Development of CAD Checks for Validating Library Views and its Reporting

Submitted By

Kinjal Pandya

14MCEI13

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2016

Development of CAD Checks for Validating Library Views and its Reporting

Major Project

Submitted in fulfillment of the requirements

for the degree of

Master of Technology in Information and Network Security

Submitted By

Kinjal Pandya

14MCEI13

Guided By

Prof. Jigna Patel

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2016

Certificate

This is to certify that the Second phase of project entitled "**Development of CAD Checks for Validating Library Views and its Reporting**" submitted by **Kinjal Pandya (14MCEI13)**, towards the fulfillment of the requirements for the degree of Master of Technology in Information And Network Security(CSE) of Nirma University, Ahmedabad is the record of work carried out by her 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 major project part-II, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. Jigna Patel
Guide & Associate Professor,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad

Dr. Sharda Valiveti
Associate Professor,
Coordinator M.Tech - INS,
Institute of Technology,
Nirma University, Ahmedabad

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

Certificate

This is to certify that the major project entitled **"Development of CAD Checks for Validating Library Views and its Reporting"** submitted by **Kinjal Pandya (Roll No: 14MCEI13)**, towards the fulfillment of the requirements for the award of degree of Master of Technology in Computer Science & Engineering (CSE) of Nirma University, Ahmedabad, is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Date:

Project Manager

Mrs. Jyoti Kumar

TRnD Department,

STMicroelectronics, India

Statement of Originality

I, **Kinjal Pandya**, Roll. No **14MCEI13**, give undertaking that the Major Project entitled "**Development of CAD Checks for Validating Library Views and its Reporting**" submitted by me, towards the fulllment of the requirements for the degree of Master of Technology in **Information and Network Security (CSE)** 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

Guide Name

Prof. Jigna Patel

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Mrs. Jyoti Kumar**, Project Manager, ST Microelectronics, Greater Noida and **Mr. Rishabh Bansal**, Technical Leader, ST Microelectronics, Greater Noida for their valuable guidance and mentorship that helped me to overcome every challenge I faced as I moved on in this project..

My deepest thanks to **Prof. Jigna Patel**, Assistant Professor, Department of Computer Science and Engineering, Nirma Institute of Technology, for giving me an opportunity and guidance throughout the project. It was only due to his valuable opinion and ever friendly nature that I was able to do part of my research work in a respectable manner.

I am highly grateful to **Dr. Sharda Valiveti**, P G Coordinator ,Information and Network Security, Nirma Institute of Technology for her kind support.

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

I would also thank my Institution, all the faculty members and my colleagues.

- Kinjal Pandya

14MCEI13

Abstract

Electronic devices that we are using in our day to day life are composed of several chips, which is fixed inside these devices for their proper functioning. This chip is a combination of different programmable logic gates, memories, flip-flops, registers and latches. Library is a collection of cells and Logic gates that are fabricated onto the chip.

This project is related to development of Plugins for 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 checks different views of library and each and every cell and macros of a library.

So in my project, I had implemented scripts of Plugins(Syntax Check, Cross Check, Modelization, Mat10-Bbview) that can validate the different views of the libraries. In S T Microelectronics we are using one framework, IPScreen software that provides GUI for loading different plugins, but in this project we are developing new architecture that will overcome all the limitations of IPScreen software. Plugin contains all the automated scripts required for validating library views. SO in this new architecture my task was to implement scripts for SyntaxCheck and Modelization Plugins and to implement "Report Generation" Module.

Abbreviations

SoC	System on Chip
IP	Intellectual Property
CAD	Computer Aided Design
RAM	Random Access Memory
ROM	Read Only Memory
USB	Universal Serial Bus
PLL	Phase Locked Loop
EDA	Electronic Design Tool
GUI	Graphical user Interface
LSF	Load Sharing Facility
SLIB	Symbolic Library
CDL	Circuit Description Language
GDS	Graphical Design System
LEF	Library Extension Format
CADVal	Computer Aided Design Validation

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Electronics devices that we are using in our day to day life are composed of several chips, which is fixed inside these devices for their proper functioning. This chip is a combination of different programmable logic gates, memories, flip-flops, registers and latches. Library is a collection of cells and Logic gates that are fabricated onto the chip.

This project is related to development of Plugins for 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 checks different views of library and each and every cell and macros of a library.

So in my project, currently I am developing Plugins(Syntax Check, Cross Check, Modelization, Mat10-Bbview) that can validate the different views of the libraries. In S T Microelectronics we are using one framework, IPScreen software that provides GUI for loading different plugins, but in this project we are developing new architecture that will overcome all the limitations of IPScreen software. Plugin contains all the automated scripts required for validating library views.

1.2 Problem Definition

Library is a collection of cells, containing different views that are useful in designing a chip,[1] So before fabricating a library onto the chip we need to validate it. If all the views of the library and its cells are correct then only we can move further towards soldering of library onto the chips (then only we can fabricate it onto the chip).

So in my project I am developing scripts of CAD-checks that can validate the library views syntactically and in many other ways using Plugins. So my task is to develop scripts for SyntaxCheck and Modelization Plugins. Also to develop scripts for generating Reports in different different formats i.e. TXT,CSV,HTML and XLS format at the end of execution of each check.
User can run the IPScreen in two different ways :

- GUI Mode
- Batch Mode

For that, First of all the library that is to be tested (has to be validated) will be loaded onto the IPscreen framework. Then, plugin is loaded and after that tool required for validating particular view will be loaded.

1.3 Introduction of Library IP

LIBRARY can be defined as a set of all design data that are available for an Intellectual Property.[2] The design data contains cell functionality, transistor level design of an IP, and the timing information of cells.[2]

We can also say that library is a collection of cells, containing different views that are useful in designing a chip. Cell is a component performing a basic function and a view is just the particular representation of that cell. Due to increased complexity of circuit and shorter time to market, SoC designers cannot really concentrate on design of basic building blocks. This database can be reused in designing various System on Chip(SoC).[3].

1.4 Types of Library

Following are the different types of libraries available according to the customers requirement, design/structure and functionality of the devices.

1.4.1 Core Library

Core Library mainly consists of a group of cells which are Standard Cells. They implement basic logic functions like flip flops, Gates, Inverter, counters, adder, subtractor, etc..

1.4.2 Input/Output Library

I/O Library consists of a group of cells called I/O buffers. I/O buffers are designed to interface off chip signals to inside chip environment and vice-versa. I/Os are placed on the periphery of the chip.[1] I/O libraries are mainly used inside the functioning of input output devices to store Input and Output bit that are passed by user.

1.4.3 Memory Library

Memory Library contains memory of different architecture. Examples of Memories are SRAM, DRAM, ROM. As there can be number of memory sizes, we implement basic building block of memory and configure the generation of different memory sizes.[3]

1.4.4 Analog and Mixed Signal library

Mixed Signal Libraries are implemented using the CORE library in a custom manner. Digital to Analog Converter, USB and PLL are the examples of Analog and Mixed Signal libraries.

Chapter 2

Literature Survey

2.1 Library Views

As discussed previously, Library is a collection of cells and each cell has views which are the representation of the cell.[2] All the cells have : Layout View, Abstract View, Schematic View, Symbolic View and Timing view. A cell is delivered as a set of all view and each view is used by different different tool in a given electronic design flow.[2] Main library views are explained below :

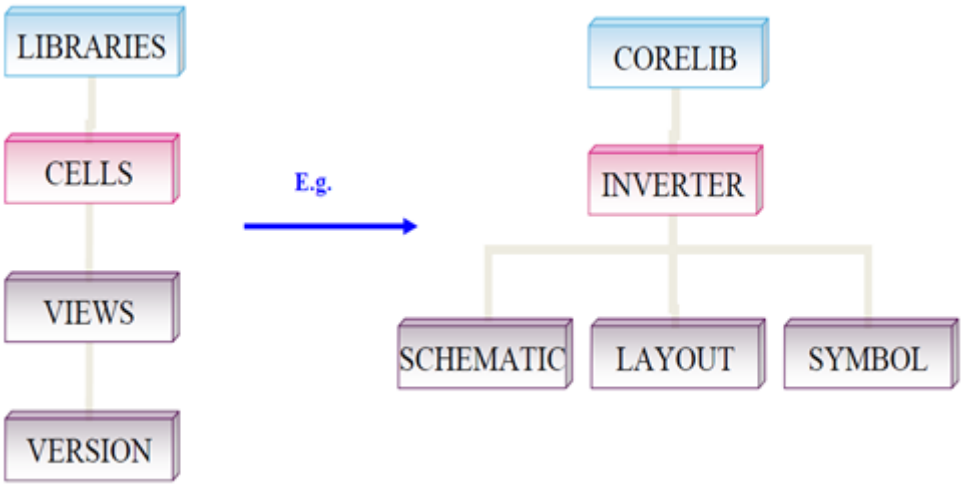

Figure 2.1: Library, Cell and View Block Diagram

2.2 Symbolic View

Symbolic View is the sketch (pictorial) representation of any cell, it gives us the idea about how the symbol design will look like when symbol is used in the Schematic View.

It includes Pins, Macros, Symbol, Labels, Selection box. Pins represents inputs and outputs of the cells. The shape of a particular symbol in the diagram represents cell's function. [1] Labels in the symbol are used to add to the documentation of the design, selection box selects the complete area for the cells. [3] Symbolic Views allows the user to abstract a complex Schematic View and replace it by a Symbolic view that can be used as in further designs.[2] Symbolic view of an Inverter is shown below:

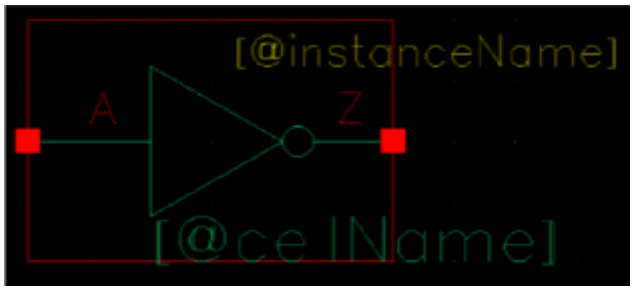

Figure 2.2: Symbol View representation of a NOT gate

2.2.1 SLIB

SLIB stands for **Symbolic Library** SLIB file is the textual representation of symbolic view as shown in figure-2.3 :

Symbol --> .SLIB

Example:

```
symbol ("IVHD") {

set_minimum_boundary (0 * SCALE, -40 *
SCALE, 120 * SCALE, 40 * SCALE);

circle (88 * SCALE, 0 * SCALE, 5 *
SCALE);

line (83 * SCALE, 0 * SCALE, 40 * SCALE,
-25 * SCALE);

.....

pin ("A", 0 * SCALE, 0 * SCALE,
ANY_ROTATION);

pin ("Z", 120 * SCALE, 0 * SCALE,
ANY_ROTATION);
```

Figure 2.3: Example of .SLIB

2.3 Schematic View

Schematic view is the simplified representation of an electronic circuit. It shows the different components of the circuit as simplified standard symbols, and the power and signal connections between the devices.[2] It shows the representation of the cell at transistor level. It represents component instances, wires and pins.

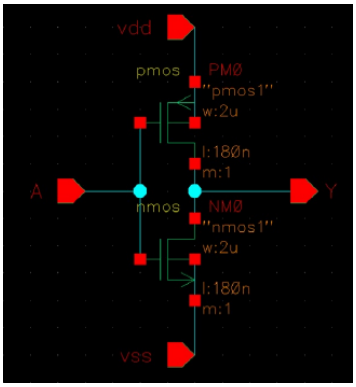

Figure 2.4: Schematic View of an Inverter

2.3.1 Circuit Description language (CDL)

CDL file is the text representation of the schematic view as shown in the figure-2.5. CDL file provides following information :[2]

- Connectivity at the transistor level
- Device parameters like device name, length, width, Area.
- Power pin and ground pin related information at the transistor level

2.4 Layout View

It is the clear physical representation of cell‘s electronic circuits that goes on the silicon for fabrication. It is the representation of IC that in terms of geometric shapes which correspond to patterns of MOS(Metal oxide Semiconductor).[2] Layout view must pass a sequence of checks during Verification process, The most important checks are Design Rule Check (DRC) and Layout vs Schematic (LVS).[**?**] The parameters for such check are given by chip manufactures.Another name of Layout View mask design, IC mask layout.

Schematic --> .CDL

```
.SUBCKT CELL1 A B C vdd vdds gnd gnds
MM8 A B gnd gnds nsvtlp w=7.4 l=0.07 nfing=1 m=1 accurateFlow=0   ngcon=1
XI20 A C / CELL2
.ENDS

.SUBCKT CELL2 P Q
DD1 P Q dpsvt25 area=0.12005 perim=1.232e-06
.ENDS
```

Figure 2.5: Example of .CDL

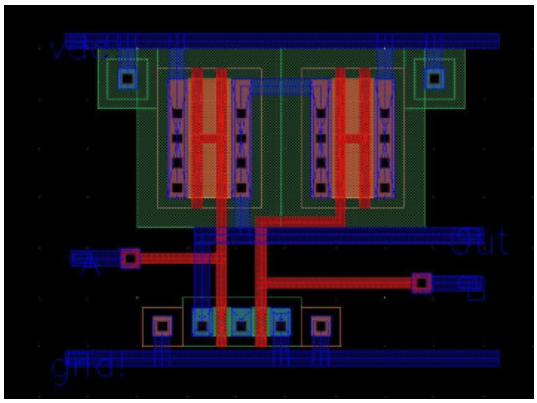

Figure 2.6: Layout View

2.4.1 Graphical Design System (GDS)

GDS file is the text representation of the Layout View.GDS file contains the same data as the layout view & is a binary format for representation of planar geometric shapes, text labels and layers, such as text, path/wire, boundary/polygon, and planar geometric shapes in hierarchical form. [3] As GDS is using its internally defined formats for its data types, GDS is platform independent.

2.5 Abstract View

From Abstract view, one can get the information about the signal & power pin layers running in the layout view. Abstract view also gives the information of area where routing is not allowed between the layers. Placement and Routing using Semi-Custom Tools does not require the full layout information but only the location of various pins to be connected and areas where routing is not allowed (obstructions)

Figure 2.7: Abstract View

2.5.1 Library Extension Format (LEF)

LEF file is nothing but an ASCII representation of the Abstract view. LEF file contains following information :

- Name of Standard Cell
- Cell Size
- Number Of Input/Output Pins in given Cell
- Definition of Pins
- Direction of Pins
- Location of pins
- OBS layer definition

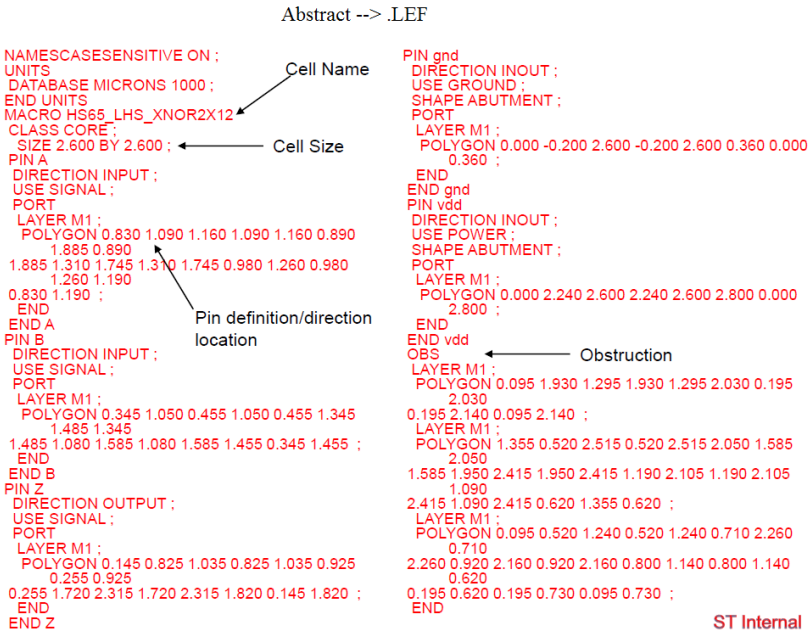

Figure 2.8: Example of .LEF

Chapter 3

Technology & Tools

Scripting Languages like TCL, Perl, Shell Scripts and Python are used for automating small tasks.These languages are interpreted and not compiled.They are dynamically typed, that is type checking is done at run time and not at compile time. Scripting Languages are used for rapid application development. Scripting languages allow batch jobs that would be entered manually entered on the command line to be executed automatically one after another using scripts. Also, writing the shell script is more fast than equivalent code in other programming language. The main advantage of scripting languages is that the commands and syntax are exactly same when used at terminal. Apart from these all,interactive debugging,easy file selection, quick start are other advantages of Scripting Languages. The Plugins are developed in C Shell and TCL.

3.1 Shell Scripts

[4]

C Shell is a command interpreter with a syntax similar to the C programming language. It can be used as both, as an interactive login shell and a shell script command processor. It provides I/O redirection, Joining of multiple commands using '&&' , ';'. Piping provides output of one command to be given as input to another command, the advantage is that both commands run in parallel. C Shell provides

Command Substitution allows output of one command to be taken as argument to another. Background Execution command & means to run command in background and prompt immediately for a new command line(command followed by ampersand). 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 aecting 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

3.2 TCL (Tool Command Language)

[5] TCL is commonly used for rapid prototyping,Automation related scripting applications, GUI development and testing.

- 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 is somehow similar to C language.
- 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.

3.3 OTcl (Object Oriented Tcl)

OTcl is nothing but an extension to Tcl with Object Oriented Programming[6]. It can be dynamically extensible. In OTcl the reserved word Class is used to represent class and method of class are declared using word instproc. The variable self is pointer to the class it is used in and is equivalent to variable this of C++/Java. The keyword -instproc is used for defining hierarchy.For example Class Son -instproc Father means that class Son inherits from class Father.

Chapter 4

IPScreen and Plugins

IPScreen[7] provides GUI which is a framework built on TCL. Libraries are loaded on the IPScreen for validation. Multiple libraries of different technology can be loaded at the same time. After the Library is Successfully Loaded, Plugins are loaded onto the IPScreen. Each tuple of library, plugin corresponds to a new tab in IPScreen window. To run the IPScreen in GUI mode user has to click on each Checks that he wants to run/execute. To run IPScreen in batch mode, all the commands are written in a command file and then command file will be passed as an argument on the terminal. Based on the availability of license for tools, the execution time of the check may vary.[7]. Time of execution varies based on the total number of cells in the library and cell size.[2]

4.1 Input to IPScreen

- **Command File :** All the tasks/checks that we want to execute are given in command file along with Library name. So finally the command file contains:
  - Command used to load Library
  - Command used to load auxiliary Library
  - Command to run specific Check/Task
- **Tool & Plugin Information:** On the basis of the technology of library(65nm, 32 nm etc) version of tool may changes, so correct tools are to be ensured for each validation of library. These tools are Electronic Design Automation Tools (EDA tools).[3] Entry of all the EDA tools that are to be used will be make in a separate file named as .plugin.list along with its path and version number.

- **Setup :** The setup script sets the environment in order to execute task on load sharing facility and other environment variables required.

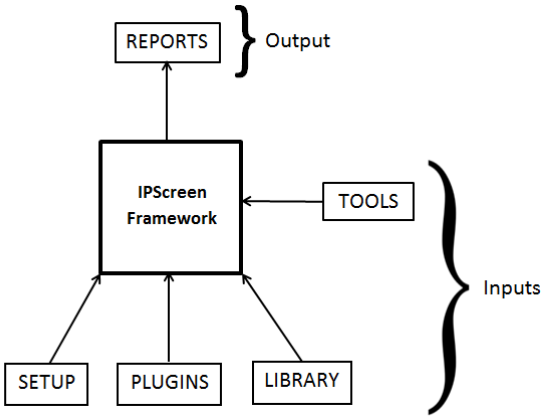

Figure 4.1: Block Diagram Of IPScreen

Figure 4.1 shows the input given by the IPScreen Framework and as shown in figure final output will be reported in a separate file. So the inputs required are : Command File, Tool Information and Plugin name. Report will be generated in the form of text, csv, xls and html.

4.2 Overview of Plugins

Plugins are the software/tools that are used to validate the different library views before soldering the electronic circuits.[8]
Various Plugins that we are developing in S T Microelectronics, to validate the library views are as follows :

- **Syntax Check :** It Checks the different views of the libraries like LEF, GDS, CDL, FRAM etc. syntactically.

- **Cross Check :** It will check the consistency of cell and pin attributes between different views. For example In a LEF view some Cell ABC is placed at some location and its origin is 2.00, then it will crosscheck that in other view say CDL that cell ABC is having same location and origin or not. So we can say it will cross verifies different pin attributes, macros and cells of libraries across different views.

- **Modelization :** Modelization mainly focuses on the placement and routing related information. It will check the layers of each macros and figure out the errors if

routing is not allowed somewhere between the layers and cell is placed at that location.

- **Mat10 Bbview :** It will compare attributes (like Cell names, Operating conditions, View paths and Input/Output parameters) of bbview file, of the loaded library by taking bbview file of auxiliary library as reference. So we can say it will compare two bbview files (bbview file is the main file of a library) and report the differences between them.

Chapter 5

Implementation Of Plugins using IPScreen Framework

5.1 Manual Approach of Validating Library Views

In the manual approach for IP Validation, the user had to firstly launch IPScreen window, manually locate the path of the IP to be loaded ,specify its path and wait till the library gets completely parsed by IPScreen. Secondly, the user then had to check if any supporting libraries are needed(auxiliary library), if yes then these all supporting libraries are to be manually loaded so that all the IP data is parsed and dumped by IPScreen. The user had to find all the latest version of the plugins needed and specify their paths in the IPScreen window.Then all the tools are evaluated based on the techology of input IP. If some missing tools are found, the cause of missing tool needs to be analysed and fixed.

5.2 Library Validation Process using IPScreen and Plugins

Following steps gives the idea about how the library has been validated using IPScreen GUI.

- As shown in Figure 5.1, user launches the IPScreen GUI by executing command on the terminal

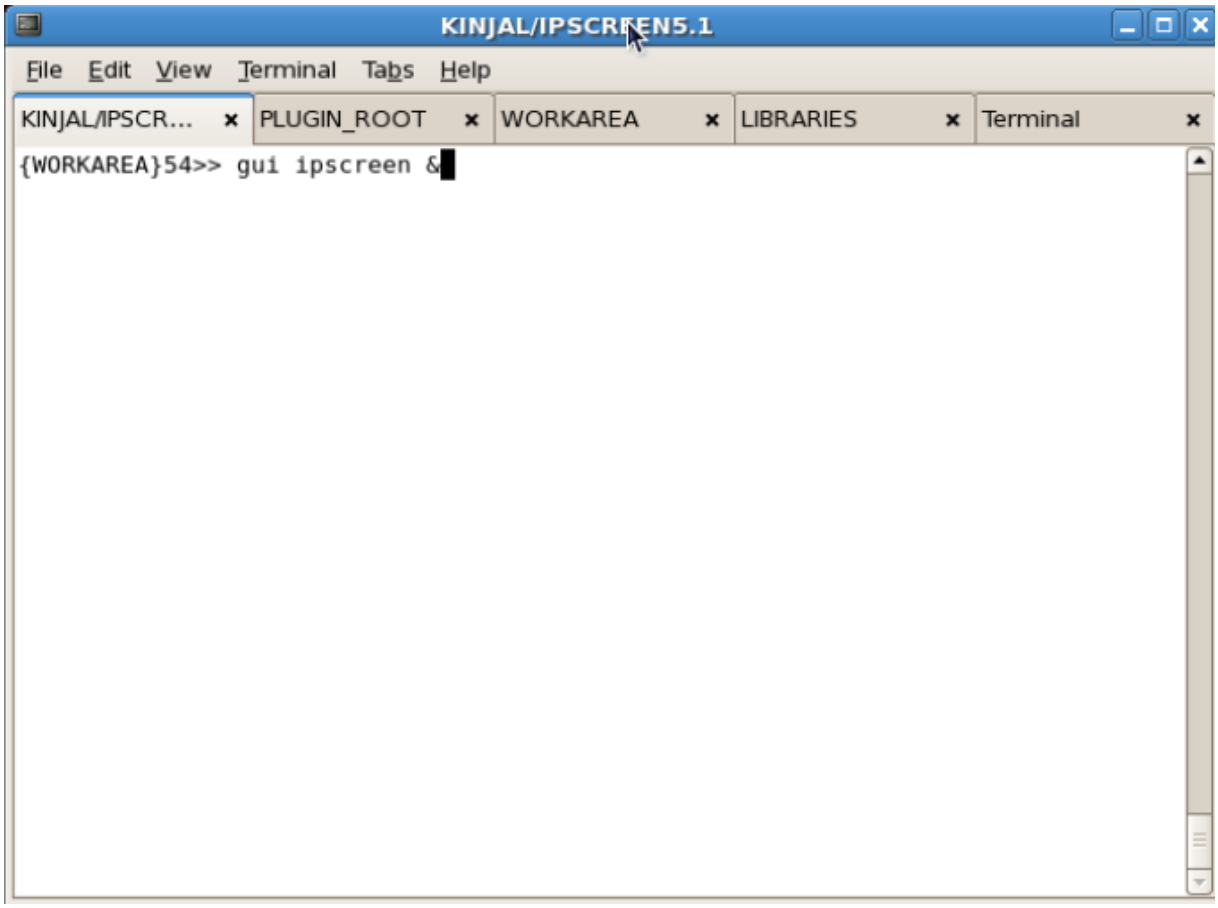

Figure 5.1: Launching IPScreen

- After executing command IPScreen will be launched and the GUI appears as shown in Figure 5.2. It will ask the user to Load the IP (i.e. Library) that user wants to validate.

- Then user has to select the IP style and path of the library which is to be validated. User has to give the full path of library upto vc.bbview file, as shown in Figure 5.3 .

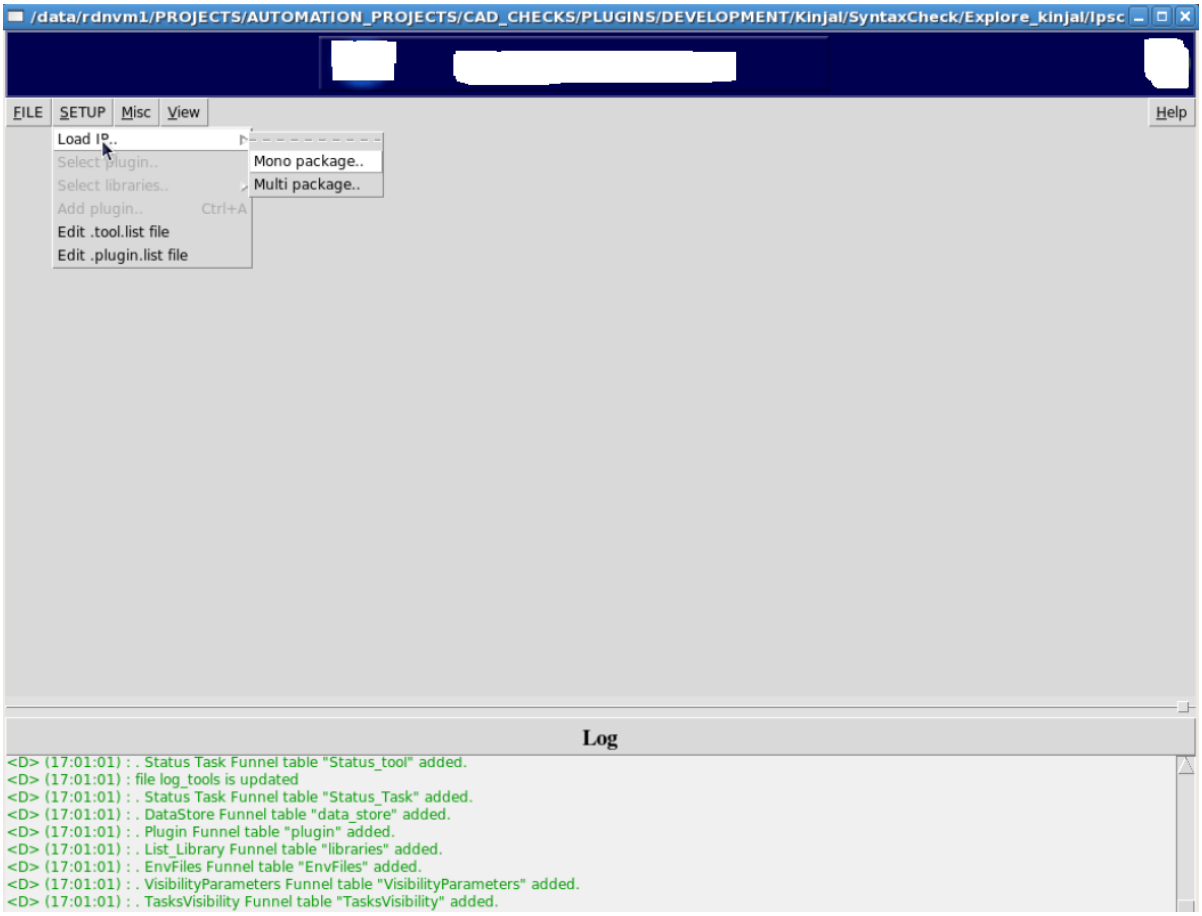

Figure 5.2: IPScreen GUI Launched

- After Loading the library successfully, the next task is to load the Plugin which contains scripts that can validate the library views and its attributes as shown in Figure 5.4.

- After Successfully loading the Library and Plugin, now user can click onto the RUN button to execute the view specific tasks. By click onto it, IPScreen and Plugin will start checking the library views as per the command given by user and verifies the view accordingly. Following figure shows the checks in running mode.
  Status of the check will appear after the particular task/check get executed. Status with:

  - **DONE with GREEN Tick :** indicates check has been executed successfully without any errors and warnings. This case is shown in figure 5.5.
  - **DONE with RED Cross Sign:** indicates check has been executed, but there is something wrong with the library view. In this case errors will be displayed in the report. This case is shown in figure 5.7.

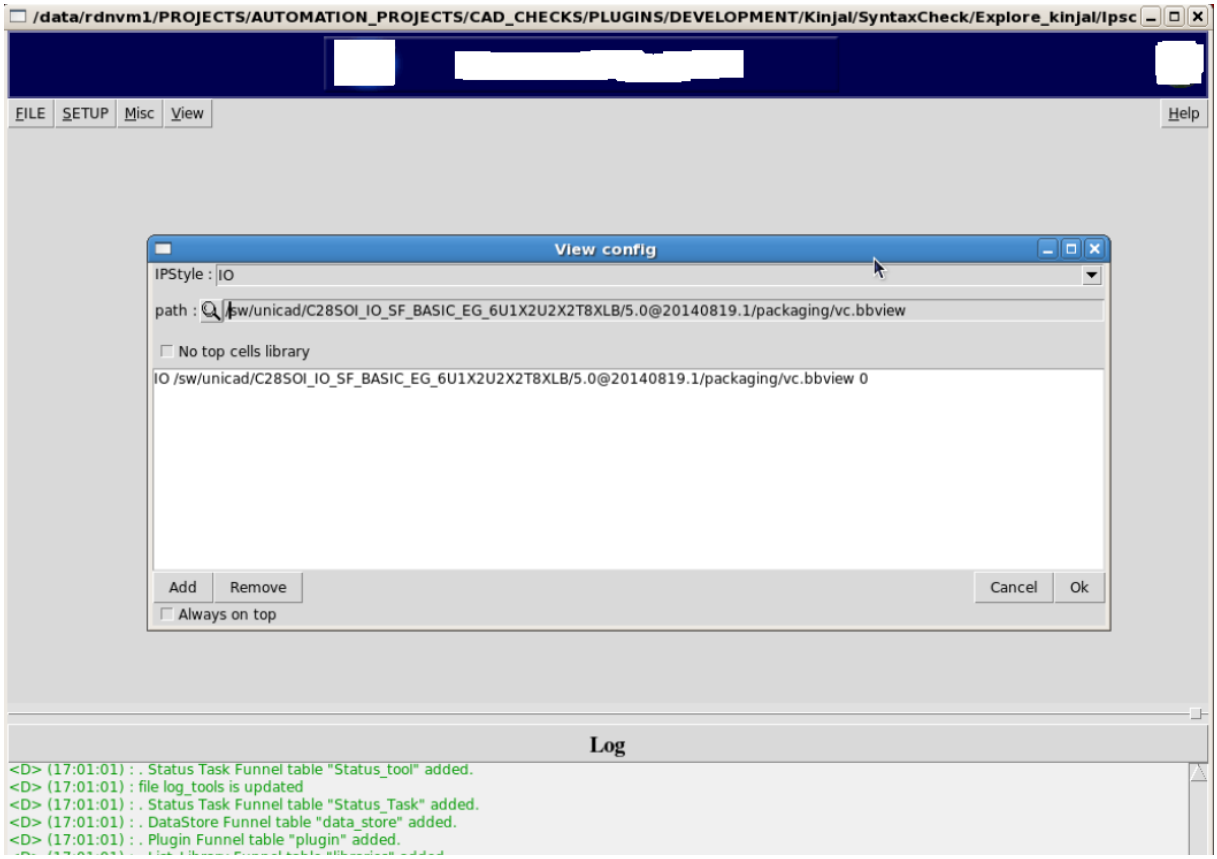

Figure 5.3: Load Library to be Validated

– **DONE with ORANGE Tick :** indicates check has been executed successfully, but there are some errors generated in the report. This case is shown in Figure 5.6.

- Finally the report will be generated in HTML form as shown in figure 5.7 and in TXT format as shown in figure 5.8.

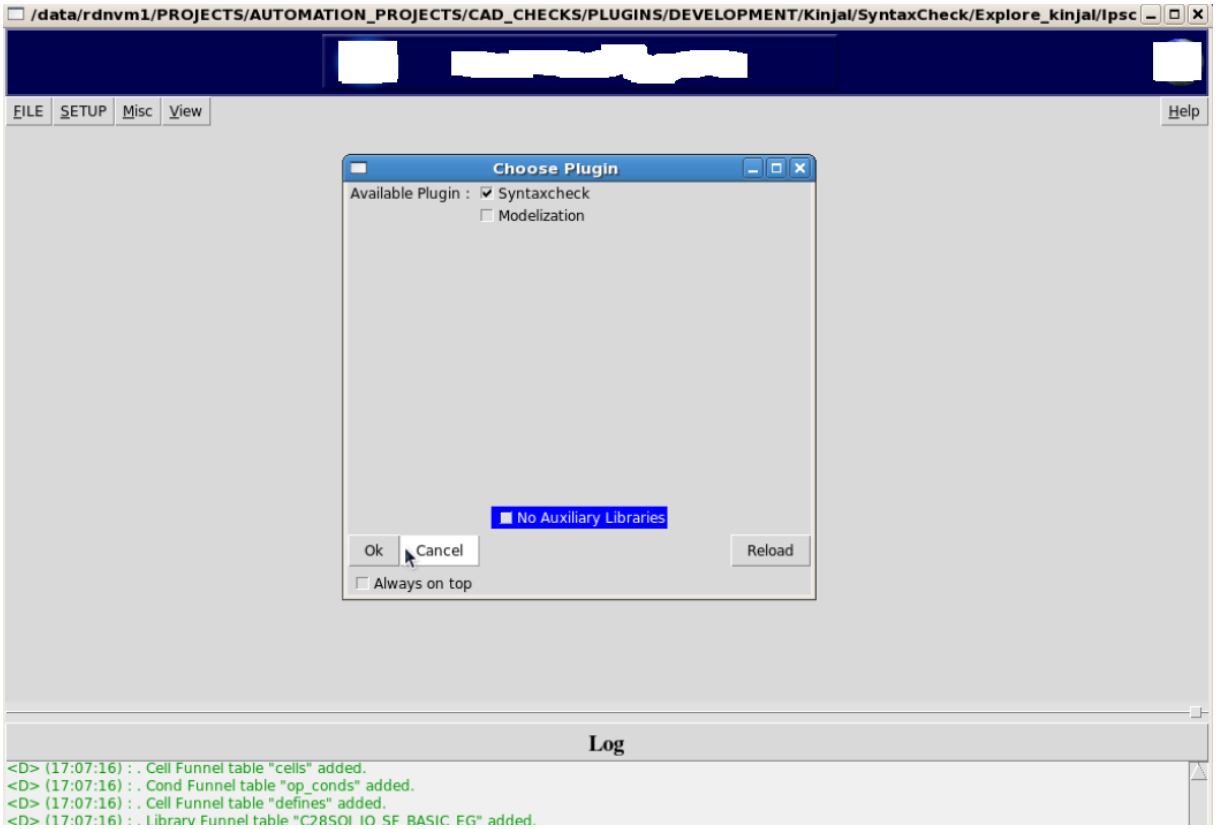

Figure 5.4: Load PlugIn

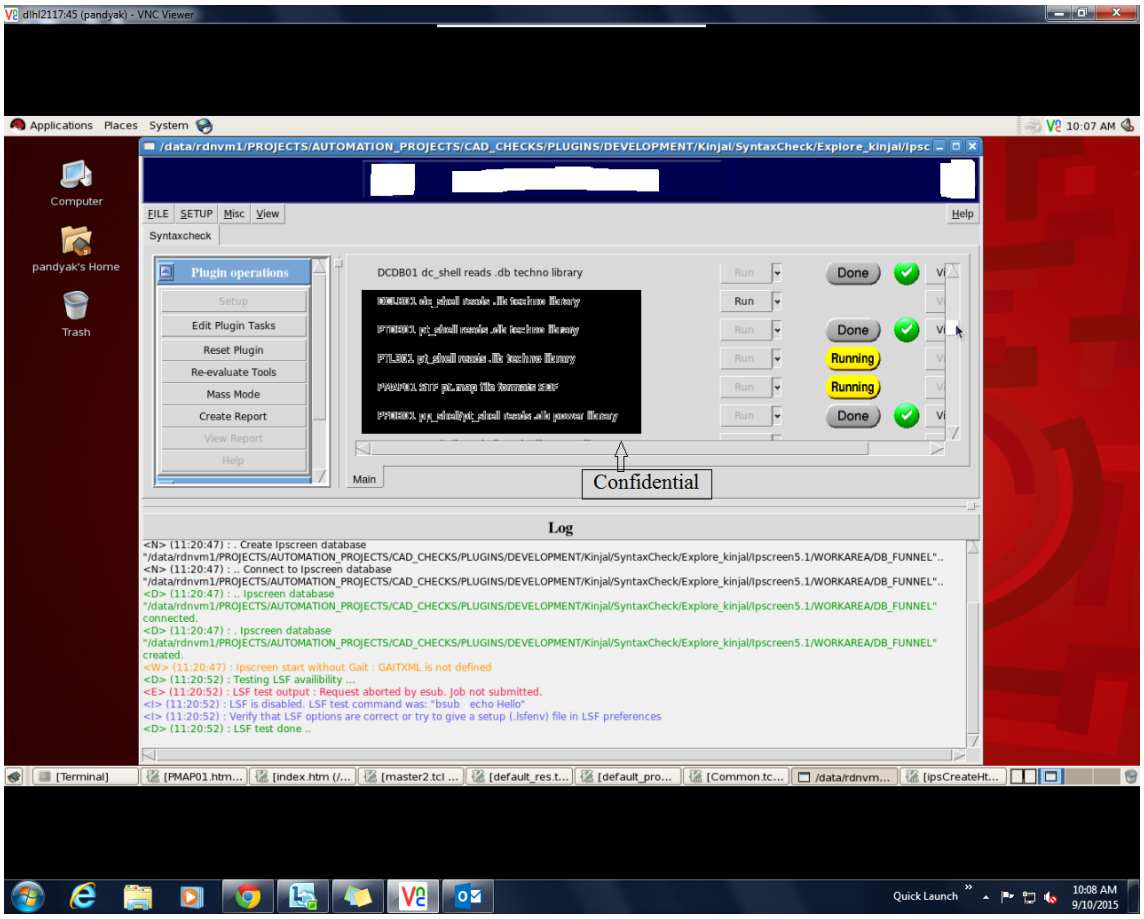

Figure 5.5: Checks executed Successfully

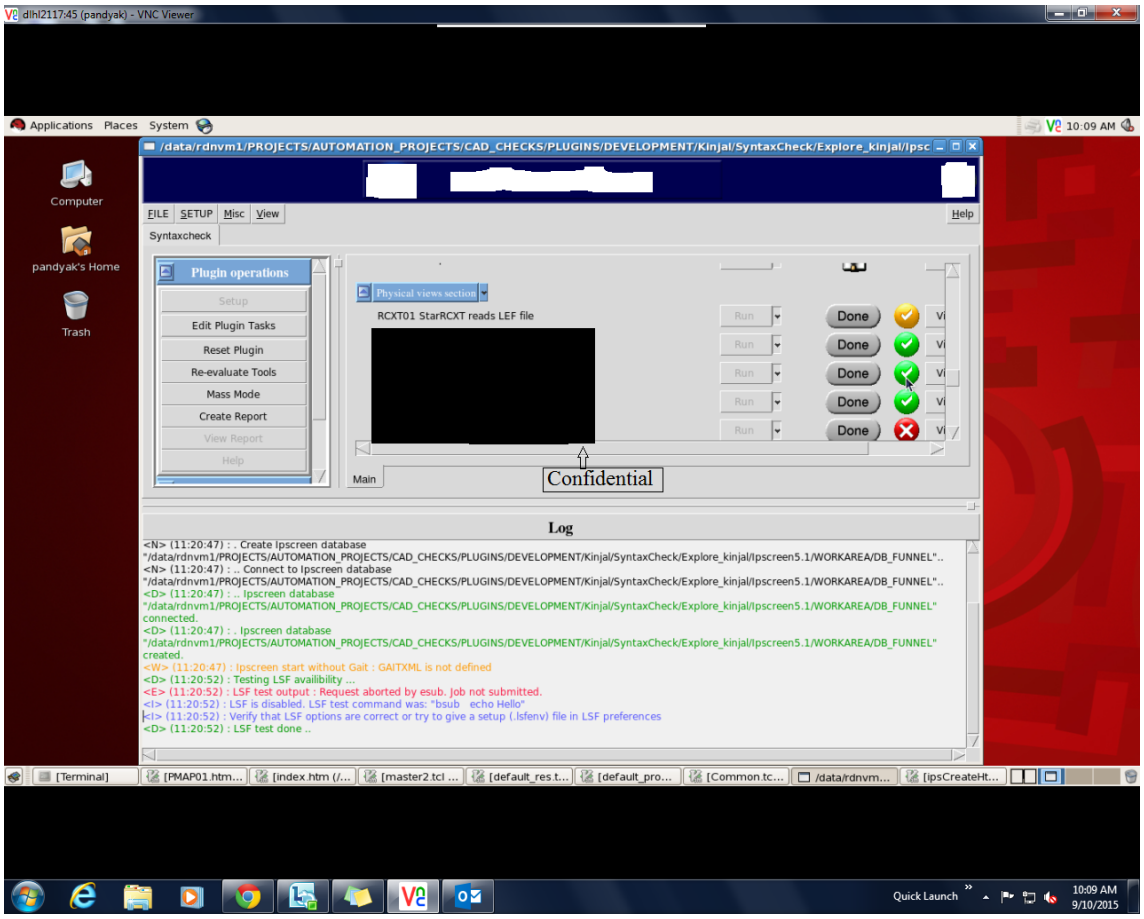

Figure 5.6: Checks generating Errors and Warnings

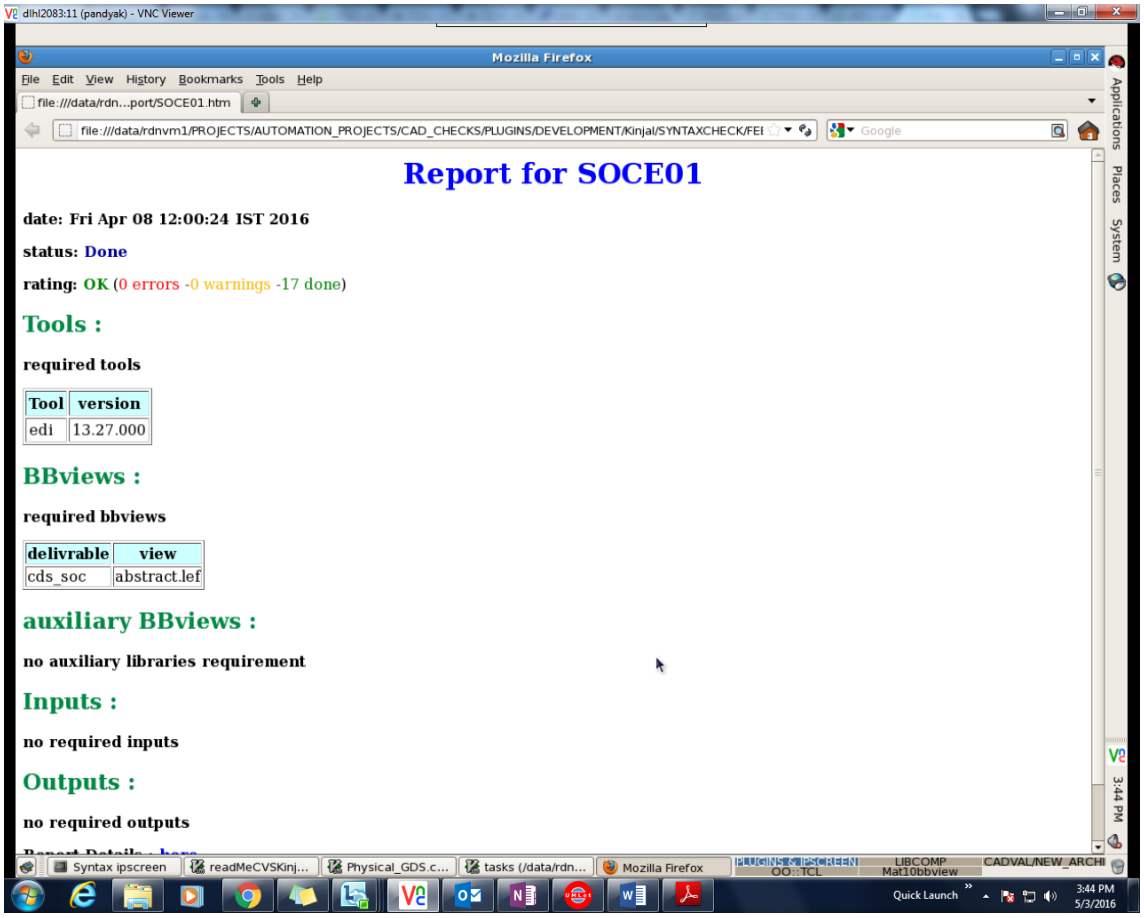

Figure 5.7: Report generated by IPScreen in HTML format

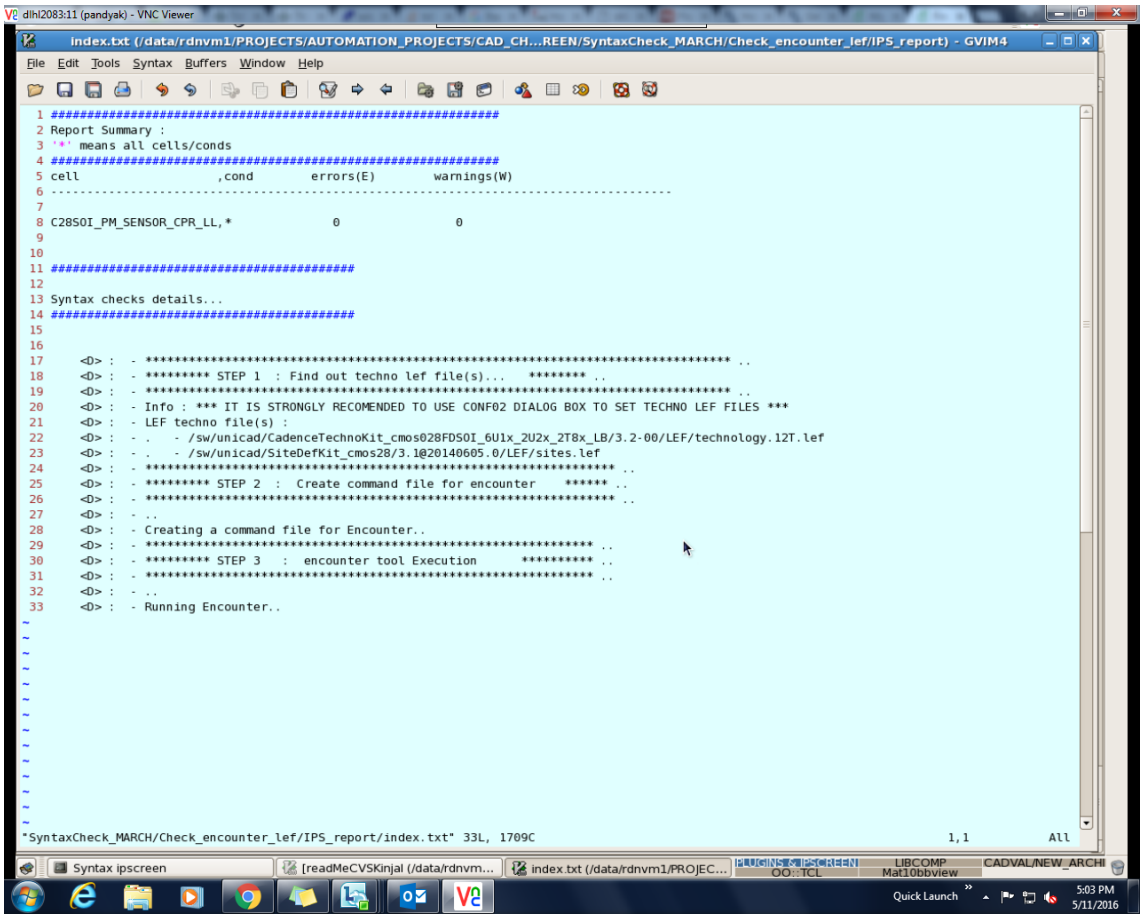

Figure 5.8: Report generated by IPScreen in TXT format

Chapter 6

Implementation Of new Architecture

As there are some limitations of validating libraries using IPScreen Framework, we are moving towards new archirtecture CadVal that will Faster the library validation process and the goal is to make this architecture as much intelligent as it can. The main goal is to minimize manual work which in turn reduces the complexity and makes validation process much faster.

6.1 Old Approach (IPScreen) vs CadVal

These plugins bundles 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 dierent 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.

6.2 Intermediate Plugin QA Kit - CadVal

Firstly by using an ST Internal tool, tool record and plugin record are prepared from technology specic data. This technology specic data is developed by the Management, and stores all the valid tools their versions, species the tools based on the type of IP(Memory/IO/CORE) etc.

I created a module that automatically parses some input given on command line and prepares input command le.The input given to module was name of library to validate, all its supporting IPs, path of the plugins to use. After the input command le is prepared, and tool records, plugin records are prepared the user automatically launches the job on the Load Sharing Facility. The validation cannot be done on local machines as the tools used require large computing resources and licenses that cannot be given on local machine and are uniformly given to users by the Load Sharing manager in a uniform manner. The gure shows block diagram of Intermediate PluginQAKit.

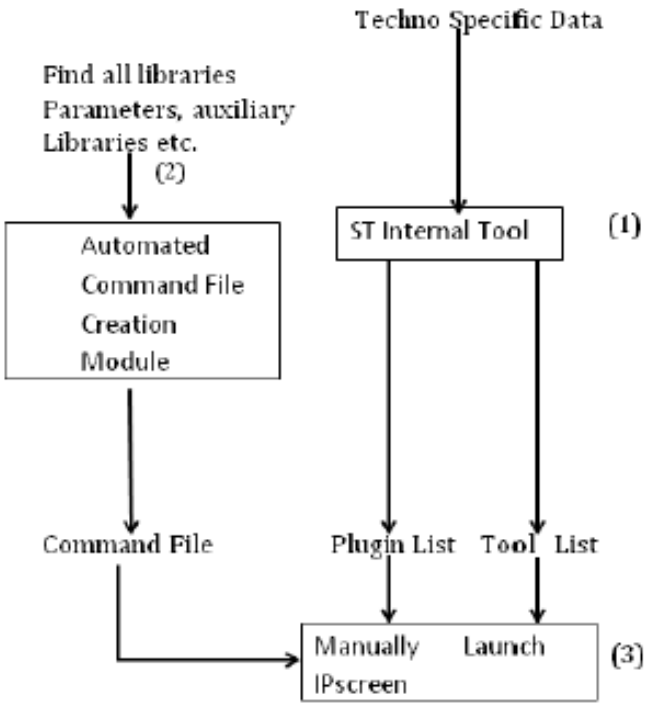

Figure 6.1: Block diagram of Intermediate Plugin QA Kit

6.3 CadVal Design and Flow

To make the library validation process much faster and to reduce the complexity, CadVal is mainly divided into 4 modules as follows:

1. **Input Processing** : As the name suggests, it will process the system inputs. Input can be of three type. It can be defualt specification file, it can the specification file given by user or it can be the input generated by tools. So this module will combine all the inputs coming from different different sources and merge all the inputs in single input file and process it. It will verify that inputs coming to the system are correct or not. Class Diagram in the Figure 6.2 shows the detail design of Input Processing Module.

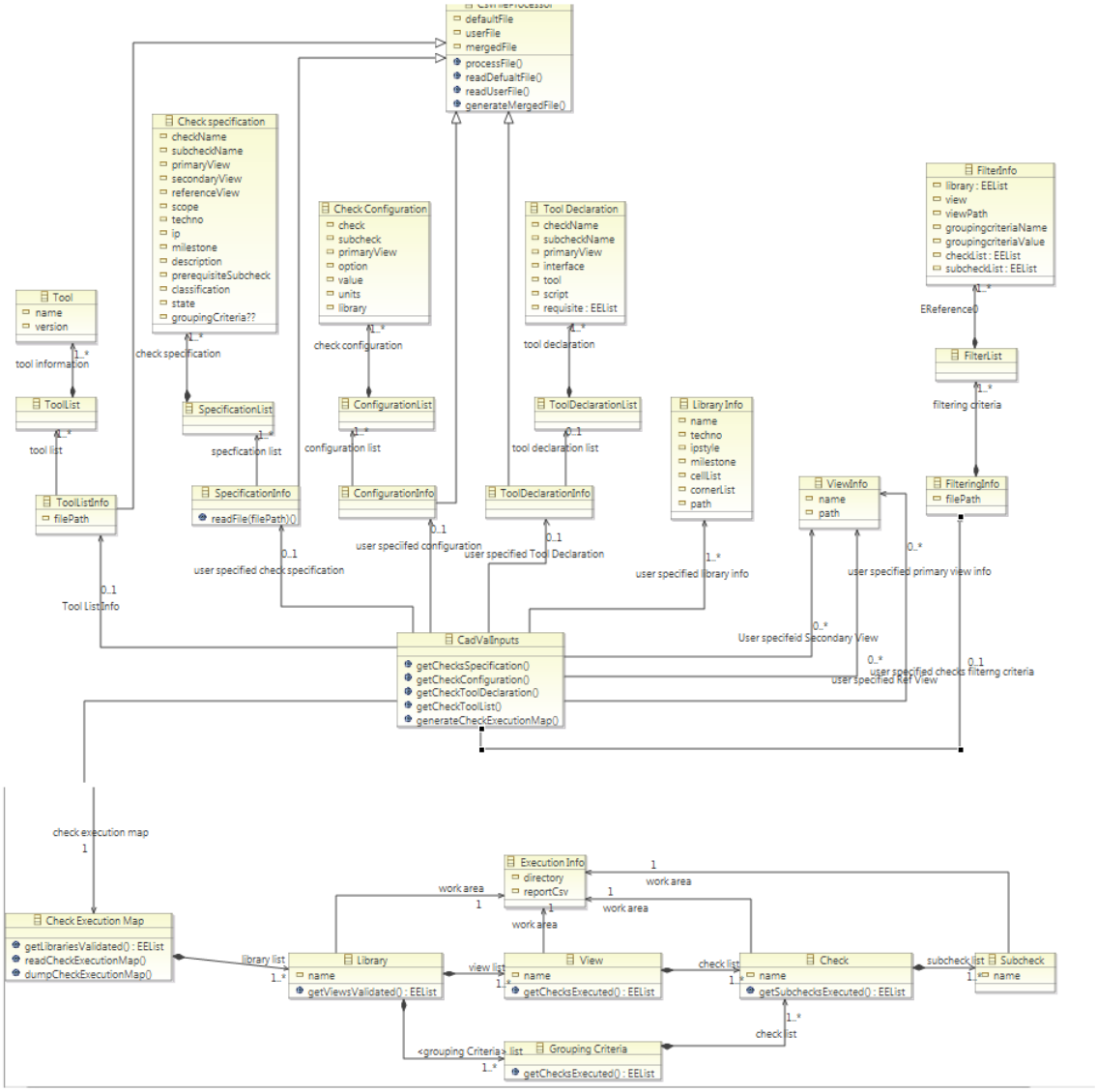

Figure 6.2: Input Processing Module

2. textbfSetup Generator : Task of Setup generator Module is to generate files and directories and to setup environment variables at each of the level (Library Level, View Level, Check Level, Sub-Check Level). so we can say setup generator will create a setup to run the system successfully. Class Diagram in the Figure 6.3 shows the detail design of Setup Generator Module.

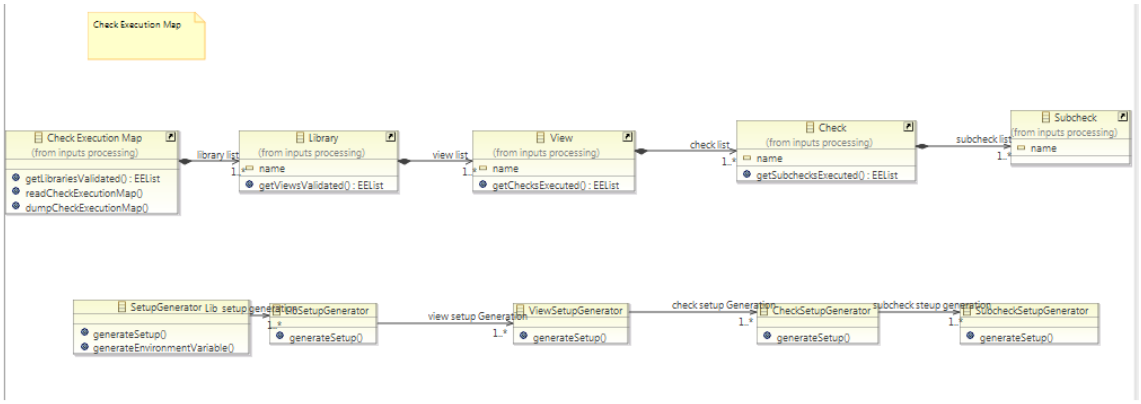

Figure 6.3: Setup Generator Module

3. **Check Executor** : As in the CadVal environment user can execute multiple Sub-Checks concurrently, task of Check Executor is to launch each Sub-check parallely on LSF (Load Sharing Facility). Check Executor also takes care about the dependency between the Sub-Checks. For example, if Sub-Check1 (LEF view Check) is dependent on Sub-Check2 (LIB View Check) then Check Executor will first execute the Sub-Check2 and after the completion of Sub-Check2 the Sub-Check1 will be executed. Class Diagram in the Figure 6.4 shows the detail design of Check Executor Module.

4. **Report Generator** : Report Generator module generates the report in different different format (CSV,TXT,XLS,HTML) by reading the LOG files generated after successful execution of each Sub-Check. When all the Sub-Checks reports get generated succefully, it will check the status of each sub-checks in the status file and then generate the Check Level Report then View Level Report and then at the end it will generate top level (i.e. Library Level) Report. Class Diagram in the Figure 6.5 shows the detail design of Report Genertion Module.

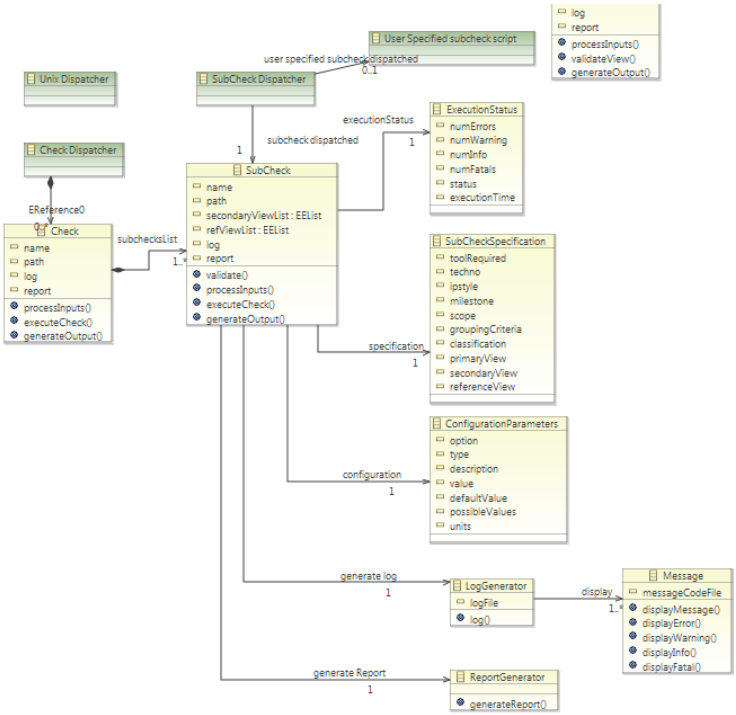

Figure 6.4: Check Executor Module

If we look at the flow and directory structure of the CadVal then it is divided as :

- Sub-Check Level : It is Similar to one task/check of an IPScreen. (E.x : SOCE01 - Encounter reads LEF File)
- Check Level : Check Level is a Collection of Multiple Sub Checks. (E.x : Syntax-Check,Cross-Check)
- View Level : View Level is a Collection of Multiple Checks. (E.x : LEF view, LIB view, CDL View)
- Library Level : The top most level in the CadVal architecture is a Library level. It is a collection of Multiple Library Views. (E.x : I/O Library, Memory Library)

So the flow will be like this, Top Most level Library level will call the functions of its inner

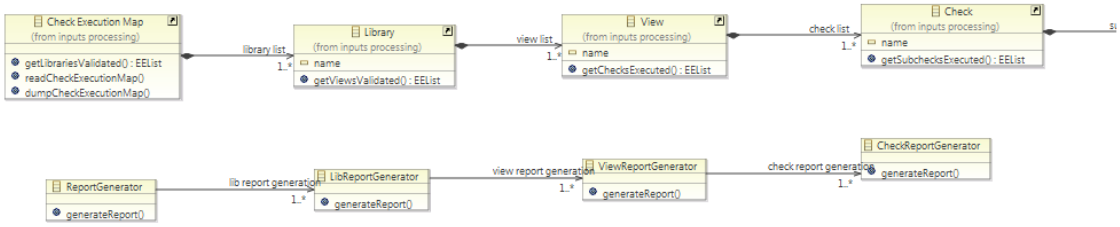

Figure 6.5: Report Generator Module

level i.e. View Level. View Level further calls Check Level and Check Level launches the individual Sub-checks on LSF by calling each Sub-Check that is specified by user in Specification File. After successful execution of each Sub-Checks, report generation will be done in Check, View and Library level.

6.4 CadVal Working

In CadVal if a user wants to validate some Library then he has to execute command by passing different different arguments and by setting options (like Library Name,Bbview-Path,View Name,sub-Check Name etc.) to those arguments. User can Run multiple Sub-Checks concurrently by launching each Individual Sub-Checks on LSF(Load Sharing Facility).

- Figure 6.6 shows the screen-shot of a script implemented in CadVal. This is a script to run EncounterLef Sub-Check to verify LEF view in SyntaxCheck Plugin. As given in name, Encounter is nothing but the name of tool used to validate the LEF view.

- To run this EncounterLEF Sub-Check onto the LSF, user has to run command on terminal that is shown in Figure 6.7

- After successful execution of EncounterLEF Sub-Check, Report Generator module will generate the reports in Txt,CSV and HTML format. Figure 6.8 shows the Sub-Check Level report in HTML format generated for EncounterLEF check ans the Figure 6.9 shows the same report in TXT Format. These reports contains the information about total no of Fatal, Errors and Warnings if occurs. They also gives detail description of each Fatal,Error,Warning and Info.

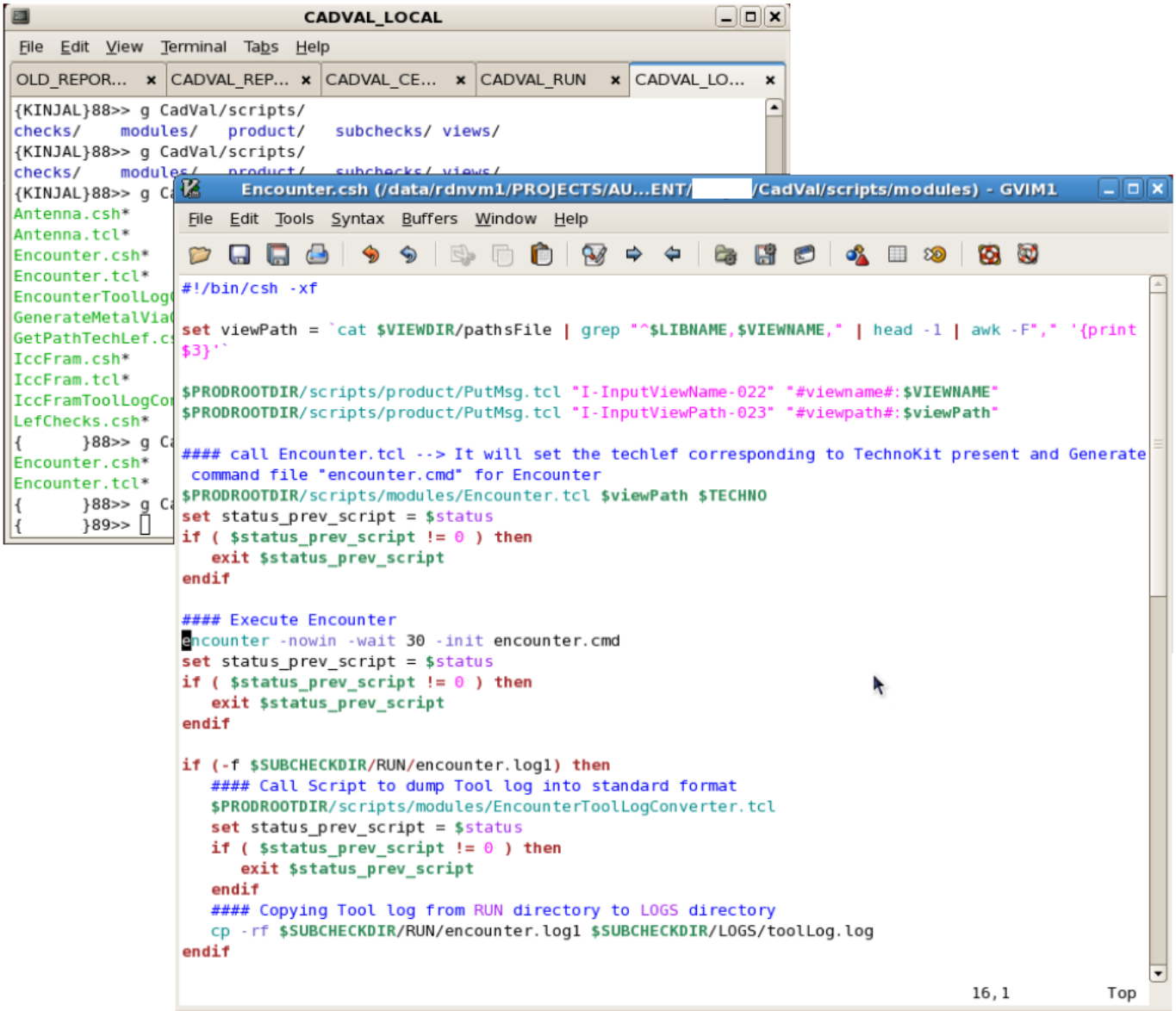

Figure 6.6: Shell Script for EncounterLEF Sub-Check

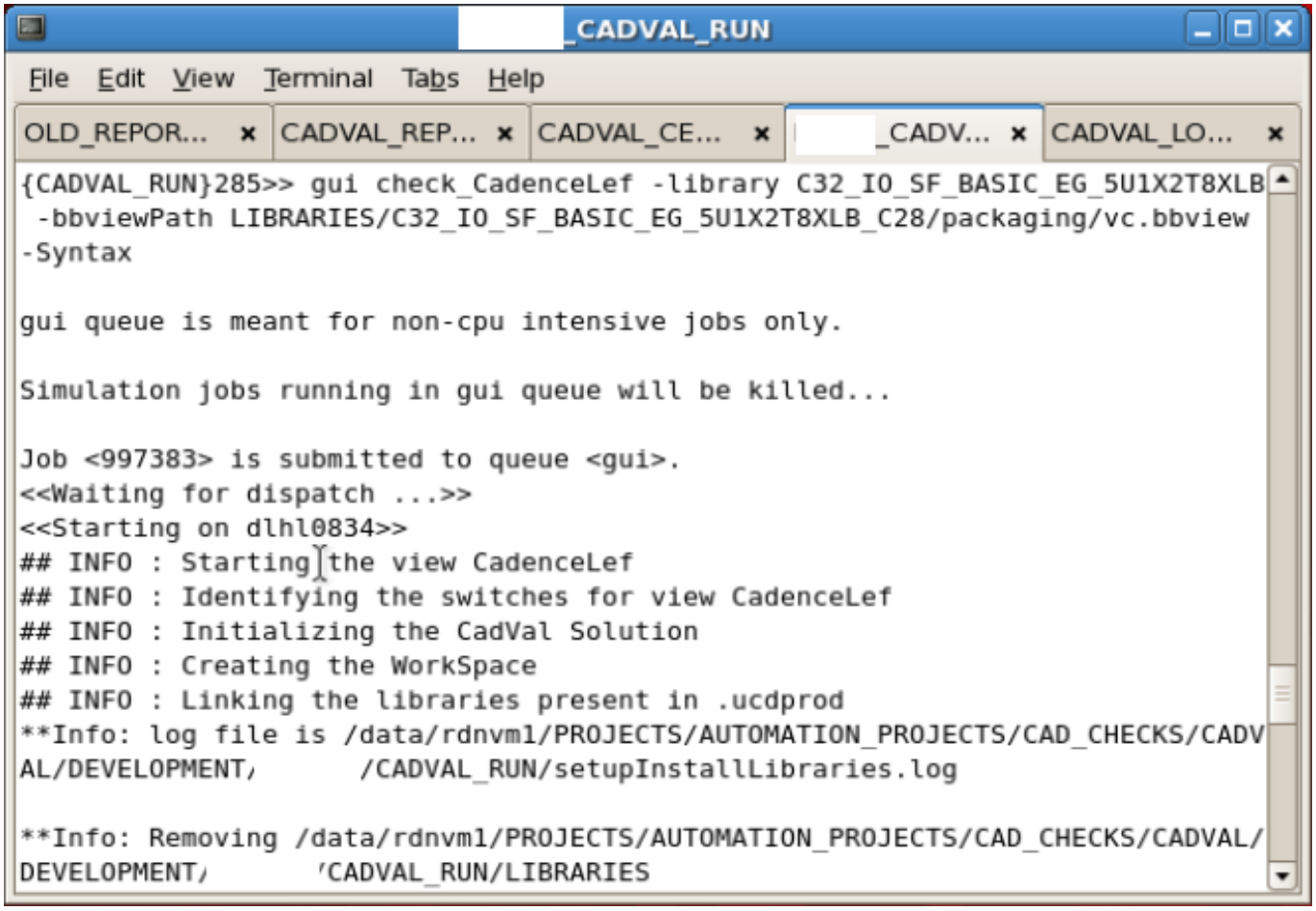

Figure 6.7: Executing EncounterLEF Sub-Check on Terminal

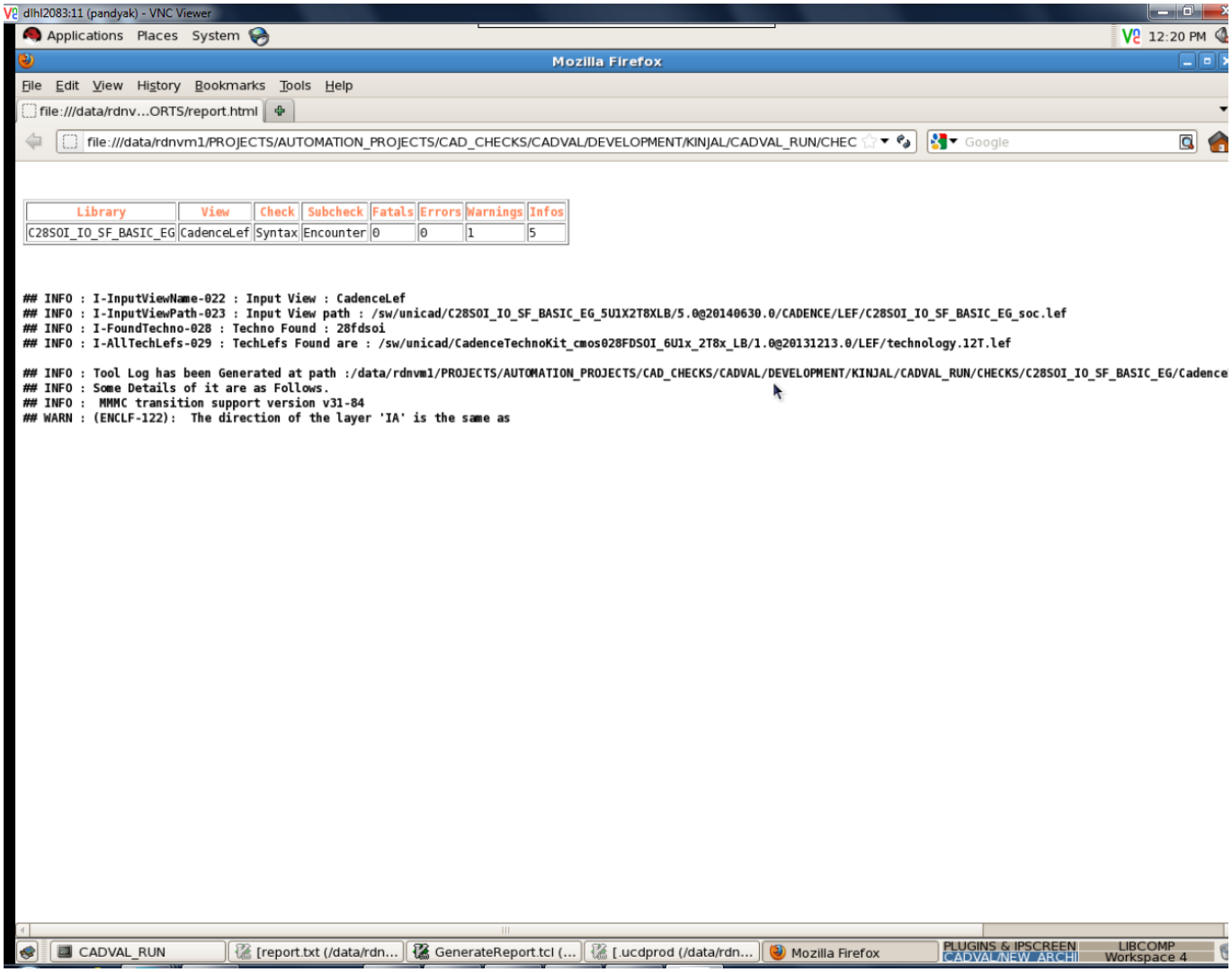

Figure 6.8: HTML Report Generated for EncounterLEF Sub-Check

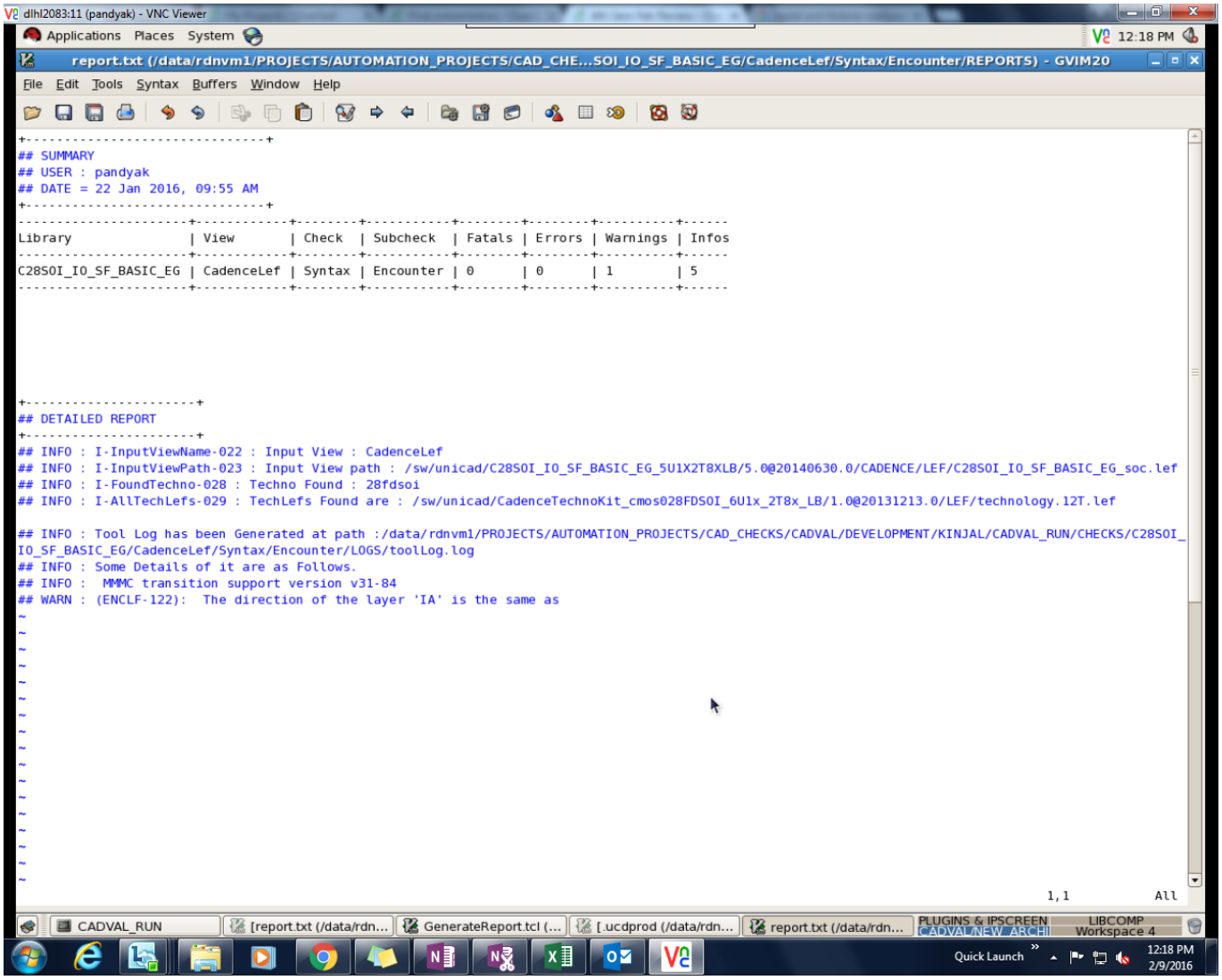

Figure 6.9: TXT Report Generated for EncounterLEF Sub-Check

Chapter 7

Future Enhancement

In future we can enhance the project by making the whole library validation process GUI (i.e. IPScreen) independent. By typing only a single command and passing the particular library views (that we want to validate) as an arguments we can get output in a few minutes/seconds. For that we have to develop CAD checks of Plugins which are independent of the environment of IPScreen. By validating the library views independent of IPScreen we can make validation process comparatively more easier and faster than using this IPScreen approach. So by this way we can enhance the project in future to make it less complex.

Chapter 8

Conclusion

So finally the conclusion is, validation process became much faster and easier using Plugin QA Kit approach rather than using manual approach of validating library views one by one using GUI.

8.1 Differences in Validation Using Manual Approach vs Plugin QA Kit

Some differences between Plugin QA Kit approach and manual approach are listed in Table 8.1 :

8.2 Time Difference Achieved After Automating Input Preparation

The Table 8.2 shows the time saved by automating the task of Input Preparation in CADVal.

Point	**Old Manual Approach to launch Plugins Using IPScreen**	**Automated Approach to launch Plugins Using CADVal**
Specifying Library, auxiliary Library	Manually find the index file of each library, its methodology and its reference libraries and write the commands in command-file.	The librarys central location is found out, and then its name,Methodology,reference library etc. are extracted and are written in input command-files.
Preparation of plugin record	File having plugin and its path was manually prepared.	The path is provided as an argument and the plugin file is automatically generated at run time.
Launching of Job on LSF	The job to be validated was manually launched on the LSF.	By default launches the job on compute farm and also provides user an option to bypass the launching of validation and just prepare the input.
Selection of Tools	Each time to validate a library all the tools required had to be written in a separate file. The correct tools depending on the techno of library had to be taken care of.	Now the correct tools are automatically picked up depending on the techno of library.
Creation of Work Space	For each library validation a different work space were manually created having command-file,plugin record,tools list.	Now,for N number of libraries of same Techno N Work Spaces will be created by CADVal's Setup Generator Module,in an automated manner.
Report Generation	Manually generate after all tasks completed using user interface.	Automatically Generated after tasks completion.

Table 8.1: Differences in Validation Using Manual Approach VS CADVal

No. of Libraries(of Same Technology)	Manual Approach of Input Preparation	Using CADVal
One Library	15-20 Minutes	2-3 Minutes
Two Libraries	25-30 Minutes	5-7 Minutes
Five Libraries	1.5-2 Hour	20-25 Minutes

Table 8.2: Time saved in Input Preparation for Validation

Bibliography

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

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

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

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

[5] http://wiki.tcl.tk/299.

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

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

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