Automation of IP CAD View Validation

Prepared By

Ritu Raj Agerwal

12MCEI01

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY, NIRMA UNIVERSITY

AHMEDABAD-382481

May 2014

Automation of IP CAD View Validation

Major Project

Submitted in the partial fulfillment of the requirements
for the degree of

Master of Technology in Information and Network Security

Prepared By

Ritu Raj Agerwal

12MCEI01

Guided By

Prof. Jigna Patel

and

Ms. Lipika Parwani

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY, NIRMA UNIVERSITY

AHMEDABAD-382481

May 2014

Certificate

This is to certify that the major project titled **"Automation of IP CAD View Validation"** submitted by Ritu Raj Agerwal (12MCEI01), towards the partial fulfillment of the requirements for the degree of Master of Technology in Information and Network Security of 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 major project, 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
Internal Guide,
Assoc. Professor,
Nirma University,
Ahmedabad

Prof. Sharda Valiveti
Program Co-ordinator - INS ,
Assoc. Professor,
Nirma University,
Ahmedabad

Ms. Lipika Parwani
External Guide, Project Manager
STMicroelectronics,
Greater Noida

Mr. Rishabh Bansal
Project Mentor
STMicroelectronics,
Greater Noida

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

Dr. K. Kotecha
Director,
Institute of Technology,
Nirma University
Ahmedabad

Undertaking for Originality of the Work

I, **Ritu Raj Agerwal (12MCEI01)**, give undertaking that the Major Project titled **"Automation of IP CAD View Validation"** submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Information and Network Security** of Nirma University, Ahmedabad, is the original work carried out by me and I give assurance that no attempt of plagiarism 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.

Ritu Raj Agerwal (12MCEI01)
Date:
Place:

Endorsed by

Prof. Jigna Patel	Ms. Lipika Parwani
Internal Guide	External Guide

Acknowledgement

I am deeply indebted to my thesis supervisors Prof. Jigna Patel, Internal Guide Nirma University and Ms. Lipika Parwani, Project Manager ST Microelectronics for their constant guidance and motivation.

I would like to express my gratitude and sincere thanks to Dr. Sanjay Garg Head of Computer Science and Engineering Department and Prof. Sharda Valiveti Coordinator M.Tech Information and Network Security program for allowing me to undertake this thesis work and for his guidelines during the review process.

I would also like to thank Dr K Kotecha Director, IT, NU for allowing me to undertake this project work.

I also wish to thank Rishabh Bansal, Amit Kotadia, Nancy Kapila and Mohit Bhasin for their help and support. Without their experience and insights, it would have been very difficult to do quality work.

I wish to thank my friends of my class for their delightful company which kept me in good humor throughout the year.

Last, but not the least, no words are enough to acknowledge constant support and sacrifices of my family members because of whom I am able to complete the degree program successfully.

- RITU RAJ AGERWAL
(12MCEI01)

Abstract

The project deals with Automation of Libraries Validation that means validation of libraries is done through Automated Scripts instead of doing it manually. This automation contains some frameworks (PLUG-INS) which check the different views of the under test library. Plug-ins check the static views, modeling views, cross views and mat structure of the library. This Plug-ins are built on the TCL (Tool Command Language).

Now a days every Validation Should be Automatic and take very less time, So in order to meet such requirement a new Unified Environment is to be made which would be much faster than existing solution and will occupy less space. This whole new cocept will be a mixture of Tcl, Shell and Perl Language.

Contents

List of Figures

Chapter 1

Introduction

SOC design trend is to integrate more and more functionalities usually named IP's (Intellectual Property), either developed concurrently in the design team or already available in market. One of the critical jobs for IP providers is related to IP models or representation needed in the design flow targeted for the SOC development. The main considerations for IP developers are:

1. Coherency between IP models
2. Accuracy between IP models and real IP
3. Compatibility of IP models with SOC design flow

This chapter consists of brief introduction to library IPs, types of libraries and definition of certification.

1.1 Introduction of Library IP

[1] A LIBRARY is defined as a set of all design data available for an IP. Design data consists of functionality, transistor level design of the IP, Actual Mask Level Design which will be fabricated and Timing information. We can also say that library is a collection of cells, comprising of various views which are useful for

designing a chip. Cell is a component performing a basic function and a view is a particular representation of a cell. Due to Increased complexity and shorter time to market SOC designers cannot really concentrate on design of basic building blocks. This database can be reused in various SOCs.

There are different types of libraries present:

1.1.1 CORE Library:

1. It consists of a group of cells called standard cells.
2. They implement the basic logic functions.
3. Examples are Inverter, AND, Flops, Latches etc.
4. Those cells are already designed in a specific process technology.
5. Physical/ logic/ timing/ electric models are already created for those cells.

1.1.2 IO Library:

1. It consists of a group of cells called I/O buffers.
2. Those cells are already designed in a specific process technology
3. I/O buffers are designed to interface the off-chip signals to inside chip environment and vice-versa
4. I/O's are placed on the periphery of the chip .Any signal which comes from off-chip environment (external voltages are at a typical voltage of 2.5V, 3.3V or 5V) into the chip, must be checked by I/O for any discrepancy in its behavior other than defined by the core for that particular signal. If I/O finds any signal defying the behavior expected from it, it modifies the signal so as to ensure proper functioning of chip.

5. I/O's also act as protection devices for the core. I/O also scans the signal which is going from core to off-chip world.

1.1.3 Memory:

1. These contain Memories of Various Architectures.

2. Examples are SRAM, DRAM and ROM

3. At ST these are implemented as generators. Since there can be a number of memory sizes we implement the basic building blocks (e.g. memory cell, Decoding logic, sense amplifier etc.) and configure the generation of various memory sizes.

4. The basic building blocks are already designed in a specific process technology.

1.1.4 Analog and Mixed Signal Library:

1. These contain IPs providing some innovative functions.

2. Examples are PLL, DAC, USB, etc.

3. These can be implemented as full custom hand crafted cells or in a semi-custom manner using the CORE libraries.

1.2 Introduction of Certification

[5] CERTIFICATION is the process of validating hard and soft IPs for CAD view compliance on various platforms (Synopsys, Cadence, Magma etc.). In Certification process we check the various views present in the library/IP, Consistency across views, Design flow compliance of views and Compliance of views in the integrated environment. The final target of the certification is to get the library IP which has

consistent views and the usability of UNICAD flow with respect to the quality and productivity of the SOC implementations.

1.3 My Contribution In The Thesis

Validation of different libraries within different technologies has done as the part of the methodology. In this part, Some Plug-ins (ex. Syntaxcheck, Modelization, Crosscheck, Mat10bbview etc.)are made which basically provide the platform to validate the library. Library has to be loaded in this platform to being validated. Some modules are also made my be me as described later(like automation of manufecturing design check, access rights checking, automated setup creation with vey less information given by the user and pre tool checking module) to automate the manual process. Now as a replacement of IPScreen framework we are going to introduce Unified environment in future .

1.4 Organization of The Thesis

In chapter 2, Literature Survey and background theory. In chapter 3, Tools and Technology which i am using.

In chapter 4, Validation and certification of many library are done. In chapter 5,New Unified environment which can be the replacement of IPScreen. In chapter 6, I have explained Implementation of my work and intermediate results which i had obtained.

In chapter 6, Conclusion and Future work explained.

Chapter 2

Literature Survey (Background Theory)

As discussed in the previous chapter that a Library is a collection of cells and each cell has a certain view which is a particular representation of the cell. Each cell may have Layout view, Schematic view, Symbolic view, Timing view etc. A cell is delivered as a set of view and each view is used by different tool in a given design flow.

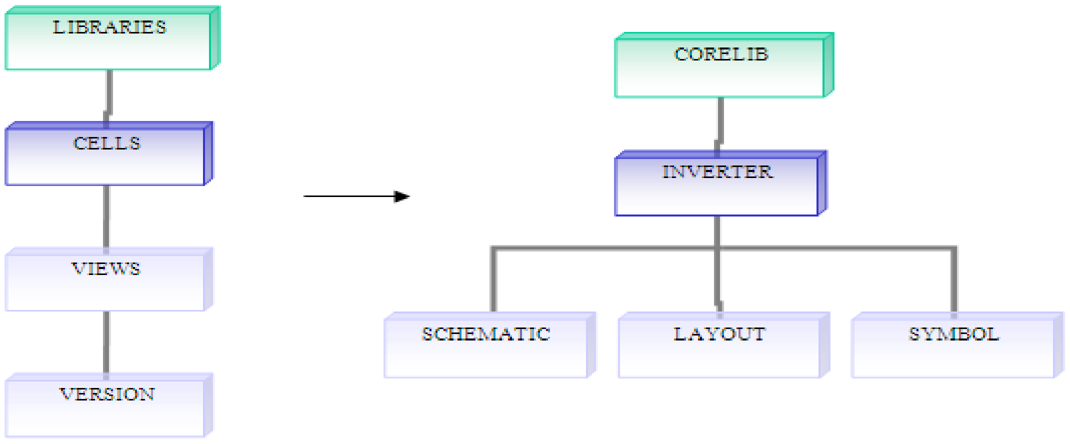


Figure 2.1: Library View Block Diagram
[2]

The basic library views are classified into:

1. Front End Views: Views related to the timings or modelling of cell.

2. Back End Views: Views related to the physical design of a cell.

They are further classified into other views as shown in the figure below:

2.1 Back End Views

[2] The different types of back end views are symbol, schematic view, layout view, abstract view and so on.

2.1.1 Symbol View

A Symbol view is a pictorial representation of a cell. It includes pins, symbol graphics, labels and a selection box. Pins are input and output of a symbol. The shape of the symbol indicates the cell function. Labels in the symbol are used to add to the documentation of the design. Selection box in the symbol, defines the

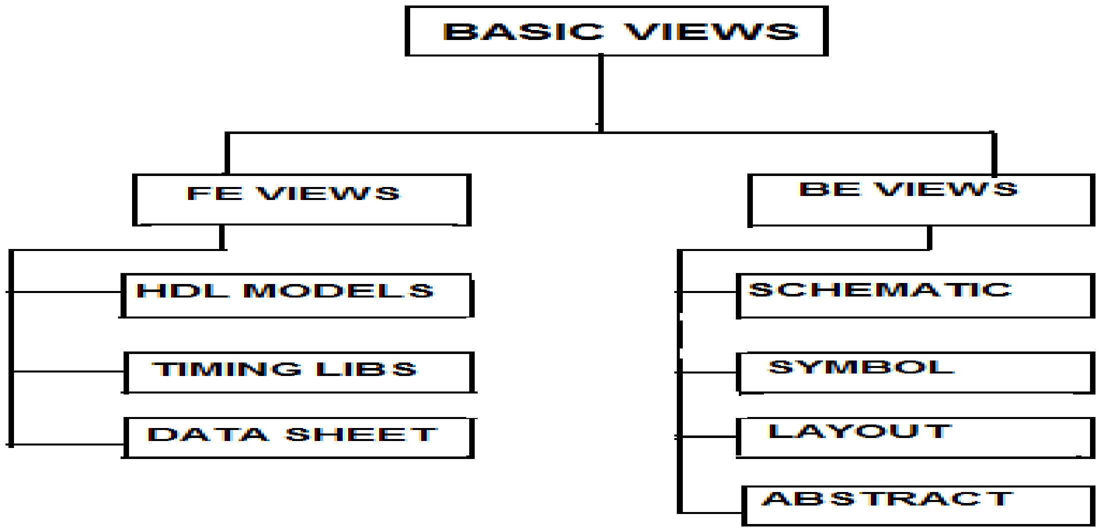


Figure 2.2: Classification of Library Views
[2]

area of the symbol that an instance will be selectable by. A symbol view for an inverter looks as shown in figure below:

2.1.2 SLIB

SLIB view is the derived view of symbol. It is actually the text representation of the symbol view.

2.1.3 Schematic View

A schematic is a simplified representation of an electrical circuit. It shows the different components of the circuit as simplified standard symbols, and the power and signal connections between the devices. Schematic is the representation of a cell at the transistor level. A schematic view includes component instances, wires and pins.

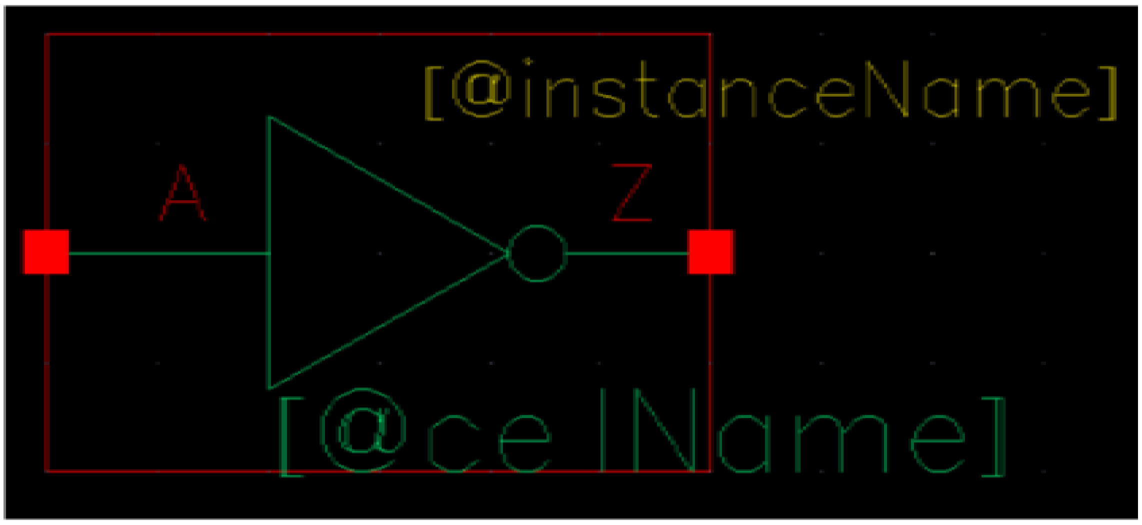


Figure 2.3: Symbol View
[2]

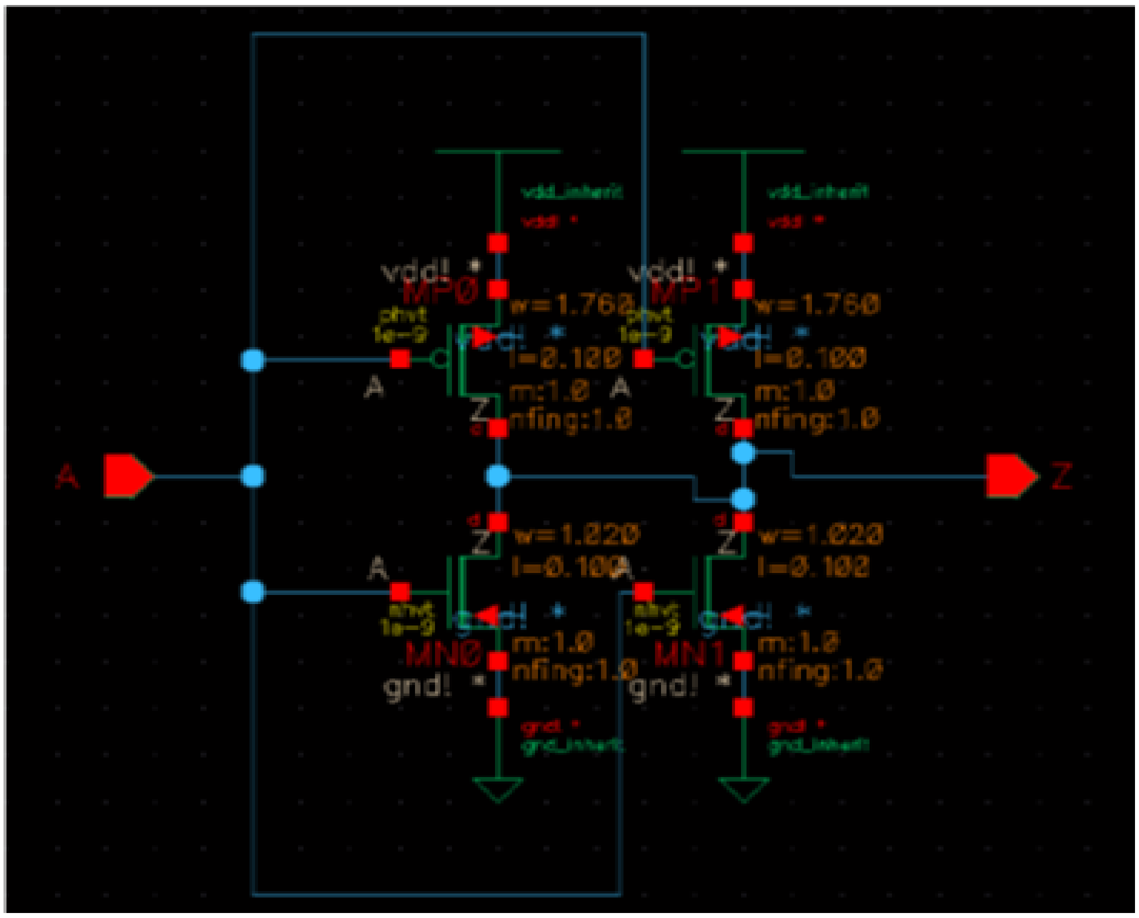


Figure 2.4: Schematic View
[2]

.ends

2.1.4 Layout View

Layout view is the actual physical representation of the electrical circuit of the cell that goes on the silicon.

2.1.5 Synopsys Technology File (.libs)

A technology library (STF/.lib) describes the structure, function, timing and environment of the ASIC technology being used. The technology library consists of the following information for synthesis:

1. Mapping: Functional information for each cell.
2. Optimization: Area and timing information for each cell.
3. DRC: DRC rule constraints on cells.

The .libs are defined on various PVT conditions. These PVT conditions basically specify the conditions on which the cells of the library are characterized.

2.2 Summary

In this chapter, it is shown that the different library views which are mainly classified in two ways. 1) FE (Front End) and 2) BE (Back End).

Chapter 3

Tools and Technology

3.1 Scripting Introduction

[9]

Scripting languages such as Perl and Tcl represent a very different style of programming than system programming languages such as C or JavaTM. Scripting languages are designed for "gluing" applications; they use typeless approaches to achieve a higher level of programming and more rapid application development than system programming languages. Increases in computer speed and changes in the application mix are making scripting languages more and more important for applications of the future.

3.1.1 Comparison chart with Non-Scripting Language

A comparison of various programming languages based on their level (higherlevel languages execute more machine instructions for each language statement) and their degree of typing. System programming languages such as C tend to be strongly typed and medium level (five to 10 instructions per statement). Scripting languages such as Tcl tend to be weakly typed and very high level (100 to1,000 instructions per statement).

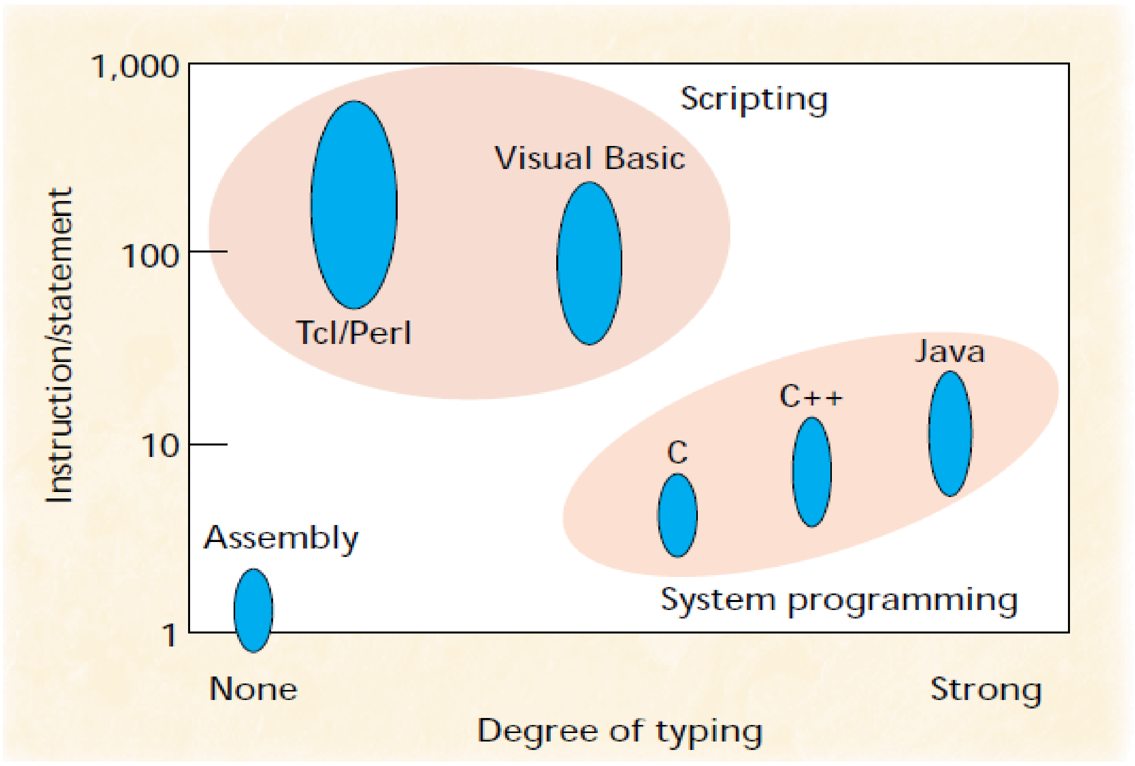


Figure 3.1: Comparison Chart with other languages
[9]

3.2 Technology Used

I have used the following scripting languages on UNIX platform.

- C Shell.[6]
- Tool Command Language (TCL)[7]
- Perl.[8]

3.2.1 C Shell

[6] Some Points about C Shell

- The C shell is a command processor typically run in a text window, allowing the user to type commands.
- It supports filename wildcarding, piping, here documents, command substitution, variables and control structures for condition-testing and iteration.
- The feature in this shell is the overall style of the language looked more like C and was seen as more readable.
- The main design objectives for the C shell were that it should look more like the C programming language and that it should be better for interactive use.
- It introduced numerous new features that made it easier, faster and more friendly to use by typing commands at a terminal. Users could get things done with a lot fewer keystrokes and it ran faster.
- The C shell operates one line at a time. Each line is tokenized into a set of words separated by spaces or other characters with special meaning, including parentheses, piping and input/output redirection operators, semicolons, and ampersands.

3.2.2 Tool Command Language (Tcl)

[9] In this section I have described Introduction and features of TCL.

- **Introduction**

  - Tcl, or Tool Command Language, is a an elegant, versatile, feature-rich, simple-to-learn yet very powerful industrial-strength open-source programming language and development platform.

  - Tcl is one of the only languages that can replace both the smallest and largest programming tasks.
  - Tcl syntax is described in just a dozen rules.
  - There are no reserved words and no specialized syntax for control structures, conditionals, and so on.
  - In addition to being a programming language, Tcl is also a cross-platform C library.

- **Features**

  - Cross-platform scripting.
  - Cross-platform C library
  - String handling
  - Regular expressions

3.2.3 Perl

[8] In this section I have described Introduction about perl.

- **Introduction**

  - Perl is a family of high-level, general-purpose, interpreted, dynamic programming languages.
  - Though Perl is not officially an acronym,[5] there are various backronyms in use, such as: Practical Extraction and Reporting Language.
  - Perl is a stable, cross platform programming language.
  - Perl is an Open Source software, licensed under its Artistic License, or the GNU General Public License (GPL).

- **Features**

  - Perl takes the best features from other languages, such as C, awk, sed, sh, and BASIC, among others.
  - Perl supports both procedural and object-oriented programming.
  - The Perl interpreter can be embedded into other systems.
  - Perl is an interpreted, which means that your code can be run as is, without a compilation stage that creates a non portable executable program.

3.3 Summary

In this section I have described literature survey of scripting language and languages which I am using in my project.

Chapter 4

IPSCREEN and PLUG-INS

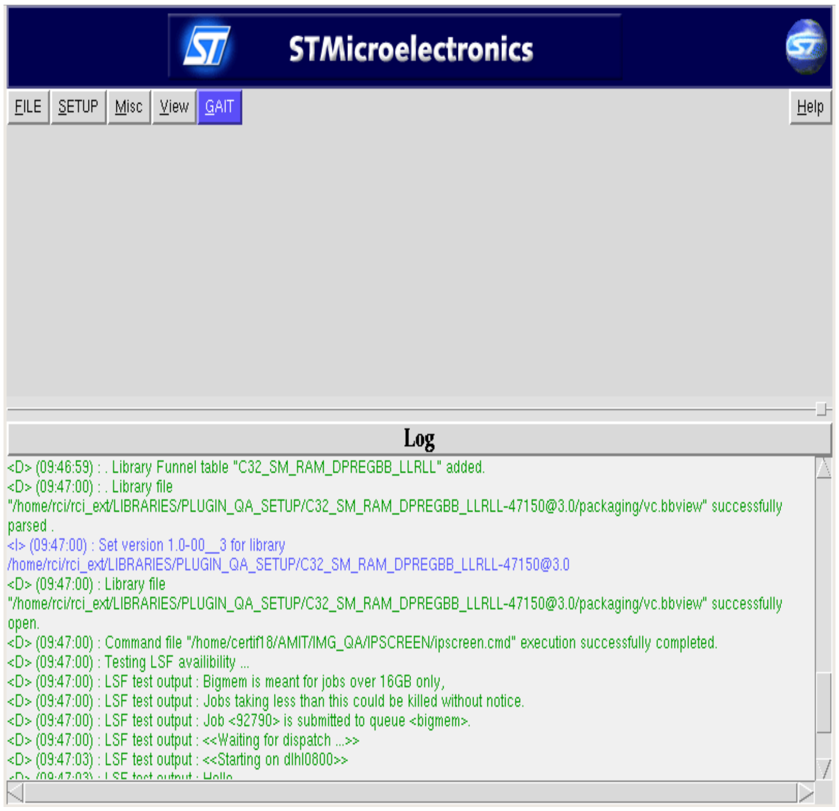
IPSCREEN[14] is the framework on which PLUG-INS are going to be run. In fact, IPSCREEN and PLUG-INS both are tools but IPSCREEN[14] provide the basic environment to execute the PLUG-INS.

Many PLUG-INS can be loaded in IPSCREEN[14] simultaneously. One PLUG-IN can also execute for many different libraries. So, Likewise every library and every PLUG-IN can be loaded in IPSCREEN[14] at the same time.

All PLUG-INS can be run at the same time for different libraries. Parallel certification can also be done for different libraries to reduce the time required for certification of those libraries while doing serially.

4.1 IPSCREEN

[14] IPSCREEN is one kind of tool which is basically used for checking the design statically. Many libraries and PLUG-INS can be loaded simultaneously. For each library having each PLUG-IN generate a new tab for certification.



```
<D> (09:46:59) : . Library Funnel table "C32_SM_RAM_DPREGBB_LLRLL" added.
<D> (09:47:00) : . Library file
"/home/rci/rci_ext/LIBRARIES/PLUGIN_QA_SETUP/C32_SM_RAM_DPREGBB_LLRLL-47150@3.0/packaging/vc.bbview" successfully
parsed .
<I> (09:47:00) : Set version 1.0-00__3 for library
/home/rci/rci_ext/LIBRARIES/PLUGIN_QA_SETUP/C32_SM_RAM_DPREGBB_LLRLL-47150@3.0
<D> (09:47:00) : Library file
"/home/rci/rci_ext/LIBRARIES/PLUGIN_QA_SETUP/C32_SM_RAM_DPREGBB_LLRLL-47150@3.0/packaging/vc.bbview" successfully
open.
<D> (09:47:00) : Command file "/home/certif18/AMIT/IMG_QA/IPSCREEN/ipscreen.cmd" execution successfully completed.
<D> (09:47:00) : Testing LSF availibility ...
<D> (09:47:00) : LSF test output : Bigmem is meant for jobs over 16GB only,
<D> (09:47:00) : LSF test output : Jobs taking less than this could be killed without notice.
<D> (09:47:00) : LSF test output : Job <92790> is submitted to queue <bigmem>.
<D> (09:47:00) : LSF test output : <<Waiting for dispatch ...>>
<D> (09:47:03) : LSF test output : <<Starting on dlhl0800>>
<D> (09:47:03) : LSF test output : Hello
```

Figure 4.1: GUI of IPSCREEN
[14]

Discussion: It shows how the PLUG-INS and library are being loaded in the IPSCREEN framework. the dialogue box part of the figure shows the loading process of the library and PLUG-INS simultaneously.

4.2 PLUG-INS

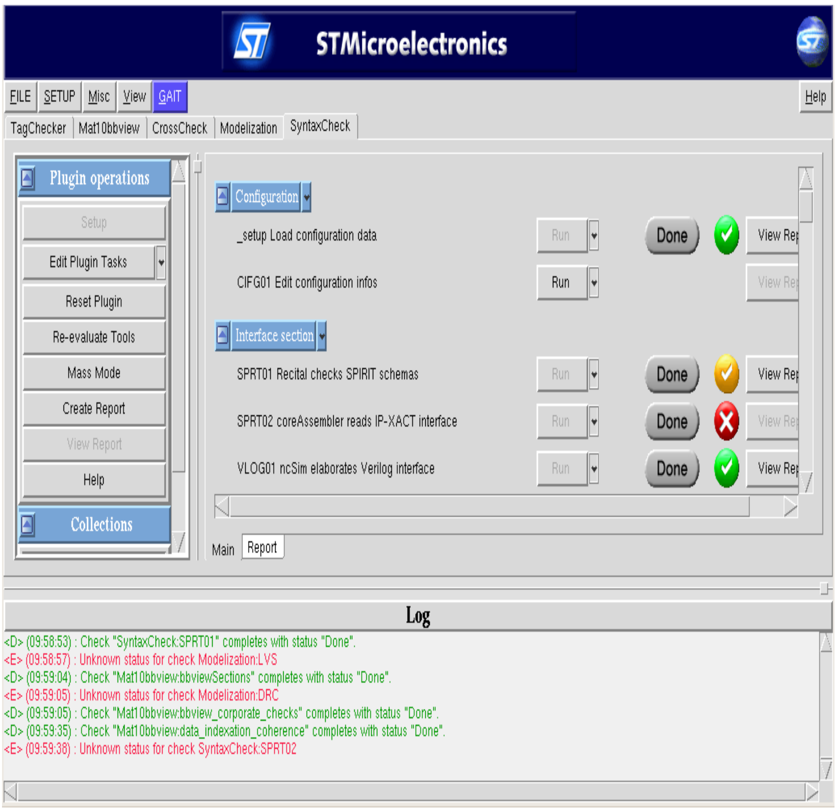
[14] PLUG-INS are very useful to check the design and their views. Plug-ins basically work on the static checks . these checks are made in the TCL(Tool Command Language)[5]. Different plug-ins are shown below.

1. Syntaxcheck
2. Modelization
3. Mat10bbview
4. Tagchecker
5. Crosscheck
6. Mat10comp

4.2.1 Syntaxcheck

This plug-in covers the syntax part of the Library. It means that the plug-in is used to check the syntax of any files (which are inside the views) is correct or not.

For ex. if one stf (Synopsys Technology File) wants to be checked syntactically by using anyone CAD tool then first the file has to be opened in the required tool. Command used to check that file is going to be executed which checks the file.



_setup Load configuration data
CIFG01 Edit configuration infos
SPRT01 Recital checks SPIRIT schemas
SPRT02 coreAssembler reads IP-XACT interface
VLOG01 ncSim elaborates Verilog interface

```
<D> (09:58:53) : Check "SyntaxCheck:SPRT01" completes with status "Done".
<E> (09:58:57) : Unknown status for check Modelization:LVS
<D> (09:59:04) : Check "Mat10bbview:bbviewSections" completes with status "Done".
<E> (09:59:05) : Unknown status for check Modelization:DRC
<D> (09:59:05) : Check "Mat10bbview:bbview_corporate_checks" completes with status "Done".
<D> (09:59:35) : Check "Mat10bbview:data_indexation_coherence" completes with status "Done".
<E> (09:59:38) : Unknown status for check SyntaxCheck:SPRT02
```

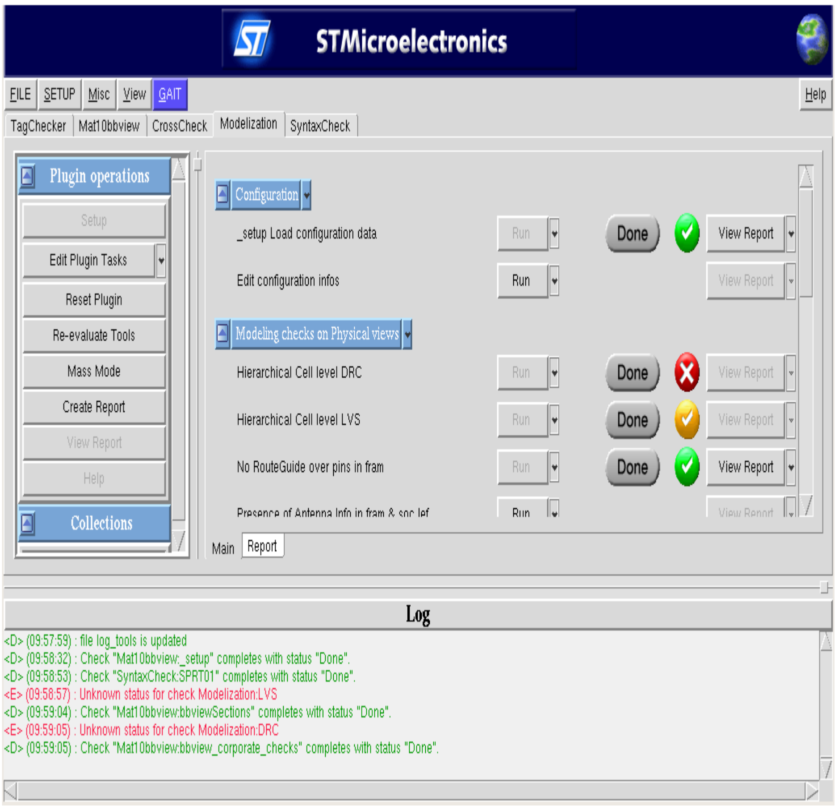
Figure 4.2: GUI of Syntaxcheck PLUG-IN
[14]

Discussion: Different tasks are shown in the task area. The side part of the task shows different status of the task. Red part shows the task is checked with errors. Yellow part shows the task is checked with warnings. Green part shows the task is checked with successfully. The dialogue box contains some red lines shows errors coming during the loading of library or PLUG-INS.

4.2.2 Modelization

This plug-in covers the modeling of the views. It means that the views are modeled correctly or not statically. This ensures accurate library functionality in the flow.

For ex. one stf (Synopsys Technology File) wants to be checked whether the required attributes are present in that file then some front end checks are going to be operated on that file and finally one command will generate the final report for the same.



```
<D> (09:57:59) : file log_tools is updated
<D> (09:58:32) : Check "Mat10bbview:_setup" completes with status "Done".
<D> (09:58:53) : Check "SyntaxCheck:SPRT01" completes with status "Done".
<E> (09:58:57) : Unknown status for check Modelization:LVS
<D> (09:59:04) : Check "Mat10bbview:bbviewSections" completes with status "Done".
<E> (09:59:05) : Unknown status for check Modelization:DRC
<D> (09:59:05) : Check "Mat10bbview:bbview_corporate_checks" completes with status "Done".
```

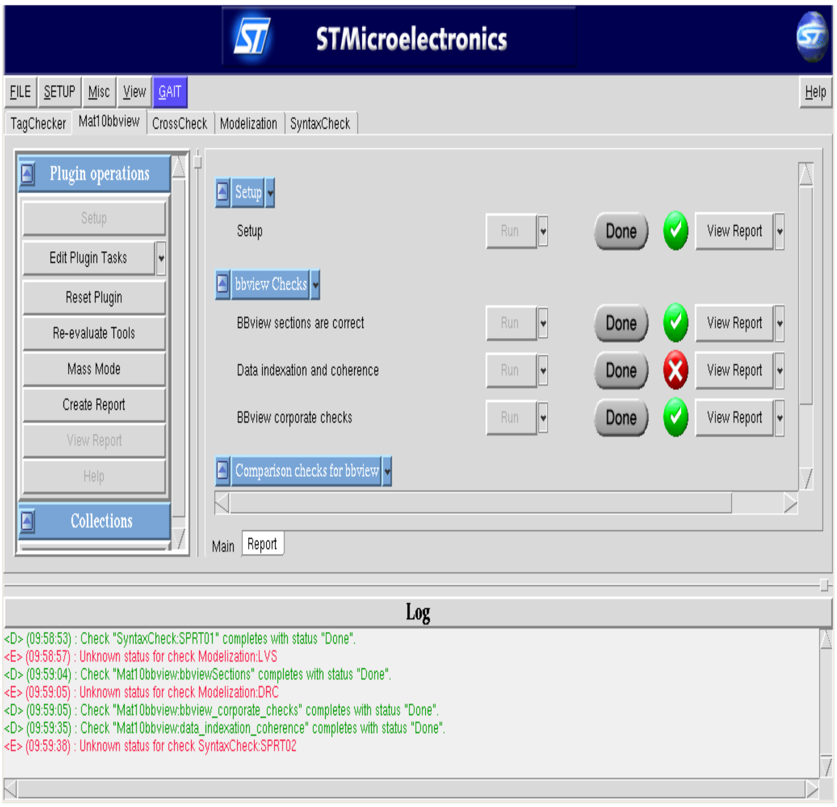
Figure 4.3: GUI of Modelization PLUG-IN
[14]

Discussion: The figure contains the different modeling tasks which are performed on the library views. The figure shows the DRC with errors, LVS with warnings and FRAM view with successful result.

4.2.3 Mat10bbview

This plug-in is basically used for checking in the sense of views presence in the library structure. It also checks that mandatory views are present, and aligned with library structure.

The indexation of any library is done correctly or not and according to that indexation, the library is structured as per the standard guidelines or not, is going to be checked statically by using the basic TCL (Tool Command Language)[13] commands.



```
<D> (09:58:53) : Check "SyntaxCheck:SPRT01" completes with status "Done".
<E> (09:58:57) : Unknown status for check Modelization:LVS
<D> (09:59:04) : Check "Mat10bbview:bbviewSections" completes with status "Done".
<E> (09:59:05) : Unknown status for check Modelization:DRC
<D> (09:59:05) : Check "Mat10bbview:bbview_corporate_checks" completes with status "Done".
<D> (09:59:35) : Check "Mat10bbview:data_indexation_coherence" completes with status "Done".
<E> (09:59:38) : Unknown status for check SyntaxCheck:SPRT02
```

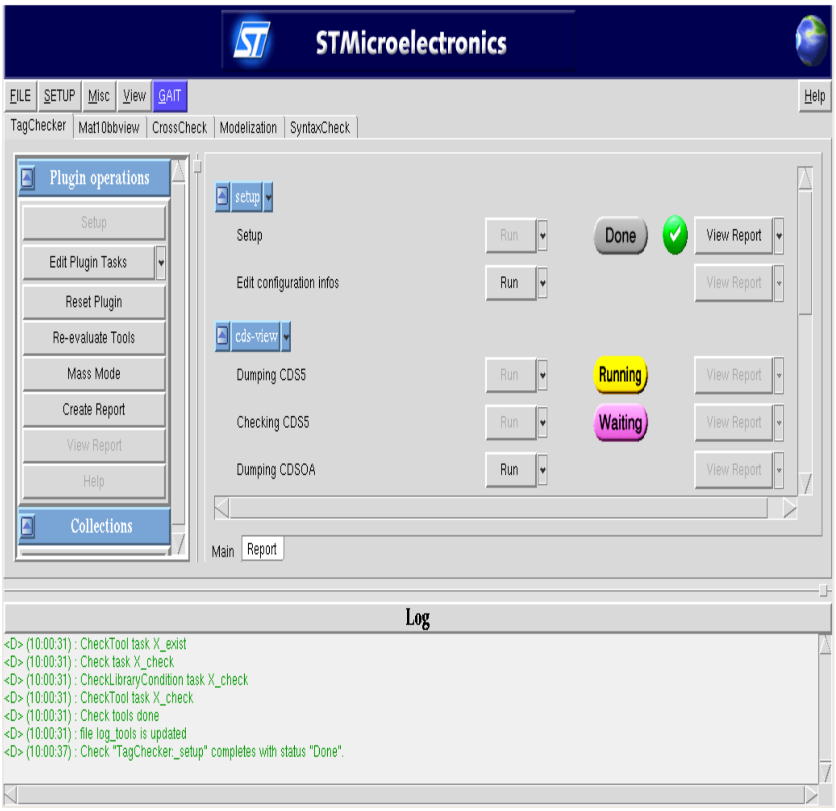
Figure 4.4: GUI of MAT10bbview PLUG-IN
[14]

Discussion: The figure contains the tasks based on the bbview file. Data indexation and coherence task is checked with some errors. If anyone wants to see those errors then one has to click on the "View Report" button.

4.2.4 Tagchecker

This plug-in is used to cover the Tags defined for the Library. It checks the Tags given to every cells in the Library and also checks the Fields inside each Tags.

Whenever any product (Library) is going to be submitted to any customer, the Tags (Fields) of the library must be checked. Tags are like Name of the library, Version of the library, Type of the library, Date, Time etc. If any of the Tags is incorrect then library would not be submitted to the customer.



```
<D> (10:00:31) : CheckTool task X_exist
<D> (10:00:31) : Check task X_check
<D> (10:00:31) : CheckLibraryCondition task X_check
<D> (10:00:31) : CheckTool task X_check
<D> (10:00:31) : Check tools done
<D> (10:00:31) : file log_tools is updated
<D> (10:00:37) : Check "TagChecker:_setup" completes with status "Done".
```

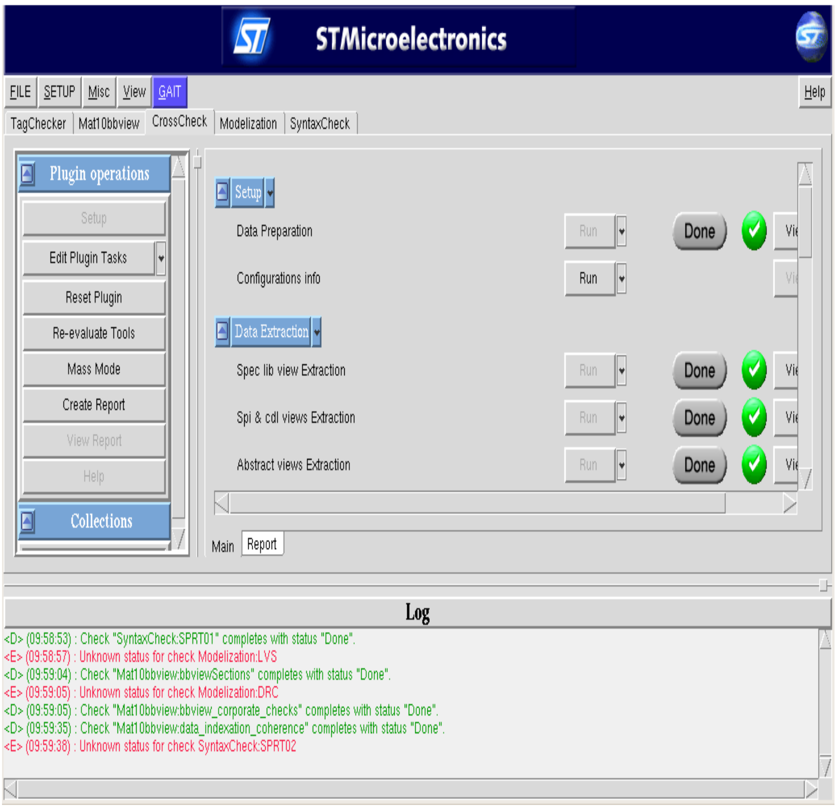
Figure 4.5: GUI of Tagchecker PLUG-IN
[14]

Discussion: Figure shows the two another status of the tasks. Yellow part shows the ”Running” status of the task. It means the task is being run. Purple part shows ”Waiting” status of the task. It means this task will run when the above task has completed.

4.2.5 Crosscheck

This plug-in is designed to ensure views consistency.Similar view consistency ensures that same type of views are consistent among them (i.e. all liberty files,all lef files, all .v files etc). Cross view consistency ensures that all different views or Cross-Views are consistent with each other (i.e. .lef vs .lib, .lef vs .v etc).

In this PLUG-IN, The basic logic which is going to be used is : if X = Y and Y = Z then X = Z. Means if .lef = .lib and .lib = .v then .lef = .v . Some basic TCL(Tool Command Language)[12] commands are used to check like this.


STMicroelectronics
FILE SETUP Misc View GAIT Help
TagChecker Mat10bbview CrossCheck Modelization SyntaxCheck
Plugin operations
Setup
Edit Plugin Tasks
Reset Plugin
Re-evaluate Tools
Mass Mode
Create Report
View Report
Help
Collections
Setup
Data Preparation Run Done
Configurations info Run
Data Extraction
Spec lib view Extraction Run Done
Spi & cdl views Extraction Run Done
Abstract views Extraction Run Done
Main Report
Log

```
<D> (09:58:53) : Check "SyntaxCheck:SPRT01" completes with status "Done".
<E> (09:58:57) : Unknown status for check Modelization:LVS
<D> (09:59:04) : Check "Mat10bbview:bbviewSections" completes with status "Done".
<E> (09:59:05) : Unknown status for check Modelization:DRC
<D> (09:59:05) : Check "Mat10bbview:bbview_corporate_checks" completes with status "Done".
<D> (09:59:35) : Check "Mat10bbview:data_indexation_coherence" completes with status "Done".
<E> (09:59:38) : Unknown status for check SyntaxCheck:SPRT02
```

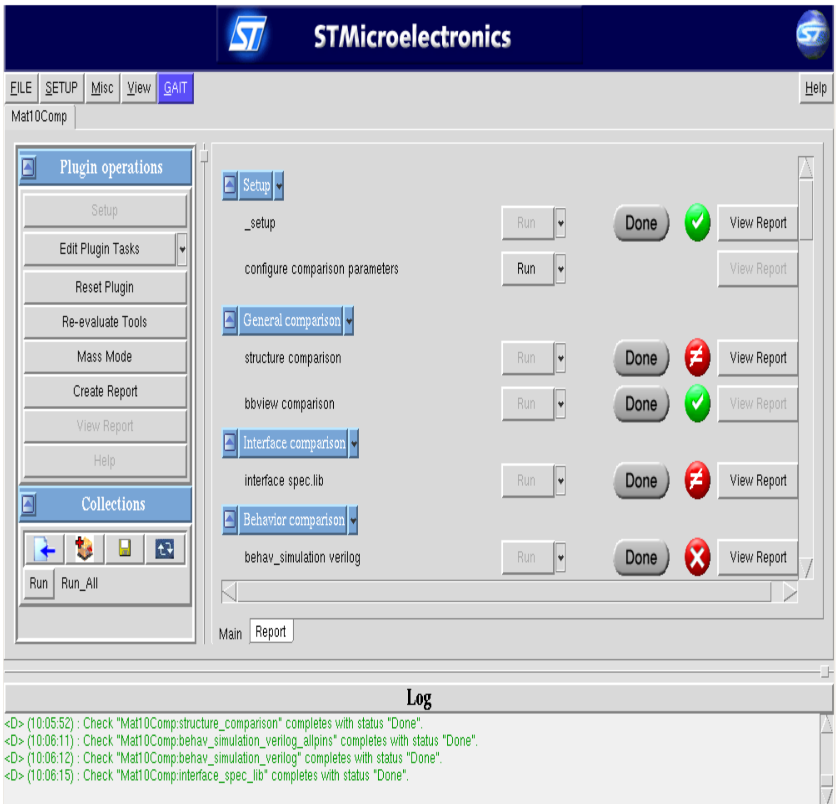
Figure 4.6: GUI of Crosscheck PLUG-IN
[14]

Discussion: Figure shows that all tasks are checked successfully. That means all the views are compliant with each other. Side bar shows the different options which are used after loading the library and PLUG-INS.

4.2.6 Mat10comp

This plug-in is designed to compare two versions of the same library. So we can elaborate the new version is how much effective then the older version.

In this PLUG-IN, the whole process is done by one tool named as LIBCOMP[14]. This tool has some command which themselves compares the different files between two versions of the same library whenever they will be called. After executed any command, it generates particular report for the same.



```
<D> (10:05:52) : Check "Mat10Comp:structure_comparison" completes with status "Done".
<D> (10:06:11) : Check "Mat10Comp:behav_simulation_verilog_allpins" completes with status "Done".
<D> (10:06:12) : Check "Mat10Comp:behav_simulation_verilog" completes with status "Done".
<D> (10:06:15) : Check "Mat10Comp:interface_spec_lib" completes with status "Done".
```

Figure 4.7: GUI of MAT10comp PLUG-IN
[14]

Discussion: Here, the red part shows that task has run successfully and It has compared those files between the two versions of the same library which has been loaded. once the ”structure comparison” task has run successfully then other tasks are seen on the screen.

4.3 Summary

In this chapter, it is explained that how ipscreen and plug-ins are useful in the checking of design statically. This chapter shows that how PLUG-INS do the hectic job in a very easy method in a very short time. Syntax, Model and structure of any library are easily checked by the PLUG-INS. we can design more checks by using TCL (Tool Command Language).

Chapter 5

Implemantation and Intermediate Results

5.1 Automatic Setup Creation

Earlier following files were required to create manually-

- Command file in which under test library's path, name and some other information required to fill manually which can maximise human mistakes.

- List of tool required by the plugins with their versions.

- List of plugins user wants to run on their libraries.

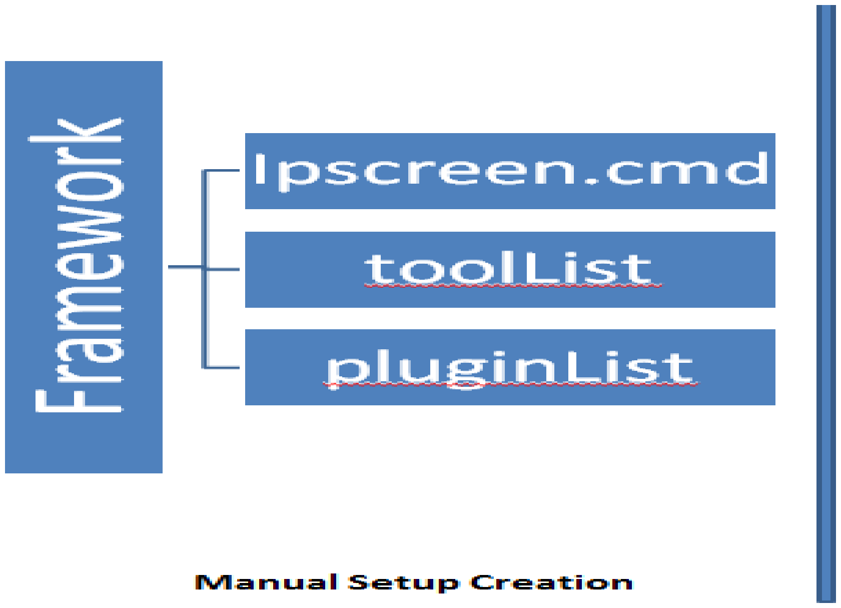


Figure 5.1: Manual Setup Creation Process

But now in this automatic setup creation module, all files are created automatically, User just required to provied a few information. Following steps I have implemented to achieve this automation-

- I have developed a TCL script which searches the" LIBRARIES" directory in current working area, this directory has the under test library.
- After the successful search, I have grepped information of library full path.
- Each library has an index file namely vc.bbview , this file contains all the information of library including library's IP type (either Memory, StdCell, Macro or IO) as well as library's actual name.
- Since this index file is very large and it is difficult to find required information directly, I have used Regular Expressions for collecting all the information.
- With regular expressions I have created some patterns by analysing all kind of index files of libraries.

- Hence here I have library name, its technology as well as library type. All these informations are redirected to a file.

Tool List creation -

- Now to create ToolList i have used al RefSpec[10] Tool which is ST specific tool.

- This tool uses a specification file or technology specification file as an input, Since this file is technology dependent hence all libraries irrespective of library type can use this file. But they should have same technology.

- Using this file refspec creates a tool List.

- After creation of this toolList file a script which is in Shell runs and removes duplicated lines, extra spaces and commented lines.

Plugin List creation -

- Earlier user needed to provide a file which contains Plugin name and thier respective path.

- Now user just need to give a plugin name on command line , and inside the script all plugins are extracted and a plugin list file is created.

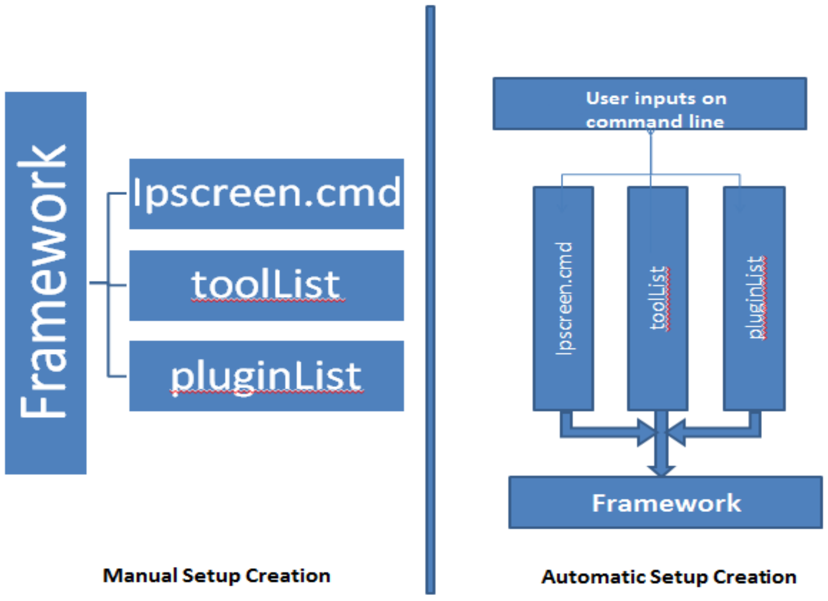


Figure 5.2: Manual versus Automatic setup creation

5.1.1 Results of Automated Setup Creation

As a result user just needed to provide library name (which is under-test), Auxiliary library plugin name in following format.

ipsSetup -libraries ”list of under test libraries seperated by spaces” -plugins ”list of plugins” -auxLib ”list of auxiliary libraries” -runbatchMode ”yes or no”

5.2 Pre Tool Checking Module

Since Plugins uses tools for validation, so we have to provide the same in the tool list. After loading the Plugin and tool list , it gives an error that tool is missing (If that tool is not available), So user have to update setup again and rerun the plugin again. This is a very irritating work to update the tool list and run the checks again. I have developed a module of pre tool checking before ipscreen runs.

- I developed the pre tool Checking module that Checks the existence of every tool that is pesent in tool list, If not present then it shows error prior to the Plugin Load.

- I have used shell scripting a parsed through each lines.
- The checks which are going to be run by user are identified and their respective tools are extracted.
- Now all required tools from tool list are picked and checked existence in the central area or the given paths.
- Since I could do the checking of whole toolList file but in order to save time the developed script only checks the required tools.

5.3 Access rights checking of whole library package

This module is to check the access rights of whole library package in order to make sure that library have defined access rights.

- To implement this module I have used Tcl as well as Shell scripting.
- I made a script which checks directory with 755 permission and files with 644 permission.
- Firstly I have listed a list using shell utility and redirected them all in a file.
- And checked the directory with "drwxr-xr-x" and files with "-rw-r–r–"

5.3.1 Results

I have added this check in syntaxcheck plugin. Now user do not need to check this manually or stand alone script. He/She just click on the button of access rights check and their library will be checked fully and results will be displayed.

5.4 Mat10Comp plugin new Archicture

Following changes are done by me in existing plugin.

- Since in existing plugin many intermediatery files are created so it is taking more runtime and space.
- Another problem is task were running on same .lib and .db files (ST specific files) on each run.
- For the first problem I used SED and AWK commands which do not create extra files but they keep data in buffer, hence it saves reading and writing time from files.
- For the second problem I have restricted to run tasks on same .lib and .db once by default.
- I have also corrected the formatting of its output through shell and tcl scripts.

5.4.1 Intermediate Results

Some of the tasks are taken from both new and old plugin that is Mat10comp.v.2.0 and Mat10comp.v.1.1 respectively. In both versions tasks are running in parellel.

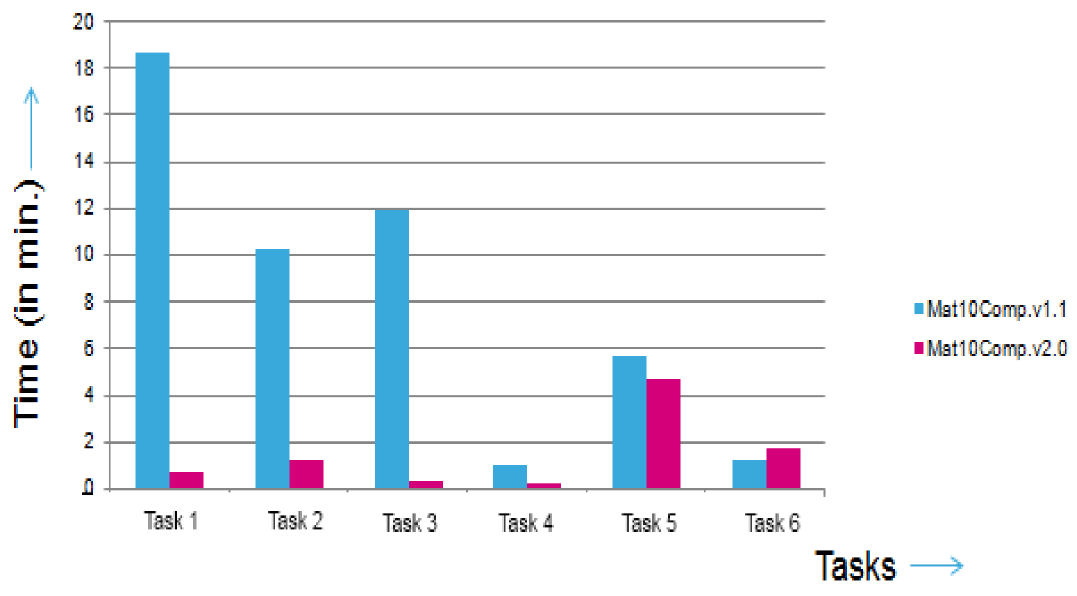


Figure 5.3: Situational Analysis

5.5 Manufecturing Design Check

Layout is a view of a cell in a Library. And this should be aligned according to the layout design rules.

- Earlier every team has to check this manually that their library has aligned with layout Design rule. So this was very tedious task.
- I developed this check in Cadence SKILL language .[12].
- For Doing this first thing needed was to Parse Rules File.[13]
- So I Developed a parser that picks appropiate values from rule files and makes a database.
- And using this file I check that under test libraries follows that rule or not.

- Layout is a view of a cell in a Library. And this should be aligned according to the layout design rules.
- Since every cell has two types of Power pins: Power and Ground Pins.
- I need to check that two adjacent pins with same polarity have the minimum distance as defined in the Layout rules.

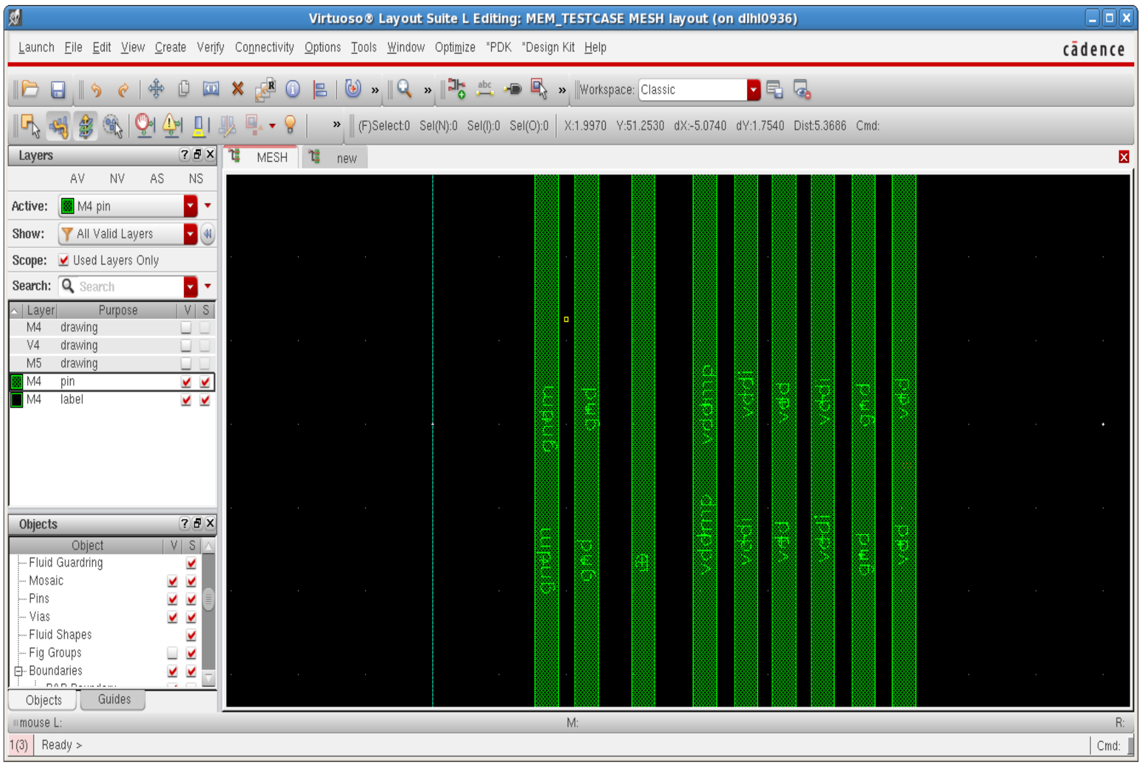


Figure 5.4: Layout View of a Cell

5.6 Summary

In this chapter, it is explained that how i implemented features and their intermediate results through chart.

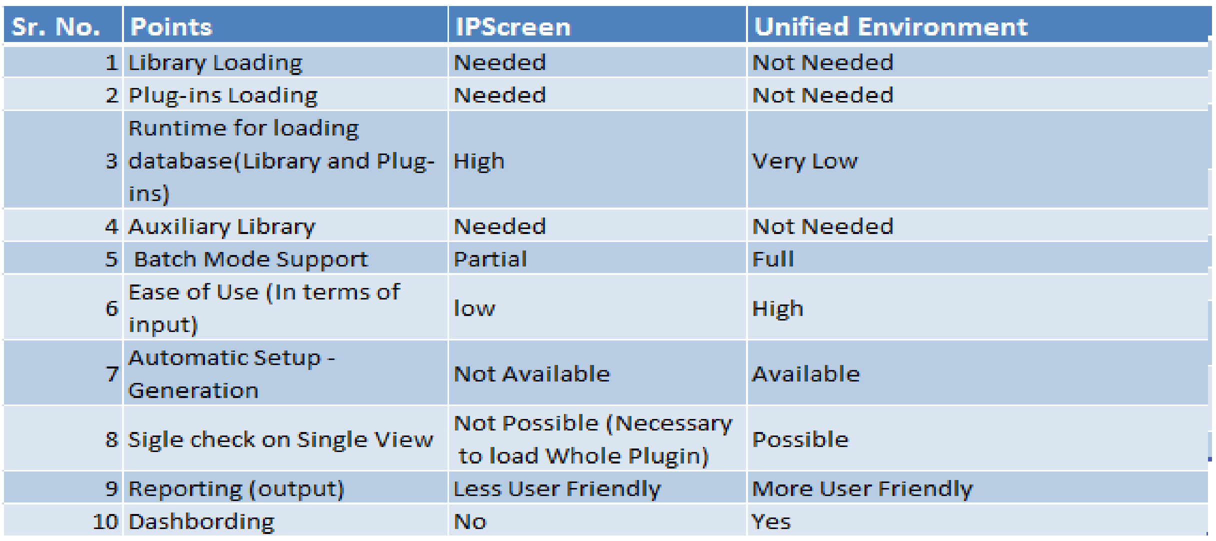
Chapter 6

New Approach Unified Environment

Since In IPScreen we have to load Libraries load PLUG-INS . This is a time as well as space consuming process to validate Libraries. To overcome some of the limitations this Approch came into the picture.

6.1 Some Points related to this New Environment

- Since it do not require to Load library and Plugins hence it takes less memory on disk and comparatively less runtime.
- Its all Setup file can be created Automatically,
- It do not Provides GUI , in first phase it will work on command line interface, GUI can be provided in its next release.
- It supports Full batch mode.
- It provides pre-checking of tools.
- Plus point of this Environment is if user want to run specific checks on specific view then he can do that without running whole plugin.



Sr. No.	Points	IPScreen	Unified Environment
1	Library Loading	Needed	Not Needed
2	Plug-ins Loading	Needed	Not Needed
3	Runtime for loading database(Library and Plug-ins)	High	Very Low
4	Auxiliary Library	Needed	Not Needed
5	Batch Mode Support	Partial	Full
6	Ease of Use (In terms of input)	low	High
7	Automatic Setup - Generation	Not Available	Available
8	Sigle check on Single View	Not Possible (Necessary to load Whole Plugin)	Possible
9	Reporting (output)	Less User Friendly	More User Friendly
10	Dashbording	No	Yes

Figure 6.1: Executive Summary old and new approach

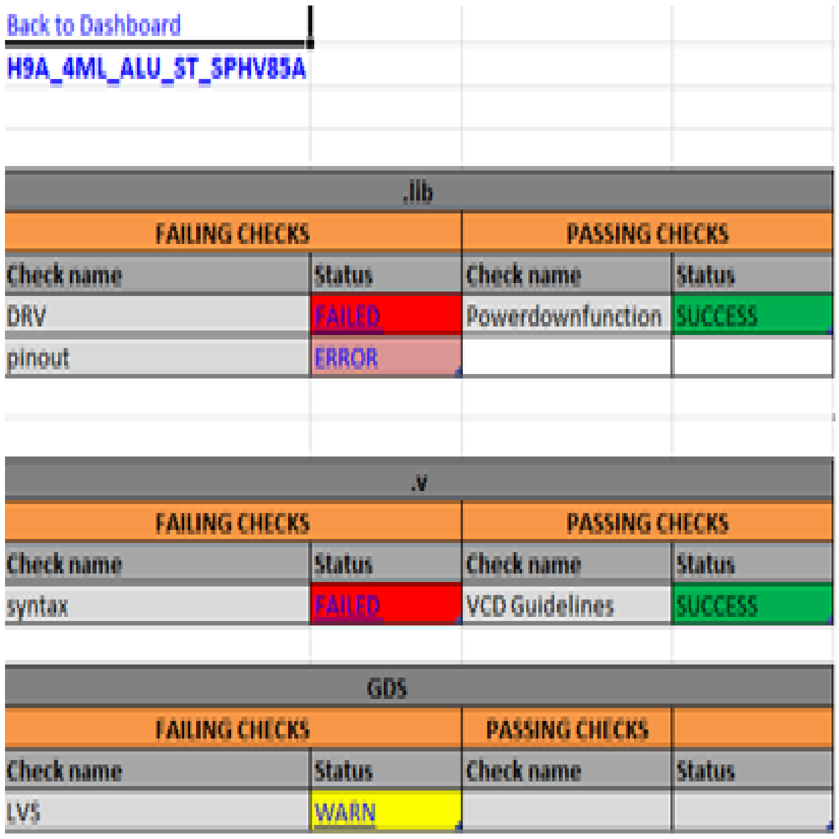
- One of the best feature of this tool is regression environment. It Provides following features-
  - It provides Run option to the failing checks through command line interface.
  - This facility is totally full batch mode supported.
  - User can run all failing checks at once or one by one.
  - IPScreen[3] provides this facility but that is time consuming and not fully batch mode supported.

6.2 My Contribution in this unified environment

- I have actively participated in unified environment architectural discussion.
- I am constantly working on its output part for making it user friendly.

6.2.1 Implementation of Dashboard Creation

When user validates their IPs , different checks performed and as the end results they get Reports. Now excel format reports are always better because user can easily write their comments on this format. But the main concern is unlike html reports excel format is not formatted well, So the very challenging task is to make it user friendly. I have used following steps to make report well formatted in excel.



Back to Dashboard

H9A_4ML_ALU_ST_SPHV85A

.lib			
FAILING CHECKS		PASSING CHECKS	
Check name	Status	Check name	Status
DRV	FAILED	Powerdownfunction	SUCCESS
pinout	ERROR		

.v			
FAILING CHECKS		PASSING CHECKS	
Check name	Status	Check name	Status
syntax	FAILED	VCD Guidelines	SUCCESS

GDS			
FAILING CHECKS		PASSING CHECKS	
Check name	Status	Check name	Status
LVS	WARN		

Figure 6.2: GUI of Crosscheck PLUG-IN

– When checks run on the views they creates their seperate areas. And makes some logs(log of scripts execution) and status files (final status) of the check run.

– To bring all the results of different checks together it was needed to parse all the files from their respective directories of checks.

- I have designed a parser in Shell and Tcl language which parses through the areas and makes a common unix report. This report can also be used by the user.
- I have developed a module in Perl[8] language to create a well formatted excel report.
- This report is such intelligent that user can find required information easily.

6.3 Summary

In this chapter, it is shown that how new environment is different and better then existing Framework that is IPScreen . And how its reporting is different from IPScreen.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In the field of validation, automation through PLUG-INS are milestones and they are now playing an importent role to validate libraries. All PLUG-INS are to be used to validate any library. By using these PLUG-INS , whole validation is done in 1 day instead of 5-6 days . So, we get the time efficiency of 86 percent.Manual work is being automated in order to save time as well as time. A new approach (Unified Environment) is going to be built with backward compatiblity , the reason behind to make validation very fast as well as view base, using this approach the disk space used would be very less as compared to previous solution..

7.2 Future Work

Since the Technology is going very fast hence I need to explore more in the field of scripting and Enhance the functionally of new unified environment and making this as much intelligent as it can minimize manual work which in turn reduces cost and increases productivity .

References

[1] ST internal documents and Training manuals on LIBRARY BASICS

[2] ST internal documents and Training manuals on LIBRARY VIEWS

[3] ST internal documents and Training manuals on CAD INFRASTRUCTURE, DESIGN KIT and Tools.

[4] Practical Programming in Tcl and Tk (4th Edition) by Brent B. Welch

[5] Tcl/Tk Cookbook by L. Sastry

[6] Wikipedia `http://en.wikipedia.org/wiki/C_shell`

[7] Tcl guide , `http://wiki.tcl.tk/299`

[8] Tutoials point ,`http://www.tutorialspoint.com/perl/perl_introduction.htm`

[9] John K. Ousterhout "Scripting: Higher Level Programming for the 21st Century" Published in IEEE Computer magazine.

[10] ST internal tool RefSpec and User Manual.

[11] Bernard Laurent and Thierry Karger "A System to Validate and Certify Soft and Hard IP" Proceedings of the Design,Automation and Test in Europe Conference and Exhibition (DATE03).

[12] Cadence SKILL guide https://secure1.cadence.com/loginapp/CSSOLogin/login.action?TAM_OP=login&ERROR_TEXT=Successful%20completion&URL=%2F&HOSTNAME=support.cadence.com&PROTOCOL=http

[13] ST internal Documents on DesignKits .

[14] ST internal documents and Training manuals on PLUG-INS.

[15] http://www.tcl.tk/doc/scripting.html