Automation in Validation of Library Views

Submitted By

Deepali Bapodara

15MCEI02

DEPARTMENT OF COMPUTER ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2017

Automation in Validation of Library Views

Major Project

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering(INS)

Submitted By

Deepali Bapodara

(15MCEI02)

Guided By

Prof. Parita Oza

Nirma University, Ahmedabad.

Mrs. Jyoti Kumar

ST Microelectronics, Greater Noida

DEPARTMENT OF COMPUTER ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2017

Certificate

This is to certify that the major project entitled **"Automation in Validation of Library Views"** submitted by **Deepali Bapodara (Roll No: 15MCEI02)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering(INS) 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, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. Parita Oza
Guide & Assistant Professor,
IT Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Sharada Valiveti
Associate Professor,
Coordinator M.Tech - CSE(INS)
Institute of Technology,
Nirma University, Ahmedabad

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

Dr. Alka Mahajan
Director,
Institute of Technology,
Nirma University, Ahmedabad

Statement of Originality

I, **Deepali Bapodara**, Roll. No. **15MCEI02**, give undertaking that the Major Project entitled "**Automation in Validation of Library Views**" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science and Engineering(INS)** 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. Parita Oza

(Signature of Guide)

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Mrs. Jyoti Kumar**, Project Manager, ST Microelectronics, Greater Noida and **Mr. Krunal Patanwadia**, Software Engineer, 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.

It gives me immense pleasure in expressing thanks and profound gratitude to **Prof. Parita Oza**, Assistant Professor, Information Technology Department, Institute of Technology, Nirma University, Ahmedabad for her valuable guidance and continual encouragement throughout this work. The appreciation and continual support she has imparted has been a great motivation to me in reaching a higher goal. Her guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank **Dr. Sanjay Garg**, Hon'ble Head of Computer 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 **Dr Alka Mahajan**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation she has extended throughout course of this work.

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

- Deepali Bapodara

15MCEI02

Abstract

Innovations in the Technology industry have made our daily life easier than ever before.Devices like Cellphones,Desktops,Wi-fi,Touch Sensors and many other components of digital world are getting efficient and smarter each day.All these devices contains a "Integrated Circuit(chip)" which is designed to perform some specific function.Chips play an important role in evolution of Technology.Intellectual Property (IP) in Integrated Circuit(IC) is portable design which can be reused.It is nothing but the block of functionality which is licensed to different vendors when used in different chip designs.

IP contains different views like gds,cdl,lib,db,lef,def etc.In order to make a device's functioning correct,one needs to validate IP which in turn leads to validation of Library Views.IP validation is an indispensable aspect that helps to detect critical issues prior to use in the actual System on Chip (SoC) Design.This demands the need for an IP validation solution that accurately verifies the correctness and sufficiency of the IP design.IP Validation can be done by Two approaches: Manual Validation and Automated Validation.Manual Validation is executed by person without any help of Tool/Software.Due to some limitations like less accurate result,less reliability, more Time consuming we require Automated Validation which is executed with the help of Software/Tool.This thesis portrays ta scalable IP Validation Solution that provides an intensive checking of the given IP views while concealing all the intricacies from the Designer and thus increasing the ease of use.The proposed solution is replacement of the existing IP validation Tool.(IPScreen)

Abbreviations

IC	Integrated Cicuit
IP	Intellectual Property
CDL	Circuit Design Language
GDS	Graphical Design System
LVS	Layout v/s Schematic
DRC	Design Rule Check
LEF	Library Exchange Format
TCL	Tool Command Language
VLSI	Very Large Scale Integration
GUI	Graphical User Interface
SLIB	Symbolic Library
EDA	Electronic Design Tool
CAD	Computer Aided Design
SoC	System On Chip
DK	Design Kit
FE	Front End
BE	Back End

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

With lower nanometer technology, the volume of an IP data is increasing exponentially to represent an electronic circuit on a piece of silicon. With such a huge amount of data an IP must be fully qualified before its actual integration into a SoC. This demands the need for a validation solution, where the objective is to ensure IP Quality. The major factors contributing to an IPs success are the quality of IP CAD views and time-to-market. According to a survey conducted by Arteris (multinational firm developing on-chip interconnect fabric technology) regarding top design concerns, quality problems contributed to approximately 20%. Thus, an improved CAD quality will help to reduce this percentage to a great extent. The later we detect the issues, the more expensive they become in terms of both time and money. This motivates in development of well designed validation methodology that helps the designers to deliver a highly reliable product and also reducing the overall cycle time. The conventional approach of validating views using the RTL2GDS flow, had a huge cycle time.[7] Using IPVS, one can achieve simultaneous generation and validation. Since a library can be validated View wise, one need not wait for the complete packaging. Any bug found in the generated view can be fixed quite early before the packaging.

1.2 Thesis Objective

A library is a consolidated data for use in designing a system on chip. The library is comprised of various views which are useful for designing a chip.So in order to make

chip's functioning correct we must have to ensure that our Libraries are properly validated.

Manual Validation of Library requires lot of time as well it is not accurate; it may contain human errors.So Validation of Library requires some Automation which can display the accurate result in less time.

Here in ST Microelectronics were using a framework named IPSCREEN earlier.[3]It is a GUI in which you have to load Library and based on the specified checks it will validate the Library.Checks are nothing but TCL/CSH scripts.Checks are collectively called as Plugin. In IPSCREEN ,User has to wait for a long time till the library gets completely parsed by the IPScreen as well as One can not parse single view or particular cell of the library. In order to overcome the limitations of IPSCREEN,the framework has now been shifted to IPVS.It is a framework in which you have to load library and based on the specified checks it will validate the library.
The main objective of this thesis has been described as follows:

- Improvement of IP validation methodology.
- Review of existing methodology (ST internal).Review was done on the following parameters :
    1. Technical(software perspective).
    2. Ease of use.
    3. Configuration of Parameters.
    4. Reporting.
    5. Quality of checks.
- Smooth deployment of enhanced validation solution at customer (IP developer)end and support to customer.

1.3 Thesis Outline

The rest of the thesis is organized as follows:
Chapter 2 describes the fundamentals required for this thesis.Library and different types of Library Views are discussed.
Chapter 3 describes the Significance of Validation Phase in the Library development Flow

as well as the the check Parameters for Validation Process are discussed.

Chapter 4 describes the review of the existing approaches for the Validation process. Pros and Cons of the Validation Approaches has been mentioned as well.

Chapter 5 describes the ST's old Framework named IPSCREEN.A detailed architecture as well as Limitations of the framework has been discussed.

Chapter 6 describes the architecture as well as the significance of IPVS framework.

Chapter 7 describes the whole development of IPVS including the Prerequisites, User Interface and the Output Structure.

Chapter 8 describes the various experiments done on IPVS and the analysis of result is outlined.

Chapter 9 describes the Observations made during the development and experiments of IPVS .

Chapter 10 describes the conclusion of the thesis.Future directions in this area is also outlined in this chapter.

Chapter 2

Literature Survey

2.1 Introduction to Library Views

For designing a chip we require a Library. Library is collection of Cells. Cells are the component inside the library which performs the actual functions.This function is the Boolean and basic function which can be AND, OR, NOT etc. Cells are represented by Views.Views can be of different type like symbolic view,layout view, schematic view etc.These views are used by different tools.

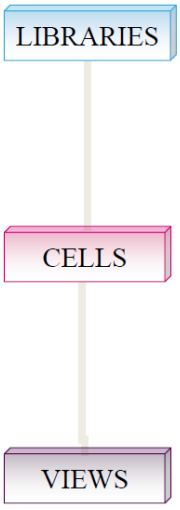


Figure 2.1: Library View

2.2 Library

A library is a consolidated data which is use in designing a System on Chip(SoC).These data can be Cell functionality, Transistor level design of an IP(Intellectual Property) or timing information of cells.

2.2.1 Classification of Libraries

Libraries are classified based on Customer requirement ,functionality and design of the device.

Some example of Libraries are listed below:

Standard Cell Library

As the name says it consists of standard cells and it is also called Core Library.They implement standard functions like Adder, Subtractor, Invert etc.

Memory Library

It contains memories like SRAM, DRAM, and ROM etc.

Analog and Mixed Signal library

It uses CORE library for implementation.Examples of Analog and Mixed Signal Library are PLL,Digital to Analog Converter, U.S.B. etc.

2.3 Views

As mentioned above,Library is collection of Cells and these cells are represented by Views.A cell is delivered as a set of view and each view is used by a different tool.

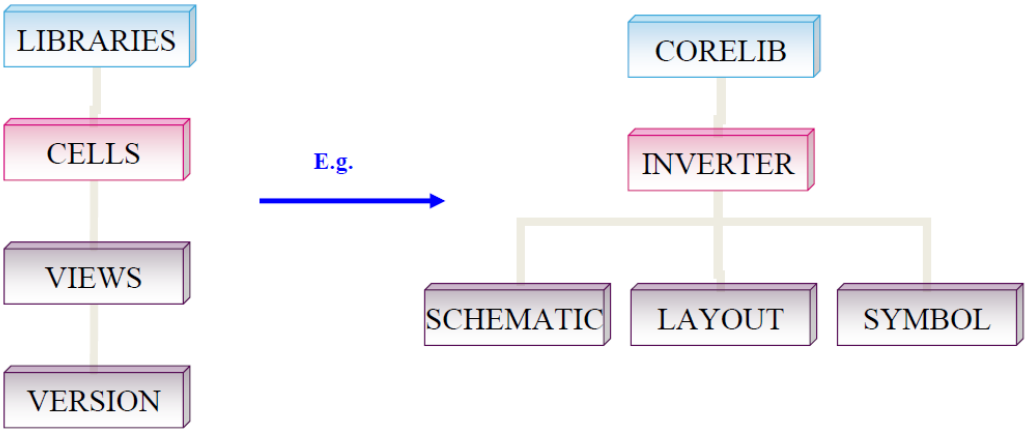


Figure 2.2: An illustration of View

2.3.1 Classification of Views

Views are classified in two major category which is Front End and Back End Views.

Front End Views : These Views are related to Physical design of the cell.

Back End Views : These Views are related to Timing/Modeling of the Cell.

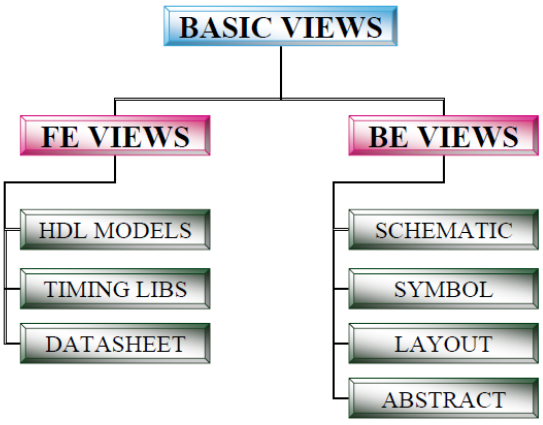


Figure 2.3: Front/Back End Views

2.3.2 Back End Views

Symbolic View

As the name indicates Symbolic View is pictorial representation of cell.It includes shape, PIN, selection box and label.[1]

- PIN- It represents Input/output of cell.
- Shape-It represents Cell's function.
- Selection box –It represents cell's area.
- Label –It is used for documentation of design.

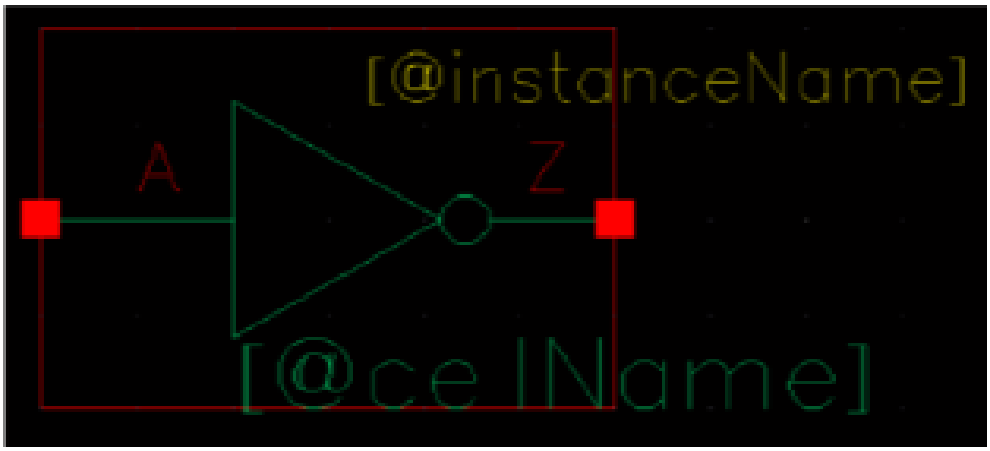


Figure 2.4: A sample Symbolic View

.SLIB

Every View has its own textual representation file. For Symbolic view we have SLIB which stands for Symbolic Library.

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 2.5: A sample .slib file

Schematic View

It is simplified representation of electronic circuit.It shows simplified standard symbols,power and signal connection between devices.[1]

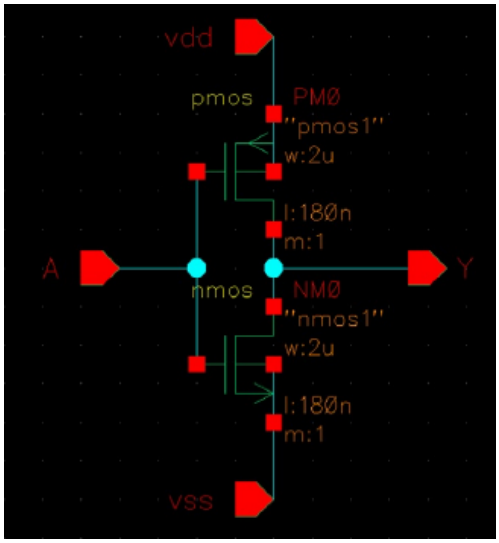


Figure 2.6: A sample Schematic View

.CDL(Circuit Description Language

It is textual representation of Schematic View.

It contains:

Power and Ground Pin Information

Connectivity at Transistor Level

Device parameters like Device name,length,area etc.

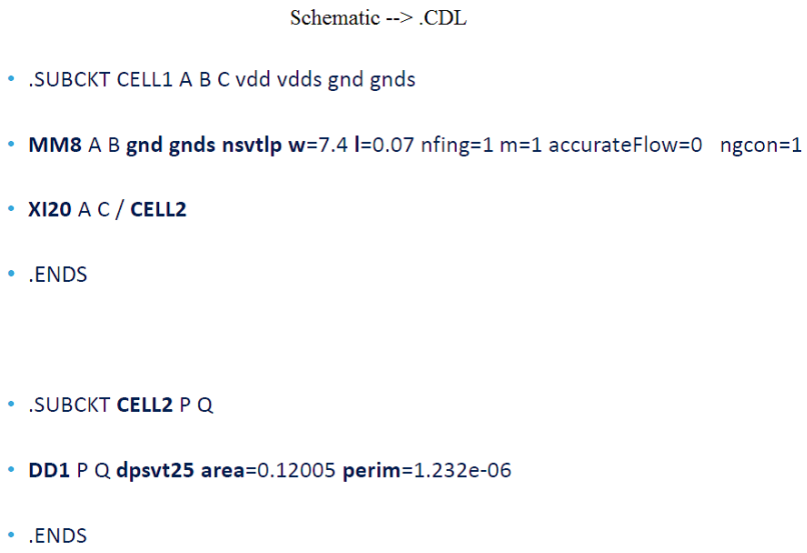


Figure 2.7: A sample .cdl file

Layout View

It is physical representation of cell's electronic circuit which goes on silicon layer.[1]

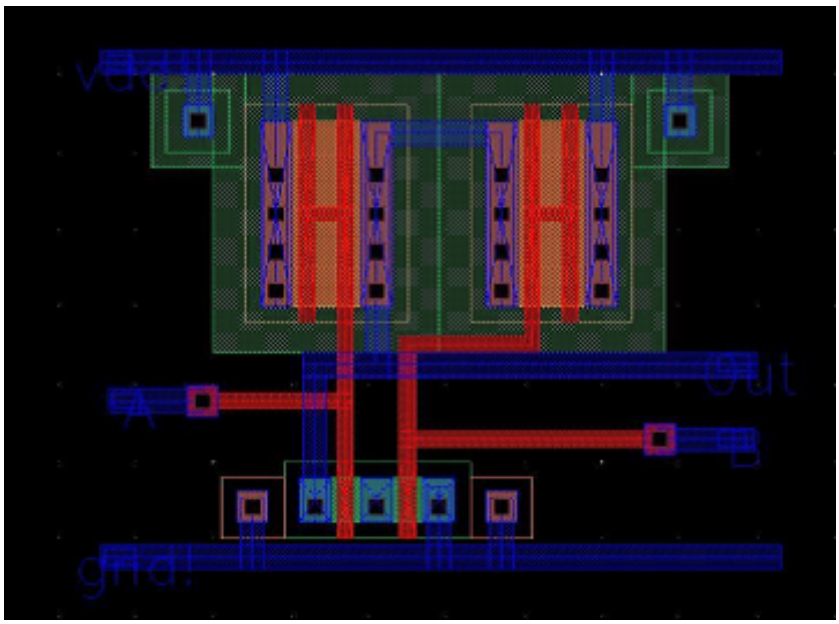


Figure 2.8: A sample Layout View

.GDS(Graphical Design System)

It contains same data as that of Layout View but it is in binary format.

Abstract View

For some tool we require small amount of information like Power and Ground Pin Information ,Obstruction which is the area where routing is not allowed.So for that Abstarct View can be used.It is subset of Layout View as it contains small amount of information compared to Layout View.[1]

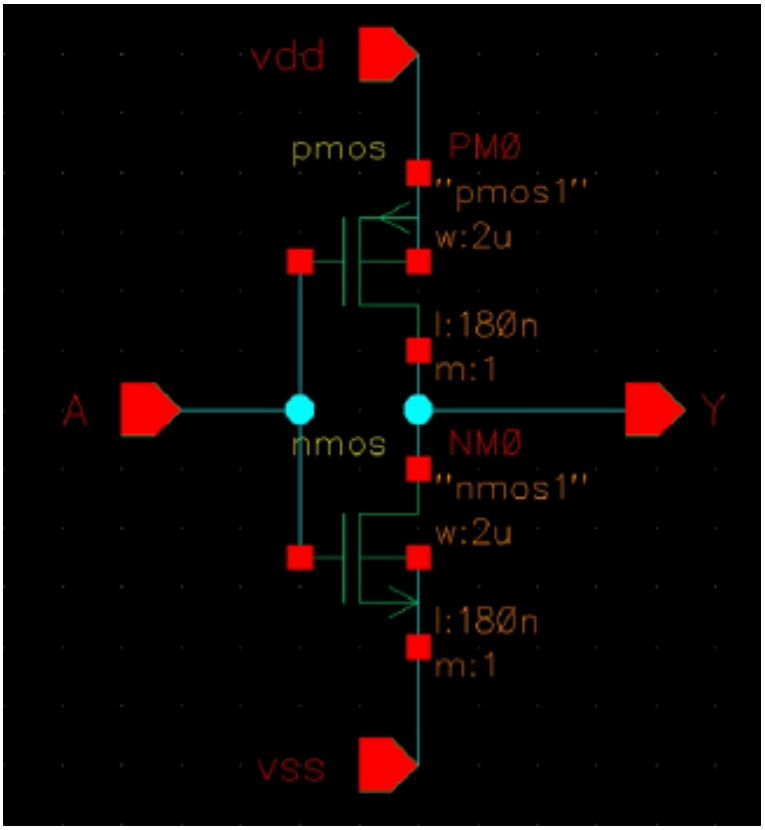


Figure 2.9: A sample Abstarct View

.LEF (Library Exchange Format)

It is ASCII representation of Abstract View.It contains following Infromation:

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 2.10: A sample .lef file

Chapter 3

Library Development Flow

3.1 Introduction

The framework configuration requires increasingly predefined libraries/IPs because of the expanding unpredictability of the framework.Foundries give framework originators an outline stage containing all fundamental libraries/IPs keeping in mind the end goal to bolster their framework design.The library incorporates a gathering of cells. Its fundamental objective is to offer an extensive variety of data about the cells to framework creator for coordinating them into his system.In the following segment a detailed Flow has been examined.[8]

3.2 ST Specific stages

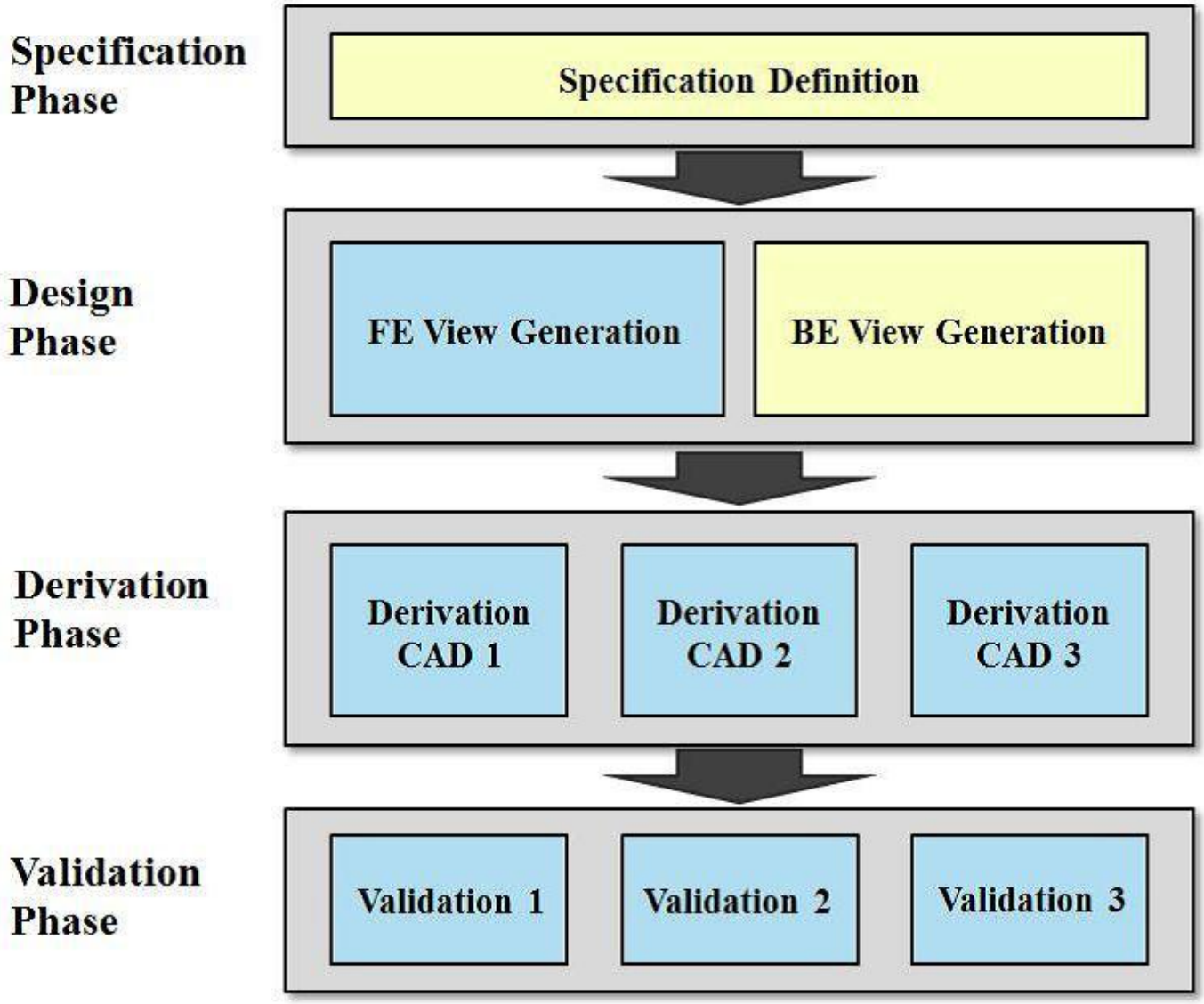


Figure 3.1: ST specific Library Development Flow

1. **Specification Phase :**

   - The specification stage is vital for development of library since it goes for gathering all required data/information.Particularly, the library development must cover an extensive variety of data. Also, the data must be gathered from a few data proprietors. Hence, the specification stage is a collaborative work. For instance, we require a set of library views and CAD tools data for library development.

- The library architect characterizes the substance of the library package for every library class. The DK engineer decides important CAD devices with a specific end goal to build up an outline unit for supporting them. Accordingly, the data related to library views and CAD tools can be gotten from them individually. At last, the assembled data is characterized in the specification.

2. Design Phase :

   - The Design stage is all about designing the cell's of library This stage produces every single basic view for FE and BE of the framework. In particular, as per the cell's specification, the cell is composed and after that its physical view is made with the use of some Editor. In any case, much exertion and time should be given to this stage as it decides the accuracy/performance/execution of the cells.

3. Derivation Phase :

   - The Derivation stage permits creating different CAD views from the basic Views keeping in mind the end goal to totally bolster client's CAD flows as some of them require their own semantics.For example, Cadence tools depend on OpenAccess database while Synopsys tools depend on MilkyWay. Thus, we should make these two physical databases for supporting both CAD usage flow. Thus, all required CAD views should be made by the Derivation stage to fulfill all client's flows.

4. Validation Phase :

   - The nature of the library and IP is straightforwardly identified with their reuse and joining as well as the productivity/efficiency of SoC. So they should be checked before their conveyance to clients.

     (a) Significance of Validation Phase:

         Errors are effectively made amid library development process.Possible errors are: design rule violation, inaccuracy,incompleteness, inconsistency, functional errors.

         With a specific end goal to make accurate cell library, for example, to make cell's functionality correct , its precise planning execution and its

layout having no design rule infringement, this stage is extraordinarily required in Library Development Flow.

(b) On the off chance that we don't deal with IP Validation, this outcomes in:

i. Costly Customer Returns

ii. Expensive Product Rework

iii. Unexpected Delays in Product Release

In this way, this demands the requirement for an IP validation Solution that precisely checks the rightness and adequacy of the IP design.

(c) This aides in,

i. Guaranteeing Customer's need are met

ii. Maintains a strategic distance from Expenses

iii. Maintains a strategic distance from the deferrals

3.3 Check Parameters for Validation Process

The Validation flow guarantees the client that the IP under test is accurately approved. The parameters which are checked amid Validation process are depicted beneath:

1. Completeness of Library:

   - The library views ought to be made by the set of library views and PVT corners given by the specification.
   - Keeping in mind the end goal to check Completeness of Library, Presence of specific view in the package is checked. In addition, the indexation of the view is being checked in the file named vc.bbview or not. If this file is not populated effectively like the view is not recorded properly then that specific view can't be taken by the EDA tools in RTL to GDS stream.
   - So, the completeness of the library can be accomplished by just checking if all required library view exist on the given view path.

2. Correctness of the library view :

   - The correctness of Library View checks the library view concerning its characteristics given by the specification.

- It is done by checking whether the view is having vital characteristics or not which are required in RTL to GDS flow by the EDA device. After checking the existence, their values are being cross confirmed to guarantee that they are modeled as per the Design Rules or not.
- It is likewise being guaranteed that the tags/labels in the layout are agreeable with the convention or not.

3. Compatibility between the library view and CAD tool

   - The library package must give framework planners a total list of library views to bolster their flow. When utilizing library files, they should not have compatibility issues with CAD devices. It implies that their syntax/structure must be correct to be intelligible by the tools.
   - In order to check Compatibility, Syntax checking is done where the view is being perused by the individual EDA tools. If in case there is an occurrence of any mistake in the view the tool won't have the capacity to process that specific view in the RTL to GDS flow.
   - Thus, the completeness of the library can be accomplished by perusing every library files with CAD tool given in the specification.

4. Consistency between Library Views

   - It is ensured that information is consistent between various views that 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.

Chapter 4

Types of Validation

4.1 Literature Review

1. Manual Validation : - It is executed by individual with no assistance of hardware/programming.

    - Advantage:
        i. No external Tool Required
        ii. No Maintanance cost as there are no tools
        iii. No Dependency
    - Disadvantage:
        i. Less Accurate
        ii. Less Reliable
        iii. Time Consuming

2. Automated Approach : - It is executed with the assistance of Software/Tool.

    (a) Standalone Approach - It is validated autonomously without intergrating with SoC.

        - Advantage:
            i. Process duration is less as Validation doesn't require intergration with SoC.
            ii. Usage of H/W is negligible

- Disadvantage:
  i. Less Reliable
  ii. Library may behave different after integrating with SoC; may be it doesn't function correctly so validation may gets fizzled.
  iii. Restricted Coverage of Checks

(b) Integration Approach - It is performed by taking the IPs under test in a dummy SoC design.

- Integration Validation is the procedure to guarantee that IPs that are piece of a Design Platform permits the design of System on Chip.
- Integration Validation is the ordinary approach most generally utilized for IP CAD approval. After approval, IPs are proclaimed CAD compliant to empower SOC design.
- In this process an RTL2GDS flow is performed by taking the IPs under test in a dummy SOC design. After creating this specific RTL, certain simulations are performed. This ensures the correctness and quality of IP in terms of the design flow requirement.
- Advantage :
  i. Highest Reliability
  ii. Highest Accuracy
- Disadvantage :
  i. Huge Cycle Time
  ii. Huge Utilization of H/W
  iii. Huge Manpower

Chapter 5

IPSCREEN

5.1 Introduction

IPScreen is a framework which is used to validate a Library.[3] Multiple libraries can be loaded into IPSCREEN at the same time.After successfully loading Libraries into IPSCREEN based on Plugin selection it will load different checks[3]

5.2 Working of IPSCREEN

Inputs:

- Setup: It contains a script which sets all required environment variables to execute task.
- Library: You have to provide the library one which you want to validate as well as the auxiliary (reference) libraries.
- Plugin : It is Software which is used to validate Library. It is usually stored in .plugin.list file which contains path and version of plugin.
- Tools: In order to run plugin on different library the required tools are stored in .tool.list file.Format is "/Ipscreen Tool Name/ /Site ToolName/ /Release number/"

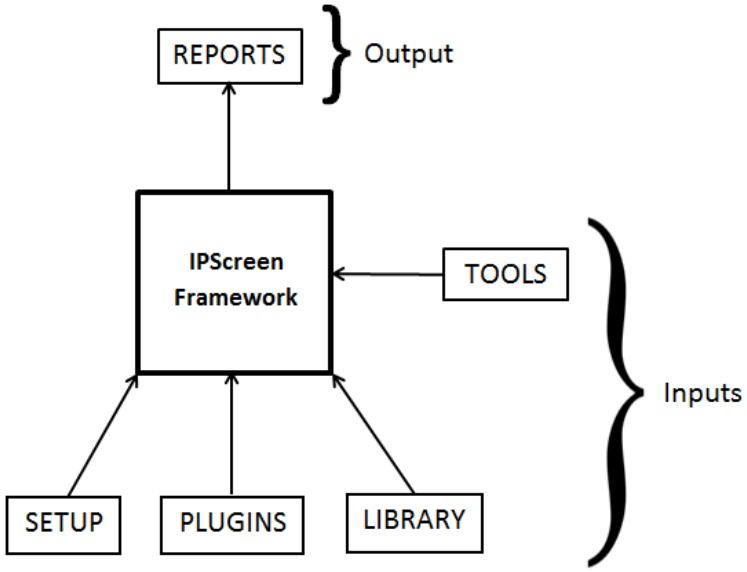


Figure 5.1: IPSCREEN Block Diagram

Output:

- Report: After running IPScreen on different libraries it will generate a report which contains error, warnings and logs. It is generated in form of text, csv, xls and html. There are basically three types of status which IPSCREEN reports:
  - Done with Green Tick: It shows that Execution was successful and Library is correctly validated.
  - Done with Brown Tick: It shows that Execution was completed with some indication of warnings.
  - Fail with Red Cross: It shows that errors are present in library and execution fails and IPScreen will not be able to generate report.

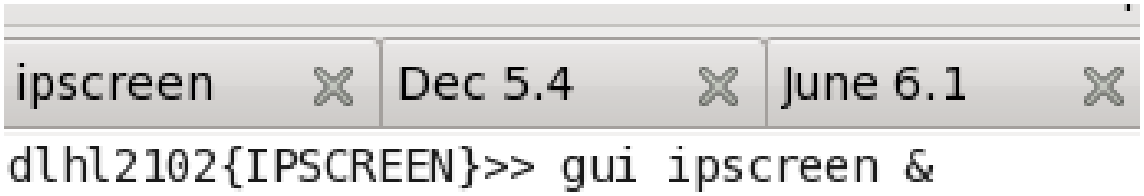


Figure 5.2: Execution Command for GUI

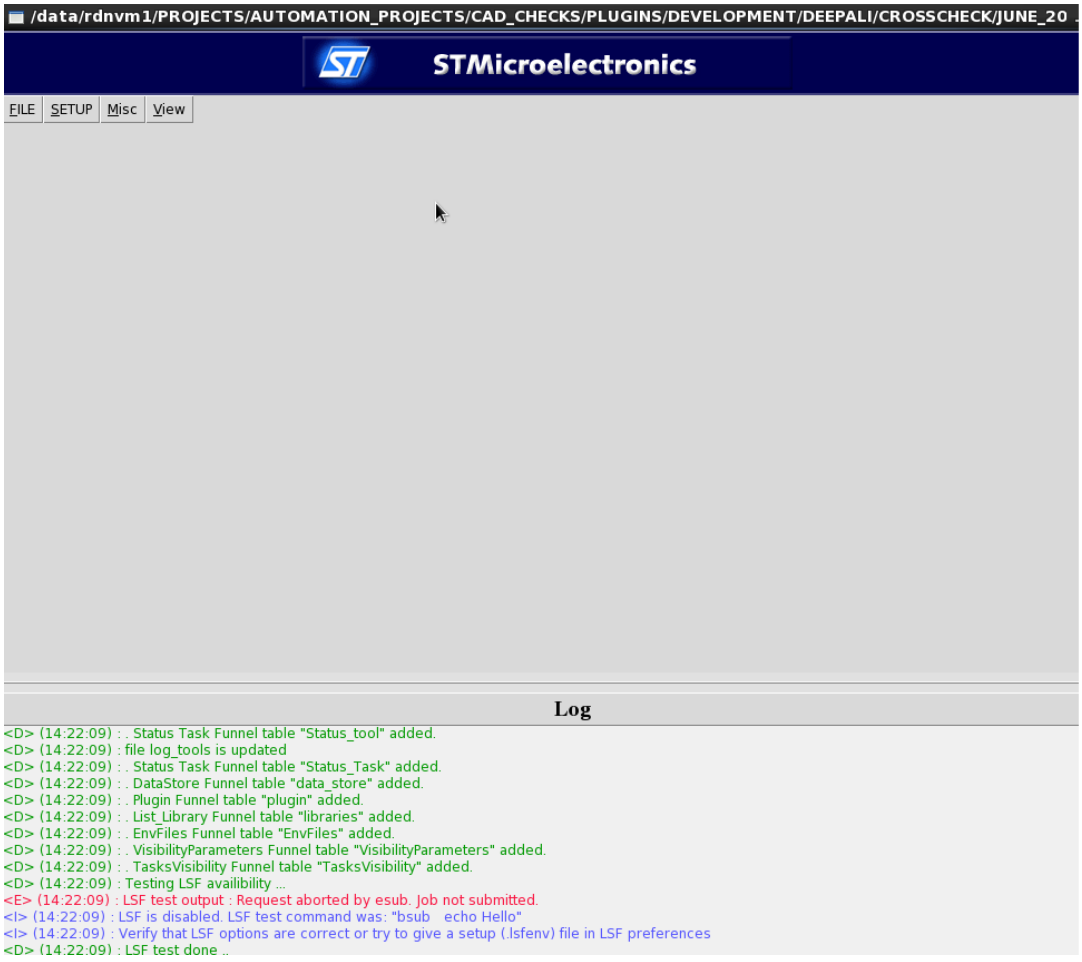


Figure 5.3: IPSCREEN GUI

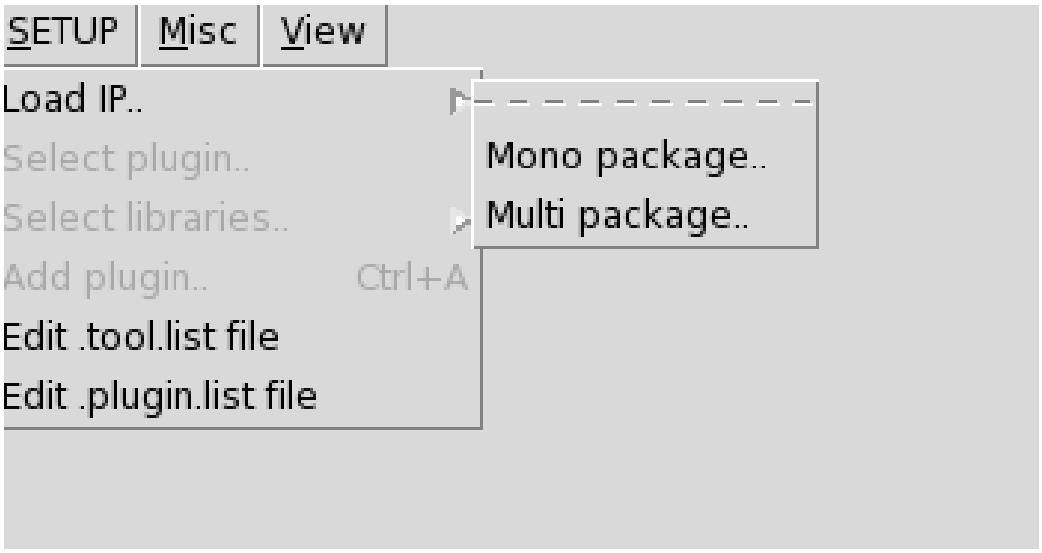


Figure 5.4: Load a Library in IPSCREEN

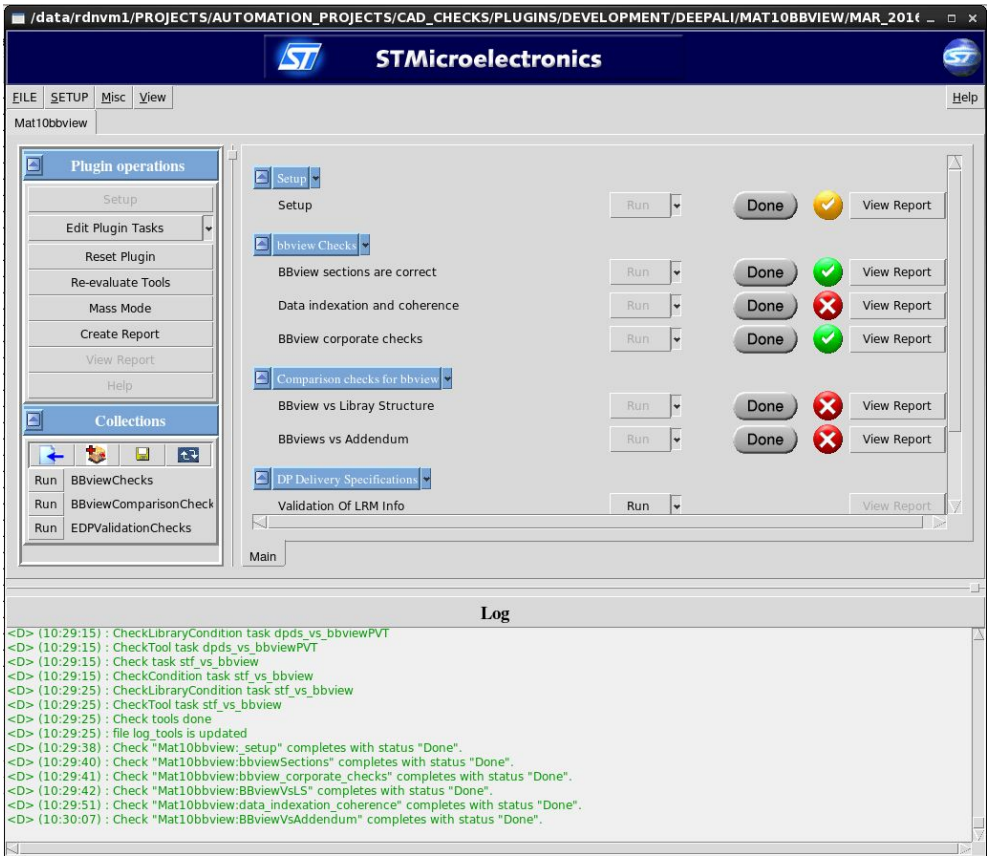


Figure 5.5: Status of Checks

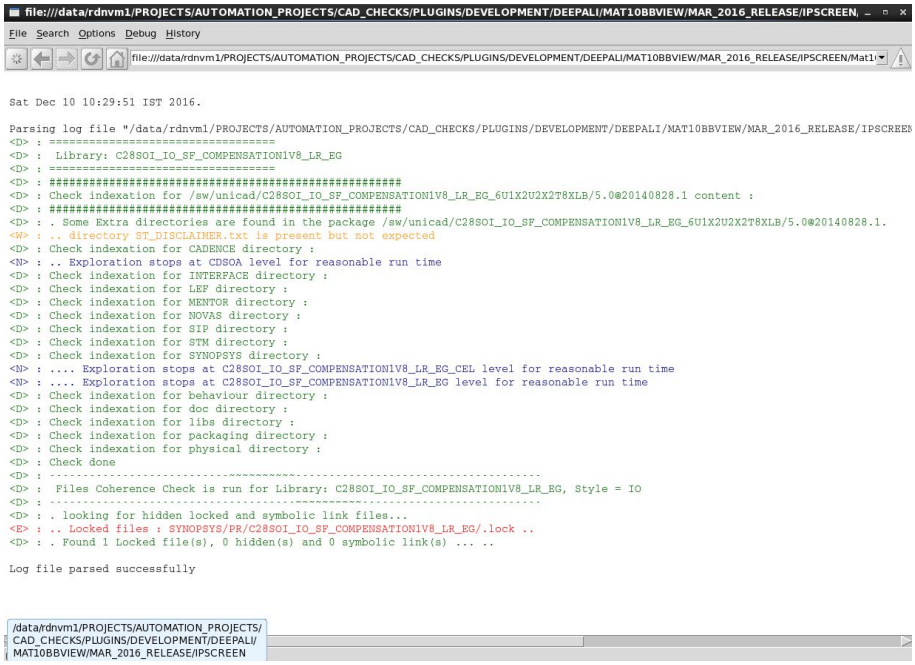


Figure 5.6: Check Report Generated by IPSCREEN

5.3 Limitations of IPSCREEN

- User has to wait for a long time till the library gets completely parsed by the IP-Screen.
- It cannot validate single view or particular cell of a library.
- Output Log was less user friendly.
- It does not support all types of report format (txt/html/xls/csv).

Chapter 6

IPVS

6.1 Introduction

IP CAD Validation is an imperative aspect that recognizes the basic issues before use in the real System on Chip (SoC) Design. This demands the requirement for an IP Validation solution that precisely confirms the correctness and accuracy of the IP design. This is the place the IP Validation Solution comes into the picture.IPVS is a Unified Framework that bundles all checks for a proficient and improved IP Validation.

6.2 Significance

IPVS system gives an extremely effective validation solution for the advanced eras of IPs. When contrasted with the past approval approaches that required mat09 or mat10 package for validation, it furnishes the Designer with the capacity to validate an independent view at any phase of the design cycle, regardless of the library structure or the level of maturity. Using IPVS, packages can be identified at the begining stage as delineated in Figure and therefore diminishing the overall feedback loop cycle.

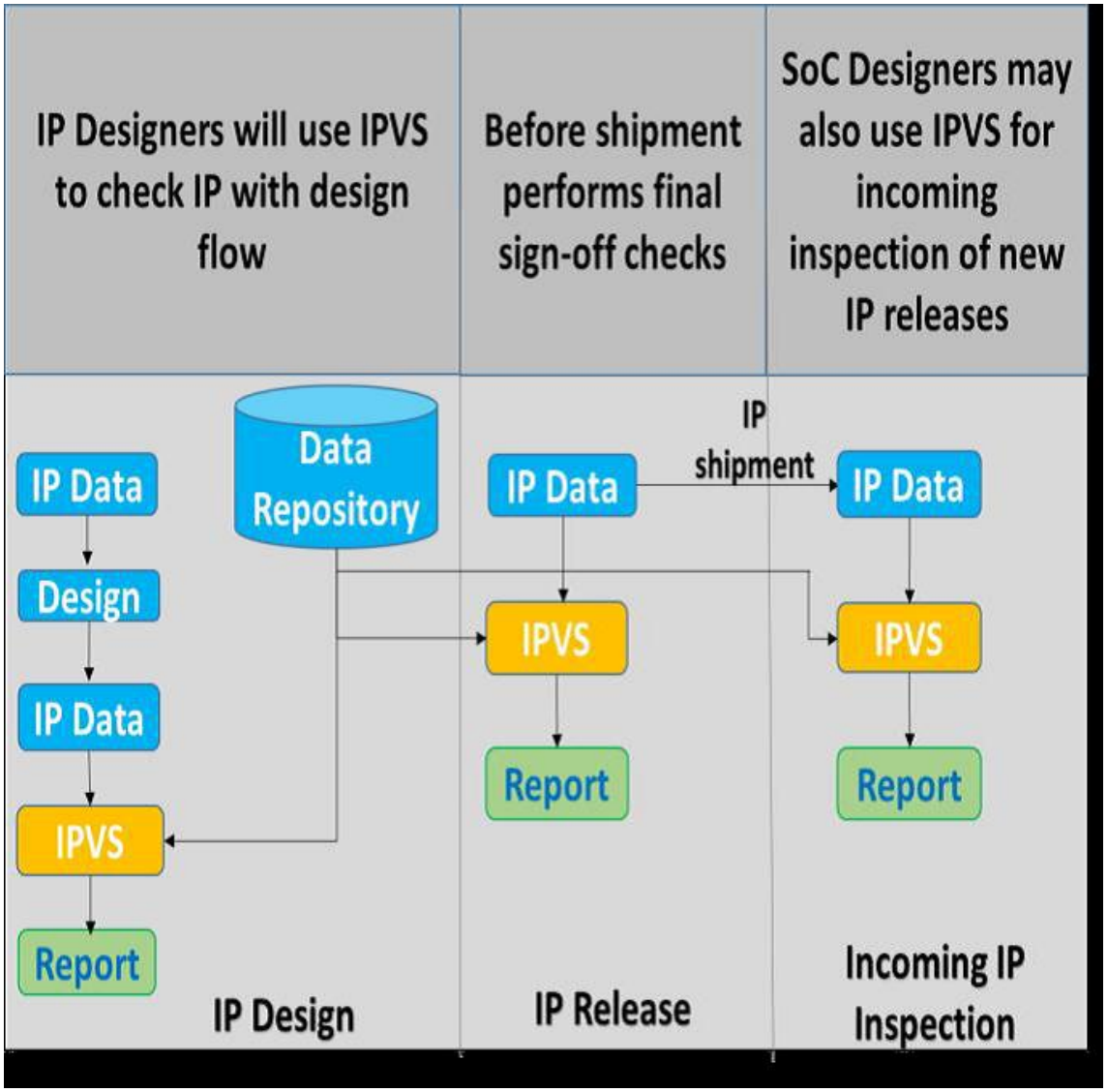


Figure 6.1: IPVS at different stages of IP design flow

- It helps to handle the typical errors like :
  - Pin direction, missing labels, pins not on grid
  - CCS curves have more than one peak or a correction current in the tail
  - Delay decreases with increasing output load, non-paired setup and hold times in Liberty file
  - ECSM curves have large deviations between ECSM and NLDM values etc.
- Checking for these sorts of issue clearly can't be left to visual examination.Using IPVS at each phase of the SoC design flow as delineated in Figure, can check for these errors before they could trigger major revamp.

- It is a unified framework since it unites checks from different tools with a solitary place which gives profitability upgrade, streamlined validation flow along with enlarged check scope.

- Using IPVS, the Designer gets the adaptability to fuse another custom check or to include any new kind of custom view. Besides, any new EDA tool based checks may likewise be added by the Designer to keep running inside a same structure so as to check the compatibility of any IP view with that tool.

- Every existance checks can be configured by the Designer according to the necessity, i.e. the Designer using the IPVS can control the parameters and refine the checks according to the need. For e.g. while checking the consistency of pins crosswise over various formats of an IP, Designer can give a list of power pins, ground pins in a parameters file.

- It helps to keep up the Quality - Run time trade off that can be changed by the Designer as desired. In the event that the Designer needs a fast run time, then either some check parameters can be modified with a specific end goal to validate just a subset of cells inside each view.

- An another favorable point of using IPVS is that setup made once can be reused to validate the view , eventually reducing the time required to validate the CAD views.

6.3 Architecture

The framework includes three layers as portrayed in Figure.

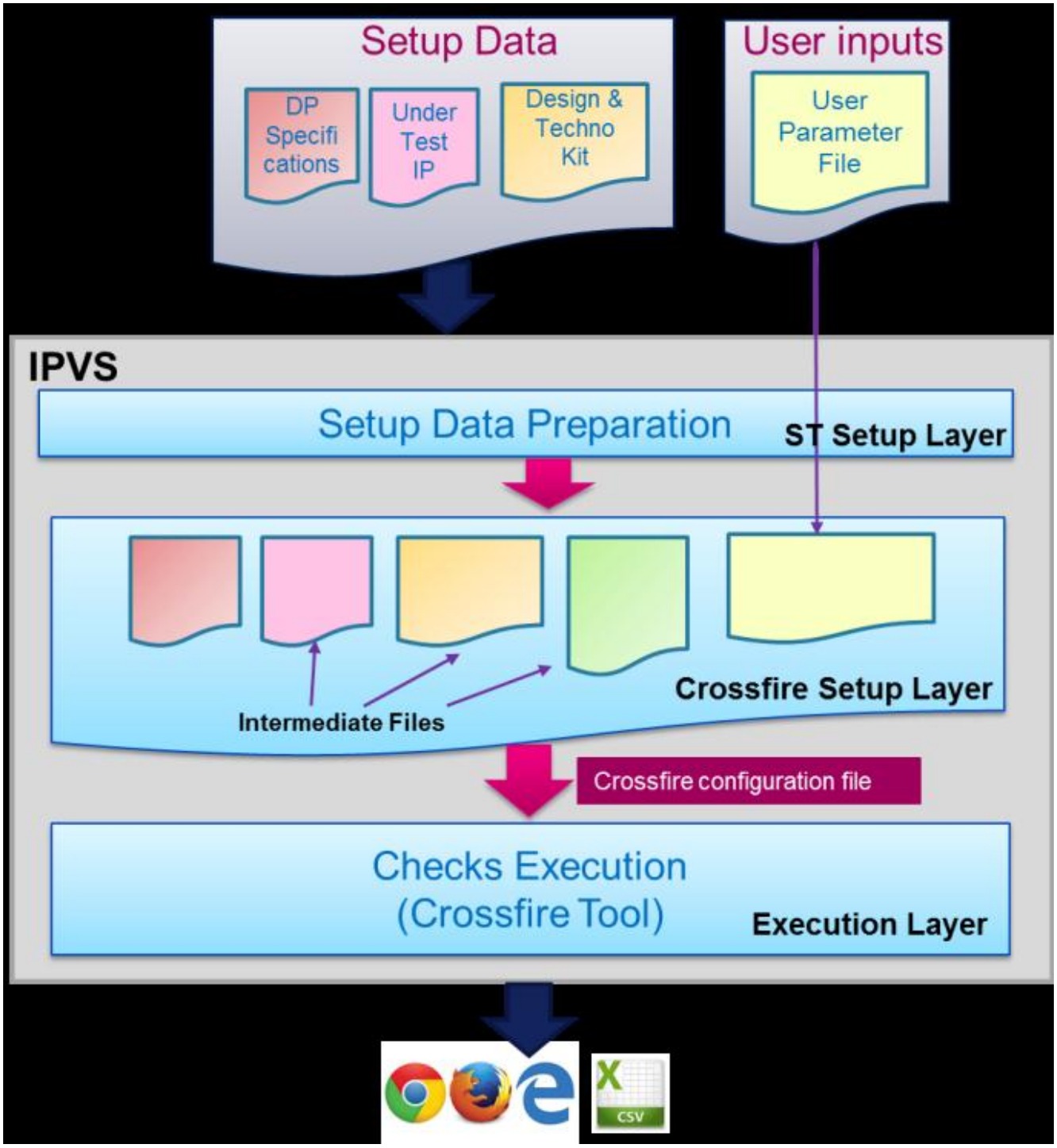


Figure 6.2: IPVS Architecture

- ST Setup Layer – IP Data and ST particular Data from DPDS and Techno Kits is perused and prepared for setup creation.
- Crossfire setup layer - Crossfire specific configuration files are prepared.
- Execution Layer - Checks are executed and the reports are generated.

The general concept residing behind using the layered architecture is to lessen the design intricacy. The layered architecture builds flexibility, modularity, and scalability of the IPVS. The 3 layers converse with each other with pre-characterized interfaces. Each layer can be developed autonomously without affecting alternate layers. Additonally, if we make improvements in any layer without touching the interfaces, the framework still stays useful.

The principal layer, ST Setup layer, primarily involves the Data Preparation stage, in which inputs from various sources like IP Data, Technology and DP Delivery Specifications are handled and associated. The second layer, the Crossfire Setup Layer, is in charge of interpretation of ST particular data to the Cross fire Tool. The third layer is the execution layer of IPVS having the Crossfire as its principle segment. This layer executes the checks and generates the Report using Crossfire Tool.

Chapter 7

Implementation

7.1 Prerequisite

7.1.1 Technical Requirement

Tool	Version
CrossfireControllerKit	1.0
CrossfireConfigurationKit	1.0
Foundation Synopsys Techno Kit	2.9
Foundation Cadence Techno Kit	3.5
Design Kit (DK)	7.0
Milkyway Tool	5.0
Ic Tool	5.0
CADINFRAKIT	1.2-d

Figure 7.1: ST Specific Tools

RAM	**4GB**
OS	**LINUX**
Platform	**LSF**

Figure 7.2: Development Environment

7.1.2 Programming Language

Python is a universally accepted language. It has extensive variety of utilization from Web advancement (Django), logical and numerical processing (Orange) to desktop GUIs (Panda3D). The syntax and structure of the language is perfectly clean and the length of the code is moderately short. It's amusing to work in Python since it enables you to consider the problem without worrying about the syntax.The whole framework is developed in Python Programming language.

7.2 User Interface

1. ***generateConfigTemplate***

   This command is used to generate the user configuration template file (based on IP Type). The user configuration file is required for validateIP command (for checks execution). In this file, user can provide parameters related to tool Setup & Checks.

   This command operates only in BATCH Mode.

   ```
   generateConfigTemplate
    -ipStyle {MACRO} [-filePath <Output filename or directory>]
   ```

   Where,

   -ipStyle: Refers to valid Type of IP.Current release is limited to MACRO only. This is MANDATORY.

   [-filePath]:It is User specific Output filename/directory/filepath.By default it will generate a template file with default name in the Current Working Directory.This is OPTIONAL.

2. ***generateViewFormatMap***

   This command is used to generate the user defined View Format Mapping file. In this file, user can provide deliverable and View of their own choice.Later on, this file is given to validateIP command.

   ```
   generateViewFormatMap [-filePath <Output filename or directory>]
   ```

Where,

[-filePath]:It is User specific Output filename/directory/filepath.By default it will generate a template file with default name in the Current Working Directory. Currently, the default name is "viewFormatMapping.csv". This is OPTIONAL.

3. ***validateIP***

   This is the main command used to execute checks and generate Reports.It operates in two modes: Batch Mode and Graphical User Interface.

```
validateIP -libraryName <library name value> -libraryPath <Library path
value> -ipStyle {MACRO} -config <user parameter config file path> [-
gui] [ -viewFormatMap <view format mapping file path> ]
```

   Where,

   -libraryName: Under Test library name.Library name should be as per the vc.bbview file.This is MANDATORY.

   -libraryPath: Under test library path till packaging directory. This is MANDATORY.

   -ipStyle:Type of IP.Current release is limited to MACRO only.This is MANDATORY.

   -config:User specified checks and parameters configuration file. This is MANDATORY.

   [-gui]: Used for GUI mode; where after setup, GUI is displayed before checks execution. In case, it's not specified, the setup is generated and checks are executed in batch mode and then final reports are displayed. This is OPTIONAL.

   [-viewFormatMap]:User defined viewFormat mapping file.This is OPTIONAL.

4. ***diagnoseIP***

   This command is used to analyze/debug the IP on which checks have already been executed.

```
diagnoseIP
[-db <Path of Crossfire Database>] [-setup <libraryName>.cfg file]
```

Where,

[-db]:The Path of database generated as an outcome of validateIP command. (By default, it's picked from current directory).This is OPTIONAL.

[-setup]: The path of IP setup Configuration file (.cfg file). In case, it's not provided.This is OPTIONAL.

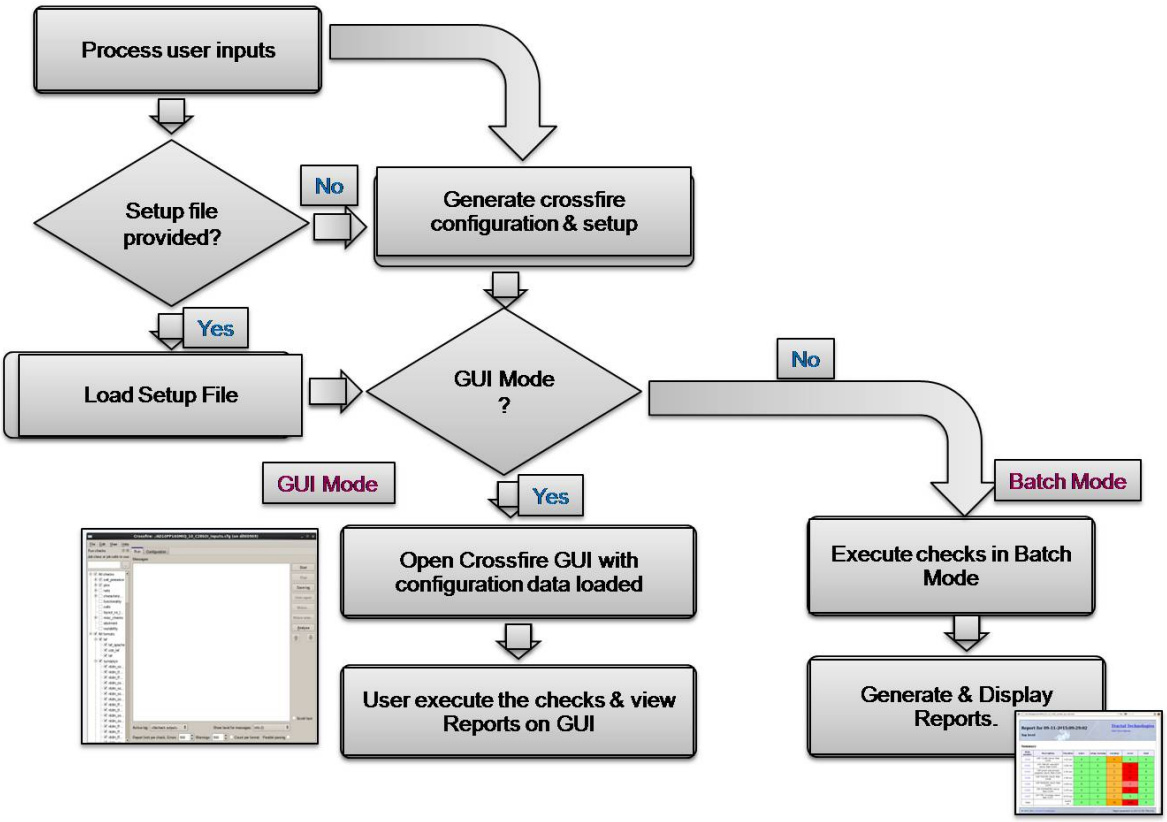


Figure 7.3: IPVS Flow Diagram

7.3 RUN Area Directory Structure

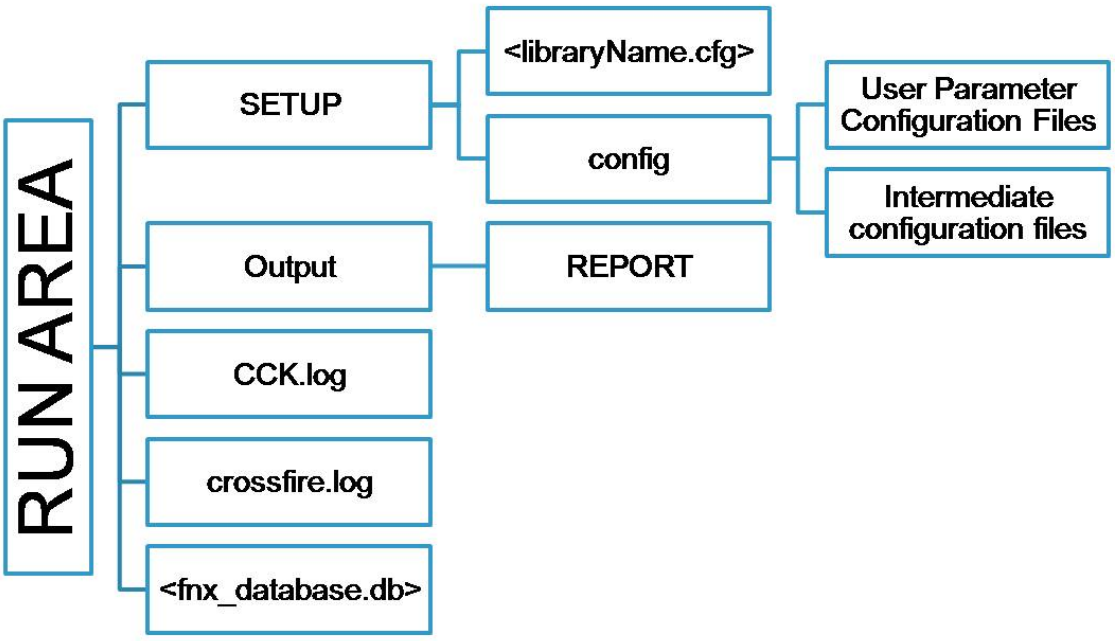


Figure 7.4: IPVS Flow Diagram

1. ***setup :*** This directory contains the setup files including all the configuration files in the config subdirectory.

   (a) ***config :***This subdirectory contains all the intermediate configuration files used for generating the global configuration file.

   (b) ***libraryName.cfg :***This is the consolidated setup file used by the Crossfire Tool to execute checks and generate Reports.

2. ***CCK.log:***It is a log file generated by Crossfire Controller Kit in the current working directory while executing any of the commands available (i.e. generateConfigTemplate, validateIP and diagnoseIP). It does not include any information related to the checks executed.

3. ***crossfire.log:***This is the log file generated by the Crossfire Tool.

4. ***fnx-database.db:*** It is a binary database file containing data related to all the checks executed by the Crossfire tool. It is the required input file to diagnoseIP command mainly used to analyze the checks and generate HTML report.

Chapter 8

Performance Evaulation of IPVS

8.1 Command Line Flow

```
dlhl2099{FRACTAL}>> gui generateConfigTemplate -ipStyle MACRO
```

Figure 8.1: IPVS command to generate Configuration Template

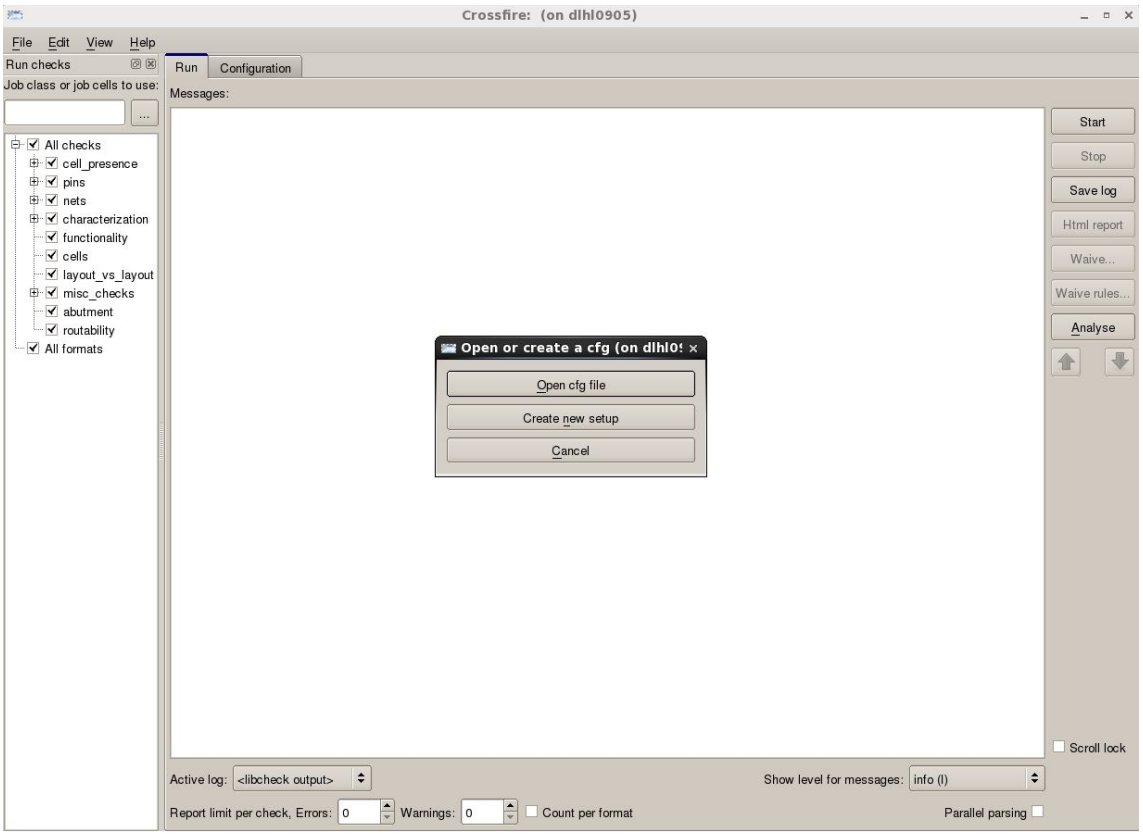


Figure 8.2: Launching of Tool

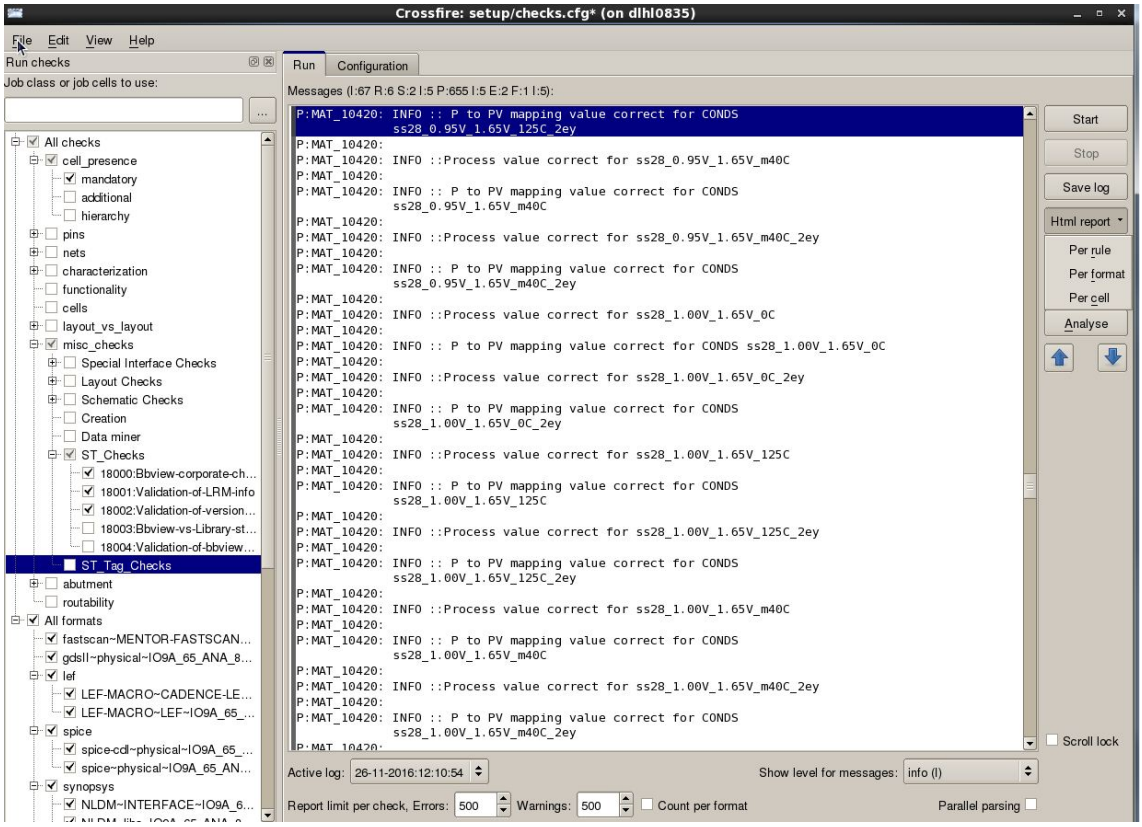


Figure 8.3: Check Execution

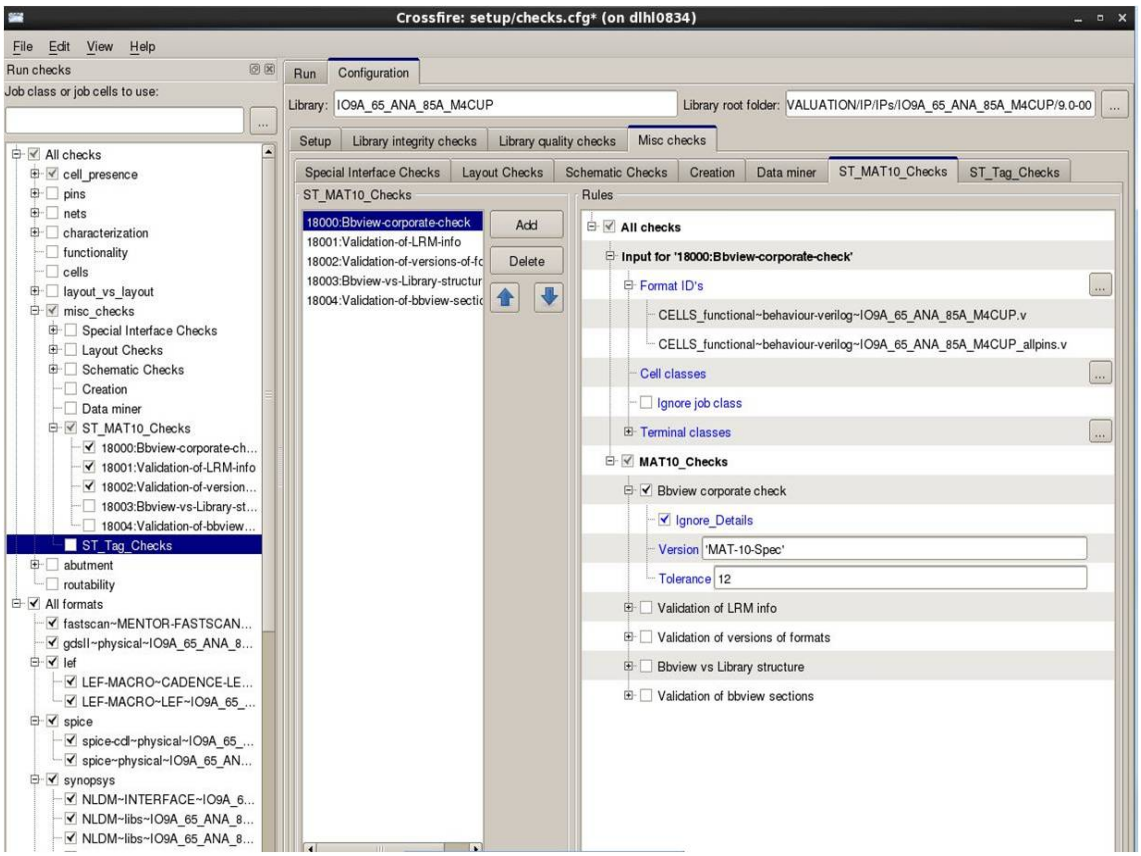


Figure 8.4: Various Checks & Parameters

```
dlhl2099{FRACTAL}>> gui diagnoseIP
```

Figure 8.5: IPVS command to Analyze report

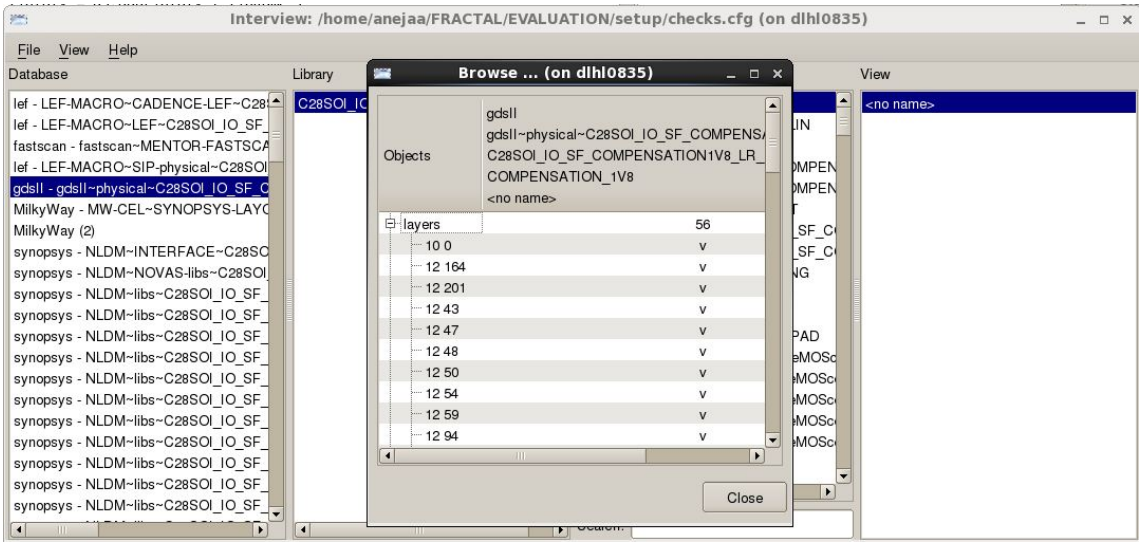


Figure 8.6: IPVS Interview

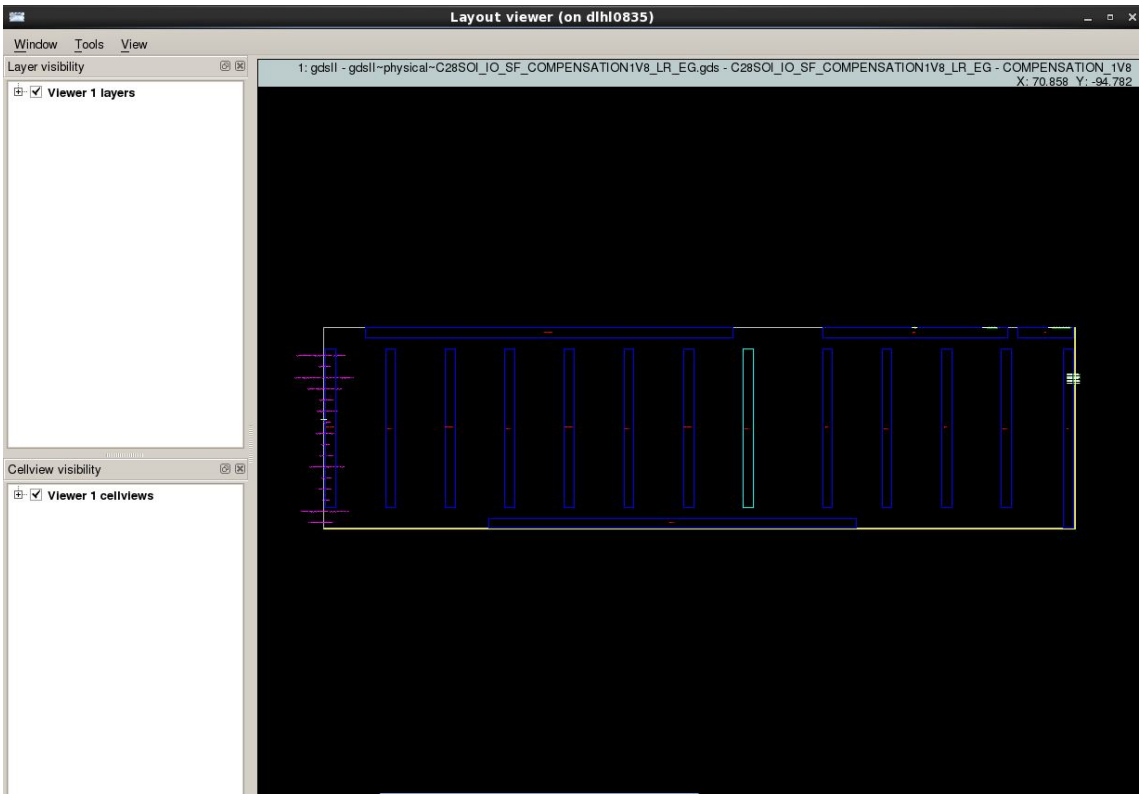


Figure 8.7: GDS View

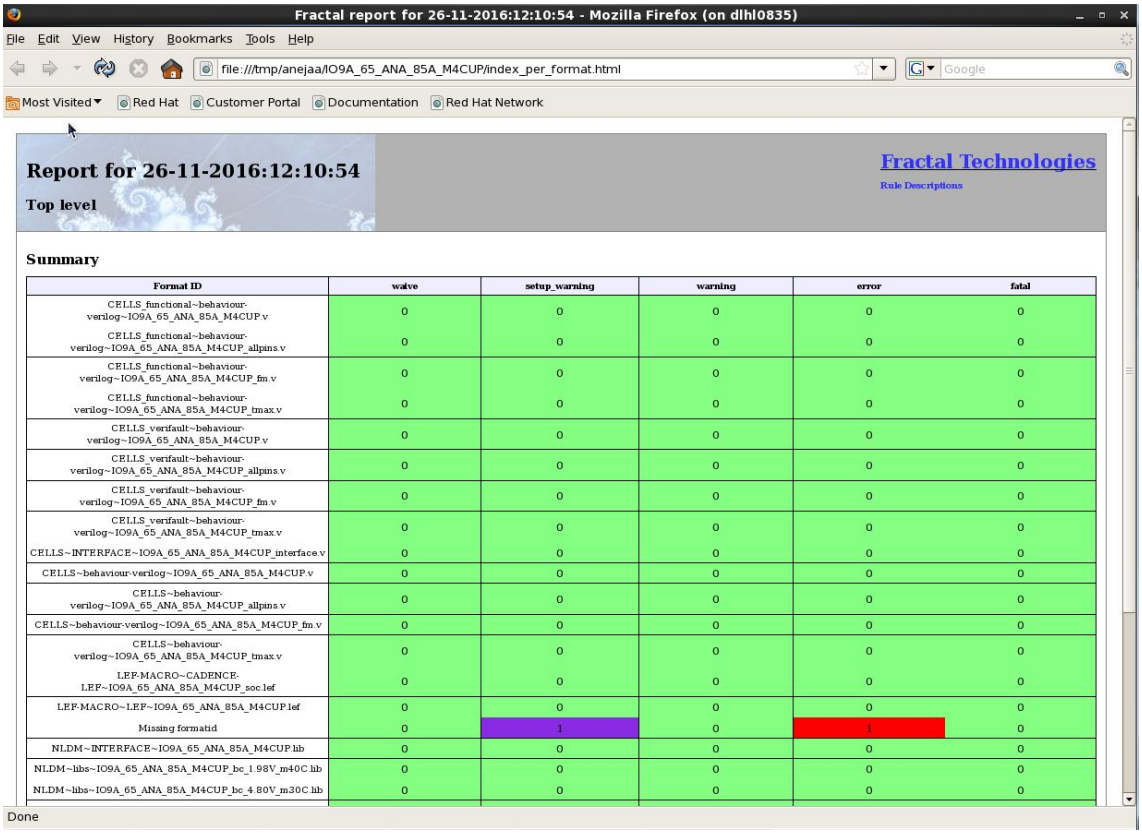
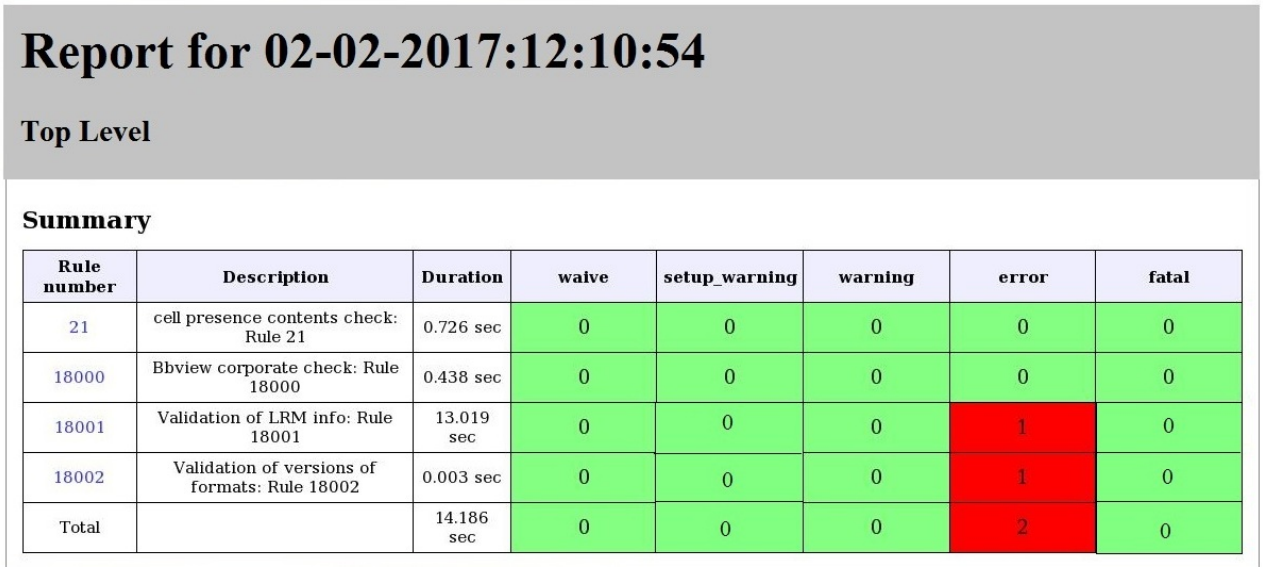


Figure 8.8: Html Report per format

8.2 Experiments

8.2.1 28nm Technology



Report for 02-02-2017:12:10:54

Top Level

Summary

Rule number	Description	Duration	waive	setup_warning	warning	error	fatal
21	cell presence contents check: Rule 21	0.726 sec	0	0	0	0	0
18000	Bbview corporate check: Rule 18000	0.438 sec	0	0	0	0	0
18001	Validation of LRM info: Rule 18001	13.019 sec	0	0	0	1	0
18002	Validation of versions of formats: Rule 18002	0.003 sec	0	0	0	1	0
Total		14.186 sec	0	0	0	2	0

Figure 8.9: Report for 28nm Library

8.2.2 65nm Technology

Report for 08-03-2017:10:07:22

Top Level

Summary

Rule number	Description	Duration	waive	setup_warning	warning	error	fatal
21	cell presence contents check: Rule 21	2.726 sec	0	0	0	0	0
18000	Bbview corporate check: Rule 18000	0.777 sec	0	0	0	0	0
18001	Validation of LRM info: Rule 18001	15.91 sec	0	0	0	0	0
18002	Validation of versions of formats: Rule 18002	0.009 sec	0	0	0	0	0
Total		19.422 sec	0	0	0	0	0

Figure 8.10: Report for 65nm Library

8.2.3 BCD Technology

Report for 04-04-2017:11:22:12

Top Level

Summary

Rule number	Description	Duration	waive	setup_warning	warning	error	fatal
21	cell presence contents check: Rule 21	0.415 sec	0	0	0	0	0
18000	Bbview corporate check: Rule 18000	0.888 sec	0	0	0	0	0
18001	Validation of LRM info: Rule 18001	10.21 sec	0	0	0	1	0
18002	Validation of versions of formats: Rule 18002	0.009 sec	0	0	0	1	0
Total		11.522 sec	0	0	0	2	0

Figure 8.11: Report for BCD Library

8.3 Comparative Analysis

8.3.1 Functional Analysis

Sr No	Points	IPScreen (Hybrid)	IPVS (Hybrid)
1	Ease of Use(in terms of Input)	Low	High
2	Plugin Loading	Needed	Not needed
3	Reporting	Less User Friendly	More User Friendly
4	Single View or Single Cell	Not Available	Available
5	Automatic –Setup Generation	Not Available	Available

Figure 8.12: IPScreen vs IPVS Functionality

8.3.2 Timing Analysis

Sr No	Library	IPScreen (Hybrid)	IPVS (Hybrid)
1	28nm	2.002 min	0.438 sec
2	65nm	3.517 min	0.777 sec
3	BCD	1.056 min	0.888 sec

Figure 8.13: IPScreen vs IPVS Timing for Bbview Corporate Check

Sr No	Library	IPScreen (Hybrid)	IPVS (Hybrid)
1	28nm	5.07 min	13.019 sec
2	65nm	5.157 min	15.91 sec
3	BCD	4.123 min	10.21 sec

Figure 8.14: IPScreen vs IPVS Timing for Validation of LRM Check

Sr No	Library	IPScreen (Hybrid)	IPVS (Hybrid)
1	28nm	2.457 min	0.003 sec
2	65nm	1.457 min	0.009 sec
3	BCD	2.113 min	0.009 sec

Figure 8.15: IPScreen vs IPVS Timing for Validation of Version of formats

Chapter 9

Observations / Findings

- Previously a lot of intermediate files were used for the processing of checks.
- In the new framework, the user gets a lot of flexibility.User can select a single view for validation.
- Using the new Framework, user has the freedom to generate different types of reports i.e. check-wise, view-wise or cell-wise.
- User gets a GUI in order to highly configure the setup given for the validation of a particular IP.
- The runtime of the checks use for the validation of IP inside the new tool depends on the configuration of the setup.

Chapter 10

Conclusion & Future Scope

10.1 Conclusion

IPVS is a framework catering to all IP types and technologies. It provides improved CAD quality which can help to detect the issues and failures at initial stages of IP development, thus reducing the cost and effort of failure at SoC design level. This framework provides a unified platform where Designer can validate the IP and check its compliance against the requirement specification using a complete coverage of checks. Library and IP Quality Assurance (QA) checks can be run both in an Interactive and Batch modes to ensure the highest design quality, and shortest time to market with IPVS.

10.2 Future Scope

This framework is extensible to handle multiple IPs simultaneously and classification checks as per taxonomy. This framework can be further scaled to provide Incremental handling of failed checks. So that Designer can re-run only the failed checks with a single command.

Bibliography

[1] STMicroElectronics Internal Document , *Library Views*

[2] https://en.wikipedia.org/wiki/Python , *Python Learning*

[3] STMicroElectronics Internal Document , *IPScreen*

[4] https://en.wikipedia.org/wiki/Application-specific$_i ntegrated_c ircuit$, *ASIC Flow*

[5] W. Agatstein, K. McFaul, and P. Themins, *Validating an asic standard cell library*

[6] K. Shuler, "What does it cost you when your SoC is late to market?" 2014. *[Online]. Available: http://www.arteris.com/blog/bid/112221/What-Does-It-Cost-You-When-Your-SoC-is-Late-to-Market.*

[7] Mohit Bhasin, Lipika Parwani, *"To develop a method for validating an IP without running a full RTL to GDS flow ", ST Internal Conference.*

[8] Jung Kyu Chae, *Specification Platform for Library IP Development. Databases [cs.DB]. Universite Pierre et Marie Curie - Paris VI, 2014. English.*

**

Appendix A

Configuration Generation Flow for LayerMap

1. Inputs
    - Cadence Mapout
    - DK Layer Map
    - DK Object Map
    - Mapout file
    - Techno file
2. Output
    - Layer Map configuration file- "LayerMap"(CFT format)
3. Flow
    - Using Opensetup, load CadenceTechnoKit, DesignKit and SynopsysTechnoKit
    - Get Below Paths from OpenSetup APIs
        (a) Cadence MapOut File
        (b) DK Layer MapFile
        (c) DK Object Map File
        (d) MapOutFile
        (e) Synopsys TechnoFile

- Invoke crossfire command to generate Crossfire specific “maplayout” by passing Milkyway db path.a. This “maplayout” contains information of Milky Way layer number and its corresponding Layer name.
- Invoke LayerMap generation script from Crossfire to dump the LayerMap configuration (layerMap.cft) file.

Appendix B

Configuration Generation Flow for Technology

1. Inputs
    - Technology.lef (from Cadence Technology Kit)
    - Technology.tf (From Synopsys Technology Kit)
2. Output
    - Technology configuration file –"technoConfig" (CFT format)
3. Flow
    - Import lefKit and tfKit from COREKIT (Under CADINFRAKIT) in order to use lefParser and tfParser.
    - Open and Read technology.lef. Extract the names of Routing Layers.
    - Open and Read technology.tf .Fetch METAL STACK Information.Extract Layer name and respective mask name from Layer Section from technology.tf. Create a MAP – MaskValue: LayerName [e.g. metal4 : M4].
    - Check the compatibility of technology files in the two different technology Kits. Check symmetric difference of lefLayerNames and tfLayerNames.If any difference found,Print the Extra layer and Exit.If not found,Print .lef and .tf are compatible and continue.

- Extract Parameters from Cadence technology.lef.Check the existence of “MANUFACTURINGGRID”. If exist Store the Value Else ”Manufacturing Grid Not Found”.Store the value of “DATABASE MICRON” and print the value in log. Print the Metal Stack Mapping.
- Extract TechnoConfig Parameters.
- Write the above stored information (technoData dictionary) into YAML file.