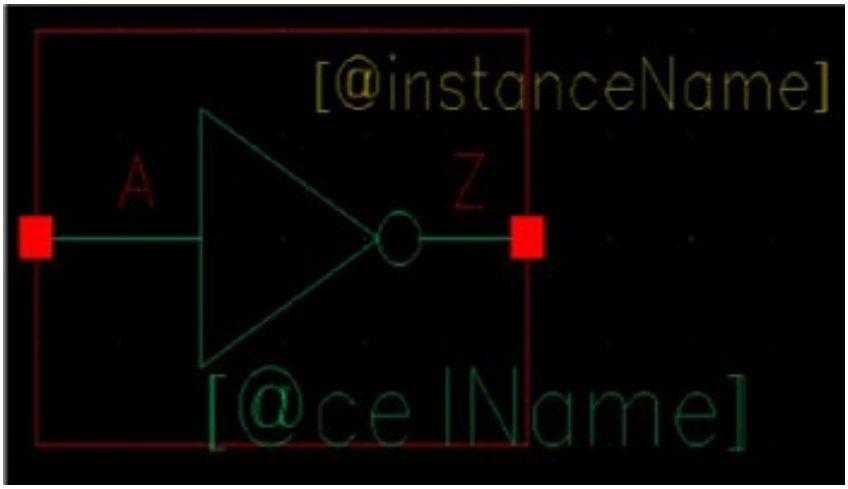

Figure 2.2: Symbolic View Of a Inverter

It is the pictorial representation of cell. It incorporates pins, image, names, choice box. Pins speak to the information and yield of the cells. The state of cells demonstrate its capacity. Names are mapped to some documentation of outline, determination box select the complete range for the cells. Typical Views permits the client to digest a mind boggling Schematic View and supplant it by a Symbolic perspective that can be utilized as a part of further plans. An image view for a NOT entryway is demonstrated as follows:

2.3.2 Circuit Description Language(CDL)

It is the literary representation of Schematic View at Transistor level. Making a subcircuit permits you to reuse the circuit different times in an outline and in future designs.

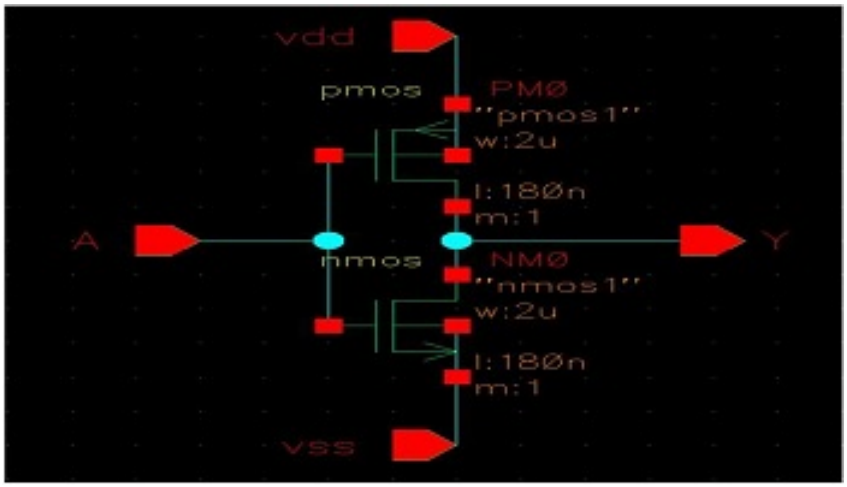

Figure 2.3: Schematic View of a Inverter as in Virtuso

Sub-circuits are like subroutines in programming. It is representation of circuit at transistor level.It is consequently produced from schematic perspective. It is utilized as a part of Design versus Schematic Check and Electronic Circuit Simulation.

2.3.3 Lef View (Library Exchange format)

An ASCII information design document, to portray physical format of an Integrated circuit. It incorporates outline governs and conceptual data about cells.

LEF has just the essential data required by the CAD device. It gives just a theoretical view and devours less capacity overhead. A LEF document contains the accompanying segments:

- Technology section: layer, Placement and Design Rules, Via definitions, metal capacitance.
- Macros Cell definitions: cell descriptions, cell dimensions, layout of pins and blockages, capacitances. These two sections can be stored in two different file if the size of file becomes large. While reading the lef view the technology file must be read first before reading Macro Cell definitions.

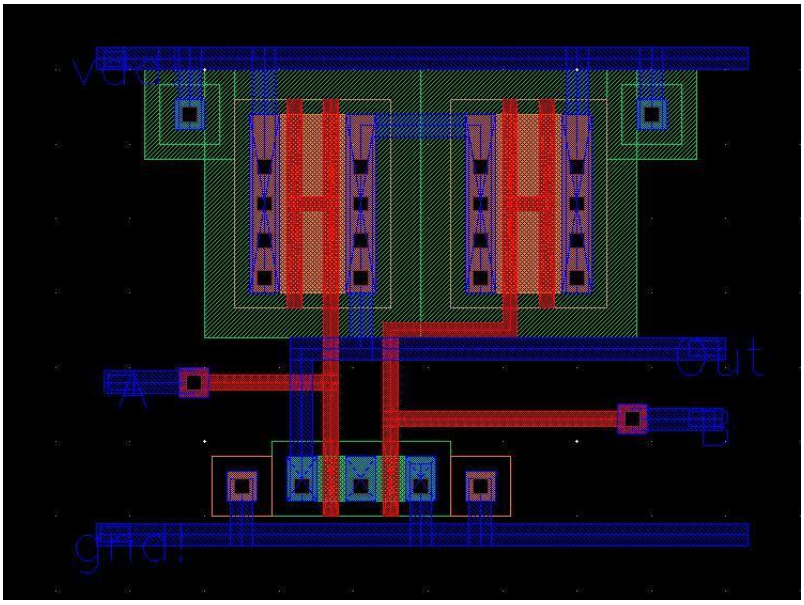

Figure 2.4: Layout View

2.4 Structure of Library

The library is kept up by a list document .The structure of record document incorporates different subsections that all together list the library documents in view of different conditions.

- Header: This section includes the library name, product name, process like 65nm, 45nm, 40nm etc, type of library example memory, standard cell, input-output etc.
- Cell: This section includes various cells.
- Conditions Section: This section includes conditions based on parameters like Process Variation(PV), Voltage(V), Temperature(T).
- Index Section: The index section includes paths to various cells based on different conditions. IPScreen parses this section to access the different views of the library.

2.5 IC Design Flow

The steps involved in IC design are as follows:

- Design Specification: It is the first step in IC design, here the design functionality is stated. All the requirements are clearly stated in terms of performance, speed, power, functionality. All the architectural part is stated clearly.

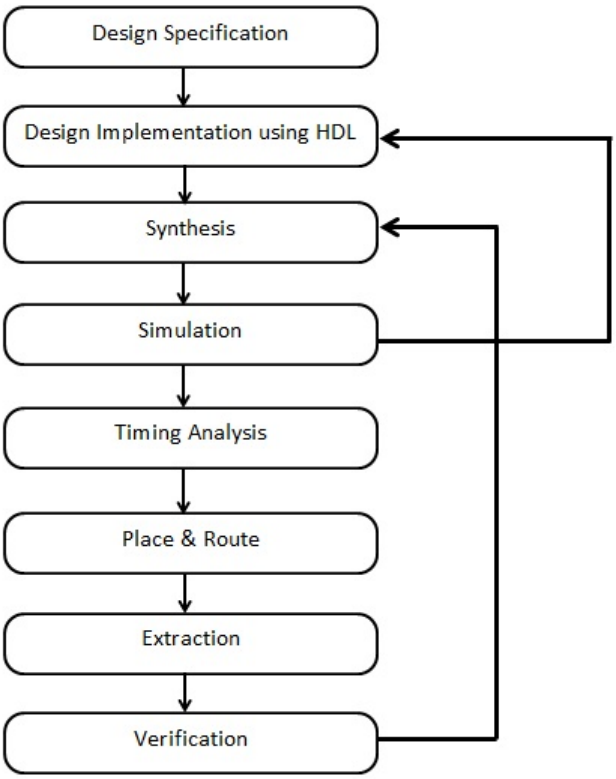

Figure 2.5: IC Design Flow

- Design Implementation using HDL: Hardware description Language allow to implement a design without going into much architecture, simulate and verify the output.For example Rather than building a MUX in hardware, Verilog code allows us to verify the functionality.

- Synthesis: A Register Transfer Language(RTL) is transformed into design implementation in terms of logic gates, using program called as synthesis tool. Examples of synthesis tool are Synopsys's Design Compiler and Candence's Ambit. At the end of this stage we have logic circuit in terms of gates and memories. The output of Synthesis is netlist. Netlist indicates all the devices and interconnection between them.

- Simulation: This netlist is simulated to verify the functionality of gate level implementation of design.

Chapter 3

Validation Methodology

Validation is one of the major step that certifies the quality of the product being provided to the user. Conventional validation approach has high cycle time and also requires high man resources so there came a need to go for Hybrid validation approach.

3.1 Conventional Validation Approach

First of all start with introducing the conventional approach where complete RTL to GDS flow is required to validate an IP. This is a very tedious task requiring a large utilization man power resources as well.

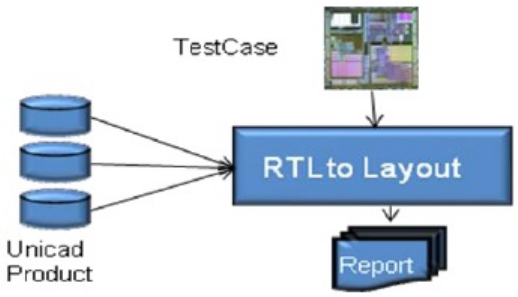

Figure 3.1: The Conventional Validation Approach

3.2 Hybrid Validation Approach

The new hybrid approach has various benefits over the conventional one such as

- Easy to debug
- Increased usability
- Increased coverage
- Covers different types of IP milestone
- New views with new CAD features can be implemented

Figure 3.2: The New Hybrid Validation Approach

3.3 Creating A Validation Environment

Increased complexities of the design has forced to opt an efficient design methodology and effective IP validation techniques. For effective validation we need to create a validation environment which have following requirements

- Checks for each of the view.
- Validation process definitions.
- Validation environment tools.
- General tools.

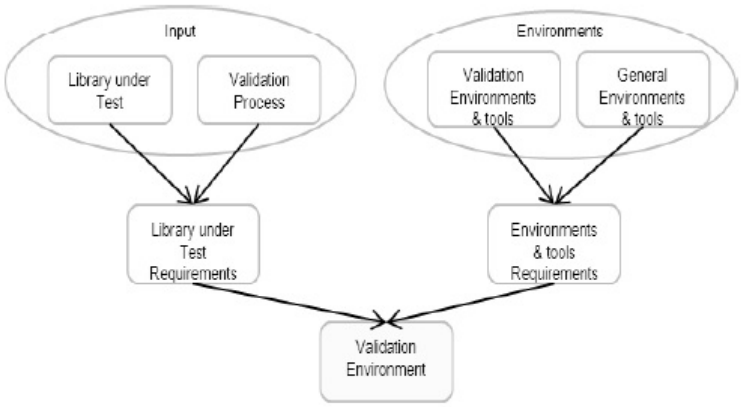

Figure 3.3: Validation environment requirement composition

Knowledge of the Library under Test (LUT) that needs to be tested from validation Environment includes structure information, modelling information and several other attributes. The knowledge mentioned is explained through the figure as knowledge chain for validation Environment designing. The concept of chain of knowledge refers to a network where the knowledge flows. Similar concept is applied to validation environment and three network nodes are defined

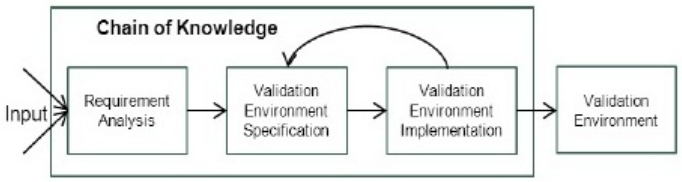

Figure 3.4: Validation environment chain of knowledge

3.3.1 Requirement Analysis

The first step is to extract the information that is relevant from the listed below sources

- Pareto analysis of bugs that are found in IP.
- Design methodologies that are dependent on EDA tools Flow requirements (view/attributes).
- User requirements.

3.3.2 Validation Environment Specification

As per the taxonomy checks were implemented in the plugins from the requirements. Validation environment specs says requirements should be converted into specifications and algorithms should be defined for modelling the checks so is the implementation

3.3.3 Validation Environment Implementation

This actually refers the implementation/modelling of the checks in the plugins for validating the IP. Based on the present taxonomies specs derived from the requirements were categorised among various plugins that is Modelization, Crosscheck, Syntax, Tag Checker .

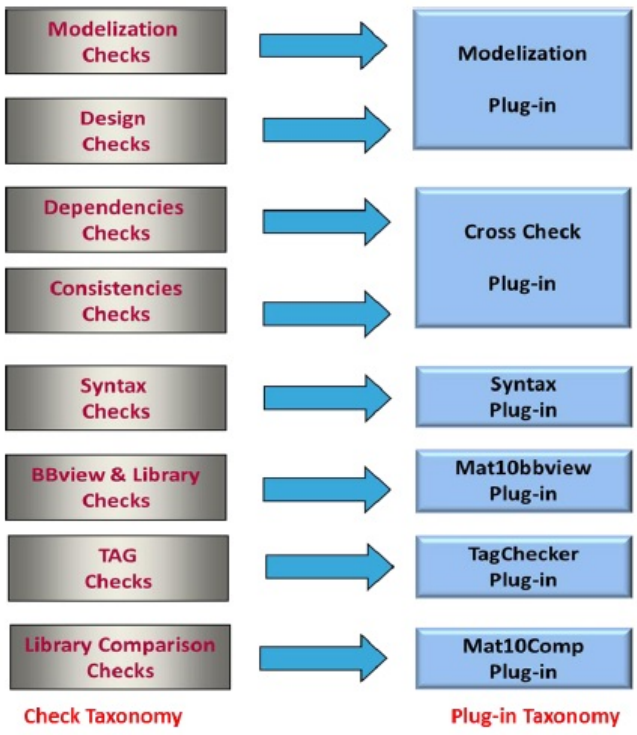

Figure 3.5: Mapping between Check Taxonomy and Plug-in Taxonomy

3.4 Validation Environment Architecture

This is three layer architecture within a framework which provides the interface to the user. Three layers are

- Plugins
- Rules for checks that are static
- Tools and kits such as EDA tools, Techno Kits, Parser Kits and Design Kits

There exists a bidirectional interaction between the first layer and plugin layer as well as between third layer and plugin layer. This bidirectional communication allows the validation environment to use various kits and EDA tools for Adaptive Methodology flow keeping in mind that rules applied are specific to standard technology and Design Platform. Communication of plugins and third layer has an additional advantage of instantiating the kits and EDA tools for every specific view present in the IP library.

Database of IPs which needs to be validated is given as input to this framework where some pre-processing is done based on the IP style to create a desired environment specific to that particular IP. After the environment is ready different views present in the package are analysed by the analysers which are responsible for parsing and dumping the view contents in a standard format. These parsers have been implemented in Tclsh, Perl and csh scripting languages. Now the static checks are being performed on the dumped standard format files following the rules or specifications ensuring the correctness as well. As an output a status report is being produced for each of the checks performed in the plugin for a particular IP

3.5 Validation Process Flow

The validation flow ensures the user that the IP under test is correctly validated. To make things simple to understand lets discuss on a single view

- In the first step, presence of particular view in the package is checked. Another thing that is being checked is that indexation of the view is properly done in the file named vc.bbview or not. If this file is not populated correctly like the view is not indexed correctly then as a resultant that particular view cannot be taken by the EDA tools in RTL to GDS flow. This is the check performed by Mat10bbview plugin
- In the second step, syntax checking is done where the view is being read by the respective EDA tool. In case of any incorrectness in the view the tool will not be

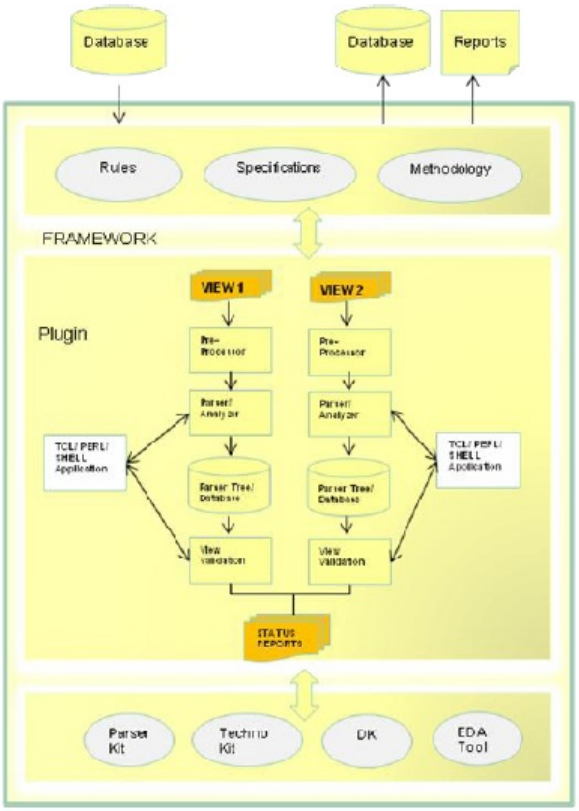

Figure 3.6: Mapping between Check Taxonomy and Plug-in Taxonomy

able to process that particular view in the RTL to GDS flow. This check is done by syntaxCheck plugin

- Once the view is checked syntactically the next step is to check that the view is having the necessary attributes which are required in RTL to GDS flow by the EDA tool. If they are found in the view then their values are being cross verified so that they are modelled according to the design rules mentioned in the kits of third layer (Validation Architecture). These two checks are being performed by Modelization plugin.int

- In the fourth step, it is ensured that information is consistent between various views thatâĂŹs what the check is called as cross view Consistency. Using RTL to GDS tools of all the different vendors the view contents are verified in terms of data consistency. This is done by CrossCheck plugin

- It is also being ensured that the tags mentioned in the layout are compliant with the convention. This check is done by a plugin named TagChecker

- Mat10Comp plugin is responsible for checking out the difference between the pre-

vious versions and the existing version of the IPs

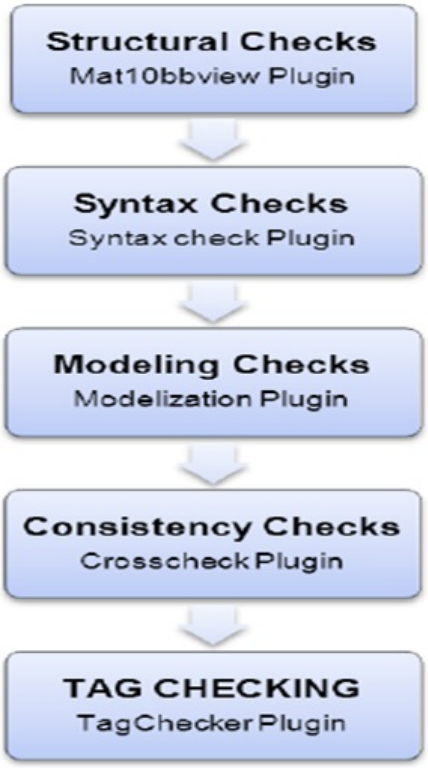

Figure 3.7: Mapping between Check Taxonomy and Plug-in Taxonomy

3.6 Reduction In Cycle Time

This validation environment has made it possible to reduce the cycle time to a large extent. Talking about any specific IP which generally takes 7-8 man days for validation following the conventional approach now can be validated only in a single man day. So this alternate approach has made a gain of 86 percentage in run time for validating

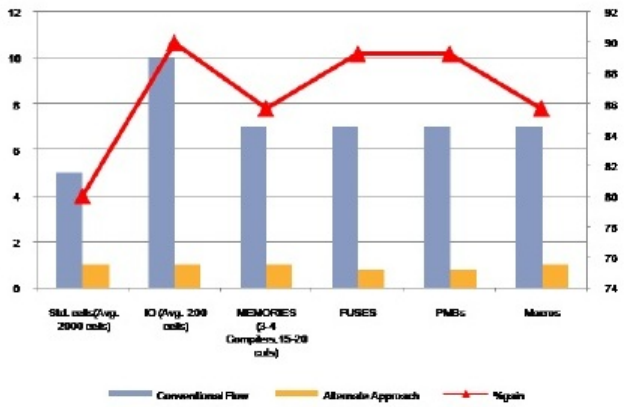

Figure 3.8: Mapping between Check Taxonomy and Plug-in Taxonomy

Chapter 4

Older Ipscreen Architecture

4.1 Existing Approach

Currently the library views are developed using software called IPSCREEN.

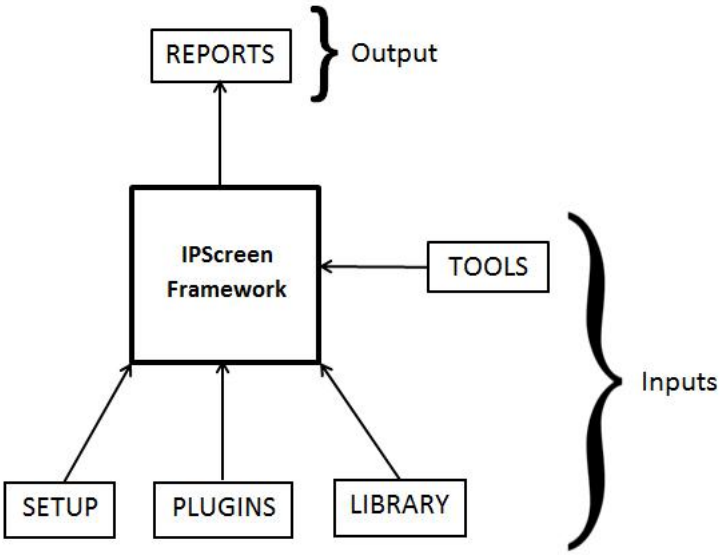

Figure 4.1: IPScreen and Plugins Block Diagram

IPScreen is a framework built using TCl and Tk on which library are loaded for validation. Multiple libraries of different technology can be loaded at the same time. Plugins are loaded after the library are loaded on IPScreen. Each tuple of library, plugin corresponds to a new tab in IPScreen window. The IPScreen can be run in GUI mode or else in Batch mode, in GUI mode the user has to click on each task that they want to run. To run in batch mode all the commands are given in a file as input. Depending

upon the availability of license for tools, the execution time of the check may vary. Also the time of execution varies on the number of cells in the library and size of each cell.[5]

4.2 Inputs to IPScreen

- Command File: All the check(s) that are to be performed are given in this file along with libraries. The contents of command file are: command to load library, command to load auxiliary library, command to run specific task.
- Tools, Plugin record: Depending upon the techno of library(65nm, 32 nm etc) the tools version may change, so correct tools are to be ensured for each validation of library. These tools are Electronic Design Automation Tools (EDA tools). All the plugins that are to be used, are specified in a seperate file along with their path.
- Setup: The setup script sets the environment in order to execute tasks on load sharing facility and other environment variables required.

4.3 Information Dumped by IPScreen on parsing an IP

After an IP is loaded and parsed by IPScreen framework,the information dumped by IPScreen is used by plugin for its specific check. The developer of plugin may use this information as per his choice. The structure of the dumped information is plugin independent.

- Status of Tools : It tells ftool name, status of toolg, where status of tool is missing/ present.
- Library Information : For all the libraries loaded following information are dumped, fName, Version,Type, path,path till index file ,product name, iptype.g
- Status of all Collection having collection name, Statusg , where Collection status is failed, warning, Done correctly.
- Status of Task: It has plugin name, task name, tool status, task status.
- List of all cells present in a specific library.

- Path of each view for each cell, condition, plugin, type of library (main or supporting library).
- Pairs of all plugin and library that are to be used.

On running, aborting, rerunning a task the Status of Task table is updated. On loading, reloading a tool Status of tools table is updated. Initially when Library is loaded Library table is created. On running and re running a collection, Collection status table is updated.All the tables are initially dumped by IpScreen, and are later modified during task run.

4.4 Final Output of Plugin Run on IpScreen

The primary level report shows ”No Error / Warnings found” if execution is correct and input library is correctly validated, a green tick is shown on IPScreen GUI along with ”Done”. If during the execution of script, if the scripts aborts due to abnormal program execution, then the task fails and primary report is not prepared, so FAILED is shown on the IpScreen GUI along with Red Cross. If there are errors present in library then the report shows ”FAILED” message, and IPscreen GUI shows ”Done” along with Red Cross. If there are no errors but warnings like ”.v file not present in library” then ”Warn” message is shown in report, along with Done on IpScreen GUI, and brown colour Tick indicating warning. The final xls reports are hierarchical and tell status of each check.

4.5 CrossCheck Plugin

This Plugin is designed to ensure view consistency. The aspects of this plugin are Similar view consistency and different view consistency

- View Consistency: Ensure that the same type of views are consistent among them self.
- Different View Consistency: Ensure that all the different type of views are consistent among each other example:(.lef vs lib, .lef vs .v). [6]

4.5.1 Tasks performed in crosscheck

- Data Extraction: During this step the layout view are dumped into .lef format and from the .lef a common tree format is built, the tree format includes inforamation

like cell name, cell area, pin name, pin direction. Also, the abstract view is dumped in a common tree format using ST internal tools, and information like cell name, cell area, Pin name, Antenna Information, Pin direction, Pin type are dumped into a textual format. Similarly, Verilog Views, Apache views are extracted and dumped into a common tree format.[5]

Figure 4.2: Load library in Ipscreen

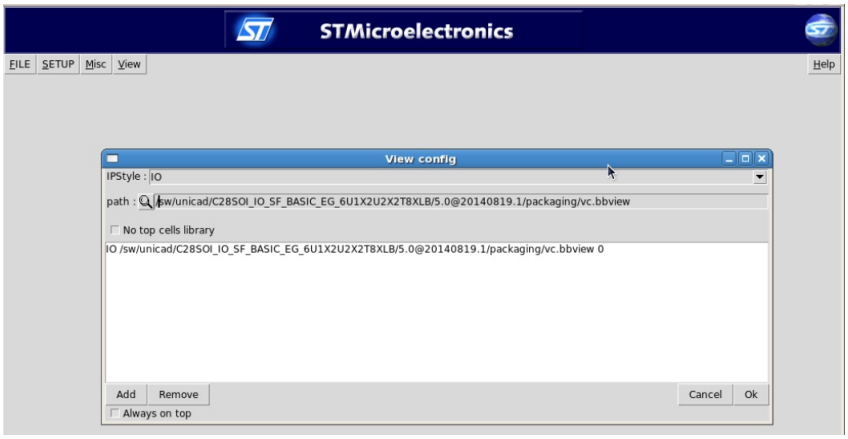

Figure 4.3: Load library Path

- Reference Preparation: During this step, the reference trees are prepared and are then compared among themselves to ensure consistency among themselves. Here, .lef, .lib and layout are compared among themselves. Out of all the reference views one view is choosen as reference further consistency check.

- Consistency Check: The consistency between all the trees dumped in Tree Formation steps checked with respect to reference views.

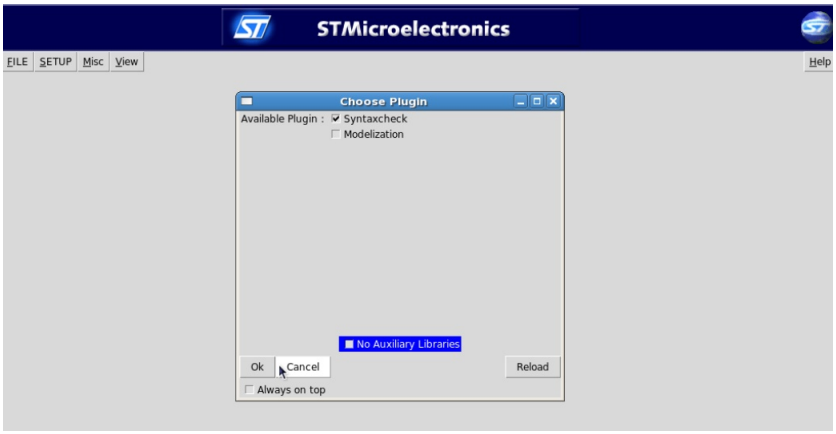

Figure 4.4: Load plugin

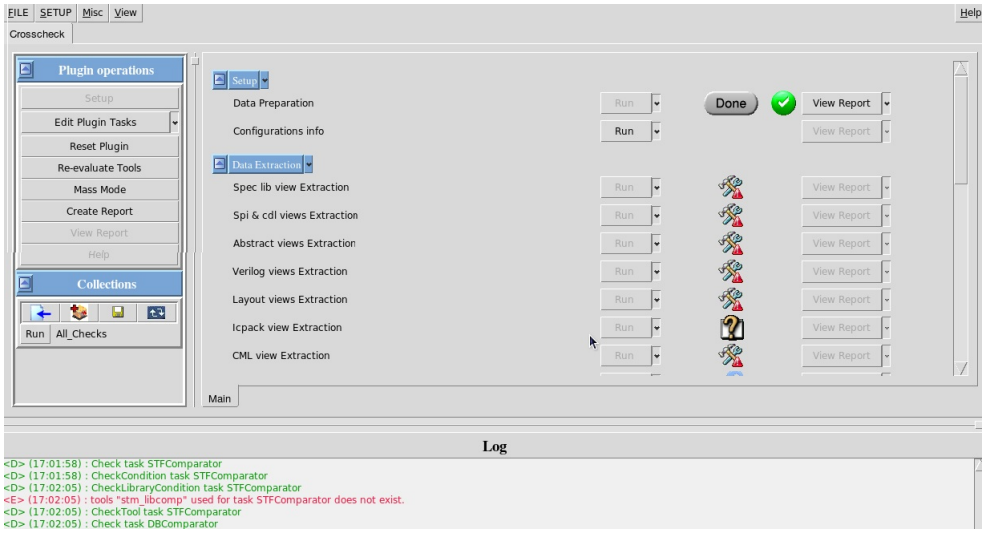

Figure 4.5: Ipscreen run of CrossCheck Plugin

- Liberty Comparator: This check check the consistency between .lib and .db respectively.Also it checks the consistency between .lib and reference choosen.

4.6 Modelization Plugin

The Following tasks are performed in Modelization Plug-in.

- After checking that the view is readable by the EDA tools, it is checked that whether the given view has necessary attributes present which is required by the EDA tools.
- If the necessary attributes are present, then it is checked that whether their value is modeled correctly or not according to the Design Rules

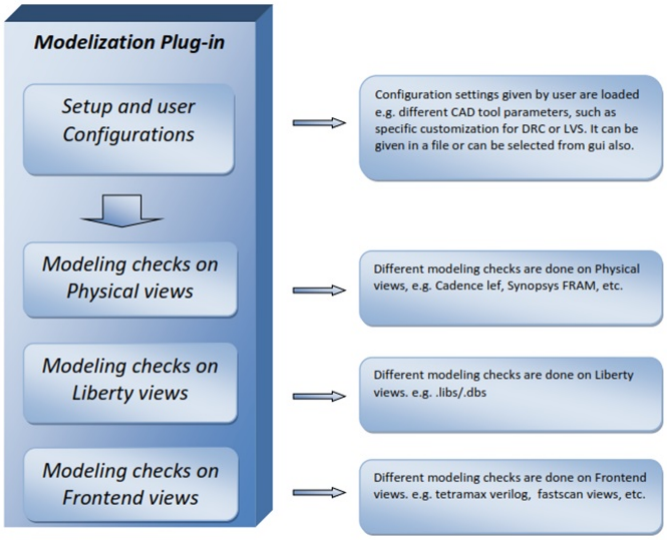

Figure 4.6: Modelization Checks Flow

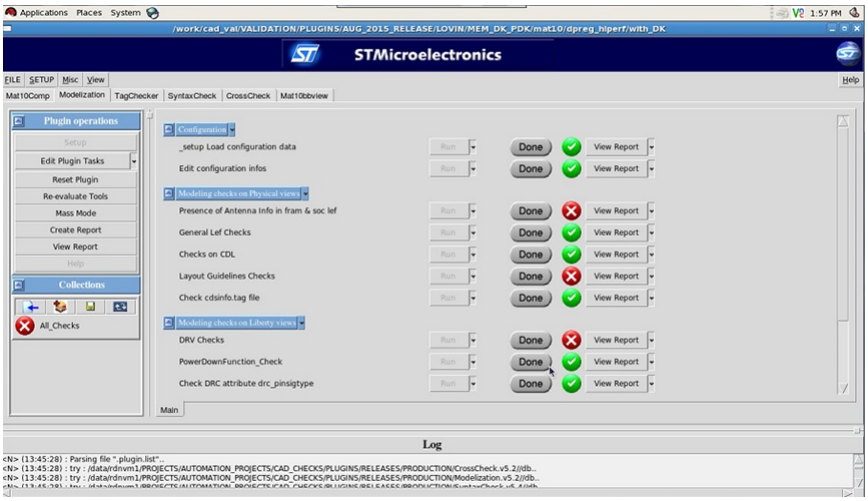

Figure 4.7: Ipscreen run of Modelization plugin

4.7 SyntaxCheck Plugin

The Following tasks are performed in Syntax Check Plug-in.

- View is checked syntactically by reading that view in its respective tool
- If the view is syntactically incorrect, the EDA tool will not be able to process the view .

4.8 New Approach Vs Older Approach

As mentioned in the implementation part about Ipscreen software was validating the whole library. Ipscreen is just the gui that was being used to validate the libraries. The

Figure 4.8: Ipscreen run of Syntax Check Plugin

plug-in which consists of developed scripts that were being loaded in the Ipscreen.

Using older Approach logs were tough to understand. In newer architecture development is using command line interface not by Gui through ipscreen.Single view validation is possible Also view level and check level checks are possible.

Chapter 5

New Architecture Implementation

5.1 Development Flow

5.1.1 Initialize

In the initialization process all the inputs are initialized with their default values. It generates the valid values for the command line input paramaters.

5.1.2 ProcessInputs

There are two ways in which user can provide inputs . One is in the command line and other is in the file . It contains parameters like library , view ,checkname , subcheckname, techno , ipstyle etc . Library name is to be given as an argument , with this name view i.e lef,stf,verilog , checkname and subcheckname is the core check script on which the check must run , techno is the techno file.

5.1.3 GenerateSetup

Based on the user inputs the setup is automatically generated and the processed files are placed under it. There are folders like Report , check , Logs and run folder .User can see the status of subcheck and reports from these directories.

5.1.4 ExecuteChecks

After generating the setup the task is to execute main check script . All the checks run in parallel. The checks dumpes their status i.e success full or unsuccessful.

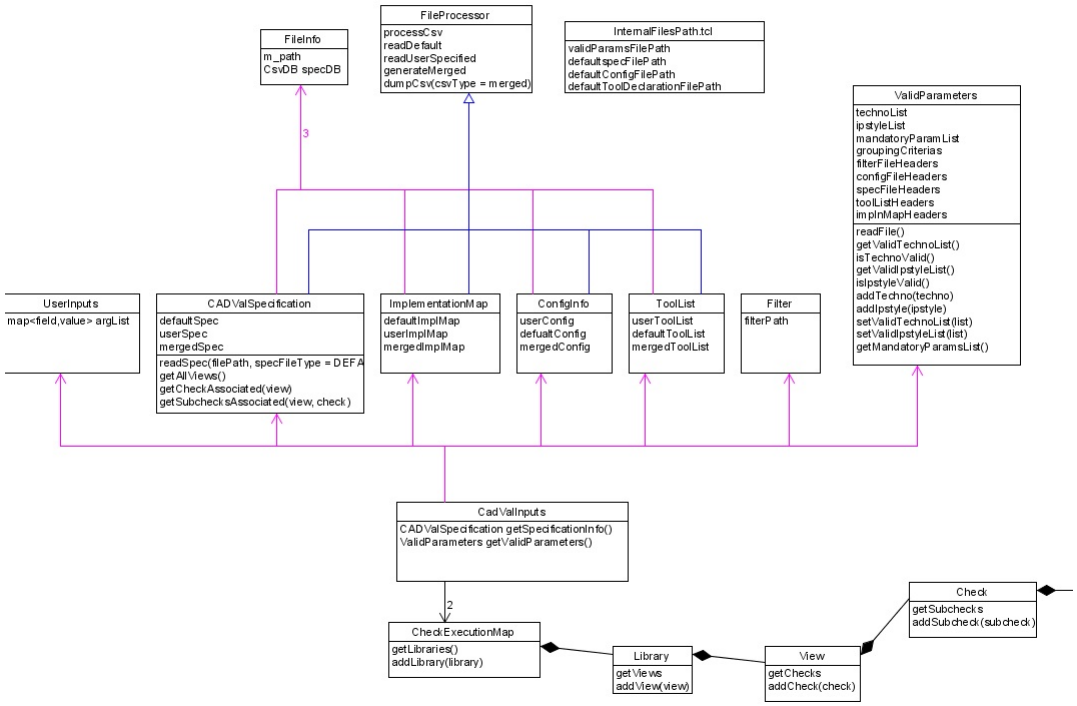

Figure 5.1: Process Inputs

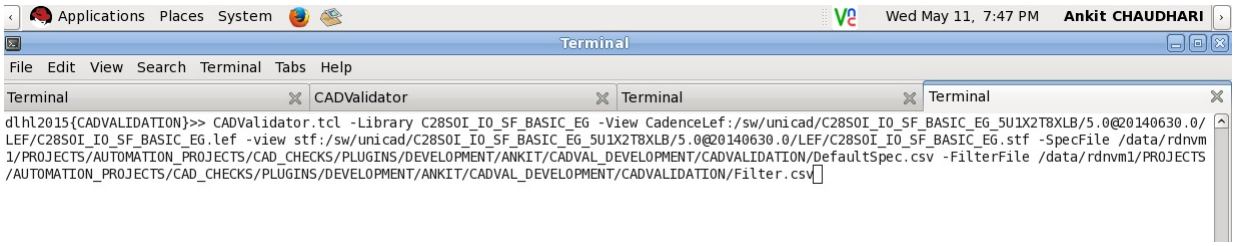

Figure 5.2: User Input Command

5.1.5 GenerateReport

In the end the report generation is carried out . It generates reports for all the subchecks.

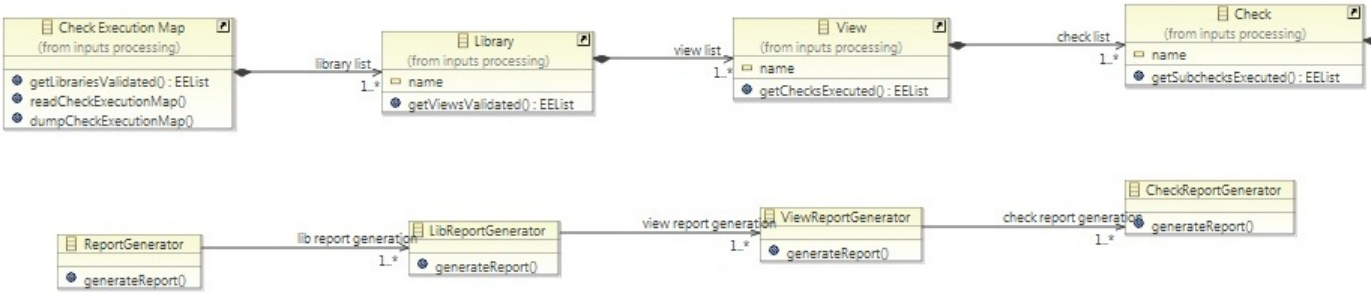

Figure 5.3: Report Generation

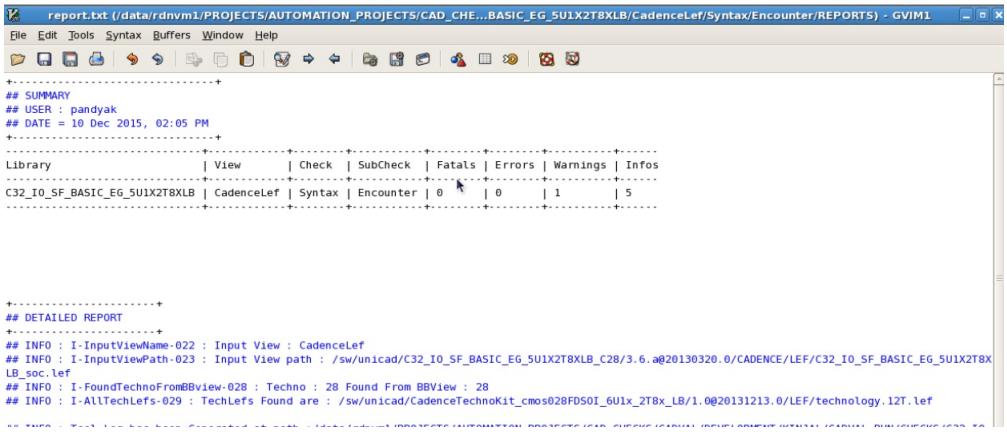

Figure 5.4: Report Of New Architecture

5.2 Technology and Tools in New Architecture

Scripting Languages like Perl,Shell Script,Bash,Korn Shell,Python are utilized for computerizing little tasks.These dialects are deciphered and not incorporated. They are progressively written, that is sort checking is done at run time and not at gather time. Scripting Dialects are utilized for quick application advancement. Scripting Languages are great at string handling. Scripting dialects permit bunch occupations that would be entered physically entered on the order line to be executed naturally in a steady progression utilizing scripts. Additionally, composing the shell script is more quick than comparable code in other programming dialect. The primary favorable position of scripting dialects is that the orders and punctuation are precisely same when utilized at terminal. Additionally, intuitive debugging,easy document choice, brisk begin are different favorable circumstances. The plugins are developed in C Shell and TCL.

5.2.1 OOTCL

Object oriented tcl is used to extend the functionalities of tcl . Its is available as an built in package in tcl 8.6. It provides all the functionalities as oop. The new architecture is developed using ootcl and cshell.[7]

5.2.2 C Shell

C Shell provides Control statements like if, while , foreach , switch etc. It also provides SubShell, in which a child copy of current shell is created inheriting the current state,

without affecting the parent. Shell script provide -x option for debugging by displaying commands as they are executed, -v option to display all lines as they are read.[8]

5.2.3 Tool Command Language(TCL)

TCL is commonly used for rapid prototyping,scripted applications, GUIs and testing. The main features of TCL are:

- The TCL uses tclsh command line interpreter. Other ways of starting TCL are using Wish or Windowing Shell.
- All data types can be manipulated as strings.
- TCL interpreter performs run time compilation of script into byte code. Running compiled code allows TCL script to run faster than Perl.
- TCL supports most of modern programming constructs like subroutines, standard programming flow constructs, rich set of variable type like lists, associative array,float, integer, string etc.
- TCL provides powerful string manipulation commands for searching and replacing, extracting portion s of string, converting strings to list.
- Extensibility: TCL extension add a few new commands to extend interpreter into new application domain
- Provides GUI interface Tk.
- All commands of TCL generate error message on incorrect usage.

5.3 CrossCheck Plugin

Crosscheck means cross verifying the views and report matching or miss matching.

As shown in the older approach of crosscheck plugin , the checks developed are different for all the views . The reports are also not aligned with each other .

The goal is to develop the check scripts which could be generic for all the views.Previously whole plugin was there for cross-check. Now the idea is to develop generic scripts which

should work for all the views.i.e if one wants to verify lef(library exchange format) view and stf(synopses technology file) view then it should report matching and mismatching off common attributes of those views and also each attribute value is to be compared.

5.3.1 Implementation

- Csh - c shell
- Tcl - tool command language
- Namespaces in tcl
- Wrapper classes are created for this purpose because fetching of attributes of different views are different
- Dynamic Arrays and configuration files and Parsers for lef , fram ,stf.

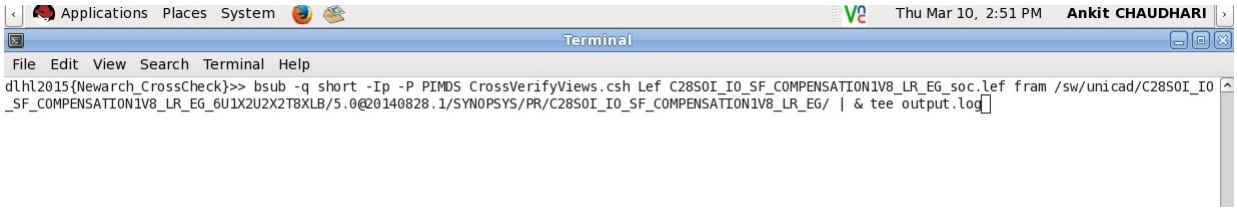

Figure 5.5: CrossCheck Command

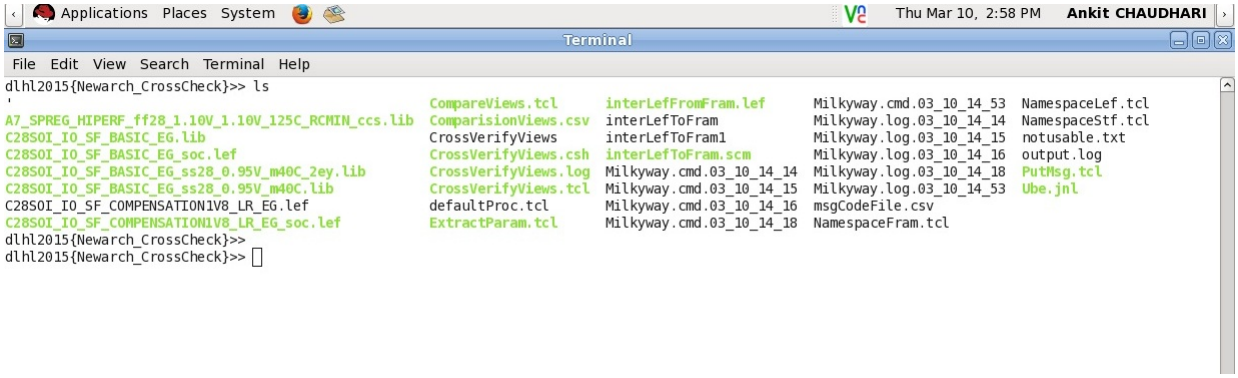

Figure 5.6: CrossCheck Folders

5.3.2 Advantages over Previous crosscheck

- The current script is generic for all the views. previously manual developed parsers were there and code was different for all the views. In current approach through name-spaces and dynamic data structures and the wrapper classes code is generic for all the views .

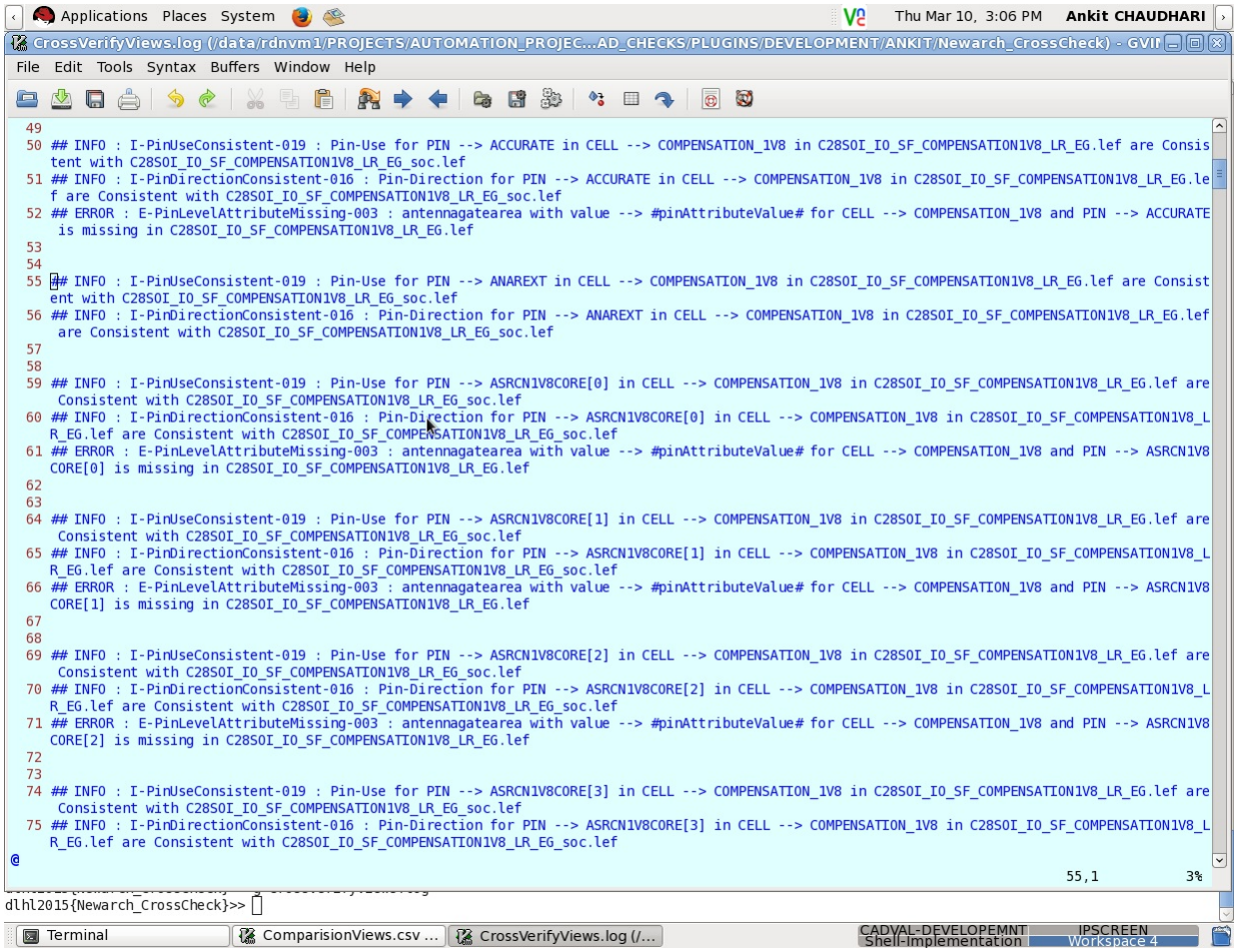

Figure 5.7: CrossCheck Output

- Debugging is much faster
- Log is consistent for all the views
- Code is divided in modules so readability is much more
- Run time is much faster than previous approach

Chapter 6

Conclusion and Future Scope

6.1 Conclusion and Future Scope

Providing a validation solution which reports mismatches or modelling errors for Libraries and IP that can seriously delay an IC design project .Some checks required the user to manually check all the relationship between various different views.Giving input to EDA tool through command line or GUI, the results had to be collected, and analyzed for each cell. This would take weeks and the results of validation may still be prone to human errors. like using older approach the whole library was being validated. One cannot validate the single view in the older approach. but using the newer approach one can individually validate the different views of the library by giving some specific files like lef file or cdl file.

- Manual testing requires much efforts and time.
- Automation reduces the efforts and time.
- So reducing effort leads to minimal cost.

Old Approach	New Approach
Inputs are only library	Input can be anything
Development is Complex	development is simple
Single view validation not possible	Single view validation possible
validation result lOG is complex	lOG is common and simple to understand

Currently new Architecture is going on for validating the checks as mentioned above. Command line validating of checks instead of using gui.

References

[1] STMicroelectronics, "Stmicroelectronics internal document on library and views,"

[2] W. Agatstein, K. McFaul, and P. Themins, "Validating an asic standard cell library," pp. P12–6, 1990.

[3] STMicroelectronics, "Stmicroelectronics internal document on plugins,"

[4] M. J. S. Smith, "Application specific integrated circuits,"

[5] STMicroelectronics, "Stmicroelectronics internal document on ipscreen,"

[6] SYNOPSYS, "Astro user manual,"

[7] https://en.wikipedia.org/wiki/OTcl.

[8] https://en.wikipedia.org/wiki/C_shell.